UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

Trabajo de Diploma para Optar por el Título de

Ingeniero en Ciencias Informáticas

Solución para gestionar la información del documento licencia de conducción en tarjetas inteligentes.

Autor:

Lissi Fernández Miyet

Tutor: Ing. Ander Sánchez Jardines

Co-tutor: Ing. Dayron Almeida Sotolongo

Consultante: MSc. Adonis César Legón Campo

"Ciudad de La Habana. Julio, 2011"

"Año 53 de la Revolución"

Declaración de Autoría

Declaro que soy la única autora de este trabajo y autorizo al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Lissi Fernández Miyet

_____ _____

Ing. Dayron Almeida Sotolongo Ing. Ander Sánchez Jardines

Datos de Contacto

Síntesis de los Tutores:

- Dayron Almeida Sotolongo. Ingeniero en Ciencias Informáticas, instructor recién graduado de la Universidad de las Ciencias Informáticas. Presenta 2 años de experiencia en el tema de tarjetas inteligentes.
- Ander Sánchez Jardines. Ingeniero en Ciencias Informáticas, instructor recién graduado de la Universidad de las Ciencias Informáticas. Presenta 1 año de experiencia en el tema de tarjetas inteligentes.

Agradecimiento

A la Revolución por darme la oportunidad de superarme en esta maravillosa escuela.

A mis tutores Dayron y Ander por el apoyo brindado durante todo el desarrollo de la tesis.

A Yurdik, Alinia, Osmín, Adonis y Lurdes por su ayuda incondicional y la atención dedicada.

A mi mamá y a mi papá por apoyarme siempre, brindarme todo lo necesario para mi superación, darme fuerzas en los momentos en que pensé que no se podía, por el amor, la dedicación y estar siempre orgullosos de mí. Los quiero mucho, son el tesoro más grande que me dio la vida.

A mi abuelito por la preocupación, su cariño y apoyo, te quiero mucho.

A mis hermanitos Guillermito y Lienys que nacieron para aumentar la felicidad en mi corazón, son mi razón de ser.

A mi hermanita Leyanis por estar siempre a mi lado.

A mis tíos Ernesto, Aracelis, Sirenia, Mariela, Walfrido y Estrellita por darme las fuerzas para seguir adelante y sus consejos.

A mis primitos lindos Frank y Elisabeth, que son para mí como hermanitos, gracias por todo el cariño que tienen para darme.

A Fidel mi novio por darme su amor y por hacerme sentir la mujer más feliz del mundo.

A mis suegros por apoyarme y ayudarme como padres en estos últimos 3 años.

A mis amigos Made, Yuliet, Lester, Yadira y Tatiana por todo el apoyo que me brindaron (y yo a ellos jajaja), la paciencia y ser los mejores amigos que he tenido, gracias por hacerme saber que siempre están ahí.

A Heriberto, mis compañeros de grupo y al colectivo del departamento de Tarjetas Inteligentes por toda la ayuda brindada.

A mis compañeras de cuarto Susana, Kirenia, Yanalia, Sahyli, Dayle, Zoraimi, Laurita e Idania por todo el apoyo.

Dedicatoria

A mis padres Liliana y Guillermo, a ellos les debo mis resultados y haber alcanzado una de mis metas más altas, espero que se sientan orgullosos de mí.
Los quiero mucho.

Resumen

La licencia de conducción[1], es un documento que existe en todas las naciones y su objetivo fundamental es legalizar ante las autoridades el permiso para conducir un determinado vehículo, por su alto valor a nivel mundial y la seguridad que requiere ha ido evolucionando desde un documento de cartón, utilizando un formato plastificado con ciertas características como imágenes codificadas, lámina holográfica de seguridad, tinta sólo visible a la luz ultravioleta y otras, hasta dispositivos electrónicos tales como las tarjetas inteligentes.

En el Departamento de Tarjetas Inteligentes perteneciente al Centro de Identificación y Seguridad Digital (CISED), la principal línea de investigación y desarrollo está dirigida a potenciar la creación de soluciones informáticas que hagan uso de tarjetas inteligentes, entre los productos del CISED se desarrolló un *applet*[2] y su *middleware*[3], los cuales permiten gestionar la información del documento licencia de conducción en tarjetas inteligentes cumpliendo con estándares definidos para esta tecnología.

El documento recoge toda la información resultante de la investigación, presentándose características de otros sistemas actualmente implementados o en desarrollo, así como la descripción del diseño y la arquitectura, además se describen los estándares que guiaron el desarrollo del sistema y tecnologías utilizadas.

PALABRAS CLAVE:

Tarjetas inteligentes, *Middleware*, *Applet*, Licencia de Conducción.

[1] Licencia de conducción: Este documento es nombrado en algunos países como licencia de conducción, permiso o carné de conducir, para referirse al mismo en el presente trabajo se nombra como “licencia de conducción”.
[2] Applet: Programa informático (aplicación) realizado en lenguaje *JavaCard* que corre sobre el sistema operativo del chip de una tarjeta inteligente.
[3] Middleware: Programa informático que permite la comunicación con el *applet*.

Índice de contenido

INTRODUCCIÓN **1**

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA **5**

INTRODUCCIÓN **5**

1.1 TARJETAS INTELIGENTES 5

1.1.1 Marco conceptual *5*

1.1.2 Estructura de una tarjeta inteligente *8*

1.1.3 Algunas clasificaciones de las tarjetas inteligentes *9*

1.1.4 Estándar ISO/IEC 7816-4 ..... *11*

1.1.5 Criptografía en tarjetas inteligentes *13*

1.1.6 Estándar PCSC *15*

1.2 LICENCIA DE CONDUCCIÓN EN TARJETAS INTELIGENTES 16

1.2.1 Soluciones similares *16*

1.2.2 Estándar ISO/IEC 18013 ..... *18*

1.3 TECNOLOGÍAS DE DESARROLLO 20

1.3.1 Selección de la metodología de desarrollo *21*

1.3.2 Lenguaje Unificado de Modelado (UML) *24*

1.3.3 Herramienta Altova UModel *24*

1.3.4 Herramienta Developer Suite *25*

1.3.5 Lenguaje de programación JavaCard *25*

1.3.6 Herramienta NetBeans *27*

1.3.7 Lenguaje Java *27*

CONCLUSIONES DEL CAPÍTULO **27**

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN **29**

INTRODUCCIÓN **29**

2.2 MODELO DE DOMINIO 29

2.2.1 Glosarios de términos del modelo de dominio *30*

2.3 FASES DE UN PROYECTO CON XP 31

2.3.1 Historias de Usuario *32*
2.3.2 Requisitos no Funcionales *35*
2.3.3 Metáfora *36*
2.3.4 Arquitectura *37*
2.3.5 Patrones Arquitectónicos y de diseño *38*
2.3.6 Estimación de Tiempo *39*
2.3.6 Plan de Iteraciones *39*
2.3.7 Tareas de Ingeniería *40*
2.3.8 Plan de Entrega *41*

CONCLUSIONES DEL CAPÍTULO **41**

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DE LA SOLUCIÓN **43**

INTRODUCCIÓN **43**

3.1 FASE DE ITERACIÓN 43
3.1.1 Diseño *43*
3.1.2 Implementación del applet y su middleware *45*
3.2 PUESTA EN PRODUCCIÓN 49
3.2.2 Pruebas unitarias *49*
3.2.1 Pruebas de aceptación *56*

CONCLUSIONES DEL CAPÍTULO **60**

CONCLUSIONES GENERALES **61**

RECOMENDACIONES ..... ¡ERROR! MARCADOR NO DEFINIDO.

BIBLIOGRAFÍA CITADA **63**

BIBLIOGRAFÍA CONSULTADA **65**

GLOSARIO DE TÉRMINOS **67**

ANEXOS **68**

Índice de tablas

Tabla 1: Estructura C-APDU.[3] 7
Tabla 2: Estructura R-APDU.[3] 8
Tabla 3: Descripción de los AIDs.[6] 26
Tabla 4: HU1-Establecer comunicación. (Creación de la autora) 33
Tabla 5: HU2-Almacenar llave semilla. (Creación de la autora) 33
Tabla 6: HU3-Gestionar el sistema de ficheros. (Creación de la autora) 34
Tabla 7: HU4-Efectuar BAP. (Creación de la autora) 34
Tabla 8: HU5-Leer grupos de datos. (Creación de la autora) 34
Tabla 9: HU6-Realizar autenticación pasiva. (Creación de la autora) 35
Tabla 10: HU7-Realizar autenticación activa. (Creación de la autora) 35
Tabla 11: Estimación de tiempo. (Creación de la autora) 39
Tabla 12: Plan de iteraciones. (Creación de la autora) 40
Tabla 13: Tareas de ingeniería de la iteración 1. (Creación de la autora) 40
Tabla 14: Tareas de ingeniería de la iteración 2. (Creación de la autora) 41
Tabla 15: HU1_T1-Establecer comunicación. (Creación de la autora) 41
Tabla 16: Plan de entrega. (Creación de la autora) 41
Tabla 17: Ficheros en el chip. (Creación de la autora) 46
Tabla 18: HU1_CP1-Establecer comunicación. (Creación de la autora) 57
Tabla 19: HU2_CP2-Almacenar llave semilla. (Creación de la autora) 58
Tabla 20: HU3_CP3-Gestionar el sistema de ficheros. (Creación de la autora) 59
Tabla 21: CRC-LicenseApplet. (Creación de la autora) 70
Tabla 22: CRC-KyeStore. (Creación de la autora) 70
Tabla 23: CRC-LicenseCrypto. (Creación de la autora) 70
Tabla 24: CRC-FileSystem. (Creación de la autora) 70

Tabla 25: CRC-CVCertificate. (Creación de la autora) 71
Tabla 26: CRC-DrivingLicense. (Creación de la autora) 71
Tabla 27: CRC-DrivingDemographicInfo. (Creación de la autora) 71
Tabla 28: CRC-SecureMessagingWrapper. (Creación de la autora) 71
Tabla 29: CRC-DataGroup. (Creación de la autora) 71
Tabla 30: CRC-DrivingLicenseFile. (Creación de la autora) 72
Tabla 31: CRC-CategoryInfo. (Creación de la autora) 72
Tabla 32: CRC-DrivingLicensePersoService. (Creación de la autora) 72
Tabla 33: HU2_T2-Almacenar llave semilla. (Creación de la autora) 72
Tabla 34: HU3_T3-Crear un EF.SOD para la autenticidad e integridad de los datos. (Creación de la autora) 73
Tabla 35: HU3_T4-Definir los grupos de datos obligatorios y opcionales. (Creación de la autora) 73
Tabla 36: HU3_T5-Almacenar información en el fichero. (Creación de la autora) 74
Tabla 37: HU4_T6-Efectuar BAP. (Creación de la autora) 74
Tabla 38: HU5_T7-Leer grupos de datos. (Creación de la autora) 74
Tabla 39: HU6_T8-Realizar autenticación pasiva. (Creación de la autora) 75
Tabla 40: HU7_T9-Realizar autenticación activa. (Creación de la autora) 75
Tabla 41: HU4_CP4-Efectuar BAP. (Creación de la autora) 76
Tabla 42: HU5_CP5-Leer grupos de datos. (Creación de la autora) 76
Tabla 43: HU6_CP6-Realizar autenticación pasiva. (Creación de la autora) 77
Tabla 44: HU7_CP7-Realizar autenticación activa. (Creación de la autora) 78

Índice de figuras

Figura 1: Diagrama de comunicación Middleware-Applet. (Creación de la autora).....6
Figura 2: Arquitectura básica de un micro-controlador.[9].....8
Figura 3: Tipos de tarjetas según su formato. [10].....11
Figura 4: Jerarquía de ficheros.[3].....12
Figura 5: Ficheros en paralelo.[3].....13
Figura 6: Estructura de los EF.[3].....13
Figura 7: Modelo de dominio. (Creación de la autora).....30
Figura 8: Arquitectura del sistema. (Creación de la autora).....37
Figura 9: Diagrama de Despliegue. (Creación de la autora).....44
Figura 10: Diagrama de estructura de ficheros.[7].....45
Figura 11: Diagrama de estados del ciclo de vida del applet. (Creación de la autora).....48
Figura 12: Diagrama de capas de comunicación. (Creación de la autora).....49
Figura 13: Código que permite seleccionar fichero - applet. (Creación de la autora).....50
Figura 14: Caso de prueba seleccionar fichero - applet. (Creación de la autora).....51
Figura 15: Código que permite enviar seleccionar fichero - middleware. (Creación de la autora).....54
Figura 16: Caso de prueba seleccionar fichero - middleware. (Creación de la autora).....54
Figura 17: Diagrama de clases del applet. (Creación de la autora).....68
Figura 18: Diagrama de Clases del Middleware. (Creación de la autora).....69

Introducción

A lo largo de la historia de la humanidad, la utilización de los diversos medios de transporte ha representado un papel fundamental en la sociedad, permitiendo el traslado de personas, bienes o mercancías e incluso el intercambio de diversos elementos culturales.

Con el devenir de la sociedad moderna, el desarrollo de la industria automovilística hizo posible el uso de estos medios a gran escala, debido a ello se hizo necesario crear una serie de medidas, regulaciones y precauciones que las personas deben tener en cuenta para mantener el orden y la seguridad vial, por lo que se establece una ley de vialidad y tránsito, el dominio de este permite la adquisición de una licencia o permiso de conducción, que acredita la capacidad del portador para operar libremente un vehículo por la vía pública.

La licencia de conducción contiene como mínimo los datos fundamentales del vehículo y su titular: nombre completo del conductor, número de identificación, fecha de expedición, organismo que la expidió, así como los permisos que posee para conducir un determinado vehículo, este documento es de un valor legal tal, que en algunos países donde no se emite una cédula de identificación personal puede ser usado para este propósito.[1] Sin embargo, siempre ha estado vigente la constante necesidad de mejorar la seguridad de documentos oficiales de identificación, coherente con este propósito se han desarrollado nuevas tecnologías como las tarjetas inteligentes para aminorar la falsificación y alteración de la legitimidad de documentos de este tipo.

Las tarjetas inteligentes entre otras cosas proveen seguridad en materia de identificación y gestión de datos. La denominación de inteligentes se debe a que poseen un módulo electrónico embebido que actúa como ordenador o microprocesador, además contienen una memoria donde pueden almacenar datos y programas como un sistema operativo, software de comunicación y algoritmos de cifrado que pueden proteger estos contenidos haciéndolos inaccesibles ante cualquier intento de acceso no autorizado, así como el uso de contraseñas y la autenticación biométrica para disminuir el riesgo de suplantación de identidad.[2]

En la actualidad, esta tecnología se ha expandido por todos los continentes, lográndose significativos adelantos en los países que la emiten, por la seguridad tanto física como lógica que brindan y la posibilidad de integrar múltiples aplicaciones en una misma tarjeta.

Con el objetivo de lograr gran aceptación y uso de las tarjetas inteligentes, se han definido estándares en busca de compatibilidad entre diferentes tecnologías de software y hardware tales como ISO/IEC 7816.[3]

Este estándar consiste en una serie de normas que definen las características físicas (ancho, largo, grosor, peso, etc.) y otras relativas a los protocolos de comunicación con el software interno de la tarjeta (comandos básicos de protocolos a distintos niveles).

En el Departamento de Tarjetas Inteligentes perteneciente al CISED se investigan las posibles áreas en las cuales se puedan introducir las tarjetas inteligentes y se desarrollan aplicaciones que hagan uso de estas. Con relación a lo antes planteado, conforma su cantera de productos el desarrollo de un *applet* y su *middleware* que permitan gestionar la información del documento licencia de conducción en tarjetas inteligentes, cumpliendo con estándares establecidos para estas tecnologías.

Teniendo en cuenta lo antes expuesto surge la siguiente **situación problémica:**

Las licencias de conducción normalmente son emitidas sobre papel laminado o plástico, lo que posibilita que presenten algunas dificultades, entre las que se encuentran:

- Pueden ser fácilmente falsificadas, debido a que las medidas de seguridad físicas pueden ser suplantadas según su grado mediante fraudes.
- Tienen capacidad limitada de datos a incluir, pues están determinados por el tamaño físico del documento.

Luego de haber analizado la situación existente, teniendo en cuenta las dificultades expuestas y la necesidad de buscar una solución a ellas, todos los esfuerzos estarán encaminados a solucionar el siguiente **problema científico:** ¿Cómo realizar la gestión de la información del documento licencia de conducción en tarjetas inteligentes?

Para ello el **objeto de estudio** es el proceso de gestión de la información contenida en tarjetas inteligentes y el **campo de acción** se enmarca en la gestión de la información del documento licencia de conducción en tarjetas inteligentes.

La investigación se sustenta en la siguiente **idea a defender**: El desarrollo de un *applet* y su *middleware* para la licencia de conducción, permitirá gestionar la información de este tipo de documento en tarjetas inteligentes.

Siendo el **objetivo general** de la investigación: Desarrollar un *applet* y su *middleware*, para gestionar la información del documento licencia de conducción en tarjetas inteligentes.

Para el correcto desarrollo del presente trabajo se plantea el cumplimiento de las siguientes **tareas de la investigación**:

- Determinación de aspectos teóricos conceptuales sobre *applet* y *middleware.*

- Análisis de estándares y normas establecidas para licencias de conducción en tarjetas inteligentes.
- Análisis valorativo de las soluciones existentes de licencia de conducción haciendo uso de tarjetas inteligentes.
- Análisis de las tecnologías para realizar la implementación.
- Elaboración de la documentación y diagramas de ingeniería de software necesarios para el análisis y diseño de la solución.
- Implementación del *applet* haciendo uso de estándares y normas establecidas para esta tecnología.
- Implementación del componente *middleware* para establecer una comunicación con las funcionalidades del *applet* embebido en la tarjeta.
- Realización de pruebas de calidad a la solución.

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

Del nivel teórico:

- **Método histórico:** permitió consultar bibliografía referente al tema de investigación, toda su trayectoria, evolución y comportamiento.
- **Método lógico:** condujo a estructurar la documentación investigada de una manera organizada y cronológica, para así tener un mejor entendimiento de la misma.
- **Método de la modelación:** facilitó la creación de modelos, representando de manera gráfica parte del contenido de la investigación.

Del nivel empírico:

- **Método de la observación:** se puso en práctica, al concebir de forma consciente la planificación de la investigación, orientada hacia el logro del objetivo determinado.
- **Método experimental:** el experimento permitió estudiar el objeto, crear las condiciones para verificar la idea a defender.

Apoyo sobre los procesos de:

- **Análisis:** permitió la división mental de lo investigado, en relaciones y componentes para una mejor comprensión.
- **Síntesis:** posibilitó establecer mentalmente la unión entre las partes previamente analizadas, resaltando sus principales características, conceptos y relaciones.

Con la investigación se pretende obtener como **posible resultado** el análisis, diseño, implementación y pruebas de la solución, para gestionar la información del documento licencia de conducción en tarjetas inteligentes, haciendo uso de estándares y normas establecidas para esta tecnología.

Estructuración de capítulos

Capítulo 1 Fundamentación Teórica: En este capítulo se identifican y analizan aplicaciones que gestionan la información de licencia de conducción en tarjetas inteligentes. Se profundiza en el estudio de las técnicas, tendencias tecnológicas y la metodología utilizada en la cual se apoya la solución del problema.

Capítulo 2 Propuesta de la Solución: Se identifican las historias de usuario y los requerimientos no funcionales, se realiza una estimación de tiempo y plan de iteraciones, se describen las tareas de ingenierías derivadas de las historias de usuario, la arquitectura del sistema y se realiza un plan de entrega.

Capítulo 3 Implementación y Pruebas de la Solución: Se muestra un breve diseño mediante la descripción de las tarjetas CRC[4] y su representación gráfica en diagramas de clases. Se presentan algunas características del *applet* y su *middleware* y finalmente se les realizan las pruebas de calidad correspondientes a la solución.

[4] Tarjetas CRC: (Clase-Responsabilidad-Colaboración) Tarjeta que representa una clase diferente en la codificación.

Capítulo 1: Fundamentación teórica

Introducción

Las tarjetas inteligentes han tenido gran aceptación a nivel mundial por parte de las personas y organizaciones que hacen uso de ellas, ofrecen sencillez, portabilidad, autenticación y seguridad, potenciando en gran medida la gestión de datos, identidad, soporte al cliente y comunicaciones. En nuestros días esto ha motivado la idea de adoptar las tarjetas inteligentes como un medio seguro para gestionar la información del documento licencia de conducción.

El primer sistema creado en el mundo de este tipo fue implementado en 1987 en Turquía, debido a los altos índices de accidentes en las carreteras en este país, se decidió instalar un Tacógrafo Digital[5] en vehículos de carga pesada, para reducir las violaciones por exceso de velocidad. La tarjeta inteligente requiere ser insertada por el conductor en el Tacógrafo antes de comenzar a manejar, de esta manera el dispositivo registra las violaciones por exceso de velocidad y las horas de viaje para cada conductor emitiendo luego un reporte impreso.[4]

De forma similar en 1995 se comenzaron a utilizar las licencias de conducción electrónicas en Argentina, específicamente la provincia de Mendoza, donde existían altos índices de accidentes, violaciones del código vial por parte de los conductores, además de un registro muy pobre de las infracciones cometidas, estas tarjetas permitieron tener actualizado un reporte de dichas violaciones y las multas no pagadas. [4]

1.1 Tarjetas inteligentes

El término de “tarjeta inteligente” se usa de forma genérica para definir una serie de circuitos, cada uno de los cuales pueden transportar datos en un dispositivo de reducidas dimensiones físicas, de forma que pueden ser fácilmente llevados en un bolsillo o en la cartera.[5]

Teniendo en cuenta lo antes planteado, las tarjetas inteligentes son dispositivos electrónicos que permiten almacenar y gestionar información proveyendo portabilidad y seguridad en cada una de sus aplicaciones, más conocidas por *applets*.

1.1.1 Marco conceptual

Las aplicaciones que se ejecutan en el sistema operativo de las tarjetas inteligentes son pequeños programas denominados *applets*, estas aplicaciones sólo procesan los comandos APDU (*Application*

[5] Tacógrafo Digital: Dispositivo electrónico de registro de eventos en la conducción de vehículos.

Protocol Data Unit) recibidos del *middleware* a través del dispositivo lector y responden a estos con APDU de respuesta. El *middleware* se ejecuta en el equipo cliente o Terminal de Servicio[6].

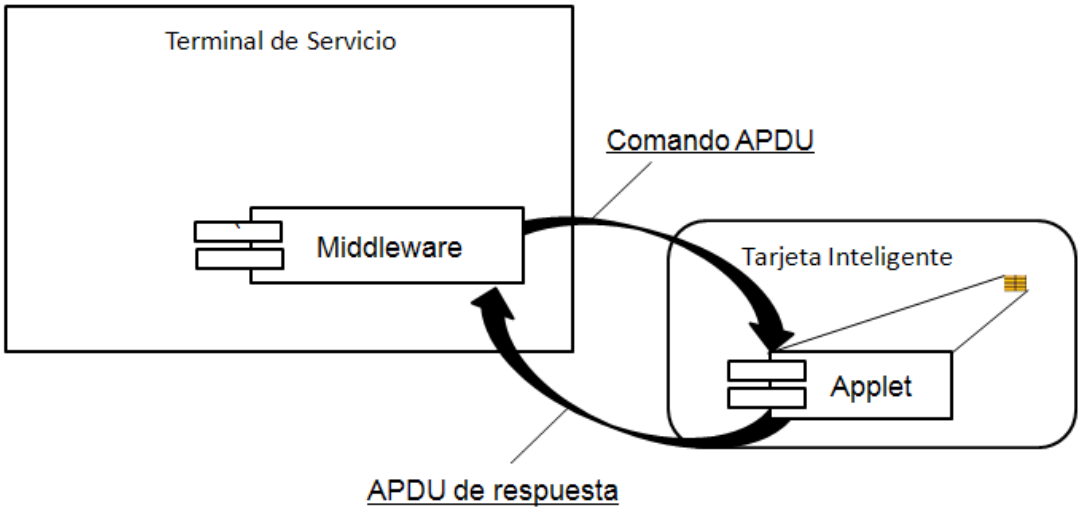

Figura 1: Diagrama de comunicación Middleware-Applet. (Creación de la autora)

Applet

Un *applet* es una aplicación de tarjeta inteligente escrito en lenguaje de programación *Java Card* el cual es un subconjunto del lenguaje *Java*. [6] Entre otras particularidades, tiene la enorme ventaja de ser un programa que se puede ejecutar directamente desde la tarjeta inteligente en la que está instalada, permite disímiles aplicaciones, desde la introducción de datos a la ejecución de las operaciones que se manejan dentro de la tarjeta inteligente.[7]

Middleware

El *middleware* puede ser considerado como un software de conectividad que permite la interconexión entre diferentes aplicaciones, el acceso a las funcionalidades y a los datos de estas, desde y a través de otros sistemas independientemente de la plataforma.[8]

El *middleware asociado* al *applet* que corre sobre el sistema operativo del chip embebido en la tarjeta inteligente se ejecuta en la terminal de servicio y funciona como una capa de software intermedio, que se intercala entre las aplicaciones del usuario y la tarjeta.

[6] Terminal de Servicio: Es un punto en una computadora donde se instalan todas las condiciones para poder interactuar con la solución.

Protocolo de comunicación con tarjetas inteligentes

La comunicación entre el *applet* y el *middleware* se realiza mediante comandos APDU, donde toda comunicación que se realice con una tarjeta es iniciada por el exterior, lo que indica que desde una tarjeta nunca se transmite información sin que antes se haya producido una petición externa por el *middleware* a través del dispositivo lector. Para identificar el APDU que recibe la tarjeta y el que envía como respuesta se denominan Comando APDU (C-APDU), usado por el *middleware* para enviar información al *applet* y APDU de Respuesta (R-APDU), usado por el *applet* para responder el comando enviado por el *middleware*.

Una vez que la tarjeta recibe el APDU, lo procesa y devuelve a la aplicación externa una notificación que indica si el proceso concluyó o si ocurrió un error al procesar el APDU recibido. La estructura general del C-APDU lo conforman siete elementos, sin embargo no siempre se envían los siete elementos, pueden ser enviados los primeros cuatro ya que los datos (*Data Field*) son opcionales debido a que no siempre la tarjeta necesita información del exterior en cada APDU que recibe. La siguiente tabla muestra la estructura de un C-APDU como lo define el estándar ISO/IEC 7816-4.[3]

Encabezado (Obligatorio)				**Cuerpo (Opcional)**		
CLA	INS	P1	P2	Lc	*Data Field*	Le

Tabla 1: Estructura C-APDU.[3]

Descripción del C-APDU

CLA: Clase de instrucción, indica la clase y la estructura.

INS: Código de instrucción, especifica la instrucción del comando.

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

Lc: Número de *bytes* en el *Data Field* del APDU.

Data Field: Secuencia de *bytes* con información.

Le: Cantidad máxima de *bytes* esperados como respuesta.

Como respuesta a un C-APDU recibido, la tarjeta inteligente devuelve a la aplicación externa el R-APDU que se conforma por tres elementos: los datos y dos palabras de estado (*Status Word*). Para el caso del R-APDU el *Data Field* es opcional, depende si la tarjeta enviará información al exterior, sin embargo el

Status Word 1 y 2 (SW1 y SW2 respectivamente) siempre son enviados al exterior. La siguiente tabla muestra la estructura de un R-APDU como lo define el estándar ISO/IEC 7816-4.[3]

Cuerpo (Opcional)	Código Respuesta (Obligatorio)	
Data Field	SW1	SW2

Tabla 2: Estructura R-APDU.[3]

Descripción del R-APDU

Data Field: Secuencia de *bytes* con información.
SW1, SW2: Denotan el estado del procesamiento del comando en la tarjeta.[3]

1.1.2 Estructura de una tarjeta inteligente

Las tarjetas inteligentes poseen un chip con su CPU (*Central Processing Unit*), ROM (*Read Only Memory*) y RAM (*Random Access Memory*). Su forma de almacenamiento puede ser EPROM (*Erasable Programmable Read Only Memory*) o EEPROM (*Electrcial Erasable Programmable Read Only Memory*). El sendero interno de comunicación entre los elementos (BUS) es totalmente inaccesible desde afuera del chip de silicio mismo, por ello la única manera de comunicarse con las tarjetas está totalmente bajo control del sistema operativo y no hay manera de poder introducir comandos falsos o requerimientos inválidos que puedan sorprender las políticas de seguridad.[9]

La figura que se presenta a continuación muestra los componentes internos de un micro-controlador.

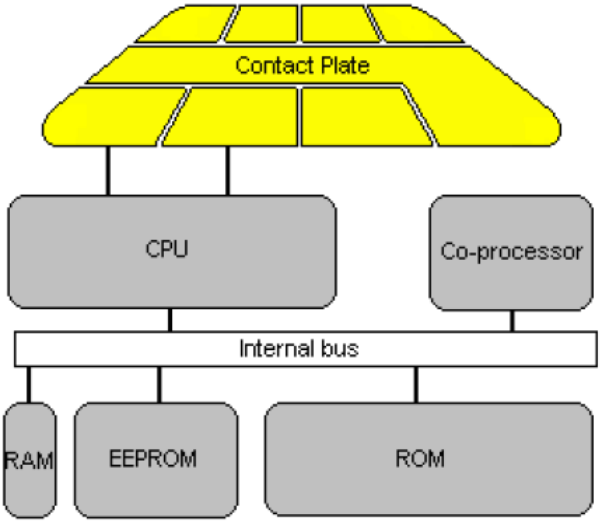

Figura 2: Arquitectura básica de un micro-controlador.[9]

Componentes internos de un micro-controlador:

- **CPU:** Procesador de la tarjeta, pueden tener opcionalmente módulos hardware para operaciones criptográficas.[9]
- **ROM:** Memoria interna en la que se establece el sistema operativo de la tarjeta, las rutinas del protocolo de comunicaciones y los algoritmos de seguridad de alto nivel por software, como su nombre indica no se puede reescribir y se inicializa durante el proceso de fabricación. No es necesaria la energía eléctrica para retener la información contenida en la misma, esta no puede ser modificada durante todo el tiempo de vida del chip.[9]
- **EEPROM:** Memoria de almacenamiento (equivalente al disco duro en un ordenador personal), es mutable y persistente, o sea, que puede ser modificada mientras la tarjeta está expuesta a la energía eléctrica y persistir una vez que la fuente es retirada, en ella está grabado el sistema de ficheros, los datos usados por las aplicaciones, llaves de seguridad y las propias aplicaciones que se ejecutan en la tarjeta.[9]
- **RAM:** Memoria volátil con que trabaja el procesador, su contenido es mutable y no se preserva una vez que se le retira la energía a la tarjeta.[9]

Las tarjetas inteligentes pueden ser clasificadas de acuerdo a las diferentes propiedades y características que presentan. A continuación se clasificarán de acuerdo a su capacidad, la interfaz y su formato.

1.1.3 Algunas clasificaciones de las tarjetas inteligentes

Tipos de tarjetas según su capacidad

De acuerdo con la capacidad para el almacenamiento y procesamiento de datos que tiene la tarjeta inteligente, pueden ser de memoria, micro-procesadas o criptográficas, como se describen a continuación:

- **Tarjetas de memoria:** Son únicamente un contenedor de ficheros, pero no almacenan aplicaciones ejecutables, son utilizadas generalmente en aplicaciones de identificación y control de acceso sin altos requisitos de seguridad.[9]
- **Tarjetas micro-procesadas:** Contienen una estructura análoga a la de un ordenador, con elementos de memoria volátil, memoria persistente y un procesador con un sistema operativo que permite el almacenamiento, procesamiento y encriptación de la información que guardan, estas tarjetas suelen usarse para identificación y pago con monederos electrónicos. [9]
- **Tarjetas criptográficas (micro-procesadoras avanzadas):** Contienen módulos de hardware para la ejecución de complejos algoritmos asociados con operaciones criptográficas.[9]

Tipos de tarjetas según la interfaz

De acuerdo con la interfaz de comunicación pueden ser clasificadas en tarjetas de contacto, tarjetas sin contacto, híbridas y duales.

- **Tarjetas de contacto:** Requieren ser insertadas en un terminal con lector para que puedan ser leídas, sus características físicas están especificadas en el estándar ISO/IEC 7816, donde se define la estructura de un conjunto de 8 pines, a través de los cuales se establece la comunicación con el lector.[9]
- **Tarjetas sin contacto:** Utilizan diferentes protocolos de transmisión en capa lógica y física, no utiliza contacto galvánico, sino que establecen comunicación a través de campos electromagnéticos usando una antena, lo que permite que se use desde media distancia sin necesidad de ser introducida en un lector. El estándar de comunicación de tarjetas inteligentes sin contacto es el ISO/IEC 14443, el mismo incluye las especificaciones de las características físicas, la energía de radiofrecuencia y la señal de interfaz, protocolos de inicialización y anticolisión y el protocolo de transmisión.[9]
- **Tarjetas híbridas:** Son tarjetas sin contacto a las cuales se les agregan un segundo chip de contacto, ambos chips pueden ser o micro-procesadores o simples chips de memoria. El chip sin contacto es generalmente usado en aplicaciones que requieran transacciones rápidas, como por ejemplo el transporte, mientras que el de contacto es generalmente utilizado en aplicaciones que requieren de alta seguridad como las bancarias.[9]
- **Tarjetas Duales:** Son similares a las tarjetas híbridas en que presentan ambas interfaces, con y sin contacto, la diferencia más importante es que este tipo de tarjeta tiene un solo circuito integrado.[9]

Tipos de tarjetas según su formato

Las tarjetas inteligentes pueden ser clasificadas en formato ID-000, ID-00 e ID-1. Como se muestra en la siguiente figura.

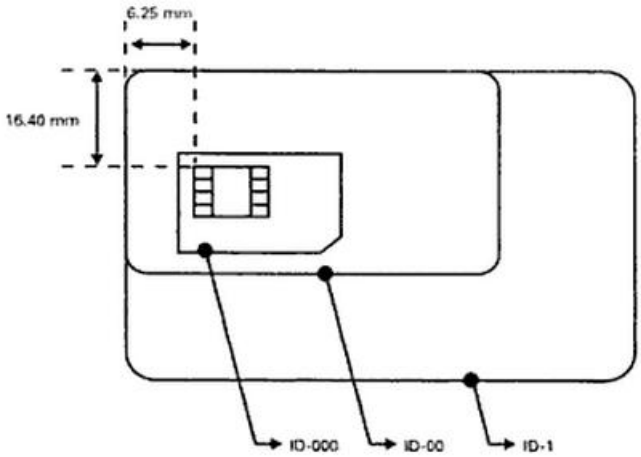

Figura 3: Tipos de tarjetas según su formato. [10]

- **Formato ID-000:** Se usa mayoritariamente en teléfonos de tecnología GSM[7]. El corte rectangular en la base, lado derecho, proporciona una mayor facilidad a la hora de insertar la tarjeta en la ranura del dispositivo lector.[10]
- **Formato ID-00:** Mini-tarjeta, cuyas dimensiones están a medio camino entre el formato ID-000 y el ID-1, este formato proporciona al portador de la tarjeta una mayor facilidad en su manejo y el coste de producción es menor.[10]
- **Formato ID-1:** Proporciona al usuario bastante comodidad en su manejo, de tal manera que la tarjeta no sea demasiado larga para llevarla en la cartera ni demasiado pequeña para que pueda perderse con facilidad.[10]

1.1.4 Estándar ISO/IEC 7816-4

La especificación ISO/IEC 7816 parte 4 define: [3]

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

[7] GSM*: Global System for Mobile Communication* por sus siglas en inglés. Es un servicio ofrecido por las empresas operadoras de telefonía móvil que permite determinar, con una cierta precisión, la posición donde se encuentra físicamente un terminal móvil determinado.

- Arquitectura de seguridad para derechos de acceso a los archivos y datos en la tarjeta.
- Comandos orientados a objetos de datos.
- Métodos de acceso a los algoritmos que procesa la tarjeta inteligente (no describe los algoritmos).
- Métodos para el intercambio seguro de mensajes.

Estructura para aplicaciones y datos

La estructura de fichero estará definida en la ISO/IEC 7816-4 para los datos y aplicaciones. El sistema de ficheros de la tarjeta estará organizado jerárquicamente en ficheros dedicados (DF) y ficheros elementales (EF), los DF contienen EF u otros DF. También esta estructura contará de manera opcional (determinado por el sistema operativo seleccionado) con un fichero maestro (MF) que puede ser la raíz del sistema de ficheros.[3]

DF: hospeda aplicaciones y/o grupos de ficheros y/o objetos de datos.

EF: almacena datos y no puede ser el padre de otro fichero.

Existen dos categorías: EF internos, que almacenan información interpretada por la tarjeta, por ejemplo la usada por la tarjeta con propósito de administración y control, por otra parte los EF de trabajo, que almacenan datos no interpretados por la tarjeta, como por ejemplo datos usados fuera de la tarjeta exclusivamente. [3]

La siguiente figura muestra la jerarquía de ficheros.

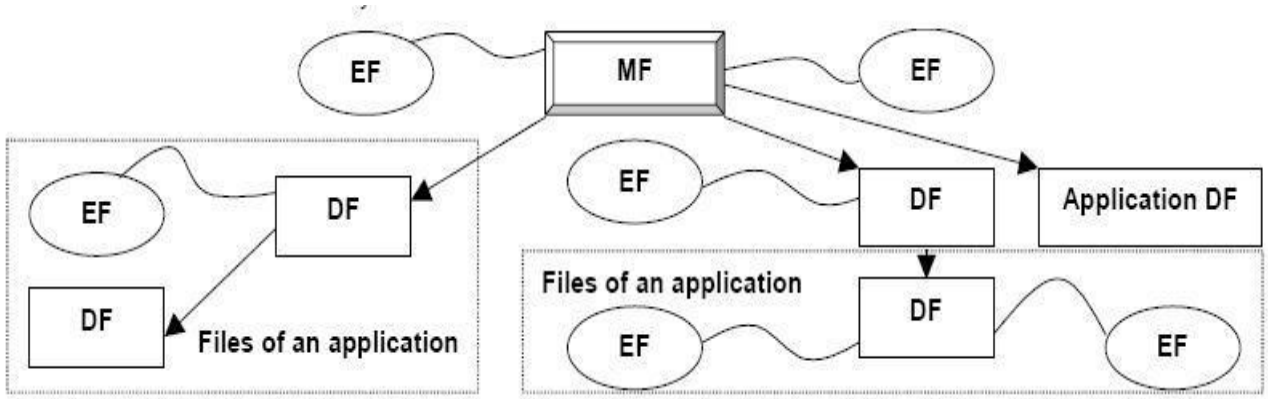

Figura 4: Jerarquía de ficheros.[3]

La siguiente figura muestra DF en paralelo sin una MF y sin una jerarquía entre ellos. Esta organización soporta que cada DF tenga su propia jerarquía con su correspondiente arquitectura.[3]

Figura 5: Ficheros en paralelo.[3]

Estructura de ficheros

Se definen tres tipos de estructura de los EF.

- **Estructura Transparente:** El EF es visto como una única secuencia continua de datos accesibles por comandos para el manejo de las unidades de datos, el tamaño de estas depende del EF. [3]
- **Estructura de Registros:** El EF es visto como una única secuencia continua de registros de identificación individual accesibles por comandos de manejos de registro. [3]
    - El tamaño de los atributos puede ser fijo o variable.
    - La organización de los registros puede ser una estructura lineal o una estructura cíclica.
- **Estructura TLV (etiqueta, longitud, valor):** El EF es visto como un conjunto de objetos de datos accesibles por comandos para el manejo de estos.[3]

Para la referencia de los datos en el EF, la tarjeta soporta al menos una de las cinco estructuras que se muestran a continuación.

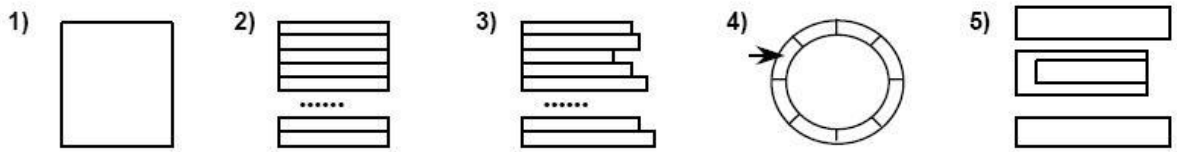

Figura 6: Estructura de los EF.[3]

1) Estructura transparente.
2) Estructura lineal con registros de tamaño fijo.
3) Estructura lineal con registros de tamaño variable.
4) Estructura cíclica con registros de tamaño fijo donde la flecha referencia al último registro escrito.
5) Estructura TLV.[3]

1.1.5 Criptografía en tarjetas inteligentes

Dependiendo del tipo de tarjeta y su fabricante varía el número de operaciones criptográficas que las tarjetas realizan, entre las que pueden tener o llegar a tener las tarjetas inteligentes se encuentran:

- Generación de llave DES[8] o 3DES[9].
- Cifrado DES o 3DES.
- Generación de llaves RSA[10] de distintas longitudes.
- Encriptación con llave pública o privada.
- Funciones *hash*[11] (MD5, SHA, y otras).
- Firma digital y su verificación.
- Cifrado y descifrado de envoltorios digitales (cifrar los datos con DES y cifrar la llave DES con RSA).[3]

El estándar ISO 7816-8 describe los comandos inter-industria relacionados con la seguridad, su entorno de seguridad describe los componentes MAC[12], *hash*, firma digital, confidencialidad y autenticación. A continuación se analizan brevemente cada uno de ellos.

- **MAC:** Los códigos detectores de errores son asignados a los datos que se desea proteger, para detectar o corregir cualquier alteración de los mismos según el código utilizado. Uno de estos códigos es el XOR, es muy fácil y rápido de calcular, detecta todos los errores de números impares de bits, así como la mayoría de los errores de números pares de bits.[3]
- **HASH:** Las funciones hash más habituales en el campo de las tarjetas inteligentes son SHA y MD5. [11] Estas funciones permiten generar un valor *hash*, mediante el cual se puede comprobar si el mensaje (o texto) ha sido modificado.[3]
- **Firma digital:** Las operaciones con firma digital son utilizadas para firmar cualquier dato arbitrario externo usando una llave privada interna, la verificación de la firma digital generada externa es realizada conociendo el método de firma digital utilizada y la llave pública del emisor. Las firmas digitales pueden ser basadas en algoritmos RSA o Criptografía de Curva Elíptica. Las tarjetas inteligentes almacenarán tanto la llave privada como la pública, la llave privada no debe ser accesible, solamente se utilizará para operaciones internas.[3]
- **Confidencialidad o intercambio de llaves:** Son usadas habitualmente en el intercambio simétrico de llaves de sesión entre dos entidades. En el estándar PC/SC se contempla las posibilidades de intercambio de llaves usando RSA o algoritmos de Diffie-Helman.[3]

[8] DES: *Data Encryption Standard*, estándar de cifrado de datos. Es un algoritmo de cifrado simétrico que utiliza una misma clave para cifrar y descifrar los datos.
[9] 3DES: Consiste en una triple encriptación DES.
[10] RSA: Es un algoritmo de cifrado asimétrico que utiliza un par de claves, pública y privada para cifrar y descifrar los datos.
[11] Hash: *Secure Hash Algorith*, algoritmo hash seguro. Un *hash e*s el resultado de una función o algoritmo que sirve para garantizar la integridad de los textos
[12] MAC: Mesage Authentication Code por sus siglas en inglés.

- **Autenticación:** El protocolo de autenticación entre la tarjeta inteligente y el lector de tarjetas está basado en llave pública usando firmas digitales con RSA u otros. La selección entre estos dos algoritmos de firma está basado en el algoritmo que soporte la tarjeta inteligente, para realizar la autenticación a la tarjeta el cliente debe poseer una llave privada y un certificado[13] que contiene la llave pública correspondiente, la llave privada se guardará dentro de la tarjeta inteligente. El certificado puede guardarse de manera conveniente, este mecanismo puede ser utilizado invirtiendo los papeles de las dos partes para autenticar al cliente.[3]

1.1.6 Estándar PKCS

En criptografía, se refiere a un grupo de estándares de criptografía de clave pública, concebidos y publicados por los laboratorios de RSA en California en cooperación con desarrolladores de sistemas de seguridad en todo el mundo con el fin de acelerar el despliegue de la criptografía de clave pública. A RSA Security se le asignaron los derechos de licenciamiento para la patente de algoritmo de clave asimétrica RSA y adquirió los derechos de licenciamiento para muchas otras patentes de claves. Publicado por primera vez en 1991 como resultado de las reuniones con un pequeño grupo de los primeros adoptantes de tecnologías de clave pública, los documentos PKCS se han convertido en referencia y ampliamente aplicados. Contribuciones de la serie PKCS se han convertido en parte de notas académicas y normas, incluyendo ANSI X9 documentos, PKIX, SET, S / MIME y SSL.

Los estándares PKCS van desde el número 1 hasta el número 15, pero se realizará un resumen de los utilizados en el desarrollo de la solución.

- **PKCS-7:** Es el estándar de la sintaxis de los mensajes criptográficos, define la sintaxis de varios tipos de mensajes criptográficos protegidos, incluyendo mensajes encriptados y mensajes con firmas digitales. Originalmente la Internet Privacy-Enhanced Mail convirtió a este estándar en especificación segura de correos electrónicos. Pero este no solo fue limitado al correo, sino que se convirtió en la base de los mensajes seguros en sistemas tan diversos como el *Secure Electronic Transaction* (SET) especificado para transacciones bancarias utilizando pagos por tarjetas.
- **PKCS-10:** Formato de los mensajes enviados a una Autoridad de certificación para solicitar la certificación de una clave pública. Describe la sintaxis de los pedidos de certificados. Un pedido de certificado consiste de un nombre distinguido, una llave pública y opcionalmente un conjunto de

[13] Certificado: Es un documento firmado por una autoridad certificadora. De manera habitual se suele usar el certificado estándar x.509.

atributos, todos firmados por una entidad de pedidos de certificación. los pedidos de certificados son enviados a una autoridad certificadora, la cual transforma el pedido en un certificado X509.[12]

1.1.7 Estándar PC/SC

El grupo de trabajo PC/SC[14], fue creado para desarrollar una especificación que facilite la interoperabilidad necesaria para que las tarjetas de circuito integrado puedan ser utilizadas eficazmente en entornos de computadoras, se centra fundamentalmente en el desarrollo de especificaciones a nivel de software y dispositivos, además se basan en los estándares ISO/IEC 7816 y se dividen en 9 partes que contienen los requisitos detallados de interoperabilidad de dispositivos compatibles, información de diseño, interfaces de programación y otras.[13]

1.2 Licencia de conducción en tarjetas inteligentes

Las tarjetas inteligentes son utilizadas en nuestros días como medio para gestionar la informacion de documentos oficiales de identificación, entre ellos la licencia de conducción. El presente epígrafe describe algunos aspectos relativos a la licencia de conducción en tarjetas inteligentes.

1.2.1 Soluciones similares

Tarjeta MyKad en Malasia

El documento electrónico MyKad es considerado como la primera cédula de identidad inteligente en el mundo, es una pieza de plástico que incorpora un microchip y tiene las dimensiones de una tarjeta de crédito que además de contener los certificados de autenticidad y firma electrónica reemplaza dos documentos oficiales: el carné de identidad y la licencia de conducción. Cumple con estándares internacionales para pasaportes electrónicos, facilitando la entrada y salida desde los puntos de inmigración del país, cuenta con información de salud del ciudadano, sirve en el sector bancario y como medio de pago en transportes y algunos establecimientos.[14] Esta solución informática es propietaria por lo que no puede ser adquirida por el país.

Licencia de Conducción Electrónica

Desde el surgimiento de las tarjetas inteligentes existen un grupo de empresas y organizaciones que llevan la delantera en lo que a estas tecnologías se refiere, en este grupo se encuentra Gemalto, compañía que se formó en junio de 2006 por la combinación de Axalto y Gemplus. Provee dispositivos

[14] PCSC: Por sus siglas en inglés *Personal Computer/Smart Card*.

personales seguros, software y servicios para aplicaciones del mundo digital. En el campo de las licencias de conducción y el registro de vehículos Gemalto colabora en implantaciones en El Salvador, Finlandia, India, Noruega, Suecia y algunos estados de México como Nuevo León, Veracruz y Sonora.[15]

Esta tarjeta inteligente realizada por Gemalto, permite a una persona específica conducir una determinada categoría de vehículo. Entre otros datos el chip almacena de forma segura una imagen de alta resolución del titular, las huellas, tipo de sangre y el historial de multas, permitiendo a las autoridades de tránsito controlar fácilmente el comportamiento de los conductores en las carreteras.[15]

En la vía el policía utiliza un terminal que puede leer el microprocesador de la licencia de conducción, el cual muestra en pantalla la información almacenada en la tarjeta de forma digitalizada incluyendo la imagen del titular para ser verificado a distancia por el policía, además permite al personal policial escribir las sanciones del conductor, la fecha y el lugar cada vez que éste infringe la ley.[15]

Estas licencias de conducción electrónicas han tenido gran aceptación en diferentes países, pues las mismas ayudan a reducir los accidentes de tránsito y las muertes relacionadas con estos accidentes, gracias al uso de la información sobre el comportamiento de los conductores, con la cual se calculan las primas de los seguros de automotores para cobrar seguros más caros a los conductores imprudentes. Sin embargo esta solución de software es propietaria de la empresa Gemalto y muy costosa, por lo que no puede ser adquirida por el país.

Electronic Driving License (eDL)

El proyecto eDL es la implementación de un *applet* y su *middleware* para gestionar la información del documento licencia de conducción en tarjetas inteligentes, es gratuito y está disponible bajo la licencia GPL[15] 2.0 en la dirección:

https://isodl.svn.sourceforge.net/svnroot/isodl

Esta solución es basada sobre el código desarrollado en el proyecto JMRTD (*Java Machine ReadableTtravel Documents*) para pasaporte electrónico, el cual se puede encontrar en la siguiente dirección:

http://jmrtd.org. [16]

Esta solución cumple con el estándar ISO/IEC 18013 para licencia de conducción electrónica y con el ISO/IEC 7816-4 para la estructura de ficheros que se almacenan en el chip de la tarjeta, permitiendo

[15] GPL: *General Public License* por sus siglas en inglés.

almacenar información en la tarjeta inteligente y obtener los datos almacenados en ella, además implementa mecanismos de seguridad para controlar el acceso a los datos, la autenticación entre el *applet* y el *middleware,* así como prevenir la sustitución del chip. Este software sirvió de base para la realización del presente trabajo.

1.2.2 Estándar ISO/IEC 18013

Este estándar consta de tres partes, la primera establece directrices para las características físicas de la tarjeta, un conjunto básico de elementos de datos obligatorios y opcionales, así como las características de seguridad física. La segunda define la estructura lógica de los datos, basándose en la estructura de ficheros que especifica la ISO/IEC 7816-4 para el almacenamiento de los datos en el chip. La tercera y última define los mecanismos de seguridad para controlar el acceso a datos, la autenticación y la validación e integridad de estos.

La parte 3 del estándar ISO/IEC 18013 establece una serie de mecanismos de seguridad para el acceso a los datos almacenados en el chip de la tarjeta, el primero que se nombra es de cumplimiento obligatorio y los restantes de uso opcional, estos son: Autenticación Pasiva (AP), Autenticación Activa (AA), Protección de Acceso Básico (BAP), Protección de Acceso Extendido (EAP) y Encriptación de datos. [17-19]

La implementación de estos mecanismos está soportada en el SO_D[16] y en las llaves (K_{ENC}[17], K_{MAC}[18], KPr_{AA}), almacenadas en la zona segura del chip, que solo puede ser accedida por el sistema operativo de la tarjeta inteligente.

- **LDS**: Grupo de datos almacenados en el chip.
- **SO_D**: Contiene los valores condensados del LDS que se están utilizando y es almacenado en el EF.
- **K_{ENC}, K_{MAC}**: Las llaves estáticas o de acceso base del documento que son almacenadas en el DF.
- **Obtención de llaves K_{ENC}, K_{MAC}**: A partir de una llave semilla (K_{seed}) o SAI[19] se realiza un cálculo de dos llaves 3DES, para establecer las llaves (K_{ENC} y K_{MAC}), de las cuales se derivan las llaves de sesión para la construcción segura de mensajes.
- **KPr_{AA}**: La llave privada de autenticación activa, se almacena en el DF.
- **KPu_{AA}**: La llave pública de autenticación activa, se almacena en un EF.[20]

[16] SO_D: Objeto de seguridad del documento.
[17] K_{ENC}: Llave de acceso base del documento, para encriptar el mensaje.
[18] K_{MAC}: Llave de acceso base del documento, para la autenticación del mensaje.
[19] SAI: Por sus siglas en inglés *Scanning Area Identifier*, es la llave que se utiliza para derivar las llaves estáticas de acceso base del documento.

Protección de acceso básico

Este mecanismo de seguridad previene la lectura no autorizada, las llaves usadas para efectuar dicho mecanismo son las llaves de sesión derivadas de las llaves estáticas K_{MAC} y K_{ENC}, las cuales se obtienen a partir de un protocolo Reto-Respuesta[20] (o *Challenge-Response*) entre el *middleware* y el *applet* al establecer una **autenticación mutua,** proceso necesario para iniciar un **canal seguro** de comunicación entre ambos. Al intentar acceder a la información almacenada en el chip, si se tiene activado este mecanismo de seguridad sin encontrarse autenticado previamente, la tarjeta responderá con el mensaje "condición de seguridad no satisfecha".[20]

- **Autenticación Mutua:** Siempre es iniciada por el *middleware*, el cual genera un *"host"* (reto) y se lo envía al *applet,* éste lo recibe y genera su propio "*card*" (reto). El *applet* usando su reto *card*, el reto *host* y las llaves estáticas, crea un par de llaves de sesión y genera su primer valor criptográfico "*card-cryptogram*" para ser transmitido al *middleware* junto al reto *card* y otros datos, el *middleware* por su parte usa un proceso similar para crear un segundo valor criptográfico y lo envía al *applet*. De esta forma la tarjeta y la terminal se autentican y obtienen mutuamente llaves de sesión KS_{ENC} y KS_{MAC}.
- **Canal Seguro:** Luego de haber realizado el proceso de autenticación mutua entre el *applet* y el *middleware* cada comando APDU se cifra con la llave de sesión KS_{ENC} y se verifica la integridad del mismo con KS_{MAC}, de forma que toda comunicación entre el terminal y la tarjeta se protegerá mediante la construcción segura de mensajes en modo Mac y Encriptado.

[20] Reto-Respuesta: El reto es un dato *Random* único que se genera para una sesión y la respuesta es la que se obtiene a partir de este *Random* recibido.

Autenticación pasiva

La AP consiste básicamente en la verificación del SO_D, este mecanismo de seguridad no requiere capacidades de procesamiento del chip, su objetivo es probar que el contenido del SO_D y su LDS son auténticos y no han sido cambiados, pero no previene la copia exacta del chip o su sustitución.[20]

Autenticación activa

El objetivo de la AA es proteger al documento de la sustitución del chip, consiste en que el chip tiene sus propias llaves asimétricas KPr_{AA} y KPu_{AA}. Luego de realizar la autenticación se obtienen los datos y se comparan con los almacenados en el chip, adicionalmente se establece un protocolo Reto-Respuesta entre el lector y el chip, el cual determina que el SO_D no ha sido copiado. Este mecanismo necesita capacidades de procesamiento del chip.[20]

Protección de acceso extendido y Encriptación de datos

Estos mecanismos de seguridad son de carácter opcional y están condicionados por las decisiones que tome el estado emisor del documento o de especificaciones bilaterales entre estados que comparten esta información, el objeto fundamental es asegurar las características biométricas adicionales (datos sensibles), dígase huellas digitales e iris, pues la imagen facial es de inclusión obligatoria. El EAP se basa en la restricción de acceso a los datos sensibles, el conjunto de llaves para el acceso puede consistir en llaves simétricas o asimétricas. La encriptación de datos consiste en almacenar los datos biométricos adicionales cifrados en el chip, la definición de los algoritmos de cifrado y descifrado de datos dependen del emisor del documento así como su distribución.[20]

La implementación de este estándar internacional brinda la posibilidad de emisión de un documento de licencia de conducción estandarizado para el reconocimiento y aceptación mundial, reemplazando el diseño actual del documento de papel en una tarjeta de plástico de tamaño ID-1, además incrementa la seguridad de los documentos actuales con respecto a los documentos de papel o plastificados.

1.3 Tecnologías de desarrollo

A continuación se presentan las tecnologías de desarrollo entre las que se incluyen la selección de la metodología de software, lenguajes y herramientas utilizadas para elaborar la solución.

1.3.1 Selección de la metodología de desarrollo

Es importante señalar la importancia que amerita una adecuada selección de la metodología a utilizar para el desarrollo del software, teniendo en cuenta que no existe una metodología absoluta, algunas se ajustan mejor que otras a las características y necesidades específicas de los proyectos de desarrollo.

Actualmente existen dos grandes grupos de metodologías de desarrollo, las metodologías tradicionales y las ágiles; las primeras se centran en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto, mientras que las segundas dan mayor importancia a la capacidad de respuesta a los cambios. Las más conocidas de cada grupo son: de las tradicionales *Rational Unified Process* (RUP) y Desarrollo de Sistemas de Jackson (JSD) y de las ágiles *Extreme Programming* (XP), *Iconix, Scrum* y *Feature Driven Development* (FDD).[21] La selección de la metodología se enmarcó en RUP y XP por ser las metodologías líderes en los grupos mencionados con anterioridad.

Rational Unified Process (RUP) [21]

RUP provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo, su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales, fue desarrollado por *Rational Software* y está integrado con toda la *Suite Rational* de herramientas, puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte, es guiado por casos de uso, centrado en la arquitectura y utiliza *Unified Modeling Language* (UML) como lenguaje de notación.

Principales elementos:

- **Trabajadores (“quién”):** Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo, ellos realizan las actividades y son propietarios de elementos.
- **Actividades (“cómo”):** Es una tarea que tiene un propósito claro y es realizada por un trabajador.
- **Artefactos (”qué”):** Productos tangibles del proyecto que son producidos, modificados y usados por las actividades, pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- **Flujo de actividades (“cuándo”):** Secuencia de actividades realizadas por trabajadores y que producen un resultado de valor observable.

Además define cuatro fases:

- Concepción
- Elaboración

- Construcción
- Transición

Estableciendo ocho flujos de trabajo presentes a lo largo de las fases mencionadas anteriormente, los flujos son:

1. Modelado del negocio
2. Requerimientos
3. Análisis y Diseño
4. Implementación y prueba
5. Despliegue
6. Administración del proyecto
7. Administración de configuración y cambio
8. Ambiente

Ventajas:

- Evaluación en cada fase que permite cambios de objetivos.
- Funciona bien en proyectos de innovación.
- Es sencillo ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software.
- Seguimiento detallado en cada una de las fases.

Desventajas:

- La evaluación de riesgos es compleja.
- Excesiva flexibilidad para algunos proyectos.
- El cliente deberá ser capaz de describir y entender a un gran nivel de detalle, para poder acordar un alcance del proyecto con él.

Extreme Programming (XP) [22]

XP es actualmente un enfoque deliberado y disciplinado para el desarrollo de software, la metodología está diseñada para entregar el software que el cliente necesita, en el momento que lo necesita, además promueve el uso de prácticas para aumentar la productividad del equipo de desarrollo y mejorar la adaptabilidad a los frecuentes cambios dentro del ciclo de vida del proyecto.

Algunas de las prácticas más usadas son:

- Entregas pequeñas y frecuentes.

- Cliente in-situ.
- Diseños simples.
- Integración continua.
- Programación en parejas.
- Desarrollo guiado por pruebas.
- Estándares y refactorización de código

Estas prácticas soportan los cuatro principios de la metodología:

1. Comunicación.
2. Simplicidad.
3. Retroalimentación.
4. Coraje.

Ventajas:

- Apropiado para entornos volátiles, equipos de desarrollo pequeños 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.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

Desventajas:

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

Fundamentación de XP como metodología a utilizar.

Después de un análisis de las características, ventajas y desventajas de ambas metodologías se identificaron ciertos factores que luego potenciaron la elección de XP como metodología a utilizar, los elementos determinantes fueron:

- El desarrollo en este caso es realizado por una sola persona.
- Imposibilidad para una sola persona, de asumir una metodología robusta, debido a la cantidad excesiva de roles y documentación generada en el ciclo de vida del proyecto.
- Problema a solucionar de complejidad media.
- Se requiere de un desarrollo incremental e iterativo, realizando pequeñas y continuas mejoras.

1.3.2 Lenguaje Unificado de Modelado (UML)

Es un lenguaje muy conocido y utilizado en la actualidad que ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso usar. Es una técnica de modelado de objetos y como tal supone una abstracción de un sistema para llegar a construirlo en términos concretos.[23, 24]

Algunas de las propiedades de UML como lenguaje de modelado estándar son:

- Concurrencia ya que es un lenguaje distribuido y adecuado a las necesidades de conectividad actual y futuras.
- Ampliamente utilizado por la industria desde su adopción por OMG[21].
- Reemplaza a decenas de notaciones empleadas con otros lenguajes.
- Modela estructuras complejas.
- La estructura más importante que soporta es que tiene su fundamento en las tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos.
- Emplea operaciones abstractas como guía para variaciones futuras, añadiendo variables si es necesario.
- Comportamiento del sistema: casos de uso, diagramas de secuencia y de colaboraciones, que sirven para evaluar el estado de las máquinas. [23, 24]

1.3.3 Herramienta Altova UModel

Altova UModel 2010 crea e interpreta diseños de software mediante la potencia del estándar UML 2.1, con la utilización de esta herramienta se puede corregir el código generado o los modelos y completar la ronda produciendo automáticamente nuevos diagramas o regenerando el código. Presenta múltiples características de usabilidad ya que es una herramienta fácil de usar, contiene elementos personalizables de diseño, estilos en cascadas y otras. Dibuja el diseño de la aplicación y puede generar código para *Java* o C# a partir de planos.[25]

Se propone la utilización de esta herramienta de modelado, debido a que permite realizar ingeniería inversa de programas de *Java* ya existentes a diagramas UML claros y precisos, con el objetivo de

[21] OMG: Object Management Group (OMG), establece las normas de UML.

obtener rápidamente su arquitectura de software, además permite obtener diagramas UML de programas de *JavaCard* de forma similar a los programas de *Java*.

1.3.4 Herramienta Developer Suite

Brinda un ambiente favorable para el diseño e implementación de *applets,* posibilita simular las aplicaciones realizadas en simuladores de tarjetas para las tarjetas *JavaCard* y otras, lo que permite probar los *applets* realizados antes de instalarlos en una tarjeta real y comprender las particularidades del lenguaje *JavaCard*. Con la utilización de esta herramienta se pretende obtener una aplicación que presente un correcto funcionamiento luego de ser instalado en la tarjeta inteligente.

1.3.5 Lenguaje de programación JavaCard

Es una combinación del lenguaje *Java* con un entorno de ejecución para tarjetas inteligentes, permite ejecutar *applets* en el chip embebido en la tarjeta, los cuales contienen funcionalidades que son reutilizables para otros componentes, se ejecutan e interactúan en todo momento con el entorno de ejecución que contiene la máquina virtual de *JavaCard*, junto a las clases y servicios definidos en las API[22] de estas.[6]

Ambiente de ejecución de aplicaciones en las tarjetas inteligentes

JCRE (*JavaCard Runtime Environment*) es el ambiente sobre el cual se ejecutan los *applets* y comprende el *JavaCard* API y la JCVM (*JavaCard Virtual Machine*). La JCVM se diferencia principalmente de una JVM (*Java Virtual Machine*) normal en que el tiempo de vida de la misma es igual al tiempo de vida de la tarjeta.[6]

La clase *javacard.framework.Applet* es una clase abstracta definida en la API de *JavaCard* provista en el entorno de desarrollo utilizado (*Developer Suite*), donde se definen cuatro métodos públicos que son utilizados por el JCRE para hacer funcionar las aplicaciones.

- **Método install(byte[ ], short, byte)**

Este método es invocado por el JCRE antes de crear una instancia del *applet* en la tarjeta, su implementación usual es llamar al constructor de la clase que normalmente es privado, luego crear todos los objetos que el *applet* necesitará para su ejecución y por último, registrar el *applet* con el método *register()*. No es estrictamente necesario crear todos los objetos en el método *install(),* sin embargo es una buena práctica de programación, pues garantiza la obtención de toda la memoria necesaria, evitando quedar más adelante en un estado inválido por falta de memoria. [6]

[22] API: *Application Programming Interface*, por sus siglas en inglés.

En caso de que se produzca una excepción durante la ejecución del método *install(),* el JCRE es responsable de realizar las actividades de limpieza pertinentes, una vez finalizado el método, el JCRE marca al *applet* como listo para ser seleccionado.[6]

- **Método select()**

Este método es invocado por el JCRE como consecuencia de la recepción a un *Select* APDU. El formato de este APDU está definido en el ISO/IEC 7816-4, contiene el identificador de la aplicación (AID de sus siglas en inglés) del *applet* a seleccionar el cual es una secuencia de entre 5 y 16 *bytes*, que identifica de forma única una aplicación para tarjetas inteligentes de acuerdo al ISO/IEC 7816-4. El formato de un AID se puede ver en la siguiente tabla:

Application Identifier (AID)	
National registered application provider (RAID)	Proprietary application identifier extension (PIX)
5 bytes	Entre 0 y 11 bytes

Tabla 3: Descripción de los AIDs.[6]

Una vez que el JCRE recibe un SELECT APDU, si hay algún *applet* seleccionado invoca a su método *deselect()* y luego invoca al método *select()* del *applet* cuyo AID fue especificado. Por distintas razones este puede declinar la selección, en cuyo caso el JCRE es responsable de responder adecuadamente al *Card Acceptance Device* (CAD, Reader o lector). En caso de que la selección se realice sin inconvenientes se pasa el SELECT APDU al método *process()* del *applet* seleccionado para que lo procese y devuelva al lector la información que sea pertinente.[6]

- **Método process(APDU)**

Cuando llega un APDU, el JCRE invoca este método del *applet* seleccionado, pasándole como parámetro el C-APDU recibido. Dentro de este método, el *applet* identifica el comando asociado al APDU y los parámetros si los hay y los procesa de acuerdo al protocolo que se haya definido para la interacción entre el *applet* y la aplicación terminal. En caso de que la ejecución finalice correctamente, el *applet* sólo debe encargarse de cargar en el R-APDU la información que va a devolver.[6]

El JCRE es responsable de resetear los SW del R-APDU al valor especificado para la ejecución exitosa (0x9000, de acuerdo a lo especificado en el ISO/IEC 7816). Durante el proceso de un APDU, el *applet* puede levantar una *ISOException* con los SW apropiados, si esta no es atrapada por el código del *applet*, es atrapada por el JCRE quien se encarga de generar el R-APDU correspondiente.[6]

- **Método deselect()**

Este método es invocado por el JCRE para avisar al *applet* que está actualmente seleccionado que va a dejar de estarlo, esto sucede cuando el JCRE recibe un *Select* APDU, aunque el AID del *applet* a seleccionar coincida con el del *applet* seleccionado. Esto brinda al *applet* la oportunidad de realizar las tareas de limpieza que sean necesarias para quedar en un estado consistente.[6]

1.3.6 Herramienta NetBeans

NetBeans es un entorno de desarrollo robusto que tiene gran aceptación, es un producto de código abierto, brinda facilidades para programar especialmente en lenguaje *Java*, aunque no se limita al uso de este lenguaje ya que posibilita desarrollar aplicaciones en la mayoría de los lenguajes existentes, permite la integración de módulos y *plug-in* desarrollados para aplicaciones específicas.

Se propone la utilización de este Entorno de Desarrollo Integrado (IDE) para la implementación del componente *middleware*. Una vez desarrollado y realizadas las pruebas de calidad correspondientes, el componente puede ser integrado al *applet* brindando una mayor comprensión de las funcionalidades que esta realiza.

1.3.7 Lenguaje Java

Con el auge de los lenguajes orientados a objetos, *Java* es uno de los principales lenguajes de programación actuales, su uso se puede centrar en todos los campos profesionales, pero específicamente se adapta muy bien a sistemas distribuidos y de comunicaciones, aplicaciones cliente-servidor, aplicaciones de gestión y otros, presenta un modelo de objetos simple y elimina herramientas de bajo nivel que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.[26]

Teniendo en cuenta las características descritas, se propone la utilización de este lenguaje para el desarrollo del *middleware*, el cual facilita la comunicación con el *applet* mediante el intercambio de mensajes. El *middleware* solicita la ejecución de servicios al *applet*, este último actúa como un componente local embebido en la tarjeta inteligente y escrito en lenguaje *JavaCard*.

Conclusiones del capítulo

- Mediante el análisis de los principales conceptos asociados a la solución se pudo obtener un mejor dimensionamiento del problema, ayudando también el estudio de las soluciones de tarjetas inteligentes para licencias de conducción, concluyendo que la solución eDL, de código abierto, es la que más se acerca a cubrir las necesidades del problema por lo que se toma como base para el desarrollo de este trabajo.

- Se analizaron las herramientas y tecnologías para el desarrollo de la solución, entre las que se encuentra la metodología de desarrollo de software XP, lenguaje de modelado UML apoyándose en la herramienta Altova UModel, la herramienta Developer Suite junto al lenguaje *JavaCard* para el desarrollo del *applet* y el entorno de desarrollo NetBeans junto al lenguaje *Java* para el desarrollo del *middleware*.

Capítulo 2: Propuesta de solución

Introducción

Para gestionar la información del documento licencia de conducción en tarjetas inteligentes, se propone realizar la implementación de un *applet* que permitirá escribir y leer datos dentro de la tarjeta, además un componente *middleware* que permitirá la comunicación entre la aplicación del cliente y el *applet* que estará instalado dentro de la tarjeta. Para el desarrollo de esta solución se parte del software *OpenSourse* eDL.

El presente capítulo muestra las fases de Planificación y Exploración definidas en la metodología de software de desarrollo *Extremme Programming* (XP), así como los diferentes artefactos generados en cada una de ellas. Se expone el Modelo de Dominio o Modelo Conceptual, el cual sirve de apoyo para una amplia comprensión de los elementos relacionados con *applet* y su *middleware*, identificando para esto las entidades principales que se tendrán y las relaciones entre ellas. Se interpretan las necesidades del sistema especificándolas mediante las historias de usuario y los requerimientos no funcionales. Se realiza una estimación de tiempo y plan de iteraciones. Se describen las tareas de ingenierías derivadas de las historias de usuario. Finalmente se describe la arquitectura del sistema y se realiza un plan de entregas.

2.2 Modelo de Dominio

Se ha procedido a crear un modelo de dominio para mostrar los principales conceptos relacionados con la solución informática que se propone desarrollar, de esta manera los usuarios, desarrolladores e interesados podrán utilizar un vocabulario común para poder entender el contexto que se dispone en este.

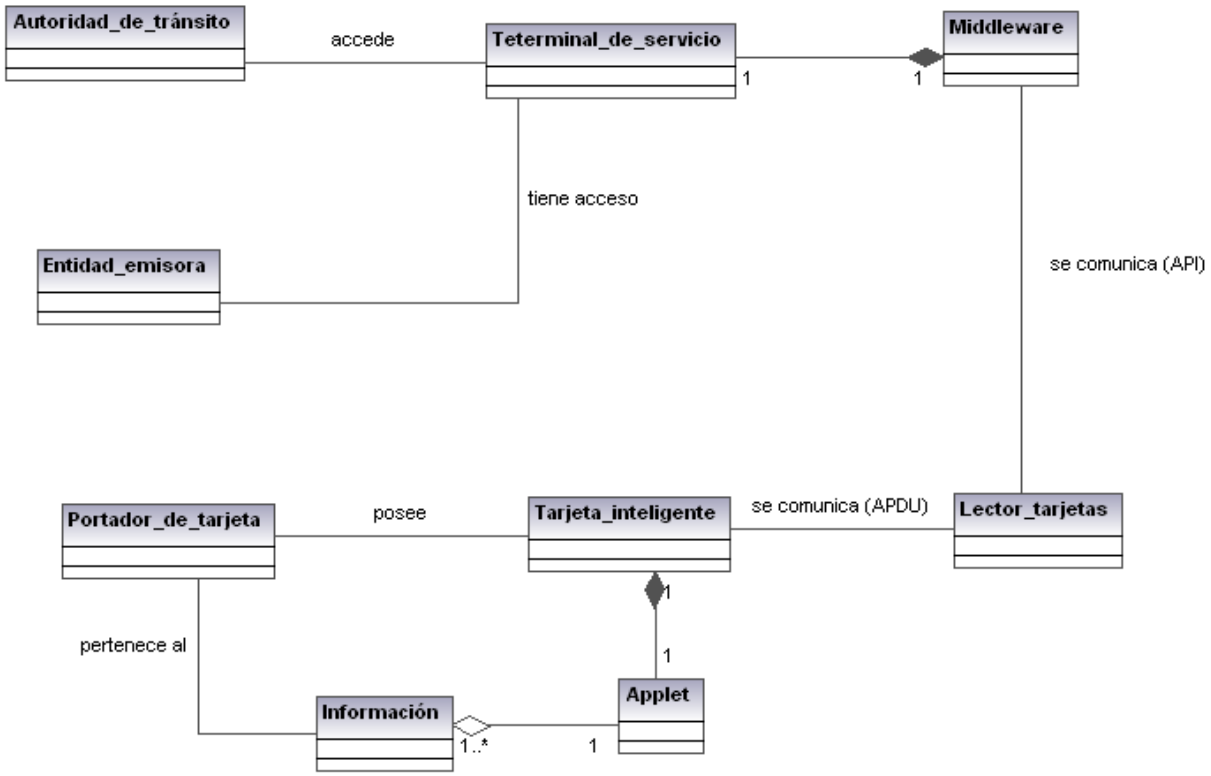

Figura 7: Modelo de dominio. (Creación de la autora)

2.2.1 Glosarios de términos del modelo de dominio

Los términos presentados a continuación conceptualizan el Modelo de Dominio de la solución propuesta.

Tarjeta_inteligente: Es un dispositivo de plástico de tamaño ID-1 que presenta 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 tales como almacenar información, cifrar información, leer y escribir datos al igual que un ordenador.

Applet: Es una aplicación implementada en *JavaCard*, instalada dentro de la tarjeta inteligente la cual permite gestionar la información que se almacene dentro de la misma.

Información: Son datos referentes al portador de la licencia de conducción electrónica, los cuales se almacenan en la misma.

Middleware: Es un componente que funciona como capa de traducción entre el *applet* y las aplicaciones del cliente, permitiendo una mejor comprensión de las respuestas obtenidas por la comunicación establecida con la aplicación instalada en la tarjeta.

Terminal_de_servicio: Es un punto en una computadora donde se instalan todas las condiciones para poder interactuar con la solución.

Lector_tarjetas: Es un lector compatible con el estándar PC/SC, el cual define la comunicación entre el *Applet_LicenciaConducir y Middleware_LicenciaConducir.*

Portador_de _tarjeta: Portador del documento electrónico de licencia de conducción.

Autoridad_tránsito: Personal policial que verifica los datos de la licencia de conducción electrónica.

Entidad_Emisora: Entidad que emite la licencia de conducción electrónica.

2.3 Fases de un proyecto con XP

El ciclo de vida de XP se basa en el concepto de iteración de desarrollo, donde una iteración de desarrollo es un espacio de tiempo en el que se realiza un conjunto de funcionalidades determinadas que en el caso de XP corresponden a un conjunto de historias de usuario. Existe una fase de análisis inicial orientada a planificar las iteraciones de desarrollo y cada iteración incluye diseño, codificación y pruebas, sub-fases superpuestas de tal manera que no se separen en el tiempo.[22]

Los artefactos esenciales en XP son:

- Historias de usuario
- Tareas de ingeniería
- Pruebas de aceptación
- Pruebas unitarias
- Plan de entrega
- Código

A continuación se verán algunos puntos fundamentales de las fases de Exploración, Planificación, Iteraciones y Producción.

Fase de Exploración

En esta fase se redactan de forma sencilla las historias de usuario, se realiza una estimación de los tiempos de desarrollo de cada historia de usuario en base a esta información, para tener como resultado una visión general del sistema. Las estimaciones realizadas pueden variar cuando se analicen más en detalle en cada iteración.[22]

Fase de Planificación

La planificación es una fase corta, en la que se establece el orden en que deberán implementarse las historias de usuario y asociadas a éstas las entregas.[22]

Fase de Iteraciones

Es la fase principal en el ciclo de desarrollo de XP, como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se describen las tareas necesarias de ingeniería.[22]

Fase de Puesta en Producción

Al final de cada iteración se entregan módulos funcionales y sin errores. En esta fase pueden ser necesarias tareas de ajuste (*fine tuning*).[22]

2.3.1 Historias de Usuario

Las historias de usuario son utilizadas en XP para especificar los requisitos del software, las mismas son escritas por los clientes como las tareas que el sistema debe tener y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas, son descripciones cortas y escritas en el lenguaje del usuario sin terminología técnica y proporcionan los detalles sobre la estimación del riesgo y cuánto tiempo llevará su implementación. [22]

Durante la fase se identificaron 7 historias de usuario que estarán organizadas por prioridad, las cuales indican las funcionalidades que el sistema debe cumplir. A continuación se presenta la descripción de cada una de las historias de usuario.

Historia de Usuario	
Número: HU1	**Nombre:** Establecer comunicación

Usuario: Terminal de servicio	
Prioridad del Negocio: Alto	**Riesgo de Desarrollo:** Alto
Puntos Estimados: 2	**Iteración Asignada:** 2
Responsable: Adonis Cesar Legón Campos, Lissi Fernández Miyet, Ander Sánchez Jardines	
Descripción: Debe permitir que se establezca una comunicación entre el *middleware* y el *applet* de la licencia de conducción electrónica, para ello la tarjeta debe estar insertada en cualquier lector que cumpla con el estándar PC/SC y esté conectado al terminal de servicio.	

Tabla 4: HU1-Establecer comunicación. (Creación de la autora)

Historia de Usuario	
Número: HU2	**Nombre:** Almacenar llave semilla
Usuario: Terminal de servicio	
Prioridad del Negocio: Alto	**Riesgo de Desarrollo:** Alto
Puntos Estimados: 2	**Iteración Asignada:** 2
Responsable: Adonis Cesar Legón Campos, Lissi Fernández Miyet, Ander Sánchez Jardines	
Descripción: Debe permitir que se almacene la llave semilla tanto en el *applet* como en el *middleware.* Esta llave es la concatenación de 16 caracteres (la política a utilizar queda a discreción de la entidad que emite el documento), de entrar menos caracteres se hace obligatorio realizar un cálculo resumen (*hash*), el cual consiste en realizarle un *hash* a la cadena de caracteres y se obtiene de este cálculo una nueva cadena de 20 *bytes,* de estos los 16 más significativos se utilizan para obtener las llaves estáticas de acceso K_{ENC} y K_{MAC}.	

Tabla 5: HU2-Almacenar llave semilla. (Creación de la autora)

Historia de Usuario	
Número: HU3	**Nombre:** Gestionar el sistema de ficheros
Usuario: Terminal de servicio	
Prioridad del Negocio: Alto	**Riesgo de Desarrollo:** Alto

Puntos Estimados: 2	**Iteración Asignada:** 2
Responsable: Adonis Cesar Legón Campos, Lissi Fernández Miyet, Ander Sánchez Jardines	
Descripción: Debe permitir que se almacene la información en los ficheros de cada grupo de datos que se quiere personalizar en la tarjeta.	

Tabla 6: HU3-Gestionar el sistema de ficheros. (Creación de la autora)

Historia de Usuario	
Número: HU4	**Nombre:** Efectuar BAP
Usuario: Terminal de servicio	
Prioridad del Negocio: Medio	**Riesgo de Desarrollo:** Alto
Puntos Estimados: 2	**Iteración Asignada:** 2
Responsable: Adonis Cesar Legón Campos, Lissi Fernández Miyet, Ander Sánchez Jardines	
Descripción: Debe permitir que se realice el mecanismo BAP siguiendo el estándar ISO/IEC 18013-3, con la finalidad de que el *applet* y el *middleware* se reconozcan como válidos para intercambiar mensajes entre sí de forma segura.	

Tabla 7: HU4-Efectuar BAP. (Creación de la autora)

Historia de Usuario	
Número: HU5	**Nombre:** Leer grupo de datos
Usuario: Terminal de servicio	
Prioridad del Negocio: Medio	**Riesgo de Desarrollo:** Alto
Puntos Estimados: 2	**Iteración Asignada:** 2
Responsable: Adonis Cesar Legón Campos, Lissi Fernández Miyet, Ander Sánchez Jardines	
Descripción: Debe facilitar la lectura de la información contenida en los ficheros de cada grupo de datos personalizados en la tarjeta.	

Tabla 8: HU5-Leer grupos de datos. (Creación de la autora)

Historia de Usuario	
Número: HU6	**Nombre:** Realizar autenticación pasiva
Usuario: Terminal de servicio	
Prioridad del Negocio: Medio	**Riesgo de Desarrollo:** Alto
Puntos Estimados: 2	**Iteración Asignada:** 2
Responsable: Adonis Cesar Legón Campos, Lissi Fernández Miyet, Ander Sánchez Jardines	
Descripción: Debe facilitar la verificación de la información contenida en el objeto de seguridad mediante el mecanismo AP siguiendo el estándar ISO/IEC 18013-3, con la finalidad de verificar la autenticidad de la información almacenada.	

Tabla 9: HU6-Realizar autenticación pasiva. (Creación de la autora)

Historia de Usuario	
Número: HU7	**Nombre:** Realizar autenticación activa
Usuario: Terminal de servicio	
Prioridad del Negocio: Medio	**Riesgo de Desarrollo:** Alto
Puntos Estimados: 2	**Iteración Asignada:** 2
Responsable: Adonis Cesar Legón Campos, Lissi Fernández Miyet, Ander Sánchez Jardines	
Descripción: Debe permitir la realización del mecanismo AA siguiendo el estándar ISO/IEC 18013-3 sólo si se realizó el mecanismo AP, con la finalidad de proteger el documento de la sustitución del chip.	

Tabla 10: HU7-Realizar autenticación activa. (Creación de la autora)

2.3.2 Requisitos no Funcionales

Además de las funcionalidades descritas en las historias de usuario se tienen en cuenta los requerimientos no funcionales o propiedades que debe cumplir el producto.

Requerimiento de Hardware:

- Lector de tarjetas inteligentes incorporado a la PC que cumpla con el estándar PC/SC.
- Tarjetas inteligentes que soporten el sistema operativo *JavaCard*.

Requerimiento de Software:

- **Usabilidad**
  - El *middleware* deberá ser de fácil de utilizar y capaz de brindar comodidad a la hora de integrarse con su *applet* correspondiente.
  - Se requiere la máquina virtual de *Java* versión 6 en adelante.
- **Rendimiento**
  - El *applet* deberá ser capaz de realizar todas sus funcionalidades en un corto intervalo de tiempo de manera eficiente.
- **Comunicación**
  - Debe estar instalado en la terminal de servicio los *drivers* del lector de tarjetas.
  - Se requiere del estándar ISO/IEC 7816-4 para establecer la comunicación entre el terminal y la tarjeta.
- **Seguridad**
  - La solución debe implementar los mecanismos de seguridad para el acceso y verificación de la información almacenada en chip según el estándar ISO/IEC 18013-3.
  - La solución deberá recuperarse en el menor tiempo posible en caso de haber una falla.
- **Interfaces de Comunicación**
  - Comunicación con lectores de tarjetas inteligentes mediante el estándar PC/SC.

2.3.3 Metáfora

Después de haberse definido las funcionalidades que el sistema debe cumplir y los requerimientos no funcionales, se da lugar a la creación de la metáfora, esta es una breve descripción de cómo debe funcionar el sistema en su totalidad y es encargada de guiar todo el desarrollo como una gran historia de usuario, ayudando a entender los elementos básicos del sistema y sus relaciones.[22]

El sistema para la gestión de la información contenida en la licencia de conducción electrónica de un ciudadano, funcionará en aquellas entidades donde se desee hacer uso de la misma y cuente con la infraestructura adecuada. Desde el momento que exista una conexión abierta entre un lector y una tarjeta inteligente, el usuario hará una solicitud que es procesada, a través de un componente cliente

(*middleware*) y transmitida a la aplicación (*applet*) instalada en la tarjeta. A esta solicitud ya sea de lectura o escritura, esta aplicación siempre devolverá una respuesta que puede ser satisfactoria, la información solicitada o un mensaje de error.

2.3.4 Arquitectura

El *applet* y su *middleware* desarrollado para gestionar la información de la licencia de conducción electrónica, basa su implementación en una arquitectura de dos capas *Card/Terminal*, la primera contiene la aplicación *Applet-DriverLicense* con la información referente al portador, la cual interactúa en todo momento con el entorno de ejecución (JCRE) que contiene la máquina virtual de *JavaCard* (JCVM) junto a las clases y servicios definidos en la API de esta, las cuales son utilizadas por este ambiente para hacer funcionar las aplicaciones embebidas en la tarjeta. Por su parte, la capa *Terminal* contiene un componente *Middleware-DriverLicense* que implementa las funcionalidades requeridas, dicho componente establece una comunicación directa con los lectores de tarjetas inteligentes a través de la API *Java SmartCard I/O*, la cual permite la comunicación con cualquier equipo que cumpla con las especificaciones PC/SC. El flujo de datos entre las dos capas se realiza a través de comandos APDU.

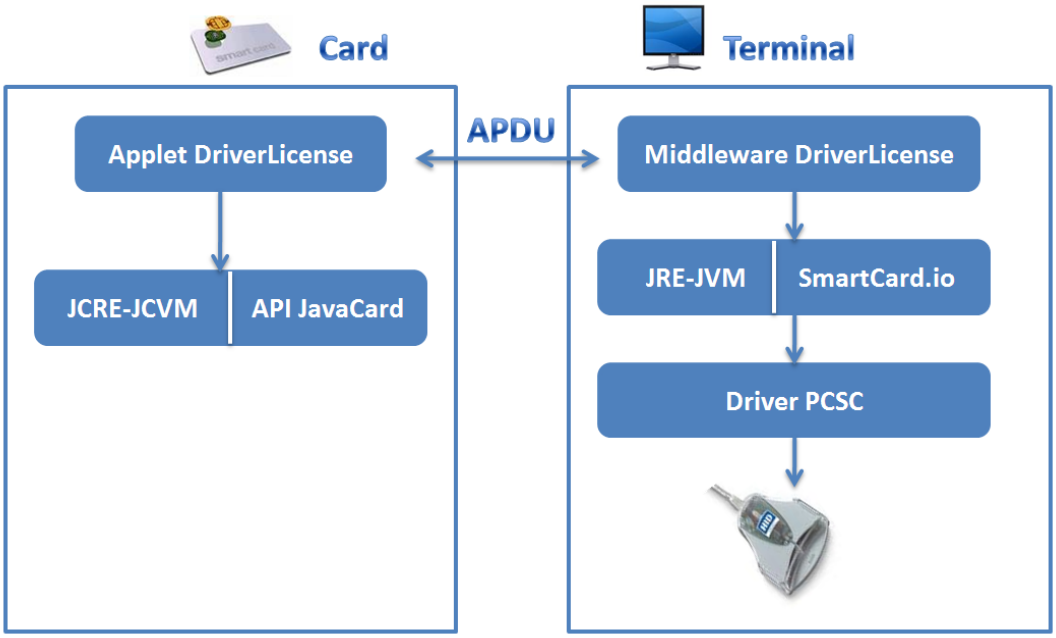

Figura 8: Arquitectura del sistema. (Creación de la autora)

2.3.5 Patrones Arquitectónicos y de Diseño

Modelo Vista Controlador (MVC)

Es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Se utilizó el patrón MVC ya que permite implementar por separado cada componente, entre sus elementos se encuentra el modelo que son los datos y reglas de negocio, la vista que muestra la información del modelo al usuario y el controlador que gestiona las entradas del usuario. En la solución el modelo es el *applet,* la vista es la interfaz de prueba[23] y la lógica del negocio es el *middleware* que es el responsable de recibir los eventos de entrada desde la vista, procesar la información y enviarla al *applet*.

Patrones de Diseño

***Interpreter* (Intérprete):** Dado un lenguaje define una representación para su gramática junto con un intérprete, que usa dicha representación para interpretar sentencias en ese lenguaje.[23]

***Chain of Responsibility* (Cadena de responsabilidades):** Permite establecer una cadena de objetos receptores, a través de los cuales se pasa una petición formulada por un objeto emisor, donde cualquiera de los objetos receptores puede responder a la petición en función de un criterio establecido.[23]

***Abstract Factory* (Fábrica abstracta):** Ofrece una interfaz para la creación de familias de productos relacionados o dependientes sin especificar las clases concretas a las que pertenecen. El problema que intenta solucionar este patrón es el de crear diferentes familias de objetos. Aísla las clases de implementación, ayuda a controlar los objetos que se creen y encapsula la responsabilidad de creación. Fomenta la consistencia entre productos.[23]

***Builder* (Constructor):** Su propósito es separar la construcción y la representación de un objeto complejo, para así permitir que el mismo proceso de construcción, sirva para crear diferentes representaciones. Como constructor de objetos complejos, este patrón es similar a Fábrica Abstracta. La diferencia reside en que Constructor, se centra en la creación de los objetos paso a paso (parte a parte), mientras Fábrica Abstracta está orientada a familias de objetos (simples o complejos). Además, en el primero el resultado se obtiene en el último paso y en el segundo se obtiene inmediatamente.[23]

[23] Interfaz de prueba: Se utiliza para probar las funcionalidades del *applet* y el *middleware*.

***Adapter* (Adaptador):** Su propósito es convertir la interfaz de una clase para que se adapte a lo que el cliente necesita, permitiendo así que trabajen juntas clases cuyas interfaces son incompatibles.[23]

2.3.6 Estimación de Tiempo

Se realiza una estimación del tiempo que se necesita para desarrollar cada historia de usuario, este valor se expresa en semanas, donde una historia de usuario no debe desarrollarse en menos de una, ni en más de dos semanas, en otro caso será necesario acoplarlas o dividirlas.[22]

Historia de Usuario	Estimación (semanas)
Establecer comunicación	2
Almacenar llave semilla	2
Gestionar el sistema de ficheros	2
Efectuar protección de acceso básico	2
Leer grupo de datos	2
Realizar autenticación pasiva	2
Realizar autenticación activa	2
Total	14

Tabla 11: Estimación de tiempo. (Creación de la autora)

2.3.6 Plan de Iteraciones

Generalmente las historias de usuario con mayor prioridad son asignadas a las primeras iteraciones y las restantes a las iteraciones posteriores, en la presente solución las historias de usuario “Establecer comunicación”, “Almacenar llave semilla” son de prioridad alta, así como “Gestionar el sistema de ficheros”, “Efectuar BAP”, “Leer grupo de datos”, “Realizar autenticación pasiva” y “Realizar autenticación activa” de prioridad media. Por lo que se definen 2 iteraciones con un total de 14 semanas de duración.

Iteración	No.HU	Historias de Usuarios	Duración Estimada
Iteración 1	HU1	Establecer comunicación	4 semanas
	HU2	Almacenar llave semilla	

Iteración 2	HU3	Gestionar el sistema de ficheros	10 semanas
	HU4	Efectuar BAP	
	HU5	Leer grupo de datos	
	HU6	Realizar autenticación pasiva	
	HU7	Realizar autenticación activa	

Tabla 12: Plan de iteraciones. (Creación de la autora)

2.3.7 Tareas de Ingeniería

Cada una de las historias de usuario definidas en la fase de Exploración se divide en tareas de ingeniería, estas explican a un nivel de detalle requerido la codificación de la solución. Una historia de usuario generalmente se divide en más de una tarea de ingeniería, donde cada una de ellas corresponderá a un periodo de uno a tres días de desarrollo.[22] Con el objetivo de planificar el trabajo, en el Plan de Iteraciones se agruparon las historias de usuario en 2 iteraciones.

Tareas de Ingeniería de la Iteración 1

Iteración	Historia de Usuario	Tarea
1	Establecer comunicación	Establecer comunicación
	Almacenar llave semilla	Almacenar llave semilla.

Tabla 13: Tareas de ingeniería de la iteración 1. (Creación de la autora)

Tareas de Ingeniería de la Iteración 2

Iteración	Historia de Usuario	Tarea
2	Gestionar el sistema de ficheros	- Crear un EF.SOD para la autenticidad e integridad de los datos. - Definir los grupos de datos obligatorios y opcionales. - Almacenar información en el fichero.
	Efectuar BAP	- Efectuar BAP
	Leer grupo de datos	- Leer grupos de datos

	Realizar autenticación pasiva	- Realizar autenticación pasiva
	Realizar autenticación activa	- Realizar autenticación activa

Tabla 14: Tareas de ingeniería de la iteración 2. (Creación de la autora)

Para desarrollar esta actividad se realiza una descripción de cada una de las tareas que se efectuarán por HU, donde se especifica su nombre, el tipo de tarea, el tiempo estimado para su realización y el programador responsable. En la siguiente tabla se describe la tarea de ingeniería perteneciente a la HU1_Establecer comunicación, el resto de las tareas se incluyen en el Anexo 3.

Tarea	
Numero de Tarea: HU1_T1	**Historia de Usuario:** Establecer comunicación
Nombre: Establecer comunicación	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Responsable: Lissi Fernández Miyet	
Descripción: Obtener los lectores conectados a la terminal de servicio y se detecta si está insertada la tarjeta en el lector, una vez finalizada la comunicación entre ambos esta debe ser cerrada. En caso de ser desconectado el lector del terminal o la tarjeta del lector se crea una nueva sesión.	

Tabla 15: HU1_T1-Establecer comunicación. (Creación de la autora)

2.3.8 Plan de Entrega

Para finalizar la fase de Planeamiento se realiza el Plan de Entregas que define las fechas en que serán liberadas las versiones funcionales del producto o el producto en su totalidad.[22]

Entregable	Fin Iteración 1
Applet y middleware.	Junio 2011

Tabla 16: Plan de entrega. (Creación de la autora)

Conclusiones del Capítulo

Luego de terminadas las fases de Exploración y Planeación de la solución propuesta, se concluye:

Teniendo en cuenta las normas de la ISO/IEC 18013 se definieron las historias de usuario y requerimientos no funcionales, se delimitaron 2 iteraciones con el objetivo de planificar el trabajo para el desarrollo de la solución y se definieron las tareas de ingeniería derivadas de las historias de usuario, además se expuso la arquitectura de la solución.

Capítulo 3: Implementación y prueba de la solución

Introducción

Luego de haber terminado las fases de Exploración y Planificación definidas en XP, se procede en el presente capítulo a realizar las fases restantes de Iteraciones a primera liberación y Producción, en las que se muestran los diferentes artefactos generados en dichas fases, donde se realiza una detallada explicación del diseño a través de tarjetas CRC y características de la solución, finalmente se exponen las diversas pruebas realizadas a la misma.

3.1 Fase de Iteración

3.1.1 Diseño

Luego de haber desglosado cada historia de usuario en tareas de ingeniería, se unifican las ideas para ejecutar dichas tareas en sesiones de diseño previas a cada iteración, en estas sesiones con el objetivo de agilizar el proceso de desarrollo, XP propone la puesta en práctica de ciertos principios descritos a continuación.

Estos principios son:

- **Simplicidad:** Un diseño simple siempre se termina más rápido y es más fácil de entender que uno complejo.
- **Uso de tarjetas CRC:** Estas tarjetas son manejadas por el equipo de desarrollo durante la codificación de la solución, generalmente cada tarjeta representa una clase diferente en la codificación y tienen como ventaja que todo el equipo contribuye a la elaboración del diseño de la solución.
- **No adicionar funcionalidades tempranamente:** Mantener el sistema lo más separado de las funcionalidades extra que no sean imprescindibles.[22]

Se realizaron las tarjetas CRC (ver Anexo 2), acompañada de la representación gráfica de diagramas de clases (ver Anexo 1), dando cumplimiento de esta manera a los principios antes mencionados, luego de haber realizado los ajustes que se describen a continuación.

Ajustes realizados a la solución

Se realizó un análisis a la solución eDL existente y mediante la herramienta de desarrollo Developer Suite de Gemalto, utilizando las tarjetas inteligentes de prueba con las que cuenta el CISED, se pudo generar el binario *JavaCard* (.cap). A partir de esto, se identificó que el mecanismo de Protección de Acceso Extendido o EAP estaba basado en el algoritmo asimétrico de curva elíptica, el cual no es soportado por estas tarjetas, por lo que se procedió a modificar la implementación actual para evitar su uso y así realizar un proceso de prueba de las restantes funcionalidades del *applet*.

Descripción del Diagrama de Despliegue

La aplicación para gestionar la información del documento de licencia de conducción debe estar instalada en cada Licencia de Conducción Electrónica (LCE) que proporcione la entidad emisora. El componente *middleware* para realizar la comunicación con la LCE, debe estar en cada uno de los terminales de servicios que existan para la realización del proceso de personalización o aplicación del documento. Esta comunicación se realizará mediante el envío y respuesta de comandos APDU entre los lectores de tarjetas y las tarjetas inteligentes. El lector se conecta al terminal de servicio mediante el estándar USB.

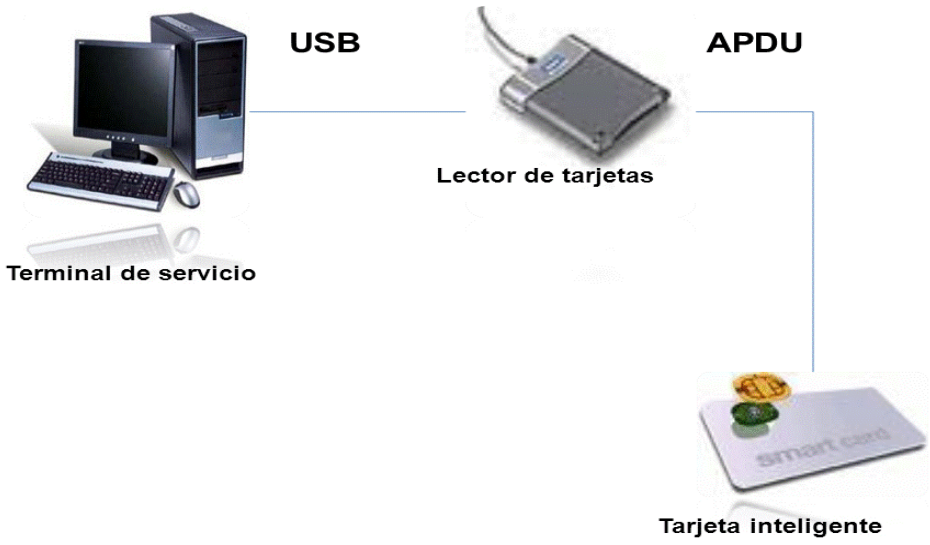

Figura 9: Diagrama de Despliegue. (Creación de la autora)

3.1.2 Implementación del applet y su middleware

Características del applet

Estructura de ficheros

El *applet* de la licencia de conducción electrónica incluye un sistema de ficheros (*File System*) que se responsabiliza por la administración de los ficheros que se almacenan. La instalación del *applet* activa una única vez la construcción de una jerarquía de ficheros dedicados, manipulados en el nivel global de la estructura general, creándose un fichero raíz que indica el comienzo de la estructura de ficheros y un fichero dedicado (*Dedicated File*), el cual es el contenedor a nivel global de todos los ficheros elementales (*Elementary File*), como se muestra en la siguiente figura.[7]

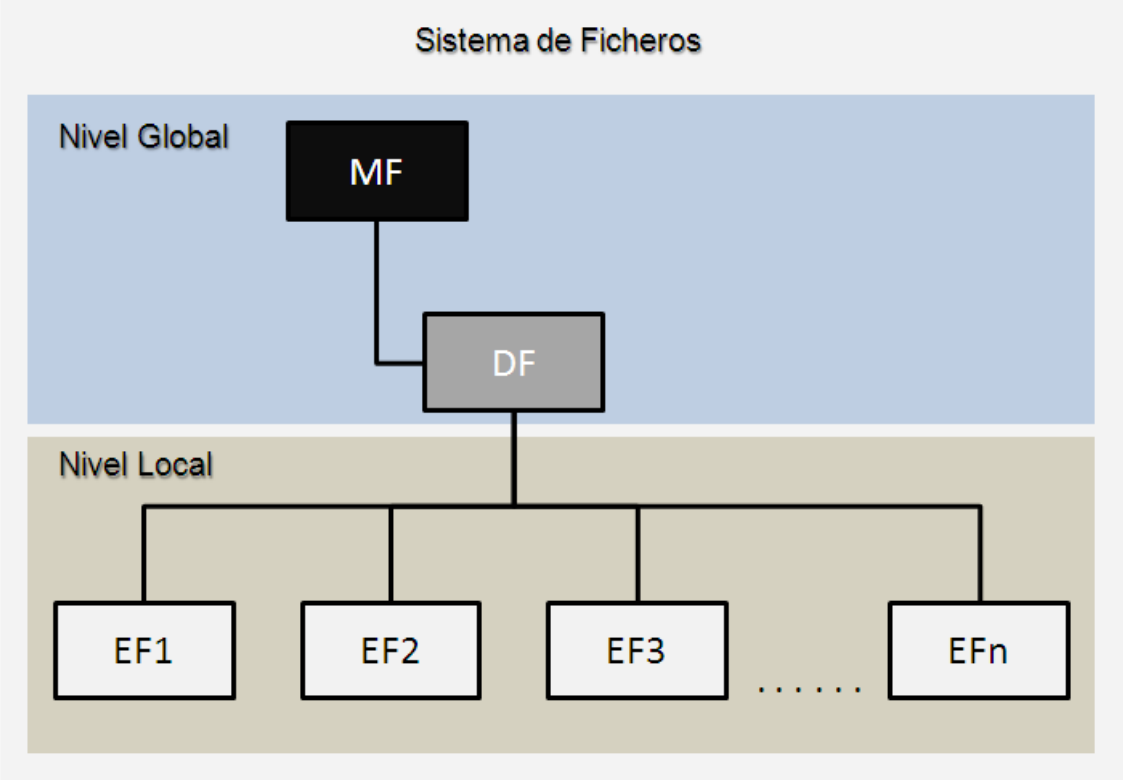

Figura 10: Diagrama de estructura de ficheros.[7]

Cada grupo de datos (*Data Groups*) consiste en una serie de datos dentro de una plantilla y será almacenado en un EF separado. Cada dato dentro del grupo de datos posee una identificación única *Tag,* que es especificada en un código hexadecimal.

Data Group	EF Name	FID	Tag
Common	EF.COM	011D	60
DG1	EF.DG1	0101	61
DG2	EF.DG2	0102	6B
DG3	EF.DG3	0103	6C
DG4	EF.DG4	0104	65
DG5	EF.DG5	0105	67
DG6	EF.DG6	0106	75
DG7	EF.DG7	0107	63
DG8	EF.DG8	0108	76
DG9	EF.DG9	0109	70
DG10	EF.DG10	010A	-
DG11	EF.DG11	010B	6D
DG12	EF.DG12	010C	71
DG13	EF.DG13	010D	6F
DG14	EF.DG14	010E	6E
DG15	EF.DG15	010F	62
Security Data	EF.SOD	011E	77

Tabla 17: Ficheros en el chip. (Creación de la autora)

El EF.COM almacena los datos comunes que corresponden fundamentalmente a la organización de los datos dentro del chip, donde 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 corresponda con el grupo de datos que contenga, el nombre del fichero EF que contiene los datos de seguridad se denomina EF.SOD.

Diagrama de estados del ciclo de vida del applet

El *applet* parte de un estado inicial que mediante una transición “instalar”, en la cual se carga e instancia el *applet* en la tarjeta, pasa a un estado **Virgen** en el que no contiene información, a partir de este por medio de la transición “Guardar llave SAI” pasa al estado **Personalizable** en el cual se encuentra listo para almacenar la información que va a ser almacenada en la tarjeta mediante una o varias transiciones “Escribir Grupo de Datos” en la que se termina de almacenar la información necesaria.

Por una transición “Finalizar Personalización” se llega al estado de **Aplicación**, que a través de “Efectuar BAP” pasa al estado **Autenticado BAP**. Una vez que la tarjeta y el terminal de servicio se autentican se procede mediante transiciones “Leer Grupos de Datos” a leer toda la información almacenada. Desde este estado mediante la transición “Realizar AA”, la cual se encarga de verificar la autenticidad del chip, puede pasar al estado **Autenticado AA** o retornar al estado de **Aplicación** por una transición “Finalizar Sesión” que puede ocurrir cuando se desee finalizar la lectura de los datos almacenados en la tarjeta, en caso de desconectarse la tarjeta del lector o el lector de la terminal de servicio.

A continuación se muestra el diagrama de estados del ciclo de vida del *applet*.

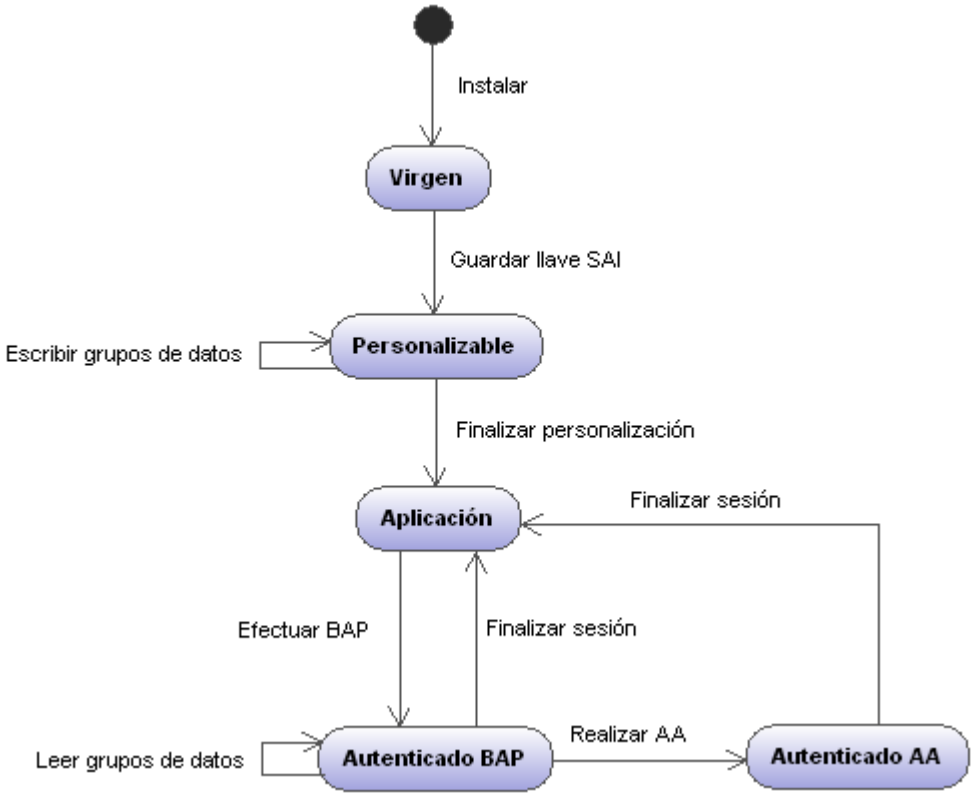

Figura 11: Diagrama de estados del ciclo de vida del applet. (Creación de la autora)

Características del middleware

El *Middleware Driver License* es un componente que actúa como una capa que aísla al humano de la comunicación directa de las operaciones que realiza el *Applet Driver License,* es capaz de incluir la definición de comandos APDU que son enviados al *applet,* los cuales están definidos en el estándar ISO/IEC 7816–4. La comunicación del *middleware* con el *applet* se realiza a través del Framework SCUBA para gestionar la comunicación con las tarjetas, el cual implementa a nivel de la API (*Java.SmartCardio*) la comunicación con los lectores de tarjetas electrónicas inteligentes.

El *Middleware Driver License* implementa los procesos definidos en los requerimientos de la solución, la comunicación entre este y el *applet* está determinada bajo capas que son necesarias para establecer un canal correcto de transmisión de los datos de información. El siguiente diagrama demuestra dichas capas de comunicación.

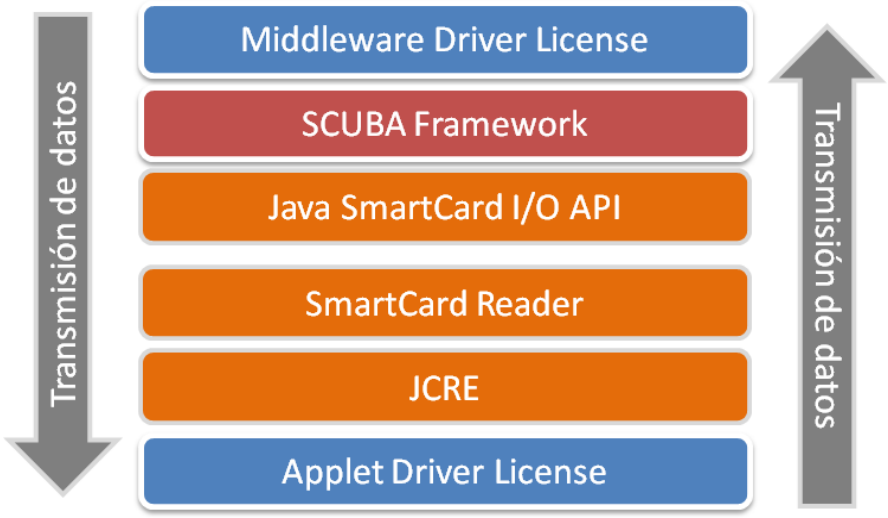

Figura 12: Diagrama de capas de comunicación. (Creación de la autora)

3.2 Puesta en producción

La fase de producción requiere de pruebas adicionales 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, debido a cambios durante esta fase. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación, por lo que el perfeccionamiento de código sólo fue posible a través de un grupo de pruebas unitarias, que aseguraron la ejecución correcta del sistema en todo el período de desarrollo.

Las pruebas del sistema se dividieron en dos grupos:

- **Pruebas unitarias:** encargadas de verificar el código y diseñadas por los programadores.
- **Pruebas de aceptación o pruebas funcionales:** destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

3.2.2 Pruebas unitarias

Las pruebas unitarias son pruebas que se realizan a lo largo de todo el proyecto para hacer más sencillas y efectivas las pruebas finales, aseguran siempre el correcto funcionamiento evitando las ambigüedades. La técnica de prueba utilizada fue la prueba de caja blanca la cual se realizó utilizando el método del camino básico, que permite obtener la complejidad de un diseño procedimental y utilizar esta medida

como guía para la definición de una serie de caminos básicos de ejecución, para ello se diseñan casos de pruebas que garanticen que cada camino se ejecuta al menos una vez.

Para las pruebas de caja blanca se definen aquellos métodos de mayor importancia en cada una de las aplicaciones, luego se crean los casos de prueba asociados a ellos definiendo los valores a los que deberá responder correctamente el sistema.[27]

Técnica utilizada: camino básico.

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control, para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática dado el código seleccionado.[27]

Caso de prueba para el código que permite seleccionar un fichero, desarrollado para el *applet.*

```
private void processSelectFile(APDU apdu) {
    byte[] buffer = apdu.getBuffer();
    byte p1 = buffer[OFFSET_P1];                          1
    byte p2 = buffer[OFFSET_P2];
    if(p1 != (byte)0x02 || p2 != (byte)0x0C) {  2
        ISOException.throwIt(SW_INCORRECT_P1P2);  3
    }
    short lc = (short) (buffer[OFFSET_LC] & 0x00FF);  4

    if (lc != 2)  5
        ISOException.throwIt(SW_WRONG_LENGTH);  6

    if (apdu.getCurrentState() == APDU.STATE_INITIAL) {  7
        apdu.setIncomingAndReceive();  8
    }
    if (apdu.getCurrentState() < APDU.STATE_FULL_INCOMING) {  9

        ISOException.throwIt(SW_INTERNAL_ERROR);  10
    }

    short fid = Util.getShort(buffer, OFFSET_CDATA);  11

    if (fileSystem.getFile(fid) != null) {  12
        selectedFile = fid;
        volatileState[0] |= FILE_SELECTED;                13
        return;
    }
    setNoFileSelected();
    ISOException.throwIt(ISO7816.SW_FILE_NOT_FOUND);      14
}  15
```

Figura 13: Código que permite seleccionar fichero - applet. (Creación de la autora)

Grafo de flujo

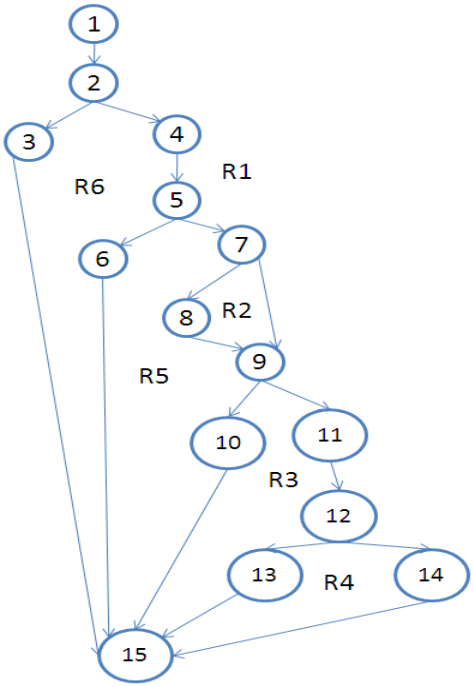

Figura 14: Caso de prueba seleccionar fichero - applet. (Creación de la autora)

Complejidad Ciclomática: Es una medida que proporciona una idea de la complejidad lógica de un programa.

La complejidad ciclomática V (G), de un grafo de flujo G, se define como:

V (G) = A – N +2

V (G) = P + 1

A: Número de aristas del grafo.

N: Número de nodos.

P: Número de nodos predicados.

V (G) = 6

La complejidad ciclomática coincide con el número de regiones del grafo de flujo.

A partir del valor de la complejidad ciclomática se obtiene el número de caminos independientes: 1-2-3-15, 1-2-4-5-6-15, 1-2-4-5-7-8-9-10-15, 1-2-4-5-7-9-10-15,1-2-4-5-7-9-11-12-13-15, 1-2-4-5-7-9-11-12-14-15.

Camino: 1-2-3-15.

Caso de prueba: Seleccionar fichero.

Entrada: Se introduce el comando APDU, del mismo se obtienen los parámetros p1 y p2, y se verifica si son los requeridos para ese comando.

Resultados: Se verifican los parámetros p1 y p2, si la verificación no es correcta se muestra el mensaje “Parámetros p1 y p2 incorrectos.”

Condiciones: Estar conectado, haber seleccionado el *applet* y haber creado el fichero.

Camino: 1-2-4-5-6-15.

Caso de prueba: Seleccionar fichero.

Entrada: Se introduce el comando APDU, del mismo se obtienen los parámetros p1 y p2 con la información requerida para ese comando, se obtiene la longitud de la respuesta (lc) y luego se verifica si no es mayor de 2 *byte*.

Resultados: Se verifica si el lc es mayor de 2 *byte*, si la verificación no es correcta se muestra el mensaje "Longitud incorrecta.".

Condiciones: Estar conectado, haber seleccionado el *applet* y haber creado el fichero.

Camino: 1-2-4-5-7-9-10-15.

Caso de prueba: Seleccionar fichero.

Entrada: Se introduce el comando APDU, del mismo se obtienen los parámetros p1 y p2 con la información requerida para ese comando. Se obtiene el lc, el cual no excede de 2 *byte*. Se obtiene el APDU y se verifica si están todos los datos en el mismo.

Resultados: Se necesitan todos los datos en un APDU, si la verificación no es correcta se muestra el mensaje "Error interno.".

Condiciones: Estar conectado, haber seleccionado el *applet* y haber creado el fichero.

Camino: 1-2-4-5-7-8-9-10-15.

Caso de prueba: Seleccionar fichero.

Entrada: Se introduce el comando APDU, del mismo se obtienen los parámetros p1 y p2 con la información requerida para ese comando. Se obtiene el lc, el cual no excede de 2 *byte*. Luego se verifica si se obtuvo el APDU anteriormente para sino obtenerlo y posteriormente se verifica si están todos los datos en el mismo.

Resultados: Se necesitan todos los datos en un APDU, si la verificación no es correcta se muestra el mensaje "Error interno.".

Condiciones: Estar conectado, haber seleccionado el *applet* y haber creado el fichero.

Camino: 1-2-4-5-7-9-11-12-13-15.

Caso de prueba: Seleccionar fichero.

Entrada: Se introduce el comando APDU, del mismo se obtienen los parámetros p1 y p2 con la información requerida para ese comando. Se obtiene el lc, el cual no excede de 2 *byte*. Luego se obtiene el identificador del fichero. Se verifica correctamente que el id del fichero no está vacío y se selecciona por éste.

Resultados: Se selecciona correctamente el fichero por el id.

Condiciones: Estar conectado, haber seleccionado el *applet* y haber creado el fichero.

Camino: 1-2-4-5-7-9-11-12-14-15

Caso de prueba: Seleccionar fichero.

Entrada: Se introduce el comando APDU, del mismo se obtienen los parámetros p1 y p2 con la información requerida para ese comando. Se obtiene el lc, el cual no excede de 2 *byte*. Luego se obtiene el identificador del fichero. Se verifica si el id del fichero está vacío, en ese caso no se selecciona el fichero.

Resultados: Se necesita que el id del fichero no esté vacío para poder seleccionarlo, si la verificación no es correcta se muestra un mensaje “Fichero no encontrado”

Condiciones: Estar conectado, haber seleccionado el *applet* y haber creado el fichero.

Caso de prueba para el código que permite saber si seleccionó un fichero, desarrollado para el *middleware.*

```
public synchronized void sendSelectFile(SecureMessagingWrapper wrapper, short fid)
        throws CardServiceException {
    CommandAPDU capdu = createSelectFileAPDU(fid); 1
    if (wrapper != null) { 2
        capdu = wrapper.wrap(capdu); 3
    }
    ResponseAPDU rapdu = transmit(capdu); 4
    if (wrapper != null) { 5
        rapdu = wrapper.unwrap(rapdu, rapdu.getBytes().length); 6
    }
    short sw = (short) rapdu.getSW(); 7
    if (sw == ISO7816.SW_FILE_NOT_FOUND) { 8
        throw new CardServiceException("File not found."); 9
    }
    if (sw != ISO7816.SW_NO_ERROR) { 10
        throw new CardServiceException("Error occured."); 11
    }
} 12
```

Figura 15: Código que permite enviar seleccionar fichero - middleware. (Creación de la autora)

Grafo de flujo

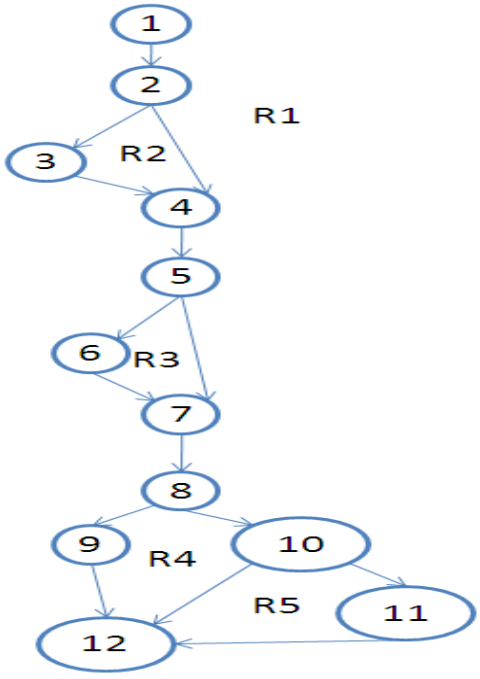

Figura 16: Caso de prueba seleccionar fichero - middleware. (Creación de la autora)

Complejidad Ciclomática: Es una medida que proporciona una idea de la complejidad lógica de un programa.

La complejidad ciclomática V (G), de un grafo de flujo G, se define como:

V (G) = A – N +2

V (G) = P + 1

A: Número de aristas del grafo.

N: Número de nodos.

P: Número de nodos predicados.

V (G) = 5

La complejidad ciclomática coincide con el número de regiones

del grafo de flujo.

A partir del valor de la complejidad ciclomática se obtiene el número de caminos independientes:1-2-4-5-7-8-10-12, 1-2-3-4-5-6-7-8-10-12, 1-2-3-6-7-8-9-12, 1-2-3-4-5-6-7-8-10-11-12, 1-2-4-5-7-8-9-12

Camino: 1-2-4-5-7-8-10-12.

Caso de prueba: Enviar seleccionar un fichero.

Entrada: Crear el comando APDU para seleccionar el fichero según su identificador. Se verifica correctamente que el comando ha sido cifrado y es enviado al *applet*, la respuesta es recibida se verifica correctamente que ha sido descifrada. Se obtiene la palabra de estado del comando APDU, que indica que el fichero ha sido seleccionado y no ha ocurrido algún error en la operación realizada.

Resultados: Se seleccionó correctamente el fichero.

Condiciones: Estar conectado, *applet* seleccionado y fichero creado.

Camino: 1-2-3-4-5-6-7-8-10-12.

Caso de prueba: Enviar seleccionar un fichero.

Entrada: Crear el comando APDU para seleccionar el fichero según su identificador. Se verifica si el comando no ha sido cifrado, en este caso se cifra su información y es enviado al *applet*. La respuesta es recibida, se verifica que el comando no ha sido descifrado, en este caso se descifra la información contenida. Se obtiene la palabra de estado del comando APDU, que indica que el fichero ha sido seleccionado y no ha ocurrido algún error en la operación realizada.

Resultados: Se seleccionó correctamente el fichero.

Condiciones: Estar conectado, *applet* seleccionado y fichero creado.

Camino: 1-2-4-5-7-8-9-10-12.

Caso de prueba: Enviar seleccionar un fichero

Entrada: Crear el comando APDU para seleccionar el fichero según su identificador. El comando APDU cifrado es enviado al *applet*. La respuesta es recibida y descifrada, se obtiene de ella la palabra de estado del comando APDU, indicando que el fichero no ha sido seleccionado, se verifica correctamente que no ha ocurrido algún error en la operación realizada.

Resultados: Si el fichero no ha sido seleccionado se muestra un mensaje “Fichero no encontrado.”.

Condiciones: Estar conectado, *applet* seleccionado y fichero creado.

Camino: 1-2-4-5-7-8-10-11-12.

Caso de prueba: Enviar seleccionar un fichero

Entrada: Crear el comando APDU para seleccionar el fichero según su identificador. El comando APDU cifrado es enviado al *applet*. La respuesta es recibida y descifrada, se obtiene de ella la palabra de estado del comando APDU indicando que el fichero ha sido seleccionado, se verifica si no ha ocurrido otro error en la operación realizada.

Resultados: Si ha ocurrido un error interno se muestra un mensaje “Ha ocurrido un error.”.

Condiciones: Estar conectado, *applet* seleccionado y fichero creado.

Camino: 1-2-4-5-7-8-9-10-11-12

Caso de prueba: Enviar seleccionar un fichero

Entrada: Crear el comando APDU para seleccionar el fichero según su identificador. El comando APDU cifrado es enviado al *applet*. La respuesta es recibida y descifrada, se obtiene de ella la palabra de estado del comando APDU. Se verifica si ha sido seleccionado el fichero o ha ocurrido algún error.

Resultados: Si la verificación es correcta en ambos casos, se muestra primeramente un mensaje “Fichero no encontrado.” y luego otro mensaje “Ha ocurrido un error.”.

Condiciones: Estar conectado, *applet* seleccionado y fichero creado.

3.2.1 Pruebas de aceptación

Las pruebas de aceptación o funcional, se realizan con el objetivo de probar que los sistemas desarrollados cumplan con las funcionalidades específicas para los cuales han sido creados. En efecto, las pruebas de aceptación corresponden a una especie de documento de requerimientos en XP, ya que marcan el camino a seguir en cada iteración indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención, permiten al cliente saber si el sistema funciona y que los programadores conozcan qué es lo que resta por hacer.[22]

A este tipo de comportamiento se les denominan también pruebas de caja negra, ya que los analistas de pruebas enfocan su atención básicamente en los casos de pruebas donde se analizan los datos de entrada y los de salida.[27]

A continuación se muestran en las tablas de la 18 a la 20 los casos de pruebas realizados a las historias de usuario que presentan una prioridad "Alta", para ello se utilizó un prototipo de interfaz de usuario que interactúa con el *middleware*, el cual define las funcionalidades que el *applet* realiza con relación a los comandos APDU, siendo evaluadas estas pruebas de satisfactorias finalmente. El resto de las tablas se incluyen en el Anexo 4.

Caso de prueba de caja negra	
Código de caso de prueba: HU1_CP1	**Nombre de la historia de usuario:** Establecer comunicación
Responsable de la prueba: Lissi Fernández Miyet	
Descripción de la prueba: Establecer comunicación con la licencia de conducción electrónica.	
Condiciones de ejecución: ✦ El lector debe estar conectado al terminal de servicio. ✦ Debe haber una tarjeta insertada en el lector.	
Entrada/Pasos de ejecución: ✦ Ejecutar la aplicación.	
Resultado esperado: Se muestran los mensajes "Lector conectado" y "Tarjeta insertada".	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 18: HU1_CP1-Establecer comunicación. (Creación de la autora)

Caso de prueba de caja negra	
Código de caso de prueba: HU2_CP2	**Nombre de la historia de usuario:** Almacenar llave semilla
Responsable de la prueba: Lissi Fernández Miyet	

Descripción de la prueba: Almacenar la llave semilla.
Condiciones de ejecución: - El lector debe estar conectado al terminal de servicio. - Debe haber una tarjeta insertada en el lector.
Entrada/Pasos de ejecución: Si es la fase de personalización de la tarjeta: - Almacenar llave semilla en el *applet* - Seleccionar si requiere o no SHA1. Si es la fase de aplicación de la tarjeta: - Almacenar llave semilla en el *middleware*, la misma que se almacenó en el *applet*. - Seleccionar si requiere o no SHA1. - Presionar botón “OK” si quiere almacenar la llave o “Cancelar” en caso contrario.
Resultado esperado: En la fase de personalización se muestra un mensaje “Personalización satisfecha”. En la fase de aplicación de la tarjeta se muestra un mensaje “BAP OK”
Evaluación de la prueba: Prueba satisfactoria.

Tabla 19: HU2_CP2-Almacenar llave semilla. (Creación de la autora)

Caso de prueba de caja negra	
Código de caso de prueba: HU3_CP3	**Nombre de la historia de usuario:** Gestionar el sistema de ficheros
Responsable de la prueba: Lissi Fernández Miyet	
Descripción de la prueba: Almacenar información en la licencia de conducción electrónica.	
Condiciones de ejecución: - El lector debe estar conectado al terminal de servicio. - Debe haber una tarjeta insertada en el lector.	

Entrada/Pasos de ejecución: ✦ Seleccionar el grupo de datos y llenar los campos. ✦ Almacenar llave semilla no menor de 16 caracteres, en caso contrario seleccionar “SHA1” ✦ Cargar certificado y llave para firmar el objeto de seguridad. ✦ Seleccionar botón “Cargar”.
Resultado esperado: Se muestra un mensaje “Personalización satisfecha”
Evaluación de la prueba: Prueba satisfactoria.

Tabla 20: HU3_CP3-Gestionar el sistema de ficheros. (Creación de la autora)

Conclusiones del Capítulo

Luego de terminadas las fases de Iteraciones a primera liberación y Producción se concluye que:

- El desglose de las historias de usuario en tareas de ingeniería, fue una buena práctica que mostró las funcionalidades específicas a implementar.
- La elaboración de la documentación y diagramas de ingeniería de software ofrecen un claro entendimiento de la solución.
- El desarrollo de la solución fue guiado y verificado por las pruebas unitarias que permitió el desarrollo y diseño del software.
- Se realizaron pruebas de aceptación para asegurar la funcionalidad del sistema concluyendo de manera exitosa.

Conclusiones Generales

Producto a la investigación llevada a cabo y para dar cumplimiento a las tareas se arribó a las siguientes conclusiones:

- La determinación de aspectos teóricos conceptuales sobre *applet* y *middleware* fueron el punto de partida de la investigación realizada.
- La asimilación del software eDL permitió que el CISED cuente con un proyecto propio de este tipo, esto hizo que se aumentara la capacidad de desarrollo de productos y soluciones en el área de documentos de identificación.
- El *applet* se corresponde con estándares y normas establecidas para licencia de conducción electrónica, permitiendo una gestión segura de la información relacionada con licencias de conducción y su integración con otros *middlewares*.
- El *middleware* establece una adecuada comunicación con las funcionalidades del *applet* embebido en la tarjeta sobre la tecnología *Java,* lo cual permite que sea usado en múltiples plataformas.
- El desarrollo guiado por pruebas aseguró el cumplimiento de los objetivos trazados en las historias de usuario.

Recomendaciones

Se exponen como recomendaciones para las siguientes fases de la solución informática:

- Implementar y probar el mecanismo de protección de acceso extendido en tarjetas que soporten los algoritmos de curva elíptica.
- Implementar la integración con un sistema de administración de llaves o (KMS) que permita la firma del SO_D de forma segura en un ambiente real de personalización.

Bibliografía Citada

1. Org.InmigrationUnitedStates. *Licencia de conducción*. 2011 [cited 2011; Available from: http://inmigracion.terra.com/licencia_conducir.html.
2. Wiley, J. and Sons, *Smart Card Handbook.* 3 ed. 2002, England.
3. ISO/IEC, *International Standard ISO/IEC 7816-4. Identification cards - Integrated circuit cards.* 2 ed. Organization, security and commands for interchange. 2005.
4. Wikimedia. *Smart Card Application*. 2011 [cited 2011; Available from: http://es.wikipedia.org/wiki/Tarjeta_inteligente.
5. Bustio, J.M.i., *Sistemas de identificación y control automáticos. Sistemas de control del flujo físico.* Vol. 2. 1994, España.
6. Chen, Z., *Java Card™ Technology for Smart Cards: Architecture and Programmer's Guide.* 1 ed. 2000.
7. Vilar, J.S. and D.A. Sotolongo, *Solución para el control de acceso a la información de las entidades externas, en la cédula de identificación electrónica de la República Bolivariana de Venezuela*, in *Facultad 1*. 2009, Universidad de las Ciencias Informáticas: Ciudad de la Habana, Cuba.
8. EBU-UER, G., *The Middleware Report. System integration in brodcast environments.* 2005.
9. Viñolo, K.P. and V.F. Santana, *Plataforma para el desarrollo de servicios en línea utilizando tarjetas inteligentes*, in *Facultad 1*. 2010, Universidad de las Ciencias Informáticas: Ciudad de la Habana, Cuba.
10. Gutiérrez, J. and J. Tena, *Protocolos Criptográficos y seguridad en redes*. 2003.
11. Gutiérrez, J. and J. Tena, *Protocolos Criptográficos y seguridad en redes*.
12. Technology, I. (2002) *Information technology - Abstract Syntax Notation One (ASN.1)*.
13. Apple Computer, I., et al., *Interoperability Specification for ICCs and Personal Computer Systems. PC/SC*, in *Introduction and Architecture Overview*. 2005.
14. Grupo-Colaborativo. *Documento de identificación.* 2004 [cited; Available from: http://www.malaysiacentral.com/information-directory/government-rules-and-politics/identification-documents/mykad-the-malaysia-government-multipurpose-smart-identity-card/.
15. Gemalto. *Electrónic Driver´s License: The New Secure ID Solution* 2010 [cited; Available from: http://www.govinfosecurity.com/podcasts.php?podcastID=879.

16. Inc.Geeknet. *ISO18013 Electronic Driving License*. 2009 [cited; Available from: http://sourceforge.net/projects/isodl/.
17. ISO/IEC, *International Standard ISO/IEC 18013-1. Information Technology - Personal identification - ISO-compliant driving licence*, in *Physical characteristics and basic data set*. 2008.
18. ISO/IEC, *International Standard ISO/IEC 18013-2. Information technology - Personal identification - ISO-compliant driving licence. Machine-readable technologies.* 2008.
19. ISO/IEC, *International Standard ISO/IEC 18013-3. Information technology - Personal identification - ISO-compliant driving licence. Access control, authentication and integrity validation.* 2009.
20. Mostowski, W. and E. Poll, *Electronic Passports in a Nutshell.*
21. Figueroa, R.G., C.J. Solís, and A.A. Cabrera, *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* 2011.
22. Joskowicz, I.J., *Reglas y Prácticas en eXtreme Programming*. 2008.
23. Larman, C., *Introducción al análisis y diseño orientado a objetos.* 1999.
24. OMG. *Introduction to OMG's Unified Modeling Language.* 2011 [cited; Available from: http://www.omg.org/gettingstarted/what_is_uml.htm.
25. Studios, G.I. *Altova Umodel*. 2011 [cited; Available from: http://download.cnet.com/Altova-UModel-Enterprise-Edition/3000-2383_4-10397824.html.
26. Hojas, L.I.H., D.G. Plaza, and M.A. Giménez (2011) *Lenguaje Java*. **Volume**,
27. Cortés, O.H.G. *Material de Caja Negra y Caja Blanca.* 2011 [cited; Available from: http://www.buenastareas.com/ensayos/Material-De-Caja-Negra-y-Caja/1819510.html.

Bibliografía Consultada

(SACALA), S.C.A.A.L., Tarjetas Inteligente y Estacionamientos. Documento de Consenso del Consejo de transporte de la Smart Card Aliance. 2006.

Association, V.I.S., Open Platform Card Specification. 2000.

ESIGN, Application interface for smart cards used as secure signature creation devices, in Additional services. 2004.

ESIGN, Application interface for smart cards used as secure signature creation devices, in Basic Requirements. 2004.

Gemalto, Gemalto extiende contrato de licencias de conducir electrónica a 3 estados adicionales de México. 2008.

Gemalto, IAS Classic Applet - Contactless. Reference Manual.

Gemalto, Java Card and STK Applet Development Guidelines.

Gemalto, MultiApp ID Combi and Derived Products. Reference Manual. 2008.

Gemplus, GemXpreso Pro R3.x. Reference Manual.

González, M.A.P., Transporte, desarrollo y calidad de vida: significado y problemas, in Revista de obras públicas. 1986.

Hilda. Concepto de licencia. 2008 [cited; Available from:
http://deconceptos.com/general/licencia.

Internacional, O.d.A.C., Documentos de viaje oficiales de lectura mecánica. 2008.

Mendoza, Y.C., Biometría en las tarjetas inteligentes.

Mostowski, W. and E. Poll, Electronic Passports in a Nutshell.

Rodriguez, L., et al., Historias Clínicas en Tarjetas Inteligentes. 2001.

SmartCardAliance.Org. Gemalto Delivers First Smart Card Driving License Program in Mexico. 2007 [cited; Available from:
http://www.smartcardalliance.org/articles/2007/06/13/gemalto-delivers-first-smart-card-driving-license-program-in-mexico.

Som, G., Criptografía práctica.

Vicente, A.S., El pasaporte electrónico.

Vilar, J.S. and D.A. Sotolongo, Solución para el control de acceso a la información de las entidades externas, en la cédula de identificación electrónica de la República Bolivariana de Venezuela, in Facultad 1. 2009, Universidad de las Ciencias Informáticas: Ciudad de la Habana, Cuba.

Viñolo, K.P. and V.F. Santana, Plataforma para el desarrollo de servicios en línea utilizando tarjetas inteligentes, in Facultad 1. 2010, Universidad de las Ciencias Informáticas: Ciudad de la Habana, Cuba.

Glosario de Términos

APDU: Protocolo de Unidad de Datos de Aplicaciones.

Applet: Aplicación que se ejecuta dentro de las tarjetas inteligentes y gestiona la información almacenada en estas.

CCI: Chip de Circuito integrado.

DF: Fichero dedicado.

EF: Fichero elemental.

FCI: Información de control de ficheros.

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

Hash: Algoritmo de Hash Seguro.

MAC: Código de Autenticación de Mensaje.

MF: Fichero maestro.

Middleware: Es una librería de software que media entre las aplicaciones que se ejecutan en la tarjeta inteligente y las que se ejecutan en una computadora.

PKCS: Estándar Criptográfico de Llave Pública.

SW: Palabra de Estado.

TLV: Etiqueta, Longitud y Valor.

UML: Lenguaje de Modelado Unificado.

Anexos

Anexo 1: Diagramas de Clases

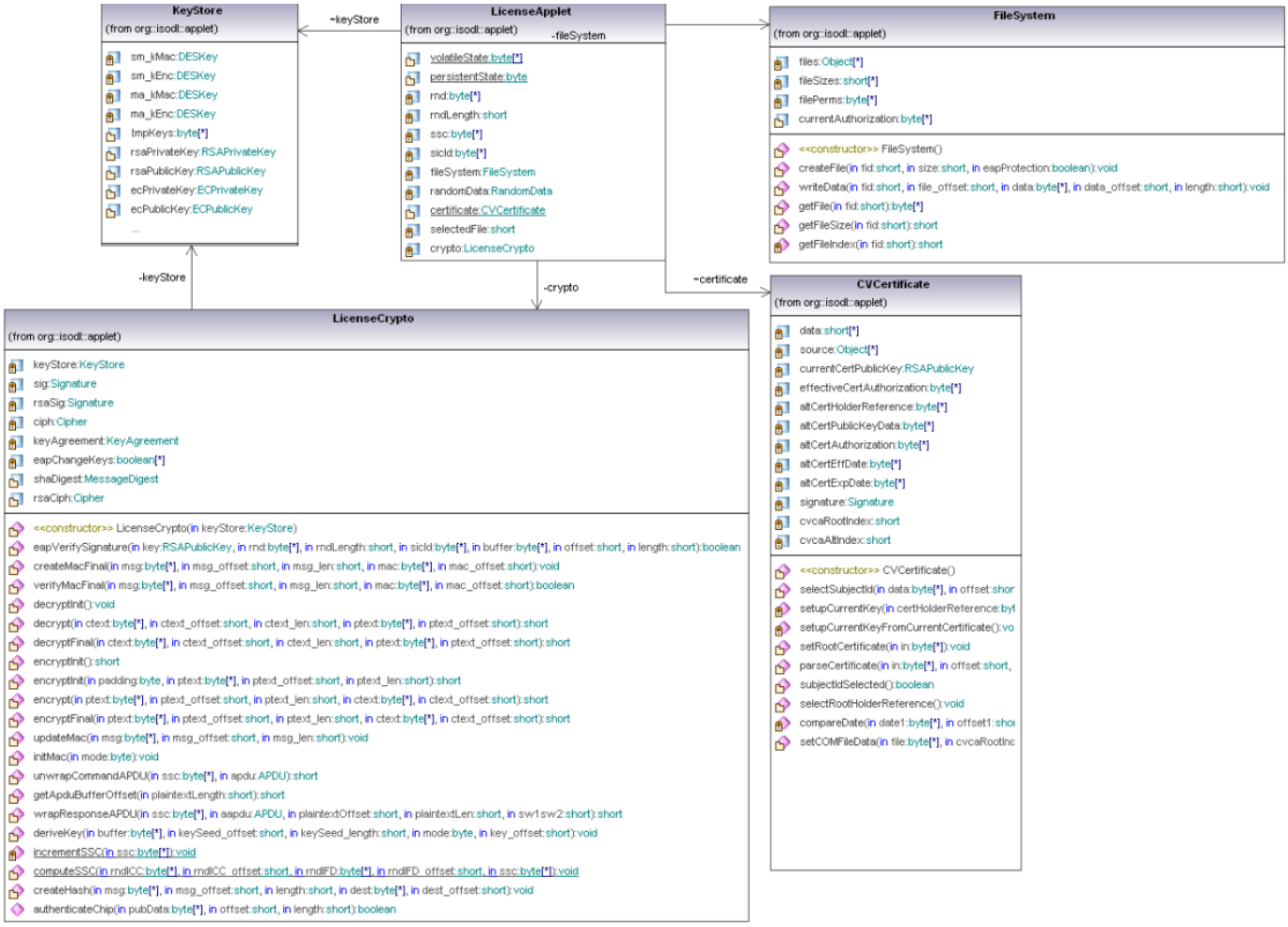

Figura 17: Diagrama de clases del applet. (Creación de la autora)

Figura 18: Diagrama de Clases del Middleware. (Creación de la autora)

Anexo 2: Tarjetas CRC.

LicenseApplet	
Responsabilidades	**Colaboradores**
- Es la clase controladora del sistema. - Selecciona, instala y procesa los comandos APDU enviados al *applet*, así como los comandos de respuestas al terminal de servicio.	- KeyStore - FileSystem - CVCertificate - Licensecrypto

Tabla 21: CRC-LicenseApplet. (Creación de la autora)

KeyStore	
Responsabilidades	**Colaboradores**
- Almacena las llaves estáticas y de sesión para realizar los mecanismos BAP y AA.	- LicenseApplet - LicenseCrypto

Tabla 22: CRC-KyeStore. (Creación de la autora)

LicenseCrypto	
Responsabilidades	**Colaboradores**
- Encargada de codificar el APDU respuesta y decodificar el comando APDU.	- KeyStore

Tabla 23: CRC-LicenseCrypto. (Creación de la autora)

FileSystem	
Responsabilidades	**Colaboradores**
- Estructura y crea el sistema de ficheros del *applet* e implementa métodos para la lectura de los mismos.	- LicenseApplet

Tabla 24: CRC-FileSystem. (Creación de la autora)

CVCertificate	
Responsabilidades	**Colaboradores**
- Realiza la verificación del certificado.	- LicenseApplet

Tabla 25: CRC-CVCertificate. (Creación de la autora)

DrivingLicense	
Responsabilidades	**Colaboradores**
- Esta clase es la que crea los grupos de ficheros. - Realiza llamadas a los mecanismos de seguridad. - Es la clase controladora del sistema.	- COMFile - SODfile - DrivingLicensePersoService

Tabla 26: CRC-DrivingLicense. (Creación de la autora)

DrivingDemographicInfo	
Responsabilidades	**Colaboradores**
- Almacena información demográfica del titular como lo es el nombre, apellidos, fecha de expedición y de expiración de la licencia, autoridad que emite la licencia entre otros.	- DG1File

Tabla 27: CRC-DrivingDemographicInfo. (Creación de la autora)

SecureMessagingWrapper	
Responsabilidades	**Colaboradores**
- Realiza un cifrado a cada comando APDU.	- DrivingLicenseService

Tabla 28: CRC-SecureMessagingWrapper. (Creación de la autora)

DataGroup	
Responsabilidades	**Colaboradores**
- Contiene la longitud y el Tag de cada *DataGroup* para acceder a estos.	- DrivingLicenseFile

Tabla 29: CRC-DataGroup. (Creación de la autora)

DrivingLicenseFile	
Responsabilidades	**Colaboradores**
- Clase base de todos los ficheros ISO/IEC	

7816-4 de los cuales heredan los grupos de datos de la ISO/IEC 18013-1 y los demás ficheros especiales (COMFile y SODFile) de este último estándar.	

Tabla 30: CRC-DrivingLicenseFile. (Creación de la autora)

CategoryInfo	
Responsabilidades	**Colaboradores**
- Obtiene las categorías de vehículos, el código y las fechas de emisión y expiración de estas.	- DG1File - DG11File

Tabla 31: CRC-CategoryInfo. (Creación de la autora)

DrivingLicensePersoService	
Responsabilidades	**Colaboradores**
- Crea la estructura y almacena los datos en la tarjeta. - Genera los datos necesarios para realizar el BAP utilizando la llave semilla.	- DrivingLicenseService

Tabla 32: CRC-DrivingLicensePersoService. (Creación de la autora)

Anexo 3: Tareas de Ingeniería.

Tarea	
Numero de Tarea: HU2_T2	**Historia de Usuario:** Almacenar llave semilla
Nombre: Almacenar llave semilla	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Responsable: Lissi Fernández Miyet	
Descripción: La llave semilla se almacena tanto en el *applet* en la fase de personalización como en el *middleware* en la fase de aplicación de la tarjeta. De esta semilla se generan las llaves estáticas para realizar el mecanismo BAP en la fase de aplicación de la tarjeta.	

Tabla 33: HU2_T2-Almacenar llave semilla. (Creación de la autora)

Tarea	
Numero de Tarea: HU3_T3	**Historia de Usuario:** Gestionar el sistema de ficheros.
Nombre: Crear un EF.SO_D para la autenticidad e integridad de los datos.	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Responsable: Lissi Fernández Miyet	
Descripción: Crear el fichero elemental SO_D, el cual debe detallar aspectos referentes a la estructura y posicionamiento de la información en el árbol jerárquico de ficheros así como elementos de seguridad.	

Tabla 34: HU3_T3-Crear un EF.SOD para la autenticidad e integridad de los datos. (Creación de la autora)

Tarea	
Numero de Tarea: HU3_T4	**Historia de Usuario:** Gestionar el sistema de ficheros.
Nombre: Definir los grupos de datos obligatorios y opcionales.	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Responsable: Lissi Fernández Miyet	
Descripción: Se define la posición y la capacidad que ocuparán los datos a almacenar en la tarjeta, estructurando los obligatorios y opcionales en ficheros específicos.	

Tabla 35: HU3_T4-Definir los grupos de datos obligatorios y opcionales. (Creación de la autora)

Tarea	
Numero de Tarea: HU3_T5	**Historia de Usuario:** Gestionar el sistema de ficheros.
Nombre: Almacenar información en el fichero.	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Responsable: Lissi Fernández Miyet	

Descripción: Una vez creado el fichero se debe seleccionar y se almacenar la información que se desea en la tarjeta.

Tabla 36: HU3_T5-Almacenar información en el fichero. (Creación de la autora)

Tarea	
Numero de Tarea: HU4_T6	**Historia de Usuario:** Efectuar BAP
Nombre: Efectuar BAP	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Responsable: Lissi Fernández Miyet	
Descripción: En el *middleware* se obtiene la llave semilla de la cual se derivan las llaves estáticas de acceso K_{ENC} y K_{MAC}, llaves que se utilizan para realizar una autenticación mutua mediante el protocolo reto-respuesta proceso necesario para establecer un canal seguro de comunicación, donde cada comando APDU se cifra con la llave de sesión KS_{ENC} y se verifica la integridad del mismo con KS_{MAC}, de forma que toda comunicación entre el terminal de servicio y la tarjeta se protegerá mediante la construcción segura de mensajes en modo Mac y Encriptado.	

Tabla 37: HU4_T6-Efectuar BAP. (Creación de la autora)

Tarea	
Numero de Tarea: HU5_T7	**Historia de Usuario:** Leer grupos de datos
Nombre: Leer grupos de datos	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Responsable: Lissi Fernández Miyet	
Descripción: Primeramente se lee el EF.COM, el cual contiene los identificadores de todos los grupos de datos personalizados en el chip. Se procede a leer cada uno de los ficheros contenidos en los grupos de datos, para ello se selecciona el fichero y se comienza a leer la información que contiene.	

Tabla 38: HU5_T7-Leer grupos de datos. (Creación de la autora)

Tarea

Numero de Tarea: HU6_T8	**Historia de Usuario:** Realizar autenticación pasiva
Nombre: Realizar autenticación pasiva	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Responsable: Lissi Fernández Miyet	
Descripción: Primero se verifica la integridad de los grupos de datos, para ello se comprueba que los *hash* que se calcula a los grupos de datos leídos con anterioridad, coincide con los *hash* que están almacenados en el SO_D. Luego se verifica la firma del SO_D utilizando la llave pública del firmante del documento obtenida del certificado.	

Tabla 39: HU6_T8-Realizar autenticación pasiva. (Creación de la autora)

Tarea	
Numero de Tarea: HU7_T9	**Historia de Usuario:** Realizar autenticación activa
Nombre: Realizar autenticación activa	
Tipo: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Responsable: Lissi Fernández Miyet	
Descripción: Este mecanismo puede ser usado para chequear la autenticidad de la licencia de conducción con un simple protocolo reto-respuesta. El terminal de servicio envía un mensaje que es firmado con la llave KPu_{AA} almacenada en el DG13, luego la tarjeta firma el mensaje con su llave privada que está almacenada de forma segura y lo envía al terminal de servicio para ser verificado. De esta forma se comprueba que el chip es genuino.	

Tabla 40: HU7_T9-Realizar autenticación activa. (Creación de la autora)

Anexo 4: Casos de prueba de caja negra.

Caso de prueba de caja negra	
Código de caso de prueba: HU4_CP4	**Nombre de la historia de usuario:** Efectuar BAP
Responsable de la prueba: Lissi Fernández Miyet	
Descripción de la prueba: Efectuar mecanismo de autenticación entre la tarjeta y el terminal de servicio.	

Condiciones de ejecución: • El lector debe estar conectado al terminal de servicio. • Debe haber una tarjeta insertada en el lector. • Debe haber insertado la llave semilla en el *middleware*.
Entrada/Pasos de ejecución: no tiene.
Resultado esperado: Se muestra el estado "BAP OK", significa que se realizó satisfactoriamente la autenticación entre la tarjeta y el terminal, reconociéndose como válidos para realizar una correcta comunicación.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 41: HU4_CP4-Efectuar BAP. (Creación de la autora)

Caso de prueba de caja negra	
Código de caso de prueba: HU5_CP5	**Nombre de la historia de usuario:** Leer grupo de datos
Responsable de la prueba: Lissi Fernández Miyet	
Descripción de la prueba: Obtener los grupos de datos almacenados en la licencia de conducción electrónica.	
Condiciones de ejecución: • El lector debe estar conectado al terminal de servicio. • Debe haber una tarjeta insertada en el lector. • Debe haber realizado el BAP.	
Entrada/Pasos de ejecución: no tiene.	
Resultado esperado: Se observan los grupos datos personalizados en la licencia de conducción electrónica, incluyendo el certificado y la llave pública que se utilizan para verificar la firma del objeto de seguridad.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 42: HU5_CP5-Leer grupos de datos. (Creación de la autora)

Caso de prueba de caja negra

Código de caso de prueba: HU6_CP6	**Nombre de la historia de usuario:** Realizar autenticación pasiva
Responsable de la prueba: Lissi Fernández Miyet	
Descripción de la prueba: Verificación de los datos.	
Condiciones de ejecución: - El lector debe estar conectado al terminal de servicio. - Debe haber una tarjeta insertada en el lector. - Debe haber realizado el BAP. - Deben haberse leídos los grupos de datos personalizados de la licencia de conducción electrónica.	
Entrada/Pasos de ejecución: no tiene.	
Resultado esperado: El *middleware* realiza la autenticación pasiva en dos pasos. Primero se verifica la integridad de los grupos de datos y se pone el estado "DI OK", significa que se realizó correctamente la verificación de la integridad del documento. Luego se verifica la firma del SO_D y se pone el estado "DS OK", significa que se realizó correctamente la verificación de la firma del documento	
Evaluación de la prueba: Prueba satisfactoria debido a que la autenticación activa no se realiza si la autenticación pasiva no se ha realizado previamente.	

Tabla 43: HU6_CP6-Realizar autenticación pasiva. (Creación de la autora)

Caso de prueba de caja negra	
Código de caso de prueba: HU7_CP7	**Nombre de la historia de usuario:** Realizar autenticación activa
Responsable de la prueba: Lissi Fernández Miyet	
Descripción de la prueba: Verificación del objeto de seguridad.	
Condiciones de ejecución: - El lector debe estar conectado al terminal de servicio. - Debe haber una tarjeta insertada en el lector.	

✦ Debe haber realizado el BAP. ✦ Deben haberse leídos los grupos de datos personalizados de la licencia de conducción electrónica. ✦ Debe haberse realizado la autenticación pasiva.
Entrada/Pasos de ejecución: no tiene.
Resultado esperado: Se muestra el estado “AA OK” , significa que se realizó correctamente la autenticación activa..
Evaluación de la prueba: Prueba satisfactoria.

Tabla 44: HU7_CP7-Realizar autenticación activa. (Creación de la autora)