UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Modelo para el desarrollo de servicios en línea utilizando Tarjetas Inteligentes

Tesis presentada en opción al título de Máster en Informática Aplicada

Maestrante: Ing. Ander Sánchez Jardines

Tutores: Dr.C. Yusnier Valle Martínez

Msc. Adonis Cesar Legón Campos

Ciudad de La Habana, Mayo2014

Declaración Jurada de Autoría

DECLARACIÓN JURADA DE AUTORÍA

Declaro por este medio que yo Ander Sánchez Jardines, con carné de identidad 87030935363, soy el autor principal del trabajo final de maestría “Modelo para el desarrollo de servicios en línea utilizando Tarjetas Inteligentes”, desarrollada como parte de la Maestría en Informática aplicada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de La Habana a los < > días del mes de Julio del año 2014.

_____

Ander Sánchez Jardines

Autor

_____

Dr.C. Yusnier Valle Martínez

Tutor

_____

Msc. Adonis Cesar Legón Campos

Tutor

Resumen

RESUMEN

La mayoría de las aplicaciones que ofrecen servicios mediante el uso de Tarjetas Inteligentes, requieren la instalación en una computadora de determinados paquetes de software, comúnmente denominados *middleware*, que posibilitan la interacción entre dichas aplicaciones y la tarjeta. La falta de una tecnología que permita la comunicación en línea con las Tarjetas Inteligentes y que además, sea soportada por diferentes navegadores *web* y sistemas operativos, ha imposibilitado un aprovechamiento óptimo de las ventajas de estos dispositivos inteligentes de identificación, en la prestación de múltiples servicios a través de Internet. A partir de la experiencia adquirida en el área de las Tarjetas Inteligentes, se ha identificado la necesidad de la definición de un modelo para el desarrollo de servicios que utlicen Tarjetas Inteligentes, que posibilite la ubicación de los mismos en el servidor, evitando al usuario la instalación de determinado *software* en su PC y la actualización periódica del mismo, además de brindar una mayor seguridad en el almacenamiento de las llaves simétricas utilizadas en la comunicación segura entre la tarjeta y el terminal.

El presente trabajo define un modelo que guía el desarrollo de aplicaciones multiplataforma para la interacción con tarjetas a través de navegadores *web*, especificando las categorías y relaciones asociadas. Finalmente la aplicación del modelo sirvió como marco de referencia y guía para el desarrollo de una plataforma que permite la comunicación en línea con Tarjetas Inteligentes, de la cual se describen las herramientas, tecnologías utilizadas y los principales artefactos generados en el proceso de desarrollo.

Palabras clave: Tarjetas Inteligentes, *Middleware*, Servicios en línea, Navegadores *web*, Canal seguro.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La historia de Internet se remonta al temprano desarrollo de las redes de comunicación en la segunda mitad del pasado siglo. Su surgimiento potenció el crecimiento vertiginoso de muchas ramas de la ciencia y la tecnología, posibilitó un acceso más fácil a la información, fomentando la creación de foros virtuales para compartir conocimientos y generar debates científicos, sociales o culturales. Con la aparición de internet muchos servicios comenzaron a prestarse a través de la red sin importar si el proveedor y el beneficiario se encontraban separados por miles de kilómetros. En la actualidad la mayoría de los procesos productivos y tecnológicos de la sociedad están sustentados en la red de redes.

La conexión a Internet es un mecanismo de enlace con que una computadora o red de computadoras cuenta para conectarse a Internet, permitiendo visualizar al usuario las páginas *web* desde un navegador. Los navegadores *web* son programas que realizan la interpretación del código fuente de las páginas *web* permitiendo la visualización de la información contenida en estas, existen varios de ellos: Internet Explorer, Mozilla Firefox, Opera, Chrome, Safari y otros, siendo los dos primeros los más utilizados (Statcounter, 2013).

Como consecuencia del crecimiento de las Tecnologías de la Información y las comunicaciones (TIC), además, el gran número de usuarios que acceden a este servicio, la seguridad informática se ha convertido en un factor importante para salvaguardar la soberanía de las redes. Muchos sistemas garantizan su seguridad mediante el conocido mecanismo basado en nombres de usuarios y contraseñas, el cual implica algunos riesgos relacionados con la protección de la palabra clave. Según (Márquez, 2006) es típico que los usuarios con poca experiencia utilicen contraseñas cortas e inseguras y las almacenen en los historiales de la computadora, facilitando de esta manera su obtención por parte de atacantes.

La *web* ofrece diferentes servicios en línea como son: las transacciones bancarias, el correo electrónico, las reservaciones hoteleras, la certificación o la tramitación de documentos oficiales. Todas estas prestaciones presentan un factor común y es el riesgo de ejecutarse a través de Internet, ya que sin la presencia física del solicitante es una preocupación lógica el querer estar seguro de la identidad del usuario y no alguien intentando suplantarla (UVA, 2012).

Actualmente se registra una tendencia al uso de dispositivos inteligentes que facilitan y aseguran el proceso de identificación. Se han desarrollado tecnologías como las Tarjetas Inteligentes para garantizar una verificación más confiable de la identidad del usuario que solicita determinado servicio.

Las Tarjetas Inteligentes o *SmartCard* son una lámina plástica de tamaño similar a las tarjetas bancarias con barra magnética, con la peculiaridad de traer incorporado un circuito integrado que permite almacenar información de forma segura e incluso la ejecución de lógica programada (Markantonakis, 2004). Contienen componentes de memoria volátil y no volátil que unidos a un microprocesador y un sistema operativo permite almacenar, cifrar y modificar información. Tienen un gran número de aplicaciones, entre las más significativas está la firma digital de documentos, que asegura la autenticidad del emisor del mensaje pues la clave privada asociada al certificado digital[1] nunca sale de la tarjeta; también se utilizan en control de acceso, como monedero electrónico, en el pago de televisión, telefonía móvil, transporte público, para acceso a sitios *web*, sistemas nacionales de salud, servicios bancarios, licencias de conducción entre otros. Además, en varios países del mundo se usan como documento oficial de identificación, entre los que están España y Finlandia.

El manejo masivo de información sensible, ha permitido el desarrollo y una mejora sustancial en las aplicaciones y protocolos de seguridad de la Tarjeta Inteligente. En un contexto organizacional estas permiten: acceso seguro y autenticado de usuarios a sistemas o redes. Existen disimiles sistemas informáticos proveedores de servicios que usan Tarjetas Inteligentes, pero en su gran mayoría están orientados a brindar un servicio determinado y para aumentar su escalabilidad, solo los desarrolladores implicados en su implementación poseen los conocimientos arquitectónicos de la aplicación para que esta reaccione y se adapte sin perder calidad ante nuevas incorporaciones de funcionalidades o requisitos.

Encaminadas al desarrollo de las TIC, en el año 2008 se crea la Universidad de las Ciencias Informáticas. Entre los centros de desarrollo que la componen se encuentra el Centro de Identificación y Seguridad Digital el cual está compuesto, entre otras, por una línea de investigación y desarrollo dirigida a impulsar la creación de servicios utilizando Tarjetas Inteligentes. Luego de varios resultados significativos, surgió en este departamento una cartera de soluciones específicas, cada una con alto grado de

[1] Es un documento digital que permite identificar como válido a un emisor de información. Los certificados digitales son emitidos por autoridades certificadoras, las que tienen como misión verificar que la compañía en cuestión sea confiable y se encuentre legalmente constituida.

independencia, desarrolladas en diferentes lenguajes y utilizando diversos mecanismos de seguridad con el objetivo de insertar los productos desarrollados utilizando este tipo de dispositivo en el mercado del *software* internacional.

El uso de Tarjetas Inteligentes para los servicios en línea potencia además, la implementación de la infraestructura PKI[2] (Carlisle Adams, 2002) por sus siglas en inglés, *Public Key Infrastructure* para la certificación de documentos y la firma digital.

La manera acostumbrada, de interactuar con las Tarjetas Inteligentes es a través de capas o librerías de *software*, denominadas *middleware*[3] (Cardlogix, 2012), que se ejecutan en el cliente y hacen función de comunicadores intermediarios entre diversas aplicaciones y los lectores de tarjetas (UAB, 2013). El manejo de las tarjetas mediante el uso de *middleware*s que se ejecuten en el cliente trae consigo algunas desventajas o riesgos considerables, para las actualizaciones del *software* o la incorporación de nuevas funcionalidades a estas librerías habría que distribuirlas por todos los clientes de un sistema dado o publicarlos en un sitio *web* para que sean descargados a través de la red. Esto, además, de ser incómodo para el cliente implica que los usuarios que pretenden interactuar con sus tarjetas tengan ciertos conocimientos para efectuar las actualizaciones y posean una serie de permisos en el manejo de los recursos de la computadora para poder instalarlas. Otra de las principales desventajas es la que está relacionada con la seguridad, pues las llaves simétricas necesarias para la comunicación segura con la tarjeta, permanecen en el cliente con peligro que puedan ser obtenidas.

Derivado de la situación anteriormente expuesta se encuentra el siguiente **Problema de investigación**:

¿Cómo aumentar la seguridad y la escalabilidad en el desarrollo de servicios en línea que utilicen Tarjetas Inteligentes?

Como **objeto de estudio** se tiene: El proceso de desarrollo de aplicaciones que utilizan Tarjetas Inteligentes.

Para dar solución al problema existente se ha tomado como **objetivo general**:

Definir un modelo para el desarrollo de servicios en línea utilizando Tarjetas Inteligentes, que aumente la seguridad garantizando el resguardo de las llaves simétricas y la escalabilidad, con la incorporación de nuevas aplicaciones.

[2] Es una infraestructura de clave pública que se compone de una combinación de *hardware* y *software*, políticas y procedimientos que permiten la ejecución con garantías de operaciones criptográficas como el cifrado, la firma digital y el no repudio de transacciones electrónicas.

[3] *Middleware* es capa mediadora entre la pc y la tarjeta

Derivándose en los siguientes **objetivos específicos**:

- Establecer fundamentos teóricos y metodológicos relacionados con las soluciones existentes para el desarrollo de los servicios utilizando Tarjetas Inteligentes.
- Definir el esquema general de la metodología de desarrollo de los servicios en línea utilizando Tarjetas Inteligentes.
- Seleccionar estándares, tecnologías y mecanismos de seguridad que formarán parte de la metodología de referencia.
- Desarrollar los componentes de una plataforma de *software* que implemente la metodología de referencia propuesta.
- Validar a través del desarrollo de un prototipo de servicio que cumpla con la seguridad de la plataforma y su incorporación como nueva aplicación.

Hipótesis

Con una correcta aplicación del modelo para el desarrollo de servicios en línea utilizando Tarjetas Inteligentes, se aumenta la seguridad, garantizando el resguardo de las llaves simétricas y la escalabilidad con la incorporación de nuevas aplicaciones.

Aporte y Novedad de la investigación:

Con el resultado de esta investigación se obtendrá un modelo para el desarrollo de servicios en línea utilizando Tarjetas Inteligentes. Este modelo pretende proporcionar a los usuarios que lo utilicen, un aumento significativo de la seguridad de los servicios, en un ambiente donde los *middleware* están centralizados.

De manera práctica, esta investigación aportará a los desarrolladores de aplicaciones una guía probada, utilizada para el desarrollo de este tipo de *software* y una manera de conmutar diferentes servicios en un mismo dispositivo, sin que esto constituya, un riesgo potencial para la seguridad de las llaves simétricas que rigen los protocolos de comunicación.

Como aporte práctico además, se proveerá la implementación de una plataforma de *software* que automatice el modelo propuesto y permita integrar nuevos servicios, facilitando la actualización o distribución de nuevas funcionalidades de los *middlewares* que están contenidos en el servidor, sin que el usuario se vea afectado en el proceso.

Permitirá al desarrollador que implemente el modelo una fácil interacción con la tarjeta sin importar el sistema operativo y otros requerimientos de *software*, garantizando una

mayor seguridad de las llaves criptográficas necesarias para establecer la comunicación segura entre el *middleware* y la Tarjeta Inteligente.

La investigación estará sustentada en los siguientes **métodos científicos**:

Teórico

El empleo del método **histórico-lógico** posibilitará la compresión lógica del objeto de estudio haciendo un análisis riguroso de sus antecedentes y el proceso evolutivo por el cual ha transitado todas las tecnologías utilizadas para establecer la comunicación entre las Tarjetas Inteligentes y las aplicaciones proveedoras de servicios.

Analítico-sintético permitirá la consulta de diversas fuentes bibliográficas y la extracción de los elementos más importantes que se relacionan con el objeto de estudio.

Inductivo-deductivo será valioso para poder realizar generalizaciones del conocimiento a partir del estudio de situaciones concretas y viceversa.

La abstracción de diversas situaciones y la representación de sus características fundamentales desde determinada perspectiva serán de gran importancia en el desarrollo de esta investigación, por lo que se usará el método **modelación**, muestra de ello serán todos los diagramas y modelos que se producirán a lo largo de la misma.

Empírico

Observación: Consiste en la percepción planificada dirigida a un fin y relativamente prolongada de un hecho o fenómeno. Este método permitirá percibir directamente cómo se desarrollan las aplicaciones y servicios de Tarjetas Inteligentes.

Estructura del documento: el documento se encuentra dividido en tres capítulos.

En el Capítulo 1 se introducen conceptos relacionados con las Tarjetas Inteligentes y sus características. Se describen los sistemas operativos libres y estándares con más relevancia en el desarrollo de aplicaciones de este tipo, se hace un estudio de las soluciones existentes haciéndose una valoración crítica de sus ventajas y desventajas.

En el Capítulo 2 se hace la propuesta de un modelo para el desarrollo de servicios en línea utilizando Tarjetas Inteligentes. Se realiza la conceptualización del modelo a partir de tecnologías y estándares existentes en el desarrollo de aplicaciones para Tarjetas Inteligentes. Se describen los principios por los cuales el modelo se rige, sus componentes principales y su comunicación, los mecanismos de seguridad que se proponen para garantizar un ambiente seguro.

En el Capítulo 3 se presenta una plataforma de *software* que implementa el modelo para el desarrollo de servicios en línea utilizando Tarjetas Inteligentes. Se describen las

tecnologías, *frameworks* y herramientas utilizadas para la construcción de la plataforma en sus diferentes ambientes. Se exponen también los aspectos fundamentales de los requisitos básicos que deben ser cumplidos por la plataforma, la implementación y funcionamiento de la misma y por último la validación con la implementación de un servicio que se incorpora a la plataforma.

Capítulo 1: Marco teórico

1. MARCO TEÓRICO

Con el objetivo de facilitar la comprensión del alcance de la investigación, en el presente capítulo se exponen conceptos asociados al dominio del problema planteado, para ello se analizarán las características y aplicaciones de las mismas, así como los estándares internacionales y las tecnologías más utilizadas. Se abordarán también los modelos de desarrollos más significativos y los diferentes enfoques existentes en cuanto al proceso de producción de *software* con Tarjetas Inteligentes. Por último, se analizarán las soluciones factibles existentes y en base a su estudio se arribará a una conclusión crítica.

1.1. INTRODUCCIÓN A LAS TARJETAS INTELIGENTES

1.1.1 TARJETAS INTELIGENTES

Las Tarjetas Inteligentes fueron inventadas y patentadas en la década del setenta del siglo pasado, respecto a ello, existen algunas discusiones de quién es el inventor original, entre los que se encuentran: Juergen Dethloff de Alemania, Arimura de Japón y Roland Moreno de Francia. Inicialmente se utilizaron para el pago telefónico público, pero con el avance en sus capacidades técnicas y campo de aplicación han tomado gran auge principalmente en la telefonía móvil, con la introducción de las tarjetas SIM[4]. También se han utilizado como identificación personal, pasaportes, licencia de conducción y tarjetas de seguros. (Cardwerk, 2011)

Una Tarjeta Inteligente es del mismo tamaño que una tarjeta de crédito, que almacena y procesa información a través de circuitos electrónicos incrustados en el plástico de la tarjeta. Estas tarjetas son portables, resistentes y tienen poder de procesamiento e información. Las Tarjetas Inteligentes según su interfaz se pueden clasificar en dos tipos: Tarjetas Inteligentes sin contacto y Tarjetas Inteligentes con contacto. (Effing, 2003), (Ortega, 2008)

1.1.2 TARJETAS INTELIGENTES SIN CONTACTO

ISO/IEC 14443 es un estándar internacional que define a las Tarjetas Inteligentes sin contacto y sus protocolos de transmisión, el estándar cuenta con 4 partes:

ISO 14443-1: características físicas, define las características físicas de la tarjeta sin contacto.

[4] Acrónimo en inglés de *subscriber identity module*, en español módulo de identificación de abonado. Es una Tarjeta Inteligente utilizada en teléfonos móviles.

ISO 14443-2: Interfaz de la energía y de la señal de radiofrecuencia, Las tarjetas pueden ser de tipo A y tipo B, los cuales se comunican por radiofrecuencia a 13,56 MHz. Las principales diferencias entre estos tipos de métodos de modulación, se refieren a los sistemas de codificación y procedimientos de inicialización del protocolo. Tanto el tipo A como las de tipo B utilizan el mismo protocolo de transmisión.

ISO 14443-3: Inicialización y anticolisión Esta parte describe las tramas de datos y las capas anticolisión utilizadas para descubrir los objetos que entran en el campo de un dispositivo inteligente; el formato de byte, los marcos y el tiempo usado durante la fase inicial de la comunicación entre la tarjeta y el terminal; la solicitud inicial y la respuesta a la solicitud del comando trasmitido. (ISO, 2006)

ISO 14443-4: Protocolo de transmisión, esta parte describe un protocolo de capa de transporte opcional. Este protocolo también se conoce como "T = CL". Este es un nombre derivado del uso común basados en los protocolos de Tarjetas Inteligentes T = 0 y T = 1. "CL[5]" (Openpcd, 2008)

Un estándar alternativo de Tarjetas Inteligentes sin contacto es el ISO 15693, el cual permite la comunicación a distancias de hasta 50 cm. Las más abundantes son las tarjetas de la familia MIFARE de Philips, las cuales representan a la ISO/IEC 14443-A. (ISO/IEC JTC1/SC17 N 1363, 1997) (smartcardbasics, 2010)

1.1.3 TARJETA INTELIGENTE DE CONTACTO

El segundo tipo de estas tarjetas disponen de unos contactos metálicos visibles y debidamente estandarizados en el ISO/IEC 7816. Estas tarjetas, por tanto, deben ser insertadas en una ranura de un lector para poder operar con ellas. A través de estos contactos el lector alimenta eléctricamente a la tarjeta y transmite los datos oportunos para operar con ella conforme al estándar. (smartcardbasics, 2010)

1.1.4 SEGURIDAD EN LAS TARJETAS INTELIGENTES.

La seguridad es una de las propiedades más importantes de las Tarjetas Inteligentes y se aplica a múltiples niveles y con diferentes mecanismos. Las medidas de protección van desde los elementos empleados en el material de la tarjeta, así como las características usadas en la impresión gráfica que dificultan la reproducción y alteración de las mismas; hasta reglas que se establecen para el acceso a la información contenida en el *chip*, según sus niveles de confidencialidad. (Markantonakis, 2004)

[5] Sin contacto

1.1.5 SEGURIDAD FÍSICA.

Según (Effing, 2003) el conjunto de medidas de seguridad física aplicable a una Tarjeta Inteligente está compuesta por los siguientes niveles:

- Nivel 1: Perceptibles mediante la vista al observar el documento. No requieren de herramientas especiales, por lo cual no es necesario entrenamiento. Pueden ser de conocimiento público.
- Nivel 2: Características escondidas que son visibles mediante equipos simples, luz ultravioleta. No requieren de un entrenamiento especial de los oficiales de seguridad.
- Nivel 3: Características de seguridad que requieren de un entrenamiento al personal y de un equipamiento especial para detectarlas, por ejemplo: un microscopio o lupas especiales.
- Nivel 4: Características de un alto nivel de seguridad sobre las que solo tiene conocimiento el personal requerido y que únicamente pueden verse mediante un equipamiento en un laboratorio especializado.

En las tarjetas se implementan distintos niveles de seguridad sobre los ficheros en dependencia de la importancia de la información contenida en ellos. Algunos elementos de seguridad son:

- Especificación de un conjunto de reglas para ejecutar comandos o acceder a datos, las cuales se denominan condiciones de acceso.
- Mensajería segura para la autenticación e integridad del intercambio de datos entre la tarjeta y los lectores.
- Protección de la información por claves que deben presentarse a la tarjeta y son verificadas por su sistema operativo y aplicaciones.
- Contador de confirmación para evitar ataques repetidos contra los ficheros protegidos.
- Mecanismos de autenticación mutua de la tarjeta y los terminales para garantizar que solo elementos autorizados puedan tener acceso a la información.

- Funciones criptográficas que permiten los procesos de firma digital para prevenir repudios.
- Utilización de la infraestructura de llave pública (PKI).
- Verificación biométrica por parte de la tarjeta. (Markantonakis, 2004)

1.1.6 COMUNICACIÓN CON LA TARJETA.

Cuando dos computadoras se comunican mutuamente, intercambian paquetes de datos, los cuales son construidos siguiendo un conjunto de protocolos. En forma similar, las Tarjetas Inteligentes se comunican al mundo exterior usando sus propios paquetes de datos llamados APDU (*Application Protocol Data Units*). Los APDU pueden contener un mensaje de comando o un mensaje de respuesta. El modelo amo– esclavo, es usado cuando el terminal o computadora cliente requiere comunicarse con la tarjeta. Por lo cual una Tarjeta Inteligente siempre juega el rol pasivo, esperando por un comando APDU desde un terminal. Esta entonces ejecuta la acción especificada en el APDU y responde a la terminal con un APDU respuesta. APDUs comandos y APDUs respuestas son intercambiados alternativamente entre una tarjeta y un terminal. (ISO/IEC, 2005)

1.1.7 ESTRUCTURA DE LOS APDU

Unidad de datos de protocolo de aplicación, es la unidad de comunicación entre un lector y una tarjeta. Su estructura está definida en el estándar ISO 7816-4, existiendo dos tipos de categorías de APDU, APDU *Command* (Comando APDU) **(Ver figura 1)** y APDU Response (APDU Respuesta). **(Ver figura 2)** (ISO/IEC, 2005)

- **Parejas de comando-respuesta**

Los comandos y respuestas se envían o se reciben de la Tarjeta Inteligente en forma de APDU. Su estructura es la siguiente:

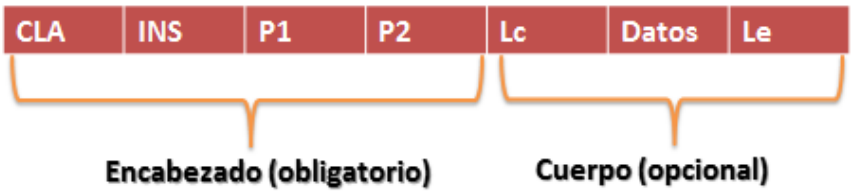


Figura 1: Estructura C- APDU.

Campos pertenecientes a la cabecera o encabezado:

1 byte para denotar la clase (CLA).
1 byte para denotar la instrucción (INS).

2 bytes para denotar los parámetros (P1-P2).

- **Campo Lc:** se compone de 0, 1 o 3 bytes si la longitud del campo de datos es mayor que cero. En caso contrario estará ausente.
- **Campo de datos:** bytes correspondientes a la longitud de la cadena de datos a enviar. Si no hay datos a enviar, no se especificará ningún valor en este campo.
- **Campo Le:** número de bytes esperados en la respuesta. Si no se espera ninguna cadena de bytes en la respuesta, no se especificará ningún valor en este campo. (ISO/IEC, 2005)

Figura 2: Respuesta con la cadena de datos a recibir y los bytes de *Status Word.*

- **Campo de datos de la respuesta**: cadena de bytes correspondiente a la longitud enviada en el comando. Si el campo Le del comando enviado no tenía asignado ningún valor, este campo no existirá.
- **Campo de Status Word**: 2 bytes denotados como SW1 y SW2 que indican el estado del proceso. (ISO/IEC, 2005)

1.2. PRINCIPALES ESTÁNDARES UTILIZADOS EN TARJETAS INTELIGENTES

Asociados a las Tarjetas Inteligentes hay un elevado número de estándares que rigen las características estructurales de las aplicaciones, dígase aplicaciones relacionadas con pasaporte, licencia de conducción y medicina, por mencionar algunas. Los estándares definen los datos a almacenar, las medidas de seguridad y la distribución de la estructura de ficheros asociada a la aplicación.

1.2.1. ISO 7816 1-15

Estándares internacionales para tarjetas con circuito integrado (Tarjetas Inteligentes). Cuyo objetivo es lograr la interoperabilidad entre distintos fabricantes de Tarjetas Inteligentes y lectores de las mismas, en lo que respecta a características físicas, comunicación de datos y seguridad. Estos estándares son basados en los ISO[6] 7810 e ISO 7811, los cuales definen características físicas de tarjetas de identificación. (cardwerk, 2010)

[6] Organización Internacional de Normalización

Capítulo 1: Marco teórico

La apariencia física de las Tarjetas Inteligentes está especificada por la Organización Internacional de Normalización (ISO) y Comisión Electrotécnica Internacional (IEC). La ISO 7816 presenta partes del estándar, las cuales son:

- 7816-1: Características físicas.
- 7816-2: Tarjetas con contactos - Dimensiones y localización de los contactos.
- 7816-3: Características eléctricas.
- 7816-4: Pares comando-respuesta.
- 7816-5: Registro de la solicitud de los proveedores.
- 7816-6: Interoperabilidad en los elementos de datos para el intercambio.
- 7816-7: Interoperabilidad en los comandos de la tarjeta (SCQL).
- 7816-8: Comandos para operaciones de seguridad.
- 7816-9: Comandos para la gestión de la tarjeta.
- 7816-10: Señales electrónicas para operación síncrona.
- 7816-11: Verificación de la identidad personal a través de métodos biométricos.
- 7816-12 Tarjetas con contactos. Interfaz eléctrica USB y procedimientos operativos.
- 7816-13: Comandos de administración de aplicaciones en múltiples aplicaciones entorno.
- 7816-15: Aplicación de información criptográfica.

El estándar 7816 abarca todo lo referente a las Tarjetas Inteligentes de contacto, para los desarrolladores de aplicaciones la parte más utilizada del estándar es la 7816-4 ya que define:

- El contenido de los pares comando-respuesta que se intercambian a nivel de interfaz.
- Estructuras para aplicaciones y datos en la tarjeta.
- La estructura y contenido de los caracteres históricos de la respuesta de *Reset*[7], los cuales describen las características de operación de la Tarjeta Inteligente.
- Métodos para extracción de objetos y elementos de datos de la tarjeta.
- Mecanismos para la identificación y direccionamiento de aplicaciones en la tarjeta.
- Estructuras de archivos y métodos de acceso.
- Arquitectura de seguridad para derechos de acceso a los archivos y datos en la tarjeta.

[7] *Reset*: Se conoce como la puesta en condiciones iniciales de un sistema. Este puede ser mecánico, electrónico o de otro tipo. Normalmente se realiza al conectar el mismo, aunque, habitualmente, existe un mecanismo, normalmente un pulsador, que sirve para realzar la puesta en condiciones iniciales manualmente.

- Comandos orientados a objetos de datos.
- Métodos de acceso a los algoritmos que procesa la Tarjeta Inteligente (no describe los algoritmos).
- Métodos para el intercambio seguro de mensajes. (ISO/IEC, 2005)

1.2.2. PC/SC

PC/SC[8] (ordenador personal/Tarjeta Inteligente) es un marco estándar para el acceso de Tarjetas Inteligentes en diferentes plataformas. La especificación de interoperabilidad de las CCI[9] y PC / SC, ha sido desarrollado para facilitar el uso de las Tarjetas Inteligentes en los ordenadores. La ventaja de PC/SC es que las aplicaciones no tienen que reconocer los detalles correspondientes al lector de tarjetas en la comunicación con la Tarjeta Inteligente. (PC/SC, 2005)

1.2.3. GLOBALPLATFORM

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

Las tarjetas, dispositivos y sistemas GlobalPlatform, son interoperables, independientemente de la tecnología del proveedor y la flexibilidad de su infraestructura técnica, garantizan que pueda responder a las necesidades básicas en el instante del despliegue inicial. Ofreciendo a los emisores la seguridad de que la infraestructura que han elegido será capaz de adaptarse y crecer a medida que cambian las condiciones de negocios. (GlobalPlatform, 2003)

Este estándar lidera mundialmente el desarrollo en temas de infraestructura de Tarjetas Inteligentes. Sus descripciones técnicas probadas para las tarjetas, dispositivos y sistemas son consideradas como las normas de la industria para lograr implementaciones

[8] Se refiere a (*Personal Computer/Smart Card*), en español, computadora personal / Tarjeta Inteligente.

[9] Tarjetas de circuito integrado.

interoperables, flexibles y sostenibles por tarjetas que soportan multiaplicación y multi-actor e implementaciones multi-modelo de negocio.

GlobalPlatform comprende la instalación y borrado de las aplicaciones y la administración de otras tareas de las tarjetas. El emisor de la tarjeta tiene el control total sobre el contenido de esta, pero puede permitir a otras instituciones administrar sus propias aplicaciones, esto se logra aplicando protocolos criptográficos, permitiendo a cada institución tener su propia área segura en la tarjeta. (GlobalPlatform, 2014)

1.3. CONCEPTOS ASOCIADOS AL DOMINIO DEL PROBLEMA

1.3.1. *MIDDLEWARE*

El *middleware* es un *software* de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de *software* distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red). El *middleware* abstrae de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API[10] para la programación y manejo de aplicaciones distribuidas. Dependiendo del problema a resolver y de las funciones necesarias, serán útiles diferentes tipos de servicios de *middleware*. (Rfidpoint, 2014)

1.3.2. *APPLET JAVA*

Un *applet* es un componente de una aplicación que se ejecuta en el contexto de otro programa. El *applet* debe ejecutarse en un contenedor, que lo proporciona un programa anfitrión, mediante un *plug-in*[11], o en aplicaciones como teléfonos móviles que soportan el modelo de programación por '*applet*s'. (Fedo, 2013)

A diferencia de un programa, un *applet* no puede ejecutarse de manera independiente, ofrece información gráfica y a veces interactúa con el usuario, típicamente carece de sesión y tiene privilegios de seguridad restringidos. Un *applet* normalmente ejecuta una función específica que carece de uso independiente.

Entre sus características se puede mencionar, un esquema de seguridad que permite que los *applet*s que se ejecutan en el equipo, no tengan acceso a partes sensibles (por ejemplo no tengan permisos de escritura de archivos), sin el nivel de autorización requerido; la

[10] API: Una interfaz de programación de aplicaciones o API (del inglés *Application Programming Interface*) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción. Son usados generalmente en las bibliotecas.

[11] Extensión en español

desventaja de este enfoque es, que la entrega de permisos es engorrosa para el usuario común, lo cual no favorece a uno de los objetivos de los Java *applet*s: proporcionar una forma fácil de ejecutar aplicaciones desde el navegador *web*. (Prezi, 2014)

1.4. SISTEMAS OPERATIVOS EN TARJETAS INTELIGENTES

Oscar es una de las primeras implementaciones de sistema operativo para Tarjetas Inteligentes de terceros realizado por la universidad de Cambridge (Reino Unido). Define en la tarjeta el almacenamiento de archivos estructurados. Su característica principal es lo que se llamó la "tarjeta virtual", que separa las zonas de almacenamiento y un mecanismo para inicializar cada zona de forma segura con su propio juego de llaves. Con Oscar, una tarjeta virtual podría ser personalizada después de la emisión. (FRZ, 2005)

Macsime/TOSCA un entorno de ejecución de tarjetas portátil diseñado e implementado en Zaandam, Países Bajos, dirigido al mercado de pagos con un enfoque en la evaluación de seguridad de alto nivel. El nombre fue cambiado a TOSCA cuando la tecnología fue transferida a la Integridad Artes de California. El equipo de desarrolladores en 1996 decide utilizar ADA como lenguaje de programación para el desarrollo del sistema operativo.

Blue sistema operativo de Tarjetas Inteligentes construido originalmente por Digicash en los Países Bajos para apoyar el protocolo de pago eCash, tenía como propósito brindar fiabilidad a los usuarios de manera que el servicio de transacciones financieras fuera respaldado por la evaluación de seguridad de alto nivel.

MultOS: sistema operativo abierto de Tarjeta Inteligente desarrollado por Maosco Ltd., una empresa inglesa asociada al sistema bancario Mondex. Este sistema operativo centra su atención en el nivel de seguridad y de rigurosidad, debido a su vinculación bancaria. Esta misma razón es la que hace bastante difícil la obtención de especificaciones y muestras de tarjetas, lo cual ocurre para incrementar el grado de seguridad. Las rutinas se realizan en un lenguaje que se comercializa bajo una licencia propietaria, denominado MEL, aunque se proporciona un conversor de lenguaje C a lenguaje MEL. (REILLO, 2000)

Java Card: Esta tecnología permite en Tarjetas Inteligentes y similares dispositivos incrustados, ejecutar aplicaciones Java. Ofrece la posibilidad y capacidad al usuario de implementar soluciones aplicadas en distintas esferas de la sociedad y la economía, con elevados niveles de seguridad, ejemplo de ello son las aplicaciones que permiten la firma digital, la autenticación de usuarios, tarjetas de monedero electrónico y además, en las tarjetas módulo de identificación del suscriptor (SIM por sus siglas en inglés) utilizadas en la telefonía celular. (Perovich, 2010)

Una Tarjeta Inteligente Java Card posee una estructura lógica además, de su estructura física. El microprocesador es el encargado de realizar todas las operaciones, mientras el sistema operativo se encarga de traducir las acciones de las capas superiores en instrucciones que el microprocesador pueda entender. La máquina virtual de Java Card permite la ejecución de aplicaciones dentro de la tarjeta, mientras que las interfaces de programación de aplicación (*APIs* por sus siglas en inglés) de poseen funcionalidades utilizadas por los *applets*. (Perovich, 2010)

La tabla 1 resume algunas de las principales características de los sistemas operativos de Tarjetas Inteligentes y su aparición ordenados de manera cronológica, actualmente según (Vázquez, 2014) el Java Card por las facilidades específicas para el manejo de transacciones con Tarjetas Inteligentes (atomicidad de un grupo de operaciones, objetos persistentes.) y bajo el lema "escribe una vez, corre en cualquier lado" hizo que los desarrolladores de este tipo de aplicación lo prefieran. El número de aplicaciones para Java Cards, y SmartCards en general, va en un aumento constante y el hecho de estar basado en lenguaje Java hace a Java Card atractivo para el desarrollo de aplicaciones (*applet*) para dispositivos inteligentes

Tabla 1: Características de los sistemas operativos en Tarjetas Inteligentes.

Tecnología/Sistema operativo	Oscar	Tosca	Blue	MultOS	Java Card
País	Reino unido	Estados unidos y los países bajos	países bajos	Reino Unido	EE.UU
Aparición mundial	1990	1994/1995	1996	1996	1997
Intérprete		SCIL / CLASP	J-Code	MEL	Java Card Byte Code
Lenguaje de programación		Data model / ADA	J-Code	MEL / C	Java
administrador de tarjetas	master file			Multos	Open Platform

Multiaplicación	x	x		x	x
Interfaz compartida		X		Vía I/O buffer	API

1.4.1. JAVA CARD *APPLET*

Un *applet* de Java Card es un programa de Java que cumple un conjunto de convenciones que permiten ejecutarlo en el entorno de ejecución de Java Card. Un *applet* de Java Card no está diseñado para ejecutarse en un explorador. Estos se pueden cargar en el entorno de ejecución después de que la tarjeta haya sido fabricada, es decir a diferencia de las aplicaciones de muchos sistemas embebidos, los *applet*s no necesitan ser "quemados" en la ROM[12] durante el proceso de fabricación, más bien los *applet*s pueden ser bajados dinámicamente a la tarjeta en un instante posterior a la fabricación. El entorno de ejecución de Java Card soporta un entorno multiaplicación, por lo que pueden coexistir múltiples *applet*s en una sola Tarjeta Inteligente y un *applet* puede tener múltiples instancias. (GlobalPlatform, 2003)

Los *applet*s Java Card son las aplicaciones que corren dentro de una tarjeta Java Card. Dichas aplicaciones interactúan en todo momento con el *Java Card Runtime–Environment* (JCRE) utilizando los servicios que este brinda, e implementan la interfaz definida en la clase abstracta javacard.framework.*Applet*.

Finalmente, luego de haber realizado un estudio previo sobre Java Card se ha llegado a la conclusión de que esta es la tecnología más adecuada para el desarrollo de aplicaciones Java que se puedan ejecutar dentro de las Tarjetas Inteligentes, siempre y cuando estas tarjetas tengan como sistema operativo Java Card.

1.5. MODELO DE DESARROLLO TRADICIONAL

Un modelo según (RAE, 2014) es un: "arquetipo o punto de referencia para imitarlo o reproducirlo". En el desarrollo tradicional de aplicaciones para Tarjetas Inteligentes Java Card, se tienen en cuenta un grupo de elementos que a pesar de ser independientes interactúan entre sí. Entre ellos se encuentran, primeramente, los *applets* Java Card que son ejecutados dentro de la tarjeta. Además, intervienen en el conjunto de una aplicación los elementos correspondientes al terminal o *host*. El terminal contiene las aplicaciones que comúnmente residen en un ordenador. (Modelo para la extensión de las capacidades de

[12] *Read only memory* en español memoria de solo lectura

procesamiento y memoria de tarjetas inteligentes Java Card, 2013) (Extending the data storage capabilities of a Java-based smartcard, 2001)

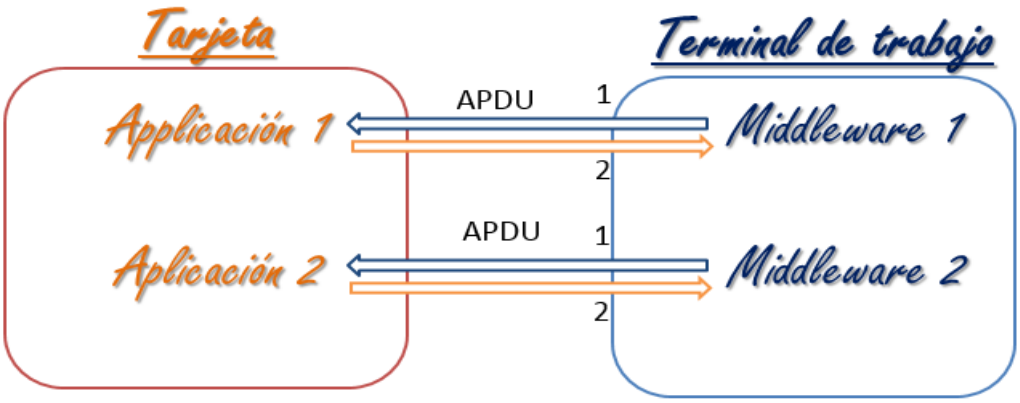


Figura 3: Comunicación tarjeta-Terminal de trabajo.

La comunicación entre el terminal y un *applet* Java Card, se basa en el modelo de comunicación por paso de mensajes, usando comandos APDU. Los *middleware*s envían comandos APDU con las peticiones a la Tarjeta Inteligente, mediante la comunicación con un lector de tarjeta que utiliza el estándar PC/SC para la conexión con el ordenador. Dichos comandos APDU son recibidos y procesados por un *applet* específico que devuelve una respuesta en el mismo formato APDU.

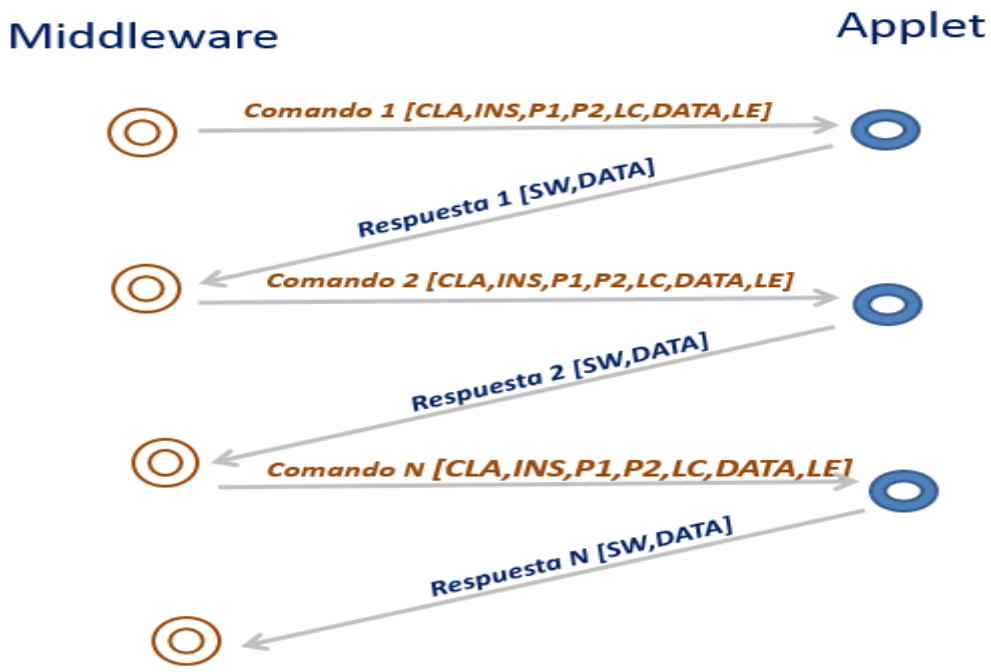


Figura 4: Envío y recepción de APDU.

1.5.1. LIMITACIONES EN EL USO DEL MODELO DE DESARROLLO TRADICIONAL EN LA *WEB*

El enfoque que se le ha dado a la interacción con Tarjetas Inteligentes, a consideración del autor de este trabajo quedó marcado por la naturaleza de la comunicación con este dispositivo, según (Ortega, 2008). Las tarjetas han sido vistas más como un elemento de soporte al servicio del usuario, que como una entidad autónoma y con capacidad de comunicarse con el sistema de forma segura. Algunas de las consideraciones que ponen de manifiesto este hecho son:

Inseguridad de las llaves. Históricamente los *middleware* para la comunicación con las Tarjetas Inteligentes se han instalado en la computadora cliente y esto genera la inseguridad de las llaves simétricas utilizadas para la comunicación segura entre el terminal y la Tarjeta Inteligente ya que estas llaves están accesibles a intrusos que accedan al terminal.

No cooperación entre *middlewares.* Frecuentemente una misma entidad es la encargada de brindar servicios que utilicen Tarjetas Inteligentes y por tanto, en un terminal pueden converger varios *middlewares* los cuales se ejecutan de manera secuencial o sea que tiene que utilizarse uno a la vez y cada uno de ellos, aunque cumplan funciones similares, no comparten información y tienen diferentes maneras de manejar la seguridad.

Marcas de tiempo. Otro condicionante histórico en la comunicación con Tarjetas Inteligentes ha sido la imposibilidad de emplear marcas de tiempo (*time-stamp*), que a menudo son requeridas en este tipo de aplicación, para garantizar la frescura de ciertos mensajes, por tanto, no es recomendable su utilización en la autenticación con Tarjetas Inteligentes ya que existen escenarios en los que el usuario debe insertar su tarjeta en un dispositivo desconocido y público.

Comunicación remota. Las Tarjetas Inteligentes no tienen protocolos de comunicación remota. Si bien para los procesos de comunicación local la tarjeta se comporta como cualquier otro dispositivo con una interfaz de comunicación, para lograr la comunicación remota, las aplicaciones de *software* se han valido de un grupo de técnicas y componentes que simulan de manera síncrona una comunicación local.

Aunque las aplicaciones que usan Tarjetas Inteligentes se desarrollan mayormente utilizando el modelo de desarrollo tradicional, cada día los servicios que utilizan este tipo de dispositivos a través de la *web* son más populares y con mayor número de usuarios asociados, actualmente las entidades que proveen estos servicios centralizan la complejidad y los tecnicismos asociados a este tipo de *software* y su seguridad, proporcionándole al

usuario una interfaz fácil de manejar desde cualquier lugar y utilizando cualquier terminal que permita la lectura de una Tarjeta Inteligente.

1.5.2. MODELO DE DESARROLLO CENTRALIZADO PARA *MIDDLEWARE* DE TARJETAS INTELIGENTES

En arquitectura este modelo es conocido como *mainframe architectures* y una de las características fundamentales es que el procesamiento de negocio se realiza en un servidor centralizado y los usuarios interactúan a través de terminales con interfaces personalizadas que permiten la comunicación con el servidor.

Las aplicaciones de escritorio que utilizan Tarjetas Inteligentes son orientadas a un servicio específico y cuentan con un *middleware* como mediador o capa de conversión entre los *applets* y el terminal.

El 9 de Mayo de 2001 la empresa Kalysis revolucionó el uso de las Tarjetas Inteligentes en la *web* con la patente P200101056. Desde entonces posee las primeras plataformas multiaplicación basadas en tecnología de Tarjetas Inteligentes contribuyendo a la transformación de la identidad y el dinero digital. (Kalysis, 2010)

La plataforma creada por la empresa cambió el paradigma tradicional de las aplicaciones de escritorio uniendo de manera congregada un grupo de servicios vinculados al sector financiero a través de la red

Algunas de las ventajas relacionadas a los servicios en línea de Tarjetas Inteligentes con el modelo de desarrollo centralizado son:

- Permisos para ejecutar varias aplicaciones en tiempo real y acceso al sistema por varios usuarios simultáneamente.
- Reduce los costos de mantenimiento y gestión de las mismas prestaciones. El mantenimiento y la reparación se pueden realizar en un ordenador central, sin la máquina. Esta característica es especialmente útil en los servicios centrales que requieren disponibilidad continua y en los que una posible interrupción del servicio podría causar un enorme daño.
- Posibilita el resguardo de las llaves simétricas encargadas de la comunicación segura entre las aplicaciones que se ejecutan en la Tarjeta Inteligente y el terminal.
- Favorece la incorporación de nuevas aplicaciones y la actualización de funcionalidades a las soluciones existentes sin afectar al usuario final.

1.6. ARQUITECTURA DE SEGURIDAD DE TIPO LÓGICA

Los estándares GlobalPlatform e ISO 7816-4 proponen arquitecturas de seguridad para Tarjetas Inteligentes, ambos definen componentes que se encuentran en el *chip* de la tarjeta, y que deben ser considerados por el terminal para poder establecer la comunicación segura con la misma.

1.6.1. ARQUITECTURA DE SEGURIDAD GLOBALPLATFORM

La principal característica de una Tarjeta Inteligente con un sistema operativo que cumpla con las especificaciones de GlobalPlatform es elevar la seguridad y asegurar la integridad de los componentes que conforman la arquitectura de la tarjeta, los componentes encargados de la seguridad dentro de GlobalPlatform son: el entorno de ejecución, el *OPEN*, el emisor de dominio de seguridad y el dominio de seguridad.

El entorno de ejecución es el responsable de realizar la gestión de memoria segura para garantizar que:

- El código de cada aplicación y los datos (incluidos los datos de la sesión transitorios), así como el tiempo de ejecución propio y sus datos (incluyendo los datos de sesión transitorios), estén protegidos.

- Los contenidos que estuvieron almacenados en la memoria transitoria no sean accesibles cuando se vuelve a utilizar la tarjeta.

- El proceso de recuperación de la memoria sea seguro y consistente en caso de una pérdida de potencia o la retirada de la tarjeta o del lector mientras que una operación está en curso.

Por otro lado, el OPEN deberá: proporcionar una interfaz para todas las aplicaciones asegurando que los mecanismos de seguridad GlobalPlatform no puedan ser desactivados, corrompidos o eludidos. Además, comprobará las reglas de acceso a las aplicaciones instaladas de acuerdo con sus privilegios, administrando su ciclo de vida.

El **Dominio de seguridad del proveedor** permitirá proporcionar servicios de protección criptográfica para las aplicaciones instaladas durante su personalización y definirá un contexto de seguridad primario llamado CardManager[13]. La Tarjeta Inteligente podrá tener otros dominios de seguridad denominados dominios de seguridad suplementarios (SSD por

[13] Es la aplicación representante del proveedor de la tarjeta.

las siglas en inglés: *Supplementary Security Domain*). Todas las aplicaciones que se instalen en una tarjeta con esta arquitectura, deben pertenecer a un dominio de seguridad (GlobalPlatform, 2003)

El dominio de seguridad puede contener varios conjuntos de llaves, que pueden ser simétricas o asimétricas. Estas llaves son utilizadas en los protocolos de autenticación mutua y canal seguro, permitiendo garantizar la autenticidad, integridad y confidencialidad entre el terminal y la tarjeta. (GlobalPlatform, 2004) El conjunto de llaves simétricas MAC[14], ENC[15] y DEK[16] son obtenidas a través de algoritmos de derivación a partir de una llave semilla.

Cada dominio de seguridad es visto como una aplicación embebida en la tarjeta, y por lo tanto, al igual que las aplicaciones que se instalan en el *chip*, tiene un conjunto de privilegios y parámetros específicos que describen su utilidad. (GlobalPlatform, 2003)

La documentación GlobalPlatform define ampliamente los procesos de autenticación mutua y mensajería segura por encima de lo que se define en las especificaciones ISO. En aras de esclarecer el proceso de envío de comandos e interpretación de las respuestas recibidas, se describirán los procesos antes mencionados.

AUTENTICACIÓN MUTUA

El proceso de autenticación mutua se basa en el principio de que ambas entidades, el terminal y la tarjeta, deben demostrar que tienen conocimiento del mismo secreto. Se logra mediante el proceso de iniciar un canal seguro garantizando a la tarjeta y a la entidad fuera de la tarjeta que se están comunicando de manera segura. Si algún paso en la autenticación mutua falla, el proceso se reanudará.

Uno de los mecanismos más utilizados es el de Reto-Respuesta (también conocido como *Challenge-Response*), mediante el cual las partes intercambian información generada de forma aleatoria y mediante la conformación de criptogramas, utilizando las llaves que ambos conocen, son intercambiados comandos para comprobar que el reto enviado por el terminal coincide con el respondido por la tarjeta y viceversa. En el protocolo de canal seguro '01' (SCP por las siglas en inglés*: Secure Channel Protocol*) o SCP01, la autenticación mutua se realiza de forma simétrica, utilizando los comandos *Initialize Update* y *External Authenticate*.

[14] Llave para el cálculo del Código de autenticación de mensaje (traducido de las siglas en inglés MAC: *Message authentication code*).

[15] Llave para el cifrado del canal seguro (traducido de las siglas en inglés ENC: *Encryption key*).

[16] Llave para el cifrado de información sensible (traducido de las siglas en inglés DEK: *Data encryption key*).

La figura 8 muestra el flujo del proceso de autenticación mutua según el estándar GlobalPlatform 2.1.1. (GlobalPlatform, 2003)

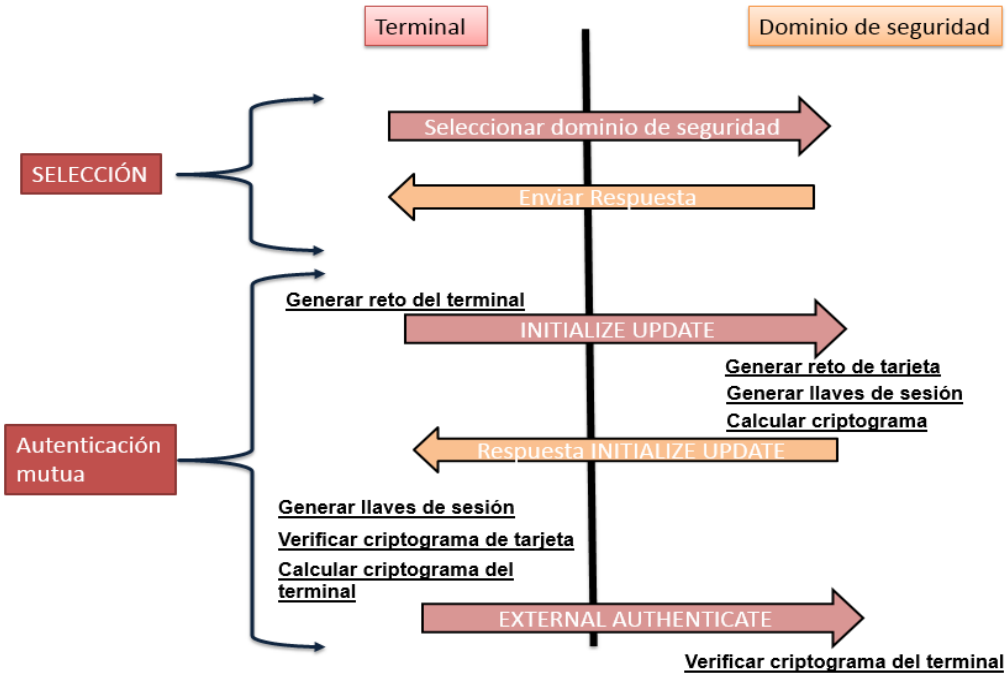


Figura 8: Flujo de autenticación mutua (Canal seguro entre el *host* o *Middleware* y el dominio de seguridad en el interior de la tarjeta) tomado de las especificaciones del GlobalPlatform.

MENSAJERÍA SEGURA

La mensajería segura es la última fase en el protocolo de canal seguro y su ciclo de vida es el mismo que el de la sesión abierta entre el terminal y la tarjeta. La misma permite a una entidad fuera de la tarjeta agregar confidencialidad e integridad a la composición de un mensaje APDU, enviándolo con el nivel de seguridad acordado durante la negociación del canal de comunicación. Una vez generadas las llaves de sesión para un canal seguro, que se derivan de las llaves estáticas para MAC y ENC, se utilizan algoritmos simétricos como el 3DES en los modos ECB/CBC para el cifrado de los datos de un comando APDU, y los algoritmos *Full* Triple DES MAC y *Retail* MAC para el cálculo de la MAC de los comandos. (GlobalPlatform, 2003)

1.6.2. ARQUITECTURA DE SEGURIDAD DE ISO 7816-4

El estándar ISO 7816-4 define contenedores denominados Ficheros Elementales (EF por las siglas en inglés: *Elementary Files*), los cuales pertenecen a un Fichero Dedicado (DF por las siglas en inglés: *Dedicated File*), conformando así un sistema de archivos cuya raíz es el

Fichero Maestro (MF por las siglas en inglés: *Master File*). Cada EF presenta determinadas condiciones de seguridad que deben ser cumplidas para permitir su acceso.

Esta arquitectura de seguridad está compuesta por 3 elementos fundamentales: los estados, atributos y mecanismos de seguridad.

El estado de seguridad es el resultado de la finalización de un procedimiento de autenticación, el estándar considera 4 estados principales:

El estado de la seguridad global: se alcanza por la terminación de un procedimiento relacionado con el MF de autenticación (por ejemplo, autenticación de la entidad por una contraseña o una llave colocada en la MF).

El estado de seguridad específica de la aplicación: se alcanza por la terminación de un procedimiento de autenticación y depende del canal lógico de comunicación.

El estado de seguridad específica de archivo: se arriba por la finalización de una autenticación relacionada con los DF.

El estado de seguridad específico del comando: solo se alcanza al procesar un comando con mensajería segura.

Los atributos de seguridad definen qué acciones están permitidas y en qué condiciones. La seguridad de un archivo depende de su categoría (DF o EF) y los parámetros opcionales en su información de control de archivos y/o en los archivos padres. Los atributos de seguridad también pueden estar asociados a los comandos, los objetos y las tablas permitiendo: Especificar el estado de seguridad de la tarjeta antes de acceder a los datos, restringir el acceso a datos de ciertas funciones (por ejemplo, solo lectura) y definir las funciones de seguridad que se llevarán a cabo para arribar a un estado de seguridad específico.

Los mecanismos de seguridad que propone el estándar son:

Autenticación de la entidad con una contraseña (Pin[17]). La tarjeta compara los datos recibidos de la entidad fuera de la tarjeta con datos internos secretos. Este mecanismo puede ser utilizado para la protección de los derechos del usuario.

Autenticación de la entidad con una llave. La entidad para autenticar tiene que demostrar el conocimiento de la correspondiente clave secreta o privada en un procedimiento de autenticación.

[17] Número de identificación personal

Autenticación de datos. De manera general la tarjeta computa un elemento de datos (suma de comprobación criptográfica o firma digital) y lo inserta en los datos enviados a la entidad fuera de la tarjeta. Este mecanismo es utilizado para la protección de los derechos de un proveedor.

Cifrado de datos. La tarjeta calcula un criptograma y lo inserta en un campo de datos, posiblemente junto con otros datos. Este mecanismo puede ser utilizado para proporcionar un servicio de confidencialidad, por ejemplo, la gestión de claves, permiso y acceso a fichero de manera segura.

1.7. SOLUCIONES DESARROLLADAS CON TARJETAS INTELIGENTES

1.7.1. SOLUCIONES INTERNACIONALES

Coesys eGov 2.0 Es una solución que simplifica radicalmente la interfaz de usuario final, para proporcionar una mayor facilidad de uso. Diseñado para ofrecer un alto nivel de seguridad en línea, la última versión ahora permite firmas digitales para documentos PDF y formularios en línea. No requiere de *software* en la computadora del cliente, simplificando el despliegue de servicios y potenciando una mayor asimilación (Gemalto, 2009).

Entre sus características principales están:

- Servicios de conectividad de Tarjetas Inteligentes.
- Servicio de autenticación.
- Federación de identidad.

La solución Coesys eGov 2.0 se compone de:

- Un servidor para servicios de post emisión y autenticación.
- *Middleware* basado en el servidor, lo que proporciona un vínculo entre la aplicación de la tarjeta de identificación electrónica y la aplicación del servidor de autenticación.
- La tecnología de conectividad *SConnect*, permitiendo una conectividad sin fallas entre la tarjeta y el servidor de autenticación.

SConnect Es una tecnología que utiliza una plataforma y un navegador permitiendo a las aplicaciones y servicios de Internet conectarse a cualquier Tarjeta Inteligente. Su objetivo principal es el de proporcionar un puente de conexión entre el JavaScript, que corre en el navegador y la Tarjeta Inteligente, permitiendo la conectividad entre estas últimas aplicaciones y los servicios *web*. (Gemalto, 2009)

SConnect: una extensión del navegador *web* que se conecta con la capa PC/SC estándar del ordenador, conectando una página *web* con una Tarjeta Inteligente, que se comunica

con un ordenador host vía PC/SC. Además, incluye un sistema de medidas de seguridad para mitigar ciertos riesgos y proteger a los usuarios y sitios *web* conectados. Estas medidas incluyen una extensión con firma digital de *SConnect*, un HTTPS reforzado, llave de conexión, validación de servidor, alarma a usuario. Esta solución brinda la posibilidad de instalar el *middleware* en el servidor, lo que trae consigo beneficios como: solución de procesos de instalación, flexibilidad en la actualización de las funciones del middleware e interoperabilidad con varias Tarjetas Inteligentes. (Gemalto, 2009)

Kalysis.mei

Plataforma Multiaplicación de Identidad y Pago Electrónico basada en Tarjetas Inteligentes que desarrolla soluciones masivas de autenticación y pago electrónico -*loyalty*, *e-Banking*, *e-purse*, *vending*, *e-gift* en redes de comunicaciones (LAN[18], WAN[19], Internet, telefonía tradicional y UMTS[20]) para Tarjetas Inteligentes. Incluye firma digital y firma electrónica avanzada (*strong authentication*). Compatible con plataformas y sistemas existentes le permite establecer sistemas propios y autónomos frente a las plataformas conocidas de autenticación y pago (firma electrónica, bancarias, tarjetas de crédito). (Kalysis-solouna, 2012)

La compañía **Kalysis** proveedora de esta plataforma cuenta con:

- Una base de clientes a los cuales ofrece de manera segura el acceso a sus servicios en línea.
- Un surtidor de productos y servicios especialmente dedicados a los lectores de Tarjetas Inteligentes, estándares y herramientas de desarrollo, para el ciclo de vida de sus proyectos basados en tarjetas.
- Una línea completa de aplicaciones dedicadas, productos de seguridad para PC y redes de comunicaciones basadas en la tecnología de Tarjetas Inteligentes.

Kalysis permite a los usuarios que se benefician de sus servicios, la descarga de un componente certificado que instalan en su computadora y así pueden tener acceso a su dispositivo inteligente a través de la *web*.

La ventaja principal de los productos de Kalysis es que las soluciones son independientes de las librerías propietarias para Smart Cards de Microsoft, o incluso de los controladores de los fabricantes, por lo que su sistema nunca quedará ligado a ningún proveedor de hardware, ni comprometido al *software* o licencias de terceros.

[18] Red de área local

[19] WAN es una red punto a punto o red de paquete conmutado.

[20] *Universal Mobile Telecommunications System*

La desventaja fundamental es que la tarjeta MEI® solo servirá para tener acceso único a multitud de servicios brindados por la empresa Kalysis.

EDL :(Licencia de Conducción Electrónica): Es la implementación de una aplicación *desktop*, que a través de un *middleware* y un *applet* permiten gestionar la información del Documento Electrónico Licencia de Conducción, siendo un *software* gratuito y disponible en la *web* bajo la licencia GPL[21] 2.0.
Esta aplicación cumple con el estándar ISO/IEC 18013 para la Licencia de Conducción Electrónica, el cual establece los datos que la misma debe contener, características, seguridad, entre otros parámetros por los cuales se debe regir. Algunos de los grupos de datos (DG) fundamentales con los cuales trabaja esta solución son:

- DG1: Establece los datos personales del titular que son almacenados en el Documento Electrónico Licencia de Conducción nombre, apellidos, fecha de nacimiento, día en que expira el documento, país donde se realiza el proceso de otorgamiento del Documento Electrónico Licencia de Conducción y autoridad que lo acredita, además, de las categorías de vehículos internacionales. (ISO/IEC 18013-3, 2009)

- DG2: Contiene datos como: sexo, altura, peso, color de los ojos, color de pelo, lugar de nacimiento y lugar de residencia.

- DG3: Contiene los datos para identificar el documento electrónico, como número de licencia y número de administración.

- DG4 y DG5: Contienen dos imágenes del titular, la primera para su identificación y la segunda una imagen con su firma digital.

- DG6789: Contienen los datos biométricos que se utilizan para realizar la protección de acceso extendido (EAP[22]).

- DG10: Es un grupo de datos reservado para alguna información específica que se le quiera agregar en un futuro a la licencia de conducción electrónica.

- DG11: Contiene las categorías de vehículos nacionales que puede conducir el titular.

[21] General Public License

[22] Protección de Acceso Extendido

- DG13: Contiene el identificador del algoritmo SHA[23] 256 con RSA[24] y la llave pública utilizada en la autenticación activa.

- COM: Elemento que muestra todos los grupos de datos e indicadores de los mecanismos de seguridad que contiene el Documento Electrónico Licencia de Conducción.

- SOD: Son todos los objetos de seguridad (certificados, algoritmos de encriptación, llaves privadas) que se almacenan en el *chip* de la tarjeta. (ISO/IEC 18013-3, 2009)

Los mecanismos de seguridad que se implementan para controlar el acceso a los datos, la autenticación entre el *applet* y el *middleware,* así como prevenir la sustitución del *chip* son:

Protección de acceso básico, este mecanismo de seguridad previene de la lectura no autorizada.

Autenticación pasiva, consiste básicamente en la verificación del SOD[25], este mecanismo de seguridad no requiere capacidades de procesamiento del *chip*, su objetivo es probar que el contenido del SOD y su LDS [26]son auténticos y no han sido cambiados, pero no previene la copia exacta del *chip* o su sustitución. (ICAO 9303 vol1part2, 2006)

Autenticación activa (AA), proteger al documento de la sustitución del *chip*. (ISO/IEC 18013-3, 2009)

Las soluciones desarrolladas con Tarjetas Inteligentes en el ámbito internacional aportaron información sobre las concepciones comunes que debe tener una aplicación que brinde servicios que utilicen Tarjetas Inteligentes, algunas de las categorías que destacaron en el modelo de desarrollo tradicional fueron: *middleware,* mecanismos de seguridad y *applet.* Para las aplicaciones que brindan servicios a través de la *web* se evidencia un tipo de conector o componente que permite la comunicación entre el JavaScript embebido en la página que se le muestra al usuario en el terminal y la tarjeta conectada al terminal. (Gemalto GemBorder , 2006)

1.7.2. SOLUCIONES EN EL DEPARTAMENTO DE TARJETAS INTELIGENTES.

Soluciones del departamento de componentes del CISED:

MediCard (Patterson, 2011): es un *applet* Java Card y su componente *middleware* para gestionar la información de los historiales clínicos utilizando Tarjetas Inteligentes, con el

[23] Secure Hash Algorithm, una función resumen

[24] (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública

[25] SOD: Contiene los valores condensados del LDS que se están utilizando y es almacenado en el EF.

[26] LDS: Grupo de datos almacenados en el *chip*.

objetivo de proporcionar un mejor manejo de la información de estos datos. Alguna de las ventajas que tiene esta aplicación es que permite la actualización desde cualquier puesto de trabajo y desde cualquier lugar, evitando los tiempos de traslado por el centro sanitario, facilita la explotación de datos para medir y mejorar la calidad de la asistencia sanitaria, imposibilitando la alteración o manipulación de la información contenida en la historia clínica electrónica por parte de personas no autorizadas.

Estándares utilizados:

- *Health Level Seven* (HL7).
- Estándar CEN TC 251 (prENV 136061)
- Estándar 18130 *Requirements for an Electronic Health Record Reference.* (CARNICERO, 2010)

Medidas de Seguridad: Autenticación mutua nivel 0(*no security*).

Lenguaje de programación utilizado: *Applet*: Java Card, *Middleware: C#.*

Control de acceso a instalaciones con Tarjetas Inteligentes (Pérez, 2010):

La solución está conformada por un *applet* que está contenido dentro de la tarjeta y un *Middleware* encargado de la interacción con el mismo. El sistema se encarga de confirmar que el usuario es el real portador de la tarjeta a través de la comparación del PIN introducido por este y el que se encuentra almacenado en la tarjeta, a partir de entonces se procede a la obtención de los datos almacenados en la misma. La instalación puede hacer uso del sistema para varias aplicaciones dentro de un mismo recinto. Algunas de las ventajas de la solución cuando se hicieron las pruebas pilotos con las tarjetas seleccionadas fueron:
Desde el punto de vista local:

- Un nivel más profesional de control y auditoría.
- Mayor trazabilidad ocasionada por la obtención de marcajes en tiempo real.
- Mayor eficiencia con menor coste humano en la gestión de RRHH.

Desde el punto de vista de los usuarios:

- Un sistema más cómodo, fiable y seguro que los anteriores.
- Eliminación de errores en la identificación.

Medidas de Seguridad: Pin y Autenticación mutual nivel 0(*no security*).

Lenguaje de programación utilizado: *Applet* –Java Card, *Middleware: C#.*

SmartCardFramework

El SmartCardFramework es un marco de trabajo que facilita la implementación de soluciones con Tarjetas Inteligentes en tecnología .Net fue realizado en la línea de Tarjetas Inteligentes perteneciente al departamento de Componentes del CISED, e implementa un grupo de estándares internacionales por el cual se rigen las aplicaciones de este tipo.

Este marco de trabajo cuenta con un núcleo que funciona como espina dorsal y se le van agregando bibliotecas como:

- **Devices.CardReaders.Win32PCSCWrapper**: encargada de la obtención y pasarela de comunicación con los lectores en el sistema operativo Windows con arquitectura de 32 bits.
- **SmartCard.Devices.CardReaders.PCSCLite**: encargada de la obtención y pasarela de comunicación con los lectores en el sistema operativo Linux.
- **SmartCard.security**: donde se implementa el canal seguro GlobalPlatform y canal seguro ISO 7816 además, se define la seguridad establecida por los estándares antes mencionados.

Este marco de trabajo comprende un grupo de funcionalidades para el trabajo con Tarjetas Inteligentes y en comparación con los que existen gratis en internet abarca mayor cantidad de funcionalidades y de utilidad para las aplicaciones de este tipo.

Las soluciones anteriormente fundamentadas por su forma tradicional de interactuar con las tarjetas traen consigo algunas restricciones como son: el dominio que deben tener los usuarios para efectuar las actualizaciones en la tarjeta, así como la necesidad de poseer una serie de permisos en el manejo de los recursos de la computadora para poder instalarlas. Además, las llaves simétricas necesarias para trabajar con las aplicaciones que se ejecutan dentro de la tarjeta, se encuentran en el terminal cliente, es importante mencionar que las medidas de seguridad utilizadas en las soluciones realizadas en el departamento de componentes del CISED son variables y no existe una homogeneidad a la hora de la implementación de las aplicaciones.

1.8. CONCLUSIONES DEL ESTADO DEL ARTE

En la revisión bibliográfica realizada a lo largo de esta investigación se evidencia el perfeccionamiento y evolución de las aplicaciones de Tarjetas Inteligentes, cómo a través de un modelo de desarrollo basado en *applet-middleware* existen una gran cantidad de aplicaciones que brindan servicios a los usuarios finales con alto nivel de seguridad. La solución internacional *Electronic Driving License* y las soluciones realizadas en el

departamento de Componentes del CISED permitieron el acercamiento a los diferentes mecanismos de seguridad empleados en estos tipos de aplicaciones así como la detección de limitaciones de este modelo de desarrollo en la *web*.

Con la tendencia de las entidades y empresas de brindar servicios en línea se referencian soluciones específicas utilizadas internacionalmente como son la **Kalysis.mei** y **Coesys eGov 2.0**, estas soluciones adaptan y modifican el modelo de desarrollo tradicional a las características de la *web* agregándole componentes necesario para que las soluciones específicas cumplan con estándares internacionales y a su vez sean seguras. Alguna de los elementos positivos que aportaron las soluciones estudiadas son:

- Se establece la comunicación en línea entre el *middleware* y el *applet* asegurando los procesos de autenticación mutua, confidencialidad e integridad de los datos.
- Implementación de mecanismos de seguridad que proporcionan un ambiente seguro en el envío y recepción a través de la *web* y entre el *middleware* y el *applet.*
- Se elimina la instalación de tecnologías en la computadora que utiliza el cliente.
- Existen varios servicios que de manera centralizada los brinda un proveedor y el usuario tiene acceso a los mismos a través de la *web*.

Sin embargo también se observan varias limitaciones importantes a considerar:

- Estas soluciones no poseen una vasta documentación y al ser productos propietarios no se tiene acceso al código fuente.
- No se permite la incorporación de nuevos servicios o *middleware* a los desarrolladores externos a la compañía que brindan los servicios.
- Los mecanismos de seguridad utilizados son diversos y no existe una explicación detallada de su funcionamiento interno.

Se demuestra la necesidad de crear un modelo de desarrollo para agrupar de manera centralizada servicios que utilicen Tarjetas Inteligentes y así manejar en un ambiente confiable las llaves necesarias para la comunicación segura y proporcionarle a los desarrolladores una guía probada de componentes que no deben faltar para el desarrollo de este tipo de aplicación.

1.9. CONCLUSIONES DEL CAPÍTULO

- Este capítulo aborda un grupo de conceptos generales que permiten el entendimiento del problema asociado a la investigación.
- La caracterización de estándares, especificaciones y esquema de seguridad para Tarjetas Inteligentes, sirvió de base para el desarrollo del modelo de desarrollo de servicios en línea que utilizan Tarjetas Inteligentes.
- Con la revisión de soluciones como EDL MediCard, Marco de Autenticación, y el estudio de las soluciones internacionales que más han aportado a esta investigación como son: SConect de Coesys Egov 2.0 y Kalysis.mei se evidenciaron un grupo de conceptos como: *middleware*, *applet* y el componente que gestiona la comunicaciones entre ellos, necesarios para el diseño e implementación de una solución que brinde servicios de Tarjetas Inteligentes.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

Este trabajo presenta un modelo para el desarrollo de servicios en línea que utilicen Tarjetas Inteligentes con el objetivo fundamental de aumentar la seguridad y el aprovechamiento de la portabilidad que brindan las mismas, a lo largo del trabajo se describen las premisas fundamentales las cuales anteceden al modelo y los elementos que intervienen en el desarrollo tradicional de aplicaciones, los componentes arquitectónicamente significativos implicados, el flujo de comunicación entre ellos y los mecanismos de seguridad utilizados.

2.1. PRINCIPIOS PARA EL MODELO DE DESARROLLO DE SERVICIOS EN LÍNEA UTILIZANDO TARJETAS INTELIGENTES.

Partiendo del modelo de desarrollo tradicional de las tarjetas y las perspectivas que tienen los desarrolladores para el trabajo con Tarjetas Inteligentes mencionadas en el epígrafe anterior. Además, con el apoyo de los estándares de comunicación ISO/IEC 7816, PC/SC y con la pretensión de darle solución a la problemática planteada se presenta en la figura 5, una vista general del modelo de desarrollo de servicios en línea utilizando Tarjetas Inteligentes, representando los componentes del modelo tradicional que apoyan el desarrollo de este nuevo modelo que se propone.

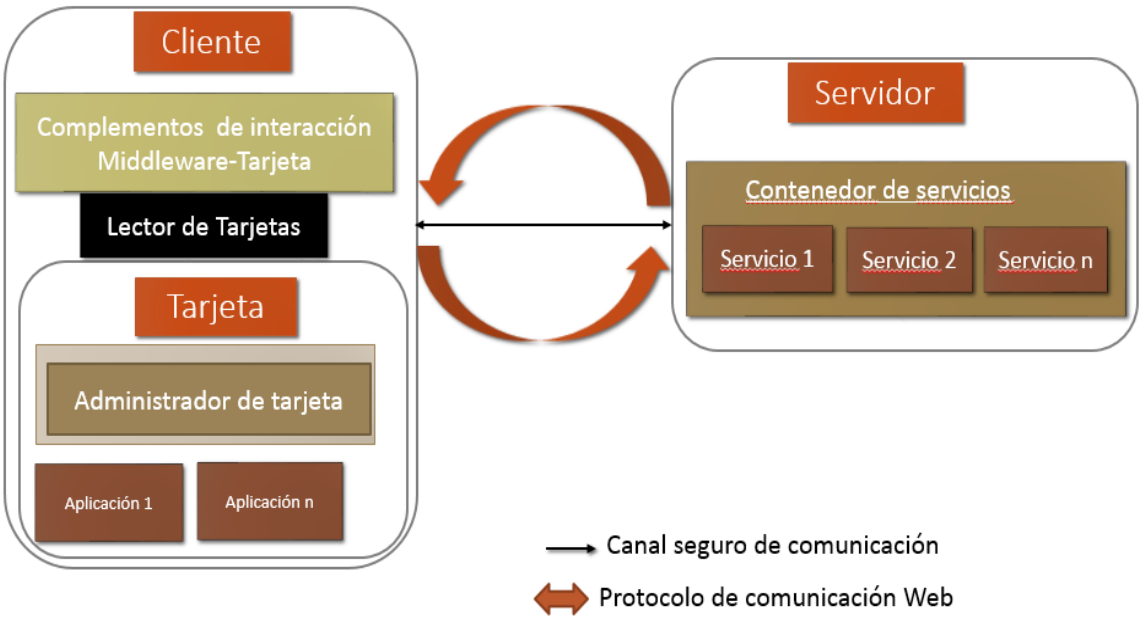


Figura 5: Vista general del modelo de desarrollo de servicios utilizando Tarjetas Inteligentes.

En el servidor se ejecutarán los *middlewares* que para este modelo tendrán el nombre de servicios y cada uno será el responsable de la lógica de negocio y la comunicación con las aplicaciones que se ejecutan en la tarjeta. Los servicios estarán alojados en un

contenedor o componente genérico que se encarga de la comunicación con la parte cliente y los esquemas de seguridad asociados. El cliente manejará el envío y recepción de los mensajes provenientes de los servicios dirigidos a las aplicaciones que se ejecutan en la tarjeta y viceversa, utilizando complementos de interacción que apoyado por el contenedor de servicios cumplirán con los principios que regirán el modelo. Los principios definidos para la propuesta del modelo para el desarrollo de servicios en línea utilizando Tarjetas Inteligentes, son los siguientes:

1. Comunicación síncrona.

2. La incorporación de un nuevo servicio no afectará a aquellos existentes.

3. La comunicación entre los servicios y las aplicaciones embebidas en la tarjeta deben implementar estándares de seguridad existentes.

Teniendo en cuenta los principios definidos y la propuesta general del modelo de desarrollo, se presentan los componentes y sus relaciones como parte de dos ambientes diferentes: el servidor donde estarán alojados los servicios y el terminal donde está insertada la Tarjeta Inteligente.

2.2. COMPONENTES DEL MODELO DE DESARROLLO DE SERVICIOS EN LÍNEA UTILIZANDO TARJETAS INTELIGENTES

El modelo que se propone integra nuevos componentes de *software* al modelo de desarrollo tradicional de aplicaciones para Tarjetas Inteligentes, los componentes se van a distribuir en dos ambientes diferentes, el terminal y un servidor de aplicaciones.

En el modelo tradicional de desarrollo de aplicaciones de Tarjetas Inteligentes cuando se hace una aplicación o *applet* se implementa un *middleware* asociado que funge de intermediario y traductor entre las capas superiores y el *applet* en el interior de la tarjeta. En caso de que se necesiten realizar otras aplicaciones sería necesario volver a implementar otros *middleware*s apareciendo prácticas que atentan contra el desarrollo de este tipo de *software*:

- Inseguridad en el manejo de las llaves.
- Diferentes maneras de implementación del *middleware*.

El modelo propuesto desacopla las responsabilidades de los componentes logrando simplicidad y unidad debido a que toma características de marcos de trabajo y buenas prácticas para el desarrollo de servicios utilizando Tarjetas Inteligentes, centrando principalmente los *middlewares* en un servidor donde se manejará la seguridad de los

datos, gestionándose la comunicación con el cliente. La figura 6 muestra una vista gráfica de los componentes que integran el modelo propuesto.

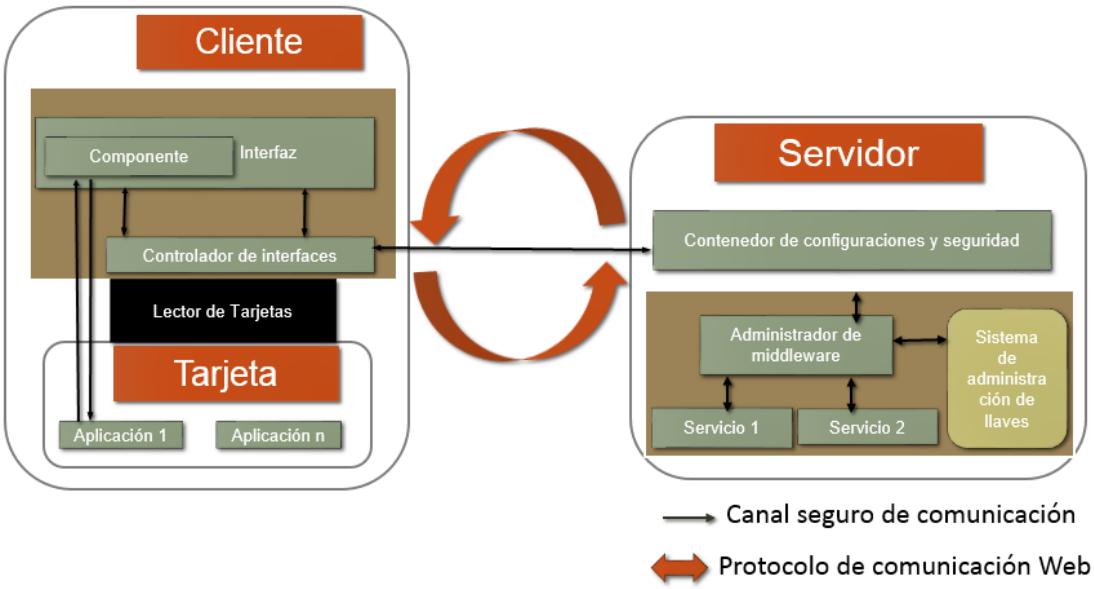


Figura 6: Vista extendida del modelo de desarrollo de servicios utilizando Tarjetas Inteligentes.

Las principales características de los componentes que propone el modelo de desarrollo de servicios utilizando Tarjetas Inteligentes son:

2.2.1. COMPONENTES GENERALES QUE PROPONE EL MODELO

Canal seguro

En la representación de los componentes del modelo en la figura 6 se incorpora el canal seguro GlobalPlatform siendo este, el pilar fundamental para establecer una comunicación segura. La autenticación mutua se logra mediante el proceso de iniciar un canal seguro y garantiza tanto a la tarjeta como a la entidad fuera de esta, la comunicación en un ambiente protegido. Si cualquier paso en el proceso de autenticación mutua falla, el proceso se reanudará. La iniciación de un canal seguro permite a la entidad fuera de la tarjeta de encargar a esta el nivel de seguridad que se requiere para el canal seguro actual (Integridad y/o confidencialidad) y aplicar este nivel de seguridad a todos los mensajes intercambiados posteriores entre la tarjeta y la entidad hasta el final de la sesión del canal seguro.

2.2.2. COMPONENTES QUE PROPONE EL MODELO DEL LADO DEL SERVIDOR

Middleware o servicio

El *middleware* es un componente de conectividad que funciona como una capa de abstracción, situado en el servidor. Se utiliza generalmente para relacionar sistemas que necesitan intercambio de información, permitiendo realizar la conexión a través de interfaces de alto nivel.

Algunas de sus funciones son:

- Transparencia de la heterogeneidad de los componentes de *hardware*, sistemas operativos y protocolos de comunicación.
- Proporciona un estándar de alto nivel de interfaces para los desarrolladores de aplicaciones, contribuyendo a su reutilización, adaptabilidad e interoperabilidad.
- Suministra un conjunto de servicios comunes a diversas funciones de propósito general, a fin de evitar la duplicación de esfuerzos y facilitar la colaboración entre las aplicaciones.
- Oculta los detalles de la programación de bajo nivel. Actualmente es muy utilizado para interactuar con las Tarjetas Inteligentes en las computadoras personales ya que hacen función de intermediarios entre diversas aplicaciones y los lectores de tarjetas.

En el modelo, el *middleware* desempeña un papel fundamental, este es el encargado de comunicarse con el *applet*, es la vía por donde transitan las instrucciones específicas logrando conmutar los eventos generados por el cliente, dándole una respuesta recibida del intercambio de información con la tarjeta. Además, establece comunicación con el gestor de llaves encargándose de implementar los diferentes mecanismos de comunicación segura propuestos por los estándares ISO/IEC y GlobalPlatform.

Administrador de _middleware_

Es el encargado de la identificación y reconocimiento de manera bidireccional de las peticiones hechas por el cliente y el servidor notificando de manera simple y transparente a su capa superior, en caso de que en el terminal cliente se haya lanzado un determinado evento, este componente se encarga de identificar a quien está dirigido y entregarlo, en caso de que sea el servidor el que envía una petición él lo entregará a una capa superior empaquetándolo en forma de mensaje.

Sistema de administración de llaves

Componente que se encarga del almacenamiento y administración de los objetos criptográficos, la aceleración de operaciones criptográficas, la generación de llaves, el cifrado y firma de datos. Además, se utiliza por su característica de proporcionar un canal seguro en la comunicación con dispositivos criptográficos.

Contenedor de configuraciones y seguridad (Servidor de configuraciones)

Este componente tendrá a su cargo 4 responsabilidades fundamentales para validar cualquier servicio que se rija por el modelo propuesto: los filtros, la seguridad, los validadores, y el registro de *plug-in*.

Los filtros se encargan de permitir o denegar determinada información ejecutándose antes y después de cada notificación de eventos.

Aspectos soportados por los filtros del modelo propuesto son:

1. Enrutamiento: La primera vez que este filtro se activa es para comprobar si el evento entrante tiene oyentes[27] en el servidor, si no, el flujo termina en ese punto. En caso de que compruebe la existencia de un manejador para el evento del lado del servidor, este filtro envía la respuesta a la capa que le sucede.

2. Seguridad: comprueba si el cliente está autenticado y si tiene los permisos necesarios para notificar un evento. En caso contrario, la notificación de eventos no está autorizada. Los controles de seguridad incluyen direcciones IP, funciones de usuario los nombres de usuario y controles, estos últimos, se ejecutan en orden. También existirá seguridad SSL[28] estableciéndose una comunicación segura entre el servidor y el terminal cliente.

3. Validación: Este filtro comprueba si cada argumento entrante/saliente definido coincide con las reglas de validación asociadas a él. El proceso de validación se asegura de que los oyentes del servidor solo reciben eventos con información válida y la parte del lado del cliente siempre recibe la respuesta semánticamente correcta.

4. Registro de *plug-in*: Este filtro se encarga de configurar los metadatos de los *middlewares* creados en el servidor.

[27] Los denominados *listeners* en inglés

[28] *Secure socket layer*

2.2.3. COMPONENTES DEL LADO DEL CLIENTE

Controlador de Interfaces

Este componente se encarga de operar los diferentes mecanismos de respaldo para la comunicación con la Tarjeta Inteligente, agrupando un conjunto de ficheros que son los encargados del manejo, recepción y envío de las peticiones de manera bidireccional, algunas de sus funciones fundamentales son:

1. Manejar las respuestas a las peticiones que se hacen cuando se dispara un evento.
2. Manejar el envío y recepción de notificaciones a/desde el servidor.
3. Generar la información de los *middlewares* o servicios del lado del cliente.
4. Manejar los filtros de validación y seguridad del lado del cliente.

Componente de comunicación con el lector

Permitirá obtener los lectores disponibles conectados a la estación cliente, establecer la comunicación con los lectores de tarjetas, notificar el estado de la conexión con la tarjeta, controlar el intercambio de comandos y respuestas APDU entre el *middleware* por el lado del servidor y la Tarjeta Inteligente en el cliente. Este componente será ejecutado en el navegador del cliente.

Interfaz

Este componente es el encargado de mostrar en interfaces la información solicitada por el usuario o información que el servidor quiera mostrarle. Además, se encarga de manejar los diferentes eventos que puedan ser generados por el usuario.

2.2.4. TARJETA

Se instalará una aplicación con el identificador definido por el estándar y en caso de que la aplicación de la tarjeta no siga un estándar entonces se procederá a determinarse un identificador único.

2.3. FLUJO DE COMUNICACIÓN ENTRE LOS COMPONENTES DEL MODELO

En la figura 7 se muestra un flujo de comunicación entre los componentes del modelo de desarrollo de servicios en línea utilizando Tarjetas Inteligentes Java Card. En este flujo de comunicación se representan a través de mensajes direccionales la interacción entre los componentes del modelo y mediante actividades las responsabilidades fundamentales de los mismos en el proceso de envío de un APDU a la tarjeta. El flujo de comunicación comienza desde que el usuario final inicia sesión en el sistema, en lo

adelante el usuario puede solicitar una determinada operación generando y enviando una petición al servidor o el servidor pudiera notificarle cualquier novedad o actualización.

Como precondiciones para comenzar el flujo que se muestra en la figura 7, el usuario debe tener un lector con tarjeta conectado en el terminal, deben estar instalados los controladores del lector y poseer un navegador *web*.

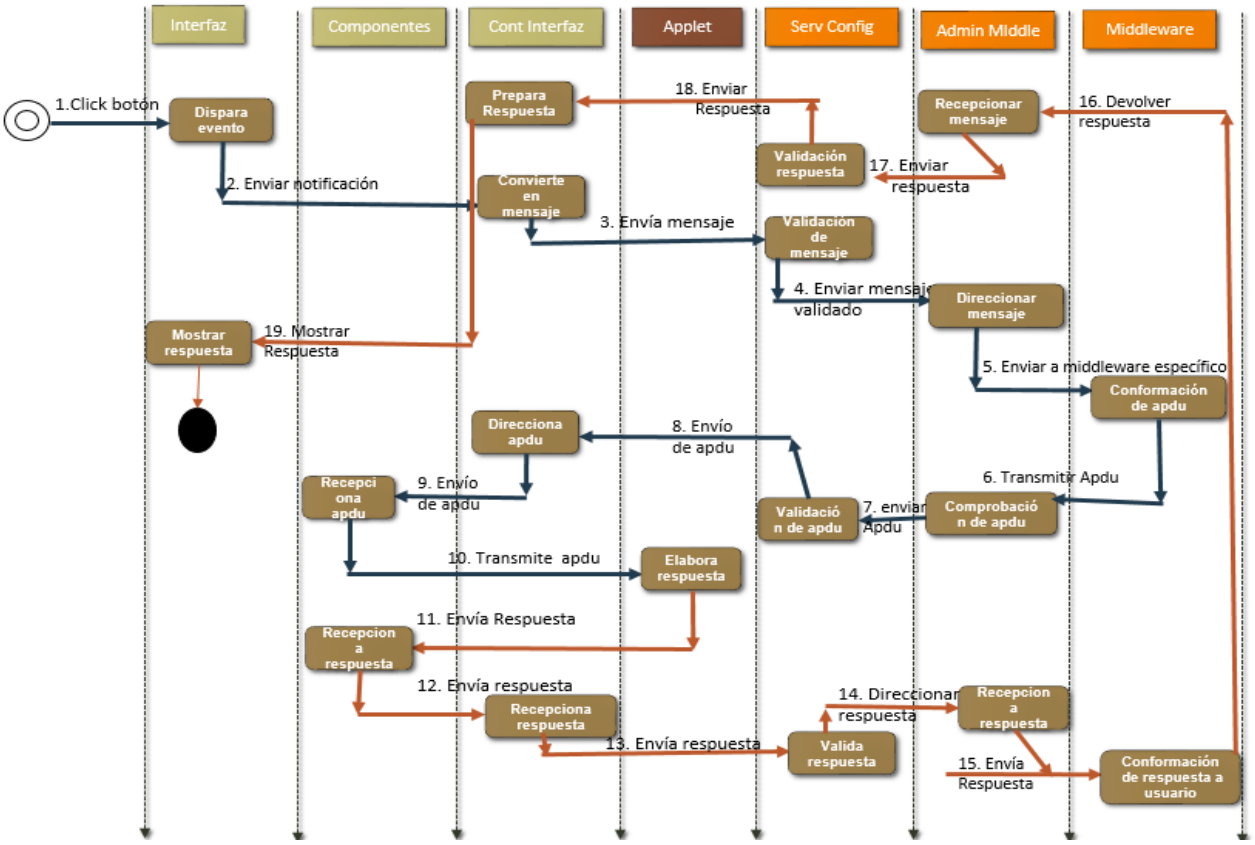


Figura 7: Flujo de comunicación entre los componentes del modelo.

Explicación del flujo de comunicación entre los componentes representado en la figura 7.

1- El usuario solicita la página principal o la URL donde una aplicación expone la portada para la interacción con un servicio que está del lado del servidor, el usuario accede a la interfaz solicitando información que está almacenada en la Tarjeta Inteligente.

2- Al acceder a la interfaz se dispara un evento notificando al controlador las características del evento, si tiene datos asociados y si fueron validados anteriormente. El controlador de interfaz se va a encargar del evento y sus

características, convirtiéndolo en un mensaje para enviar al servidor de configuraciones.

3- El mensaje llega al contenedor de configuraciones en el servidor, allí pasa por los filtros de seguridad y validación necesarios para comprobar los datos enviados del cliente, luego se envía el mensaje al administrador de *middlewares*.
4- El administrador de *middleware* se encarga de direccionarlo hacia el *middleware* específico que tiene el manejador necesario para tramitar la solicitud proveniente del cliente.
5- En el *middleware*, se hacen las operaciones pertinentes, se conforma el APDU y se envía nuevamente al administrador de *middleware*.
6- Este lo retransmite al servidor de configuraciones.
7- Nuevamente pasa por un filtrado, el servidor de configuraciones transmite el comando APDU al controlador de interfaz.
8- Este lo direcciona al componente, este se encarga de transmitir el comando al *applet* contenido en la Tarjeta Inteligente insertada en el lector.
9- El *applet* devuelve una respuesta, esta puede ser información almacenada en la tarjeta o simplemente un error, esta información se reenvía al componente.
10- El componente transmite la respuesta al controlador de interfaz.
11- El controlador de interfaz envía la respuesta al servidor de configuraciones.
12- El servidor de configuraciones filtra nuevamente si la respuesta proviene de un destino confiable luego se envía al administrador de *middleware*.
13- Este envía la respuesta para el *middleware* que se encarga de interpretarla, en caso de que sea un error prepara un mensaje explicando al usuario los motivos del error y si es la información requerida, se traduce a información legible para el usuario y se envía para el administrador de *middleware*.
14- Este envía la respuesta hacia el servidor de configuraciones para que esta sea validad y posteriormente transmitida hacia el controlador de interfaces.
15- El controlador de interfaces envía la respuesta a la interfaz específica.
16- La interfaz específica muestra la respuesta a la solicitud del usuario.

2.4. REQUISITOS PARA LA INTEGRACIÓN DE *MIDDLEWARE*S UTILIZANDO EL MODELO

2.4.1. GENERAL

El modelo propone para implementar una solución guiada que se utilice la programación basada en eventos, la misma, es un paradigma en la que el flujo del programa es determinado por los acontecimientos, las acciones del usuario (*click* del ratón, conectar dispositivo) o mensajes de otros programas o subprocesos. Programación orientada a

eventos se puede definir como una técnica de arquitectura donde la aplicación tiene un bucle principal que se divide en dos secciones: la primera es la selección de eventos (o la detección de eventos) y la segunda es el control de los mismos.

2.4.2. SERVIDOR

Del lado del servidor en la figura 6 se puede observar un componente que se llama servicio, se sugiere que este componente esté divido en 4 directorios que a continuación se muestran detallando su uso:

Carpeta de eventos: donde se almacenarán los diferentes eventos que se lanzarán desde el cliente y se especificará el evento y sus atributos.

Carpeta de comandos: este directorio cumplirá la tarea de almacenar los diferentes comandos APDU, esto se hace para que la aplicación tenga una mayor organización debido a que los comandos APDU cuando la aplicación es poco compleja pueden crearse dentro del propio código, pero no se garantiza la legibilidad del mismo por otros programadores.

Carpeta de la lógica programada: En esta carpeta se organizarán las clases que intervienen en el desarrollo del *middleware*, las clases auxiliares y principales que le darán respaldo y soporte al flujo de eventos generados por el usuario.

Carpeta de *Plug-in*: Este directorio almacenará las extensiones o *plug-in* donde conmutarán los comandos, los eventos y la lógica programada, los *plug-in* son la espina dorsal del *middleware*, los encargados de recibir las peticiones o mensajes de eventos, procesarlos y darle una respuesta apoyado en los demás componentes almacenados en los otros directorios.

El componente que lleva por nombre Servidor de configuraciones y seguridad, este modelo propone que se agreguen allí las configuraciones de la seguridad y los filtros personalizados pertenecientes al *middleware* que se implementa.

Este componente servirá como enlace entre los datos enviados y recibidos del lado del cliente y los datos enviados y recibidos al servidor, allí se recogerán los nombres de los atributos pertenecientes a los eventos y según el tipo de estos, serán validados.

2.4.3. CLIENTE

La parte del cliente perteneciente al *middleware* que está implementado del lado del servidor cuenta con 3 componentes fundamentales.

Controlador de interfaces: Se van a implementar ficheros JavaScript con el propósito de la recepción y envío de información del lado del servidor, estos ficheros generarán los *plug-in* con toda la información del lado del servidor, se encargarán de las notificaciones en ambos sentidos, de la administración de los componentes encargados de la comunicación con la tarjeta y de la muestra de información en las interfaces.

Componente: El modelo propone un componente que se encargue de la comunicación con las Tarjetas Inteligentes, básicamente este componente estará compuesto por las funcionalidades que a continuación se mencionan.

- Transmitir un comando APDU a un terminal teniendo como parámetro un comando APDU codificado en base 64 y esperando como respuesta un APDU Response.
- La lista de nombres de los terminales activos en la PC cliente.
- Las funcionalidades para saber cuándo se conectó y desconectó un determinado lector.

Interfaces: Las interfaces tendrán que ser acorde a las necesidades del *middleware* a las cuales responden, el modelo propuesto no define una manera de hacerlas aunque sugiere que utilicen ExtJS 4.0 debido a que este marco de trabajo JavaScript extiende la librería YUI e integra AJAX, Prototype y Scriptaculous. (Leyva, 2008)

2.5. SEGURIDAD

2.5.1. MECANISMOS DE SEGURIDAD DEL MODELO PARA EL DESARROLLO DE SERVICIOS EN LÍNEA UTILIZANDO TARJETAS INTELIGENTES

La característica fundamental que persigue este modelo es la seguridad en la comunicación entre las diferentes aplicaciones centralizadas en el servidor y los *applet* que se ejecutan en la Tarjeta Inteligente, por tanto, se considera de vital importancia que se definan los conceptos necesarios para garantizar mecanismos de seguridad que protejan la información dentro y fuera de la tarjeta.

En el capítulo anterior se revisaron las arquitecturas de seguridad que proponen los estándares GlobalPlatform e ISO7816-4 y los elementos fundamentales que permiten establecer una comunicación segura entre el terminal y el dispositivo inteligente. El estudio de estos estándares permitió seleccionar como modelo de seguridad el propuesto por GlobalPlatform debido a que la API provista por este estándar permite hacer uso de las funcionalidades, objetos de seguridad y protocolos de canal seguro, correspondientes al Dominio de Seguridad al que pertenece una determinada aplicación

sin la modificación del *applet* que se ejecuta dentro de la tarjeta siempre y cuando el Sistema Operativo de la misma cumpla con las especificaciones del estándar.

Los protocolos de canal seguro de GlobalPlatform seleccionado, aseguran tres niveles de seguridad (GLOBAL PLATFORM, 2003)

- Autenticación mutua: en que la tarjeta y la entidad fuera de esta demuestran que tienen conocimientos sobre un mismo secreto.
- Integridad y autenticación del origen de datos: en la que la tarjeta se asegura de que los datos recibidos de la entidad sean realmente la secuencia correcta y no han sido alterados y puesto en riesgo.
- Confidencialidad: en la que los datos se transmiten desde la entidad fuera de la tarjeta a la tarjeta, no es visible por una entidad no autorizada.

La flexibilidad del modelo que se propone permite a los desarrolladores implementar las medidas de seguridad en sus aplicaciones, sin apoyarse en las que opcionalmente brinda el modelo propuesto, el cual utilizando los componentes: sistema administrador de llaves, administrador de *middleware*, componente y *applet* facilitan la implementación del canal seguro GlobalPlatform (ver figura 8)

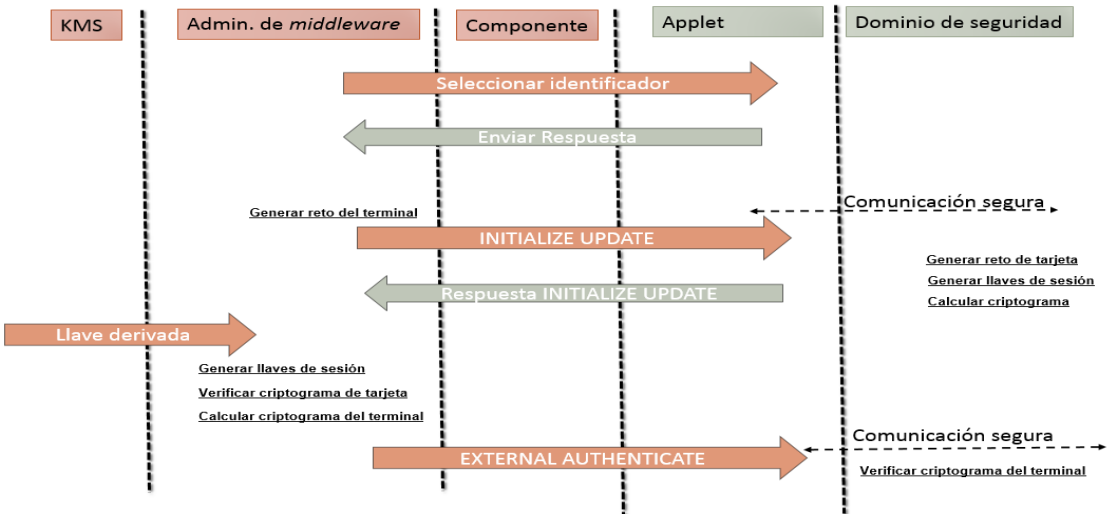


Figura 8: Canal seguro GlobalPlatform y los componentes asociados en el modelo de desarrollo propuesto.

El componente sistema administrador de llaves (KMS), pertenece a un sistema externo el cual brinda servicios relacionados con criptografía y seguridad de las llaves, es el responsable de almacenar de manera segura la llave semilla y de brindarle al administrador de *middleware* las derivaciones que requiera para continuar con el

proceso de autenticación mutua. En la tabla 2 se realiza una descripción del proceso de canal seguro GlobalPlatform entre el Administrador de *middleware* y el *applet.*

Tabla 2: Descripción del proceso de canal seguro GlobalPlatform propuesto por el modelo.

Administrador del *middleware*	***Applet***
-Envía un comando *Select* definido en el estándar GlobalPlatform para seleccionar el *applet* con el que se quiere establecer el canal seguro de comunicación. Para esto es necesario conocer el identificador del *applet*.	-Se selecciona el *applet* en caso de que exista y se envía una respuesta.
-Genera un reto para enviar a la tarjeta (arreglo de bytes de longitud 8 y en su interior números aleatorios). -Envía comando *Initialize Update* donde se define el número de versión del conjunto de llaves que se utilizarán para que se establezca la sesión del canal seguro y envía el reto generado por el terminal.	-Genera un reto para enviar al terminal (arreglo de bytes de longitud 8 y en su interior números aleatorios.) -Genera las llaves de sesión. -Calcula el criptograma de la tarjeta -Envía respuesta del comando con la siguiente información: Datos de diversificación de llave (10 bytes). Información de la llave (número de versión de la llave, identificador del protocolo de canal seguro (en este caso el 01)) (2 bytes). El reto generado por la tarjeta (8 bytes). El criptograma de la tarjeta que es un criptograma de autenticación (8 bytes).
-Genera las llaves estáticas. -Genera las llaves de sesión (SK_ENC y SK_MAC, SK_DEK). -Verifica el criptograma de la tarjeta. -Calcula el criptograma del terminal. -Envía comando *External Authenticate* que es utilizado por la tarjeta para autenticar el terminal y para determinar el nivel de seguridad requerido para todos los comandos posteriores.	-Verifica el criptograma del terminal -Comienza la comunicación segura, dada por el nivel de seguridad que se especificó en el P1 del comando *External Authenticate.*

Terminado el proceso de autenticación mutua se comienza entonces con la mensajería segura entre el administrador de *middleware* y el *applet* definido en el nivel de seguridad establecido en el envío del comando EXTERNAL AUTHENTICATE

La terminación de la sesión del canal seguro sucede cuando se produce una de las siguientes condiciones:

- La aplicación en la tarjeta recibe el primer comando APDU con una protección criptográfica errónea.
- La aplicación en la tarjeta recibe un comando APDU sin el conjunto criptográfico, protección requerida durante el inicio de sesión del canal seguro.

El período de sesiones de la aplicación en la tarjeta se termina, cuando se selecciona otra aplicación (es decir, la sesión de aplicación termina) en el mismo canal lógico. Es responsabilidad de la nueva aplicación proceder a la resolución de la sesión del canal seguro cuando esto ocurre.

El canal lógico asociado se cierra de forma explícita cuando la tarjeta está apagada o se reinicia la potencia (es decir, la sesión de la tarjeta termina). Una sesión de canal seguro es un canal lógico, por tanto, al terminarse por algunas de las razones antes mencionadas solo puede ser iniciado el proceso nuevamente.

2.5.2. WEBOSOCKET SEGURO

El modelo propone la obtención de al menos un certificado SSL[29] para el servidor: usado para identificar a un servidor ante un cliente en comunicaciones mediante el protocolo *Secure Socket Layer*, y se expiden generalmente a nombre de la empresa propietaria del servidor seguro o del servicio que este va a ofrecer, vinculando también el dominio por el que se debe acceder al servidor. La presencia de este certificado es condición imprescindible para establecer comunicaciones seguras SSL utilizando el protocolo websocket.

PROTOCOLOS SECURE SOCKET LAYER

Para establecer una comunicación SSL, independientemente del protocolo de comunicación que se esté usando, es necesario que previamente el cliente y el servidor realicen un proceso de reconocimiento mutuo y de petición de conexión que, al igual que en otros tipos de comunicaciones, recibe el nombre de apretón de manos o *Handshake*, que en este caso está controlado por el Protocolo SSL *Handshake*, que se encarga de establecer, mantener y finalizar las conexiones SSL. Durante el mismo se negocian los parámetros generales de la sesión y los particulares de cada conexión. (Delitosinformaticos, 2012)

El modelo que se desarrolla propone un componente llamado contenedor de configuraciones y seguridad el cual permite como una de sus responsabilidades obtener

[29] *Secure Socket Layer*

y almacenar un certificado SSL para el servidor y proporcionar las configuraciones necesarias para que un desarrollador de manera opcional utilice conexión segura entre cliente y servidor.

2.6. CONCLUSIONES PARCIALES

En este capítulo:

Se presentó la propuesta del modelo de desarrollo de servicios en línea que utilizan Tarjetas Inteligentes definiendo premisas que le permitirán a los desarrolladores tener una visión acerca de las pautas a seguir para lograr una solución segura y escalable.

La presentación de los componentes y sus relaciones demostró que el modelo propuesto generaliza e innova conceptos y categorías pertenecientes al modelo de desarrollo tradicional adaptándolas a un entorno *web.*

El modelo proporciona mecanismos de seguridad basados en los protocolos de autenticación mutua de GlobalPlatform permitiendo integridad y confidencialidad entre las aplicaciones que se ejecutan en la tarjeta y los servicios centralizados en el servidor y SSL que proporcionan un ambiente seguro entre cliente y servidor.

CAPÍTULO 3: Plataforma de servicios en línea utilizando Tarjetas Inteligentes (SmarCoP).

En este capítulo se describe el ambiente de desarrollo seleccionado, los requerimientos definidos para la plataforma, la arquitectura de *software* general así como los componentes guiados por el modelo propuesto en el capítulo 2 y los elementos para su implementación y despliegue, se propone una guía de cómo agregar un *plug-in* a la plataforma y los resultados de las pruebas realizadas.

3.1. AMBIENTE DE DESARROLLO

SmartCoP (*SmartCard Online Platform*) es la plataforma desarrollada siguiendo el modelo expuesto en el capítulo 2, permite agrupar aplicaciones que utilicen Tarjetas Inteligentes, está pensada para integrarse a los sitios *web* que brinden servicios a través de internet y que desee sustentar su seguridad y/o funcionalidad en el uso de estos dispositivos. Además, permitirá la escalabilidad de nuevas extensiones y la actualización e incorporación de funcionalidades.

Para el desarrollo de SmartCoP se tuvieron en cuenta un grupo de herramientas, estándares y metodología.

3.1.1. METODOLOGÍA

Después de realizar un estudio sobre las metodologías ágiles, pesadas e híbridas buscando la más adecuada y guiados por la experiencia adquirida en desarrollos similares llevados a cabo en el centro, se decidió que la metodología XP sería la que guiaría la implementación de la plataforma, entre los aspectos esenciales que incidieron en la elección de esta metodología se encuentran:

- Necesidad de obtener un producto a corto plazo.
- Cantidad de miembros del equipo de desarrollo (dos personas).
- Presencia del cliente en el grupo de desarrollo. (Fowler, 2000)
- Asumir una metodología robusta implica una cantidad excesiva de roles y gran volumen de información generada durante todo el ciclo de vida del proyecto. Es por ello que se hace difícil la utilización de este tipo de metodología por un equipo pequeño.

3.1.2. MODELACIÓN VISUAL

Como lenguaje de modelado se escogió el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language). Es el lenguaje de modelado de sistemas de *software* más conocido y utilizado en la actualidad y como herramienta de

modelación el equipo se decantó por el Altova UModel ya que es una herramienta que diseña visualmente modelos de aplicaciones en UML y genera código fuente en los lenguajes Java, C#, o Visual Basic .NET y documentación del proyecto. Algunas características de UModel para el desarrollo de *software* basado en las capacidades de modelado avanzado son: (Altova Inc, 2010)

- Soporte para los 14 tipos de diagramas UML.
- Modelado de esquemas XML en diagramas UML.
- Generación de código fuente en lenguajes Java, C#, y VB.NET.
- Ingeniería inversa de código fuente y ficheros binarios en lenguajes Java, C# y VB.NET.
- Crea diagramas de secuencia desde el código fuente de la ingeniería inversa.
- Generación de documentación personalizable de proyecto.

3.1.3. MARCO DE TRABAJO

Se revisaron un grupo de marcos de trabajo que ayudaron en la definición de un arquetipo de la solución que se perseguía, dentro de los revisados el de mayor impacto, y nivel de reutilización, según las opiniones de los especialistas de más experiencia en el campo de los dispositivos inteligentes y encargados del desarrollo de la plataforma de servicios en línea utilizando Tarjetas Inteligentes, pertenecientes al departamento de Componentes del Centro de Identificación y Seguridad Digital, fue el JWebSocket.

Alguna de las principales ventajas del marco de trabajo:

- El proyecto jWebSocket utiliza los beneficios de Jetty (Eclipse, 2009), Netty (Netti, 2010) y Grizzly (Grizzly, 2010), integrándolos como un núcleo y reutilizando la gestión del protocolo, manejo de las conexiones, la transmisión de mensajes y el consumo de memoria. jWebSocket se ratifica como un servidor extensible, altamente configurable y robusto. (Masó, 2012)
- Java es el lenguaje utilizado para programar las aplicaciones en el lado del servidor con el marco de trabajo jWebSocket.
- Tiene soporte para la *web* utilizando características de *framework* estables y potentes como Spring MVC.
- El equipo de desarrollo que labora en el marco de trabajo JWebSocket pertenecía a la Facultad Regional Mártires de Artemisa, el departamento de Tarjetas Inteligentes se incorporó al trabajo con la comunidad JWebSocket retroalimentándose de las nuevas actualizaciones y apoyándolos en cuestiones técnicas sobre el tema de Tarjetas Inteligentes.

Desventajas

- La curva de aprendizaje del marco de trabajo jWebSocket es bastante elevada ya que no existe todavía una documentación extensa.
- Al ser un marco de trabajo joven y una comunidad joven a menudo el equipo de desarrollo se encontraba con problemas no resueltos por la comunidad y se tenían que dar soluciones temporales que no pertenecían al marco de trabajo esperando por su resolución en el futuro.

3.1.4. LENGUAJES DE PROGRAMACIÓN, HERRAMIENTAS Y ESTÁNDARES

Del lado del servidor se eligió como lenguaje de desarrollo Java debido a que es robusto, maduro y escogido para la implementación del marco de trabajo JWebSocket, a la vez este lenguaje ofrece la biblioteca *java.smartcardio*, la cual establece una API para la comunicación con las Tarjetas Inteligentes, usando la norma ISO/IEC 7816-4. Esta biblioteca permite que las aplicaciones Java puedan interactuar con las aplicaciones que se ejecutan en el interior de la Tarjeta Inteligente, para almacenar y recuperar datos de la misma. (Oracle, 2013) En el desarrollo de la solución se utilizan de esta biblioteca las clases *ComandAPDU* y *ResponseAPDU*, para el envío y respuestas de los comandos con una estructura definida por el estándar ISO/IEC 7816-4.

Del lado del cliente se eligió como lenguaje de desarrollo *ExtJS,* una biblioteca de JavaScript para el desarrollo de aplicaciones *web* interactivas usando tecnologías como *AJAX*[30]*, DHTML*[31] y *DOM*[32]*.* Ofrece interfaces de usuarios funcionales, fáciles de usar y muy parecidas a las conocidas aplicaciones de escritorio. (Groner, 2011)

Se escogió para el desarrollo en la parte cliente debido a que:

- El cliente exige que se utilice ExtJS en sus interfaces.
- Existe un conocimiento avanzado por parte de los desarrolladores encargados de la implementación de la plataforma.
- Existe un balance entre Cliente–Servidor: la carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo. (Stuart Ashworth, 2012)

[30] Acrónimo de *Asynchronous JavaScript And XML*, en español JavaScript asíncrono y XML.
[31] DHTML, siglas de *Dynamic HyperText Markup Language* (Lenguaje de Marcado HiperTextual Dinámico)
[32] *Document Object Model,* según sus siglas en español Modelo de Objetos del Documento o Modelo en Objetos para la Representación de Documentos.

- Comunicación asíncrona: ExtJS puede comunicarse con el servidor sin necesidad de estar sujeto a una acción del usuario, dándole la libertad de cargar información que no ocurre en completa correspondencia temporal con otras peticiones que se estén ejecutando paralelamente. (Stuart Ashworth, 2012)

Como estándares fundamentales utilizados en la implementación de la plataforma:

GlobalPlatform para el establecimiento del canal seguro, ISO 7816-4 para la definición de los comandos APDU y las estructuras de ficheros de la tarjeta y PC/SC para la comunicación del lector con la tarjeta.

El entorno de desarrollo seleccionado para desarrollar la plataforma fue Netbeans 8.0 para el trabajo con el lenguaje Java en la parte del servidor, se utilizó además el módulo de desarrollo de html5 que posee este entorno para el trabajo con ExtJS y la plataforma integrada de Java Card 3.0 para el trabajo con Tarjetas Inteligentes.

3.2. REQUISITOS DE LA PLATAFORMA

Se definieron los requerimientos funcionales a tener en cuenta para el diseño e implementación de cada uno de los componentes de la plataforma.

3.2.1. REQUISITOS FUNCIONALES

Se definieron los requerimientos funcionales a tener en cuenta para el diseño e implementación de cada uno de los componentes de la plataforma. Como la metodología que se sigue es XP estos requerimientos se detallan en historias de usuarios, artefacto que utiliza la metodología para describir los requisitos de una forma menos detallada y desde la perspectiva del cliente. En el anexo 1 se listan los requisitos funcionales, para ver su descripción detallada consultar el documento: 0113_Especificación de Requisitos de *Software* perteneciente al expediente del Proyecto: Plataforma para el desarrollo de servicios en línea utilizando Tarjetas Inteligentes.

3.2.2. REQUISITOS NO FUNCIONALES

Un requisito no funcional es una característica requerida del sistema, del proceso de desarrollo, del servicio prestado o de cualquier otro aspecto del desarrollo, que señala una restricción del mismo. Constituyen todas las exigencias de cualidades que se imponen al proyecto, ya sean exigencias de usar un cierto lenguaje de programación o una plataforma tecnológica específica. (SOMMERVILLE, 2005)

En el anexo 2 se muestra el listado de requisitos no funcionales.

3.3. ARQUITECTURA DE LA PLATAFORMA

La figura 9 muestra la arquitectura del sistema, la cual es de tipo cliente-servidor y está estructurada por componentes ya que cada uno debe describir de forma completa las interfaces que ofrece, así como las interfaces que requiere para su operación y su correcto funcionamiento con independencia de los mecanismos internos que utilice para soportar la funcionalidad de la interfaz.

Cliente-servidor: se refiere formalmente a un modelo lógico que proporciona una división de tareas dentro de las capas (o niveles) “cliente” y “servidor” (E. U. Informática en Segovia, 2013), la utilización de este tipo de arquitectura posibilita, independencia, encapsulación y modularidad:

Independencia: se mantiene una clara separación entre la lógica de negocio, la presentación y el acceso a datos. Permitiendo flexibilidad y facilidad a la hora de realizar posibles modificaciones.

Encapsulación de servicios: en el caso de la plataforma SmartCoP el servidor es un especialista, cuando se le entrega un mensaje solicitando un servicio, él determina cómo realizar el trabajo. Los servicios se pueden actualizar sin afectar a los clientes en tanto que la interfaz pública de mensajes que se utiliza por ambos lados, permanece sin cambios. (E. U. Informática en Segovia, 2013)

Modularidad, diseño extensible: el diseño modular de SmartCoP permite la tolerancia a fallos amortiguando su ocurrencia sin causar la interrupción de los demás servicios.

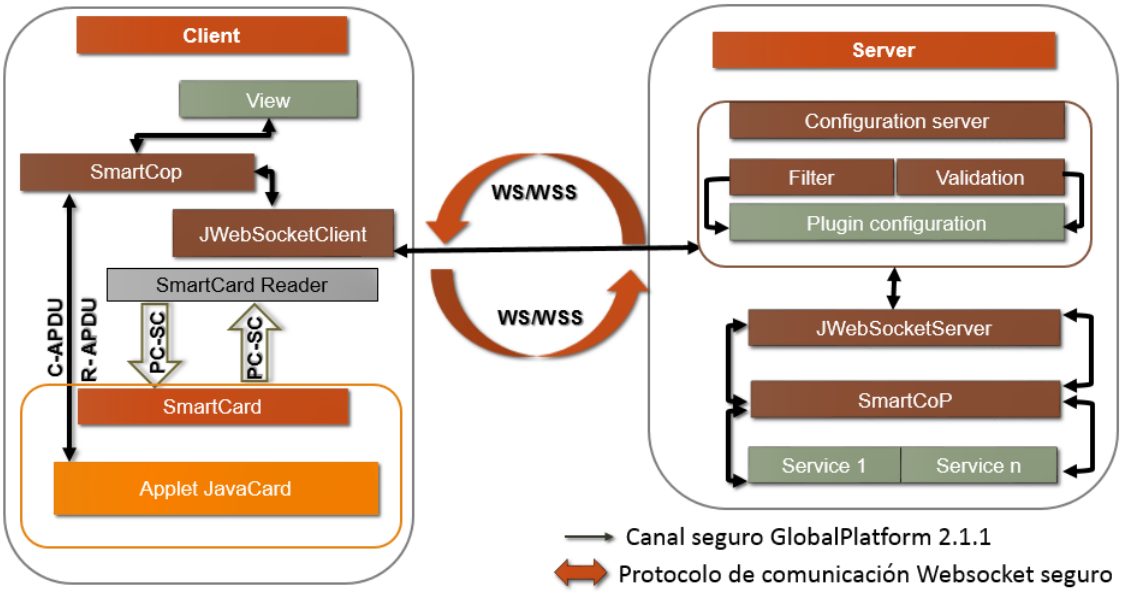


Figura 9: Arquitectura de la plataforma SmartCoP.

3.4. DIAGRAMA DE COMPONENTES

Una representación de los componentes de la solución permite identificar los diferentes elementos del diseño del sistema y sus relaciones. Además de visualizar con más facilidad su estructura general y el comportamiento que proporcionan los diferentes componentes.

Se detallarán cada uno de los componentes visualizados en la figura 10, se dividirá para su análisis en 3 partes fundamentales: la del servidor, la del cliente y la de la tarjeta.

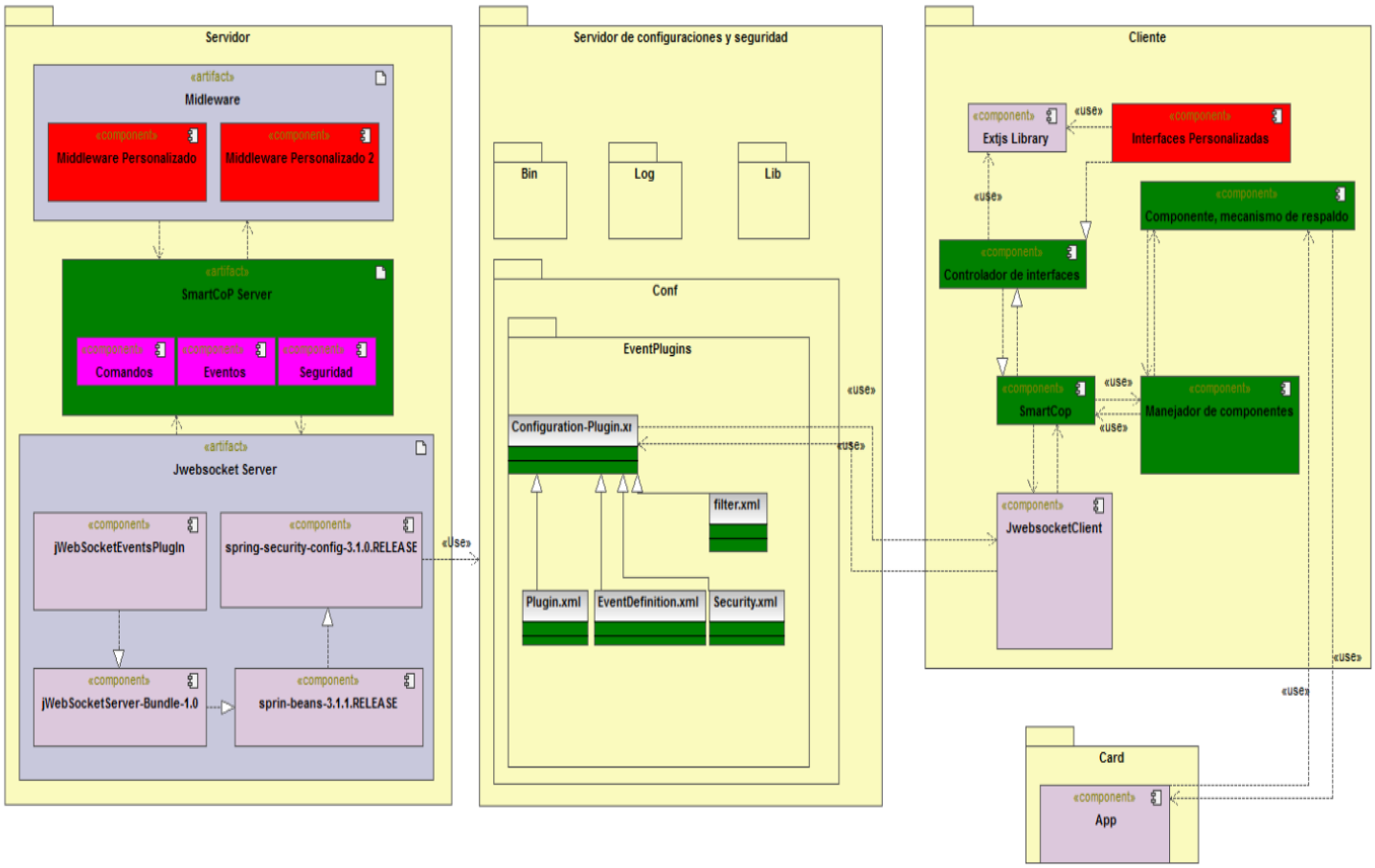


Figura 10: Diagrama de componentes.

3.4.1. SERVIDOR

El artefacto ***middleware*** contiene un grupo de extensiones o *plug-in* que va a aumentar el valor agregado de la plataforma, cada extensión será la implementación de un *middleware* regido por estándares internacionales o de creación propia del autor y guiado su desarrollo por el modelo que se describió en el capítulo 2. La característica fundamental de este componente es que permitirá al desarrollador a través de interfaces de programación centrar la implementación en las características específicas de su servicio, abstrayéndose de temas como el envío/recepción de información y la seguridad de la comunicación con el terminal cliente. Este componente con mínimos cambios puede ser utilizado de manera independiente a la plataforma ya que contiene las implementaciones de diferentes servicios.

El **servidor SmartCoP** brindará un grupo de componentes comunes que tienen que ver con el manejo de los diferentes elementos que rigen un servicio:

Comandos, este componente contendrá un administrador de comandos y los comandos fundamentales regidos por GlobalPlatform como: *Initialize Update, External Authenticate* y *Select* entre otros. Este componente es completamente reutilizable ya

que se encarga de la administración de comandos a través de bibliotecas de Java cumpliendo con el formato que describe el estándar ISO 7816-4.

Evento, este componente es desarrollado particularmente para la plataforma SmartCoP y es el encargado de la recepción y atención de las peticiones que arriban del cliente, los eventos asociados están en contacto directo con los comandos debido a que por cada evento que sea capturado del lado del servidor puede que exista envío y recepción de información a la tarjeta, ellos enviarán/recibirán información privilegiada como la cantidad de datos que necesita una funcionalidad para su correcta ejecución o el inicio y fin de una sesión abierta. Algunos de los eventos que facilitan el trabajo a los desarrolladores de la plataforma son: TerminalReady y TerminalNotReady eventos encargados de ejecutarse cuando el lector de tarjeta está conectado y desconectado respectivamente.

Seguridad, este componente será el encargado de manejar la seguridad de la comunicación cliente-servidor con la incorporación de manera opcional para los desarrolladores de una capa SSL y asegurar el proceso de mensajería segura con la propuesta de una implementación de canal seguro GlobalPlatform, utilizando los 3 niveles de seguridad y protegiendo la llave madre (*Key Seed*) a través de diferentes mecanismos de protección como es el uso de un KMS o un fichero cifrado. Este componente es reutilizable con algunas modificaciones de configuración.

Servidor de JWebSocket, este artefacto contiene un grupo de componentes que pertenecen al marco de trabajo JWebSocket. (jwebsocket, 2014) Y que se reutilizaron en el desarrollo de la plataforma.

El **EventsPlugin,** está diseñado de acuerdo con el paradigma de programación orientada a eventos, en el servidor y en el cliente, su intención es el apoyo a eventos bidireccionales y que las notificaciones sean lo más transparente y simple posible. Cuando el EventsPlugin recibe un evento del cliente, una nueva instancia de evento se genera según el tipo de mensaje. (Rolando Santamaría Masó, 2010)

Servidor jWebSocket, es el servidor de configuraciones y marco de trabajo que implementa el protocolo *web*socket y es la piedra angular sobre la que se sostienen todos los *plug-in*.

Librerías de Spring, aportan al servidor de jWebSocket:

Inversión de Control (IoC): el objetivo del contenedor IoC es encargarse de instanciar los objetos del sistema, denominados *beans*, y asignarle dependencias. Para que el

contenedor pueda llevar a cabo esta tarea, se debe, mediante información de configuración, indicarle dónde se encuentran los *beans.* (Epidataconsulting, 2010)

Seguridad: Spring Security proporciona servicios integrales de seguridad para aplicaciones de *software* empresarial.

Validación: Spring desarrolla una interfaz de validación que es a la vez básica y eminentemente utilizable en todas las capas de una aplicación. (Rolando Santamaría Masó, 2010)

Servidor de configuraciones y seguridad, será el encargado de realizar las validaciones que propone el marco de trabajo Spring asociadas al marco de trabajo JWebSocket, se harán validaciones sobre los *plug-in* que se adicionan a la plataforma, se definirán filtros sobre la seguridad y los eventos. Una parte de este componente estará personalizada por el desarrollador con las configuraciones requeridas por el servicio que se esté implementando.

3.4.2. CLIENTE

JWebSocketclient, Es el encargado de la comunicación con el servidor de jWebSocket y el intercambio de información entre ellos es a nivel de mensaje, informando o notificando al servidor cualquier mensaje proveniente del cliente. Es un componente que brinda la solución desarrollada y que puede ser reutilizado por otras aplicaciones adaptándolo a sus especificaciones, contiene los ficheros jWebSocketClientSide que permite la generación y transacción de información de manera bidireccional y JwsClientCache que se encarga del almacenamiento en caché de la información.

SmartCoP, se encarga de centralizar las peticiones y direccionarlas para el componente que está destinada, a través de SmartCoP se transmite la información que se va a mostrar en la interfaz y a la vez, la que se va a enviar a la Tarjeta Inteligente además, se encargará de validar la información proveniente del lado del servidor así como la clasificación de las peticiones y conversión de un evento a mensaje. Este es un componente específico desarrollado para la plataforma.

Manejador de componentes, este componente está conformado por un grupo de ficheros JavaScript que se encargarán del manejo del *applet* y los mecanismos de respaldo que utiliza la plataforma en el navegador para la comunicación con el lector instalado en el terminal.

El fichero JcManager.js maneja el *applet* Java que está alojado en el navegador, esta tecnología tiene la ventaja que funciona con todos los navegadores pero tiene la

desventaja que necesita para su ejecución la instalación en el cliente de la máquina virtual de Java por eso los desarrolladores decidieron implementar los mecanismos de respaldo y manejadores en caso del ActiveX para Internet Explorer y su fichero manejador OnLineSmartCardPlatform.js y un Add-on para Mozilla manejado por el fichero olsmcp_Interfac.js. Este componente es reutilizable ya que cualquier solución que necesite comunicación con un dispositivo inteligente a través de la *web* pudiera valerse del *applet* Java implementado o de cualquiera de los mecanismos de respaldo propuestos.

El **controlador de interfaces** implementa un patrón observador que está al tanto de cualquier evento que se haya ejecutado en su entorno, gestionando las transacciones efectuadas por el usuario y las validaciones JavaScript de los datos introducidos.

El componente utiliza las librerías de EXTJS 4.1 para la visualización de la página *web* y la muestra de las interfaces personalizadas en dependencia de la extensión del lado del servidor que se esté implementando. Las interfaces van a estar a la espera de un evento para desencadenar una acción o de cualquier notificación del lado del servidor con información que se quiera mostrar.

Componente *applet* y los componentes de respaldo, envían y reciben información de la Tarjeta Inteligente conectada al terminal además, notifican cuando un lector de tarjetas se conecta o se desconecta. Son de independiente uso aplicables a cualquier solución que necesite la comunicación en línea con Tarjetas Inteligentes.

3.4.3. TARJETA

El componente contenido dentro de la tarjeta será el *applet* Java Card, su implementación puede ser responsabilidad de los desarrolladores del servicio o de un equipo de desarrollo externo que proporcione la documentación detallada de su estructura. En la actualidad existen *applets* Java Card que cumplen con estándares internacionales, gratuitos y su código fuente, de libre distribución en la red, por tanto, en muchos casos al obtener un *applet* Java Card con estas características solo queda la revisión a fondo para constatar de que cumple con los requerimientos exigidos por el cliente y desarrollar el servicio para establecer la comunicación con el mismo.

3.5. DESPLIEGUE DE LA PLATAFORMA SMARTCOP

El diagrama de despliegue muestra la disposición física de los distintos nodos que componen un sistema y se indican cuáles son los enlaces de comunicación existentes entre los componentes en tiempo de ejecución. En la figura 11 se muestra el diagrama de despliegue de la aplicación (ver anexo 3).

El sistema debe funcionar en cualquier estación de trabajo que cumpla con los requisitos de *hardware* que se mencionaron en el subtópico 3.2. A través de estas estaciones de trabajo, el cliente puede acceder al portal de SmartCoP mediante internet o una red local, esto le va a permitir solicitar los permisos necesarios para el desarrollo de un nuevo servicio o la utilización de un servicio que brinde la plataforma. Posteriormente estas estaciones de trabajo estarán conectadas a un servidor de aplicaciones donde se ejecutan los servicios implementados por los desarrolladores. Las estaciones de trabajo cuentan con un lector de tarjetas conectado que cumpla con el estándar PC/SC. Ambos elementos se comunican mediante comandos APDU. La conexión entre las estaciones de trabajo (Estación de trabajo del cliente) y el servidor de aplicaciones (PC Servidor) se realizarán mediante el protocolo *web*socket seguro (WSS).

3.6. PRUEBAS REALIZADAS

Luego de ser generado el código fuente, es necesario probar el *software* para corregir la mayor cantidad de errores posibles. Para poder lograr esto, hay que tener en cuenta el desarrollo de una serie de casos de pruebas que tengan una alta probabilidad de encontrar errores en el sistema. Un programa que no es probado con anterioridad antes de ser utilizado en un entorno real, trae consigo un mal funcionamiento e inestabilidad, lo que puede conllevar a que el sistema colapse. El único instrumento adecuado para determinar el *status* de la calidad de un producto de *software* es el proceso de pruebas. (Pruebasdesoftware, 2005)

Con el objetivo de evaluar la calidad de la plataforma se realizó un proceso de pruebas exhaustivo. Para ello la metodología XP determina aplicar pruebas unitarias y pruebas de aceptación. Esto no significa que a la plataforma no se le puedan aplicar otros tipos de pruebas, que el equipo de desarrollo estime que sean necesarias.

La práctica que se utilizó para desarrollar las pruebas unitarias fue el Desarrollo Dirigido por Pruebas (TDD): es una técnica de programación que plantea escribir primero los casos de prueba y luego implementar lo necesario para ejecutarla. Antes de escribir cualquier fragmento de código, se debe escribir las pruebas automatizadas para comprobar la funcionalidad de ese futuro código. Como el código no existe, inicialmente la prueba falla. Una vez comenzadas las pruebas, se debe eliminar el código duplicado. (Programacion, 2010)

Con ayuda de la herramienta Junit para la automatización de los casos de pruebas, las funcionalidades de la plataforma fueron correctamente desarrolladas y probadas en el nivel de pruebas de unidad.

Capítulo 3: Aplicación demostrativa

A medida que fue avanzando el desarrollo de la plataforma comenzaron a integrarse los diferentes componentes utilizando la Integración Ascendente: La prueba de integración ascendente, comenzando con la construcción y la prueba con los módulos atómicos.

Dado que estos se integran de abajo hacia arriba, el proceso requerido de los módulos subordinados siempre está disponible y se elimina la necesidad de resguardos.

Pasos que se siguieron para implementar una estrategia de integración ascendente:

1- Se combinaron los módulos de bajo nivel en grupos que realicen una sub-función específica de la plataforma.
2- Se escribió un controlador para coordinar la entrada y la salida de los casos de prueba.
3- Se prueba el grupo.
4- Se eliminaron los controladores y se combinan los grupos moviéndose hacia arriba por la estructura del programa. (Pressman, 2008)

```
2014-06-24 16:08:08,220 DEBUG - TCPEngine: Starting TCP engine 'tcp0' at port 8787 with default timeout infinite...
2014-06-24 16:08:08,224 INFO  - TCPEngine: TCP engine 'tcp0' started' at port 8787 with default timeout infinite...
2014-06-24 16:08:08,225 DEBUG - TCPEngine: Starting SSL engine 'tcp0' at port 9797, with default timeout infinite...
2014-06-24 16:08:08,225 DEBUG - TokenServer: Processing engine 'tcp0' started...
2014-06-24 16:08:08,226 DEBUG - BasePlugInChain: Notifying plug-ins of server 'ts0' that engine 'tcp0' started...
2014-06-24 16:08:08,227 DEBUG - FlashBridgePlugIn: Engine 'tcp0' started.
2014-06-24 16:08:08,227 DEBUG - ChannelPlugIn: Engine started, starting channels...
2014-06-24 16:08:08,253 INFO  - ChannelPlugIn: 6 channels started.
```

Figura 11: Integración de componentes en el servidor.

La ilustración 12 muestra un fragmento de las pruebas de integración ascendente y el tiempo necesario para lograr interrelación entre los distintos componentes, las pruebas se realizaron en la herramienta de desarrollo Netbeans 8.0.

Luego de implementar algunos componentes del lado del servidor y del cliente e integrarlos con los del marco de trabajo JWebSocket se pasó al siguiente nivel de pruebas.

Para las pruebas del sistema se utilizó la plataforma SmartCoP en su versión 1.0 y una extensión implementada para la plataforma llamada “Extensión para la lectura de pasaportes electrónicos”. (Ge, 2012)

Prueba de Recuperación de datos: Es una prueba del sistema que fuerza el fallo del *software* de disímiles maneras y verifica que la recuperación se lleva a cabo apropiadamente, en los dispositivos inteligentes esta prueba se enfoca directamente en

la no violación o recuperación de información cuando sucede una situación anómala en la transmisión de información.

El *software* abre una sesión para establecer una comunicación segura con la tarjeta que está insertada en el lector del lado del cliente, cuando comienza el proceso de lectura de pasaporte se empiezan a enviar comandos APDU entre el *middleware* y el *chip*, si durante esa comunicación ocurriera algún tipo de interrupción, terminaría la sesión abierta y el sistema elimina cualquier información extraída del *chip*, hasta que se comience nuevamente el proceso y la transacción sea completada. Siempre que el proceso sea interrumpido, para su recuperación se inicia una nueva sesión asegurando la integridad y confidencialidad de los datos enviados.

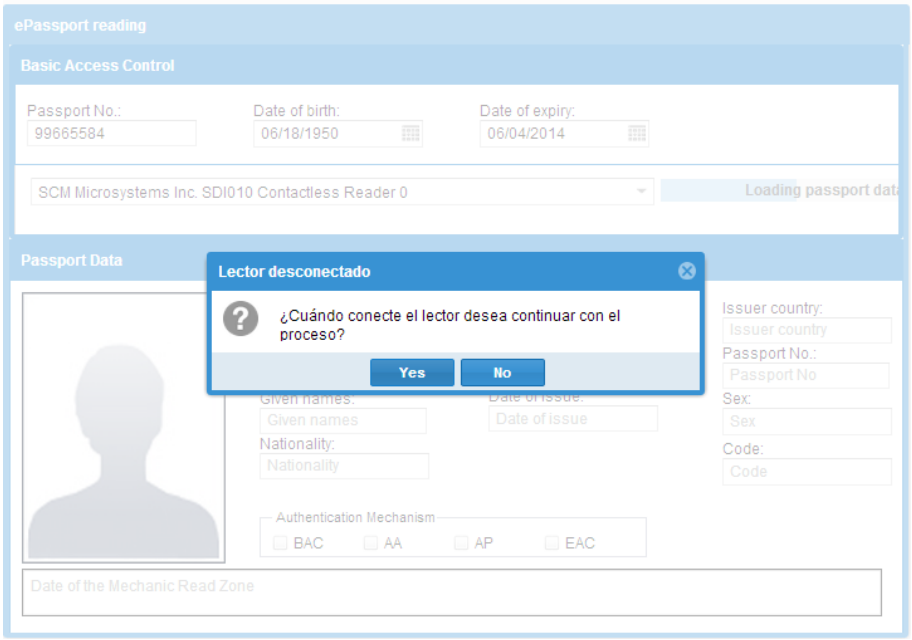


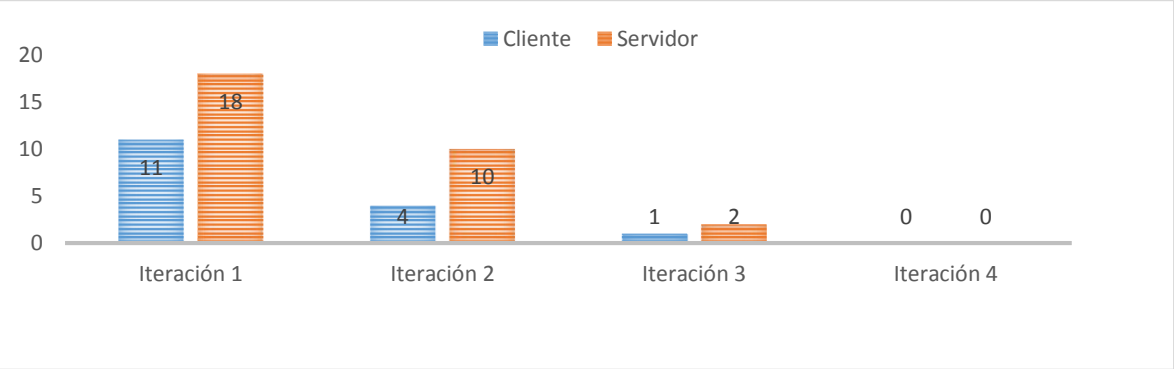
Figura 12: Error que muestra el sistema cuando se desconecta el lector de Tarjetas Inteligentes.

El sistema respondió positivamente ante esta prueba mientras se enviaban datos al *chip* del pasaporte se quitó el lector y el sistema se detuvo mostrando un error (ver figura 13) y ningún dato fue almacenado ni mostrado en la interfaz, luego de conectar el lector comenzó nuevamente la operación y el sistema generó nuevas llaves de sesión hasta que la comunicación entre el *chip* y el *middleware* no fue completada no se mostró la información del proveedor del pasaporte, quedando demostrado la capacidad de recuperación de la plataforma ante estos incidentes y su implementación transaccional.

En el nivel de pruebas de aceptación se hicieron las pruebas de caja negra para determinar si el sistema cumple con lo deseado por el cliente final.

A medida que se iban desarrollando las pruebas se detectaron un grupo de no conformidades, se realizaron 4 iteraciones en las 3 primeras tanto en la parte del cliente como del servidor aparecieron no conformidades y a medida que pasaba el *software* a otra iteración esas no conformidades se iban resolviendo hasta la 4ta iteración que las pruebas no arrojaron no conformidades ver tabla.3

Tabla 3: Resultados de las pruebas de aceptación.



Las funcionalidades comprobadas fueron:

- Gestionar la comunicación con lectores de pasaportes disponibles.
- Permitir la autenticación mediante el Control de Acceso Básico (BAC).
- Enviar y recibir instrucciones del pasaporte electrónico.
- Gestionar la transmisión de información entre el cliente *web* y el servicio que se ejecuta en el servidor
- Procesar las operaciones del *middleware* en el servidor.
- Mostrar al usuario el resultado tras la culminación de la operación del servicio en el servidor

En la tabla 4 se puede observar un ejemplo de un caso de prueba a la funcionalidad autenticar mediante el BAC (ver anexo 4).

El resultado final de las pruebas de aceptación fue la lectura de manera exitosa del pasaporte electrónico venezolano. La figura 14 muestra una foto tomada a la página principal de un pasaporte de pruebas emitido en Venezuela y la figura 15 muestra la lectura realizada al *chip* con el servicio de: Lectura de pasaporte electrónico que fue añadido a la plataforma arrojando, su comparación, resultados idénticos.

Figura 13: Imagen de la página principal del pasaporte electrónico venezolano donde está alojado el *chip*.

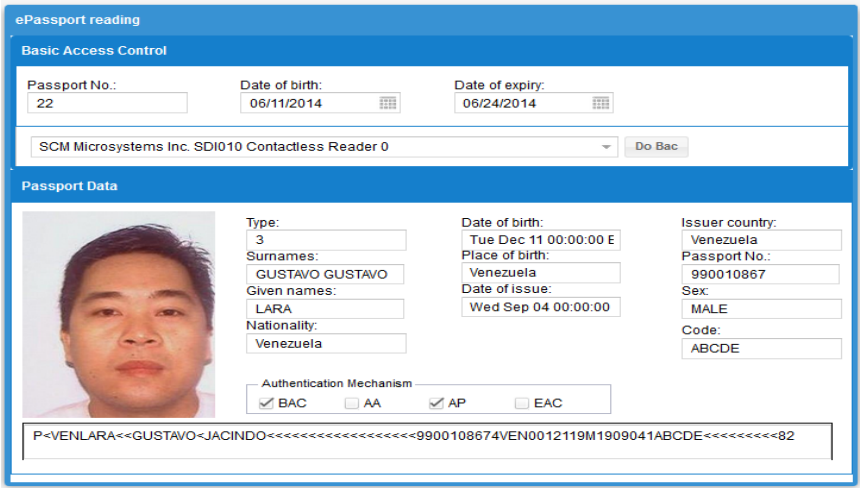


Figura 14: Imagen de la interfaz correspondiente al servicio de lectura de pasaporte electrónico con la información perteneciente al propietario del pasaporte de la figura 14.

3.7. VALIDACIONES REALIZADAS

Para las pruebas de recuperación se utilizó la versión 1.0 de la plataforma SmartCoP, un pasaporte de pruebas emitido por la República Bolivariana de Venezuela, un lector OmniKey 5321 FW5.10 híbrido y para otros servicios agregados a la plataforma se utilizaron las tarjetas Gemplus GemXpresso Pro R3.2 E32 PK (GEMALTO, 2004).

3.7.1. EXTENSIÓN CANAL SEGURO

Incorporación a la plataforma SmartCoP de una extensión demostrando la capacidad y flexibilidad para la incorporación de nuevas aplicaciones, este nuevo servicio no detiene la ejecución del *middleware* de lectura de pasaporte que se utilizó para las pruebas de recuperación en la etapa anterior, sino que extiende la cartera de servicios de la plataforma y es una guía para los desarrolladores que deseen agregar nuevos servicios.

Capítulo 3: Aplicación demostrativa

Se escoge de ejemplo la implementación de un canal seguro GlobalPlatform debido a que se requiere un servicio que cumpla con la norma de seguridad GlobalPlatform y que pueda ser utilizado de manera independiente o ser incorporado y reutilizado por futuras soluciones que demanden seguridad.

Paso 1: Instalación del servidor

El servidor tiene un grupo de componentes que se han ido analizando a lo largo de esta investigación algunos tomados del marco de trabajo JWebSocket perteneciente a la comunidad con ese mismo nombre y otros de elaboración autónoma del autor con alto grado de reusabilidad.

Herramientas

- Netbeans cualquier versión se recomienda a partir de la 7.0 RC
- Máquina virtual de Java JDK versión 7.0
- Maven versión 2.2

Elementos copiados en la máquina donde se piensa instalar el servidor.

- Instalador jWebSocketSamples.
- Servidor de configuraciones
- Instalación de la herramienta apache Maven 2.2 que es la utilizada en el departamento, en caso de tener acceso a internet entonces conectarse al repositorio internacional de Maven situado en la URL http://Maven.apache.org/download.html y descargar las dependencias.
- Para trabajar con el cliente también debe tener en su poder un grupo de ficheros y componentes, los esenciales son:

1. jWebSocket.js
2. jwsEvents*Plug-in*.js
3. JcManager.js (encargado del intercambio con el lector de tarjeta).
4. El *applet* de Java que viene con estos ficheros y los mecanismos de respaldo que vienen con su guía de instalación.

Instalación

1- Instalar la máquina virtual de Java versión 7.0
2- Instalar el Netbeans
3- Instalar Maven

Crear dos variables globales una que lleve por nombre JWEBSOCKET_HOME que apunta a la dirección donde tiene el servidor de configuraciones antes mencionado \...\JWEBSOCKET_SERVER_HOME\jWebSocket-1.0 y otra variable que se llame

JAVA_HOME que apunta a la dirección donde está el JDK de Java ejemplo C:\Program Files\Java\jdk1.7.0.

Se instala el Maven, insertas la dirección de la ubicación física del Maven en el netbeans, en opciones, y ya instalado el Maven previamente se crea en C:\Users\Administrador la carpeta .m2 que es donde se descargan las dependencias del repositorio Maven que el proyecto va a ir utilizando a lo largo del desarrollo.

Paso 2: Creación de una extensión, ejemplo canal seguro

1- Creación del *plug-in* o extensión es una clase que se va a crear en la raíz del proyecto, extenderá de la clase JcPlugin y será la que regirá todo el desarrollo de la extensión ya que interconectará a los diferentes componentes y se encargará de la comunicación con el cliente y la tarjeta.

El nombre de la extensión será canalseguroPlugin

2- Creación de los eventos que serán definiciones orientadas a objeto que usa la plataforma para enviar/recibir mensajes a/desde la parte cliente, el evento llevará por nombre canalseguroEvent.

3- Creación de los comandos APDU necesarios asociados al evento canalseguroEvent, los comandos implementarán una interfaz que lleva por nombre APDU y la estructura del comando definirá su función ejemplo para el comando SelectCommand su estructura es: Cla = (byte) 0x0; Ins= (byte) 0xA4; P1= (byte) 0x4; P2= (byte) 0x0; LC = (byte) mAppName.length; Data= mAppName.

4- En la clase principal que lleva por nombre canalseguroPlugin se comienza la programación de la extensión utilizando una función que brinda la plataforma llamada processEvent, esta función permite el tratamiento de un evento y la recepción de una respuesta generada por ese evento, en el interior de esta funcionalidad es donde se estarán transmitiendo los comandos orientados a la tarjeta y la petición de respuesta.

5- Debido a que el envío/recepción de comandos APDU a la tarjeta se hace de manera síncrona y la plataforma envía y recepciona peticiones de manera asíncrona se hizo necesario que para el trabajo con Tarjetas Inteligentes desde la plataforma se utilizaran estados asociados a las diferentes etapas por la cual atraviesa la transmisión de comandos APDU.

6- Se recomienda que el código asociado a la lógica de la extensión se cree en una carpeta separada del *plug-in*, esto permite una mayor organización y ayuda al mantenimiento y reutilización del código asociado a la extensión.

7- Un evento generado en la interfaz de usuario, acarrea un grupo de información como es: información del terminal, de los lectores asociados a ese terminal, el

navegador que está utilizando, si es compatible o no con *web*socket, una marca de tiempo y con toda esta información se crea una sesión la cual tendrá un tiempo de vida limitado y su función fundamental es asegurar el flujo de comunicación entre los componentes de la plataforma.

8- El modelo de desarrollo de la plataforma SmartCoP permite no solo que se comience las peticiones del lado del cliente sino que también se pueda originar del lado del servidor, dándole información a los usuarios acerca de nuevos productos o permitiéndoles actualizaciones de aplicaciones no solicitadas por él sino sugeridas por la plataforma.

9- Después de ser transmitidos los comandos a la tarjeta y con la recepción de una respuesta la plataforma permite enviar la información organizada a la interfaz visual del cliente estructurando un mapa[33] y utilizando para su transmisión el contexto que enmarca la sesión asociada a la petición.

10- Cada evento tiene asociado un grupo de definiciones que se configuran de manera manual en el servidor de configuraciones:

Tabla 4: Atributos y definiciones manuales de un evento.

#	Atributo	Descripción
1	*String id*	El nombre del identificador de la extensión.
2	*String ns*	La clase usada para la creación de la instancia de la extensión.
3	*Set<Argument>* incomingArgsValidation	El conjunto de argumentos de entrada que tendrá asociado el evento.
4	*Set<Argument>* outgoingArgsValidation	El conjunto de argumentos de salida que tendrá asociado el evento.
5	*Boolean* responseRequired	Indica si el evento requiere respuesta del cliente por defecto este atributo tiene valor falso.
6	*Boolean* responseToOwnerConnector	Indica si el cliente que generó el evento está esperando una respuesta y por defecto este atributo tiene valor true.
7	*Boolean* responseAsync	Indica si la respuesta es asíncrona o no Por defecto este atributo tiene valor falso.
8	*Boolean* notificationConcurrent	Indica si los oyentes pueden ser notificados concurrentemente o en una iteración del mismo hilo de ejecución. Por defecto este atributo tiene valor falso.
9	*Boolean* cacheEnabled	Indica si el evento se le habilita la caché

[33] Estructura de datos clave valor

		Por defecto este atributo tiene valor falso.
10	*Integer* cacheTime	Tiempo que demora almacenada en caché la respuesta del evento. Por defecto es 0
11	*Boolean* securityEnabled	Indica si la seguridad esta activada para un evento
12	*Set<String>* roles	Restricciones de roles para que no sean notificados de la respuesta de un evento
13	*Set<String>* ipAddresses	Restricciones de IP para que no sean notificados de la respuesta de un evento
14	*Set<String>* users	Restricciones de usuarios para que no sean notificados de la respuesta de un evento
15	*Validator validator*	La instancia principal de *Validator*
16	*Integer* timeout	El tiempo límite asociado a la respuesta del servidor dadas en milisegundos La respuesta es 1000 (1 *second*).

Para la adición de nuevos atributos se debe agregar un nuevo filtro en la cadena de filtros del servidor de configuraciones. (Masó, 2010)

Para el caso del canalseguroPlugin, en la figura 13 se muestra la configuración básica para su correcto funcionamiento y ejecución.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

    <bean id="JcGlobPlugIn" class="org.jwebsocket.plugins.jc.GlobalPlatformPlugIn" parent="AbstractPlugIn">
        <property name="id" value="jcg.auth" />
        <property name="emEventClassesAndClientAPI">
        <map>
                <entry key="autenticar" value="org.jwebsocket.plugins.jc.event.Canal_seguro"/>
                <entry key="terminalReady" value="org.jwebsocket.eventmodel.event.card.JcTerminalReady"/>
                <entry key="terminalNotReady" value="org.jwebsocket.eventmodel.event.card.JcTerminalNotReady"/>
            </map>
        </property>
        <property name="eventsDefinitions">
            <set>
                <bean parent="AbstractEventDefinition">
                    <property name="id" value="jc.autenticar" />
                    <property name="ns" value="org.jwebsocket.plugins.jc.event.Canal_seguro" />
                    <property name="responseRequired" value="true" />
                    <property name="responseToOwnerConnector" value="true" />
                    <property name="incomingArgsValidation">
                        <set>
                            <bean class="org.jwebsocket.eventmodel.filter.validator.Argument" >
                                <property name="name" value="nivelseguridad" />
                                <property name="type" value="string" />
                                <property name="optional" value="false" />
                            </bean>
                            <bean class="org.jwebsocket.eventmodel.filter.validator.Argument" >
                                <property name="name" value="appName" />
                                <property name="type" value="string" />
                                <property name="optional" value="false" />
                            </bean>
                        </set>
                    </property>
                </bean>
            </set>
        </property>
    </bean>
</beans>
```

Figura 13: Servidor de configuraciones para el canalseguroPlugin.

11- En el cliente se utilizarán los ficheros JavaScript que propone la plataforma y se crearán los de uso propio implementando la funcionalidad checkwebsocketSupport ver figura 14.

12- La funcionalidad anteriormente mencionada permite la creación de una cadena de filtros y notificadores de eventos, la generación de los *plug-in* y la comprobación de la transmisión de información entre el componente que radica en el cliente encargado de la comunicación con la tarjeta y la tarjeta.

```
function checkWebSocketSupport(){
    if( jws.browserSupportsWebSockets() ) {
        lWSC = new jws.jWebSocketJSONClient();
        lWSC.open("ws://127.0.0.1:8787/jWebSocket/jWebSocket", {
            OnOpen: function(aToken){


                //Creating the filter chain
                securityFilter = new jws.SecurityFilter();

               cacheFilter = new jws.CacheFilter();
                cacheFilter.cache = new Cache();
                validatorFiler = new jws.ValidatorFilter();

                //Creating a event notifier
                notifier = new jws.EventsNotifier();
                notifier.ID = "notifier0";
                notifier.NS = "jc";
                notifier.jwsClient = lWSC;
                notifier.filterChain = [securityFilter, cacheFilter, validatorFiler];
                notifier.initialize();

                generator = new jws.EventsPlugInGenerator();

                //Generating the auth & test plug-ins.
                auth = generator.generate("auth", notifier, function(){

                });

                jc = generator.generate("jc.auth", notifier, function(){

                    jc.transmit = function(e){
                        var lResponseAPDU = JcManager.transmit(e.terminal, e.apdu);

                        return lResponseAPDU;
```

```
                    }
                    JcManager.addListener({
                        OnTerminalReady: function(aTerminal){

                            jc.terminalReady({
                                args: {
                                    terminal: aTerminal

                                },
                                OnSuccess: function(aToken){

                                                    }
                                }
                            });
                        },
                        OnTerminalNotReady: function(aTerminal){

                            jc.terminalNotReady({
                                args: {
                                    terminal: aTerminal
                                }
                            });
                            terminallista=false;
                            if (lWSC.isLoggedIn() && lWSC.getUsername() != "anonymous")
                                w.auth.logoff();
                        }
                    });  });  },      })
    } else {
        var lMsg = jws.MSG_WS_NOT_SUPPORTED;
        alert( lMsg );
        console.info( lMsg );
    }
}
```

Figura 14: Implementación en el cliente de la funcionalidad checkwebsocketSupport.

Capítulo 3: Aplicación demostrativa

Para la culminación en la parte cliente del desarrollo del canal seguro se debe llamar al *plug-in* generado que lleva por nombre canalseguroPlugin y enviarle hacia el servidor un nivel de seguridad que puede ser 0,1 o 3.

Otros resultados que le dan validez a la plataforma y demuestran su usabilidad son los trabajos de diploma:

1- Aplicación *web* para la administración de Tarjetas Inteligentes con GlobalPlatform de los autores Yasiel Conde Bernal y Rita Milena Hernández Díaz.

2- Aplicación *web* para la lectura de pasaporte electrónico de los autores Reisel de la Rosa Ge y Daniel Fuentes Castro.

3- Aplicación *web* para la lectura y personalización del documento Licencia de conducción electrónica de los autores Laura Calderon Ramos y Robin Hernández Herrera.

3.8. CONCLUSIONES PARCIALES

En este capítulo:

- Los componentes que forman parte del sistema final, son soluciones de un alto nivel de independencia, que, con mínimos cambios en su configuración, pueden ser usados en otros sistemas.
- Los diagramas generados en este capítulo permitieron un mayor entendimiento de la plataforma y mejor comunicación entre los integrantes del equipo de desarrollo.
- Como validación del desarrollo de la plataforma se procedió a la explicación de cómo realizar una extensión de canal seguro y adicionarla a la plataforma con el propósito de que sirva de ejemplo a futuros desarrolladores que deseen colaborar con la plataforma y su extensibilidad.

Conclusiones y recomendaciones

CONCLUSIONES

- Con la elaboración del marco teórico se evidenció las deficiencias del modelo de desarrollo existente para la implementación de servicios en línea utilizando Tarjetas Inteligentes y aportó la base teórica – metodológica para la elaboración del modelo propuesto.
- La definición del Modelo para el desarrollo de servicios en línea utilizando Tarjetas Inteligentes sirvió como marco de referencia y guía para el desarrollo de una plataforma que permite la comunicación en línea con Tarjetas Inteligentes, funcional para diferentes tipos de navegadores y sistemas operativos, aprovechando al máximo todas las ventajas de estas a través de Internet.
- El desarrollo de la plataforma SmartCoP permitió la comunicación entre una aplicación *web* y las Tarjetas Inteligentes, conteniendo el *middleware* del lado del servidor, lo cual garantiza a los usuarios una mayor seguridad y confianza en aplicaciones de este tipo y facilita la actualización de las soluciones anexadas al servidor, así como la incorporación de nuevas funcionalidades.

RECOMENDACIONES

Extender el modelo propuesto a desarrolladores de aplicaciones Java Card fuera de la Universidad de las Ciencias Informáticas.

Integrar la plataforma a la solución de *Identity Management System* desarrollada en el departamento de Componentes del Centro de Identificación y Seguridad Digital.

Referencias

REFERENCIAS BIBLIOGRÁFICAS

Altova Inc. 2010. altova.com. *altova.com.* [En línea] 8 de 2010. [Citado el: 30 de 1 de 2014.] http://www.altova.com/umodel.html.

Cardlogix. 2012. http://www.cardlogix.com. *http://www.cardlogix.com.* [En línea] 21 de 3 de 2012. [Citado el: 22 de 8 de 2013.] http://www.cardlogix.com/products/software/middleware.asp.

cardwerk. 2010. cardwerk. *cardwerk.* [En línea] 5 de 2010. [Citado el: 1 de 2 de 2014.] http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816.aspx.

Cardwerk. 2011. cardwerk.com. *cardwerk.com.* [En línea] 2011. [Citado el: 11 de 2 de 2014.] http://www.cardwerk.com/smartcards/smartcard_history.aspx.

Carlisle Adams, Steve Lloyd. 2002. *Understanding PKI: Concepts, Standards, and Deployment Considerations, Second Edition.* s.l. : Addison Wesley, 2002. 0-672-32391-5.

CARNICERO, J. 2010. De la historia clínica a la historia de salud electrónica. [En línea] 2010. [Citado el: 12 de 8 de 2013.] http://www.conganat.org/seis/informes/2003/PDF/capitulo1.pdf.

Delitosinformaticos. 2012. delitosinformaticos.com. *delitosinformaticos.com.* [En línea] 35 de 10 de 2012. [Citado el: 17 de 3 de 2014.] http://www.delitosinformaticos.com/especial/seguridad/ssl.shtml.

E. U. Informática en Segovia. 2013. *EL MODELO CLIENTE/SERVIDOR.* Valladolid : s.n., 2013.

Eclipse. 2009. http://www.eclipse.org. *http://www.eclipse.org.* [En línea] 11 de 3 de 2009. [Citado el: 1 de 5 de 2014.] http://www.eclipse.org/jetty/.

Effing, Wolfgang Rankl and Wolfgang. 2003. *Smart Card HandBook.* New York : John Wiley & Sons Ltd, Baffins Lane, Chichester, 2003.

Epidataconsulting. 2010. www.epidataconsulting.com. *www.epidataconsulting.com.* [En línea] 6 de 2010. [Citado el: 26 de 6 de 2014.] http://www.epidataconsulting.com/tikiwiki/tiki-index.php?page=Spring+IoC+Container.

Extending the data storage capabilities of a Java-based smartcard. **CAP, CLEMENS H y MAIBAUM, NICO y HEYDEN, LARS. 2001.** Sixth IEEE Symposium on Computers and Communications. : s.n., 2001. 0769511775.

Fedo. 2013. fedo.org. *fedo.org.* [En línea] 21 de 10 de 2013. [Citado el: 10 de 4 de 2014.] http://www.fedo.org/web/cartografia/soporte-tecnico/applets.

Fowler, Kent Beck y Martin. 2000. *Programming Planning Extreme.* s.l. : Addison Wesley, 2000. 0-201-71091-9.

FRZ, Pieter H. Hartel Eduard K. de jong. 2005. *Smart Cards and card operating systems.* Reino unido : s.n., 2005.

Ge, Reisel de la Rosa. 2012. *Aplicación web para la lectura de pasaportes electrónicos. .* La Habana : s.n., 2012.

Gemalto GemBorder . 2006. *GemBorder Applet Reference Manual.* Francia : s.n., 2006.

Gemalto. 2009. sconnect. *sconnect.* [En línea] 2009. [Citado el: 21 de 1 de 2013.] http://sconnect.software.informer.com/.

Referencias

GlobalPlatform. 2003. *Card Specification GlobalPlatform.* 2003.

—. 2004. *Card Specification version 2.1.1.* 2004.

—. 2014. www.globalplatform.org. *www.globalplatform.org.* [En línea] 14 de 5 de 2014. [Citado el: 20 de 6 de 2014.] http://www.globalplatform.org/specifications.asp.

Grizzly. 2010. https://grizzly.java.net/. *https://grizzly.java.net/.* [En línea] 8 de 2010. [Citado el: 21 de 2 de 2014.] https://grizzly.java.net/.

Groner, Loiane. 2011. *Ext JS 4 First Look.* Mumbai : Packt Publishing, 2011.

ICAO 9303 vol1part2. 2006. *Especificaciones para pasaportes electrónicos .* 2006.

ISO. 2006. www.iso.org. *www.iso.org.* [En línea] 2006. [Citado el: 21 de 1 de 2014.] http://www.iso.org/iso/catalogue_detail.htm?csnumber=50942.

ISO/IEC 18013-3. 2009. *Personal identification - Access control, authentication and integrity validation.* 2009.

ISO/IEC. 2005. *ISO/IEC 7816-4.* 2005.

ISO/IEC JTC1/SC17 N 1363. 1997. waazaa.org. *waazaa.org.* [En línea] INTERNATIONAL STANDARD, 1997. [Citado el: 21 de 3 de 2014.] http://www.waazaa.org/download/fcd-14443-1.pdf.

jwebsocket. 2014. jwebsocket.org. *jwebsocket.org.* [En línea] 21 de 3 de 2014. [Citado el: 201 de 6 de 2014.] https://jwebsocket.org/.

Kalysis. 2010. Kalysis.com. *Kalysis.com.* [En línea] 10 de 5 de 2010. [Citado el: 21 de 1 de 2014.] http://kalysis.com/content/modules.php?op=modload&name=EasyContent&file=index&menu=1501&page_id=13.

Kalysis-solouna. 2012. Solouna.com. *Solouna.com.* [En línea] 2 de 5 de 2012. [Citado el: 21 de 3 de 2014.] http://solouna.com/lineas/.

Leyva, Yandier Mayo. 2008. *Implementación de las validaciones en JavaScript para la AGR. Adaptación al framework ExtJS.* La Habana : s.n., 2008.

Markantonakis, Keith E. Mayes and Konstantinos. 2004. *Smart Cards, Tokens, Security and Applications.* New York : s.n., 2004. 978-0-387-72197-2.

Márquez, Joaquín Torres. 2006. *Nuevo Marco de Autenticación para Tarjetas Inteligentes en Red. Aplicación al Pago Electrónico en entornos Inalámbricos.* Leganes : s.n., 2006.

Masó, Rolando Santamaría. 2010. *EventsPlugIn Developer Guide.* Artemisa : s.n., 2010.

—. 2012. *Extensión del marco de trabajo jWebSocket para el desarrollo de aplicaciones web empresariales.* La Habana : s.n., 2012.

Modelo para la extensión de las capacidades de procesamiento y memoria de tarjetas inteligentes Java Card. **Susana María Ramirez Brey, Adonis César Legón Campo, Yusnier Valle Martínez. 2013.** 1, La Habana : RCCI, 2013, Vol. 8.

Netti. 2010. http://netty.io/. *http://netty.io/.* [En línea] 5 de 2010. [Citado el: 2 de 5 de 2014.] http://netty.io/.

Referencias

Openpcd. 2008. Openpcd.org. *Openpcd.org.* [En línea] 21 de 2 de 2008. [Citado el: 21 de 6 de 2014.] http://www.openpcd.org/ISO14443.

Oracle. 2013. http://docs.oracle.com. *http://docs.oracle.com.* [En línea] 12 de 2 de 2013. [Citado el: 20 de 6 de 2014.] http://docs.oracle.com/javase/7/docs/jre/api/security/smartcardio/spec/javax/smartcardio/package-summary.html.

Ortega, M. 2008. *Implementación de Tarjeta Inteligente Java Card para el Control de Acceso a Instalaciones.* [http://www.eatis.org/eatis2010/portal/paper/memoria/html/files/sistemas/Maria%20Ortega.pdf] 2008.

Patterson, Diana R. Zapata Vizcaíno y Raidel Abreu. 2011. *Applet y Middleware para gestionar la información de historiales clinicos en tarjetas inteligentes.* La Habana : s.n., 2011.

PC/SC. 2005. *Interoperability Specification for ICCs and Personal Computer Systems.* 2005.

Pérez, Orén Fornaris Cabreja y Kirenia Garcia. 2010. *Solución para gestionar el control de acceso a instalaciones utilizando tarjetas inteligentes.* La Habana : s.n., 2010.

Perovich, Daniel, Rodríguez, Leonardo and Martín, Varela. 2010. *Proyecto de Taller V Programación de Java Card.* 2010.

Pressman. 2008. *Ingenieria de Software Un Enfoque Practico Pressman 5th Ed.* 2008.

Prezi. 2014. prezi.com. *prezi.com.* [En línea] 2014. [Citado el: 2 de 4 de 2014.] http://prezi.com/xq-ar0rsvul_/programas-en-java/.

Programacion. 2010. www.programacion.com. *www.programacion.com.* [En línea] 2010. [Citado el: 21 de 6 de 2014.] http://www.programacion.com/articulo/eclipse_--_ii_--_integracion_con_junit_291.

Pruebasdesoftware. 2005. pruebasdesoftware.com. *pruebasdesoftware.com.* [En línea] 21 de 8 de 2005. [Citado el: 20 de 6 de 2014.] http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

RAE. 2014. Real Academia Española. *Real Academia Española.* [En línea] 17 de 6 de 2014. [Citado el:] http://buscon.rae.es/drae/srv/search?id=xzHZWdlqrDXX2u7iR2i0.

REILLO, RAÚL SÁNCHEZ. 2000. *Mecanismo de autenticación biométrica mediante Tarjetas Inteligentes.* madrid : s.n., 2000.

Rfidpoint. 2014. RFIDPOINT.com |. *RFIDPOINT.com* |. [En línea] 2014. [Citado el: 15 de 3 de 2014.] http://www.rfidpoint.com/fundamentos/middleware/.

Rolando Santamaría Masó. 2010. *EventsPlugIn Developer Guide.* La Habana : s.n., 2010.

smartcardbasics. 2010. smartcardbasics.com. *smartcardbasics.com.* [En línea] 8 de 2010. [Citado el: 25 de 1 de 2014.] http://www.smartcardbasics.com/smart-card-standards.html.

SOMMERVILLE, IAN. 2005. *Ingeniería del software Séptima edición.* Madrid : Addison Wesley, 2005. 84-7829-074-5.

Statcounter. 2013. gs.statcounter. *gs.statcounter.* [En línea] 2010 de agosto de 2013. [Citado el: 21 de enero de 2014.] http://gs.statcounter.com/#browser-ww-monthly-201303-201402.

Referencias

Stuart Ashworth, Andrew Duncan. 2012. *Ext JS 4 Web Application Development Cookbook.* Mumbai : Packt Publishing, 2012.

UAB. 2013. Universidad autónoma de Barcelona. *Universidad autónoma de Barcelona.* [En línea] 23 de 1 de 2013. [Citado el: 20 de 9 de 2013.] http://www.uab.cat/servlet/Satellite/como-se-utiliza/lector-y-activacion-del-servicio-1266477271107.html.

UVA. 2012. Universidad de Valladolid. *Universidad de Valladolid.* [En línea] 21 de 3 de 2012. [Citado el: 21 de 6 de 2013.] http://www.uva.es/opencms/contenidos/serviciosAdministrativos/infraestructuras/serviciosTecnologiasInformacion/3CatalogoDeServicios/tarjetaInteligente/serviciosTarjetaInteligente.

Vázquez, Mª Celeste Campo. 2014. www.it.uc3m.es. *www.it.uc3m.es.* [En línea] 21 de 3 de 2014. [Citado el: 16 de 6 de 2014.] http://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=8&ved=0CGYQFjAH&url=http%3A%2F%2Fwww.it.uc3m.es%2F~celeste%2Fdocencia%2Femaster%2F2005%2FTarjetas_JavaCard_B2C2005.ppt&ei=ZQWiU6nlFsiMqgbVpoC4Bw&usg=AFQjCNEMirTtIBcuGlzdL4P5Q0-L-KyXNQ&bvm=bv..