Universidad de las Ciencias Informáticas

Facultad 1

Título: Componente para la interacción entre los lectores de tarjetas inteligentes que cumplan el estándar PC/SC y el navegador web Mozilla Firefox.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

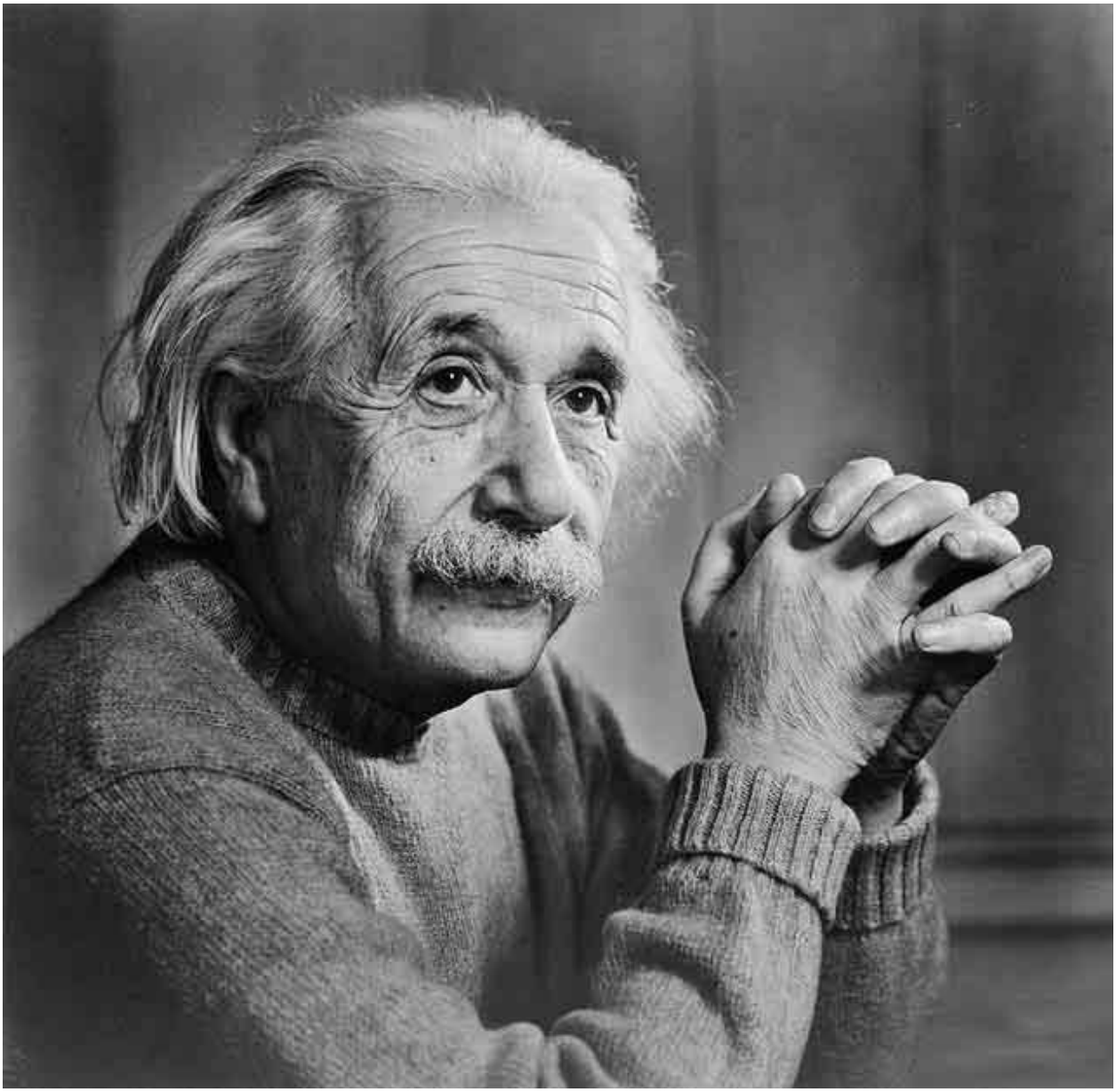
Autores: Yadira García Huelga

Angel Manuel Romero Alfonso

Tutores: Ing. Katerina Pereda Viñolo

Ing. Vismar Fernández Santana

"Ciudad de La Habana. Junio, 2010"



"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein. *(1879-1955) Científico alemán nacionalizado estadounidense.*

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____ _____

Yadira García Huelga Angel Manuel Romero Alfonso

_____ _____

Ing. Katerina Pereda Viñolo. Ing. Vismar Fernández Santana.

Dedicatoria

Yadira:

Especialmente a mis padres, quienes desde niña con su amor y educación, me inculcaron valores que contribuyeron a mi formación durante estos 5 años.

A mi familia: a Erika, esa pequeñita que cuando me dice tata hace que todo lo que me rodea se vuelva pequeño a su alrededor y para quien pretendo ser un ejemplo a seguir. A mis tías y tíos, en especial a Rosa que donde quiera que este, sé que estará orgullosa de verme al fin graduada de Ingeniera. A mis abuelas y primos que siempre han estado para mí desde que tengo uso de razón.

Angel:

Le dedico esta tesis a mi mama por darlo todo para mí y a mi papa, ya fallecido, que siempre se sacrificó para que no me faltara nada, aun cuando estaba enfermo. A mi hermano Gilbertico, que siempre me dio ánimo para que nunca dejara, ni deje de estudiar. A mis tías y tíos por sus consejos y preocupación, a mis primos Alejandro y Linda que los quiero mucho y a todos mis amigos.

Agradecimientos:

A la Revolución por ofrecernos la posibilidad de estudiar y realizarnos como profesionales. A todos los profesores que de una forma u otra influyeron en nuestra preparación. Al equipo del laboratorio, por su aporte en este trabajo. A los tutores y a Ander por la paciencia y dedicación que tuvieron con nosotros. A los miembros del tribunal y el oponente que influyeron en gran medida en el perfeccionamiento de esta investigación.

Yadira:

A mi mamita, gracias por estar siempre disponible para mí; por ser un ejemplo de madre, de hija, de profesional y de mujer; por todos los esfuerzos que has hecho para que yo logre este sueño que al final sabemos que es de ambas , por tu educación, gracias por existir mamá.

A papá, por estar siempre ahí incondicionalmente, por apoyarme en todo, aunque a la larga sepas que me estoy equivocando; por darme esa hermanita tan linda que me ha hecho despertar sentimientos que no conocía, mil gracias por ser el padre que eres.

A mis tías (Lucy y Nancy), por escucharme cuando más lo necesito, por darme apoyo y sus sabios consejos.

A mi tío (Juan Antonio), gracias por estar ahí siempre aunque por tu carácter no lo demuestres.

A mis abuelitas lindas (Anita y Titirí), por darme esos padres de los cual vivo orgullosa, las quiero mucho.

A mis primos (Juan Carlos,Yamilka, Maydel y Alexander), gracias por demostrarme que estarán ahí cuando lo necesite.

Ale, mi amor sabes que sin ti se me hubiera hecho más difícil el camino, eres una de las mejores cosas que me llevo de la UCI, muchas, muchas, muchas gracias por tu amor, por ser mi familia aquí durante este último curso, te amo.

A Daymí, gracias por ser la hermana mayor que nunca tuve (aunque solo nos llevemos 1 mes).

A mis amigos (Luz, Yari, Arle, Lissi y Abraham) que más que eso han sido mi familia en esta escuela.

A mis compañeros de aula y de apartamento, gracias por soportarme durante estos 5 años.

A mi compañero de tesis, desde el principio supe que lo lograríamos y al final no me equivoqué, ya estamos aquí, gracias por todo.

En general a todos los que de una forma u otra han contribuido a mi formación durante estos 5 años.

Angel:

Agradezco ante todo, a Dios quien me lo dio todo, la vida, mis padres, mi hermano, mis tías y mis tíos, mi novia y mis amigos y amistades.

Agradezco a mi mamá, gracias por estar siempre disponible para mí; por ser un ejemplo de madre y de mujer; por todos los esfuerzos que has hecho para que yo lograra este sueño, por tu educación, tu dedicación, tus oraciones, tus peleas y regaños, tus consejos, gracias por existir.

A mi papá que aunque no está en vida, pero formó en mí lo que ahora soy, por educarme, por su preocupación que tuvo siempre por mí.

A mi hermano Gilbertico, por apoyarme en todo lo que necesité, tanto material y espiritual. Además por darme ánimo en los momentos tristes y consejos.

A mis tías y tíos Yuli, Ismael, Reglita, por escucharme cuando más lo necesito, por darme apoyo y sus sabios consejos.

Aliana, gracias por apoyarme, por darme ánimo, siempre me apoyaste, me soportaste. En los momentos que más te necesitaba estabas ahí, para darme esa

ayuda spiritual que es invaluable, única. Siempre vas a estar en mi corazón porque has dejado tu huella ahí para siempre.

A todas mis amistades.

Resumen

Con el avance de las tecnologías y las comunicaciones, el uso de internet ha revolucionado la utilización de los servicios en líneas. La mayoría de estos sitios que se encuentran en la web ofrecen servicios utilizando tarjetas inteligentes, esto lo hacen a través de una capa intermedia que se deben instalar en las computadoras del usuario para interactuar con las mismas. La ausencia de una tecnología que posibilite la comunicación en línea que utilizando estos dispositivos inteligentes y que además, sea funcional para diferentes navegadores web y sistemas operativos, ha hecho que no se aprovechen al máximo las ventajas de las tarjetas a través de Internet.

En el año 2008 se crea el Centro de Identificación y Seguridad Digital en la Universidad de las Ciencias Informáticas, el cual dedica una de sus líneas de trabajo a software que hagan uso de las tarjetas inteligentes. El mismo ha identificado la necesidad del desarrollo de un componente que permita la interacción entre los lectores de tarjetas y navegadores web, para acceder a sitios en línea que utilizan esta tecnología.

En la investigación se hace un estudio de las tarjetas inteligentes y sus ventajas. Se obtuvo como resultado un componente que permite la interacción con tarjetas mediante el navegador web Mozilla Firefox.

El documento recoge los resultados del trabajo realizado, describiéndose las principales características de los sistemas analizados, la arquitectura y el diseño del sistema propuesto. Se describen las herramientas y tecnologías utilizadas además de los artefactos generados en el proceso de desarrollo.

Palabras Claves: librerías de software, navegador web, Internet, componente, sitios en líneas.

Nota: Para las palabras técnicas se usó una fuente de letra distinta a la del documento.

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 y además provocó que muchos servicios como el correo, las reservaciones de viaje, las transacciones bancarias entre otros, comenzaran a prestarse a través de la red de redes 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, de una manera u otra por Internet.

La conexión a Internet es el mecanismo que se establece entre las computadora para conectarse a esta red, permitiendo visualizar al usuario las páginas web desde un navegador. Los navegadores web son programas que permiten visualizar la información que contienen las páginas web. Existen varios de ellos, Internet Explorer, Mozilla Firefox, Opera, Safari y otros, siendo los dos primeros los más utilizados.

Como consecuencia del auge que ha tenido Internet y las comunicaciones, además el gran número de usuarios que acceden a este servicio, la seguridad informática se ha convertido en un factor importante a tener en cuenta. 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 esa palabra clave. Es típico que los usuarios con poca experiencia o bajo nivel de conciencia utilicen contraseñas cortas e inseguras y además las dejen guardadas en los historiales de la computadora, facilitando el trabajo a quien pretende sustraerlas. Por otra parte, la frecuencia del cambio de contraseñas en Internet, se ha vuelto cada día más obligatorio en inicios de sesión de sitios web; no es recomendable usar la misma clave en varios sitios web, y al usuario se le dificulta memorizar tantas, recurriendo así, en algunos casos, a dejarlas plasmadas en papeles que pueden caer

en manos de personas malintencionadas. Para lograr la seguridad de las contraseñas, esta debe estar compuesta lo más posible de un conjunto de caracteres inteligibles.

Desde hace un tiempo se viene desarrollando la tecnología de los dispositivos inteligentes, ayudando a facilitar la seguridad de los sistemas de autenticación usados en combinación. Como parte de estos las tarjetas inteligentes constituyen uno de los avances tecnológicos que han venido a revolucionar en gran medida muchos de los procesos en la sociedad. Estas no son más que una lámina plástica que traen incorporados circuitos integrados que permiten almacenar información de forma segura e incluso la ejecución de cierta lógica programada. Contienen componentes de memoria volátil y no volátil que, unidos a un microprocesador y un sistema operativo, permiten almacenar, encriptar y modificar información. Tienen un gran número de aplicaciones, entre las más significativas está la firma digital de documentos, en sistemas de control de acceso, como monedero electrónico, en el pago de la televisión, telefonía móvil, transporte público, para acceso a sitios web, sistemas nacionales de salud, servicios bancarios, licencias de conducción, etc. Además, en varios países del mundo se usan como documento oficial de identificación, entre los que están España y Finlandia.

Una ventaja de esta tarjeta es que la clave privada de una persona está codificada dentro de la tarjeta y nunca sale de ella, así se hace muy difícil obtener la clave privada; esto evita que se pueda comprometer la seguridad del sistema. Además, estos dispositivos permiten guardar más de un certificado en una misma tarjeta.

El uso de Internet ha promovido la utilización de los servicios en líneas, entre los que se pueden destacar el gobierno en línea, las pasarelas de pago, los servicios de correo electrónico, la reservación hoteles o de viaje, la certificación de documentos, la emisión de declaraciones de impuestos o la tramitación de documentos oficiales de una nación determinada. En muchos de los servicios que brindan estos sitios se puede utilizar las tarjetas inteligentes incrementando la calidad y seguridad de los mismos.

A finales del 2004 aparece en el mercado Firefox, una rama de desarrollo de Mozilla. Este proyecto pretendía eliminar todas las funciones ajenas a un navegador propiamente dicho y mejorar su código e interfaz, haciéndolo más rápido, seguro, y completamente personalizable por el usuario.

Para acceder a la web uno de los navegadores más utilizados es el Mozilla Firefox, pues puede ser personalizado por el usuario por mediación de extensiones o componentes denominados `add-on`[1].Las extensiones son muy populares con Firefox, ya que la intención de los desarrolladores de Mozilla es reducir los errores de `software`, manteniendo un alto grado de extensibilidad, para que los usuarios individuales puedan agregar las características que ellos prefieren. Actualmente existen varios sitios web donde los usuarios pueden subir las extensiones hechas por ellos para compartirlas con las demás personas.

El desarrollo de la informática en nuestro país ha crecido vertiginosamente en los últimos años. Como parte de las acciones encaminadas a incrementar el nivel científico en esta rama surgió en el año 2008 el Centro de Identificación y Seguridad Digital en la Universidad de las Ciencias Informáticas. Una de las actuales líneas de investigación y desarrollo está dirigida a potenciar la creación de soluciones informáticas que hagan uso de las tarjetas inteligentes. Luego de varios resultados significativos, surgió en este departamento la idea de impulsar el desarrollo de los servicios en línea mediante el uso de tarjetas inteligentes, pues el país no cuenta con ningún software que resuelva este problema y los existentes en el mundo son de origen propietario.

Entre los productos desarrollados en esta línea se encuentra una solución que posibilita el uso de las tarjetas inteligentes a través de la web, situando los correspondientes `middlewares` [2] del lado del servidor, lo cual facilita la actualización de los sistemas y la incorporación de nuevas funcionalidades de manera centralizada, evita los riesgos

[1] Son programas opcionales, que solo funcionan anexados a otros y que sirven para incrementar o complementar sus funcionalidades.

[2] Es un software de conectividad que permite la interconexión entre diferentes aplicaciones y el acceso a las funcionalidades y datos de estas, desde y a través de otros sistemas, independientemente de la plataforma y el sistema operativo

de seguridad que implicaría situarlos en la propia estación cliente y lo que es más importante, brinda el soporte para la prestación de servicios en línea que utilicen las tarjetas inteligentes, incrementando la seguridad y calidad de los mismos. Si bien la solución ha logrado una versión completamente funcional de los componentes que se ubican del lado del servidor, el componente del lado del cliente, hasta hoy, solo se ha adaptado para el navegador Internet Explorer, lo cual impide que todos los usuarios asiduos de Mozilla Firefox puedan hacer uso de la misma.

Partiendo de todo lo anteriormente abordado se manifiesta la siguiente **situación problémica**: Actualmente el manejo de tarjetas inteligentes en el mundo se efectúa mediante el uso de capas intermedias de software conocidas como `middlewares` que se ejecutan en la computadora del usuario, almacenando en ella las llaves simétricas que rigen los protocolos de comunicación con las tarjetas. Esto constituye un riesgo potencial para la seguridad de las llaves pues implica una mayor dificultad para la actualización e incorporación de nuevas funcionalidades, además de limitar el uso que pudiera dársele a las tarjetas inteligentes para impulsar el desarrollo y la seguridad de los servicios en línea en internet. Por otra parte, se hace incómodo para el cliente realizar actualizaciones de estos `middlewares` que se encuentran en su computadora, ya que el mismo tendría que poseer conocimientos acerca de estas tecnologías, así como permisos de administración para poder instalarlas. Dentro del Centro de Identificación y Seguridad Digital (CISED) se ha resuelto parcialmente este problema con la creación de una solución que permite la interacción entre el navegador web [3] Internet Explorer y los lectores de tarjeta, sin embargo, para Mozilla Firefox sigue existiendo la misma limitación.

A partir de la situación anteriormente expuesta se encuentra el siguiente **problema científico:** ¿Cómo establecer la interacción entre lectores de tarjetas inteligentes que

[3] Es un programa que permite visualizar la información que contiene una página web, interpreta el código de la página, HTML generalmente y lo presenta en el monitor de la computadora permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos.

cumplan el estándar PC/SC y el navegador Mozilla Firefox para acceder a servicios en línea?

Como **objeto de estudio** se tiene: El proceso de desarrollo de componentes para la interacción con tarjetas inteligentes. El **campo de acción** se enmarca en el proceso de desarrollo de componentes para la interacción con tarjetas inteligentes desde el navegador web Mozilla Firefox.

Para dar solución al problema existente se ha tomado como **objetivo general**: desarrollar un componente que permita la interacción entre el navegador web Mozilla Firefox y los lectores de tarjetas inteligentes para acceder a los servicios en línea.

Para dar solución a la interrogante se plantea la siguiente **idea a defender**: La implementación de un `add-on` para el navegador web Mozilla Firefox que interactúe con los lectores de tarjetas inteligentes permitirá acceder a los servicios en línea que utilicen esta tecnología.

Dada la hipótesis planteada anteriormente se pueden definir como **variables de la investigación**:

Variable Independiente: implementación de un componente (`add-on`).

Variable dependiente: la interacción entre los lectores de tarjetas y el Mozilla.

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

- Caracterizar las tecnologías relacionadas con tarjetas inteligentes.
- Caracterizar el desarrollo de componentes para el navegador web Mozilla Firefox.
- Caracterizar el estándar PC/SC.
- Elaborar la documentación y diagramas de ingeniería de software necesarios para el análisis y diseño de la solución.

- Desarrollar un componente que se integre con el navegador web Mozilla Firefox para la comunicación con los lectores de tarjetas inteligentes.
- Desarrollar un prototipo de servicio en línea que utilice esta solución
- Realizar las pruebas de calidad al componente desarrollado.
- Realizar las pruebas de calidad al prototipo.

Los métodos utilizados en el desarrollo de este trabajo están determinados por el objetivo general y las tareas de investigación concebidas.

Del Nivel Teórico:

Método Histórico*:* permitió consultar bibliografía referente al tema de investigación, toda su trayectoria, evolución y comportamiento.

Método Lógico: condujo a estructurar la documentación investigada de una manera organizada y cronológica, para así tener un mejor entendimiento de la misma.

Método de la Modelación*:* facilitó la creación de modelos, representando de manera gráfica parte del contenido de la investigación

Método Sistémico*:* se utilizó este método para estudiar la integración de las tecnologías utilizadas, mediante la determinación de sus componentes, así como la relación entre ellos. Esta relación determina por un lado la estructura y la jerarquía de cada componente y por otra parte su dinámica, siendo también la expresión del comportamiento del sistema como totalidad en que un componente depende de otro u otros.

Del Nivel Empírico:

Método de la observación*:* se puso en práctica, al concebir de forma consciente la planificación de la investigación, orientada hacia el logro del objetivo determinado.

Método experimental*:* el experimento permitió estudiar el objeto, crear las condiciones para verificar la hipótesis.

Apoyo sobre los procesos de:

Análisis: permitió la división mental de lo investigado, en relaciones y componentes para una mejor comprensión.

Síntesis: posibilitó establecer mentalmente la unión entre las partes previamente analizadas, resaltando sus principales características, conceptos y relaciones.

El trabajo se encuentra estructurado de la siguiente forma:

Capítulo 1: Fundamentación teórica de la investigación: este capítulo contiene una base teórica para entender el problema planteado, en él se describen los conceptos fundamentales relacionados con tarjetas inteligentes y las tecnologías relacionadas con los servicios en línea y componentes para el navegador Mozilla Firefox.

Capítulo 2: Propuesta de solución del componente: se presentan las fases de Exploración y Planificación definidas por la metodología Extreme Programing para dar solución al problema científico. Se seleccionan las herramientas y tecnologías para desarrollar la solución. Se identifican las Historias de Usuarios y los requerimientos no funcionales, se realiza el Plan de Iteraciones y Plan de Entregas.

Capítulo 3: Implementación y prueba del componente: se da cumplimiento a los planes trazados a través de las fases: Iteraciones a primera liberación y Producción, se codifica la solución diseñada y finalmente se realizan las pruebas de aceptación con el cliente.

Tareas	Responsable	Fecha de Entrega
Caracterizar las tecnologías relacionadas con tarjetas inteligentes.	Yadira y Ángel	1 de noviembre del 2010
Caracterizar el desarrollo de extensiones para el navegador web Mozilla Firefox.	Yadira y Ángel	10 de noviembre del 2010
Caracterizar el estándar PC/SC.	Yadira y Ángel	15 de mayo del 2011
Desarrollar la documentación y diagramas de ingeniería de software necesarios para el análisis y diseño de la solución.	Yadira	1 de junio del 2011
Desarrollar un componente que se integre con el navegador web Mozilla Firefox.	Yadira y Ángel	10 de junio del 2011
Proponer un prototipo de servicio en línea que utilice esta solución.	Ángel	20 de junio del 2011

Realizar las pruebas de calidad al componente desarrollado de Identificación y Seguridad Digital.	Yadira	10 de junio del 2011
Realizar las pruebas de calidad al prototipo.	Ángel	10 de junio del 2010

Tabla 1. Tareas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN

Introducción

El desarrollo conseguido en los últimos años por las tarjetas inteligentes estimula a pensar en cómo se pudiera fomentar la seguridad de los servicios en línea en la red de redes mediante el uso de las mismas. Por lo que se hace necesario encontrar una manera para interactuar con las tarjetas desde el navegador web Mozilla Firefox, ya que este es uno de los de mayor popularidad y su utilización ha crecido notablemente en los últimos años.

En el presente capítulo se hará un estudio de los principales retos o desventajas que asumen los prestadores de servicios en línea a través de la web y como pueden estos erradicarse mediante el uso de las tarjetas inteligentes. Para ello se analizarán las características y aplicaciones de las mismas, así como los estándares internacionales y las tecnologías más utilizadas. Se analizarán las soluciones factibles existentes y las tecnologías y herramientas que puedan servir en la implementación de la propuesta de solución.

1.1 Aplicaciones y características de las tarjetas inteligentes.

Las tarjetas inteligentes son dispositivos que contienen un chip en su interior, por lo que también se les conoce como tarjetas con circuito integrado (TCI). La presencia de dicho circuito posibilita la ejecución de cierta lógica programada. Por su parte el plástico que contienen, puede ser impreso para contener información adicional y visible a los ojos de los humanos acerca del portador o la entidad que las emite.

1.1.1 Evolución y desarrollo alcanzado por las tarjetas inteligentes

Debido a la avanzada tecnología que se presenta en el mundo se hace necesario que constantemente se esté en evolución y aprovechando las ventajas que esta nos ofrece en cualquiera de los servicios donde se aplique y que posea la necesidad del manejo de la información en forma oportuna, rápida y sin límites de papeleos o demoras para

su consecución, se hace necesario el conocimiento general de la tecnología de tarjetas inteligentes por parte de la comunidad para su uso masivo.

¿Qué son las tarjetas inteligentes?

Son tarjetas de plástico similares en tamaño y otros estándares físicos a las tarjetas de crédito que llevan incrustado un circuito integrado. Éste puede ser de solo memoria o contener un microprocesador (CPU) con un sistema operativo que le permite una serie de tareas como: almacenar, encriptar información y leer y escribir datos, como un ordenador.

A principios de los años 90 las tarjetas inteligentes inician su despegue al empezar la telefonía móvil GSM, inicialmente con tarjetas con 1K de memoria. La Fase 1 de GSM requería muy poca capacidad de memoria.

Se empiezan a usar de forma masiva al iniciarse la telefonía GSM. Se comenzó directamente con GSM Fase 2 en septiembre de 1995 empleando tarjetas con 8K de memoria. A finales de 1997 aparecieron las tarjetas de 16K, algunas de las cuales ya implementaban GSM Fase 2+ con SIM ApplicationToolkit. A lo largo de 1999 aparecen diferentes tarjetas Java, aunque no son compatibles entre sí, y luego, las tarjetas de 32K.

El objetivo de la tarjeta inteligente es ofrecer a los clientes un servicio con muchos más beneficios que le facilite su desenvolvimiento diario permitiendo su poseedor adquirir bienes y servicios dentro de una red de entidades.

En los últimos años hemos visto evolucionar el sector de las tarjetas inteligentes desde el momento en que un circuito integrado fue incluido en ellas. El abanico de posibilidades ofrecidas por ellas se multiplica cada día, en parte impulsado por las nuevas posibilidades que presentan las tarjetas inteligentes frente a las tarjetas convencionales:

- Permite la utilización de una única tarjeta para aplicaciones variadas y muy distintas.

- Generan menores costes por transacción que las tarjetas de plástico convencionales. El coste por tarjeta también se reduce debido, sobre todo, al mayor tiempo de vida de la tarjeta y a que ésta puede actualizarse.
- Las tarjetas inteligentes permiten un alto grado de seguridad en las transacciones con ellas efectuadas frente a las tarjetas convencionales.

1.1.2 Estructura de una tarjeta inteligente

Una tarjeta inteligente contiene un microprocesador de 8 Bytes con su `CPU`, su `RAM` y su `ROM`, su forma de almacenamiento puede ser `EPROM` o `EEPROM` (ver Figura 1), el programa `ROM` consta de un sistema operativo que maneja la asignación de almacenamiento de la memoria, la protección de accesos y maneja las comunicaciones. El sendero interno de comunicación entre los elementos (`BUS`) es totalmente inaccesible desde afuera del chip de silicona mismo por ello la única manera de comunicar está totalmente bajo control de sistema operativo y no hay manera de poder introducir comandos falsos o requerimientos inválidos que puedan sorprender las políticas de seguridad. Las dimensiones y ubicación de las mismas están especificadas en el estándar ISO 7816 - 2.

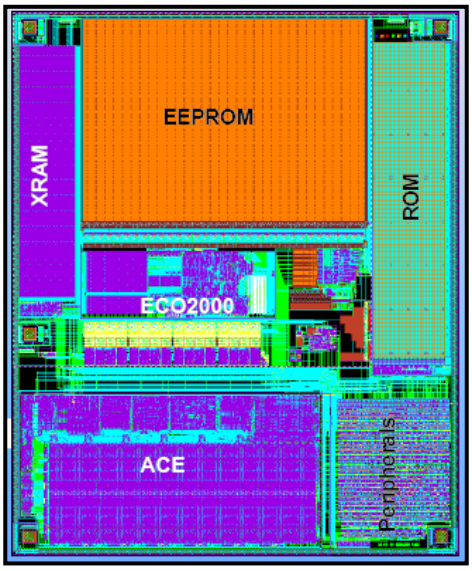


Ilustración 1. Distribución física de los tipos de memorias de un chip de `SmartCard` (Almeida Sotolongo, 2009)

La interfaz de comunicación de las tarjetas inteligentes, están hecha para comunicarse con un dispositivo que acepte tarjetas (CAD – Card Acceptance Device) a través de un conjunto de 8 pines.

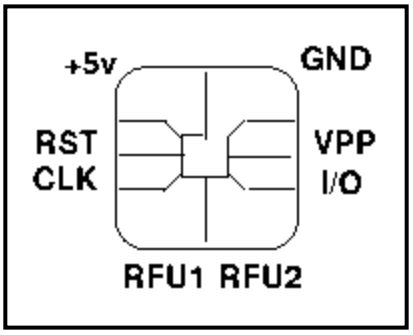


Ilustración 2. Composición del chip (Almeida Sotolongo, 2009)

- **+5V** – **GND**: suministro de energía.
- **I/O**: datos.
- **RST**: reset.
- **CLK**: señal del reloj (lo usual es < 5MHz).
- **VPP**: señal usada para suministrar energía a alguna área en particular de la tarjeta, o para borrar la memoria no-volátil de la tarjeta. Por esencia la idea es que es controlable por software.
- **RFU1** – **RFU2**: reservados para uso futuro.

La tarjeta sólo reacciona a los requerimientos de datos externos, nunca inicia por sí sola una comunicación. Todo el protocolo de comunicaciones, dimensiones, resistencia, y otros, está claramente establecido en el estándar ISO-7816.

Dentro de la categoría de las tarjetas inteligentes con microprocesador se encuentran las llamadas JavaCards o SmartCards. Una JavaCard es una SmartCard capaz de ejecutar programas desarrollados en Java. En pocas palabras, una SmartCard es

una tarjeta con microprocesador que puede ejecutar programas (llamados Applets) escritos en un subconjunto del lenguaje Java.(Almeida Sotolongo, 2009)

1.1.3 Protocolo de comunicación entre las tarjetas inteligentes y los lectores conectados a las computadoras

APDU

Unidad de datos de protocolo de aplicación (Application Protocol Data Unit), es la unidad de comunicación entre un lector y una tarjeta. Su estructura está definida en el estándar ISO 7816, existiendo dos tipos de categorías de APDU, APDU Command (Comando APDU) y APDU Response (APDU Respuesta). (ISO Organization, 2005)

1 APDU Command (C-APDU: este comando es usado por el lector para enviar información a la tarjeta.)

Encabezado (obligatorio)				Cuerpo (opcional)		
CLA	INS	P1	P2	Lc	Data Field	Le

Tabla 2. Estructura Comando APDU.

CLA: Clase de instrucción. Indica la clase y la estructura.

INS: Código de instrucción. Especifica la instrucción del comando.

P1, P2: Parámetros de instrucción. Proveen más información sobre la instrucción.

LC: Número de bytes en el Data Field del APDU.

Data Field: Secuencia de bytes con información.

LE: Cantidad máxima de bytes esperados como respuesta.

2 APDU Response(R-APDU: este comando es usado por la tarjeta para responder al comando enviado por el lector.

Cuerpo (opcional)	Código Respuesta (obligatorio)

Data Field	SW1	SW1

Tabla 3. Estructura APDU Respuesta.

`Data Field`: Secuencia de bytes con información.

"`SW1`, `SW2`: Status Word (palabra de estado): Denotan el estado del procesamiento del comando en la tarjeta."(Almeida Sotolongo, 2009)

1.1.4 Aplicaciones de las tarjetas inteligentes

Hoy en día las tarjetas inteligentes tienen un gran número de aplicaciones entre las que se puede mencionar el control de acceso. Este tipo de aplicación está ligada a puertas automatizadas que permiten o impiden el paso físico de una persona a un área determinada. También son muy utilizadas como un monedero electrónico, a finales del siglo pasado se registraban ya 65 millones que se usaban con este fin. Estas disponen normalmente de un fichero protegido que almacena un contador de saldo y comandos para decrementar e incrementar el mismo (esto último sólo con claves de seguridad especiales). Con esta aplicación, el `chip` de la tarjeta inteligente puede ser 'cargado' con dinero (en terminales autorizados que dispongan de las claves de seguridad). Este dinero virtual puede ser utilizado en parquímetros, máquinas expendedoras u otros mercados. El monedero electrónico extiende el uso de las tarjetas a acciones tan simples como el pago de televisión, telefonía móvil, transporte público, acceso a sitios web por solo mencionar un par de ejemplos.

Otra importante aplicación en la actualidad, es su vinculación a la firma digital de documentos, o sea que permiten almacenar un certificado digital dentro de la tarjeta y firmar con él, documentos electrónicos, sin que en ningún momento el certificado y su clave privada salgan del almacenamiento seguro en el que están confinados.

El uso de la tarjeta anula la necesidad de tener que compartir una inmensa base de datos y tener que hacer réplicas periódicas. Esta misma facilidad es la que se aprovecha al utilizar las tarjetas inteligentes en algunas clínicas y en sistemas nacionales de salud donde se está implementando un sistema de identificación de

pacientes y control de los datos del historial clínico o información relativa a enfermedades crónicas o alérgicas dentro de la tarjeta personal.

El DNI español es una tarjeta de policarbonato con chip integrado y se utiliza tanto para la identificación presencial del portador como para la identificación electrónica mediante los certificados contenidos en la memoria no volátil del chip. También se utiliza en el sector privado para: acceder a sitios web, firma de contratos, verificación de autenticidad de documentos y en el sector público para la declaración anual de impuestos, interacción con la Administración Pública para la obtención de formularios, servicios en línea y registros criminales. Como se ha descrito las aplicaciones de las tarjetas inteligentes son muy variadas.

1.1.5 Estándar PC/SC

PC / SC (Personal Computer/Smart Card) es un estándar para las tarjetas inteligentes de acceso en plataformas Windows (incluido en Windows 2000).

La especificación de interoperabilidad para las comunidades y el personal de sistemas informáticos han sido desarrollados para facilitar la introducción de tarjetas inteligentes en el mundo de las computadoras. La principal ventaja de PC/SC es que las aplicaciones no tienen que estar al tanto de los detalles relativos al lector de tarjetas inteligentes con el fin de comunicarse con ellas. Por otra parte, la aplicación puede funcionar con cualquier lector que cumpla con este estándar.

El API de PC/SC está incorporada en sistemas Microsoft Windows 200x/XP y Microsoft Windows NT/9x. También hay una implementación libre, de código abierto, llamada PC/SC Lite (proyecto MUSCLE) para sistemas operativos GNU Linux.

La especificación se divide en 10 partes que contienen los requisitos detallados de interoperabilidad de dispositivos compatibles, información de diseño, interfaces de programación y otras. (PC/SC Workgroup, 2010)
Parte 1. Introducción y visión general de la arquitectura.

Parte 2. Requisitos de interoperabilidad para las tarjetas y los lectores.

Parte 3. Requisitos de interoperabilidad para los lectores conectados.

Parte 4. Consideraciones de diseño e información de referencia de los lectores.

Parte 5. Definición de la interfaz del Resource Manager.

Parte 6. Definición de la interfaz del Service Provider.

Parte 7. Consideraciones de diseño para el desarrollo de aplicaciones.

Parte 8. Recomendación para la implementación de servicios de seguridad y privacidad con tarjetas inteligentes.

Parte 9. Lectores con capacidades extendidas.

Parte 10. Lectores con capacidades de entrada de PIN de seguridad.

1.2 Tecnologías a usar durante el desarrollo del add-ons y componentes para Mozilla Firefox

*1.2.*1 XPCOM

XPCOM (en inglés Cross Platform Component Object Model) es un marco de trabajo para escribir software multiplataforma similar a Microsoft COM. Tal como una aplicación, usa un conjunto de librerías para cargar y manipular componentes XPCOM, estos pueden ser escritos en lenguajes como C, C++, JavaScript y pueden ser usados desde estos mismos lenguajes con extensiones para Perl y Python.

XPCOM utiliza las siguientes funciones para el desarrollador de este software: componente de gestión, archivo de abstracción, objeto de paso de mensajes y gestión de la memoria.

La flexibilidad de reutilizar los componentes XPCOM de la biblioteca Gecko y desarrollar nuevos componentes que se ejecutan en distintas plataformas facilita el desarrollo rápido de aplicaciones y los resultados en una aplicación que es más productivo y fácil de mantener.

1.2.3 JavaScript

JavaScript es un lenguaje de programación que se define como orientado a objetos. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de

un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor. Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF y aplicaciones de escritorio es también significativo.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Tradicionalmente se venía utilizando en páginas web HTML (por sus siglas en inglés, HyperText Markup Language) para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor.

1.2.4 JSCtypes

Js-ctypes es una biblioteca para trabajar en lenguaje JavaScript en Mozilla. Proporciona los tipos de datos compatibles en C y permite que el código JavaScript pueda llamar a funciones C implementadas en las bibliotecas compartidas y así poder usar dichas funciones en JavaScript. La interfaz y la aplicación se inspiran en la ctypes de módulos de Python.

El objetivo principal de la biblioteca es ayudar a los desarrolladores a evitar la construcción binaria en (C++) de los componentes XPCOM cuando con sólo un simple envoltorio se resuelve el problema.

Algunos de estos problemas pueden ser:

- El desarrollador quiere funcionalidades no integradas en la plataforma Mozilla, pero que son soportadas por el sistema operativo.
- EL desarrollador quiere usar una colección de librerías en una extensión o aplicación XUL.
- EL desarrollador tiene métodos de código nativo por razones de rendimiento.

La respuesta habitual a estos problemas es la creación de un sistema binario (C + +) que actúa como un contenedor para las características nativas, pueden estar expuestos a través de JavaScript y XPCOM. Sin embargo, el proceso de construcción binaria de los componentes XPCOM es considerablemente más duro que los componentes de JavaScript. Para muchos casos, es demasiado trabajo.

El objetivo de js-ctypes es permitir a los desarrolladores declarar métodos en las bibliotecas y luego exponer los métodos, llamando a las funciones a través de código JavaScript. Los desarrolladores pueden crear pura biblioteca de los envoltorios alrededor de bibliotecas JavaScript binario - sin hacer binarios envoltorios de XPCOM.

1.2.5 XUL (Extensible User interface Language)

"XUL fue creado para facilitar y acelerar el desarrollo del navegador Mozilla. Es un lenguaje XML, por lo tanto todas las características del XML están también en XUL. Este lenguaje proporciona la habilidad de crear la mayoría de los elementos encontrados en las interfaces gráficas modernas. Es tan general que puede ser aplicado a las necesidades específicas de ciertos dispositivos y tan poderoso que los desarrolladores pueden crear sofisticadas interfaces con éste.

Algunos elementos que pueden ser creados son:

- Controles de entrada tales como cuadros de texto y cajas de chequeo.
- Barra de herramientas con botones u otros contenidos.
- Menús en barras de menú o menú emergente.
- Pestañas de diálogo.
- Árbol de información jerárquica o tabulada.
- Teclas de accesos directo.

El contenido mostrado puede ser creado desde el contenido de un archivo XUL o con datos de una fuente de datos. En Mozilla, tales fuentes de datos son utilizadas para los mensajes de una cuenta de correo, los marca-páginas y los resultados de búsqueda. El contenido de los menús, árboles y otros elementos pueden ser llenados con estos datos, o con sus propios datos suministrados en un archivo con extensión RDF." (2011)

1.2.5 Desarrollo de Add-ons

En este epígrafe se exponen los elementos necesarios para la creación de un `add-on`. Se detallan las carpetas y los archivos que debe contener el mismo para instalarlo en el navegador Mozilla Firefox, así como los archivos que intervienen en la interfaz, la lógica, entre otras de sus características.

1.2.5.1 Carpetas que componen un add-on

Existe una estructura obligatoriamente definida que los `add-ons` deben cumplir. Esta estructura define el nombre de las carpetas, la estructura jerárquica, el tipo de archivos que contendrá cada una y el lugar donde se deberán almacenar. A continuación se describe esta estructura.

1.2.5.2 Carpeta raíz

Aquí es donde se crean las demás carpetas y archivos que compondrán el `add-on`. El nombre de esta carpeta raíz deberá corresponder con el archivo de instalación, como se describirá más adelante.

1.2.5.3 Carpeta chrome

Dentro de la carpeta raíz se encuentra `chrome`. Esta carpeta es necesaria para poder cargar los archivos `XUL` del navegador. El objetivo de chrome es que los archivos XUL se almacenen bajo una dirección que no varíe de plataforma en plataforma, sino que sea estándar para todas las implementaciones, independientemente del sistema operativo. Por ejemplo, el archivo `browser.xul` del navegador Mozilla Firefox lo podemos encontrar en `chrome://browser/content/browser.xul`. Además mediante el nombre de esa carpeta chrome, se le indica al navegador Mozilla Firefox, que lo que se encuentra dentro de ella deberá ser interpretado por el navegador y gestionado de forma especial.

Dentro de la carpeta chrome se encuentran los archivos correspondientes para la creación de la interfaz, la lógica y los estilos.

Carpeta content

La carpeta `content`, contenida dentro de `chrome`, contendrá todo el código para definir la interfaz, a través de los archivos `XUL`, y la lógica necesaria para implementar el comportamiento de la extensión, mediante `JavaScript`.

Carpeta locale

Dentro de la carpeta locale, contenida dentro de chrome, se almacenan los archivos de configuración para distintos idiomas. De esta forma se permite generar una extensión multilenguaje, traduciendo la interfaz a un idioma determinado.

Carpeta skin

Dentro de la carpeta `skin`, contenida dentro de chrome, se guardan los estilos de Style Sheet Cascading o conocido como archivos `CSS`. También aquí se encuentran las imágenes a utilizar en la interfaz.

1.2.5.4 Carpeta default

La carpeta `default`, que se encuentra dentro de la carpeta raíz, al mismo nivel que `chrome`, permite definir los textos y mensajes que serán utilizados en el `add-on`. Esta carpeta no es obligatoria, ya que los textos y mensajes pueden estar embebidos dentro de código `JavaScript`. Esta carpeta tiene como objetivo facilitar el mantenimiento y la actualización de los mensajes.

1.2.5.5 Carpeta components

Se encuentra dentro de la carpeta raíz, al mismo nivel de `chrome` y `default`. Aquí es donde se encuentran las librerías utilizadas por el `add-on`.

1.2.6 Archivos de configuración

A continuación se describirán las dos de los archivos más importantes dentro del `add-on`.

1.2.6.1 Install.rdf

Este archivo es el instalador del `add-on` y se debe encontrar en la carpeta raíz y no dentro de otra carpeta. El mismo contiene información sobre el `add-on`, por ejemplo, versión desarrollada, autor y sitio oficial del proveedor, así como versiones compatibles, actualizaciones, la GUID de la aplicación, entre otros.

Para ver con detalles cómo debe estar configurado un archivo `install.rdf`, así como la descripción de cada uno de sus elementos ver el Anexo 13.

1.2.6.2 Chrome.manifest

Dentro de este archivo se declarará el tipo de material que se encuentra dentro del paquete chrome, el nombre del paquete, y la ruta hacia el mismo. Además se define cual es el archivo `XUL` que se va a fusionar con el archivo `browser.xul` del navegador que viene por defecto con el Mozilla Firefox.

El tipo de material puede ser `content`, como se ha mencionado anteriormente son los archivos `XUL`, `JavaScript`, entre otros; `locale`, archivos de definición de idiomas, y `skin`, archivos CSS e imágenes que conforman el diseño de la interfaz.

Para ver con detalles cómo debe estar configurado un archivo `chrome.manifest`, así como la descripción de cada uno de sus elementos ver el Anexo 13.

1.2.7 Generar el instalador

Una vez que se crearon todos los archivos dentro de sus carpetas correspondientes, es necesario crear un instalador del add-on. Para esto sencillamente se tiene que crear un archivo, con el nombre del `add-on`, compactado con extensión ZIP con todo el

contenido que se encuentra debajo de la carpeta raíz, y luego renombrar esta extensión a `XPI`.

1.3 Análisis de otras soluciones

1.3.1 OnlineSmartCardPlatform.

Esta solución fue creada en el departamento de tarjetas inteligentes del Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI). Su papel principal es el de mediar entre los sitios web que presten servicios en línea, a través del uso exclusivamente del navegador web Internet Explorer, y los lectores de tarjetas inteligentes establecidos en la estación del cliente.

Esta solución está basada en la arquitectura de dos capas. La primera, identificada como capa cliente, contiene los componentes para comunicarse con el lector de tarjetas inteligentes y con el `middleware` en el servidor; la segunda, identificada como capa servidor, resuelve las solicitudes que vienen desde el cliente, además de gestionar los `middleware` que se encuentran asociados a la plataforma y las funcionalidades contenidas en ellos accediendo a sus librerías.

El patrón arquitectónico utilizado en esta solución el MVC, conocido también como Modelo Vista Controlador.

Esta solución se compone de:

- Una interfaz que permite la interacción con el usuario de manera agradable y sencilla.
- Un servidor donde se encuentran todos los `middlewares` correspondientes y necesarios para poder establecer la comunicación con los `applets` de las tarjetas inteligentes
- Un componente que maneja la tecnología `ActiveX`, y donde se encuentran las funcionalidades de listar lectores conectados, identificar cuando existe una navegación abierta con una tarjeta, gestionar el envío de comandos `APDU` hacia la tarjeta, así como obtener los que esta envía.

1.3.2 ActiveX

ActiveX es una tecnología orientada a la Internet desarrollada por Microsoft, generalmente ejecutada a través de navegadores como Internet Explorer. Agregan dinamismo a las páginas web y en ocasiones aporta al diseño de la misma.

Los controles ActiveX son pequeños programas que se incluyen dentro de las páginas web para dotarlas de mayores funcionalidades. Están conceptualmente divididos en servidores, objetos que hacen que sus métodos y propiedades estén disponibles para los clientes, aplicaciones que usan los métodos y propiedades expuestos por el servidor.

1.3.3 Coesys eGov 2.0

El objetivo principal de este producto es que los ciudadanos puedan conectarse de forma instantánea a servicios electrónicos del gobierno mediante la identificación electrónica de tarjetas inteligentes.

Esta solución de autenticación segura no requiere ningún software en el ordenador de los usuarios, funciona con todo navegador en cualquier sistema operativo, que permite la conectividad plug-and-play[4]desde un ordenador.

Algunas de sus características principales son: servicios de conectividad de tarjetas inteligentes, servicio de autenticación y 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 que proporciona un vínculo entre la aplicación de la tarjeta de identificación electrónica y la aplicación de servidor de autenticación. La solución **Coesys eGov 2.0** se compone de:
Un servidor: para servicios de post-emisión y autenticación.

[4]Es un sistema que permite conectar cualquier dispositivo de *hardware* al ordenador, sin tener que incorporar o instalar ningún controlador, pues la configuración se realiza de forma automática. Esto supone un aumento en la facilidad de instalación y configuración de nuevos periféricos.

1.3.4 SConnect

SConnect es un complemento para los navegadores más importantes, es compatible con los sistemas operativos Windows, Mac OSX y Linux. Su objetivo principal es el de proporcionar un puente de conexión entre el JavaScript, que corre en la página web de un navegador y la tarjeta inteligente, permitiendo la conectividad entre estas últimas aplicaciones y los servicios web.

SConnect consiste en dos partes:
Una extensión del navegador web que 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.

Una librería JavaScript que permite a los desarrolladores de aplicaciones web tener acceso a tarjetas inteligentes mediante SConnect.

SConnect 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.
El aspecto de conectividad de SConnect es la clave en la innovación de la solución de **Coesys eGov 2.0**. Además trae integrado un paquete de herramientas para desarrollar Smart Card Aware Web Applications(SAWA por sus siglas en inglés).
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 y la interoperabilidad con varias tarjetas inteligentes.
Además de todo lo anteriormente planteado, estos productos poseen como desventajas: el precio altísimo en el mercado del software y los mecanismos de incorporación de nuevos middleware no son transparentes.

1.4 Metodología, Tecnologías y Herramientas propuestas para la solución

1.4.1 Metodologías de desarrollo

El éxito de un proyecto depende fundamentalmente en la elección de una metodología de software adecuada. Esta selección depende principalmente de dos factores, el tipo de proyectos y la cultura que exista en la empresa.

Dentro de las metodologías de desarrollo existen dos grandes grupos, las conocidas como Tradicionales y Ágiles. Las primeras se caracterizan por un uso intensivo de documentación durante todo el ciclo de vida del proyecto y es recomendada para los proyectos con grandes equipos de desarrollo. La metodología más utilizada dentro de este grupo es RUP (en inglés, Rational Unified Process), resultado de varios años de desarrollo y uso práctico en el que se han unificado varias técnicas de desarrollo. Por su parte las ágiles enfatizan las comunicaciones cara a cara en vez de la documentación. A continuación se presenta una tabla que muestra las principales diferencias entre estos dos grupos de metodologías.

Metodología Ágiles	**Metodologías Tradicionales**
Especialmente preparada para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado , con pocos principios	Procesos mucho más controlado , con numerosas políticas o normas
No existe contrato tradicional , o al menos es bastante flexible	Es un contrato prefijado
El cliente es parte del equipo de desarrollo	EL cliente interactúa con el equipo de desarrollo mediante reuniones

Grupos pequeños(menos de 10) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Menos artefactos
Pocos roles	Menos roles
Menos énfasis en la arquitectura de software	La arquitectura de software es esencial y se expresa mediante modelos

Tabla 4. Comparación de Metodologías Ágiles y Tradicionales.

Por todo lo anteriormente planteado se decideguiar el proceso de desarrollo del componentecon usando metodologías ágiles, pues estas se adecuan más a las condiciones objetivas del software y del equipo de trabajo.

Ahora bien, dentro de este grupo de metodologías se necesita escoger la mas idónea, por lo que se hizo un estudio de todas las existentes, dentro de las cuales se encuentran algunas como :

- **XP:** está diseñada para entregar el software que el cliente necesita, en el momento que lo necesita. Además promueve el uso de prácticas para aumentar la productividad del equipo de desarrollo y mejorar la adaptabilidad a los frecuentes cambios dentro del ciclo de vida del proyecto.

Dentro de las ventajas que nos brinda la metodología XP podemos encontrar que es la más apropiada para entornos volátiles, equipos de desarrollo pequeños (de 2 a 10 desarrolladores) y proyectos de alto riesgo. También permite una mejor adaptabilidad a los cambios, que se traduce en una reducción de costos, la planificación es a corto plazo y es más transparente para los clientes, ya que conocen las fechas de entrega de funcionalidades vitales para su negocio y permite tener retroalimentación continua de los usuarios a través de las entregas frecuentes. Esta metodología también posee

desventajas dentro de las que se encuentra la dificultad en muchas ocasiones para delimitar el alcance del proyecto con el cliente.

- **Scrum**. La intención es maximizar la realimentación sobre el desarrollo, se pueden corregir problemas y mitigar riesgos de forma temprana. Es muy habitual que Scrum se complemente con XP; en estos casos, Scrum suministra un marco de administración basado en patrones organizacionales, mientras XP constituye la práctica de programación, usualmente orientada a objetos y con fuerte uso de patrones de diseño.

- **FDD (en inglés, Feature Driven Development).** A diferencia de otras metodologías ágiles esta no cubre todo el ciclo de vida sino sólo las fases de diseño y construcción y se considera adecuado para proyectos mayores y de misión crítica. FDD no requiere un modelo específico de proceso y se complementa con otras metodologías. Enfatiza cuestiones de calidad y define claramente entregas tangibles y formas de evaluación del progreso.

Luego de una profunda búsqueda se determinó escoger XP como metodología de desarrollo.

1.4.1.1 Fundamentación de XP como metodología a utilizar.

Entre los aspectos fundamentales para la elección de esta metodología se encuentran:

- El tamaño del grupo de desarrollo es pequeño.
- Necesidad de resultados tangibles a corto plazo.
- El cliente se involucra más en lo que se está haciendo.

1.5 UML (Unified Modeling Language)

El Lenguaje Unificado de Modelado es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aun cuando todavía no es un estándar oficial, está apoyado en gran manera por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema

de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

1.5 Herramientas propuestas para la solución

1.6.1 Altova UModel

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

1.6.2 Microsoft Visual Studio 2010

Microsoft Visual Studio 2010 es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio 2010 permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

1.6.3 Developer Suite

Herramienta que nos brinda un ambiente favorable para el diseño y la implementación de `applets`, además nos posibilita simular las funcionalidades de los `applets` antes de ser instalados en las tarjetas inteligentes.

Conclusiones

Las tarjetas inteligentes constituyen una interesante alternativa al débil vínculo de los sistemas de autenticación basado en nombre de usuario y contraseña. El estudio de las principales características y aplicaciones de las tarjetas inteligentes, así como la experiencia adquirida en el trabajo con las mismas en el Centro de Identificación y Seguridad Digital, refuerzan la idea a defender de que la creación de una plataforma, que permita su utilización a través de la web, podría potenciar en gran medida la seguridad de los servicios en línea. Para esto es vital el desarrollo de un componente que garantice la interacción de los lectores de tarjetas conectados a la computadora con el navegador web Mozilla Firefox, por ser este uno de los más usados a nivel mundial.

En este capítulo se hizo un estudio de los estándares internacionales relacionados con las tarjetas inteligentes, las tecnologías y metodologías relacionadas con el desarrollo de componentes para servicios web, concluyendo que la metodología XP guiara el proceso de desarrollo, pues las características antes mencionadas se adaptan a las condiciones del equipo y el entorno de desarrollo.

A través del estudio de diversas tecnologías se define como lenguaje de programación para la implementación de las funcionalidades de la librería a la cual accederá el componente Visual C++ con Microsoft Foundation Classes (MFC por sus siglas en inglés), teniendo en cuenta las posibilidades que brinda de dotar al usuario de una mayor libertad en cuanto al uso de la plataforma a la cual se añadirá el add-on. Igualmente se utilizará el lenguaje `JavaScript` para el manejo de la extensión y los eventos en la página y `XUL` para manejar la interfaz visual del navegador Mozilla.

CAPITULO 2: PROPUESTA DE SOLUCIÓN DEL COMPONENTE

Introducción

La solución a desarrollar, debe ser fruto de un correcto análisis y amplia comprensión de todos los elementos que se relacionan con la elaboración del `add-on` que permita la interacción de los lectores de tarjetas inteligentes con el navegador Mozilla para acceder a servicios que se prestan a través de la web.

Con la elaboración de este capítulo se propone la solución al problema científico utilizando la metodología Extreme Programming normalmente conocida como XP.

La metodología XP ha revolucionado la comunidad de ingeniería de software, ya que hace prever una fuerte proyección industrial, se ajusta a proyectos en los cuales los equipos de desarrollo son pequeños, o tienen plazos reducidos, o están basados en nuevas tecnologías. Debido a las características del software que es producido por el Departamento de Tarjetas Inteligentes del Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas, el mismo ha asumido el uso de metodologías ágiles dentro de las que se incluye XP para guiar los procesos de desarrollo que se llevan a cabo en esta línea.

El objetivo que se persigue con la elaboración de este capítulo es mostrar la evolución de la solución durante las fases iniciales de Planificación y Exploración, además de presentar los diferentes artefactos generados en las mismas, los cuales serán premisas cruciales para la entrega final de la plataforma.

2.1 Metodología XP

XP, tal como fue abordado anteriormente, forma parte del conjunto de métodos ágiles que centran sus prioridades en las personas, no en los procesos. En la actualidad XP se proyecta a ser un modelo de desarrollo común, sencillo y adaptable a las características cambiantes y exigentes de empresas y clientes, es por ello que a continuación se presentan en forma resumida las características principales, las actividades, las prácticas, el ciclo de vida y los artefactos de esta metodología.

Prácticas

De forma aislada, cualquier práctica individual de XP tiene poco sentido, pero en conjunto, unas compensan las carencias que las otras puedan tener.

- Cliente en el equipo
- El juego de la Planificación
- Versiones Pequeñas
- Metáfora del Sistema
- Diseño Simple
- Pruebas Continuas
- Refactorización
- Programación por parejas
- Posesión colectiva del código
- Integración continua
- Semana laboral de 40 horas
- Estándares de Codificación

Variables

XP define cuatro variables para proyectos de software: coste, tiempo, calidad y ámbito.

Además de estas cuatro variables, el creador de la metodología XP, propone que sólo tres puedan ser establecidas por las fuerzas externas (jefes de proyecto y clientes),

mientras que el valor de la cuarta variable debe ser establecido por los programadores en función de las otras tres.

Artefactos esenciales en XP

Los principales artefactos de XP son: Historias de usuario, tareas de ingeniería, pruebas de aceptación, pruebas unitarias y de integración, plan de entrega y código.

Fases de XP

XP propone un ciclo de vida dinámico, donde se admite expresamente que, en muchos casos, los clientes no son capaces de especificar sus requerimientos al comienzo de un proyecto. Por esto, se trata de realizar ciclos de desarrollo cortos (llamados iteraciones), con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas, pero utilizando un conjunto de reglas y prácticas que caracterizan a XP (y que serán detalladas más adelante).

- **Fase de exploración**

Aquí se define el alcance general del proyecto. El cliente define lo que necesita, quedando plasmado en las historias de usuario. Los programadores estiman los tiempos de desarrollo en base a esta información.

- **Fase de planificación**

La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas. El resultado de esta fase es un Plan de Entregas.

- **Fase de iteraciones**

Es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración.

- **Fase de puesta en producción**

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. "*Si bien al final de cada*

iteración se entregan módulos funcionales y sin errores, puede ser deseable por parte del cliente no poner el sistema en producción hasta tanto no se tenga la funcionalidad completa. En esta fase no se realizan más desarrollos funcionales, pero pueden ser necesarias tareas de ajuste."(Reglas y Prácticas en extreme Programing, 2010)

2.2 Propuesta de solución

2.2.1 Metáfora o Propuesta de solución

"*Una metáfora es algo que todos entienden, sin necesidad de mayores explicaciones. La metodología XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, y guiar la estructura y arquitectura del mismo. Por ejemplo, puede ser una guía para la nomenclatura de los métodos y las clases utilizadas en el diseño del código. Tener nombres claros, que no requieran de mayores explicaciones, redunda en un ahorro de tiempo. Es muy importante que el cliente y el grupo de desarrolladores estén de acuerdo y compartan esta "metáfora", para que puedan dialogar en un "mismo idioma". Una buena metáfora debe ser fácil de comprender para el cliente y a su vez debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto.*"(Wells, 2009)

Teniendo en cuenta la descripción anterior del concepto hemos formulado la siguiente metáfora para nuestro sistema.

El componente o add-on para el navegador web Mozilla Firefox puede ser utilizado por cualquier cliente que posea un sitio web, que brinde servicios a través de internet y desee sustentar la seguridad y/o funcionalidad de los mismos, con el uso de tarjetas inteligentes.

El sitio que ofrece los servicios a través de la web brindará la posibilidad de descarga e instalación del componente en el navegador Mozilla Firefox, permitiendo la comunicación con la tarjeta inteligente que está en el lector conectado a la estación cliente. Una vez instalado el add-on se podrán realizar operaciones básicas como reconocer los lectores conectados a la máquina y establecer una conexión con el

seleccionado, notificar los eventos relacionados con la tarjeta y lo más importante transmitir comandos `APDU` y recibir las respuestas de la tarjeta.

2.2.2 Estructura de los archivos del add-on en la solución

- El archivo `sample.xul` se encuentra dentro de la carpeta `content`, y la misma dentro de `chrome`. Este es el encargado de llamar a las funcionalidades del `add-on`, en dependencia del evento que venga desde la interfaz.
- En la carpeta `component` se puede encontrar el archivo `loadcard.js` el cual accede directamente a las funcionalidades implementadas en la librería `OnlineSmartdCardPlatform.dll` situada en esta misma carpeta.
- Los archivos `chrome.manifest` y `install.rdf` son los encargados de manejar la información relacionada con la instalación de la solución, la versión del Mozilla que se va a utilizar y los nombres de los demás archivos, entre otros. Estos se encuentran situados en la carpeta raíz.

2.2.3 Modelo de dominio

Se ha procedido a crear un modelo de dominio donde se identificará los principales conceptos sociales y tecnológicos del entorno en que se desarrolla la interacción entre la tarjeta inteligente, el `add-on` y el servicio en línea porqueel problema científico que el componente pretende resolver no está determinado a partir de procesos bien definidos. (Ver Anexo 11)

Conceptos

Usuario: Persona portadora de la tarjeta inteligente que accede a la aplicación web a través de una computadora.

Sistema Operativo: Sistema operativo instalado en la computadora.

Navegador Web: Permite visualizar la información que contiene la página web y a través de él se accederá al servicio web, será la interfaz gráfica para la comunicación con el usuario.

Estación Cliente: Computadora que utiliza el usuario.

Lector de Tarjetas: Es un lector compatible con el estándar PC/SC, el cual sirve de mediador para la comunicación entre la estación cliente y la tarjeta inteligente.

Tarjeta Inteligente: Como se plantea anteriormente es un dispositivo de plástico similar en tamaño y otros estándares físicos a las tarjetas de crédito, presentan un circuito integrado, el mismo puede ser de sólo memoria o contener un microprocesador (CPU) con un sistema operativo que le permita una serie de funcionalidades como almacenar información, encriptar información, leer y escribir datos, similar a un ordenador.

Servicio en Línea: Hace referencia a la diversidad de servicios que se ofrecen por Internet y a los que se pueden acceder utilizando las tarjetas inteligentes, a través de la plataforma y un navegador web.

Applet: Es una aplicación instalada dentro de la tarjeta inteligente que permite gestionar la información que se almacena en ella.

2.4.3 Historias de Usuario

Las historias de usuario son utilizadas en XP para especificar los requisitos del software desde el punto de vista del cliente. Estas se caracterizan por establecer la descripción de los requisitos funcionales del cliente, describir pequeños trozos de funcionalidad que aportan valor al desarrollo de la aplicación, son asignadas a la persona o desarrollador encargado de la programación con un número de horas de Desarrollo estimado, establecen la prioridad en la ejecución del sistema de la funcionalidad descrita, son descritas entre el cliente y el analista; las historias de usuario guían la construcción de las pruebas de aceptación (casos de prueba).

	Historia de usuario
Número: HU_1	**Nombre de Historia de Usuario:** Acceder a la librería OnLineSmartCardPlatform.dll

Modificación de Historia de Usuario Número: Ninguna	
Usuario:	**Iteración asignada:** 1
Prioridad en negocio: Media	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos reales:** 1
Descripción: Para acceder a la librería `OnLineSmartCardPlatform.dll` el desarrollador debe llamar a la funcionalidad `ctypes.open ()` y ubicar la dirección donde se encuentra dicha librería.	
Observaciones: Si no se llama a la función `ctypes.open()` no se podrá acceder a las funcionalidades de la librería `OnLineSmartCardPlatform.dll` y se lanzará un error en el código.	

Tabla 5. HU_1 Acceder a la librería OnLineSmartCardPlatform.dll.

	Historia de usuario
Número: HU_2	**Nombre de Historia de Usuario:** Cerrar la conexión con la librería `OnLineSmartCardPlatform.dll`
Modificación de Historia de Usuario Número: Ninguna	
Usuario:	**Iteración asignada:** 1

Prioridad en negocio: Media	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos reales:** 1
Descripción: Una vez que el usuario termine de trabajar con la librería OnLineSmartCardPlatform.dll debe llamar a la función `ctypes.close()` para cerrarla.	
Observaciones: Si no se cierra la conexión librería OnLineSmartCardPlatform.dll se lanzará un error en el código	

Tabla 6. HU_2 Cerrar la librería OnLineSmartCardPlatform.dll.

Historia de usuario	
Número: HU_3	**Nombre de Historia de Usuario:** Escoger lectores disponibles conectados a la computadora del cliente
Modificación de Historia de Usuario Número: Ninguna	
Usuario:	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Descripción: Cuando el usuario accede a la página principal, esta le ofrece una relación de los lectores de tarjetas inteligentes conectados a la computadora, y le da la posibilidad de escoger uno para establecer la conexión con la tarjeta, si no el sistema escoge el primero de la lista por defecto.	
Observaciones: Si no existe ningún lector de tarjeta conectado a la computadora , el sistema muestra un aviso.	

Tabla 7. HU_3 Obtener lectores disponibles conectados a la estación cliente.

Historia de usuario	
Número: HU_4	**Nombre de Historia de Usuario:** Gestionar comunicación de comandos APDU entre el servidor y el cliente web.
Modificación de Historia de Usuario Número: Ninguna	
Usuario:	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Descripción: Cuando el usuario solicita un servicio, el cliente se encarga de enviarlo al servidor para que haga una petición al `middleware` responsable, el mismo envía un primer comando `APDU` comenzando la comunicación de comandos y repuestas `APDU` entre el `middleware` y el `applet` de la tarjeta	
Observaciones: Si la comunicación entre el cliente y el servidor falla, se muestra un mensaje de aviso al usuario.	

Tabla 8. HU_4 Gestionar comunicación de comandos `APDU` entre el servidor y el cliente web.

	Historia de usuario
Número: HU_5	**Nombre de Historia de Usuario:** Enviar y recibir comandos `APDU` de la tarjeta.
Modificación de Historia de Usuario Número: Ninguna	
Usuario:	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 2.8
Riesgo en desarrollo: Medio	**Puntos reales:** 2.5
Descripción: Cuando existe una conexión abierta entre un lector y una tarjeta inteligente, el usuario hace una petición de un servicio mediante la página web que es procesada por el servidor y este envía los correspondientes comandos `APDU` que son transmitidos a la tarjeta a través de un componente cliente, que accede a la capa que se comunica con la tarjeta. A este comando la tarjeta siempre devolverá una respuesta que puede ser satisfactoria, la información solicitada o un mensaje de error. La tarjeta procesa el comando `APDU` enviado y manda la respuesta, que luego el componente cliente se encargará de enviar al servidor nuevamente hasta que termine la operación.	
Observaciones: Si la tarjeta no está conectada al lector la aplicación notificará un mensaje de error.	

Tabla 9. HU_5 Enviar y recibir comandos `APDU` de la tarjeta.

	Historia de usuario
Número: HU_6	**Nombre de Historia de Usuario:** Abrir conexión con el lector de tarjeta.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario:	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 3
Riesgo en desarrollo: Medio	**Puntos reales:** 3
Descripción: Luego que el usuario haya accedido a la página principal, se deberá establecer una conexión con la tarjeta que permitirá iniciar la comunicación de la plataforma con la misma comenzando siempre desde el servidor, para que el usuario pueda acceder a los servicios que brinda la aplicación. Una vez establecida la comunicación con el lector el sistema le mostrará al usuario mediante un mensaje si la conexión con la tarjeta se ha establecido. Esta tiene dos posibles estados: conectada y desconectada.	
Observaciones: Si no existiera ningún lector conectado a la computadora el sistema notificaría un mensaje de error.	

Tabla 10. HU_6 Gestionar comunicación con la tarjeta inteligente.

Historia de usuario	
Número: HU_7	**Nombre de Historia de Usuario:** Verificar estados (insertada/retirada) de una tarjeta en un lector.
Modificación de Historia de Usuario Número: Ninguna	
Usuario:	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 3
Riesgo en desarrollo: Medio	**Puntos reales:** 3
Descripción: Una vez que el usuario haya accedido a la página principal y lector y la tarjeta estén conectados a la computadora, la aplicación web deberá notificarle al usuario que la tarjeta ha sido insertada. En caso de que el usuario la retire, se le deberá notificar que la tarjeta ha sido retirada.	
Observaciones:	

Tabla 11. HU_7 Realizar operaciones de un `middleware` en el servidor.

2.4.4 Requerimientos no funcionales

Requerimientos mínimos de hardware:

- Lector de tarjetas incorporado a la PC que cumpla con el estándar PC/SC versión 1.0 o superior.

Requerimientos de software:

- Para el funcionamiento del componente se necesita que estén instalados en la PC cliente los driver del lector de tarjetas.
- En la PC cliente se podrá utilizar desde Microsoft Windows XP en adelante.
- **Portabilidad:**

    - El complemento debe ser compatible con cualquier lector de tarjetas que cumpla con el estándar PC/SC.
    - El componente debe ser funcional para el navegador Mozilla Firefox versión superior a la 3.6

2.4.5 Arquitectura

Para el desarrollo de la solución se utilizó una arquitectura basada en componentes, la misma consiste en una rama de la Ingeniería de software en la cual se trata con énfasis la descomposición del software en componentes funcionales. Esta descomposición permite convertir componentes pre-existentes en piezas más grandes de software.

Este proceso de construcción de una pieza de software con componentes ya existentes, da origen al principio de reutilización del software, mediante el cual se promueve que los componentes sean implementados de una forma que permita su utilización funcional sobre diferentes sistemas en el futuro.

Basado en los 5 principios de Clemens Szyperski [5]and David Messerschmitt [6]se decide escoger esta arquitectura pues:

- La solución está escrita dentro de un contexto que permite que su funcionalidad sea útil en la creación de distintas piezas de software.
- Está planteada de una forma general que permite su adaptación en distintas plataformas.
- El resto de componentes y piezas de software dentro de la plataforma a la que se quiere añadir, no se verán afectados por cambios en el diseño de la solución.
- La solución pueda ser desarrollada de manera independiente, agregando nuevas funcionalidades, sin afectar significativamente el resto de la plataforma.

La arquitectura de la solución está estructurada de la siguiente forma:

- **El nombre de los componentes**: el nombre del componente es la identificación de la funcionalidad y uso que tiene.

- **La interfaz de los componentes**: es el área de intercambio entre el interior y el exterior de un componente de software. La misma permite acceder a los datos y funcionalidades que estén especificadas en el interior del componente (acceder funcionalmente, no a su especificación). En este caso los archivos que pertenecen a esta parte son: sample.xul y olsmcp_Interface .js.
- **Cuerpo y código de implementación**: es la parte del componente que provee la forma (implementación) sobre la cual un fragmento del componente realiza sus servicios y funcionalidades. Los archivos que pertenecen a esta área son: OnlineSmartCardPlatform.dll y loadcard.js

[5] Destacado arquitecto de software y miembro de la Microsoft donde desempeña esta misma función.
[6] Ingeniero y profesor de la Universidad de California en el departamento de Ingeniería Eléctrica y Ciencias de la Computación

El flujo de procesos que sigue la solución aplicado a la arquitectura basada en componentes es el siguiente:

- El usuario interactúa con la interfaz de usuario, solicitando una operación en la página web, mediante el uso de su tarjeta inteligente.
- La página web llama a la funcionalidad correspondiente a la solicitud hecha por el usuario mediante la interfaz olsmcp_Interface la cual forma parte de la vista.
- La página java script olsmcp_interface envía la solicitud hecha al fichero sample.xul.
- En dependencia del tipo de evento que venga desde la interface olsmcp_interface, el fichero xul llama a la función correspondiente con esta implementada en loadcard.js, donde se crea una instancia de la librería OnlineSmartdCard.dll.
- La librería devuelve el resultado del pedido realizado, ascendentemente llevando a cabo la secuencia anterior.
- La aplicación web muestra los resultados al usuario a través de la página web y espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

2.5 Plan de Entrega

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. (Ver Anexo 1).

2.6 Plan de Iteraciones

Como parte del ciclo de vida de un proyecto usando la metodología XP se crea el plan de duración de cada una de las iteraciones que se han definido, que tiene como objetivo mostrar la duración y el orden en que serán implementadas las historias de

usuario dentro de cada iteración. Para la solución se han definido 8 historias de usuario divididas en 2 iteraciones, de acuerdo a los intereses del cliente, para una duración total del proyecto de 19 semanas. (Ver Anexo 3).

2.7 Estudio de Factibilidad

2.7.1 Estimación de Tiempo

Los desarrolladores estiman cuanto tiempo es necesario para implementar cada una de las Historias de usuarios. Si ese tiempo supera en alguna las tres semanas se debe desglosar en otras más pequeñas. Si es inferior a una semana, la historia de usuario ha descendido a un nivel de detalle excesivo y habrá que combinarla con otra. Esto se hace concluida la fase de Exploración. Como se ha dicho anteriormente, este valor es estimado y se irá acercando a la realidad con el transcurso de las iteraciones. En la presente solución se han identificado 7 Historias de Usuarios y quedan definidas las dos iteración con un total de 12 semanas de duración; de ser necesaria otra iteración se valoraría. (Ver Anexo 2).

2.7.2 Costos y Beneficios

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

El desarrollo del componente que interactúa con la tarjeta en el lenguaje C++ evitará que el usuario tenga que instalar middleware en la PC cliente, lo que hace más cómodo el uso de la plataforma.

A pesar de estar desarrollado para su integración con una solución específica, el add-on puede comercializarse por independiente para clientes que necesiten interactuar con tarjetas inteligentes utilizando Mozilla Firefox.

Conclusiones

En el presente capítulo se hace un detallado análisis de la metodología de software XP, generando artefactos como las Historias de Usuarios, Plan de entrega, Plan de iteraciones, además se definieron los requisitos no funcionales. También se generó el diagrama de clases perteneciente al modelo de dominio .Se diseñó la arquitectura de la aplicación. Luego de un análisis de los gastos incurridos y tomando en cuenta los beneficios que reportará se concluye que es factible su desarrollo.

CAPITULO 3: IMPLEMENTACIÓN Y PRUEBA

Introducción

En el presente capítulo se le da continuación a las fases de Iteraciones a primera liberación y Producción. Luego de definir las iteraciones, diseñar y probar las historias de usuario, corresponde ahora mostrar la evolución de la solución durante estas dos fases.

El objetivo fundamental de este capítulo es explicar el diseño de la solución; así como los principales componentes y las relaciones que existen entre ellos. A través de diferentes diagramas se ofrece una panorámica del funcionamiento del componente dentro de la plataforma y cómo ocurren los principales flujos de procesos.

3.2 Iteraciones a primera liberación

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de entrega está compuesto por iteraciones de no más de tres semanas de trabajo. Al principio de cada iteración se realizan las tareas necesarias de análisis, consultando con el cliente todos los datos que sean necesarios. Al final de la última iteración el sistema estará listo para entrar en la fase de producción. En esta fase es donde se da cumplimiento al plan de iteración, los elementos que se deben tomar en cuenta para la confección de este artefacto son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

3.3 Tareas de ingeniería

Antes de comenzar a codificar la solución es necesario saber qué codificar. Las historias de usuarios no ofrecen el nivel de detalle requerido para llevar a cabo esta acometida, es por eso que son divididas en tareas de la ingeniería. Una historia de usuario generalmente se divide en más de una tarea de la ingeniería y a partir de estas

tareas comienza el ciclo de la fase de Iteraciones a primera liberación. Las historias de usuario se agruparon en dos iteraciones. A continuación se muestran las tareas de ingeniería derivadas de cada historia de usuario.

Iteración	Historias de Usuarios	Tareas
1	Acceder a la librería OnLineSmartCardPlatform.dll	- Abrir conexión con la librería OnLineSmartCardPlatform.dll
	Abrir conexión con el lector de tarjeta	-Establecer una conexión con los lectores de tarjetas conectados a la computadora.
	Escoger lectores disponibles conectados a la computadora del cliente	-Listar lectores conectados.
2	Verificar estados (insertada/retirada) de una tarjeta en un lector.	-Notificar al usuario que la tarjeta ha sido insertada en el lector. -Notificar al usuario que la tarjeta ha sido desconectada del lector.
	Enviar y recibir comandos APDU de la tarjeta.	-Enviar comandos APDU a la tarjeta. -Recibir respuesta APDU de

		la tarjeta.
	Cerrar la conexión con la librería OnLineSmartCardPlatform.dll	-Cerrar la conexión con la librería OnLineSmartCardPlatform.dll

Tabla 12. Distribución de las tareas de ingeniería por iteraciones.

3.4 Diseño de la solución:

Una vez desglosadas las historias de usuario en tareas, algunos miembros del equipo de desarrollo comienzan a generar ideas de cómo ejecutarlas. Estas ideas se unifican en las sesiones de diseño previas a cada iteración. En estas sesiones, XP propone la puesta en práctica de ciertos principios, descritos a continuación, para garantizar la agilidad en el proceso de desarrollo.

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

- **Simplicidad**: Un diseño simple siempre se termina más rápido y es más fácil de entender que uno complejo.
- **Uso de tarjetas CRC**: (clase, responsabilidad, colaboración), estas tarjetas son manejadas por el equipo de desarrollo durante la codificación de la solución; generalmente cada tarjeta representa una clase diferente en la codificación y tienen como ventaja que todo el equipo contribuye a la elaboración del diseño de la solución.
- **No adicionar funcionalidades tempranamente**: Mantener el sistema lo más separado de las funcionalidades extras que no sean imprescindibles. Solo el 10% de las funcionalidades extras son utilizadas y hacen perder el 90% del tiempo. (Bondartchuk, 2009).

El diseño de la solución se realizó por iteraciones, agregando solo las funcionalidades necesarias en cada iteración. Además, se sustituyeron las tarjetas CRC por diagramas de clases de UML (Ver Anexo 5), que aunque algunos autores no lo aconsejan, fue posible dado que la complejidad del problema a resolver no implicaba la elaboración de diagramas complejos. De esta manera, se dio cumplimiento a los principios antes mencionados.

A continuación se hace una breve descripción de los principales componentes de la solución para entender el funcionamiento de la misma, dentro de la plataforma:

OnLineSmartCardPlatform.dll: biblioteca que expone las funcionalidades necesarias para conectarse y desconectarse con el lector de tarjetas, transmitir comandos `APDU`,

abrir conexión con el lector de tarjetas y mostrar cantidad de lectores conectados a la computadora.

olsmcp_Interface.js: clase `JavaScript` que actúa de mediador, estableciendo la comunicación entre la interfaz del usuario y el `add-on OLSMCP`, utilizando las funcionalidades contenidas en el mismo.

sample.xul: Este archivo `XUL` es el encargado de abrir la puerta para la comunicación entre la clase `olsmcp_Interface.js` y el resto del `add-on`, gestionando así el envío de solicitud y respuesta.

loadcard.js: clase `JavaScript` que exporta las funcionalidades desarrolladas en la librería `OnLineSmartCardPlatform.dll`.

Install.rdf: archivo que representa el "activador" del `add-on OLSMCP` y contiene la información del mismo, así como las versiones compatibles, las actualizaciones, la plataforma donde se encuentra disponible, los desarrolladores entre otros datos.

chrome.manifest: Declara el nombre del paquete donde se encuentra los archivos `XUL` y `JavaScript`, así como la ruta hacia el mismo. Por otra parte define también, cuál es el archivo `XUL` que se va a fusionar con el archivo del navegador Mozilla Firefox.

3.3.2 Descripción del flujo de proceso: Solicitud de un servicio.

El proceso fundamental que ejecuta la plataforma es atender la solicitud de un servicio hecha por un usuario. A continuación una breve descripción:

- El usuario hace click en un botón de la página web.
- La página web siempre hace una recarga parcial antes de enviar cualquier petición al servidor para actualizar los cambios que pudieran haber ocurrido.
- En cada recarga se obtiene una instancia de la clase `MiddlewareMediator` (controlador en el servidor). La instancia es única para una misma sesión web y

es creada en su primera llamada, a partir de entonces se obtiene la instancia ya existente.

- Como resultado al click del usuario la página intenta, a través de la instancia de `MiddlewareMediator` invocar la operación específica de `middleware` que da respuesta a la solicitud; para ello envía el nombre de la operación, los parámetros necesarios para la ejecución de la misma y el manejador de eventos que se encargará de procesar la respuesta una vez concluido el proceso.
- Inmediatamente después de la invocación se le ordena a la clase `SmartCardOnlinePlatform_addon` que comience el chequeo constante en búsqueda de nuevos comandos enviados por el `middleware` específico.
- En cada chequeo si la instancia de `MiddlewareMediator` se encuentra en estado "procesando" se envían al cliente los comandos (uno en cada petición o recarga parcial de la página) que llegan como resultado de la operación previamente ordenada a un `middleware` específico.
- Una vez que `SmartCardOnlinePlatform_addon` ha obtenido el comando `APDU` recibido desde `MiddlewareMediator`, utiliza una instancia de `SmartCardPlatform_addon` para enviar dicho comando a la tarjeta mediante el componente de extensión (`add-on`). La respuesta `APDU` de la tarjeta retorna por el mismo canal hacia el `SmartCardOnlinePlatform_addon`.
- En la próxima recarga parcial de la página la respuesta `APDU` será enviada a `MiddlewareMediator`, quién lo enviará al `middleware` específico que está aún procesando la operación que fue invocada. Si existe algún nuevo comando `APDU` se enviará hasta la tarjeta por el mismo procedimiento explicado anteriormente, repitiendo este proceso hasta que el `middleware` haya obtenido los datos necesarios para elaborar una respuesta entendible para el usuario.
- Una vez que el `middleware` le notifica a `MiddlewareMediator` la culminación de la operación, llamará al manejador que fue especificado en el momento de la invocación, para que procese la respuesta y la envíe al

`SmartCardOnlinePlatform_addon`. Este último gestiona la interfaz para mostrarle el resultado al usuario.

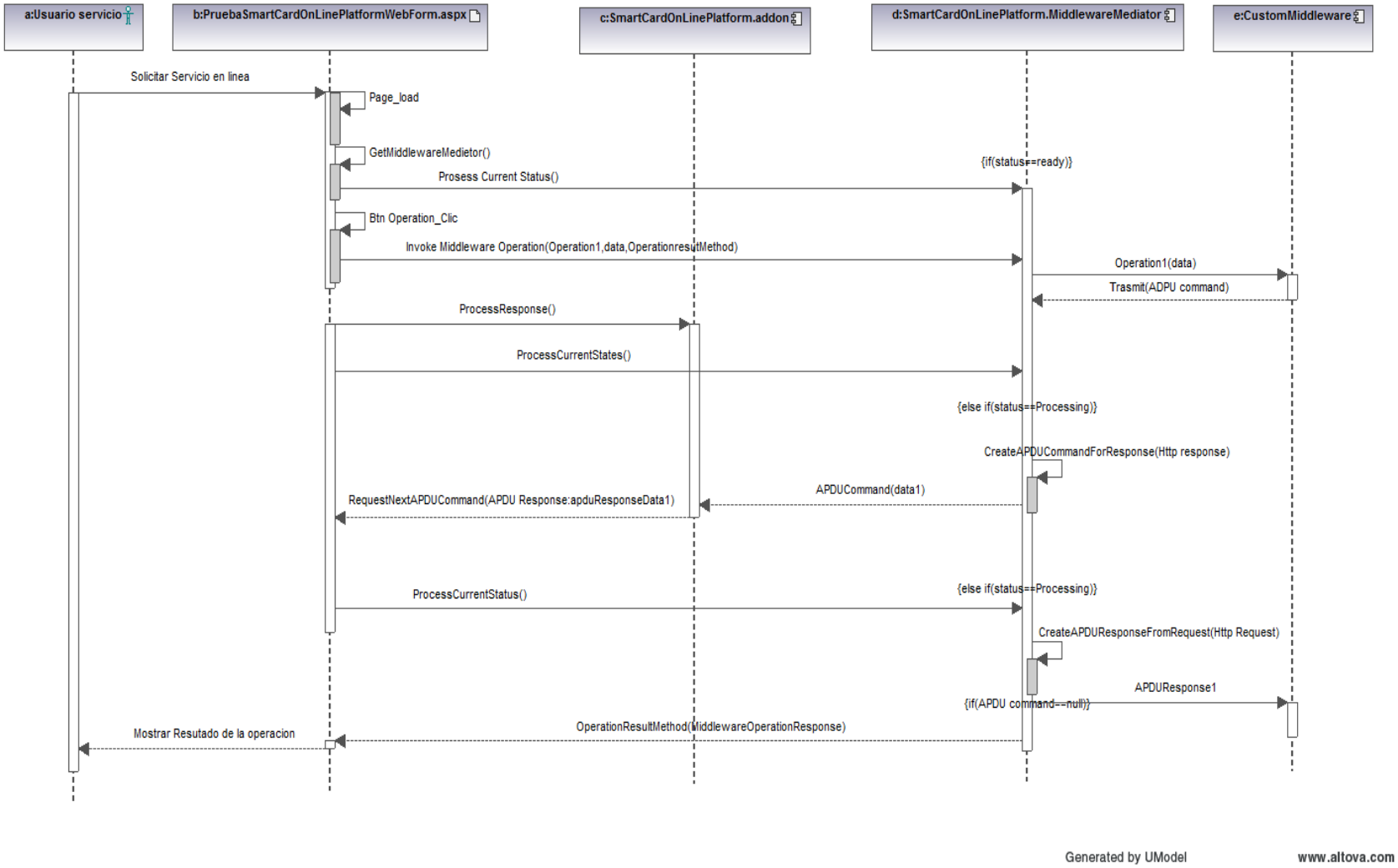


Ilustración 6. Diagrama de secuencia del proceso solicitud de un servicio

3.4.1 Pruebas Unitarias y Desarrollo Guiado por Pruebas

Las pruebas Unitarias se basan en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar.

Es por ello que se considera a la prueba de Caja Blanca como uno de los métodos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

Para la realización de las pruebas unitarias se utilizó el método de Caja Blanca y la herramienta fue `Visual Studio TeamSystem 2010`. Estas pruebas se utilizan para ejecutar otro código fuente llamando directamente a los métodos de una clase, pasándole los parámetros apropiados. Los métodos de pruebas unitarias residen en clases `Test`, que se almacenan en archivos de código fuente. Los resultados de las pruebas fueron satisfactorios. (Ver Anexo 10).

3.5 Fase de producción

Con la culminación de las iteraciones comienza la fase de producción, donde se llevan a cabo ajustes del rendimiento del sistema con vista a obtener una aplicación más rápida, robusta y lista para ser probada por el cliente. En esta fase se elaboran las pruebas de aceptación con el fin de lograr una retroalimentación del cliente donde generalmente se introducen nuevas funcionalidades o se rediseñan las existentes.

3.5.1 Pruebas de aceptación

Las pruebas del cliente también conocidas como pruebas de aceptación son definidas por este y su objetivo es probar que las funcionalidades del sistema corresponden con las historias de usuarios definidas por él. Estas son pruebas de caja negra que se

realizan a través de las interfaces del sistema. Los resultados de las pruebas fueron satisfactorios. (Ver Anexo 12)

Conclusiones

Luego de terminadas las fases de Iteraciones a Primera Liberación y Producción de la solución propuesta, se concluye que a partir del desglose de las historias de usuario en tareas de la ingeniería fue posible la realización del diseño y codificación.

El desarrollo guiado por pruebas aseguró la ejecución correcta de la solución en todo el período de desarrollo, aminorando el tiempo invertido en el ciclo compilación-ejecución.

Las pruebas de aceptación concluyeron de manera exitosa demostrando la satisfacción del cliente con la solución.

CONCLUSIONES

Con el desarrollo de esta investigación se han cumplido los objetivos principales definidos para la creación de un componente que posibilite la comunicación entre el navegador web Mozilla Firefox y los lectores de tarjetas inteligentes conectados a la computadora, impulsando de esta forma el desarrollo de los servicios en línea que la utilicen. Luego de concluida la primera versión del producto se arribó a las siguientes conclusiones:

- La investigación de las tecnologías relacionadas con el objeto de estudio permitió una correcta elección de la metodología y herramientas para el desarrollo de la solución.
- La definición de la arquitectura de componentes permitió garantizar una mayor flexibilidad en la ejecución de la propuesta de solución.
- El desarrollo del componente permite la comunicación entre una aplicación web que utilice tarjetas inteligentes y las tarjetas, con el `middleware` del lado del servidor, ofrece a los usuarios una mayor seguridad y confianza en este tipo de aplicaciones y facilita la actualización de las aplicaciones anexadas al servidor, así como la incorporación de nuevas funcionalidades, todo de forma transparente al usuario final, impulsando de esta forma el desarrollo de los servicios en línea que utilicen estas tecnologías.
- Las pruebas realizadas a la solución permitieron corregir los errores que se fueron dando al terminar cada iteración, logrando así que los resultados finales hayan sido satisfactorios.
- La utilización de una tecnología nativa para el desarrollo de un componente de extensión para el navegador a utilizar, facilitará el uso de la plataforma por parte del cliente evitándole el tener que instalar software adicional en su PC.

RECOMENDACIONES

Al concluir la investigación, se plantean las siguientes recomendaciones:

- Desarrollar un componente que permita la comunicación con los lectores de tarjetas conectados a la PC desde otros sistemas operativos.

- Confeccionar una documentación detallada de la plataforma para los desarrolladores a forma de Manual de usuario, pues dicho componente será anexado a esta.

- Mejorar la interfaz gráfica del `add-on` siendo más amigable para el usuario.

BIBLIOGRAFÍA CONSULTADA

Pereda Viñolo, Katerina y Fernández Santana, Vismar. 2010. *Plataforma para el desarrollo de servicios en línea utilizando tarjetas inteligentes.* Ciudad de La Habana, Cuba: Universidad de las Ciencias Informáticas, 2010.

Almeida Sotolongo, Dayron y Sáez Vilar, Joel. 2009. *Solución para el control de acceso a la información de las entidades externas, en la cédula de identificación electrónica de la República Bolivariana de Venezuela.* Ciudad de La Habana, Cuba: Universidad de las Ciencias Informáticas, 2009.

Don Wells. 2009. *Extreme Programming.* [En línea] 2009. http://www.extremeprogramming.org/.

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

GlobalPlatform, Inc. 2010. *GlobalPlatform.* [En línea] 2010. http://www.globalplatform.org/specifications.asp.

MDN/XPCOM. 2010. *Mozilla <developer center>.* [En línea] 2010. https://developer.mozilla.org/en/XPCOM.

PC/SC Workgroup. 2010. *PC/SC Workgroup.* [En línea] 2010. http://www.pcscworkgroup.com/.

Pérez Costa, Michel Rafael y Fernández Leyva, Luis. 2008. *Sistema de Administración de Tarjetas Inteligentes y Aplicaciones para la Cédula de Identidad Electrónica de la República Bolivariana de Venezuela.* Ciudad de La Habana, Cuba: Universidad de las Ciencias Informáticas, 2008.

Wells, J. Donovan. 1999. Extreme Programming: A gentle introduction. [En línea] 1999. http://www.extremeprogramming.org/.

Yurdik Cervantes Mendoza. *Biometría en las tarjetas inteligentes.* Centro Universidad de las Ciencias Informáticas.

Yahya Haghiri, Thomas Tarantino. 2002. *Smart Card Manufacturing. A practical guide.*

Prof. Dr. Andreas Steffen. 2009. *SmartCards. Internet Security.* Institute fot Internet Technologies and Applications (ITA).

Graig Larman. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.*

GemXpresso Pro R3.x. 2004. *Reference Manual.* Gemplus S.A. France.

Gemalto. 2008 *CryptoManager Applet 2.0. Reference Manual.* Gemalto N.V.

Roberto Quiñones Bondartchuk. 2009. *Firma Digital de Documentos utilizando Smart Cards.* Universidad de las Ciencias Informáticas.

José Joskowickz. 2008. *Reglas y Prácticas en eXtrme Programming.* Doctorado de Ingeniería Telemática de la Universidad de Vigo, España.

ISO/IEC 7816-4. 2005. *International Standard. Identification cards-Integrated circuit cards.* Segunda Edición.

Burton S. Kaliski Jr. 1991. *An Overview of the PKCS Standards.*

Sun Microsystems. 2003. *Application Programming Interface. Java Card Platform.*

BIBLIOGRAFÍA REFERENCIADA

Almeida Sotolongo, Dayron y Sáez Vilar, Joel. 2009.*Solución para el control de acceso a la información de las entidades externas, en la cédula de identificación electrónica de la República Bolivariana de Venezuela.* Ciudad de La Habana, Cuba: Universidad de las Ciencias Informáticas : s.n., 2009. 4.

—. 2009.*Solución para el control de acceso a la información de las entidades externas, en la cédula de identificación electrónica de la República Bolivariana de Venezuela.* Ciudad de La Habana, Cuba: Universidad de las Ciencias Informáticas : s.n., 2009.

Bondartchuk, ing.Roberto Quiñones. 2009.*Firma Digital de Documentos utilizando Smart Cards.* Ciudad de la Habana : UCI, 2009.

2011. Free Diccionary. [Online] 2011. www.thefreedictionary.com.

2011. Free Diccionary. [Online] 2011. www.thefreedictionary.com. 3.

2010. Global Platforrm, Inc. [Online] 2010. www.globalplatform.org/specifications.asp. 6.

2011. Glosarium. [Online] 2011. www.glosarium.com.

2011. Glosarium. [Online] 2011. www.glosarium.com. 4.

Joskowicz, Ing.José.*Reglas y Prácticas en eXtreme Programing.*

Katerina Preda Viñolo, Vismar Fernández Santana. 2010.*Plataforma para el desarrollo de servicios en línea.* 2010.

2011. Mozilla. [Online] 2011. http://www.mozilla.org.

2011. Mozilla. [Online] 2011. www.mozilla.org.

NUnit. Manual de instalacion al cliente.

Organization, ISO. 2005.*INTERNATIONAL STANDARD ISO/IEC 7816-4.* 2005.

—. 2005.*INTERNATIONAL STANDARD ISO/IEC 7816-4.* 2005. 5.

Reglas y Prácticas en extreme Programing. **Joskowicz, Ing. José. 2010.** 2010.

—.**Joskowicz, Ing. José. 2010.**2010.

Wells, Don. 2009. Extreme Programing. [Online] 2009. www.extremeprogramming.org.

—. 2009. Extreme Programing. [Online] 2009. www.extremeprogramming.org.

[En línea] 2010. www.globalplatform.org/specifications.asp. [Online] [En línea] 2010.

GLOSARIO DE TÉRMINOS

Navegador web

Un navegador es un programa que permite visualizar la información que contiene una página web, interpreta el código de la página, HTML generalmente y lo presenta en el monitor de la computadora permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos.

Los documentos pueden estar ubicados en la computadora del usuario, pero también pueden estar en cualquier dispositivo que esté conectado a la computadora del mismo a través de Internet. Tales documentos, comúnmente denominados páginas web, poseen hipervínculos que enlazan una porción de texto o una imagen a otro, normalmente relacionado. La comunicación entre el servidor web y el navegador se realiza generalmente mediante el protocolo HTTP.

Entre los navegadores más utilizados a nivel mundial se encuentran Internet Explorer desarrollado por Windows, siendo el más utilizado desde 1999, lo que ha ido disminuyendo paulatinamente por la renovada competencia de navegadores como *Mozilla Firefox*. Este último fue desarrollado por la Fundación Mozilla y un gran número de voluntarios externos, siendo un navegador multiplataforma y de código fuente abierto.

Add-ons

“Los add-ons son pequeños programas ejecutables que sólo funcionan anexados a otro y que sirven para aportar o complementar alguna función. Estos complementos permiten que los desarrolladores externos colaboren con la aplicación principal extendiendo sus funciones, reduciendo el tamaño y separando el código fuente de la aplicación a causa de la incompatibilidad de las licencias de software.

Entre las aplicaciones más comunes que suelen incluirlos están los navegadores web ya que amplían las funciones de las páginas web para ver contenidos interactivos,

videos y cosas similares. Son especialmente populares para Mozilla Firefox. En general, cualquier aplicación puede añadir soporte para los add-ons." (2011)

Middleware

Un `middleware` es un software que funciona como una capa de abstracción, situada entre la de aplicación y las inferiores haciendo posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. En general son usados 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 e integradores de aplicaciones, para que estas puedan ser fácilmente integradas, reutilizadas, adaptadas y hechas para interoperar.
- Suministro de 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.
- Actualmente son muy comunes para la interacción con las tarjetas inteligentes en los computadores personales ya que hacen función de intermediarios entre diversas aplicaciones y los lectores de tarjetas.

Applets

Los `applets`, en el contexto de las tarjetas inteligentes, son aplicaciones implementadas utilizando la tecnología `JavaCard` y ejecutadas dentro de estas. Un `applet` gestiona la información de las entidades externas contenida en la Cédula de Identificación Electrónica (en lo adelante CIE) se ejecuta en el contexto del `JavaCard` Runtime–Environment (en lo adelante JCRE), es el encargado de ejecutar las

operaciones que maneja, mediante los comandos APDU que le son enviados, retornando los resultados mediante APDU de respuestas. (Almeida Sotolongo, 2009)

DLL (Dynamic-Link Library)

Una dll es una librería de vínculos dinámicos, esto no es más que un archivo que contiene funciones que se pueden llamar desde aplicaciones u otras dll. Los desarrolladores utilizan las librerías para poder reciclar el código y aislar las diferentes tareas. Las dll no pueden ejecutarse directamente, es necesario llamarlas desde un código externo. En Windows estas bibliotecas están muy extendidas y son compartidas por múltiples aplicaciones.

Cliente

Un cliente es la persona o institución que adquiere un determinado software o sistema para su beneficio y lo pone a disposición de otros.

Usuario

Un usuario es la persona que utiliza o trabaja con algún software. En sentido general es un conjunto de permisos y de recursos (o dispositivos) a los cuales se tiene acceso. Es decir, un usuario puede ser tanto una persona, una máquina o un programa.

ANEXOS

Anexo 1. Plan de entregas.

Entregable	Fin Iteración 1	Fin Iteración 2
Addon Smart Card	Febrero 2011	Mayo 2011

Tabla 13. Plan de entregas.

Anexo 2. Estimación de tiempo de las historias de usuarios.

Historias de usuarios	Estimación
Acceder a la librería `OnLineSmartCardPlatform.dll`	1
Abrir conexión con el lector de tarjeta	3
Escoger lectores disponibles conectados a la computadora del cliente	1
Verificar estados (insertada/retirada) de una tarjeta en un lector.	3
Enviar y recibir comandos `APDU` de la tarjeta.	3
Cerrar la librería `OnLineSmartCardPlatform.dll`	1

Tabla 134. Estimación de tiempo de las historias de usuario.

Anexo 3. Plan de Iteraciones.

Iteración	No. HU	Historia de Usuario	Duración Estimada(semanas)
Iteración 1	HU_1	Acceder a la librería `OnLineSmartCardPlatform.dll`.	1
	HU_5	Abrir conexión con el lector de tarjeta.	3
	HU_3	Escoger lectores disponibles conectados a la computadora del cliente.	1
Iteración 2	HU_6	Verificar estados (insertada/retirada) de una tarjeta en un lector.	3
	HU_4	Enviar y recibir comandos `APDU` de la tarjeta.	3
	HU_2	Cerrar la librería `OnLineSmartCardPlatform.dll`.	1

Tabla 1514. Plan de iteraciones.

Anexo 4. Especificación de las tareas de ingeniería.

Tarea de ingeniería	
Número de tarea: HU_1 T1	**Historia de usuario:** Acceder a la librería `OnLineSmartCardPlatform.dll`
Nombre: Abrir conexión con la librería `OnLineSmartCardPlatform.dll`	
Tipo: Desarrollo	**Puntos estimados:**1
Fecha inicio:	**Fecha fin:**
Responsable: Yadira García Huelga	
Descripción: Para acceder a la librería que contiene las funcionalidades principales se debe llamar a la funcionalidad `ctypes.open ()`	

Tabla 156. HU1_T1 Abrir conexión con la librería OnLineSmartCardPlatform.dll.

Tarea de ingeniería	
Número de tarea: HU_5 T1	**Historia de usuario:** Abrir conexión con el lector de tarjeta
Nombre: Establecer una conexión con los lectores de tarjetas conectados en a la PC.	
Tipo: Desarrollo	**Puntos estimados:**3
Fecha inicio:	**Fecha fin:**
Responsable: Yadira García Huelga	
Descripción: se establece una conexión con la tarjeta que permita iniciar la comunicación de la plataforma con la misma. Una vez establecida la comunicación con el lector se le muestra al usuario mediante un mensaje si la conexión con la tarjeta se ha establecido	

Tabla 167. HU5_T1 Abrir conexión con la librería OnLineSmartCardPlatform.dll.

Tarea de ingeniería	
Número de tarea: HU_3 T1	**Historia de usuario:** Escoger lectores disponibles conectados a la computadora del cliente

Nombre: Listar lectores conectados.	
Tipo: Desarrollo	**Puntos estimados:**1
Fecha inicio:	**Fecha fin:**
Responsable: Yadira García Huelga	
Descripción: Accediendo a la `OnLineSmartCardPlatform.dll` se obtiene la lista de lectores conectados al ordenador del usuario, mostrándole una lista desplegable con los nombres de los lectores disponibles para que escoja uno.	

Tabla 18. HU3_T1 Listar lectores conectados.

	Tarea de ingeniería
Número de tarea: HU_6 T1	**Historia de usuario:** Verificar estados (insertada/retirada) de una tarjeta en un lector.
Nombre: Notificar al usuario que la tarjeta ha sido insertada en el lector	
Tipo: Desarrollo	**Puntos estimados:**3
Fecha inicio:	**Fecha fin:**
Responsable: Ángel Manuel Romero	
Descripción: el sistema hace un chequeo periódico del estado de la tarjeta dentro del lector y cuando se registra un cambio se le notificará al usuario a través de un mensaje que la tarjeta está conectada. .	

Tabla 19. HU6_T1 Notificar al usuario que la tarjeta ha sido insertada en el lector.

	Tarea de ingeniería
Número de tarea: HU_6 T2	**Historia de usuario:** Verificar estados (insertada/retirada) de una tarjeta en un lector.
Nombre: Notificar al usuario que la tarjeta ha sido desconectada en el lector	
Tipo: Desarrollo	**Puntos estimados:**3
Fecha inicio:	**Fecha fin:**

Responsable: Ángel Manuel Romero
Descripción: el sistema hace un chequeo periódico del estado de la tarjeta dentro del lector y cuando se registra un cambio se le notificará al usuario a través de un mensaje que la tarjeta esta desconectada.

Tabla 20. HU6_T1 Notificar al usuario que la tarjeta ha sido desconectada en el lector.

	Tarea de ingeniería
Número de tarea: HU_4 T1	**Historia de usuario:** Enviar y recibir comandos APDU de la tarjeta.
Nombre: Enviar comandos APDU a la tarjeta.	
Tipo: Desarrollo	**Puntos estimados:**3
Fecha inicio:	**Fecha fin:**
Responsable: Ángel Manuel Romero	
Descripción: el componente cliente recibe el comando `APDU` desde el servidor y se lo envía a la tarjeta a través del elemento encargado de la comunicación con la misma.	

Tabla 21. HU4_T1 Enviar comandos APDU a la tarjeta.

	Tarea de ingeniería
Número de tarea: HU_4 T2	**Historia de usuario:** Enviar y recibir comandos `APDU` de la tarjeta.
Nombre: Recibir respuesta `APDU` de la tarjeta.	
Tipo: Desarrollo	**Puntos estimados:**3
Fecha inicio:	**Fecha fin:**
Responsable: Ángel Manuel Romero	
Descripción: luego que se le envía el primer comando `APDU` a la tarjeta, el sistema se quedará esperando una respuesta. La tarjeta procesa el comando `APDU` enviado desde el componente y devuelve una respuesta, comenzando el ciclo nuevamente.	

Tabla 22. HU4_T2 Recibir respuesta APDU de la tarjeta.

Tarea de ingeniería	
Número de tarea: HU_2 T1	**Historia de usuario:** Cerrar la conexión con la librería `OnLineSmartCardPlatform.dll`
Nombre: Cerrar la conexión con la libreríaOnLineSmartCardPlatform.dll	
Tipo: Desarrollo	**Puntos estimados:**1
Fecha inicio:	**Fecha fin:**
Responsable: Ángel Manuel Romero	
Descripción: una vez que se concluya el trabajo con la librería OnLineSmartCardPlatform.dll se debe cerrar la conexión con la misma llamando a la función *ctypes.close().*	

Tabla 23. HU4_T2 Cerrar la conexión con la librería OnLineSmartCardPlatform.dll.

Anexo 5. Diagrama de clases de la capa cliente.

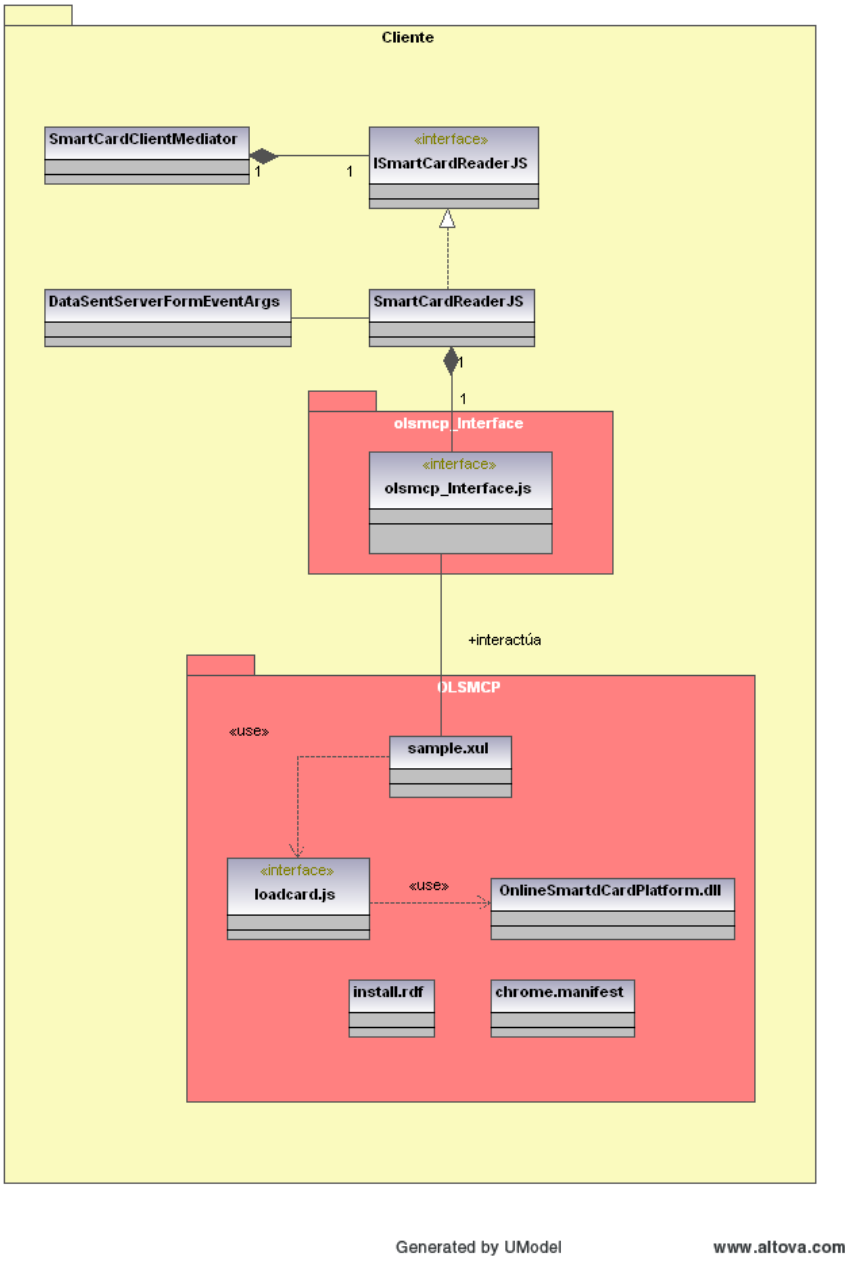


Ilustración 3. Diagrama de clases de la capa cliente

Anexo 6. Pruebas Unitarias

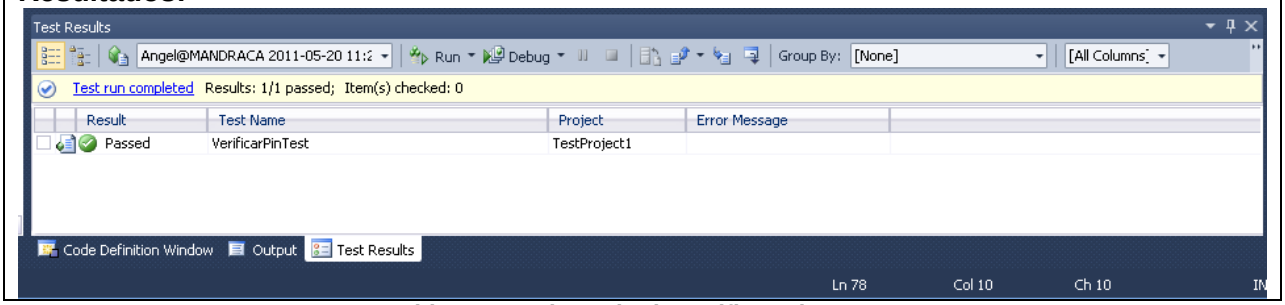
Prueba de Unidad		
Nombre de prueba: Verificar Ping Test		
Estado: Satisfactoria	Tipo: Caja Blanca	**Última ejecución:**
Ejecutado por: Yadira García Huelga		**Verificado por:**
Descripción: Para poder ejecutar la prueba previamente se deben haber asignado los valores del pin (*0000*) si los datos son correctos la verificación se realiza de forma satisfactoria, de lo contrario, se lanza un mensaje mostrando el error que se originó.		
Criterio de Aceptación : Verificar Pin		
Resultados: 		

Tabla 24. Prueba Unitaria Verificar Ping Test.

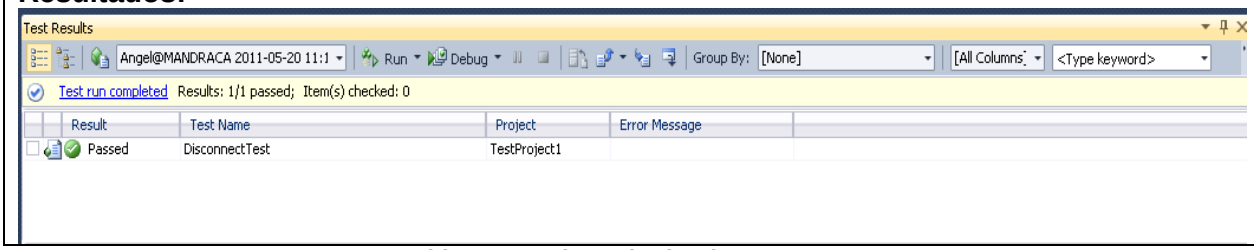
Prueba de Unidad		
Nombre de prueba: Disconnect Test		
Estado: Satisfactoria	Tipo: Caja Blanca	**Última ejecución:**
Ejecutado por: Yadira García Huelga		**Verificado por:**
Descripción: Si existe un lector conectado a la computadora, se realiza la prueba satisfactoriamente, de lo contrario se lanza un mensaje mostrando el error que se originó.		
Criterio de Aceptación :desconectar lector		
Resultados: 		

Tabla 25. Prueba Unitaria Disconnect Test.

Anexo 7. Diagrama de clases del modelo de dominio.

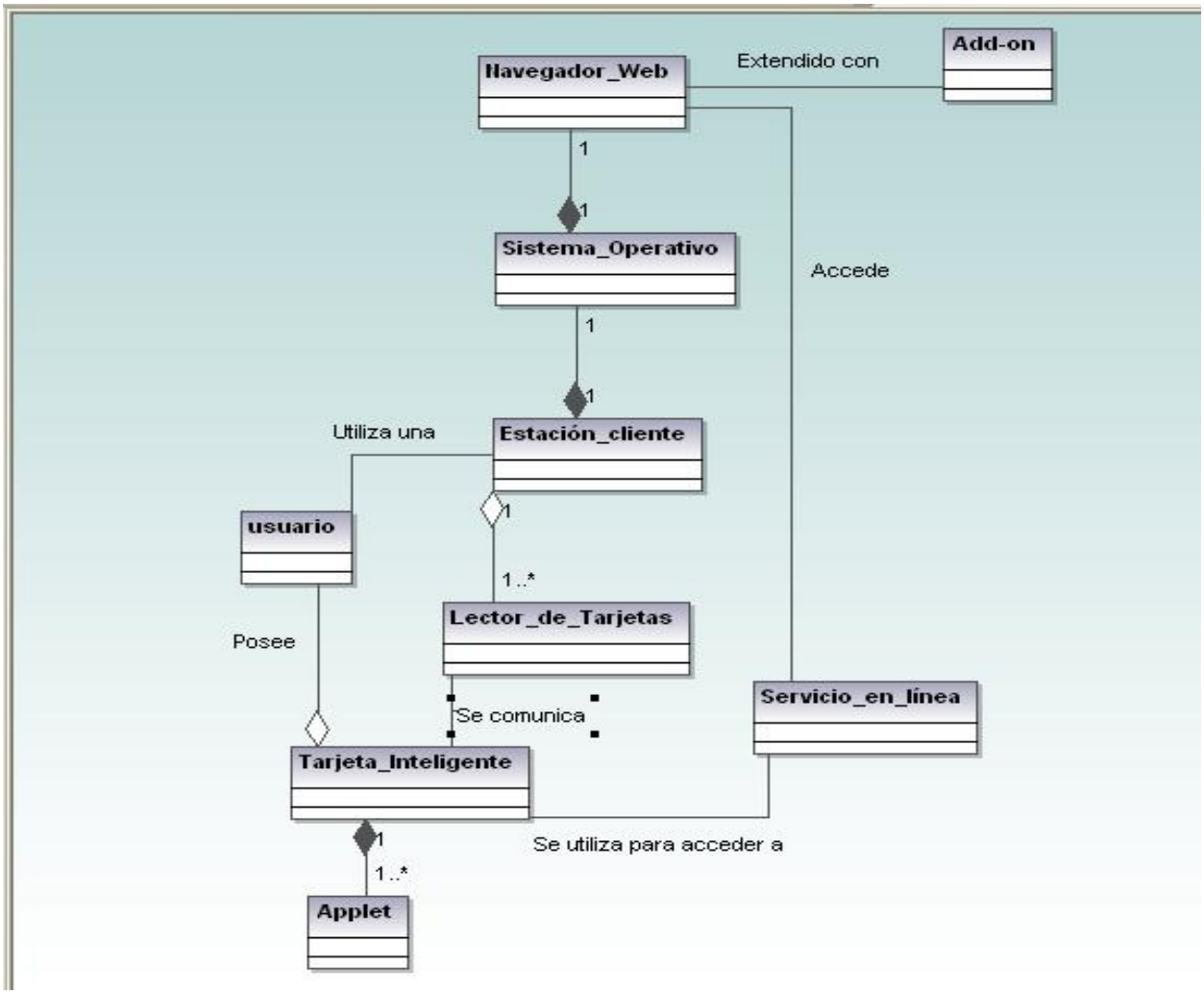


Ilustración 4. Diagrama de clases del modelo de domino

Anexo 8. Casos de pruebas de aceptación.

Caso de prueba de aceptación	
Código de caso de prueba: HU1_CP1	**Nombre de la historia de usuario:** Cerrar la conexión con la librería OnLineSmartCardPlatform.dll
Responsable de la prueba: Yadira García Huelga	
Descripción de la prueba: Prueba de funcionalidad para cerrar la conexión con la librería OnLineSmartCardPlatform.dll	
Condiciones de ejecución: Debe existir la librería OnLineSmartCardPlatform.dll	
Entrada/Pasos de ejecución: • El desarrollador importa la librería OnLineSmartCardPlatform.dll • Se establece la conexión con la librería OnLineSmartCardPlatform.dll	
Resultado esperado: Se establece satisfactoriamente la conexión con la librería OnLineSmartCardPlatform.dll	
Evaluación de la prueba: Prueba Satisfactoria.	

Tabla 26. HU_1CP1 Acceder a la librería OnLineSmartCardPlatform.dll.

Caso de prueba de aceptación	
Código de caso de prueba: HU2_CP2	**Nombre de la historia de usuario:** Acceder a la librería OnLineSmartCardPlatform.dll
Responsable de la prueba: Yadira García Huelga	
Descripción de la prueba: Prueba de funcionalidad para acceder a la libreríaOnLineSmartCardPlatform.dll	
Condiciones de ejecución: Debe existir la librería OnLineSmartCardPlatform.dll	
Entrada/Pasos de ejecución: • El desarrollador cierra la conexión con la librería OnLineSmartCardPlatform.dll mediante la función *ctype.close()*.	

Tabla 27. HU_2CP2 Cerrar la librería OnLineSmartCardPlatform.dll.

Caso de prueba de aceptación	
Código de caso de prueba: HU3_CP3	**Nombre de la historia de usuario:** Escoger lectores disponibles conectados a la computadora del cliente
Responsable de la prueba: Yadira García Huelga	
Descripción de la prueba: Prueba de funcionalidad para listar los lectores conectados.	
Condiciones de ejecución: Debe existir algún lector conectado.	
Entrada/Pasos de ejecución: • El usuario se registra en la aplicación. • Se descarga e instala el componente en el navegador web Mozilla. • Aparece en la página una lista desplegable con los nombres de los lectores conectados.	
Resultado esperado: Mostrar satisfactoriamente el listado con los nombres de los lectores conectados.	
Evaluación de la prueba: Prueba Satisfactoria.	

Tabla 28. HU_3CP3 Escoger lectores disponibles conectados a la computadora del cliente.

Caso de prueba de aceptación	
Código de caso de prueba: HU4_CP4	**Nombre de la historia de usuario:** Gestionar transmisión de comandos APDU entre el servidor y el cliente web.
Responsable de la prueba: Ángel Manuel Romero	
Descripción de la prueba: Prueba de funcionalidad para las transmisiones de comandos APDU entre el servidor y el cliente web.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución:	

• El usuario hace una solicitud de servicio. • El componente cliente la envía al controlador en el servidor. • Una vez allí se envía al middleware que se está invocando, quien comienza la secuencia de los APDU. • El controlador gestiona el envío de este comando APDU al cliente web haciendo de mediador iniciando la comunicación.
Resultado esperado: Se establece una comunicación entre el cliente web y el servidor
Evaluación de la prueba: Prueba Satisfactoria.

Tabla 29. HU_4CP4 Gestionar transmisión de comandos APDU entre el servidor y el cliente web.

Caso de prueba de aceptación	
Código de caso de prueba: HU5_CP5	**Nombre de la historia de usuario:** Enviar y recibir comandos APDU de la tarjeta.
Responsable de la prueba: Ángel Manuel Romero	
Descripción de la prueba: Prueba de funcionalidad para el intercambio de APDU entre el lector y la tarjeta.	
Condiciones de ejecución: Debe haber una tarjeta insertada en el lector.	
Entrada/Pasos de ejecución: • El usuario hace una solicitud de servicio. • Se envía la petición al middleware en el servidor. • El middleware inicia la comunicación enviando el primer comando APDU al cliente web. • La capa JavaScript lo envía al componente que se encargará de enviarlo a la tarjeta. • La tarjeta procesa el comando APDU y devuelve una respuesta que será enviada al middleware en el servidor nuevamente.	
Resultado esperado: Se muestra al usuario el resultado del servicio solicitado.	

Evaluación de la prueba: Prueba Satisfactoria.

Tabla 30. HU_5CP5 Enviar y recibir comandos APDU de la tarjeta.

Caso de prueba de aceptación	
Código de caso de prueba: HU5_CP5	**Nombre de la historia de usuario:** Abrir conexión con el lector de tarjeta
Responsable de la prueba: Ángel Manuel Romero	
Descripción de la prueba: Prueba de funcionalidad para establecer la conexión entre el lector y la tarjeta.	
Condiciones de ejecución: Debe haber una tarjeta insertada en el lector.	
Entrada/Pasos de ejecución: • El usuario se registra en la aplicación. • Selecciona un lector para establecer la conexión. • El lector establece conexión con la tarjeta inteligente.	
Resultado esperado: Se muestra una notificación al usuario que se ha establecido la conexión.	
Evaluación de la prueba: Prueba Satisfactoria.	

Tabla 31. HU_6CP6 Abrir conexión con el lector de tarjeta.

Anexo 9. Desarrollo de add-ons

Ejemplo de un archivo install.rdf con la descripción de su sintaxis.

```
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#&quot;
xmlns:em="http://www.mozilla.org/2004/em-rdf#&quot;>
   <Description about="urn:mozilla:install-manifest">
   <em:id>miextension@undominio</em:id>
   <em:name>Nombre de la Extensión</em:name>
   <em:version>1.0</em:version>
   <em:description>Descripción de la extensión</em:description>
   <em:creator>Nombre del autor/res</em:creator>
   <em:targetApplication>
     <Description>
       <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
       <em:minVersion>Versión Mínima soportada</em:minVersion>
       <em:maxVersion>Versión Máxima soportada</em:maxVersion>
     </Description>
   </em:targetApplication>
 </Description>
</RDF>
```

***id*:** El identificador, debe ser único. Como se ha mencionado, el mismo puede ser un GUID o un identificador con formato de correo electrónico. En el ejemplo presentado tiene formato de correo electrónico. Al ser único se evita problemas de conflictos entre extensiones.

***name*:** Es el nombre completo del add-on.

***version*:** Identifica la versión de la extensión, ejemplo: minVersion=3.0 y maxVersion=3.0.* que quiere decir de la versión 3.0 en adelante.

***description*:** Permite definir una descripción para la extensión.

***creator*:** El nombre del autor/es de la extensión.

***GUID de la aplicación*:** Es el identificador de la aplicación sobre la cual la extensión va a correr:

- Firefox `{ec8030f7-c20a-464f-9b0e-13a3a9e97384}`

- Mozilla Suite {86c18b42-e466-45a9-ae7a-9b95ba6f5640}
- SeaMonkey {92650c4d-4b8e-4d2a-b7eb-24ecf4f6b63a}
- Sunbird {718e30fb-e89b-41dd-9da7-e25a45638b28}
- Netscape {3db10fab-e461-4c80-8b97-957ad5f8ea47}
- Flock {a463f10c-3994-11da-9945-000d60ca027b}
- Instantbird {33cb9019-c295-46dd-be21-8c4936574bee}

***minVersion*:** Indica la compatibilidad de la extensión con respecto a una versión mínima del navegador Mozilla Firefox.

***maxVersion*:** Indica la compatibilidad con una versión máxima del navegador Mozilla Firefox.

Ejemplo de un archivo chrome.manifest con la descripción de su sintaxis.

Para poder declarar el material asociado a la extensión se debe seguir la siguiente sintaxis: *chrome://miextension/content/unArchivo.XUL*. Con esto estamos diciendo que se deberá cargar la interfaz definida en el archivo *unArchivo.XUL* en la sección de **content** del add-on. Si el material a definir es una imagen, lo podemos declarar de la siguiente manera: *chrome://miextension/skin/miImagen.jpg*, donde estamos diciendo que se cargue la imagen *miImagen.jpg* en la sección skin del add-on.

```
# Registro del tipo de material que contiene la extensión.
# Tipo de material - Nombre del paquete chrome (nombre de la extension) - Ubicación
de los archivos.
 content miextension chrome/content/
 locale miextension chrome/locale/
 skin miextension chrome/skin/
# Fusión del navegador browser.XUL del navegador Mozilla Firefox con la
interfaz de la extensión.
overlay chrome://browser/content/browser.xul
chrome://miextension/content/menuPrincipal.xul
```