Universidad de las Ciencias Informáticas

Facultad 1

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

Título:

Aplicación web para la lectura de pasaportes electrónicos.

Autor(es):

Reisel de la Rosa Ge.

Daniel Fuentes Castro.

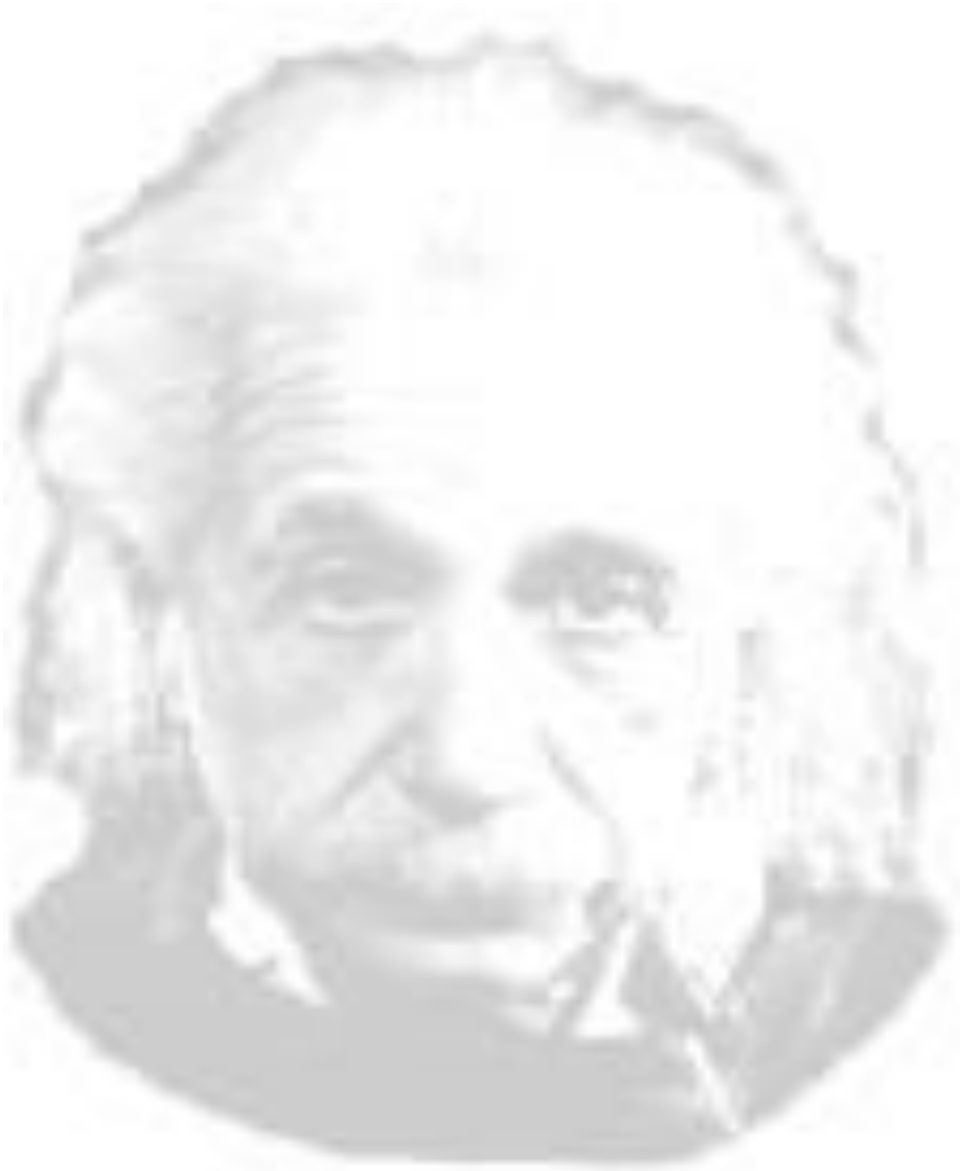
Tutor(es):

Ing. Ander Sánchez Jardines.

Ing. Alexander López Pupo.

La Habana, Cuba.

Junio 2013.



"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad."

Albert Einstein:(14 de marzo de 1879 - 18 de abril de 1955), Físico y matemático alemán, nacionalizado posteriormente como estadounidense. Descubridor del movimiento browniano, el efecto fotoeléctrico y la teoría de la relatividad. Fue uno de los genios más polémicos de todos los tiempos, considerado el científico más conocido e importante del siglo XX. Extravagante y distraído, pero también hombre simple, se interesó profundamente por los asuntos del mundo y tuvo fe en la grandeza del ser humano.

Declaración de autoría.

Declaramos ser autores del presente trabajo de diploma y autorizamos al Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

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

Reisel de la Rosa Ge.

_____

Firma del Autor.

Daniel Fuentes Castro.

_____

Firma del Autor.

Ing. Ander Sánchez Jardines.

_____

Firma del Tutor.

Ing. Alexander López Pupo.

_____

Firma del Tutor.

Datos de Contacto.

Tutor: Ing. Ander Sánchez Jardines.

- Ingeniero en Ciencias Informáticas.
- Profesor instructor de dos años de experiencia.
- Actualmente se encuentra vinculado al departamento de Tarjetas Inteligentes de la Universidad de las Ciencias informáticas.

Correo electrónico: ajardines@uci.cu

Tutor: Ing. Alexander López Pupo.

- Ingeniero en Ciencias Informáticas.
- Profesor instructor de cinco años de experiencia.
- Actualmente se encuentra vinculado al departamento Tarjetas Inteligentes del Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas.

Correo electrónico: alopezp@uci.cu

Dedicatoria.

A mi mamá y a mi papá, los mejores padres del mundo, que en todo momento me han apoyado y han estado pendientes de mis caprichos siempre dando lo mejor de sí, a quienes le debo la vida y estar graduándome como ingeniero.

A mi hermanita preciosa y mi sobrinito, a quienes quiero con la vida.

A mi familia en general por estar siempre pendientes y preocupados respecto a mí, en especial a mi tía Isabel, Sandro, Zoe, Kenia, Tomás, Mima, Alejandro, Gumercindo, Nani, Magdiel, Mario, Chicho, etcétera.

Una dedicación especial a mi prima del alma Dilaida, así como a Máximo, Adialid y Annalia por dedicarse a mí por completo y gracias al apoyo de ellos hoy soy quien soy.

Agradecimientos.

A la Revolución por permitirme formar parte de este gran proyecto, forjado a raíz de la batalla de ideas.

Agradecer infinitamente en primer lugar a mis padres por apoyarme y guiarme siempre, por su amor, dedicación y comprensión.

A mi tía Isabel y mi prima Dilaida por aconsejarme todo este tiempo y en todo momento, ustedes también son parte de este triunfo.

A los tutores Ander y Alexander por atenderme y dedicarme su tiempo a comprender y desarrollar este trabajo.

A todos los profesores que de una forma u otra han contribuido con mi formación profesional.

A todos mis compañeros de brigada, amigos y personas importantes durante estos 5 años, que resultaron ser personas agradables y buenos compañeros.

A un grupo de personas que sin dudas se han ganado un lugar en mi corazón, ellas son mis queridas amigas: Anisleidy, Isardis, Yadis, Glenys, Yenni , Geydita, Adyaris, Rosalina, Rocío, Lianet, gracias por demostrarme en más de una ocasión que podía contar con ustedes para lo que fuera, además de aguantar muchos de mis caprichos, las quiero.

Un agradecimiento especial a tres personitas especiales que más que mis amigas han sido mis hermanitas, ellas son mi queridísima bobis (Alicia),Yanara y Yaumara, ustedes han llegado llegaron de una forma especial y se quedaran de manera inolvidable en mi mente y corazón.

A todos muchísimas gracias…

Resumen.

La utilización de los pasaportes electrónicos ha ido en aumento en los últimos tiempos, definiendo diversos estándares a seguir por los países que utilizan este tipo de documentos, permitiendo así una interoperabilidad entre todos los sistemas que siguen sus especificaciones.

La mayoría de las aplicaciones web que ofrecen servicios usando pasaportes electrónicos, lo hacen a través de middleware que se deben instalar en su computadora personal (PC). La falta de una aplicación que permita la comunicación en línea con los pasaportes, siendo funcional para diferentes tipos de navegadores web y sistemas operativos, ha hecho que no se aprovechen al máximo todas las ventajas de estos a través de internet.

El Centro de Identificación y Seguridad Digital (CISED), con la experiencia adquirida en el área de los pasaportes electrónicos, ha identificado la necesidad del desarrollo de una aplicación con el middleware en el lado del servidor, evitando la instalación de software en el lado del cliente, brindando mayor seguridad en la comunicación con los pasaportes en la web.

En el presente trabajo se presenta un estudio sobre los actuales sistemas que utilizan pasaportes electrónicos, así como la descripción de los elementos que conforman el software. Como resultado, se desarrolla una aplicación para interactuar con los pasaportes que cumplan con los estándares definidos por la Organización Internacional de la Aviación Civil (OACI). El documento recoge los principales resultados, arrojados en la arquitectura y el diseño del sistema. Se describen las herramientas, tecnologías utilizadas y los artefactos generados en el proceso de desarrollo.

Palabras claves: pasaportes electrónicos, seguridad, servicios en línea, middleware, navegadores web.

Índice de contenidos.

Índice de tablas.

Índice de figuras.

Introducción.

El pasaporte es un documento oficial de viaje que contiene la información referente a las personas que viajan de un Estado a otro, aunque han existido durante mucho tiempo su utilización alcanzó mayor auge en los años posteriores a la primera guerra mundial, ya muchos gobiernos lo emplearon como medida de seguridad y para el control de la emigración e inmigración de ciudadanos en sus fronteras.

Con el objetivo de estandarizar dichos pasaportes a nivel internacional la OACI, como organización rectora de las normas internacionales y regulaciones necesarias para la seguridad, la eficiencia y la regularidad del transporte aéreo, promovió en 1980 la normalización y reglamentos oficiales relacionados con dichos documentos de viaje.

Debido al desarrollo y crecimiento de las nuevas Tecnologías de la Información y las Comunicaciones (TIC) y en un esfuerzo por hacer pasaportes más seguros se incorpora a los pasaportes la tecnología de tarjetas inteligentes como dispositivo de seguridad y control asegurando así la autenticidad e integridad de los datos del titular y dando paso a un nuevo concepto de pasaporte, el pasaporte electrónico.

Las soluciones existentes para el manejo de pasaportes electrónicos encargadas de leer y gestionar la información presente en este nuevo tipo de pasaporte se realizan a través de capas o librerías de software, técnicamente conocidas como middleware, desarrollados comúnmente como aplicaciones de escritorio. Esta tendencia trae consigo algunas desventajas a tener en cuenta; por ejemplo requieren de la presencia del middleware instalado de forma local en cada estación de trabajo para poder interactuar con los mismos, requiere que los usuarios posean ciertos conocimientos relacionados con el negocio presente en este tipo de documentos y para actualizar estos middleware, además de la necesidad de tener permisos de administración para la instalación de dichas tecnologías.

Mediante el uso de la tecnología web, aplicaciones de este tipo pueden ser accesibles desde cualquier punto de la red a través de una computadora, aportando una mayor independencia sobre la plataforma en la que se ejecuten, siendo a la vez más fácil entender su funcionamiento así como la facilidad para actualizar y mantener las aplicaciones sin distribuir e instalar software a miles de usuarios potenciales. Sin embargo dichos pasaportes se pueden utilizar en los servicios en línea, poniendo los correspondientes middleware del lado del servidor incrementando más la seguridad al usuario, así como la calidad, el fácil acceso y manejo de la información.

El departamento de Tarjetas Inteligentes del CISED cuenta con una plataforma para el desarrollo de servicios en línea con tarjetas inteligentes, denominada SmartCoP. Este sistema integra a JWebSocket, el

cual es un servidor de aplicaciones y a la vez un marco de trabajo para desarrollar aplicaciones en tiempo real para la web utilizando el protocolo WebSocket, que constituye una tecnología que proporciona un canal de comunicación bidireccional sobre un único socket TCP. Además de estar diseñada para ser implementada en navegadores y servidores web, por lo que puede utilizarse además por cualquier aplicación cliente/servidor.

Partiendo de todo lo anteriormente abordado se manifiesta la siguiente **situación problemática**:

La manera actual de interactuar con los pasaportes insertados en los lectores, es mediante middlewares que se encuentran del lado de la computadora del cliente. Esto, además de dificultar la incorporación de nuevas funcionalidades de forma transparente para el usuario, limita el acceso a los servicios en línea. Por otra parte, se necesita de cierto grado de conocimiento entorno al negocio presente en dichos pasaportes a la hora de actualizar estos middlewares, además de poseer permisos para poder instalar ciertas tecnologías.

Derivado de la situación anteriormente expuesta se encuentra el siguiente **problema de investigación:** ¿Cómo facilitar la lectura de pasaportes electrónicos garantizando la seguridad durante la transmisión de información a través de la web cumpliendo con los estándares internacionales de la OACI?

Por tanto la presente investigación centra su **objeto de estudio** en: Proceso de lectura de pasaportes electrónicos garantizando la seguridad durante transmisión de información.

Para dar solución al problema existente se ha tomado como **objetivo general:** Desarrollar mediante una aplicación web, la lectura de pasaportes electrónicos garantizando la seguridad durante la transmisión de información cumpliendo con los estándares internacionales de la OACI, utilizando la plataforma para el desarrollo de servicios en línea con tarjetas inteligentes, independizando al usuario de la instalación y actualización de middleware en el cliente.

Para dar cumplimiento al objetivo general planteado se han derivado los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación.
- Realizar el diseño de la solución para la lectura de pasaportes electrónicos.
- Implementar la solución para la lectura de pasaportes electrónicos.
- Realizar pruebas a la solución web desarrollada.

Por lo anteriormente expuesto el presente trabajo está sustentado en la siguiente **idea a defender**: El diseño e implementación de una aplicación web para la lectura de pasaportes electrónicos garantiza la transmisión de información de forma segura sin la necesidad de la instalación de middlewares en el cliente, cumpliendo con los estándares definidos por la OACI.

Para dar respuesta a los objetivos específicos trazados se plantea el cumplimiento de las siguientes **tareas de la investigación:**

1. Fundamentación teórica de la investigación y estudio de la situación actual sobre los pasaportes electrónicos en aplicaciones web.
2. Fundamentación de la metodología, estándares, herramientas y tecnologías a utilizar para el desarrollo de la solución propuesta.
3. Realización del análisis y diseño de las extensiones (plugin) que actúan como eje del middleware del lado del servidor JWebSocket.
4. Realización del diseño de interfaces web correspondientes a las extensiones generadas en el servidor.
5. Implementación de las extensiones del lado del servidor.
6. Implementación de las Interfaces correspondientes a las extensiones del lado del cliente.
7. Realización de las pruebas unitarias y de aceptación a la aplicación web para verificar el cumplimiento de los requisitos definidos.
8. Realización de pruebas de funcionamiento a la aplicación utilizando las tarjetas disponibles en el Dpto. de tarjetas inteligentes del CISED.

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

Métodos Teóricos:

- **Histórico-lógico:** Se utilizó para obtener conocimiento sobre el estudio de los antecedentes, evolución y desarrollo que han tenido los sistemas de lectura de documentos y los diferentes elementos asociados a las tecnologías de tarjetas inteligentes y de los pasaportes electrónicos, así como valorar las tendencias actuales sobre los métodos de lectura y procesamiento de las tarjetas inteligentes y pasaportes electrónicos empleados en las aplicaciones, permitiendo valorar deficiencias y proponer soluciones acorde a las necesidades.
- **Analítico-sintético:** Se empleó con el propósito para analizar toda la teoría recopilada a través de los diferentes medios bibliográficos que pueda contribuir a un mejor desarrollo del diseño del sistema, definir sus características generales, además de examinar elementos sobre el tema en cuestión.

- **Modelación:** Posibilitó realizar una representación de la situación que se analiza. Dando la posibilidad de obtener mediante diagramas y objetos una mayor compresión del problema y desarrollar un modelo para la aplicación a partir de la situación problemática.

Métodos Empíricos:

- **Entrevista:** Se aplicó con el propósito de recopilar información sobre los principales sistemas de lectura de pasaportes electrónicos, a partir de entrevistas y consultas a los tutores y otros especialistas en el tema de tarjetas inteligentes.

Los **posibles resultados** a obtener con el desarrollo de la investigación son:

- El desarrollo de una herramienta web que permita la lectura de pasaportes electrónicos cumpliendo con los estándares internacionales de OACI, esta herramienta facilitará al usuario interactuar con el pasaporte y leer los datos almacenados en él, comprobando su identidad y autenticidad, sin tener la necesidad de instalar ningún middleware en el cliente.
- Se agrega además, la documentación detallada con toda la base teórico-práctica asociada a la aplicación web y generada por la metodología que se seleccione para guiar el desarrollo.

Justificación de la investigación:

El departamento de Tarjetas Inteligentes del CISED inmerso en la obtención y desarrollo de aplicaciones para tarjetas inteligentes traza entre sus objetivos, aprovechando las ventajas de los servicios web en relación a componentes de pasaportes electrónicos, contar dentro de su gama de productos con una aplicación web que permita la lectura de pasaportes electrónicos cumpliendo con los estándares de la OACI, permitiendo obtener la información referente al titular y garantizando la seguridad durante la transmisión de información. Por tanto el aporte práctico de la presente investigación va dirigido a la obtención de:

- Aplicación web para la lectura de pasaportes electrónicos.
- Un prototipo de un sistema plataforma que forme parte de una futura herramienta que permita interactuar con los pasaportes electrónicos y terminales que sigan los estándares definidos por la OACI.

Para una mejor comprensión de la investigación, el contenido del presente trabajo de diploma se encuentra estructurado de la siguiente manera:

Capitulo I. Fundamentación teórica: Durante el desarrollo de este capítulo, se realiza la fundamentación teórica de la investigación, puntualizando los principales conceptos relacionados con el tema. Se expone un estudio del estado del arte en la gestión de los pasaportes mediante la web haciendo hincapié en el

proceso de lectura de pasaportes electrónicos en la actualidad, así como las principales metodologías, tecnologías, herramientas y lenguajes para el desarrollo de la aplicación en cuestión.

Capítulo II. Características, Análisis y Diseño del Sistema: En este capítulo se realiza una descripción general de la propuesta del sistema guiada por la metodología más conveniente para dar solución al problema científico. Se brinda una fundamentación de la solución propuesta, a partir de la cual se capturan los requisitos funcionales y no funcionales, se confeccionan las Historias de Usuarios(HU), se define un diagrama que organice la estructura lógica del producto y se describen otras actividades de análisis de la solución, seguidas por la descripción de los procesos del sistema y de la etapa de diseño, con el objetivo de plasmar las relaciones y posibles dependencias entre las distintas funcionalidades que la aplicación debe poseer.

Capítulo III. Implementación, Pruebas y Validación del Sistema: En el presente capítulo, se describe la etapa de implementación que conlleva a la obtención del software. Se elaboran, realizan y documentan las pruebas realizadas a la solución propuesta para demostrar el correcto funcionamiento de los requerimientos definidos en la etapa de análisis y diseño. Se realiza un análisis de los resultados de la aplicación en un entorno real, que tributen a la construcción de casos de pruebas de aceptación en base a las historias de usuarios.

Capítulo 1: Fundamentación Teórica.

1.1- Introducción.

El presente capítulo aborda una serie de aspectos relacionados con los pasaportes electrónicos, la seguridad de dicha tecnología y su uso actual en los países del mundo. Tiene como objetivo tratar sobre el estado del arte que presenta el tema que se investiga. Se hace mención de los estándares para regirse a la hora de trabajar con dicha tecnología y ser aplicada en los pasaportes, así como, la realización de un estudio de las metodologías de desarrollo de software, tecnologías, herramientas, lenguajes de programación y Entornos Integrados de Desarrollo (IDE) mayormente usadas para el desarrollo de aplicaciones en tarjetas inteligentes, específicamente para el desarrollo de la solución.

1.2- Pasaportes electrónicos.

El pasaporte electrónico también conocido como pasaporte biométrico, es un documento de identidad que combina el papel y una tarjeta inteligente que se comunica por radiofrecuencia. El mismo usa la biometría para autentificar la ciudadanía de los viajeros. La incorporación del minúsculo chip de identificación por radiofrecuencia (RFID, por sus siglas en inglés) en el documento permite tanto almacenar información adicional como duplicar la que se encuentra impresa en la página que contiene los datos del titular del pasaporte, permitiendo a través de una infraestructura de clave pública la certificación de la veracidad de los datos contenidos en él, haciendo virtualmente imposible forjar identidades falsas. Un pasaporte electrónico debe ajustarse totalmente a las especificaciones definidas en el estándar vinculado a los Documentos de Viaje de Lectura Mecánica (DVLM), denominado documento 9303, Parte 1, Volumen 1 así como, Volumen 2. El pasaporte-e tiene un período de validez el cual queda a discreción del estado expedidor, el que recomienda un período de validez que no supere los 10 años (1).

Los Estados podrían considerar un período más breve para permitir la actualización progresiva del pasaporte-e a medida que evoluciona la tecnología. Todos los pasaportes-e llevarán el símbolo siguiente como se muestra en la figura 1.

Figura 1: Logotipo de pasaporte-e. (1)

El chip utilizado en los pasaportes de lectura mecánica (PLM), es un chip sin contacto, que cumple con la Organización Internacional de Normalización (ISO, por sus siglas en inglés) 14443 (proximidad entre 0-10 cm); las razones por las que se seleccionó este estándar son:

1. **Interoperabilidad global:** al operar por radiofrecuencia (RF) hay diferentes bandas de RF usadas, pero la definida en la 14443 está disponible mundialmente. El uso de un estándar también prevé el uso de implementaciones propietarias.
2. **Diferentes métodos de inspección en la frontera:** pudiera ser que el portador del documento lo presente ante un oficial de inspección o un sistema automático, mediante el cual el chip es interrogado y se otorga o deniega automáticamente el permiso de entrada. La proximidad seleccionada admite cualquiera de estos dos tipos de inspección.
3. **El lector del chip:** solamente es necesario que requieran chips conformes a la ISO 14443-A y B.

En el PLM electrónico, el chip es pasivo, lo que significa que no contiene fuente de energía por sí solo. La razón para esto es que si contiene fuente de energía, no sería posible que durara el tiempo de vida esperado del PLM que es de 10 años, lo cual se logra siendo el lector el que provee la energía para comunicarse con el chip. El circuito integrado está formado por el chip y una antena, representado en la figura 2.

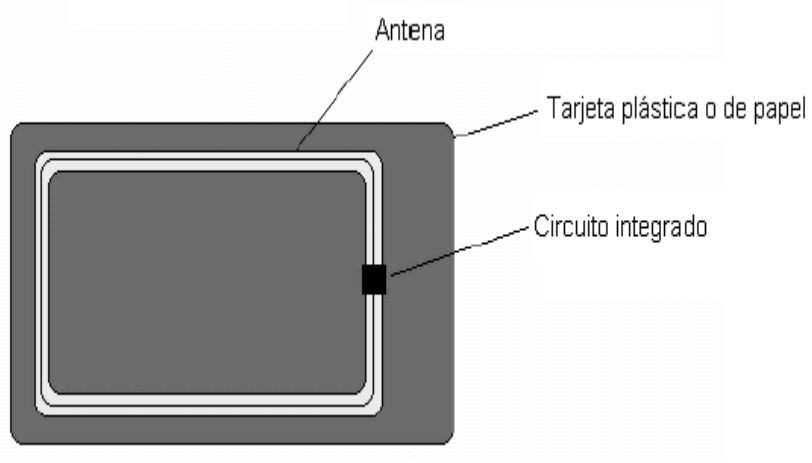


Figura 2: Estructura del chip (2)**.**

Los datos al ser almacenados en el chip requieren una estructura de datos estandarizada para habilitar una interoperabilidad global para PLM electrónicos facilitando que todas las naciones tengan conocimiento de cómo está estructurado el documento. La Estructura Lógica de Datos (LDS) constituye un estándar que define la colección de agrupaciones de elementos de datos almacenados dentro del chip de pasaporte, proporcionando detalles sobre la capacidad de la información, posicionamiento de los datos en la estructura de archivos y define medidas de seguridad para proteger la información interna del pasaporte.

Está conformado por elementos de datos (DE) de uso obligatorio u opcional y un orden agrupado en elementos de datos.Dentro de la LDS, los elementos de datos se agrupan según su organización lógica y son definidos como grupo de datos (DG), cada uno de ellos está identificado con un número. (Ver Anexo 1)

Los datos en el chip están almacenados en el sistema de ficheros definido por la ISO 7816-4. Los ficheros están organizados jerárquicamente en ficheros dedicados (DF) y ficheros elementales (EF). Los DF contienen los ficheros elementales y otros ficheros dedicados. Y un fichero maestro (MF), determinado por el sistema operativo, que será la raíz del sistema de ficheros. (Ver Anexo 2)

Cada grupo de datos consiste en una serie de datos dentro de una plantilla y será almacenado en un EF separado. La estructura y codificación de los datos está definida en la ISO 7816-4 y 7816-6. Cada dato posee una identificación Tag, que es especificada en un código hexadecimal. Cada dato contenido dentro de un grupo de datos posee una única Tag, longitud y valor, como se ilustra en la figura 3 (1).

ISSUING STATE OR ORGANIZATION APPLICATION				
Data Group	EF Name	Short EF identifier	FID	Tag
Common	EF.COM	'1E'	'01 1E'	'60'
DG1	EF.DG1	'01'	'01 01'	'61'
DG2	EF.DG2	'02'	'01 02'	'75'
DG3	EF.DG3	'03'	'01 03'	'63'
DG4	EF.DG4	'04'	'01 04'	'76'
DG5	EF.DG5	'05'	'01 05'	'65'
DG6	EF.DG6	'06'	'01 06'	'66'
DG7	EF.DG7	'07'	'01 07'	'67'
DG8	EF.DG8	'08'	'01 08'	'68'
DG9	EF.DG9	'09'	'01 09'	'69'
DG10	EF.DG10	'0A'	'01 0A'	'6A'
DG11	EF.DG11	'0B'	'01 0B'	'6B'
DG12	EF.DG12	'0C'	'01 0C'	'6C'
DG13	EF.DG13	'0D'	'01 0D'	'6D'
DG14	EF.DG14	'0E'	'01 0E'	'6E'
DG15	EF.DG15	'0F'	'01 0F'	'6F'
DG16	EF.DG16	'10'	'01 10'	'70'
Security Data	$EF.SO_D$	'1D'	'01 1D'	'77'

Figura 3: Ficheros almacenados en el chip del pasaporte. (3)

El EF.COM almacena los datos comunes que corresponden fundamentalmente a la organización de los datos dentro del chip. El identificador corto del fichero es 30 ('1E'). Cada grupo de datos deberá ser almacenado en un EF accesible por un identificador corto del fichero. El EF deberá tener nombre de fichero que se corresponderá con el grupo de datos que contenga, el nombre del fichero EF que contiene los datos de seguridad se denomina EF.SOD (1).

Rótulo	**L**	**Valor**
'5F01'	04	Número de versión LDS con formato aabb, donde aa define la versión de la LDS y bb define el nivel de actualización.
'5F36'	06	Número de versión Unicode con formato aabbcc, donde aa define la versión mayor,bb la versión menor y cc define el nivel de liberación.
'5C'	X	Lista de rótulos. Lista de todos los grupos de datos presentes.

Tabla 1. Ejemplo de los valores en el EF.COM.

El siguiente ejemplo indica la implementación del LDS versión 1.7, usando la versión unicote 4.0.0 teniendo los grupos 1 (tag '61'), 2 (tag '75'), 4 (tag '76'), y 12 (tag '6C') presentes.

En este ejemplo está escrito en negrita el valor del Tag, en cursiva la longitud y los valores en negro.

'60'*'16'*

'5F01'*'04'*0107'

'5F36'*'06'*040000'

'5C'*'04'*6175766C'

Estos tipos de documentos requieren una infraestructura tecnológica necesaria para la captura de datos e imágenes de los ciudadanos, validación de la identidad, personalización del documento y entrega del mismo, paso fundamental para la emisión del pasaporte, ilustrado en la figura 3. Cada uno de estos componentes se concibe de forma independiente, funcionando de forma modular.

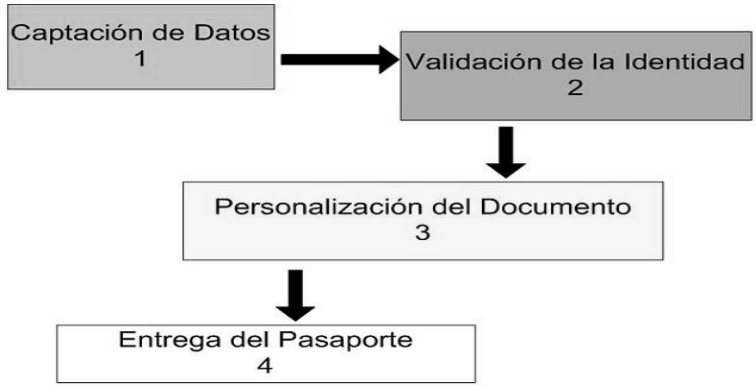


Figura 4: Pasos para la emisión de pasaporte. (2)

La personalización de este tipo de documento cuenta con dos factores:

- Personalización Eléctrica: se refiere a la escritura de los datos en el chip (4).
- Personalización Óptica: Se personalizan los datos que se encuentran en la zona de inspección visual (ZIV) (4).

1.3- Seguridad en los pasaportes electrónicos.

En los últimos años se han reportado gran cantidad de falsificaciones, fraudes y copias ilegales de pasaportes como consecuencia del incremento en la delincuencia internacional y la inmigración ilegal, se ha agudizado la preocupación por la seguridad de los documentos de viaje y por lo que podría hacerse para mejorar su resistencia a las adulteraciones o a un uso indebido. Con este objetivo los asesores técnicos de la OACI, decidieron que sería conveniente publicar un conjunto de normas mínimas de seguridad recomendadas como orientación para todos los Estados que expidan pasaportes electrónicos.
La OACI propone varios mecanismos de seguridad, de ellos algunos son de cumplimiento obligatorio y otros opcionales, según decisión del país emisor. Estos mecanismos se mencionan a continuación:

- Autenticación Pasiva (Es de uso obligatorio).
- Autenticación Activa (Es de uso opcional).
- Control de Acceso Básico (Es de uso opcional).
- Control de Acceso Extendido (Es de uso opcional).
- Encriptación de Datos (Es de uso opcional).

1.4- Tarjetas inteligentes.

Las tarjetas inteligentes fueron concebidas y patentadas por alemanes, japoneses y franceses en 1970. Estos dispositivos son tarjetas de plástico que contienen un circuito integrado, son semejantes en tamaño y otros estándares físicos a las tarjetas de crédito. Dicho circuito puede ser de solo memoria o contener un microprocesador con un sistema operativo que le permite un grupo de tareas tales como:

1. Almacenar información.
2. Incriptar información.
3. Leer y escribir datos, como una computadora (5).

Las tarjetas inteligentes con su seguridad hacen que los datos que se consideran personales solo sean accesibles por los usuarios apropiados. Las tarjetas inteligentes aseguran la portabilidad, seguridad y confiabilidad en los datos (5).
Son las tarjetas inteligentes basadas en estándares las utilizadas para autenticar la identidad de una persona o entidad. Es importante destacar el nivel de seguridad que poseen para el almacenamiento de la

información. La tarjeta inteligente puede ser multi-aplicación, o sea puede contener información de una o varias aplicaciones al mismo tiempo, esto puede ocurrir debido a la característica que poseen, ya que tienen una jerarquía de almacenamiento de la información. Cada una de las aplicaciones contenidas en la tarjeta se protege independientemente (5).

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

- Telefonía Móvil Digital (GSM): donde la identificación del titular del número telefónico, la herramienta de cifrado y la base de datos del propietario constituyen las funciones de la tarjeta inteligente.
- Banca: su desarrollo está dado a través de los monederos electrónicos, permitiéndole al usuario cargarlo con una determinada cantidad de dinero y realizar compras teniendo en cuenta el saldo disponible en la tarjeta (5).

Las tarjetas inteligentes son clasificadas de acuerdo a sus capacidades, estructura del sistema operativo, formato e interfaz de comunicación. Esta última está compuesta por otras dos clasificaciones: tarjetas de contacto y tarjetas sin contacto. Las tarjetas de contacto disponen de unos contactos metálicos visibles y debidamente estandarizados (parte 2 de la ISO y la Comisión Electrotécnica Internacional (IEC) 7816), no siendo así en las tarjetas sin contacto. Las tarjetas de contacto, por su parte, deben ser insertadas en una ranura de un lector para llevar a cabo operaciones en las mismas. Luego el lector alimenta eléctricamente a la tarjeta y transmite los datos oportunos para operar con ella conforme al estándar. Por su parte las tarjetas sin contacto utilizan la tecnología de identificación por RFID para leer y escribir desde el chip incrustado en la tarjeta.

1.5- XML.

Es un Lenguaje de Marcado Extensible que ofrece un formato para la descripción de datos estructurados. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Es un subconjunto especializado en gestionar información para la web. Proporciona un estándar de datos que puede codificar el contenido, la semántica y los esquemas de una gran variedad de casos, desde los más simples a los más complejos. XML no es más que un conjunto de reglas para definir etiquetas semánticas que organizan un documento en diferentes partes.XML garantiza que los datos estructurados sean independientes de aplicaciones o fabricantes. La interoperabilidad que se ha logrado está creando rápidamente una nueva generación de aplicaciones de comercio electrónico en la web (6).

1.6- Librería ExtJS.

ExtJS es una biblioteca o conjunto de librerías de JavaScript para el desarrollo de aplicaciones web interactivas que además de flexibilizar el manejo de componentes de la página como el DOM[1], peticiones AJAX[2] y DHTML[3], tiene la gran funcionalidad de crear completas interfaces de usuario bastante funcionales, fáciles de usar, muy parecidas a las conocidas aplicaciones de escritorio. Esto permite a los desarrolladores web concentrarse en la funcionalidad de las aplicaciones en vez de en las advertencias técnicas.

Esta librería incluye:

- Componentes UI (User Interface) del alto rendimiento y personalizables.
- Modelo de componentes extensibles.
- Un API (Application Programming Interface) fácil de usar.
- Licencias de código abierto (Licencia Pública General, GPL según sus siglas en inglés) y comerciales.

Cuenta con un framework que posee un conjunto de componentes, denominados “widgets”, los cuales pueden ser incluidos dentro de una aplicación web. Varios de estos componentes están dotados de comunicación con el servidor usando AJAX. También contiene numerosas funcionalidades que permiten añadir interactividad a las páginas HTML (7).

ExtJS soporta todos los navegadores web más importantes, incluyendo Mozilla FireFox 1.5 o superior (PC, Mac),Safari 3.0 o superior, Google Chrome 3.0 o superior, Opera 9.0 o superior (PC, Mac) e Internet Explorer 6.0 o superior.

Ventajas:

1. Una de las grandes ventajas de utilizar ExtJS es que nos permite crear aplicaciones complejas utilizando componentes predefinidos.
2. Evita el problema de tener que validar el código para que funcione bien en cada uno de los navegadores (Mozilla Firefox, Internet Explorer, Safari, Opera etc.).
3. El funcionamiento de las ventanas flotantes lo pone por encima de cualquier otro.
4. Relación entre cliente-servidor balanceado: Se distribuye la carga de procesamiento entre el cliente y el servidor, permitiendo que el servidor pueda atender más clientes al mismo tiempo.

[1] ***Document Object Model,*** según sus siglas en español Modelo de Objetos del Documento o Modelo en Objetos para la Representación de Documentos.
[2] acrónimo de ***Asynchronous JavaScript And XML,*** en español JavaScript Asíncrono y XML.
[3] **DHTML**, siglas de ***Dynamic HyperText Markup Language***(Lenguaje de Marcado HiperTextual Dinámico)

5. Eficiencia de la red: Disminuye el tráfico en la red pues las aplicaciones cuentan con las posibilidades de elegir que datos desean trasmitir al servidor y viceversa (puede variar con el uso de aplicaciones de pre-carga).
6. Comunicación asíncrona: En este tipo de aplicación el cliente puede comunicarse con el servidor sin necesidad de estar sujeto a un clic o una acción del usuario, dándole la libertad de cargar información sin que el usuario se dé cuenta.

Desventajas:

1. Necesidad de una plataforma: pues dependemos del paquete ExtJS para obtener los resultados deseados.
2. Para algunos, el no contar con una licencia LGPL[4].

1.7- Protocolo WebSocket.

Desde sus inicios la comunicación en la web fue soportada por el protocolo HTTP (Hypertext Transfer Protocol). Inicialmente diseñado como protocolo de solicitud-respuesta para el intercambio de documentos, HTTP no logra satisfacer las nuevas necesidades de la red de redes. A lo largo de los últimos años, se han implementado varias técnicas y tecnologías en aras de lograr una comunicación web en tiempo real y bidireccional. Técnicas como el polling, longpolling, streaming y tecnologías como AJAX y Comet han formado parte de este esfuerzo, logrando algunas de ellas resultados favorables pero muy costosos.

Sin embargo el protocolo WebSocket define los procedimientos para actualizar la conexión a través de HTTP a una conexión mediante WebSocket totalmente bidireccional usando TCP, logrando el tiempo real con altos niveles de escalabilidad. WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y full-dúplex sobre un único socket TCP (Transmission Control Protocol). Soluciona las limitaciones del protocolo HTTP, al establecer una comunicación full-dúplex (TCP) entre el cliente y el servidor, sustituyendo la comunicación half-duplex (HTTP). Se reduce, en grandes proporciones, el tráfico en la red teniendo en cuenta que al establecer la comunicación WebSockets entre el cliente y el servidor solo hay un envío de 2 bits, eliminando las cabeceras HTTP.

Los principales servidores que soportan WebSockets para el desarrollo de aplicaciones hoy día son, la pasarela WebSockets de Kaazing, JettyWebSocketServlet, Socket.IO, django-websocket del proyecto Python y JWebSocket. JWebSocket es un framework de código abierto para el desarrollo de aplicaciones

[4] ***Lesser General Public Licence*** (en español, Licencia Pública General Reducida): Licencia de software que pretende garantizar la libertad de compartir y modificar el software, sea propietario o no, cubierto por ella, asegurando que el software es libre para todos sus usuarios.

web estacionarias y móviles. El cliente JWebSocket se basa en el lenguaje de programación JavaScript con una arquitectura de plugin que permite aumentar con facilidad sus funcionalidades y por el lado del servidor se hace uso del lenguaje de programación Java, este está diseñado para funcionar como servidor de comunicaciones o como servidor web, ofreciendo las ventajas de ejecutarse fácilmente desde una línea de comandos, integrarse a la biblioteca de una aplicación existente de Java y una total flexibilidad.

1.8- Estándares.

1.8.1- Documento 9303.

El documento 9303 es un estándar para ciertos tipos de documentos de viajes, cuenta con tres partes y ha sido aprobado por la OACI con carácter de normas ISO 7501-1, 7501-2 y 7501-3. El documento contiene las especificaciones actuales de la OACI para hacer funcionales y seguros los pasaportes de lectura mecánica y electrónicos, visados, además de documentos de identidad utilizados por cualquier estado.

Se definen normas a las que deben ajustarse, elementos de seguridad física y lógica. Medidas para la elaboración de los documentos, la transportación y la personalización de los mismos, así como el formato que deben llevar, las dimensiones, la calidad y el posicionamiento de los elementos en el documento. Con respecto a los pasaportes electrónicos se proponen algunos métodos de seguridad lógica, en los cuales se incluye la infraestructura de clave pública, así como la estructura lógica que deben tener los datos dentro del circuito integrado para que puedan ser leídos desde cualquier sistema que se ajuste a dichas normas (8).

1.8.2- ISO 14443.

ISO 14443 es un estándar internacional relacionado con las tarjetas inteligentes sin contacto, gestionado conjuntamente por la ISO y la IEC. Este estándar define los parámetros a seguir por las tarjetas de proximidad que utilizan el sistema RFID, componentes que deben incluir la misma, frecuencia por la que debe operar la antena incluida en el sistema RFID, así como el comportamiento del lector con respecto a la tarjeta.

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

- ISO/IEC 14443-1: Características físicas.

- ISO/IEC 14443-2: Energía de radiofrecuencia y señal de interfaz.
- ISO/IEC 14443-3: Inicialización y anticolisión.
- ISO/IEC 14443-4: Protocolo de transmisión.

1.8.3- ISO 7816.

Toda tarjeta que posea un circuito integrado debe cumplir los estándares de la serie ISO 7816 con el objetivo de estos es lograr la interoperabilidad entre distintos fabricantes de tarjetas y lectores de las mismas, en lo que respecta a características físicas, comunicación de datos y seguridad. Estos estándares son basados en los ISO 7810 e ISO 7811, los cuales definen características físicas de tarjetas de identificación.

A continuación se describen las diferentes partes del estándar ISO 7816 (9):

- Parte 1: Características físicas.
- Parte 2: Tamaño y localización de los contactos.
- Parte 3: Señales electrónicas y protocolos de transmisión.
- Parte 4: Organización, seguridad y comandos para el intercambio de información.
- Parte 5: Sistema de numeración y procedimiento de registro para los identificadores de aplicación.
- Parte 6: Interoperabilidad en los elementos de datos para el intercambio.
- Parte 7: Interoperabilidad en los comandos de la tarjeta.
- Parte 8: Comandos para operaciones de seguridad.
- Parte 9: Comandos para la gestión de la tarjeta.
- Parte 10: Señales electrónicas para tarjetas síncronas.
- Parte 11: Verificación de la identidad personal a través de métodos biométricos.
- Parte 12: Tarjetas con contactos. Interfaz eléctrica Universal Serial Bus (USB) y procedimientos operativos.
- Parte 15: Aplicación de información criptográfica.

1.8.4- Estándar PC/SC.

PC/SC (del inglés Personal Computer/Smart Card) es un conjunto de especificaciones para la integración de tarjetas inteligentes en ordenadores personales. En particular se define un API de programación, que permite a los desarrolladores trabajar de forma uniforme con lectores de tarjetas de distintos fabricantes (que cumplan con la especificación).La API de PC/SC está incorporada en sistemas Microsoft Windows 200x/XP y disponible también Microsoft Windows NT/9x. También hay una implementación libre, de código abierto, llamada PC/SC Lite (proyecto MUSCLE) para sistemas operativos GNU Linux.

La ventaja de PC/SC es que las aplicaciones no tienen que reconocer los detalles correspondientes al lector de tarjetas inteligentes en la comunicación con la tarjeta inteligente. Esta aplicación puede funcionar con cualquier lector que cumple con el estándar PC / SC (10).

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 (11).

- 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 técnica de recursos.
- Parte 6: Definición de la interfaz del proveedor de servicios.
- 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.

1.9- Entorno de desarrollo para la solución del problema.

1.9.1- Metodologías de desarrollo del software.

Para alcanzar el éxito en el proceso de desarrollo de software y ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto, se hace necesario contar con una metodología de desarrollo de software que guie el proceso de los elementos involucrados hasta alcanzar la satisfacción de los clientes que lo utilicen.

Una metodología de desarrollo de software, en ingeniería de software, es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. El éxito del producto depende en gran parte de la metodología escogida por el equipo, ya sea tradicional o ágil, donde los equipos maximicen su potencial, aumenten la calidad del producto con los recursos y tiempos establecidos.

1.9.1.1- Metodologías Tradicionales.

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo de software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del

producto. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar.

Las metodologías tradicionales (formales) se focalizan en la documentación, planificación y procesos. Entre las metodologías tradicionales o pesadas podemos citar: RUP (*Rational Unified Procces*), MSF (*Microsoft Solution Framework*), Win-Win Model Spiral, Iconix, entre otras. En el caso particular de RUP, se presenta un especial énfasis en cuanto a su adaptación a las condiciones del proyecto mediante su configuración previa a aplicarse.

RUP (Proceso Unificado de Rational) es una metodología que constituye un proceso de desarrollo formal, cuyo objetivo fundamental se centra en la producción de software con alto nivel de calidad. Satisfacer los requerimientos de los usuarios finales también es premisa para RUP, todos estos requisitos deben cumplirse sin violar el cronograma ni el presupuesto propuesto. RUP es un proceso iterativo e incremental, centrado en la arquitectura y guiado por los casos de uso, utiliza UML como lenguaje de notación. Incluye artefactos y roles (12).

Esta metodología organiza el proceso de desarrollo de software en cuatro fases: Inicio, Elaboración, Construcción y Transición, en las que se pueden distinguir nueve flujos o disciplinas a realizar en mayor o menor medida en dependencia de cada etapa. Es el proceso de desarrollo más general de los existentes actualmente.

1.9.1.2- Metodologías Ágiles.

Constituye una metodología para el desarrollo de software y consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. Entre las metodologías ágiles más destacadas hasta el momento se puede nombrar: XP (*Extreme Programming*), Scrum, Crystal Clear, DSDM (*Dynamic Systems Development Method*), FDD (*Feature Driven Development*), ASD (*Adaptive Software Development*), XBreed, Extreme Modeling, entre otras.

Scrum: está indicado para proyectos en entornos complejos, donde los requisitos sean cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales. El software se desarrolla mediante iteraciones conocidas como *sprints*, el resultado de estas iteraciones se le muestra al cliente. Se realizan además diferentes reuniones, por ejemplo, la reunión diaria de 15 minutos (13).

XP (*eXtreme Programing*): es la más sobresaliente dentro de las metodologías ágiles. Está diseñada para que el cliente reciba el software que necesita en el tiempo que lo necesita. Su principal objetivo es satisfacer las necesidades de los usuarios y es por ello que se considera un éxito. El trabajo es desarrollado en equipo, preocupándose en todo momento del aprendizaje de los desarrolladores y estableciendo un buen clima de trabajo. XP le confiere las facultades a los desarrolladores para que puedan responder con confianza a las necesidades cambiantes de los clientes. Es la metodología más apropiada para entornos volátiles. Este tipo de metodología es la adecuada para los proyectos con requisitos imprecisos, muy cambiantes y con un riesgo técnico excesivo (14).

Ventajas:

- Apropiado para entornos volátiles, equipos de desarrollo pequeños (de 2 a 10 desarrolladores) y proyectos de alto riesgo.
- Permite una mejor adaptabilidad a los cambios, que se traduce en una reducción de costos.
- Planificación a corto plazo y más transparente para los clientes, ya que conocen las fechas de entrega de funcionalidades vitales para su negocio.
- Permite tener retroalimentación continua de los usuarios a través de las entregas frecuentes.

Desventajas:

- A veces cuesta delimitar el alcance del proyecto con el cliente.

Propone cuatro fases para el proceso de desarrollo del software conjugadas con el uso de doce prácticas. La mayoría de estas prácticas no son nuevas, sino que han sido reconocidas por la industria como las mejores prácticas durante el año:

Prácticas:

- Planificación incremental.
- Pruebas.
- Programación en parejas.
- Refactorización.
- Diseño simple.
- Propiedad colectiva del código.
- Integración continua.
- Cliente en el equipo.
- Entregas pequeñas.
- Semanas de 40 horas.
- Estándares de codificación.
- Uso de metáforas.

Artefactos esenciales en XP:

- Historias del usuario.
- Tareas de ingeniería.
- Pruebas de aceptación.

- Pruebas unitarias y de integración.
- Plan de entrega.
- Código.

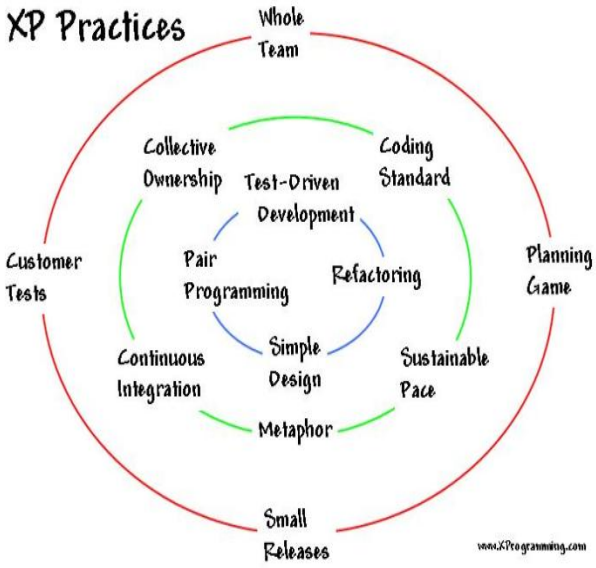


Figura 5: Modelo de Extreme Programming (15).

Fases de XP:

La metodología XP, como metodología ágil, enfatiza en el carácter interactivo e incremental del desarrollo, donde una iteración es un período de una a cuatro semanas, en el cual el cliente selecciona las funcionalidades que desea que se implementen en dicha iteración. Las fases en las que se subdivide el ciclo de vida de un proyecto de software con XP se describen a continuación:

1. **Exploración.**

En esta fase, los clientes plantean a grandes rasgos las Historias de Usuario (HU) que son de interés para la primera entrega del producto, además se confecciona la metáfora del sistema ayudando al equipo a entender las relaciones entre los principales componentes del sistema. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.

2. **Planeamiento.**

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. Una entrega debería obtenerse en no más de tres meses.

En esta fase los artefactos que se generan son el plan de iteraciones donde los elementos a tener en cuenta durante su elaboración son: historias de usuario no abordadas, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior y el plan de entrega, compuesto por iteraciones de no más de tres semanas.

3. **Iteraciones.**

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. 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. Se diseñan las tarjetas CRC (Clase, Responsabilidad, Colaboración).

Para cada una de estas iteraciones se confeccionarán las unidades de prueba y se aprobarán las mismas antes de empezar a codificar la solución, estas ayudarán a los programadores a tener una mejor visión del comportamiento del programa. Luego se aplicarán las pruebas unitarias y de aceptación, estas últimas diseñadas por el cliente, destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida. El objetivo de estas pruebas es verificar el cumplimiento de los requisitos.

4. **Producción.**

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, ya que realizar cambios durante esta fase implica costos, entonces se tendría que analizar la factibilidad de estos cambios.

1.9.1.3- Criterio de selección de la metodología a utilizar.

A la hora de seleccionar la metodología que guiará el proceso de desarrollo de la aplicación, realizamos un estudio previo seleccionando dos de las metodologías que son fácilmente reconocibles por cualquier ingeniero de software y profesionales en el sector y que nos permitía sacar a relucir sus características principales. A raíz de lo abordado anteriormente establecimos una comparación entre las mismas mediante la especificación de varias características. (Ver diferencias entre metodologías: Anexo 3)

Luego de la comparación entre las metodologías seleccionadas se llegó a la conclusión que la metodología XP regirá el proceso de desarrollo de software por estar basada en la simplicidad, la comunicación, la retroalimentación y la refactorización de código. Además como toda metodología ágil intenta reducir la complejidad del software por medio de un trabajo orientado directamente al objetivo, apoyada en las relaciones interpersonales y la velocidad de reacción.

1.9.2- Lenguaje de Modelación Visual.

El modelado de sistemas de software es una técnica que permite manejar la complejidad de los sistemas a analizar o diseñar. Para facilitar el modelado en forma general es necesario abstraer la complejidad en modelos de forma que el ser humano pueda entender. Estos tienen como objetivo capturar las partes esenciales del sistema y por otra parte es independiente del lenguaje de implementación, de tal forma que los diseños realizados usando lenguaje de modelación unificado se puedan implementar en cualquier lenguaje soportando las principales posibilidades de estos orientados a objetos.

- ***El Lenguaje de Modelado Unificado*** (UML – *Unified Modeling Language*) es un lenguaje que permite modelar, construir y documentar los elementos que forman un producto de software, ofrece aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados; incluidos en un estándar que permite describir mediante un plano, partes que conforman el modelo del sistema de una manera simple (16).

Entre sus objetivos fundamentales se encuentran:

1. Ser tan simple como sea posible, pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesita construir.
2. Necesita ser lo suficientemente expresivo para manejar todos los conceptos que se originan en un sistema moderno, tales como la concurrencia y distribución, así como también los mecanismos de la ingeniería de software, como son el encapsulamiento y los componentes.
3. Debe ser un lenguaje universal, como cualquier lenguaje de propósito general.
4. Imponer un estándar mundial.

1.9.3- Herramientas para el diseño de la aplicación.

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Las mismas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. Además ayuda a la reutilización del software, portabilidad y estandarización de la documentación y facilita el uso de las distintas metodologías propias de la ingeniería del software. Entre las herramientas CASE más utilizadas podemos citar: Erwin, ERStudio, System Architect, Rational Rosse, Visual Paradigm, Altova Umodel, entre otras.

1.9.3.1- Altova Umodel.

Es una herramienta de modelado basado en UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite crear e interpretar diseños software mediante la potencia del estándar UML, hacer ingeniería inversa de programas existentes a diagramas UML claros y precisos para abarcar rápidamente su arquitectura de software, generar código Java o C# a partir de sus planos y la documentación del proyecto.

Tiene la capacidad de crear todos los diagramas UML descritos relacionados con el sistema, así como también diagramas *SysML*[5], o proyectos orientados a la generación de *Schemas XML*[6]. Su rica interfaz visual y usabilidad superior ayudan a disminuir la curva de aprendizaje de UML, permitiendo que desarrolladores, incluso los nuevos en el modelado de software, manejen rápidamente UML, potenciando su productividad y maximizando resultados; además apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros (17).

Tiene como ventajas:

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Generación de código e ingeniería inversa: Brinda la posibilidad de generar código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.
- Generación de documentación: Brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.
- Soporta la especificación de intercambio XML 2.1 para permitir abrir y editar modelos creados en herramientas UML más caras o de manejo más complicado.

[5] Es un lenguaje de modelado visual para ingenieros de sistemas, derivado de UML, que proporciona un lenguaje de modelado de propósito general para apoyar la especificación, análisis, diseño y verificación de los sistemas complejos.

[6] Lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML.

- Es compatible con diferentes entornos de desarrollo: Eclipse, Borland, Jbuilder, Microsoft Visual J# .NET.

Desventajas:

- No permite definir asociaciones entre estereotipos en un perfil. Estas asociaciones son básicas en la creación de perfiles, pero esta restricción no constituye un impedimento fundamental para su utilización.

Luego de una minuciosa apreciación entre características, ventajas y desventajas se decidió utilizar la herramienta Altova Umodel en la representación de diagramas, ya que permite crear e interpretar diseños de software mediante la potencia del estándar UML. Por otra parte, genera código Java o C# a partir de sus planos, o realizar ingeniería inversa de programas existentes a diagramas UML claros y precisos para abarcar rápidamente su arquitectura de software.

1.9.4- Lenguajes de programación.

Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser ejecutados por máquinas computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación.

1.9.4.1- Lenguajes en el lado del cliente.

JavaScript es un lenguaje de script basado en objetos, utilizado para acceder a objetos en aplicaciones. Principalmente se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. Aunque comparte muchas de las características y de las estructuras del lenguaje Java, fue desarrollado independientemente. El lenguaje JavaScript puede interactuar con el código HTML, permitiendo a los programadores web utilizar contenido dinámico.

No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos. Es utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos (18).

1.9.4.2- Lenguajes en el lado del servidor.

El lenguaje de programación **Java,** fue diseñado por la compañía Sun Microsystems Inc., con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma (19).

El lenguaje fue diseñado con las siguientes características:

- Simple: Elimina la complejidad de los lenguajes como "C" y da paso al contexto de los lenguajes modernos orientados a objetos. Orientado a Objetos. La filosofía de programación orientada a objetos es diferente a la programación convencional.
- Robusto: Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.
- Seguro: El sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.
- Portable: Especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.
- Independiente a la arquitectura: Al compilar un programa en Java, el código resultante un tipo de código binario conocido como byte code. Este código es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma.
- Multiproceso: Puede ejecutar diferentes líneas de código al mismo tiempo.
- Interpretado: Java corre en máquina virtual.

Para la implementación del lado del servidor el lenguaje que se propone a utilizar es Java ya que constituye el lenguaje sobre el que se implementa el marco de trabajo JWebSocket.

1.9.5- Entornos integrados de desarrollo.

Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de *Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, que

consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje programación Java. Existe además un número importante de módulos para extenderlo. Constituye un proyecto de código abierto de gran éxito con una gran base de usuarios y una comunidad en constante crecimiento. Su plataforma permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos (20). Para el desarrollo de la solución se utilizará como IDE de desarrollo NetBeans ya que dará soporte al lenguaje seleccionado con el objetivo de crear aplicaciones multiplataforma.

1.10- Propuesta del entorno de desarrollo.

Tras el análisis de las diversas ventajas y desventajas de las metodologías, tecnologías y herramientas existentes y más usadas en la actualidad se define el uso de XP como metodología de desarrollo de software por estar definida para equipos pequeños y comprometidos con el desarrollo. El modelado del software se realiza a través del UML con el objetivo de visualizar, especificar, construir y documentar un sistema. La herramientas CASE a utilizar es la versión 10.0 de AltovaUModel Enterprise Edition, por ser un software que permite realizar ingeniería tanto directa como inversa y soportar múltiples usuarios trabajando sobre el mismo proyecto, además de generar la documentación del proyecto automáticamente y es además una herramienta multiplataforma. El lenguaje de programación que se utiliza es Java por ser orientado a objetos, incluyendo mejoras derivadas de otros lenguajes, además de constituir el lenguaje sobre el cual se implementa JWebsocket, este lenguaje dará soporte a la metodología de desarrollo de software seleccionada. Dicho lenguaje se pone en práctica haciendo uso de Netbeans como IDE al permitir la creación de aplicaciones de alta calidad con gran rapidez, seguridad y confiabilidad. Además de se utiliza la librería ExtJS para la creación de la interfaz de usuario y el lenguaje XML para la especificación de datos estructurados.

1.11- Análisis de soluciones existentes.

1.11.1- CoesyseGov 2.0.

El objetivo de este producto es el de autenticar a los ciudadanos a través de la web para que tengan acceso a los servicios en línea de e-Government[7]. CoesyseGov 2.0 producido por Gemalto[8], permite un

[7] e-Goverment: gobierno electrónico.

servicio de identificación electrónica mediante tarjetas inteligentes basado en la web, en vez de un software basado en un cliente de autenticación de instalación local. Esta solución evita la administración de un software middleware en el cliente, toda la funcionalidad requerida se centraliza en un servidor. CoesyseGov 2.0 se presenta para solucionar el problema de emisión de certificados, pues la generación de llaves y solicitud/carga de certificados necesitan llevarse a cabo en las tarjetas inteligentes en modo de post-emisión. No requiere software en la computadora del cliente, simplificando el despliegue de servicios y potenciando una mayor asimilación. Entre sus características principales están:

- Servicios de conectividad de tarjetas inteligentes.
- Servicio de autenticación.
- Federación de identidad.

La solución CoesyseGov 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 tecnología de conectividad SConnect: conectividad sin fallas entre la tarjeta y el servidor de autenticación.

CoesyseGov 2.0 funciona con cualquier navegador en cualquier sistema operativo permitiendo una conexión “plug and play[9]” desde todos los ordenadores. Su tecnología principal se sustenta en el producto SConnect de la misma compañía, que permite una conexión neutral con tarjetas inteligentes. Representa un cambio en el paradigma de las tarjetas inteligentes y los servicios web (21).

SConnect:

SConnect es una extensión 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.

[8] Empresa internacional de seguridad digital que proporciona aplicaciones de software, dispositivos de seguridad personal como tarjetas inteligentes, además de servicios gestionados.

[9] 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.

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

1.11.2- JMRTD.

JMRTD[10] es aplicación gratuita y de código abierto de los DVLM, estándares especificados por la OACI. El pasaporte electrónico es una aplicación de estas normas. El proyecto desarrolló una aplicación del lado de la tarjeta (el "applet pasaporte") y una API del lado del anfitrión para acceder a los pasaportes electrónicos. El applet pasaporte permite crear sus propios pasaportes. La API del lado del anfitrión hace que sea posible autenticar con un pasaporte y leer la información en el chip (22).
Principales características:

- Licencias: LGPL.
- Sistemas operativos: Independiente del SO.
- Implementación: Java.

Esta solución brinda ayuda, soporte y herramientas para los interesados en crear proyectos y aplicaciones acerca de los pasaportes electrónicos, lo cual nos sirvió de base para la realización de nuestro trabajo.

A partir del análisis realizado a los sistemas estudiados, estos solo ayudaron a comprender características y aspectos teóricos relacionados con el proceso de lectura. Aunque se pudo apreciar que en la actualidad existen aplicaciones capaces de realizar la lectura de pasaportes electrónicos, resultan ser sistemas propietarios como el caso de CoesyseGov 2.0 y poder adquirirlos resulta muy costoso, en cambio JMRTD es una aplicación de código abierto que necesita de la instalación de software adicional, por esta razón se propone realizar una aplicación con características similares.

1.12- Conclusiones parciales.

- El estudio de las principales características de las tarjetas inteligentes y la organización de la información en los pasaportes electrónicos, permitió identificar los principales elementos de seguridad y los estándares asociados a este tipo de tecnologías, facilitando su comprensión.

- Partiendo del análisis de las diferentes metodologías, tecnologías y lenguajes según las necesidades de la solución, se determinó la utilización de la metodología XP, UML como lenguaje de modelado y la herramienta Altova UModel como lenguaje de modelado. Como lenguaje de programación Java, usando el IDE de desarrollo Netbeans.

[10] Software para la lectura de pasaportes implementada en Java.

Capítulo 2: Características, Análisis y Diseño del Sistema.

2.1- Introducción.

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 definidas en la metodología de desarrollo de software Extreme Programming, normalmente conocida como XP, además de presentar los diferentes artefactos generados en las mismas, los cuales serán premisas cruciales para la entrega final de la aplicación.

2.2- Descripción del problema.

En el departamento de Tarjetas Inteligentes, perteneciente al Centro de Identificación y Seguridad Digital se desarrollan productos y soluciones relacionadas con tarjetas inteligentes. Actualmente existe una plataforma para el desarrollo de servicios en líneas con tarjetas inteligentes (SmartCoP) que pretende integrar soluciones y componentes para trabajar con pasaportes electrónicos. La aplicación a desarrollar permite la lectura de los pasaportes electrónicos a partir de la colocación de éste en el lector y aporta al departamento la visualización de la información contenida del titular presente en los mismos que pudiera ser utilizada para detectar posibles vulnerabilidades de acceso en los sistemas de reconocimiento de identidad. Desarrollar esta aplicación permitirá probar la seguridad de los sistemas de identificación desarrollados por el centro.

2.2.1- Modelo de dominio.

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los conceptos que existen o los eventos que suceden en el entorno en el que trabaja el sistema. Dicho modelo de dominio se describe mediante diagramas de UML, especialmente mediante diagramas de clases que muestran a los clientes, usuarios, revisores y otros desarrolladores las clases del dominio y cómo se relacionan unas con otras mediante asociaciones (12).

Teniendo en cuenta el problema científico que la aplicación pretende resolver, se procede a crear un modelo de dominio. El mismo permitirá identificar los principales conceptos sociales y tecnológicos del entorno en que se desarrolla el software, así como las relaciones entre ellos.

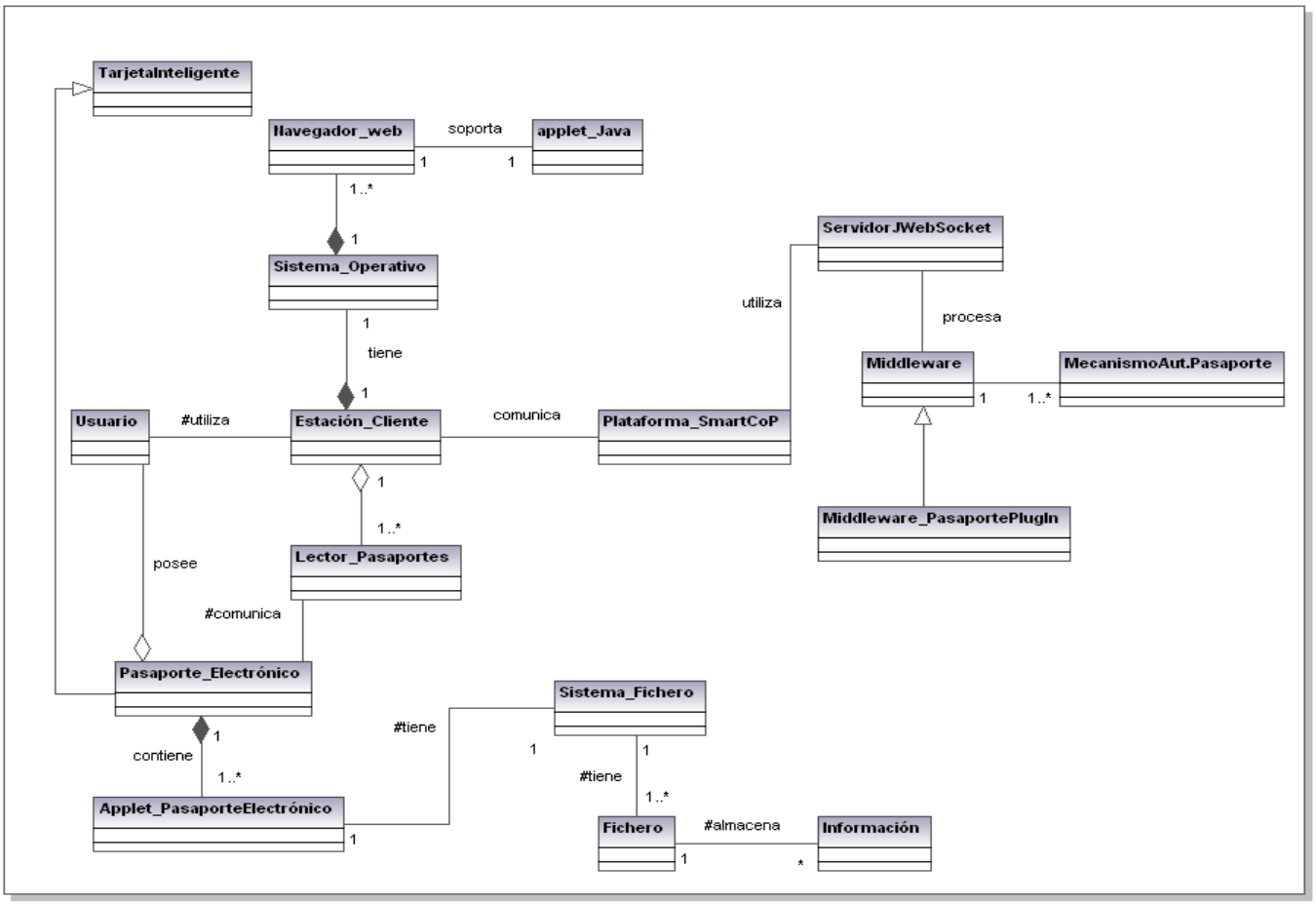


Figura 6: Modelo de dominio del sistema.

2.2.2- Glosario de términos del modelo de dominio.

Usuario: Persona portadora o no del pasaporte electrónico que accede a la aplicación web a través de una computadora.

Applet_Java: Se encarga de gestionar la comunicación entre el navegador web y el pasaporte.

SistemaOperativo: Sistema operativo instalado en la computadora.

NavegadorWeb: 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_Pasaportes: Es un lector compatible con el estándar PC/SC, el cual sirve de mediador para la comunicación entre la estación cliente y los pasaportes.

Tarjeta_Inteligente: Es un dispositivo de plástico similar en tamaño y otros estándares físicos a las tarjetas de crédito, presentan un circuito integrado, el mismo puede ser de sólo memoria o contener un

microprocesador (CPU) con un sistema operativo que le permita una serie de funcionalidades como almacenar información, incriptar información, leer y escribir datos; similar a un ordenador.

Pasaporte_Electrónico: Es una tarjeta inteligente que contiene tecnología JavaCard, contiene datos de identificación de su portador y permite una mayor automatización de los procesos de identificación, logrando una mayor seguridad y efectividad de forma simple.

Applet_PasaporteElectrónico: Es una aplicación instalada dentro del pasaporte electrónico que permite gestionar la información que se almacena en él.

Sistema_Fichero: Sistema de Archivos Elementales (EF) y Archivos Dedicados (DF) definido en el estándar ISO/IEC 7816-4. Incluye además el procesamiento de los principales comandos definidos por este estándar, para la gestión de archivos y objetos de seguridad, y el soporte para mecanismos de autenticación mutua y de canal seguro con el middleware en el terminal.

Fichero: Los ficheros pueden ser elementales (EF) o dedicados (DF), los primeros almacenan información sobre los objetos de seguridad del applet y los segundos pueden guardar otros ficheros u objetos del sistema.

Información: Datos de los objetos de seguridad que se almacenan en el Applet_PasaporteElectrónico.

Plataforma_SmartCoP: Plataforma que provee servicios en línea para tarjetas y pasaportes electrónicos.

ServidorJWebsocket: Contiene los componentes necesarios como comandos, eventos y servicios para la gestión de la información.

Middleware: Software o módulo que se encuentra anexado en el servidor brindando diferentes funcionalidades, se autentica con el pasaporte para iniciar una comunicación segura.

Middleware_PasaportePlugIn: Middleware para leer información de un pasaporte electrónico.

MecanismoAut.Pasaporte: Mecanismo que utiliza el pasaporte para establecer la transmisión segura con el servidor.

2.3- Modelo del sistema.

Con el conocimiento adquirido hasta el momento sobre los conceptos que rodean al objeto de estudio, se definen las características que debe tener el sistema para que cumplan los objetivos planteados al inicio. Para ello se identifican los actores del sistema, requisitos funcionales y no funcionales, modelando los requisitos funcionales en términos de Historias de Usuario (HU).

2.3.1- Propuesta del sistema.

El presente trabajo pretende desarrollar una aplicación para la lectura de pasaportes electrónicos, con el objetivo de obtener y mostrar la información del titular detectando posibles violaciones de acceso en los sistemas de reconocimiento de identidad.

La información de los datos del portador se visualiza en una interfaz web, comenzando con la colocación previa del documento en el lector de pasaporte, seguidamente de la interacción del usuario con la interfaz web solicitando la operación en la página web y por último la visualización de la de la información contenida a través de la interfaz.

2.3.2- Captura de requisitos.

Entre las principales funcionalidades que provee la aplicación podemos citar:

1) Gestionar la comunicación con lectores de pasaportes disponibles.
2) Permitir la autenticación mediante el Control de Acceso Básico (BAC).
3) Enviar y recibir instrucciones del pasaporte electrónico.
4) Gestionar la transmisión de información entre el cliente web y el servidor JWebsocket.
5) Procesar las operaciones del middleware en el servidor.
6) Mostrar al usuario el resultado tras la culminación de la operación del middleware en el servidor.

2.3.2.1- Historias de usuario.

Las HU son utilizadas en la metodología XP para describir los requisitos de una forma menos detallada y desde la perspectiva del cliente. Una historia de usuario debe responder a las preguntas: ¿Quién se beneficia?, ¿Qué se quiere? y ¿Cuál es el beneficio?

La descripción de cada una de las HU está especificada en los anexos 4, 5, 6, 7, 8 y 9. (Ver Anexos)

2.3.2.2- Requisitos no funcionales.

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad y apenas se aplican a características o servicios individuales del sistema.

RNF1.Software.

- ✓ Navegadores web a partir de la versión que soporta el protocolo WebSocket, entre los más populares podemos citar a Google Chrome versión 16.0, Mozilla Firefox versión 10.0, Internet Explorer 9.0.
- ✓ Java Runtime Environment (JRE) para poder ejecutar el applet en el navegador.
- ✓ Controladores que cumplan con el estándar PC/SC.

RNF2.Hardware.

- ✓ Lector de tarjetas incorporado a la PC que cumpla con el estándar PC/SC versión 1.0 o superior.
- ✓ PC Pentium 4 o superior (1GB de Memoria RAM o superior, CPU 3GHZ o superior).

RNF3.Restricciones en el diseño y la implementación.

- ✓ Se empleará el protocolo WebSocket.
- ✓ Se utilizará Netbeans como Entorno Integrado de Desarrollo (IDE).
- ✓ En el lado del servidor el lenguaje de programación a utilizar es Java.
- ✓ En el lado del cliente se hará uso de los lenguajes JavaScript y HTML.

RNF4. Apariencia o interfaz externa.

- ✓ La aplicación deberá tener una interfaz externa amigable, que sea sencilla y fácil de entender por el usuario.

RNF5. Seguridad.

- ✓ La aplicación debe recuperarse en caso de producirse una falla.
- ✓ La información manejada en el servidor estará protegida de ataques externos o internos a través de la seguridad de su sistema operativo.
- ✓ La información contenida en la tarjeta o pasaporte, será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, además, por la seguridad que defina el proveedor de las mismas y por el canal seguro establecido, previo a su comunicación.

En aras de establecer la seguridad presente tanto en las estaciones de trabajo como en el servidor se establecieron una serie de políticas de seguridad con el objetivo de contrarrestar ciertas amenazas entre las que podemos citar: ataques externos e internos, fallos de aplicaciones o hardware, acceso no autorizado, entre otras.

Las políticas de seguridad son:

1. Con respecto a la tecnología:
   - ✓ Se deben realizar salvas de la información (1 total el último día laborable del mes a las 17:00 horas, y el resto de los días una copia incremental a las 13:00 horas), las cuales deben ser almacenadas en lugares seguros.
   - ✓ Haber instalado el antivirus en las máquinas clientes, una vez instalado mantenerlo actualizado de forma manual o mediante especificación de una dirección para su actualización de forma automática.
   - ✓ Cada máquina deberá de tener un backup para evitar daños en la información en caso de una avería eléctrica.
2. Con respecto al sistema operativo:
   - ✓ Todas las máquinas deben tener el firewall activado.
   - ✓ Especificar contraseñas en el SETUP.
3. Con respecto al personal

- ✓ El personal hará uso sólo de los bienes informáticos correspondientes.
- ✓ El acceso a los locales está restringido, solo a las personas autorizadas.
- ✓ El personal que trabaje con las tecnologías informáticas y de comunicaciones responderá por la protección de la información que se le confíe.
- ✓ La contraseña empleada por los usuarios para acceder a los servicios brindados tendrá una longitud de 7 caracteres como mínimo, y debe cumplir con varios requerimientos de complejidad como empleo de números, caracteres especiales, letras mayúsculas y minúsculas.
- ✓ De ocurrir algún fallo de hardware informar al responsable de seguridad y no asumir funciones que no le corresponden al usuario.

4. Con respecto a conexiones externas:
   - ✓ Asegurar la definición e implementación de procedimientos pertinentes para el control de las actividades de usuarios externos del organismo a fin de garantizar la adecuada protección de los bienes de información de la organización.
   - ✓ Debe asegurarse que la totalidad del tráfico entrante y saliente de la red interna, sea filtrado y controlado por un firewall prohibiendo el pasaje de todo el tráfico que no se encuentre expresamente autorizado.
   - ✓ El esquema de direcciones de la red interna no debe ser visible ante las conexiones externas.

2.4- Metáfora.

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, así como guiar la arquitectura y estructura del mismo. Es muy importante que el cliente y el grupo de desarrolladores estén de acuerdo y compartan esta metáfora para mantener coherencia en los mismos términos y de una forma más precisa y de dominio de todos tributar a lo que se va a implementar.

La aplicación para la lectura de pasaportes electrónicos funcionará con lectores de pasaportes, mientras los mismos hayan sido diseñados siguiendo los estándares que redactó la OACI para la normalización de esta tecnología. Esta a su vez será integrada a la plataforma de servicios en línea utilizando tarjetas inteligentes. Para que un determinado evento se comunique con el pasaporte insertado en un lector a la estación cliente, el sistema provee un componente instalado en el navegador del cliente con el primer acceso del usuario a la aplicación. Este control posibilitará realizar operaciones básicas como reconocer el lector conectado a la máquina cliente y establecer una conexión de canal seguro entre el pasaporte y el servidor, notificar los eventos relacionados con el pasaporte y lo más importante transmitir comandos

APDU[11] y recibir las respuestas del pasaporte. Las respuestas emitidas por el pasaporte serán enviadas al componente controlador que se encuentra del lado del servidor y que se encargará de invocar al plugin correspondiente. A la aplicación se le podrá integrar múltiples extensiones en dependencia de los servicios que el desarrollador de la aplicación quiera brindar en su interfaz web; lo más importante es que cada una tiene que implementar una interfaz de programación establecida por la aplicación para garantizar la comunicación con el pasaporte.

2.5- Diseño.

2.5.1- Arquitectura.

La propuesta de la solución para la lectura de pasaportes electrónicos basa su implementación en la arquitectura cliente-servidor. Teniendo en cuenta las características de la aplicación, cuyo principal papel es mediar entre la aplicación a desarrollar y el lector de pasaporte que contiene dicho documento establecido en la estación cliente, se identifica una capa cliente que contiene los componentes para comunicarse con el lector de tarjetas y con los plugins en el servidor.
En el lado del cliente se encuentran los siguientes componentes:

- Visual_LP.
- SmartCoP.
- Jwebsocket_Client.

El cliente se comunica con el pasaporte previamente colocado en el lector a través del estándar PC/SC y a su vez con el servidor mediante el protocolo Websocket.

En segundo lugar se tiene la capa servidor, donde el resto de los componentes del sistema se relacionan entre sí para resolver las solicitudes que vienen desde el cliente, además de gestionar los plugins que se encuentran asociados a la aplicación y las funcionalidades contenidas en ellos accediendo a sus librerías. El servidor contiene además un conjunto de configuraciones especificadas en varios xml denominado Configuration_Plugins y un servidor jwebsocket que incluye los componentes que a continuación se muestran:

- JwebSocket_Server.
- SmartCoPServer.
- MiddlewareLP.

[11] Unidad de datos de protocolo de aplicación (por sus siglas en inglés, **Application Protocol Data Unit**) es la unidad de comunicación entre un lector de tarjetas inteligentes y una tarjeta inteligente.

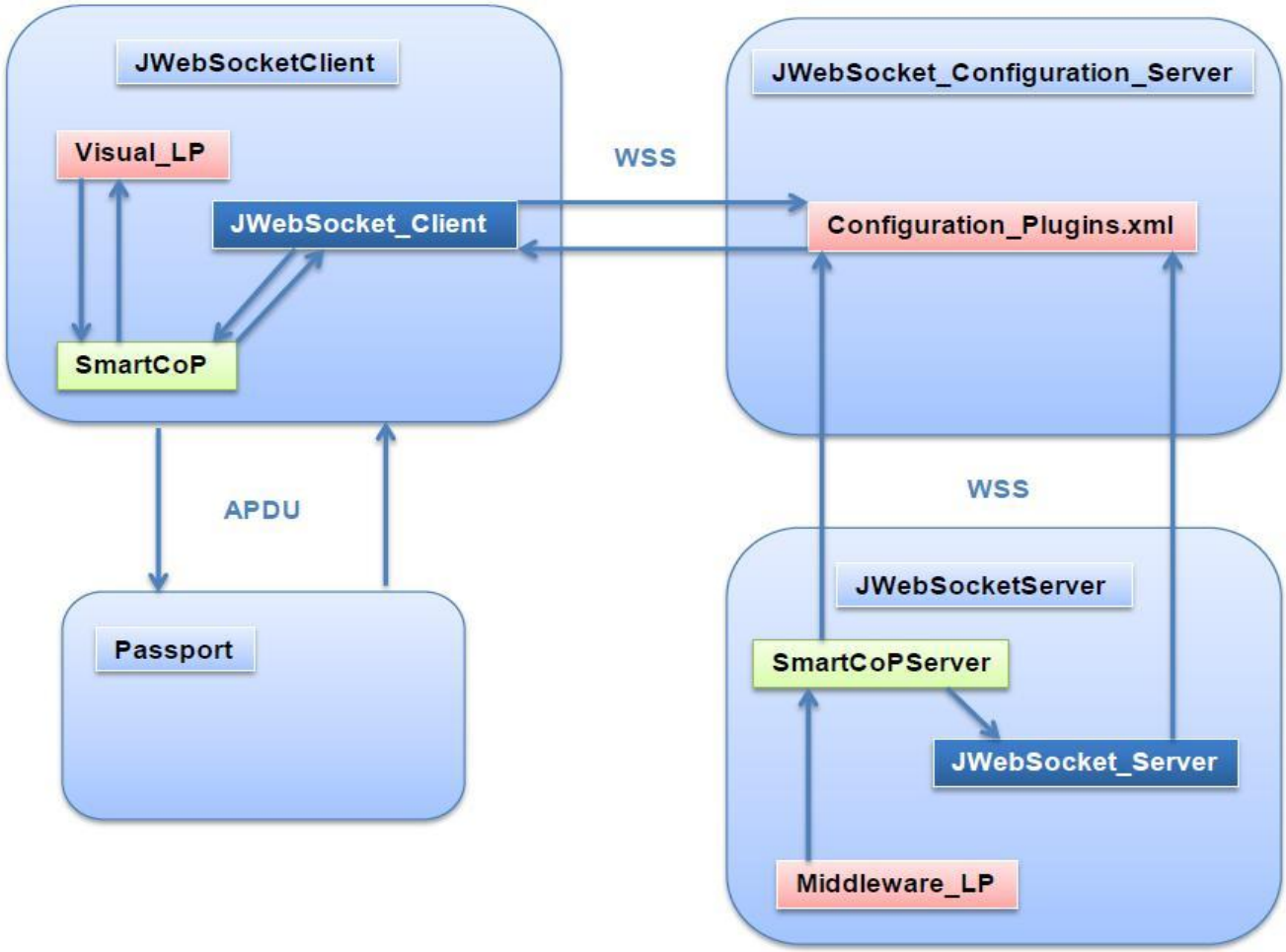


Figura 7: Arquitectura del sistema.

El cliente posee los siguientes componentes:

- Interfaz.
- Vistas.
- Controladora.

El componente interfaz es el encargado de vincular los restantes componentes. El componente Vistas es el que posee las interfaces que se muestran al usuario y el componente Controladora posee las funcionalidades para la gestión de las transacciones efectuadas por el usuario y la validación de los datos introducidos.

El cliente utiliza las librerías de ExtJS 4.1 para la visualización de la página web, y por último complementos empleado para establecer los vínculos entre el cliente y el servidor.

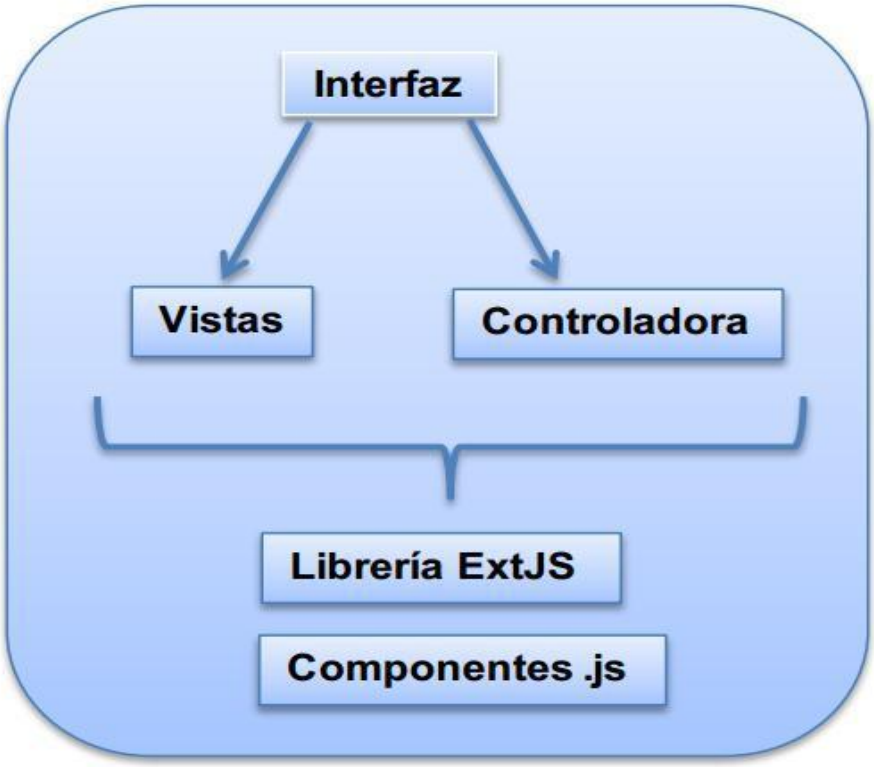


Figura 8: Arquitectura del cliente.

El servidor posee los siguientes componentes:

- PasaportePlugIn.
- Eventos.
- Comandos.
- Servicios.

El componente principal del servidor es el primero antes mencionado, el mismo gestiona un grupo de eventos, utiliza servicios y comandos para la gestión de la información y además cada evento puede tener implícito para su realización uno o varios comandos.

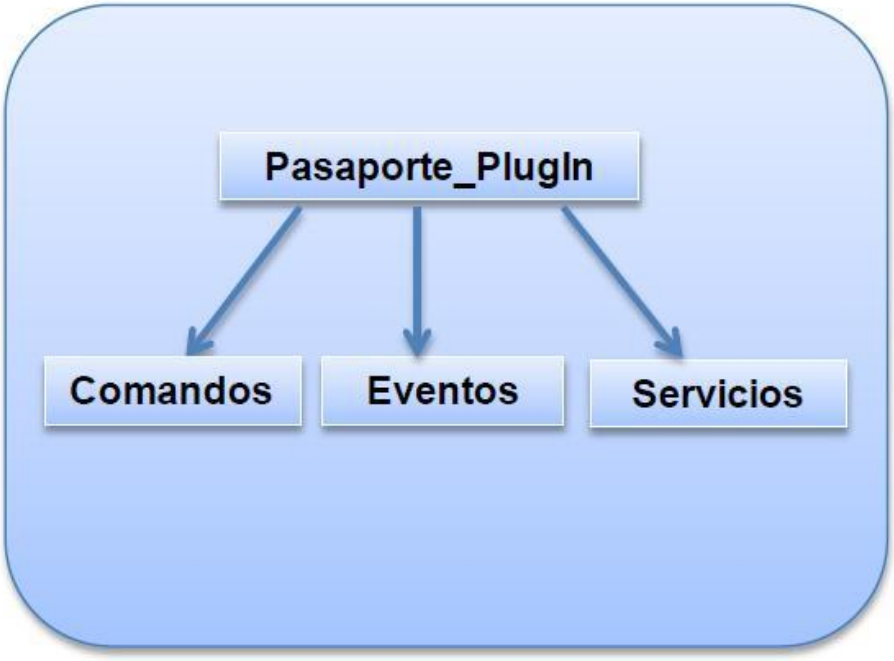


Figura 9: Arquitectura del servidor.

2.5.2- Patrones arquitectónicos.

A diferencia de los patrones de diseño que especifican la estructura y comportamiento de una sociedad de clases, los patrones arquitectónicos especifican la estructura y comportamiento de un sistema completo. Estos a su vez, describen los principios fundamentales de la arquitectura de un sistema de software, identifican los subsistemas, definen sus responsabilidades y establecen las reglas y guías para organizar las relaciones entre ellos. En el diseño de la solución se aplicó el patrón arquitectónico Modelo-Vista-Controlador (MVC, por sus siglas en inglés).El estilo de llamada y retorno presente en este tipo de patrón, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es la lógica del negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista. En la solución se identifican además de estos componentes, un controlador en el cliente que gestiona los datos de la interfaz que se envían al servidor o vienen de este.

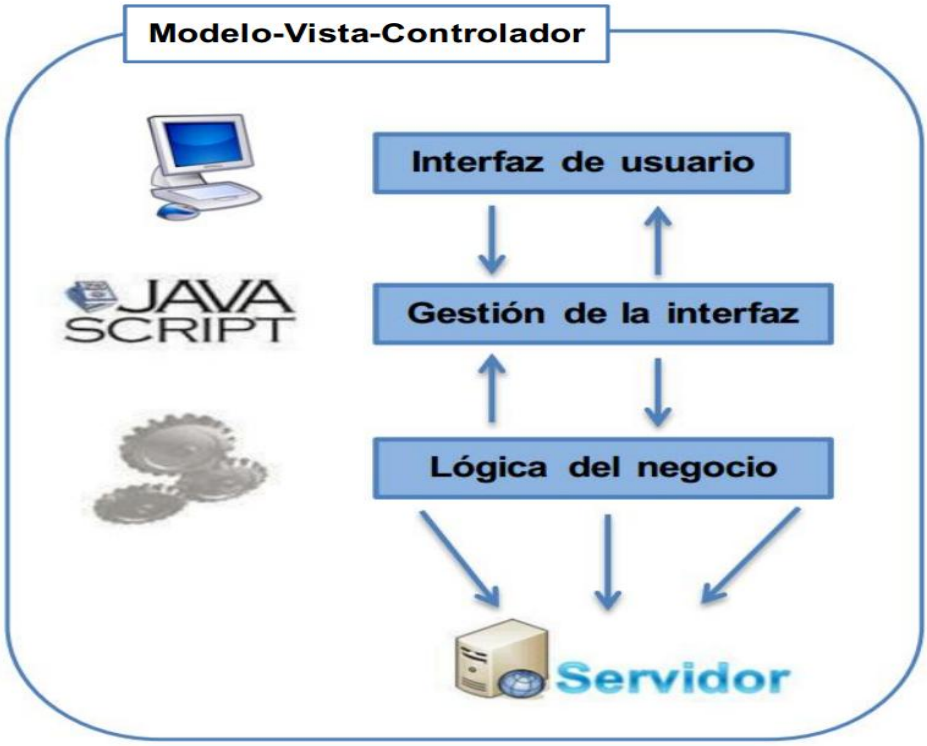


Figura 10: Patrón MVC aplicado a la solución.

2.5.3- Patrones de diseño.

En el diseño de software orientado a objetos los patrones GRASP[12] describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades, su uso tiene una gran importancia debido a que dan solución a muchos de los problemas que se puedan presentar en la programación.

Para el diseño del aplicación se tuvieron en cuenta los patrones GRASP: Experto, Bajo acoplamiento y Alta cohesión.

El patrón Experto utilizado en el diseño de la aplicación define como asignar de forma adecuada las responsabilidades en un modelo de clases. Este indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer en la clase que conoce toda la información necesaria para crearlo, dicho patrón se pone de manifiesto en la clase controladora *PasaportePlugIn.java*. De este modo se obtendrá un diseño de mayor cohesión y así la información se mantiene encapsulada, disminuyendo el acoplamiento.

[12] Patrones generales de software para asignación de responsabilidades, del acrónimo GRASP (***General Responsibility Assignment Software Patterns***)

El patrón Bajo acoplamiento utilizado para el diseño de la aplicación mantiene lo menos posible ligada las clases que forman la solución, evitando así que una modificación en alguna de ellas suponga una gran repercusión en las restantes, potenciando así la reutilización, y disminuyendo la dependencia entre las clases, dicho patrón se pone de manifiesto en la clase *CBEFFDataGroup.java*. De este modo el patrón propone el diseño de clases más independientes, lo que reduce el impacto del cambio y facilita la reutilización en otros sistemas.

El patrón Alta cohesión utilizado en el diseño de la aplicación plantea que la información que almacena una clase debe de ser coherente y debe estar relacionada con la clase, este patrón se pone de manifiesto en la clase MRZInfo.java.

Para la especificación de ejemplos relacionados con cada patrón de diseño. (Ver anexo 10)

2.6-Tareas de ingeniería.

Al inicio de la primera iteración se debe tener claro qué es lo que se debe codificar. Todo el trabajo de la iteración es expresado en tareas de programación, las cuales se realizan para especificar las acciones llevadas a cabo por los programadores en cada historia de usuario, aunque cada historia de usuario describe a un alto nivel lo que se necesita implementar, no brindan los detalles suficientes para poder hacer una codificación correcta.

Según el Plan de iteraciones las historias de usuario se agruparon en dos iteraciones. A continuación se muestran las tareas de ingeniería derivadas de cada historia de usuario. (Ver Anexo 11 para las especificaciones de cada una)

Iteración	Historia de Usuario	Tarea
1	Gestionar la comunicación con lectores de pasaportes disponibles.	- Establecer conexión con el pasaporte. - Cerrar conexión con el pasaporte.
	Permitir la autenticación mediante el Control de Acceso Básico (BAC).	- Obtener los datos del código de barra de la zona de lectura mecánica.
	Enviar y recibir instrucciones del pasaporte electrónico.	- Enviar peticiones al pasaporte. - Recibir respuesta del pasaporte.
2	Gestionar la transmisión de información entre el cliente web y el servidor JWebSocket.	-Enviar respuesta que genera el middleware al cliente. -Enviar respuesta recibida del pasaporte al servidor.

	Procesar las operaciones del middleware en el servidor.	-Invocar funcionalidad de un middleware específico.
	Mostrar al usuario el resultado tras la culminación de la operación del middleware en el servidor.	- Enviar respuesta al cliente web.

Tabla 2. Distribución de las tareas de ingeniería por iteraciones.

2.7- Diseño de la solución.

Una vez desglosadas las historias de usuario en tareas, se comienza a pensar en la forma de llevarlas al código, para ello se crean sesiones con el equipo de trabajo donde se presentarán las tarjetas CRC, que darán la idea de la cantidad de clases a implementar y las responsabilidades que estas tendrán. XP propone la puesta en práctica de ciertos principios a la hora de realizar estas sesiones de trabajo, para garantizar la agilidad en el proceso de desarrollo.

2.7.1- Tarjetas CRC.

La técnica de las tarjetas CRC, se puede usar para guiar el sistema a través de análisis encaminados por la responsabilidad. Las clases se examinan, se filtran y se refinan en base a sus responsabilidades con respecto al sistema, y las clases con las que necesitan colaborar para completar sus responsabilidades. El diseño de la solución se realizó por iteraciones donde se confeccionaron las tarjetas CRC. La aplicación para la lectura de pasaportes electrónicos está formada por un conjunto de clases, entre las que podemos citar:

- PassportPlugin.
- PassportApduService.
- CBEFFDataGroup.
- MRZInfo.
- FaceInfo.

Para una mejor comprensión en la descripción de cada una de estas clases. (Ver Anexo 12)

2.8- Plan de entrega.

Una vez que se concluye la tarea por parte del cliente de elaborar las distintas historias de usuario, se comienza con la creación del Plan de entrega, para estimar el tiempo de desarrollo de las mismas. Este artefacto será el resultado de una reunión en la cual participan los desarrolladores y clientes, estos últimos seleccionarán según sus necesidades las historias de usuario con mayor prioridad y los desarrolladores proponen el período de tiempo que puede tardar la implementación de las mismas; quedando listo el cronograma de entregas en el cual se definen las fechas en que serán liberadas las versiones funcionales de producto. (Ver Anexo 13)

2.9- Estimación de tiempo.

Los programadores se encargan de estimar el tiempo que cada historia de usuario puede tomarles para el desarrollo de la misma. El valor de tiempo se expresa en semanas y en un principio no es totalmente exacto, pero en el transcurso de las iteraciones y el desarrollo del producto se irá acercando a la realidad. (Ver Anexo 14)

2.10- Plan de iteraciones.

Correspondiente a la segunda fase y como parte del ciclo de vida del 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. Por tanto, en la presente solución se han definido 6 historias de usuario, divididas en 2 iteraciones, de acuerdo a los intereses del cliente, para duración total del proyecto de 18 semanas. (Ver Anexo 15)

2.11- Conclusiones parciales

- Se permitió determinar las bases del por qué la elección de un modelo de dominio, el cual agrupa todos los conceptos asociados a la solución, así como las relaciones entre estos conceptos.
- El análisis de los sistemas existentes y la profundización del objeto de estudio permitió definir los requisitos que debe cumplir el sistema.
- El desglose de las HU en tareas de la ingeniería constituye una buena práctica que muestra a los programadores las funcionalidades específicas a implementar.
- Mediante la definición de las HU que caracterizan a la solución y los requerimientos no funcionales que brindan las cualidades que deben tener en cuenta para desarrollar la solución adecuada, se obtuvo una visión para definir la arquitectura del sistema a desarrollar.
- Con el uso de las tarjetas CRC se permitió ver las clases no solo como almacenadora de los datos, sino también, el comportamiento de estas, sus colaboradores y sus responsabilidades.

Capítulo 3: Implementación, Prueba y Validación del Sistema.

3.1- Introducción.

Seguido de la fase de exploración y planificación, XP define las fases Iteraciones a primera liberación y producción. En la planificación se definieron las iteraciones y en cada iteración se diseñan, prueban y codifican cada una de las historias de usuario.

En el siguiente capítulo se muestra todo lo relacionado a cada iteración mediante la descripción de elementos en el proceso de implementación y a través de diferentes diagramas que ofrecen una panorámica del funcionamiento de la aplicación, así como ocurren los principales flujos de procesos. Se visualizan las interfaces de usuario y se da una explicación de cada una de ellas para un mayor entendimiento acerca de la estructura de la aplicación. Se realizan las pruebas donde se valida el funcionamiento de los requisitos funcionales.

3.2- Comandos utilizados.

El middleware para la lectura de pasaportes electrónicos soporta los comandos que se muestran a continuación:

Comando	Descripción
SELECT FILE	Comando que se utiliza para seleccionar un archivo especificado ya sea un DF o un EF.
GET CHALLENGE	Comando que genera un número aleatorio para ser usado por los comandos MUTUAL AUTHENTICATE, este comando inicia el proceso de autenticación mutua.
MUTUAL AUTHENTICATE	Comando definido por el estándar ISO/IEC 7816-4, que permite a la tarjeta y el terminal autenticarse el uno con el otro, verificando la presencia de las dos llaves secretas K.ENC y K.MAC, este comando es válido sólo en la fase de aplicación, en la fase de personalización existe el comando EXTERNAL AUTHENTICATE para el mismo propósito.
CREATE FILE	Comando que se utiliza para crear un EF o un DF.
READ BINARY	Comando que se utiliza para leer la información contenida en un EF.

Tabla 3: Comandos utilizados.

3.2.1 Comando SELECT FILE.

El comando SELECT FILE selecciona un archivo DF o EF.

El formato del comando es el siguiente:

CLA	INS	P1	P2	Lc	Data	Le
CLA	A4h	RefCtrl	RefCtrl	Lc	Data	Le

Tabla 4: Descripción del comando SELECT FILE.

Donde:

CLA: 00h: Transmite normalmente.

: 04h: Transmite con mensajería segura.

P1: Es un parámetro de control de referencia. La configuración de bits es la siguiente:

P1	b8	b7	b6	b5	b4	b3	b2	b1
RFU.	0	0	0	0	x	x	x	x
Seleccionar por la ruta desde el MF.					1	0	0	0
Seleccionar EF por el ID del archivo bajo el DF.					0	0	1	0
Seleccionar MF, DF o EF por el ID del archivo.					0	0	0	0

Tabla 5: Ejemplo de la configuración de bits para P1.

P2: Es un parámetro de control de referencia que especifica el formato de la respuesta o que no existe respuesta. La configuración de bits es la siguiente:

P2	b8	b7	b6	b5	b4	b3	b2	b1
RFU.	0	0	0	0				
Opción FCI.					x	x		
Devolver plantilla FCI.					0	0		
Devolver plantilla FCP.					0	1		
Sin respuesta.					1	1		
Primera o única ocurrencia del archivo.							0	0

Tabla 6: Ejemplo de la configuración de bits para P2.

Lc: Es la longitud del campo Data como se indica a continuación:

- 02h: Para la selección del archivo usando el ID.
- 02h o 04h: Para la selección por camino.

Data: contiene la referencia del archivo y es uno de los siguientes:

- ID del archivo: Identificador del archivo a ser seleccionado.
- Camino: Es la concatenación de los ID del MF o DF actual hasta el ID del archivo a ser seleccionado, no incluyendo el ID del MF o DF actual.

Le: Es la longitud de los datos que deben ser devueltos, y está determinado por la configuración de P2.

Respuesta:

El pasaporte retorna los datos solicitados por el parámetro de control de referencia P2, seguido por los códigos de estado SW1 y SW2. Los datos posibles de respuesta son: la plantilla FCI, la plantilla FCP o no existe (P2 = 0Ch). La plantilla FCP contiene los parámetros de control del archivo que fueron

especificados cuando fue creado el archivo. La información FCI retornada por el comando (P2 = 00h) es mostrada a continuación:

Desplazamiento	Datos	Descripción
0	6Fh	Etiqueta de la plantilla FCI.
1	L	Longitud de la data FCI.
2	83h	Etiqueta del ID del archivo.
3	02h	Longitud del ID del archivo.
4-5	ID del archivo	Valor del ID del archivo.
6	8Ch	Etiqueta de atributos de seguridad.
7	L	Longitud de atributos de seguridad.
8	AM	Byte Modo de Acceso.
9-(8+X)	SC	Bytes Condición de Seguridad (X).
9+X	84h	Etiqueta del nombre del DF.
10+X	L	Longitud del nombre del DF.
11+X-	Nombre del DF	Valor del nombre del DF (hasta 16 bytes).

Tabla 7: FCI para DFs.

Desplazamiento	Datos	Descripción
0	6Fh	Etiqueta de la plantilla FCI.
1	L	Longitud de la data FCI.
2	81h	Etiqueta de tamaño de archivo.
3	02h	Longitud de tamaño de archivo.
4-5	Tamaño del archivo	Valor de tamaño de archivo.
6	82h	Etiqueta de FDB.
7	01h	Longitud de FDB.
8	FDB	Valor de FDB.
9	83h	Etiqueta del ID del archivo.
10	02h	Longitud del ID del archivo.
11-12	ID del archivo	Valor del ID del archivo.
13	8Ah	Etiqueta del byte de estado del ciclo de vida del archivo.
14	01h	Longitud del byte de estado del ciclo de vida del archivo.
15	Var	Valor del byte de estado del ciclo de vida del archivo.

16	8Ch	Etiqueta de atributos de seguridad.
17	L	Longitud de atributos de seguridad.
18	AM	Byte Modo de Acceso.
19-(18+X)	SC	Bytes Condición de Seguridad (X).

Tabla 8: FCI para EFs.

Los posibles valores para SW1 y SW2 son los siguientes:

SW1	SW2	Descripción
90h	00h	Comando procesado sin error. No retornados datos FCI.
67h	00h	Longitud incorrecta de Lc.
69h	82h	Estado de seguridad no satisfecho, error durante el envío de mensaje seguro.
6Ah	82h	Archivo no encontrado.
6Ah	86h	Incorrectos parámetros P1 y P2.

Tabla 9: Posibles valores para SW1 y SW2 SELECT FILE.

3.2.2 Comando GET CHALLENGE.

Este comando genera una respuesta que es usada por el comando MUTUAL AUTHENTICATE (para la autenticación mutua simétrica). El número aleatorio generado por el comando es válido solamente para el próximo comando. Este comando es ejecutado sin mensajería segura.

El formato del comando es el siguiente:

CLA	INS	P1	P2	Le
80h	84h	00h	00h	08h

Tabla 10: Descripción del comando GET CHALLENGE.

Respuesta:

La respuesta es un reto de ocho bytes RND.ICC.

Después de la respuesta, el pasaporte retorna los códigos de estado SW1 y SW2. Los posibles valores para SW1 y SW2 son los siguientes:

SW1	SW2	Descripción
90h	00h	Comando procesado sin error.
67h	00h	Longitud incorrecta.
69h	82h	Condiciones de seguridad no satisfechas.
69h	86h	Comando solo soportado en fase de aplicación.

Tabla 11: Posibles valores para SW1 y SW2 GET CHALLEGE.

3.2.3 Comando MUTUAL AUTHENTICATE.

Este comando permite al pasaporte y el terminal autenticarse el uno con el otro, verificando la presencia de las dos llaves secretas K.ENC y K.MAC. Para adicionar seguridad los valores de K.ENC y K.MAC almacenados en el pasaporte pueden ser diversificados. Es necesario que haya sido enviado un comando GET CHALLENGE antes de este comando con el fin de obtener el RND.ICC.

El formato del comando es el siguiente:

CLA	INS	P1	P2	Lc	Data	Le
80h	82h	00h	00h	00h o 48h	Data	48h

Tabla 12: Descripción del comando MUTUAL AUTHENTICATE.

Donde:

Lc: Es la longitud de los datos (48h = 72 bytes).

Data: Es S'|| MAC

Donde:

S = RND.IFD || SN.IFD || RND.ICC || SN.ICC || TRnd.

RND.IFD: Es un reto de 8 bytes generados por la terminal.

SN.IFD: Es el número de serie del terminal.

RND.ICC: Es un reto de 8 bytes generados por el pasaporte.

SN.ICC: Es el número de serie del chip.

TRnd: Es un número aleatorio de 32 bytes generado por el terminal.

Le: Es la longitud de la respuesta (siempre 72 bytes (48h)).

Respuesta:

Donde Data es SS'|| MAC.

Después de la respuesta, el pasaporte retorna los códigos de estado SW1 y SW2. Los posibles valores para SW1 y SW2 son los siguientes:

SW1	SW2	Descripción
90h	00h	Comando procesado sin error.
67h	00h	Incorrecta longitud.
69h	85h	Condiciones del comando no satisfechas.
6Ah	80h	Datos no encontrados.
6Ah	86h	Incorrectos parámetros P1 y P2.

Tabla 13: Posibles valores para SW1 y SW2 MUTUAL AUTHENTICATE.

3.2.4 Comando READ BINARY.

Este comando lee una parte o todo de un EF. Si el EF que se quiere leer contiene datos protegidos por un atributo de seguridad, entonces este atributo debe ser introducido para poder ejecutar el comando. El EF debe haber sido seleccionado previamente usando el comando SELECT FILE. Los datos son leídos directamente desde el EF actual. En este caso el offset es especificado sobre P1 y P2.

El formato del comando es el siguiente:

CLA	INS	P1	P2	Le
CLA	B0h	P1	P2	Le

Tabla 14: Descripción del comando MUTUAL AUTHENTICATE.

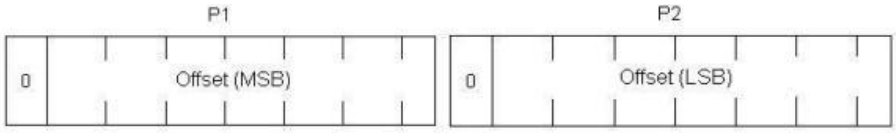
Donde:

CLA: 00h: Transmite normalmente.

: 04h: Transmite con mensajería segura.

P1, P2: Son los bytes de los parámetros.

Para leer del EF que se encuentra seleccionado, P1 y P2 especifican el offset, en bytes, desde el comienzo del archivo al primer byte a ser leído. P1 y P2 tienen el siguiente formato:



Le: Indica el número de bytes a leer. Si está vacío el campo o es cero, entonces el comando lee hasta el fin del archivo, un máximo de 255 bytes cuando no se usa mensajería segura.

Respuesta:

En el campo Data vienen los datos leídos desde el pasaporte, seguidos por los códigos de estado SW1 y SW2. Los posibles valores para SW1 y SW2 son los siguientes:

SW1	SW2	Descripción
90h	00h	Comando procesado sin error.
67h	00h	Longitud incorrecta de Lc (si usa mensajería segura).
69h	82h	Condición de seguridad no satisfecha, por ejemplo: • Los atributos de seguridad no han sido satisfechos. • Error durante la mensajería segura.
69h	86h	Comando no permitido (No EF actual).
6Ah	80h	Incorrectos Lpv, Lcg o Le cuando usa mensajería segura.
6Ah	81h	Función no soportada: SM en modo ENC/MAC especificado la respuesta.

6Ah	86h	Incorrectos P1 y P2 (se chequea que el offset es dentro del EF).

Tabla 15: Posibles valores para SW1 y SW2 READ BINARY.

3.3- Estándar de Codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software (23).

Estilos	Descripción
Pascal	La primera letra en el identificador y la primera letra de cada subsiguiente palabra concatenada se capitalizan. Se utiliza en caso de identificadores con tres o más caracteres.
Camello	La primera letra en el identificador está en minúscula y la primera letra de cada subsiguiente palabra concatenada en mayúscula.
Mayúsculas	Todas las letras en el identificador se capitalizan. Esta convención se utilizará sólo para los identificadores que constan de dos o menos letras (24).

Tabla 16. Estándares de codificación.

Sensibilidad a mayúsculas.

Para evitar confusiones y garantizar la interoperabilidad entre lenguajes, se siguieron las siguientes reglas sobre el uso de mayúsculas y minúsculas:

- No utilizar nombres o identificadores que requieran ser *case sensitive*[13] ya que los componentes deben ser completamente funcionales tanto para los lenguajes case-insensitive como para los case-sensitive.
- No crear dos *namespaces*[14] que se diferencien solo en el uso de las mayúsculas.
- No crear clases con propiedades o métodos que se diferencien únicamente en el uso de las mayúsculas (24).

3.3.1- Reglas de codificación.

- El código fuente debe ser escrito en idioma inglés.

[13] Los lenguajes case-insentitive no pueden distinguir entre dos nombres en el mismo contexto que se diferencien solo en el uso de las mayúsculas.

[14] Del inglés Espacio de nombres.

- Se deben evitar las líneas de más de 80 caracteres, ya que no son bien manejadas por muchas herramientas.
- Cada línea debe contener cuando más una sentencia.
- Cada funcionalidad debe tener comentario de su funcionamiento.

Tipos de identificadores	Reglas de nombre	Ejemplos
Clases o Interfaces.	Los nombres de las clases o interfaces debe tener la primera letra de cada palabra en mayúscula.	PasaportePlugIn, PassportApduService.
Métodos.	Los nombres de los métodos deben reflejar la acción a realizar y siempre comenzando con letra mayúscula, en caso de ser compuesto ambas palabras en Mayúscula.	createSelectFileAPDU, createReadBinaryAPDU.
Variables.	Todas las variables empezarán con minúscula y la primera letra de las siguientes palabras en minúscula.	rawstream, aresponse.
Constantes.	Cada caracter que pertenezca al nombre de la constante se escribirá en mayúscula y en caso de ser un nombre compuesto, cada palabra se separará por un guión bajo "_".	OFFSET_CLA, OFFSET_INS.

Tabla 17. Reglas de codificación.

3.4- Diagrama de Componentes.

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre los mismos. Los componentes físicos incluyen archivos, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema (25).

El diagrama de componentes de la solución para la lectura de pasaportes electrónicos refleja todos aquellos componentes que conforman la solución propuesta, correspondiéndose los mismos con la arquitectura definida para su implementación.

PruebaClienteWebePassport: Página web utilizada por el usuario final, mostrándole una interfaz amigable.

ExtJS Library: Librería que permite crear rápidamente páginas web haciendo la interfaz de usuario más familiar, es compatible con varios navegadores y se utilizan características como comunicación asincrónica, serialización XML y servicios de aplicación.

PruebaePassportOnLineApplicationWeb.java: Contiene el código de la página del lado del servidor donde se gestionan los eventos de la misma.

ePassportOnLineApplication.MiddlewareMediator: Componente mediador entre el middleware y el cliente web. Sus principales funciones son invocar las operaciones sobre una solicitud del usuario y dirigir las transmisiones de APDU hacia y desde el cliente.

Configuration-JWebsocket.xml: Paquete que contiene las clases que se encargan de la comunicación entre el servidor y el pasaporte, gestiona eventos como la llegada de APDU desde el servidor, la emisión de una respuesta del pasaporte y la información a mostrar al usuario.

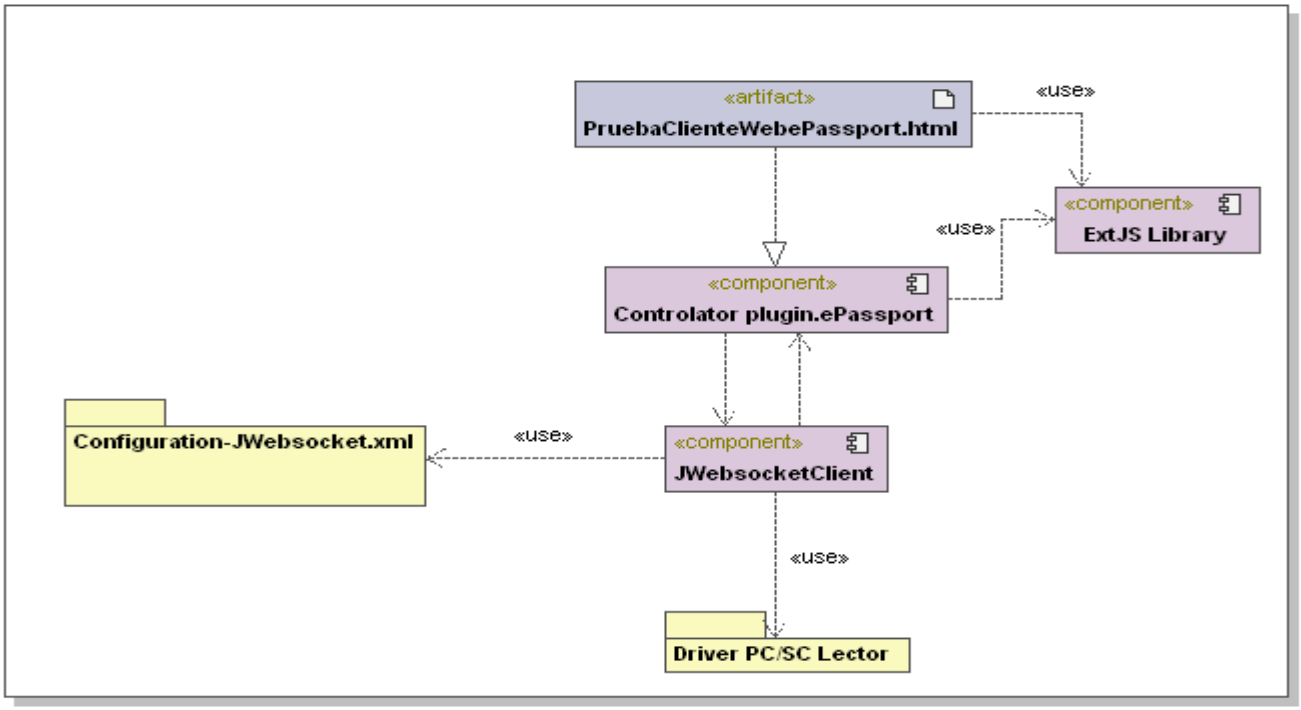


Figura 11: Diagrama de componentes del cliente.

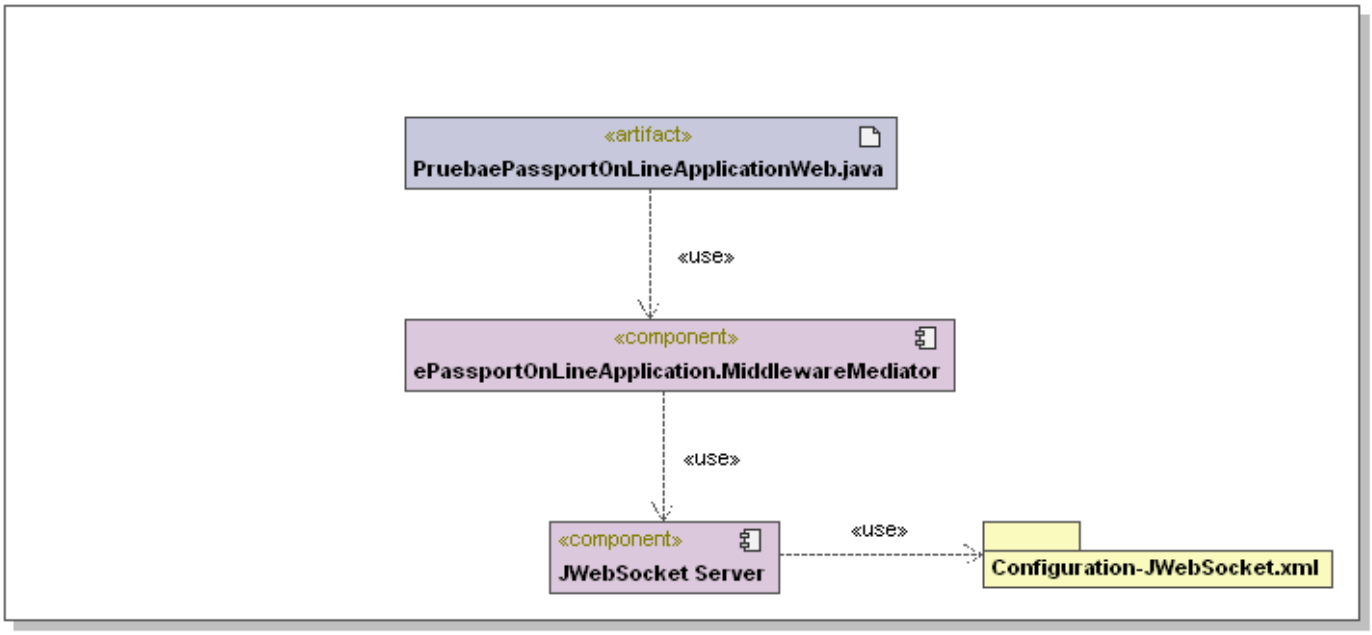


Figura 12: Diagrama de componentes del servidor.

3.5- Diagrama de Despliegue.

El diagrama de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos, componentes y asociaciones. Un artefacto puede ser algo como un archivo, un programa, una biblioteca, o una base de datos construida o modificada en un proyecto. Estos artefactos implementan colecciones de componentes. Los nodos internos indican ambientes, un concepto más amplio que el hardware propiamente dicho, ya que un ambiente puede incluir al lenguaje de programación, a un sistema operativo, un ordenador o un clúster de terminales (26).

La aplicación web para la lectura de pasaportes electrónicos se utiliza en todos los proyectos que cuenten con lectores de pasaportes que cumplan los estándares de la OACI. El sistema debe funcionar en las estaciones de trabajo de los clientes, encargados de realizar la verificación correspondiente en la gestión de los pasaportes. Estas estaciones de trabajo estarán conectadas a un servidor de aplicaciones. Las estaciones de trabajo dedicadas a la lectura de la información contenida en los pasaportes del titular cuentan con un lector de pasaportes conectado que cumplan con el estándar PC/SC. La conexión entre las estaciones de trabajo y el servidor de aplicaciones se realizarán mediante el protocolo WebSocket (WSS).A continuación se muestra el diagrama de despliegue de la solución propuesta.

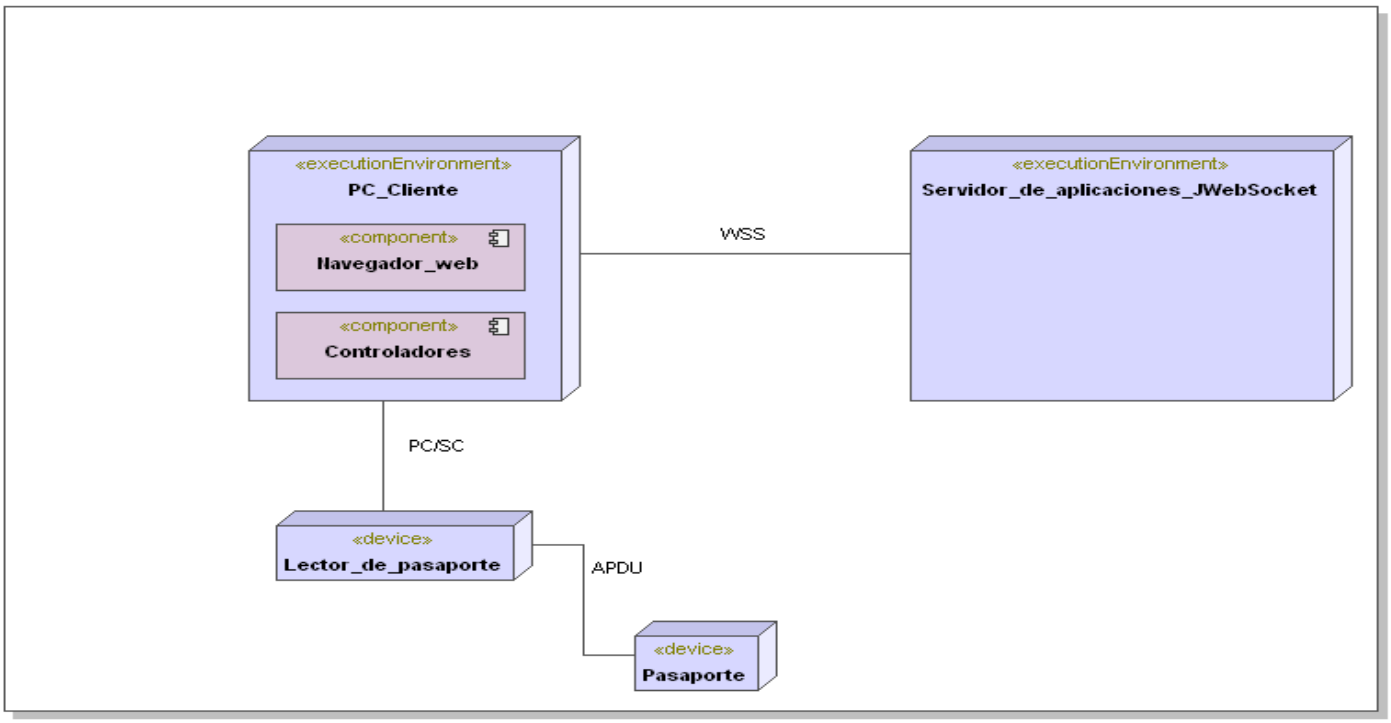


Figura 13: Diagrama de despliegue de la aplicación para la lectura de pasaportes electrónicos.

3.6- Interfaz de Usuario.

El diseño de interfaces de usuario es una tarea que ha adquirido relevancia en el desarrollo de un sistema ya que la calidad de la interfaz de usuario puede ser uno de los motivos que conduzcan a su aceptación. La interfaz de usuario de un programa es un conjunto de elementos que presentan información al usuario y le permiten interactuar con la información y con la computadora. Es de vital importancia que las interfaces sean amigables y sencillas de comprender, pues de ello depende la aceptación del sistema por parte de los usuarios, la claridad que estos tengan de los procesos y el éxito final que el sistema tenga en la empresa que sea desplegado (27).

En la figura 14 se muestra la interfaz principal de la aplicación para la lectura de pasaportes electrónicos, creada con el objetivo de intercambiar información con los usuarios de la forma más intuitiva y sencilla posible, tratando de hacer la solución desarrollada agradable a la vista de los mismos.

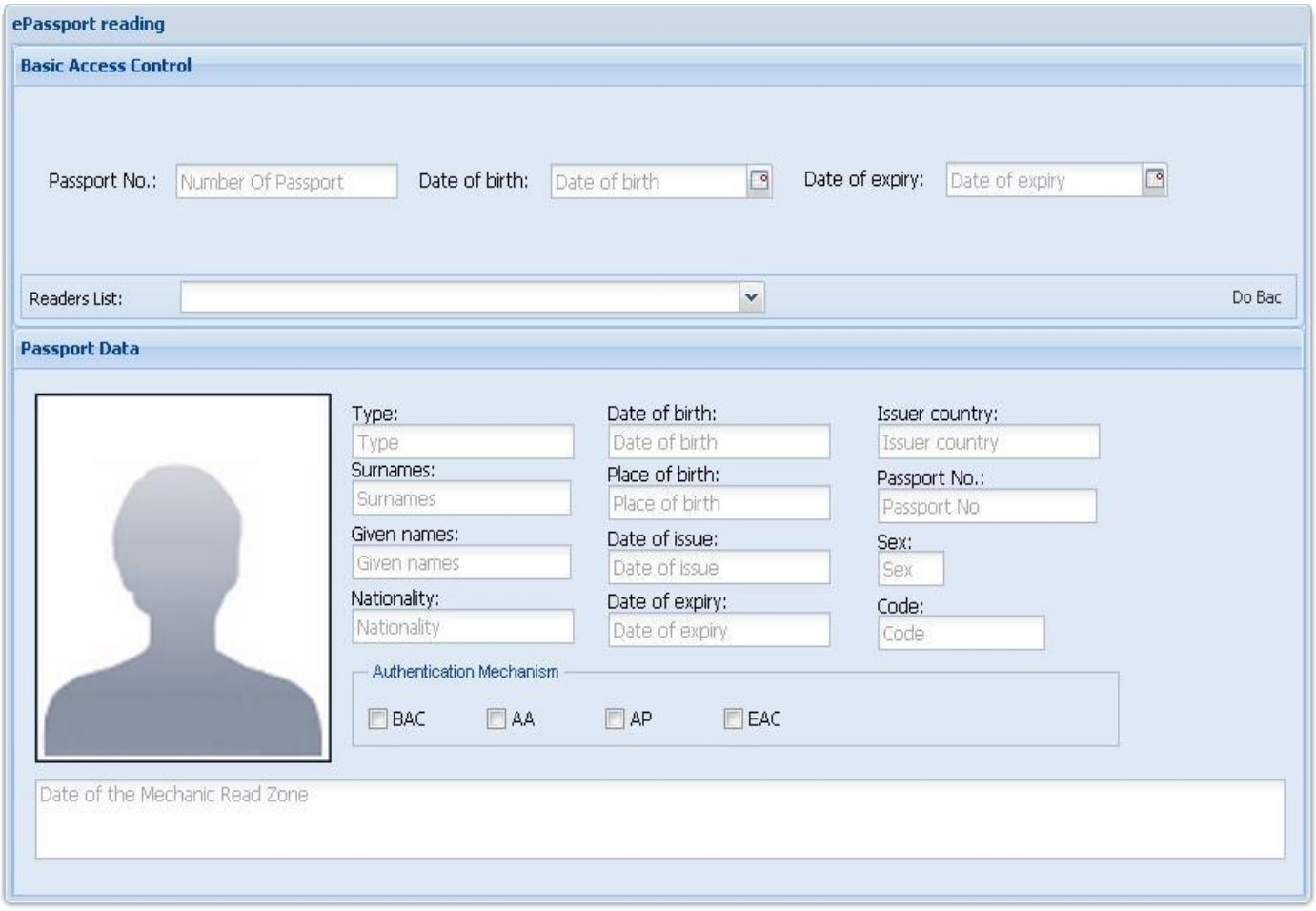


Figura 14: Interfaz para la lectura de documentos electrónicos.

3.7- Fase de producción.

El único instrumento adecuado para determinar el estado de la calidad de un producto de software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el mismo cumple con los requerimientos. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante técnicas de prueba (28).

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

Para la validación de la propuesta de solución, XP divide las pruebas del sistema en dos grupos: pruebas unitarias (caja blanca o estructural) y pruebas de Aceptación (caja negra o funcional).

3.7.1- Pruebas unitarias.

Los programadores continuamente escriben pruebas unitarias antes de empezar a codificar lo cual hará más sencillas y efectivas las pruebas finales. Se corren reiteradamente a lo largo de todo el proyecto, asegurando siempre el funcionamiento correcto de cada componente por individual antes de realizar su integración. Evitan las ambigüedades y los requerimientos quedan afinados en las pruebas.

Las pruebas unitarias sirven para utilizar otro código fuente, llamando directamente a los métodos de una clase pasando los parámetros apropiados y comparar los valores que se generan con los valores esperados. El objetivo de estas pruebas es aislar cada parte del programa y mostrar que las partes individuales son correctas, las cuales van a proporcionar una mejora en la documentación del código, una clara separación de la interfaz y la implementación, los errores están más acotados y son más fáciles de localizar (29).

3.7.1.1- Pruebas de Caja Blanca o Estructurales.

La prueba de caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba. Las pruebas de caja blanca intentan garantizar que se ejecutan al menos una vez todos los caminos independientes de cada módulo así como que sean utilizadas las decisiones en su parte verdadera y en su parte falsa (30).

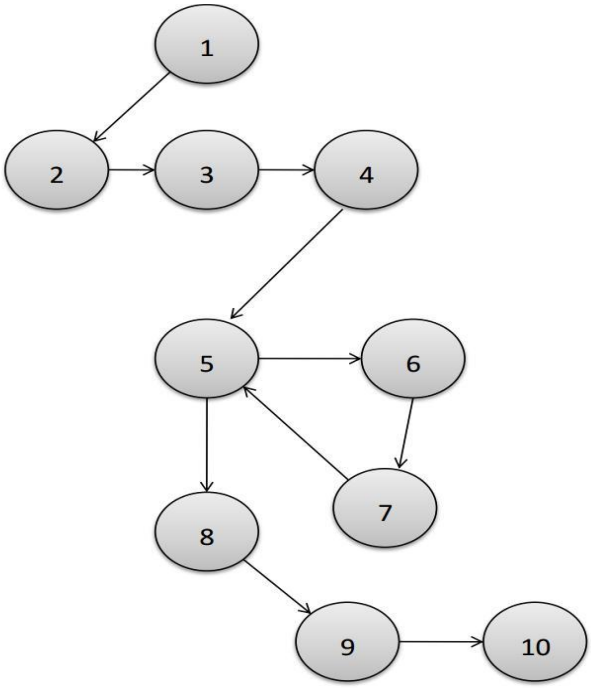
3.7.1.1.1- Casos de prueba.

De acuerdo al siguiente fragmento de código correspondiente a la historia de usuario “Permitir la autenticación mediante el Control de Acceso Básico”, se realiza la prueba de caja blanca.

```
public void success(ResponseAPDU aResponse, String aFrom) {
   try {
      servi = new PassportService();(1)
      cipher = Cipher.getInstance("DESede/CBC/NoPadding", PROVIDER); (1)
      mac = Mac.getInstance("ISO9797Alg3Mac", PROVIDER); (1)
      random = new Random();(1)
      if (mState.equals(State.READY)) (2)
      {
         mState = State.APP_SELECTED; (2)
         String NI = "990010867"; (2)
         Date birth = new Date("1900/12/11");(2)
         Date exp = new Date("2019/09/04"); (2)
         keySeed = Util.computeKeySeed(NI, SDF.format(birth), SDF.format(exp)); (2)
         kEnc = Util.deriveKey(keySeed, Util.ENC_MODE); (2)
         kMac = Util.deriveKey(keySeed, Util.MAC_MODE); (2)
```

```
            transmit(aFrom, getTerminals(aFrom).toArray()[0].toString(),
servi.createGetChallengeAPDU(), this); (2)
        }
        else if (mState.equals(State.APP_SELECTED)) (3)
        {
            mState = State.GETCHALLENGE; (3)
            rndICC = aResponse.getData();(3)
            rndIFD = new byte[8]; (3)
            random.nextBytes(rndIFD); (3)
            kIFD = new byte[16]; (3)
            random.nextBytes(kIFD); (3)
       CommandAPDU aPDU = servi.createMutualAuthAPDU(rndIFD, rndICC, kIFD, kEnc, kMac); (3)
            transmit(aFrom, getTerminals(aFrom).toArray()[0].toString(), aPDU, this); (3)
         }
        else if (mState.equals(State.GETCHALLENGE)) (4)
        {
            mState = State.GETMUTUALAUTHENTICATE; (4)
            byte[] response = servi.sendMutualAuth(aResponse, kEnc); (4)
            byte[] kICC = new byte[16]; (4)
            System.arraycopy(response, 16, kICC, 0, 16); (4)
            keySeed = new byte[16]; (4)
            for (int i = 0; i < 16; i++) (5)
                    {
                            keySeed[i] = (byte) ((kIFD[i] & 0xFF) ^ (kICC[i] & 0xFF)); (6)
                    } (7)
            SecretKey ksEnc = Util.deriveKey(keySeed, Util.ENC_MODE); (8)
            SecretKey ksMac = Util.deriveKey(keySeed, Util.MAC_MODE); (8)
            long ssc = Util.computeSendSequenceCounter(rndICC, rndIFD); (8)
            wrapper = new SecureMessagingWrapper(ksEnc, ksMac, ssc); (9)
            service = new PassportService();(9)
            service.setWrapper(wrapper); (9)
            mLog.debug("bac realizado ******************************************");
        } (10)
```

A partir de las sentencias de código enumeradas se crea el grafo de flujo asociado a estas.



Complejidad ciclomática.

V (G): Número de regiones del grafo.

V (G) = A – N + 2.

V (G) = P + 1.

A: Número de aristas del grafo.

N: Números de nodos.

P: Números de nodos predicados.

A = 10.

N = 10.

P = 0.

V (G) = 2.

Caminos:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

1, 2, 3, 4, 5, 8, 9, 10.

Casos de prueba para cada camino.

Camino: 1, 2, 3, 4, 5, 8, 9, 10.

- ✓ Entrada: Llaves para permitir la autenticación mutua.

- ✓ Salida: Excepción especificando que las condiciones de seguridad necesarias no están satisfechas.
- ✓ Precondiciones: No haber especificado llaves para la autenticación mutua.

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

- ✓ Entrada: Llaves para permitir la autenticación mutua.
- ✓ Salida: 8 bytes aleatorios seguido de 90 00 especificando que la ejecución del comando terminó con éxito.
- ✓ Precondiciones: haber especificado llaves para la autenticación mutua y no haber realizado una autenticación mutua.

3.7.2- Pruebas de aceptación.

Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente, por esto, el cliente es la persona adecuada para diseñar las pruebas de aceptación. Son creadas a partir de las historias de usuario. Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento y no se considera completa hasta que no supera sus pruebas de aceptación. Es responsabilidad del cliente verificar la corrección de las pruebas y tomar decisiones acerca de las mismas.

Las pruebas de aceptación buscan discrepancias entre el programa y sus objetivos o requerimientos, enfocándose en los errores hechos durante la transición del proceso al diseñar la especificación funcional. Esto hace a las pruebas de sistema un proceso vital de pruebas, ya que en términos de producto, número de errores hechos, y severidad de esos errores, es un paso en el ciclo de desarrollo generalmente propenso a la mayoría de los errores. Las pruebas del sistema tienen un propósito particular: comparar el sistema o el programa con sus objetivos originales (requerimientos funcionales y no funcionales) (31).

3.7.2.1- Pruebas de Caja Negra o Funcionales.

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa. Este tipo de pruebas pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene (30).

Por otra parte este tipo de pruebas permite detectar estos tipos de errores:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación (16).

Caso de prueba de aceptación.	
Código de caso de prueba: HU1_CP1	**Nombre de la historia de usuario:** Gestionar la comunicación con lectores de pasaportes disponibles.
Responsable de la prueba: Reisel de la Rosa Ge.	
Descripción de las pruebas: Prueba de funcionalidad para obtener el lector conectado y establecer la conexión con el pasaporte.	
Condiciones de ejecución: Debe haber un pasaporte colocado en el lector previamente conectado.	
Entrada/Pasos de ejecución: • Se reconoce automáticamente el lector para leer el pasaporte electrónico. • Envió de comandos APDU al pasaporte electrónico. • Recibir APDU de respuesta desde el pasaporte electrónico.	
Resultado esperado: Reconocer satisfactoriamente el lector, así como enviar y recibir datos mediantes APDU.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 18: HU1_CP1. Prueba de funcionalidad para obtener el lector conectado y establecer la conexión con el pasaporte.

La especificación de los restantes casos de prueba de aceptación concerniente a cada historia de usuario se encuentra en los anexos 16, 17, 18, 19 y 20. (Ver Anexos)

3.7.3- Resultado de las pruebas.

En las pruebas de caja negras realizadas a las funcionalidades se identificaron 6 casos de prueba en total para las 4 iteraciones. En las correspondientes iteraciones se realizó un caso de prueba por cada iteración, detectándose 20, 12, 6 y 3 no conformidades correspondientes. En las cuatros iteraciones efectuadas se detectaron un total 42 no conformidades, las cuales en su mayoría respondían a errores de bajo impacto en el correcto funcionamiento de la aplicación y todas tuvieron solución en un tiempo máximo de 6 días. Posteriormente se realizó una iteración adicional para verificar que no existieran no conformidades, la misma arrojó un resultado de cero no conformidades, lo que indica que la aplicación web para la lectura de pasaportes electrónicos cumple con las funcionalidades definidas.

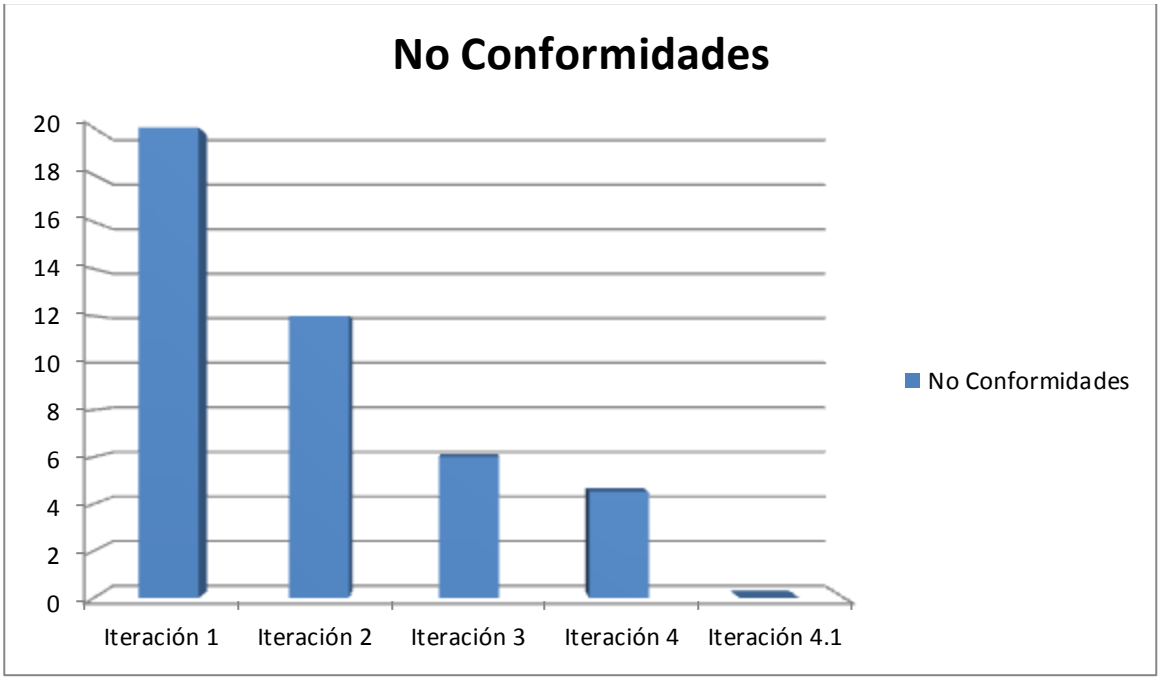


Figura 15: Resultado de las pruebas de caja negra.

3.8- Conclusiones parciales.

- La realización de los diferentes diagramas de ingeniería, específicamente el diagrama de componentes brinda una solución multiplataforma en la cual, tanto en sistemas operativos propietarios como libres.
- Quedaron definidos los tipos y casos de uso de pruebas que se realizaron al sistema para comprobar su buen funcionamiento.
- Con las pruebas realizadas a la aplicación se validó y verificó las funcionalidades descritas, comprobando que el sistema cumple con el nivel de seguridad requerido.

Conclusiones generales.

Producto a la investigación llevada a cabo y como conclusiones generales de la misma, se muestran alcanzados los objetivos propuestos satisfactoriamente:

- El análisis de los principales sistemas que realizan la lectura de pasaportes electrónicos, así como los estándares relacionados con este tipo de documentos permitieron la definición de los requisitos del sistema.
- El estudio de las herramientas y tecnologías necesarias para el desarrollo de la aplicación, permitió definir el entorno de desarrollo para su posterior implementación, así como la metodología que guió el proceso de desarrollo.
- Para la implementación de la solución se realizó la definición de un estándar de codificación que permitió comprender todos los aspectos en cuanto a la generación de código.
- La ejecución de casos de pruebas permitió validar el correcto funcionamiento del sistema, donde se llegó a la conclusión de que los objetivos de la propuesta de solución fueron cumplidos satisfactoriamente.

Recomendaciones.

Al finalizar la presente investigación conjuntamente a partir de la experiencia obtenida en el desarrollo del trabajo y con vista a lograr un aprovechamiento óptimo del resultado alcanzado, se plantean las siguientes recomendaciones para las siguientes versiones de la solución:

- Utilizar la propuesta en los proyectos de software relacionados con la lectura de pasaportes electrónicos.
- Incorporar el proceso de personalización de pasaportes, como nueva funcionalidad a la solución web.

Glosario de términos.

A

Autenticación: Acciones tomadas para asegurar que un documento de viaje es genuino. La autenticación puede incluir tecnología biométrica aplicada a los DVLM.

C

Chip: Un pequeño disco de material conductivo, capaz de almacenar datos y procesarlos para añadirlos a los documentos de viaje. El chip interactuará con un lector del documento para asistir en la autenticación y/o verificación.

D

Documento 9303: Publicación de la OACI que ofrece especificaciones para los DVLM. Actualmente está publicado en tres partes:

- Parte 1. Pasaportes de Lectura Mecánica.
- Parte 2. Visas de Lectura Mecánica.
- Parte 3. Documentos Oficiales de Viaje de Lectura Mecánica.

I

ISO: Organización Internacional para la Normalización.
IEC: Comisión Electrotécnica Internacional.

M

Middleware: Es una librería de software que media entre las aplicaciones que corren en la Tarjeta inteligente y las que corren en una computadora.
MRP: Pasaporte de Lectura Mecánica.
MRTD: Documento de Viaje de Lectura Mecánica, un documento oficial emitido por un Estado u Organización que es utilizado por su titular para viajes internacionales. Este documento contendrá los datos visuales requeridos (lectura ocular) y un resumen de datos separado obligatorio, en un formato capaz de ser leído por una máquina.
MRZ: Zona de Lectura Mecánica de un documento de viaje o visa de lectura mecánica.

O

OACI: Organización de Aviación Civil Internacional: un órgano de la ONU de más de 180 miembros que fija los estándares para los pasaportes y otros documentos de viaje, incluyendo visas.
OCR-B: El estilo de impresión de Reconocimiento de Caracteres Óptico adoptado para uso en la zona de lectura mecánica de un documento de viaje de lectura mecánica.

P

Pasaporte: Documento oficial emitido por un Estado para otorgar permiso a la(s) persona(s) identificada(s) en el pasaporte para viajar, ser readmitido(s) y gozar de protección mientras se encuentra(n) en el extranjero.

R

RFID: siglas de *Radio Frequency Identification,* en español identificación por radiofrecuencia es un sistema de almacenamiento y recuperación de datos remoto que usa dispositivos denominados etiquetas, tarjetas, transponedores RFID.

T

TIC: Tecnologías de la Información y las Comunicaciones.

U

UML: Lenguaje de Modelado Unificado (LUM o UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (*Object Management Group*). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

V

Visa: Autorización temporal o permanente de los viajeros de buena fe, emitida para mostrar que han cumplido los requisitos de admisión a un Estado del cual no son nacionales.

VIZ: Zona de Inspección Visual de un documento de viaje de lectura mecánica.

Bibliografía referenciada.

1. **Organización de Aviación Civil Internacional.** *Organización de Aviación Civil Internacional 9303 Parte I.* Canada : Organización de Aviación Civil Internacional, 2008. Vol. 2. ISBN: 978-92-9231-339-5.

2. **Vicente., Alina Surós.** *El pasaporte electrónico.*

3. **INTERNATIONAL CIVIL AVIATION ORGANIZATION.** *DEVELOPMENT OF A LOGICAL DATA STRUCTURE - LDS For OPTIONAL CAPACITY EXPANSION TECHNOLOGIES.* 2004.

4. **Organización de Aviación Civil Internacional.** *Documentos de Viaje de Lectura Mecánica 9303 Parte I.* Montréal : Organización de Aviación Civil Internacional, 2008. Vol. 1. ISBN: 92-9194-871-3.

5. **Wolfang, Rankl y Effing, Wolfang.** *Smartcard handbook.* Alemania : s.n., 2002. 0-470-85668-8...

6. **W3C.** Extensible Markup Language (XML). *Extensible Markup Language (XML).* [En línea] 2013. http://www.w3.org/XML/.

7. **Frederick, Shea, Ramsay, Colin y Blades, Steve 'Cutter'.** *Learning Ext JS: Build dynamic, desktop-style user interfaces for your data-driven web applications.* s.l. : Packt Publishing Ltd., 2008. 978-1-847195-14-2.

8. **Nelson Hernández Guerra, Adrián Martínez Pérez.** *Applet y su componente middleware para la gestión de información en los pasaportes electrónicos.* Ciudad de La Habana : s.n., Junio 2011. Tesis de diploma.

9. **ISO Organization. 2005.** *INTERNATIONAL STANDARD ISO/IEC 7816-4.* Second Edition 2005.

10. **Gemalto NV.** Gemalto,security to be free. *Gemalto,security to be free.* [En línea] 2006. http://www.gemalto.com/techno/pcsc/.

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

12. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE.* Madrid : addison Wesley Longman Inc., 2000. 84-7829-036-2.

13. **proyectos Ágiles.** Qué es SCRUM | proyectos Ágiles. [En línea] 2012. http://www.proyectosagiles.org/que-es-scrum.

14. **Joskowicz, Ing. José.** *Reglas y Prácticas en eXtreme Programming.* 2008.

15. **Roberth G. Figueroa, Camilo J. Solís,Armando A. Cabrera.** *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* 2008.

16. **Pressman, Roger S.** *Ingeniería del Software: Un Enfoque Práctico(Sexta Edición).* 2005. 9701054733.

17. **ALTOVA.** UModel: una herramienta UML para el modelado de software y desarrollo de aplicaciones. *UModel: una herramienta UML para el modelado de software y desarrollo de aplicaciones.* [En línea] 2013. http://www.altova.com/es/umodel.html.

18. **Valdés, Damián Pérez.** maestros del web. *maestros del web.* [En línea] 1997. http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

19. **ORACLE.** ¿Qué es la tecnología Java y por qué lo necesito? *¿Qué es la tecnología Java y por qué lo necesito?* [En línea] 2010. http://www.java.com/es/download/faq/whatis_java.xml.

20. **Oracle.** NetBeans IDE, The Smarter and Faster Way to Code. *NetBeans IDE, The Smarter and Faster Way to Code.* [En línea] 2013. http://netbeans.org/features/index.html.

21. **Gemalto.** *Coesys Issuance Solutions for the Public Sector.* 2011.

22. **JRMTD team.** JRMTD. [En línea] 2006. [Citado el: 13 de Enero de 2013.] http://jmrtd.org/.

23. **Microsoft Corporation.** MSDN. [En línea] 2013. [Citado el: 25 de Marzo de 2013.] http://msdn.microsoft.com/es-es/library/aa291591%28VS.71%29.aspx.

24. **Vega, Erik de la.** *Estándares de codificación para el Proyecto Identificación, Inmigración y Extranjería de la República de Cuba.* 2010.

25. **Javier, Enrique &.** Rational Rose. [En línea] 2013. [Citado el: 25 de Marzo de 2013.] http://www.buenastareas.com/ensayos/Rational-Rose/1900782.html.

26. **Michael, Marca Huallpara Hugo y Susana, Quisbert Limachi Nancy.** Diagrama de despliegue. [En línea] 2013. [Citado el: 25 de Marzo de 2013.] http://virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc.

27. **Gómez, Leopoldo Sebastián M.** Web Adentro,Diseño de Interfaces de Usuario Principios , Prototipos y Heurísticas para Evaluación. [En línea] Mayo de 2008. [Citado el: 25 de Marzo de 2013.]

http://webadentro.wordpress.com/2008/05/05/diseno-de-interfaces-de-usuario-principios-prototipos-y-heuristicas-para-evaluacion/.

28. **PRUEBAS DE SOFTWARE.** Gestión de Calidad y Pruebas de Software. [En línea] 2005. [Citado el: 26 de Marzo de 2013.] http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

29. **Microsoft.** MSDN, Información general de pruebas unitarias. [En línea] 2013. [Citado el: 26 de Marzo de 2013.] http://msdn.microsoft.com/es-es/library/ms182516%28v=vs.80%29.aspx.

30. **Universidad de Almeria.** Web de la asignatura Laboratorio de proyectos. [En línea] 2013. [Citado el: 26 de Marzo de 2013.] http://indalog.ual.es/mtorres/LP/Prueba.pdf.

31. **PRUEBAS DE SOFTWARE.** Gestión de Calidad y Pruebas de Software. [En línea] 2005. [Citado el:26 de Marzo de 2013.]http://www.pruebasdesoftware.com/pruebadeaceptacion.htm.

Bibliografía consultada

- Extreme Programming: A gentle introduction. *Extreme Programming: A gentle introduction.* [En línea] 1999. http://www.extremeprogramming.org/.

- **ISO 2010.** International Oganization Standardization. *International Oganization Standardization.* [En línea] 2013. http://www.iso.org/iso.

- **Organización Internacional de Aviación Civil.** www.icao.int. *www.icao.int.* [En línea] [Citado el: 19 de Octubre de 2012.] http://www.icao.int.

- **Conallen, Jim.** *Building Web Applications with UML.* 2000.

- **Fernández Santana, Vismar y Pereda Viñolo, Katerina.** *Plataforma para el desarrollo de servicios en línea utilizando tarjetas inteligentes.* 2010.

- **Rolando Santamaría Masó.** *EventsPlugIn Developer Guide Version 1.0.* La Habana : s.n., 2012.

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

- **Fernández, Carlos alberto Fernández y.** *El Proceso Unificado Rational para el desarrollo de software.* Huajaupan de León, Oaxaca : s.n.

- **Consulting, Jacquinot Inc.** Smarter Card Solutions. *Smarter Card Solutions.* [En línea] 2013. http://www.cardwerk.com.

- **Masó, Rolando Santamaría.** *Extensión del marco de trabajo jWebSocket para el desarrollo de aplicaciones web empresariales(EventsPlugIn).* Artemisa,Cuba : s.n., Junio 2012.

- **Freire, Marta Rodríguez.** *API para la gestión de tarjetas inteligentes.* Artemisa,Cuba : s.n., 2012. Tesis de diploma.

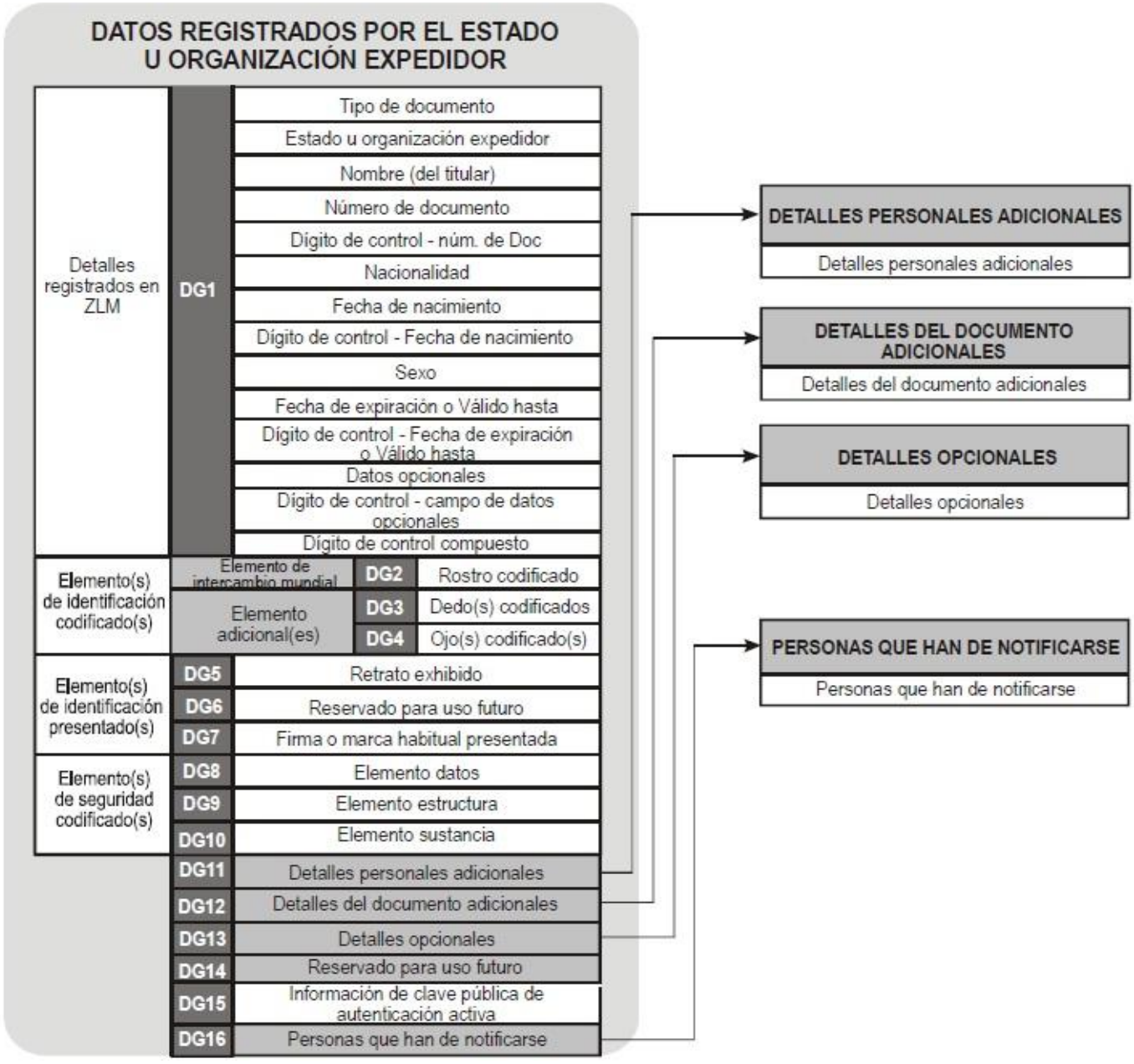
- **Isaías Carrillo Pérez, Rodrigo Pérez González, Aureliano David Rodríguez Martín.** *Metodología de desarrollo del software.* 2008.

- **Gemalto.** Gemalto.2010. DeveloperSuite. [En línea] 2010. [Citado el: 14 de Enero de 2013.] http://www.gemalto.com/.

- **Grupo Soluciones Innova S.A.** GSINNOVA Grupo de Soluciones. [En línea] 2013. http://www.rational.com.ar/herramientas/roseenterprise.html.

Anexos.

Anexo 1: Distribución de los grupos de datos dentro del chip.



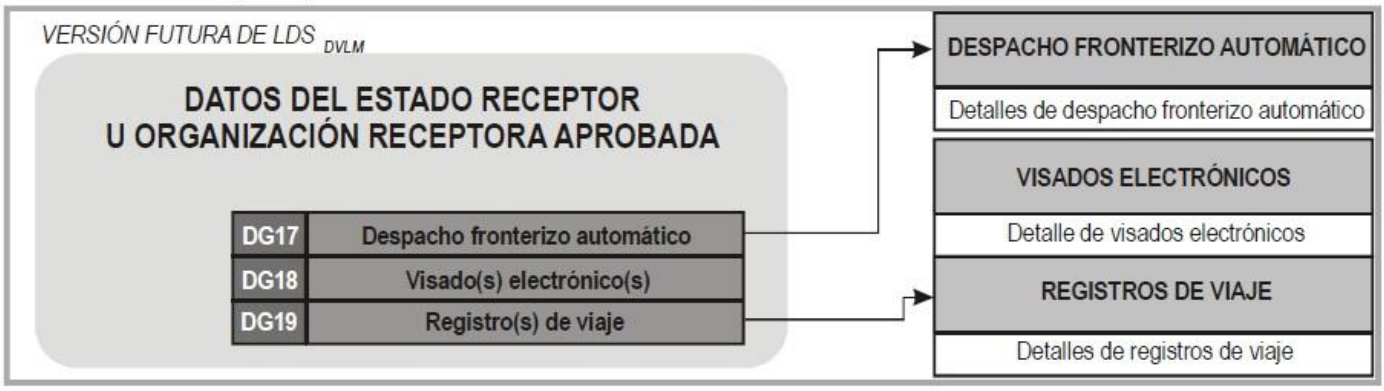


Figura 16: Distribución de los grupos de datos dentro del chip (1).

Anexo 2: Estructura de ficheros almacenados en el pasaporte electrónico.

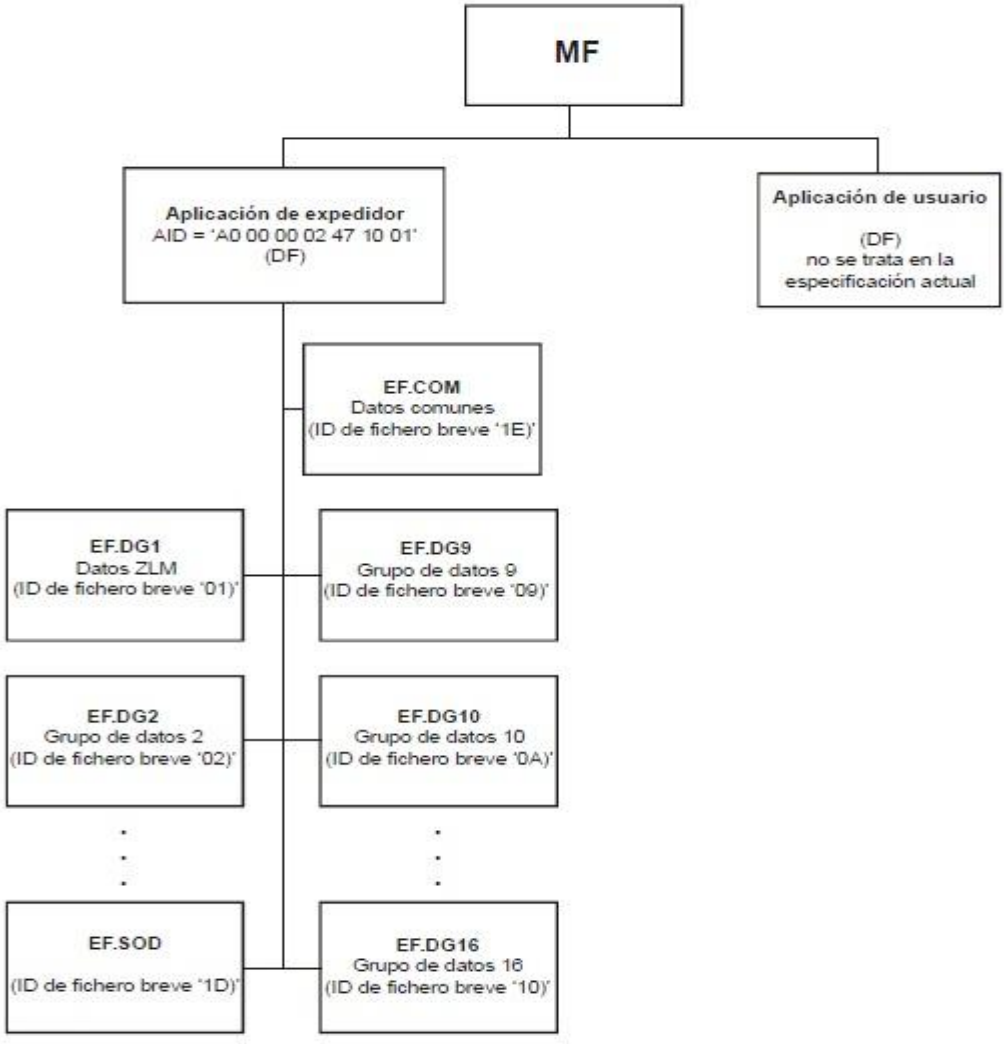


Figura 17: Estructura de ficheros almacenados en el pasaporte electrónico (1).

Anexo 3: Comparación entre las metodologías RUP y XP.

Características	RUP	XP
Tamaño del grupo.	Para grupos grandes y posiblemente distribuidos.	Para grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio.
Obtención de requisitos.	Casos de uso.	Historia de Usuario.
Carga de trabajo.	Mayor.	Menor.
Relación con el cliente.	Se presentan artefactos, es muy formal.	No tiene formalismos.
Documentación.	Mucha.	Poca.
Duración de proyectos.	Largo.	Corto.
Detención de errores.	En forma temprana.	A largo plazo.
Reutilización del código.	Sí.	Sí.
Simplicidad en el diseño.	No.	Sí.
Centrado en la arquitectura.	Sí.	No.
Soporte técnico continuo.	Menor.	Mayor.
Diseño simple.	No.	Sí.
Desarrollo.	Iterativo.	Iterativo.
Evaluación del estado de proyecto	Largo.	Corto.
Accesibilidad al código fuente por parte del cliente.	Poca.	Mucha.

Tabla 19: Comparación entre las metodologías RUP y XP.

Anexo 4: HU_1 Gestionar la comunicación con lectores de pasaportes disponibles.

Historia de usuario	
Número: HU_1	**Usuario:** Desarrollador.
Nombre de Historia de Usuario: Gestionar la comunicación con lectores de pasaportes disponibles.	
Prioridad en negocio: Alta.	**Iteración asignada:** 1
Riesgo en desarrollo: Baja.	**Puntos estimados:** 2
Programador responsable: Reisel de la Rosa Ge.	
Descripción: Permitirá la comunicación mediante el estándar PC/SC, obteniendo los lectores disponibles conectados a la terminal y estableciendo la conexión con el documento de viaje electrónico. Permitirá además enviar peticiones y obtener la respuesta del pasaporte electrónico. Esta tiene dos estados posibles: • Conectada en el lector. • Desconectada del lector. Existen varios casos en que la conexión con el pasaporte puede ser interrumpido: • Si el usuario extrae el pasaporte del lector.	

• Si el usuario desconecta el lector de la computadora. • Si el usuario cierra la conexión a través de la interfaz web.
Observaciones: En caso de que no haya lector conectado la aplicación mostrará la ausencia de los mismos al usuario.

Tabla 20. HU_1 Gestionar la comunicación con lectores de pasaportes disponibles.

Anexo 5: HU_2 Permitir la autenticación mediante el Control de Acceso Básico (BAC).

Historia de usuario	
Número: HU_2	**Usuario:** Desarrollador.
Nombre de Historia de Usuario: Permitir la autenticación mediante el Control de Acceso Básico (BAC).	
Prioridad en negocio: Alta.	**Iteración asignada:** 1
Riesgo en desarrollo: Alta.	**Puntos estimados:** 3
Programador responsable: Reisel de la Rosa Ge.	
Descripción: Permitirá la autenticación mediante el mecanismo de control de acceso básico con el pasaporte electrónico para el establecimiento de un canal seguro con el terminal.	
Observaciones:	

Tabla 21. HU_2 Permitir la autenticación mediante el Control de Acceso Básico (BAC).

Anexo 6: HU_3 Enviar y recibir instrucciones del pasaporte electrónico.

Historia de usuario	
Número: HU_3	**Usuario:** Desarrollador.
Nombre de Historia de Usuario: Enviar y recibir instrucciones del pasaporte electrónico.	
Prioridad en negocio: Alta.	**Iteración asignada:** 1
Riesgo en desarrollo: Media.	**Puntos estimados:** 2
Programador responsable: Reisel de la Rosa Ge.	
Descripción: Permitirá verificar las condiciones de acceso en el pasaporte electrónico para posteriormente leer y mostrar la información contenida en el mismo. Cuando hay una conexión abierta entre un lector y el pasaporte electrónico, el usuario hace la solicitud del servicio a través de la página web, que es procesada por el servidor y este envía los correspondientes comandos que son transmitidos al pasaporte a través de un componente cliente, que accede a la capa que se comunica con el mismo. A este comando el pasaporte siempre devolverá una respuesta que puede ser satisfactoria, la información solicitada o un mensaje de error. El pasaporte procesa el comando enviado y manda la respuesta, que luego el componente cliente se encargará de enviar al servidor nuevamente hasta que termine la operación.	
Observaciones: Si el pasaporte no está en estado conectado en el lector, se notificara la ocurrencia de un error.	

Tabla 22. HU_3 Enviar y recibir comandos APDU del pasaporte electrónico.

Anexo 7: HU_4 Gestionar la transmisión de información entre el cliente web y el servidor JWebSocket.

Historia de usuario	
Número: HU_4	**Usuario:** Desarrollador.
Nombre de Historia de Usuario: Gestionar la transmisión de información entre el cliente web y el servidor JWebsocket.	
Prioridad en negocio: Alta.	**Iteración asignada:** 2
Riesgo en desarrollo: Media.	**Puntos estimados:** 2.8
Programador responsable: Reisel de la Rosa Ge.	
Descripción: Cuando el usuario solicita un servicio, el cliente se encarga de enviarlo al servidor, este envía un primer comando comenzando el intercambio de comandos y respuestas entre el middleware y el pasaporte. El puente para esta comunicación será el controlador del lado del servidor.	
Observaciones:	

Tabla 23. HU_4 Gestionar transmisión de información entre el cliente web y el servidor JWebsocket.

Anexo 8: HU_5 Procesar las operaciones del middleware en el servidor.

Historia de usuario	
Número: HU_5	**Usuario:** Desarrollador.
Nombre de Historia de Usuario: Procesar las operaciones del middleware en el servidor.	
Prioridad en negocio: Alta.	**Iteración asignada:** 2
Riesgo en desarrollo: Media.	**Puntos estimados:** 3
Programador responsable: Reisel de la Rosa Ge.	
Descripción: Teniendo en cuenta que las operaciones comienzan en el servidor, una vez recibida la primera petición del cliente de un servicio, se invoca la función correspondiente al middleware que comenzará el intercambio de información.	
Observaciones:	

Tabla 24. HU_5 Procesar las operaciones del middleware en el servidor.

Anexo 9: HU_6 Procesar Mostrar al usuario el resultado tras la culminación de la operación del middleware en el servidor.

Historia de usuario	
Número: HU_6	**Usuario:** Desarrollador.
Nombre de Historia de Usuario: Mostrar al usuario el resultado tras la culminación de la operación del middleware en el servidor.	
Prioridad en negocio: Alta.	**Iteración asignada:** 2
Riesgo en desarrollo: Media.	**Puntos estimados:** 1
Programador responsable: Reisel de la Rosa Ge.	

Descripción: Cuando el middleware que se encuentra implementado en el servidor termina el envío de información hacia el cliente y termina el procesamiento de las respuestas enviadas desde el pasaporte, se le notifica a la aplicación que ha concluido la operación en curso y le envía los resultados. El controlador del servidor se encargará de enviar la respuesta al cliente web para que se le muestre al usuario la información final.
Observaciones: Siempre se le mostrará al usuario el resultado de la operación finalizada.

Tabla 25. HU_6 Mostrar al usuario los resultados tras la culminación de cada una de las operaciones.

Anexo 10: Ejemplos relacionados a cada patrón de diseño.

```
public class PasaportePlugIn extends JcPlugIn {

    private static Logger mLog = Logging.getLogger(CanalSeguroPlugIn.class);
    private byte[] mAppName = new byte[]{(byte) 0xA0, 0x00, 0x00, 0x02, 0x47, 0x10, 0x01};
    private State mState;
    int contarsbytedefotos=0;
    BASE64Encoder encoder = new BASE64Encoder();
    HashMap rawstream= new HashMap();
    private ArrayList<byte[]>aRespuesta= new ArrayList<>();
    Map pasaport = new FastMap<>();
    byte[] rndICC;
    short[] fid = new short[]{286,259, 285, 257, 258};
    byte[] rndIFD;
    byte[] kIFD;
    private static final SimpleDateFormat SDF = new SimpleDateFormat("yyMMdd");
    private static final Provider PROVIDER = new org.bouncycastle.jce.provider.BouncyCastleProvider();
    private Random random;
    Passport passport;
    JcResponseCallback callback;
    protected SecureMessagingWrapper wrapper;
    C2SEvent even = null;
    private Signature aaSignature;
    PassportApduService servi;
    PassportService service;
    byte[] keySeed;
    SecretKey kEnc;
    SecretKey kMac;
    ResponseAPDU aresponse = null;
    /**...*/
    private Cipher cipher;
    /**...*/
    private Mac mac;
    private String aux;
    /**...*/
    private static final IvParameterSpec ZERO_IV_PARAM_SPEC = new IvParameterSpec(
            new byte[8]);


    public void processEvent(ReadPassport aEvent, C2SResponseEvent aResponseEvent) throws Exception {

    public byte[]  joinArray(ArrayList<byte []>lista)
    {...}
```

Figura 18: Ejemplo de patrón Experto.

```
abstract class CBEFFDataGroup extends DataGroup
{
        static final int BIOMETRIC_INFORMATION_GROUP_TEMPLATE_TAG = 0x7F61;
        static final int BIOMETRIC_INFORMATION_TEMPLATE_TAG = 0x7F60;

        static final int BIOMETRIC_INFO_COUNT_TAG = 0x02;
        static final int BIOMETRIC_HEADER_TEMPLATE_BASE_TAG = (byte) 0xA1;
        static final int BIOMETRIC_DATA_BLOCK_TAG = 0x5F2E;
        static final int BIOMETRIC_DATA_BLOCK_TAG_ALT = 0x7F2E;

        static final int FORMAT_OWNER_TAG = 0x87;
        static final int FORMAT_TYPE_TAG = 0x88;

        /** From ISO7816-11: Secure Messaging Template tags. */
        static final int
        SMT_TAG = 0x7D,
        SMT_DO_PV = 0x81,
        SMT_DO_CG = 0x85,
        SMT_DO_CC = 0x8E,
        SMT_DO_DS = 0x9E;

        protected List<byte[]> templates;

        protected CBEFFDataGroup() {...}

        /**...*/
        public CBEFFDataGroup(InputStream in) {...}

        private void readBIT(BERTLVInputStream tlvIn, int templateIndex) throws IOException {...}

        /**...*/
        private void readBHT(int headerTemplateTag, int headerTemplateLength, int templateIndex, BERTLV

        /**...*/
        private void readStaticallyProtectedBIT(int tag, int length, int templateIndex, BERTLVInputStre

        private byte[] decodeSMTValue(BERTLVInputStream tlvIn) throws IOException {...}

        private void readBiometricDataBlock(BERTLVInputStream tlvIn) throws IOException {...}

        /**...*/
```

Figura 19: Ejemplo de patrón Bajo Acoplamiento.

```
public class MRZInfo
{
        /** Unspecified document type (do not use, choose ID1 or ID3). */
        public static final int DOC_TYPE_UNSPECIFIED = 0;
        /** ID1 document type for credit card sized national identity cards. */
        public static final int DOC_TYPE_ID1 = 1;
        /** ID2 document type. */
        public static final int DOC_TYPE_ID2 = 2;
        /** ID3 document type for passport booklets. */
        public static final int DOC_TYPE_ID3 = 3;
        PassportService servi;
        String string= "";

        private static final String MRZ_CHARS = "<0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";

        private static final SimpleDateFormat SDF =
                new SimpleDateFormat("yyMMdd");

        private String documentType;
        private Country issuingState;
        private String primaryIdentifier;
        private String[] secondaryIdentifiers;
        private Country nationality;
        private String documentNumber;
        private String personalNumber;
        private Date dateOfBirth;
        private Gender gender;
        private Date dateOfExpiry;
        private char documentNumberCheckDigit;
        private char dateOfBirthCheckDigit;
        private char dateOfExpiryCheckDigit;
        private char personalNumberCheckDigit;
        private char compositeCheckDigit;
        private String optionalData2; // FIXME: Last field on line 2 of ID3 MRZ.
```

Figura 20: Ejemplo de patrón Alta Cohesión.

Anexo 11: Especificaciones de las tareas de ingeniería.

Tarea de ingeniería	
Número de tarea: HU1_T1	**Historia de Usuario:** Gestionar la comunicación con lectores de pasaportes disponibles.
Nombre: Establecer conexión con el pasaporte.	
Tipo: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 14/1/2013.	**Fecha fin:** 18/1/2013.
Responsable: Reisel de la Rosa Ge.	

Descripción: Inicialmente cuando se carga la página, el sistema intenta establecer conexión con el pasaporte si está insertado, sino luego que el usuario lo introduzca, dicho sistema debe hacerlo para iniciar la conexión.

Tabla 26. HU1_T1 Establecer conexión con el pasaporte.

Tarea de ingeniería	
Número de tarea: HU1_T1	**Historia de Usuario:** Gestionar la comunicación con lectores de pasaportes disponibles.
Nombre: Cerrar conexión con el pasaporte.	
Tipo: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 21/1/2013.	**Fecha fin:** 25/1/2013.
Responsable: Reisel de la Rosa Ge.	
Descripción: La conexión con la tarjeta puede ser cerrada en cualquiera de estos escenarios: • Si el usuario saca la tarjeta del lector. • Si el usuario desconecta el lector de la computadora. • Si el usuario cierra la conexión a través de la página web.	

Tabla 27. HU1_T1 Cerrar conexión con el pasaporte.

Tarea de ingeniería	
Número de tarea: HU2_T1	**Historia de Usuario:** Permitir la autenticación mediante el Control de Acceso Básico (BAC).
Nombre: Obtener los datos del código de barra de la zona de lectura mecánica.	
Tipo: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 04/2/2013.	**Fecha fin:** 08/2/2013.
Responsable: Reisel de la Rosa Ge.	
Descripción: El lector envía los datos previamente obtenidos de la zona de lectura mecánica hacia el pasaporte para que los mismos sean verificados y así otorgar el permiso para iniciar la comunicación de forma segura.	

Tabla 28. HU2_T1 Obtener los datos del código de barra de la zona de lectura mecánica.

Tarea de ingeniería	
Número de tarea: HU3_T1	**Historia de Usuario:** Enviar y recibir instrucciones del pasaporte electrónico.
Nombre: Enviar peticiones al pasaporte.	
Tipo: Desarrollo.	**Puntos estimados:** 2
Fecha inicio: 11/2/2013.	**Fecha fin:** 22/2/2013.
Responsable: Reisel de la Rosa Ge.	
Descripción: El componente cliente recibe la instrucción desde el servidor y se lo envía al pasaporte a través del elemento encargado de la comunicación con la misma.	

Tabla 29. HU3_T1 Enviar peticiones al pasaporte.

Tarea de ingeniería	
Número de tarea: HU3_T2	**Historia de Usuario:** Enviar y recibir instrucciones del pasaporte electrónico.
Nombre: Recibir respuesta del pasaporte.	
Tipo: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 25/2/2013.	**Fecha fin:** 28/2/2013.
Responsable: Reisel de la Rosa Ge.	
Descripción: Luego que se le envía la primera instrucción al pasaporte, el sistema se quedará esperando una respuesta. El pasaporte procesa la instrucción enviada desde el componente y devuelve una respuesta, comenzando el ciclo nuevamente.	

Tabla 30. HU3_T2 Recibir respuesta del pasaporte.

Tarea de ingeniería	
Número de tarea: HU4_T1	**Historia de Usuario:** Gestionar la transmisión de información entre el cliente web y el servidor JWebSocket.
Nombre: Enviar respuesta que genera el middleware al cliente.	
Tipo: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 11/3/2013.	**Fecha fin:** 15/3/2013.
Responsable: Reisel de la Rosa Ge.	
Descripción: Luego que el usuario solicita un servicio y la petición es enviada al servidor para ser atendida por el middleware que lo implementa, este elabora la respuesta para enviarlo al pasaporte y pedir la información que necesita, el controlador en el servidor se encargará del envío del comando al cliente web.	

Tabla 31. HU4_T1 Enviar respuesta que genera el middleware al cliente.

Tarea de ingeniería	
Número de tarea: HU4_T2	**Historia de Usuario:** Gestionar la transmisión de información entre el cliente web y el servidor JWebSocket.
Nombre: Enviar respuesta recibida del pasaporte al servidor.	
Tipo: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 25/3/2013.	**Fecha fin:** 26/4/2013.
Responsable: Reisel de la Rosa Ge.	
Descripción: El pasaporte cuando recibe una instrucción siempre devuelve una respuesta, que es manejada por el elemento que interactúa con él enviándolo al componente cliente para mandarlo al servidor donde será procesado.	

Tabla 32. HU4_T2 Enviar respuesta recibida de la tarjeta al servidor.

Tarea de ingeniería	
Número de tarea: HU5_T1	**Historia de Usuario:** Procesar las operaciones del middleware en el servidor.
Nombre: Invocar funcionalidad de un middleware específico.	

Tipo: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 29/4/2013.	**Fecha fin:** 03/5/2013.
Responsable: Reisel de la Rosa Ge.	
Descripción: Usando una instancia del middleware al que se está invocando se accede a la funcionalidad requerida por el usuario, pasando los parámetros necesarios.	

Tabla 33. HU5_T1 Invocar funcionalidad de un middleware específico.

Tarea de ingeniería	
Número de tarea: HU6_T1	**Historia de Usuario:** Mostrar al usuario el resultado tras la culminación de la operación del middleware en el servidor.
Nombre: Enviar respuesta al cliente web.	
Tipo: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 06/5/2013.	**Fecha fin:** 10/5/2013.
Responsable: Reisel de la Rosa Ge.	
Descripción: Una vez que el middleware específico procesa las respuestas enviadas desde la tarjeta, elabora la respuesta y notifica la culminación de la tarea, enviando la respuesta página web para que sea mostrada al usuario.	

Tabla 34. HU6_T1 Enviar respuestas al cliente web.

Anexo 12: Descripción de las tarjetas CRC.

CBEFFDataGroup	
Responsabilidades	**Colaboradores**
Crea un fichero en el que se define la estructura jerárquica de los archivos que almacenan los datos biométricos.	java.io.ByteArrayInputStream java.io.ByteArrayOutputStream java.io.DataInputStream java.io.IOException java.io.InputStream java.util.ArrayList java.util.List org.jwebsocket.plugins.jc.pasaporte.tlv.BERTLVInputStream

Tabla 35: Tabla CRC CBEFFDataGroup.

MRZInfo	
Responsabilidades	**Colaboradores**
Se crea una estructura de los datos que deben ser almacenados en la zona de lectura mecánica del pasaporte electrónico.	java.io.DataOutputStream java.io.IOException java.text.SimpleDateFormat java.util.ArrayList java.util.Collection

	java.util.Date java.util.GregorianCalendar java.util.StringTokenizer org.jwebsocket.plugins.jc.pasaporte.data.Country org.jwebsocket.plugins.jc.pasaporte.smartcards.CardServiceException test.pasaporte.data.Gender

Tabla 36: Tabla CRC MRZInfo.

FaceInfo	
Responsabilidades	**Colaboradores**
Se crea una estructura para el almacenamiento de la imagen del titular del pasaporte electrónico.	java.awt.Rectangle java.awt.image.BufferedImage java.io.ByteArrayInputStream java.io.ByteArrayOutputStream java.io.DataInputStream java.io.DataOutputStream java.io.IOException java.io.InputStream java.io.OutputStream java.util.ArrayList java.util.Collection java.util.Iterator javax.imageio.ImageIO javax.imageio.ImageReadParam javax.imageio.ImageReader javax.imageio.ImageWriteParam javax.imageio.ImageWriter javax.imageio.stream.ImageInputStream javax.imageio.stream.ImageOutputStream test.pasaporte.data.Gender

Tabla 37: Tarjeta CRC FaceInfo.

PassportPlugin	
Responsabilidades	**Colaboradores**
Se genera la zona de lectura mecánica. Se crean todos los grupos de ficheros. Lectura de los ficheros.	import java.io.BufferedInputStream; java.io.ByteArrayInputStream java.io.ByteArrayOutputStream

Llamadas a los mecanismos de seguridad. Clase controladora del sistema.	java.io.IOException java.io.InputStream java.io.PipedInputStream java.io.PipedOutputStream java.security.PrivateKey java.util.ArrayList java.util.HashMap java.util.List java.util.Map java.math.BigInteger java.security.GeneralSecurityException java.security.KeyPair java.security.KeyPairGenerator java.security.MessageDigest java.security.PublicKey java.security.cert.X509Certificate java.util.Calendar java.util.Collections java.util.Date java.util.TreeMap org.bouncycastle.asn1.x509.X509Name org.bouncycastle.x509.X509V3CertificateGenerator org.jwebsocket.plugins.jc.pasaporte.smartcards.CardFileInputStream org.jwebsocket.plugins.jc.pasaporte.smartcards.CardServiceException

Tabla 38: Tarjeta CRC Passport.

PassportApduService	
Responsabilidades	**Colaboradores**
Encargada de crear la estructura del APDU.	import java.security.GeneralSecurityException; import java.security.Provider; import java.security.Signature; java.text.SimpleDateFormat java.util.List java.util.Random javax.crypto.Cipher javax.crypto.Mac

	javax.crypto.SecretKey javax.crypto.spec.IvParameterSpec javax.smartcardio.Card javax.smartcardio.CardTerminal javax.smartcardio.CardTerminals javax.smartcardio.CommandAPDU javax.smartcardio.ResponseAPDU javax.smartcardio.TerminalFactory org.jwebsocket.eventmodel.event.C2SEvent org.jwebsocket.eventmodel.exception.MissingTokenSenderException org.jwebsocket.eventmodel.plugin.jc.JcResponseCallback org.jwebsocket.plugins.jc.pasaporte.data.Hex org.jwebsocket.plugins.jc.pasaporte.smartcards.CardService org.jwebsocket.plugins.jc.pasaporte.smartcards.CardServiceException org.jwebsocket.plugins.jc.pasaporte.smartcards.ISO7816

Tabla 39: Tarjeta CRC PassportApduService.

Anexo 13: Plan de entrega.

Entregable	Fin Iteración 1	Fin Iteración 2
Aplicación web para la lectura de pasaportes electrónicos.	Marzo 2013	Mayo 2013

Tabla 40. Plan de entregas.

Anexo 14: Estimación de tiempo.

Historias de Usuario	Estimación
Gestionar la comunicación con lectores de pasaportes.	**1**
Permitir la autenticación mediante el Control de Acceso Básico (BAC).	**3**
Enviar y recibir instrucciones del pasaporte electrónico.	**2**
Gestionar la transmisión de información entre el cliente web y el servidor JWebSocket.	**6**
Procesar las operaciones del middleware en el servidor.	**3**
Mostrar al usuario el resultado tras la culminación de la operación del middleware en el servidor.	**2**

Tabla 41. Estimación de tiempo de las Historias de Usuario (HU).

Anexo 15: Plan de iteraciones.

Iteración	No. HU	Historia de Usuario	Duración estimada (semanas)
	HU_1	Gestionar la comunicación con lectores de pasaportes disponibles.	

Iteración 1	HU_2	Permitir la autenticación mediante el Control de Acceso Básico (BAC).	**8**
	HU_3	Enviar y recibir instrucciones del pasaporte electrónico.	
Iteración 2	HU_4	Gestionar la transmisión de información entre el cliente web y el servidor JWebSocket.	**10**
	HU_5	Procesar las operaciones del middleware en el servidor.	
	HU_6	Mostrar al usuario el resultado tras la culminación de la operación del middleware en el servidor.	

Tabla 42. Plan de iteraciones.

Anexo 16: HU2_CP1 Permitir la autenticación mediante el Control de Acceso Básico (BAC).

Caso de prueba de aceptación.	
Código de caso de prueba: HU2_CP1	**Nombre de la historia de usuario:** Permitir la autenticación mediante el Control de Acceso Básico (BAC).
Responsable de la prueba: Reisel de la Rosa Ge.	
Descripción de las pruebas: Prueba de funcionalidad para verificar la comunicación con el terminal mediante un canal seguro.	
Condiciones de ejecución: Debe existir algún lector conectado para leer la zona de lectura mecánica o introducir los datos de la zona de lectura mecánica manualmente.	
Entrada/Pasos de ejecución: • Recibir los datos de la zona de lectura mecánica enviados desde el lector o introducidos manualmente. • Generar clave mediante los datos de la zona de lectura mecánica. • Comprobar la clave generada con la almacenada en el chip del pasaporte electrónico. • Permitir la comunicación cifrando los datos.	
Resultado esperado: Obtención y autenticación de la clave de la zona de lectura mecánica y permitir la comunicación segura con la terminal.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 43: HU2_CP1. Prueba de funcionalidad para verificar la comunicación con el terminal mediante un canal seguro.

Anexo 17: HU3_CP1 Enviar y recibir instrucciones del pasaporte electrónico.

Caso de prueba de aceptación.	
Código de caso de prueba: HU3_CP1	**Nombre de la historia de usuario:** Enviar y recibir instrucciones del pasaporte electrónico.
Responsable de la prueba: Reisel de la Rosa Ge.	
Descripción de las pruebas: Prueba de funcionalidad para el intercambio de APDU entre el lector y el pasaporte.	
Condiciones de ejecución: Debe existir comunicación con la terminal mediante un canal seguro.	

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 al pasaporte. • El pasaporte procesa el comando APDU y devuelve una respuesta que será • enviada al middleware en el servidor nuevamente.
Resultado esperado: Leer y modificar satisfactoriamente la información almacenada en los pasaportes electrónicos.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 44: HU3_CP1. Prueba de funcionalidad para el intercambio de APDU entre el lector y el pasaporte.

Anexo 18: HU4_CP1 Gestionar la transmisión de información entre el cliente web y el servidor JWebSocket.

	Caso de prueba de aceptación.
Código de caso de prueba: HU4_CP1	**Nombre de la historia de usuario:** Gestionar la transmisión de información entre el cliente web y el servidor JWebSocket.
Responsable de la prueba: Reisel de la Rosa Ge.	
Descripción de las pruebas: Prueba de funcionalidad para las transmisión de comandos APDU entre el servidor y el cliente web.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: • El usuario hace una solicitud de servicio. • Una vez allí se envía al middleware que se está invocando, quien comienza la secuencia de los APDU. • Se gestiona el envío de este comando APDU al cliente web iniciándose la comunicación.	
Resultado esperado: Se establece una comunicación entre el cliente web y el servidor.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 45: HU4_CP1. Prueba de funcionalidad para las transmisión de comandos APDU entre el servidor y el cliente web.

Anexo 19: HU5_CP1 Procesar las operaciones del middleware en el servidor.

	Caso de prueba de aceptación.
Código de caso de prueba: HU5_CP1	**Nombre de la historia de usuario:** Procesar las operaciones del middleware en el servidor.
Responsable de la prueba: Reisel de la Rosa Ge.	
Descripción de las pruebas: Prueba de funcionalidad para realizar las operaciones del middleware.	

Condiciones de ejecución:
Entrada/Pasos de ejecución: • El usuario hace una solicitud de servicio. • Esta es enviada al servidor con los parámetros necesarios. • En el servidor el middleware invoca el método correspondiente para dar solución a la solicitud del usuario.
Resultado esperado: Se establece una comunicación entre el cliente web y el servidor.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 46: HU5_CP1. Prueba de funcionalidad para realizar las operaciones del middleware.

Anexo 20: HU6_CP1: Mostrar al usuario el resultado tras la culminación de la operación del middleware en el servidor.

	Caso de prueba de aceptación.
Código de caso de prueba: HU6_CP1	**Nombre de la historia de usuario:** Mostrar al usuario el resultado tras la culminación de la operación del middleware en el servidor.
Responsable de la prueba: Reisel de la Rosa Ge.	
Descripción de las pruebas: Prueba de funcionalidad para la notificación al usuario del resultado del servicio solicitado.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: • El usuario solicita un servicio que es enviada al servidor a través del cliente. • Se establece la comunicación entre el middleware y el pasaporte. • Se concluye la operación iniciada. • Se muestran los resultados al usuario.	
Resultado esperado: Se muestran los resultados al usuario a través de la interfaz web.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 47: HU6_CP1. Prueba de funcionalidad para la notificación al usuario del resultado del servicio solicitado.