Facultad
de
Ciencias

SOLUCIÓN DE FIRMA ELECTRÓNICA EN MOVILIDAD BASADA EN TARJETA INTELIGENTE

(Digital Signature on Mobility Using Smart Cards)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Javier Almaraz Valdizan

Director: Roberto Sanz Gil
Co-Director: Jorge Lanza Calderón

Junio – 2017

AGRADECIMIENTOS

Tras varios años de duro esfuerzo y sacrificio para conseguir dos de las metas que tenía en mi vida, obtener el empleo de Guardia Civil y conseguir el título como Graduado en Ingeniería Informática, hoy puedo decir que llego la hora, de finalizar, quiero recordar y agradecer a todos aquellos que estuvieron a mi lado apoyándome tanto en los momentos buenos como en los malos y gracias a los cuales hoy puedo decir que fue duro pero lo conseguí.

En primer lugar, quiero agradecer a la Universidad de Cantabria, a la Facultad de Ciencias y por supuesto al gran equipo de docentes que en ella trabaja, por todo los conocimientos tanto personales como profesionales que tuve la oportunidad de aprender, lo cuales me ayudaron a crecer en mi vida personal y particular.

A mi tutor del Trabajo de fin de Grado Jorge Lanza, ya que estos dos últimos años sé que no ha sido fácil tener que lidiar con el proyecto de un alumno que se encontraba en Madrid estudiando una oposición el primer año y el segundo en una academia con régimen interno. Pese a todo esto el no tiro la toalla conmigo y se preocupaba tanto por las oposiciones como por mi trabajo, por todo esto quiero decir que no tengo las palabras suficientes para agradecerle toda esta preocupación.

A mis compañeros de promoción, gracias a los cuales hacían que el día a día se hiciese más ameno y que ir a clase a las ocho de la mañana costase algo menos, gracias por esos buenos ratos pasados tanto en los momentos de estudio, clases como en nuestro tiempo de ocio.

A mis padres, Luis Javier y María Jesús, por educarme y formarme como persona y sobre todo quiero agradecerles todos los esfuerzos que han tenido que realizar durante estos años para que yo pudiese conseguir esto. Ellos siempre confiaron y me animaron a seguir en el día a día, también agradecerles por aguantarme mis dotes pesimistas cada vez que realizaba un examen o alguna prueba importante de las cuales nunca me habían salido bien. Agradecerles todos los consejos a lo largo de mi paso por esta etapa y a mi madre por llenar mi nevera con los famosos *tuppers* que casi todo universitario tiene.

Al resto de mi familia, que siempre me apoyo al igual que mis padres, y estuvieron en todo aquello que me pudiesen ayudar.

A mi pareja Mar Santiago, por todo el tiempo junto a ti y sobre todo por todo tu apoyo y consejo en los días duros donde te dan ganas a dejarlo todo.

A todos aquellos que no recordé citar, gracias de corazón.

RESUMEN

En la actualidad, la seguridad informática es uno de los campos más importantes. Ya que el crecimiento de los dispositivos inteligentes uso de estos por las personas físicas en actividades cotidianas tanto a nivel laboral como personal ha sufrido un elevado aumento en los últimos años. Cualquier persona a lo largo de un día utiliza varios dispositivos inteligentes. De aquí la importancia de la seguridad informática, con la cual se intenta que a ningún tipo de persona le sean violados sus derechos constitucionales.

Es importante también mencionar que cada vez existen más usuarios maliciosos que tratan de obtener beneficios ilícitamente mediante robo de datos, estafas, etc.

Este proyecto se ha realizado con el fin de disminuir la brecha que existe entre la firma electrónica y los usuarios menos formados en las nuevas tecnologías. Así se ha diseñado e implementado una solución funcional que de una forma sencilla e intuitiva permite a los usuarios realizar firma electrónica de documentos. Con esto se intenta poner otro nivel más de seguridad antes de que se produzca cualquier tipo de delito en el ciberespacio.

La tarjeta inteligente es considerada uno de los dispositivos más seguros en la actualidad. Por este motivo, cada usuario dispondrá de una tarjeta única, personal e intransferible en la cual porta las claves que le identificará y donde se ejecutarán los algoritmos y procedimientos criptográficos necesarios.

Palabras clave: Tarjeta inteligente, Firma digital, JavaCard, NFC.

ABSTRACT

Nowadays computer security is one of the most important study fields. The growth of intelligent devices and their use by people in their jobs and their daily life have suffered a huge increase in the last few years. Everybody uses several intelligent devices along their day. In this sense, computer security has also gain relevance, in order to secure the communications but also guarantee people's privacy and rights.

It is important to note that this new environment has led to the appearance of malicious users who try to illicitly obtain benefits by user data theft, scamming or other criminal options.

This project has been developed with the aim of designing a solution that guarantees user identity validity when performing actions such as sending documents, etc. Digital signature is the most commonly used mean to achieve this, but the lack of knowledge on the related technologies makes it difficult to not well-trained users to properly use it. With this objective in mind, we have proposed a solution that makes it easy and intuitive to perform digital signature, therefore increasing the levels of security and avoiding potential cybercrime.

The use of smart card is envisage as they are consider one of the most secure devices available. A person will own an unique, personal and non-transferrable smart card which carries the codes and algorithms for user authentication and digitial signature.

Key words: SmartCard, Electronic signature, JavaCard, NFC.

ÍNDICE

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

El inicio del nuevo milenio, acrecentado durante los últimos años, ha supuesto una gran aceleración en el crecimiento del uso de las nuevas tecnologías, que hoy en día se han convertido en un elemento más de nuestra vida cotidiana. Dispositivos englobados bajo el término *smart* (inteligente) como teléfonos inteligentes (*smartphones*), relojes inteligentes (*smartwatch*), pulseras cuantificadoras (*smartbands*), etc. se han concebido con el objetivo proveer inmediatez en el acceso a la información y tratar de facilitar nuestro día a día.

La aparición de estos elementos busca dar respuesta a la necesidad impuesta por una sociedad que desea estar permanentemente conectada. La comunicación de todos ellos entre sí crea un gran flujo de información, prácticamente continuo, en el que la importancia de aspectos de seguridad tales como la identificación o la confidencialidad quedan en un segundo plano, primando otros aspectos como la accesibilidad.

Este modelo de uso supone grandes riesgos y puede suponer el acceso a información privada y/o confidencial. Las numerosas vulnerabilidades existentes deben ser tenidas en cuenta y solventadas en el corto plazo. Existen numerosas soluciones que permiten, sino solventar completamente esos problemas, si reducir el impacto de acciones malintencionadas y los potenciales riesgos asociados.

Partiendo de lo anterior, el dispositivo que mayor impacto ha tenido en la sociedad y cuyo grado de implantación es más elevado es el teléfono inteligente. La mayoría de la gente lo utiliza para el intercambio y acceso rápido a información. Algunas de estas aplicaciones (p. ej. el envío y recepción de documentos) requieren acreditar tanto la identidad en el acceso como la identidad del generador y autenticidad de la información. Gracias a las soluciones criptográficas basadas en sistemas de clave pública es posible dar respuesta a estas necesidades. Sin embargo, el almacenaje, distribución y uso seguro de las credenciales de los usuarios se realiza usualmente empleando elementos software, expuestos en muchos casos a grandes riesgos de seguridad (p. ej. virus, troyanos, etc.).

El empleo de elementos hardware como las tarjetas inteligentes (*smartcards*) permite incrementar notablemente los niveles de seguridad y confianza en los procesos, al tiempo que permite un uso amigable. Si bien este tipo de dispositivos no tenían una

elevada implantación, el despliegue de soluciones basadas en tarjetas inteligentes sin contactos en entornos de transporte público o la distribución del nuevo DNI electrónico sobre soporte tarjeta inteligente ha permitido acercar la tecnología al ciudadano.

Las sinergias existentes entre el entorno de los teléfonos móviles y las tarjetas inteligentes permiten y permitirán cada vez más el desarrollo y acceso a soluciones seguras en cualquier lugar y momento. La tecnología Near Field Communication (NFC) está haciendo realidad este nuevo ecosistema que permitirá identificar de forma segura a los usuarios mediante sistemas de autenticación en dos factores. Además, los mismos soportes, sin gastos adicionales para el usuario final, permitirán firmar y cifrar documentos con todas las garantías.

1.1 Objetivos y motivación

Todo lo anterior, unido a la reciente entrada en vigor el 2 de octubre de 2016 de la nueva ley de procedimiento administrativo 39/15 [2], la cual como uno de sus objetivos trata de conseguir una transición a la era digital en la administración pública. Buscando eliminar el papel y tratando de extender la firma electrónica a cualquier trámite, motivan el presente trabajo.

El objetivo principal de este trabajo de fin de grado será por tanto diseñar y crear una estructura en la que se interconecten diferentes dispositivos para que cumplan las siguientes funciones:

- Evitar la suplantación de identidad de la persona hacia la que va dirigido el documento que se pretende firmar o bien acreditar la identidad de la persona que envía un documento
- Permitir a todo usuario, poder utilizar su firma electrónica única y personal desde una smartcard.
- Reducir la dificultad del uso de la firma electrónica de las smartcards.
- Reutilizar el hardware disponible por los usuarios, es decir, sus teléfonos inteligentes para la realización de la firma electrónica.

1.2 Contenido de la memoria

Esta memoria se ha estructurado en varios capítulos, que se describen brevemente a continuación.

El Capítulo 1 Introducción introduce este trabajo y describe los motivos de su por qué y los objetivos que se buscan.

El Capítulo 2 Criptografía realiza un estudio de la criptografía haciendo más hincapié en la criptografía moderna. A consiguiente se detallan los conceptos de firma electrónica y certificados digitales.

El Capítulo 3 Tarjetas inteligentes se describe la tecnología de tarjeta inteligente, incluyendo los diferentes tipos, arquitectura interna, comandos y procedimiento de

intercambio de información. Así mismo detalla los mecanismos para la programación y ejecución de aplicaciones en las tarjetas.

El Capítulo 4 Diseño muestra la arquitectura de alto nivel de la solución planteada. Tras un planteamiento general de la visión de la solución, mediante diagramas de bloques y sus interrelaciones se profundiza en la operativa del sistema como paso previo a su implementación.

El Capítulo 5 Desarrollo explica cómo se ha implementado todo el sistema para poder cumplir los objetivos impuestos. Presenta las herramientas empleadas en el desarrollo de la solución, para a continuación, desglosar el proceso de implementación de cada una de las partes que componen la solución. Incluye también el proceso de integración y evaluación del sistema completo.

Por último, el Capítulo 6 Conclusiones y Líneas futuras resume el trabajo realizado y enumera las posibles evoluciones futuras para añadir más funcionalidades o mejorar las ya implementadas.

CRIPTOGRAFÍA 2

Con el aumento del uso de las nuevas tecnologías en la vida de las personas nace la necesidad de securizar la información. En la época anterior al uso de los ordenadores se utilizaban documentos físicos, donde la seguridad se ponía a través de medios físico. En los documentos digitales se necesita poder verificar la integridad, identidad y confidencialidad, combinando estas tres necesidades en función de la sensibilidad de la información. Con el objeto de conseguir solventar estos problemas con los documentos digitales nace la criptografía moderna.

2.1 CRIPTOGRAFÍA

La criptografía es la ciencia que estudia como diseñar dispositivos o funciones, capaces de cifrar mensajes legibles tratando de ocultar su verdadero significado, de forma que éstos puedan ser descifrados por un usuario legítimo del mensaje.

El estudio de la criptografía se puede dividir en dos partes, clásica y moderna. La clásica es toda aquella utilizada anterior a la época digital que se está viviendo. Era utilizada especialmente en épocas de guerra, con el objetivo de que si un enemigo interceptaba un mensaje no pudiese averiguar el contenido del mismo.

Por su parte, la criptográfica moderna en la actualidad se divide en dos grandes bloques, simétrica y asimétrica. A continuación se explican con más detalle estos tipos de criptografía moderna.

2.1.1 CRIPTOGRAFÍA SIMÉTRICA.

En este tipo de criptografía se posee una llave secreta, mediante la cual a través de una función matemática se cifra y descifra el mensaje, tal como se aprecia en la Figura 2.1 Cifrado Criptografía simétrica. La criptógrafa simétrica tiene el problema de que todo usuario legítimo del mensaje necesita la clave secreta, residiendo la potencial debilidad en su difusión usando un entorno no seguro.

Figura 2.1 Cifrado Criptografía simétrica.

2.1.2 Criptografía asimétrica.

La criptografía asimétrica basa su funcionamiento en dos claves, una pública y otra privada. Esta última no debe ser difundida bajo ningún concepto y debe ser guardada de forma segura, ya que es personal y única.

La operativa de este tipo de criptografía se describe en la Figura 2.2 Cifrado Criptografía asimétrica.. Para cifrar un mensaje se emplea la clave privada, la cual únicamente tiene el emisor en su poder, aplicando un algoritmo criptográfico sobre la información. El destinatario empleará la clave pública, previamente proporcionada por el remitente, para descifrar el mensaje recibido y tener acceso a la información en él codificada.

Figura 2.2 Cifrado Criptografía asimétrica.

La autenticación del emisor e integridad del documento se consigue generando en primer lugar un hash del documento a enviar, que seguidamente se firma con la clave privada del usuario. Esta información se adjunta al mensaje. Una vez el usuario destinatario reciba el mensaje, por un lado generara el hash de la parte útil del mensaje y por otro empleando la clave pública descifrará la firma obteniendo el hash o resumen original. El resultado de ambas operaciones se compara, siendo éste el indicador de que el mensaje recibido no ha sufrido ningún tipo de manipulación por el camino. Este proceso se describe en la Figura 2.3 Autenticación e integridad criptografía asimétrica..

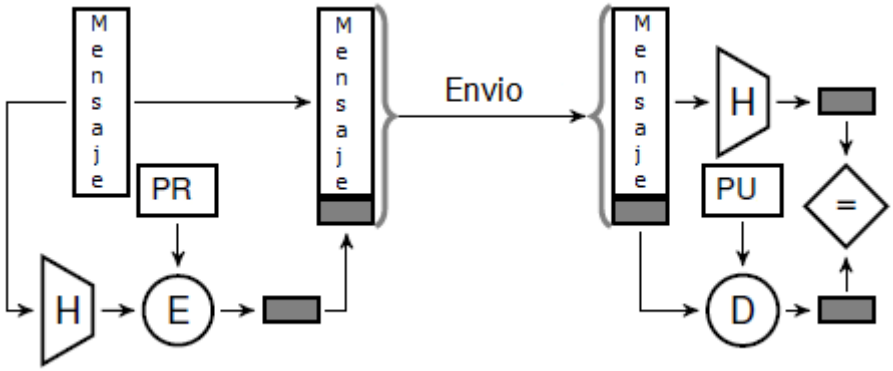

Figura 2.3 Autenticación e integridad criptografía asimétrica.

Este tipo de criptografía es más robusta y versátil que la simétrica, pero tiene el inconveniente de que el tiempo necesario para realizar las operaciones criptográficas es mayor.

2.2 Firma electrónica

La firma electrónica es una cadena de caracteres generada por un algoritmo matemático al cual se le pasa como parámetros un documento digital y una clave privada de un usuario. Proporciona de una forma segura, a la vez que sencilla, el poder comprobar tanto la identidad de la persona remitente como la integridad del mensaje recibido.

Según la ley 59/2003[1], la cual regula la firma electrónica en España, la firma electrónica es equivalente a la firma manuscrita de una persona, pero es necesario que sea una firma electrónica reconocida. Se denomina firma electrónica reconocida aquella firma electrónica que haya sido expedida por una unidad certificadora autorizada mediante un dispositivo seguro. Un dispositivo seguro puede ser el caso de una tarjeta inteligente en la que las claves nunca serán exportadas. Así la ley española define tres modelos de firma electrónica:

- Firma electrónica como conjunto de datos con forma electrónica utilizados para acreditar la entidad del firmante.
- Firma electrónica avanzada permite identificar al firmante y detectar cualquier modificación, creada por medios que el firmante puede mantener siempre bajo su control.
- Firma electrónica reconocida es la firma electrónica avanzada basada en un certificado reconocido generado en un dispositivo seguro.

2.3 Certificados digitales.

Un certificado digital es el simil en el mundo digital a una tarjeta de identificación personal física. Es utilizado para acreditar la identidad de una persona.

Un certificado confirma que una clave pública pertenece a la persona que quiere autenticarse. Para que un certificado digital goce de las garantías suficientes a la hora de acreditar una identidad es necesario que haya sido expedido y firmado por una autoridad certificadora previamente autorizada, no gozando de esta validez si uno firma su propio certificado.

TARJETA INTELIGENTE 3

Desde la aparición de la primera tarjeta de plástico a mediados del siglo XX hasta nuestros días, éstas han evolucionado adaptándose a las necesidades que el mercado ha ido requiriendo.

Las primeras tarjetas plásticas constaban de una banda magnética como mecanismo de almacenaje de datos. Estas tarjetas fueron utilizadas como métodos de pago o para acreditar la identidad de una persona. En dicha banda magnética se grababan datos, los cuales podían ser leídos con los lectores adecuados. El mayor inconveniente de estas tarjetas, antecesoras de las actuales tarjetas inteligentes, es que estos datos podían ser leídos por usuarios no autorizados, por lo que con el paso del tiempo fue necesario evolucionar para dotar a estos sistemas de un grado más de seguridad.

No es hasta 1968 cuando dos científicos alemanes Jürgen Dethloff y Helmut Grötrupp incorporan el primer circuito integrado en una tarjeta plástica [5]. Un par de años más tarde en 1970 el científico japonés K.Arimura sería el encargado de introducir esta vez en la tarjeta plástica la lógica aritmética y el almacenaje en chip, lo que podría denominarse como la primera tarjeta inteligente. De esta forma lograba atajarse el problema ocasionado en las anteriores tarjetas respecto al acceso no autorizado de los datos. Este hecho supone un gran avance y será el precursor de los futuros usos y desarrollos de las tarjetas inteligentes.

Sin embargo, no sería hasta a partir de los años 80 cuando empieza la comercialización masiva de las tarjetas inteligentes, también llamadas smartcards. En sus inicios se enfocaron al uso en las cabinas telefónicas. Su eclosión coincide con el despliegue de las redes móviles GSM en los años 90, pues todo móvil GSM debería incluir una tarjeta como elemento identificador del abonado. Adicionalmente, las tarjetas inteligentes son muy populares en entornos financieros y de pagos en transporte público.

Los avances tecnológicos producidos en las tecnologías asociadas a los circuitos integrados o embebidos han supuesto que las tarjetas inteligentes hayan aumentado de manera muy significativa su potencial. Permitiendo en un mismo espacio almacenar más

datos y ejecutar programas más complejos y extensos en un menor tiempo. Gracias a todo esto se han hecho especialmente atractivas para la sociedad.

3.1 TIPOS DE TARJETA CON CIRCUITO INTEGRADO

Una tarjeta con circuito integrado es aquella que posee una serie de conexiones electrónicas que le permiten el almacenaje de datos y procesamiento de los mismos. Este tipo de tarjetas poseen una serie de ventajas respecto a las tarjetas de banda magnética:

- Pueden almacenar mayor volumen de datos.
- Son más seguras a la hora de evitar accesos no autorizados.
- Los datos almacenados son menos vulnerables a la hora de ser modificados por fuentes exteriores.

A menudo es frecuente llamar a toda tarjeta plástica con un chip tarjeta inteligente. Esto es un término erróneo, ya que como veremos a continuación no todas las tarjetas con un chip son tarjetas inteligentes. Según el tipo de circuito integrado podemos dividir las tarjetas en dos grandes grupos, tarjetas de memoria y tarjetas inteligentes.

Las tarjetas de memoria permiten únicamente el almacenado de una serie de datos. Pudiendo encontrarlas con seguridad en el acceso a los datos o sin seguridad. Un ejemplo de este tipo de tarjetas son las tarjetas de saldo utilizadas en las cabinas telefónicas.

Por su parte, las tarjetas inteligentes se caracterizan por tener embebido en el plástico un microprocesador, el cual va a permitir operar con los datos almacenados o bien con datos aportados por un dispositivo exterior. Este microprocesador puede incluso proporcionar la opción de cargar diferentes programas en una misma tarjeta, ofreciendo una gran flexibilidad a la hora de usar una misma tarjeta en diferentes campos y usos.

A la vez este tipo de tarjetas también se divide en otros dos tipos de tarjetas según la forma en la que se traspasan los datos entre la tarjeta y el dispositivo externo:

- Tarjetas con contactos

Este tipo de tarjetas se caracteriza por necesitar un contacto físico con un lector para poder transmitir los datos o comandos con el dispositivo que se quiera comunicar.

- Tarjetas sin contactos

Estas son las más modernas y su principal ventaja es que no requieren de un contacto físico para utilizarlas, es decir, pueden operar simplemente aproximándolas al lector. La transferencia de información se realiza a través de ondas electromagnéticas. El inconveniente que tiene este tipo de tarjetas es que el campo de actuación de las ondas electromagnéticas es muy reducido.

Las tarjetas que implementan ambas interfaces de comunicación se denominan tarjetas de interfaz dual.

Las tarjetas inteligentes están en auge en los últimos años, ya que están siendo utilizadas a diario por una gran cantidad de personas con usos muy diversos, gracias a la sencillez y rapidez de uso. Algunos de ellos son: tarjetas monedero y/o financieras, tarjetas de transporte público, control de accesos.

3.2 ARQUITECTURA

La arquitectura de las tarjetas inteligentes, mostrada en la Figura 3.1 Arquitectura de la tarjeta inteligente., consta de cuatro piezas fundamentales:

- La **Central Process Unit** o **CPU** encargada de toda la inteligencia en cuanto a cálculos y procesamiento de la información necesitada. Normalmente suelen ser de 8 bits, pero pueden llegar hasta 32 bits dependiendo del uso y la complejidad a la que se vaya a destinar la tarjeta.
- **Read Only Memory** o **ROM** esta memoria se graba durante la fabricación y únicamente nos sirve de lectura, en ella se encuentra almacenado el sistema operativo.
- **Random Access Memory** o **RAM** se trata de una memoria volátil, en la que se almacenaran los datos de las operaciones realizadas en lo que la tarjeta posea energía.
- **Electrically Erasable And Programable Read Only Memory** o **EEPROM** es una memoria no volátil, en ella es donde se almacenaran todos los datos o programas cargados por el usuario, que se quieren utilizar de una forma continuada sin necesidad de cargarlo en cada uso.

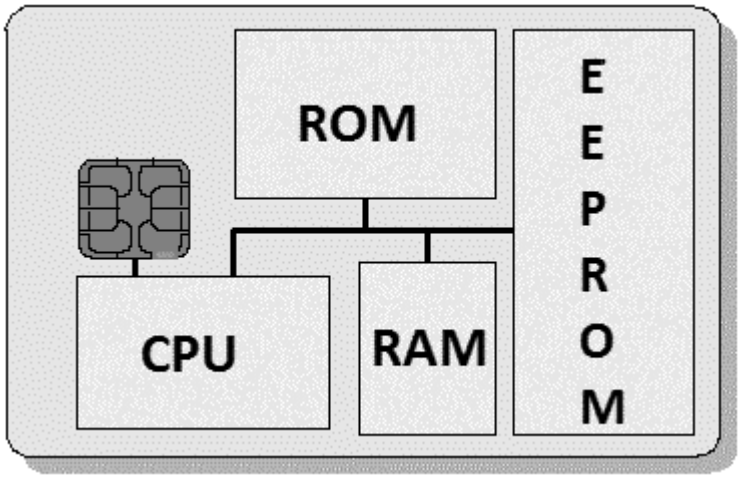

Figura 3.1 Arquitectura de la tarjeta inteligente.

3.3 Características físicas

Una de las principales características físicas de este tipo de dispositivos es su tamaño y forma. El aspecto físico de las tarjetas viene estandarizado por los ISO 7816-1 [13] y 7816-2 [13], mostrados en la Figura 3.2 SIM Stander, Mini SIM, Micro Sim, Nano Simy detallada en la Tabla 3.1 Características de las tarjetas. Los formatos más utilizados son los siguientes:

- ID-1: El utilizado en las tarjetas bancarias, DNI, etc.
- ID-000: El empleado en la tarjeta SIM utilizadas en la telefonía móvil.

En estos últimos años con la aparición de los smartphone y el empeño en reducir el tamaño de las cosas, las tarjetas MicroSim y NanoSim han ido ganando fuerza en el mercado. Este tipo de tarjeta también está estandarizado por ETSI en el estándar TS 102 221 [7].

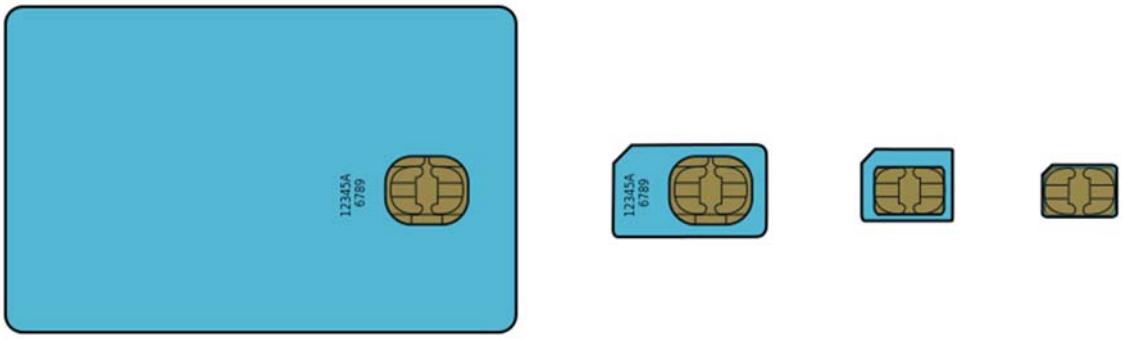

Figura 3.2 SIM Stander, Mini SIM, Micro Sim, Nano Sim

Tipo de tarjeta	Largo(mm)	Ancho(mm)	Grosor(mm)
SIM estándar	85.6	54	0.76
Mini SIM	25	15	0.76
Micro SIM	15	12	0.76
Nano SIM	12.30	8.80	0.67

Tabla 3.1 Características de las tarjetas

3.4 Comunicación

Todo tipo de comunicación entre una tarjeta inteligente y un dispositivo exterior es iniciada siempre por el dispositivo externo. Esto quiere decir que una tarjeta se limita a contestar las peticiones de otro dispositivo, tal y como se muestra en la Figura 3.3

Modelo de comunicación. Por tanto estamos ante una relación entre los dispositivos de maestro-esclavo, en la que la tarjeta actúa como esclavo y el dispositivo externo como el maestro.

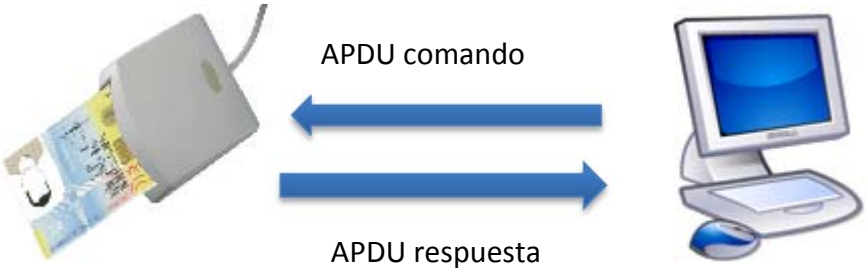

Figura 3.3 Modelo de comunicación

La transferencia de información se realiza sobre un canal de comunicación half-duplex empleando APDUs (Application Protocal Data Unit). La estructura de una APDU, la cual se describe en la Figura 3.4 Formato de las APDUs, viene recogida en el estándar ISO 7816-4[13]. Se definen dos tipos de APDU: APDU comando, enviada por el dispositivo externo y en la que se encapsulan las instrucciones y los datos necesarios para instar a la tarjeta a realizar las operaciones, y APDU respuesta remitida desde la tarjeta inteligente hacia el dispositivo externo como respuesta a la anterior conteniendo los datos generados por la ejecución del comando.

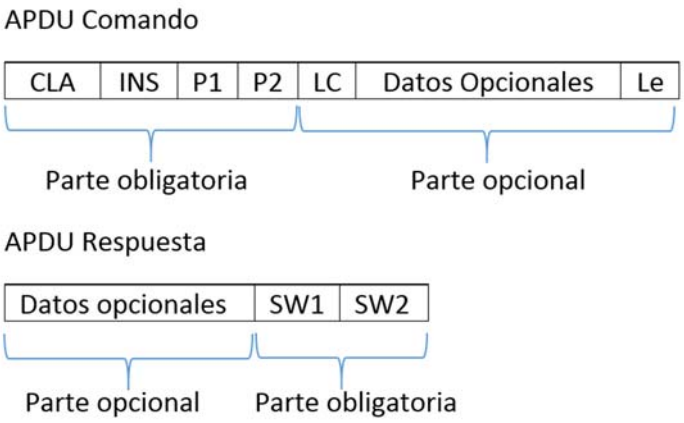

Figura 3.4 Formato de las APDUs

Los diferentes campos de las APDU comando son los siguientes:

- CLA (1 byte): indica el tipo de clase que se requiere.
- INS (1 byte): indica el tipo de instrucción específica dentro de la aplicación seleccionada.
- P1 y P2 (1 byte para cada uno): actúan como parámetros de entrada a la instrucción seleccionada.
- Lc (1 byte): indica la longitud de los datos que se transfieren en la APDU.
- Datos opcionales (longitud variable): campo utilizado para encapsular los datos.
- Le (1 byte): indica la longitud que debe tomar la APDU de respuesta enviada por la tarjeta.

Por su parte los campos de la APDU respuesta son:

- Datos opcionales (longitud variable): incluye los datos que va a retornar la operación realizada, siempre y cuando los genere. Su tamaño va a depender del campo Le de la APDU comando.
- SW1 y SW2 (1 byte para cada uno): Status Word indica el estado de la ejecución de la operación. Los valores más usuales vienen definidos en la ISO 7816-4.

3.5 SISTEMA OPERATIVO

Los sistemas operativos como norma general se colocan entre el hardware del dispositivo y las aplicaciones del usuario y tienen como misión administrar los recursos y gestionar la ejecución de las aplicaciones.

En el caso de las tarjetas inteligentes el sistema operativo viene grabado de fábrica en la memoria ROM, por lo que los usuarios no tienen opción de modificarlo. Esto evita que se puedan producir vulnerabilidades porque usuarios no autorizados lo modifiquen. Se pueden definir dos tipos de sistemas operativos en función de las aplicaciones que son capaces de correr sobre ellos, hablando entonces de sistemas operativos cerrados y abiertos, o mono-aplicación y multi-aplicación.

Los sistemas operativos cerrados como norma general suele llevar un hardware específico para su función, por lo que ejecutan sus aplicaciones de una manera más eficiente. Por otro lado los sistemas operativos abiertos suelen ser más flexibles a la hora de ejecutar diferentes aplicaciones. Sin embargo, y aunque en la actualidad las diferencias se han visto reducidas notablemente, las tarjetas que usan estos sistemas operativos suelen ser menos eficientes, ya que el hardware no está diseñado exclusivamente para una función y el código utilizado necesita ser interpretado por el sistema.

Algunos de los sistemas operativos más utilizados en las tarjetas inteligentes son Java Card, MultOS, Cyberflex, StarCOS, MFC, etc. El sistema operativo Java Card es el más utilizado en la actualidad por tener una mayor comunidad de usuarios detrás de Java.

3.6 JAVA CARD

Antiguamente cada fabricante de tarjetas inteligentes desarrollaba completamente el sistema operativo haciéndolo específico para la aplicación a la que estaba orientado. Esto ocasionaba la necesidad de un alto grado de especialización, y la problemática de portabilidad entre unas tarjetas y otras. Aunque en este campo muchos aspectos se encuentran estandarizados, la estructura interna de las tarjetas aún sigue dependiendo del propio fabricante, por lo que en muchos casos las aplicaciones solo se pueden correr en única plataforma. La erupción de la tecnología Java Card pretende proporcionar una solución a este problema, permitiendo ejecutar las aplicaciones desarrolladas con este lenguaje en tarjetas de diferentes fabricantes. Java Card se define

como una tecnología segura, portátil, multiplicación la cual aporta las ventajas del lenguaje de programación Java:

- Facilidad a la hora de desarrollar aplicación por su gran parecido a Java.
- Seguridad.
- Independencia del hardware.
- Soporte multiplicación.
- Soporte del estándar ISO 7816.

Java como lenguaje orientado a objetos proporciona un alto nivel de abstracción. Pero todo esto tiene el inconveniente que las aplicaciones Java son ejecutadas sobre su propio intérprete, lo que las hace perder rendimiento y un gran uso de recursos. La ejecución de aplicaciones Java en entornos tan limitados de recursos como las tarjetas inteligentes es un tanto sorprendente. Sin embargo, su alto grado de portabilidad hace que haya sido aceptado por la mayoría de los desarrolladores y fabricantes. La similitud entre Java y Java Card, ha permitido que la gran comunidad de usuarios con conocimientos en este lenguaje puedan extender su experiencia al mundo de las tarjetas inteligentes.

La primera vez que se intentó aplicar Java en las tarjetas inteligentes data del 1996, cuando un equipo de ingenieros de Schlumberger propone la Application Programing Interface (API) que consistía en un Java optimizado para ejecutar en las tarjetas inteligentes. Esta API se podría considerar el nacimiento de Java Card. Unos meses más tarde, varios fabricantes de tarjetas se unen al Java Card Forum con la intención de mejorar y promover el uso de esta tecnología. Un año más tarde en 1997, Sun Microsystems lanza Java Card 2.0, la cual incluía algunas funciones criptográficas y la mayoría de las capacidades soportadas en Java.

Java Card [14] establece un sistema multiplicación, en el cual pueden convivir varias aplicaciones en un mismo chip de una tarjeta. En torno a 1998 VISA introduce en sus tarjetas OpenPlatform [18], a la cual daría acceso a la comunidad Java Card en 1999. OpenPlatform es una arquitectura para la gestión de entornos multiaplicación implementada en las tarjetas, permitiendo cargar e instalar diferentes aplicaciones de una manera segura. Hoy en día su extensión GlobalPlatform [11] se ha convertido en un estándar de facto, hasta llegar al punto de que es incorporado en los chips a la hora de su fabricación [5].

En 1999 se lanza la siguiente evolución JavaCard 2.1 [16], introduciendo esta vez alguna funcionalidad en el campo de la seguridad, como mecanismos *firewall* entre las diferentes aplicaciones alojadas en la tarjeta.

La siguiente evolución Java Card 2.2 introduce la gestión para abrir varios canales de comunicación entre el dispositivo exterior y la tarjeta. También mejora la manera de invocar y eliminar los objetos respecto a las versiones anteriores. De esta versión existen diferentes evoluciones la 2.2.1 y 2.2.2 [14].

La última versión Java Card 3.0 data de 2008, y en esta se introducen novedades para soportar conexiones HTTP y HTTPS.

3.6.1 JAVA CARD VIRTUAL MACHINE

La Java Virtual Machine (JVM) puede considerarse un emulador software de un procesador Java en el que se van a interpretar todo el conjunto de instrucciones Java en un lenguaje maquina propio. Este emulador puede correr en cualquier sistema real.

Java Card se ha desarrollado a través de la evolución de Java intentando dar seguridad durante todo el proceso de desarrollo de las aplicaciones. Por la gran limitación de recursos que tienen las tarjetas inteligentes, la Java Card Virtual Machine (JCVM) introduce una serie de limitaciones con respecto a la JVM.

La principal diferencia entre JVM y JCVM es que esta última se encuentra dividida en dos partes, una primera fuera de la tarjeta (off-card) situada en un PC u otro dispositivo y la segunda parte dentro de la propia tarjeta (on-card), como a continuación se muestra en la Figura 3.5 Máquina virtual Java Card..

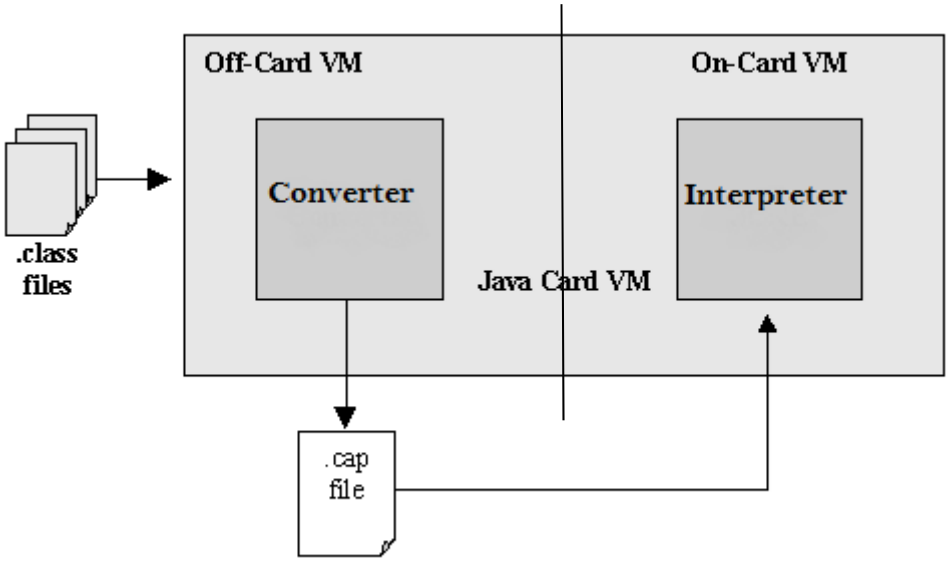

Figura 3.5 Máquina virtual Java Card.

El principal componente de la parte off-card de la máquina virtual Java Card es el converter. Tiene como principal misión la compilación y optimización del código. Estas operaciones no son necesarias realizarlas en tiempo de ejecución por lo que pueden desarrollarse en un entorno exterior sin ninguna limitación de recursos. Una vez el converter finaliza todas sus tareas se genera un archivo CAP (Card Aplication file), que contiene el código binario necesario para ejecutar la aplicación en la parte on-card de la JCVM.

La parte on-card de la JCVM es la responsable de interpretar el código binario una vez se ha cargado en la tarjeta. Este código es transformado al lenguaje específico del procesador subyacente, permitiendo la ejecución de la aplicación. Además durante el tiempo de ejecución también se encarga de realizar un control de la memoria, la creación de objetos y garantizar que todo ello se va a realizar en un entorno seguro.

3.6.2 JAVA CARD RUNTIME ENVIROMENT

Java Card Runtime Environment (JCRE) [15], el cual se muestra en la Figura 3.6 Java Card Runtime Environment, se compone de una serie de entidades las cuales se ejecutan en el interior de la tarjeta. Realiza la ejecución del código binario de las aplicaciones, gestionando los recursos, librerías adicionales, comunicaciones, etc. Se puede dividir el JCRE en tres capas:

- La primera empezando por la parte inferior, en la cual se encuentra la parte on-card de la JCVM, aquí encontramos el intérprete. Esta capa es la encargada de realizar la comunicación con las capas de más bajo nivel.
- En la capa intermedia encontramos las clases del sistema. Es donde se realizará todo labores de gestión de las aplicaciones.
- En la capa superior nos encontramos todos los APIs de Java Card, las cuales nos aportan librerías y servicios que van a ser utilizados por las aplicaciones instaladas en las tarjetas. En esta capa también vamos a encontrar otro elemento importante como es el instalador de Java Card.

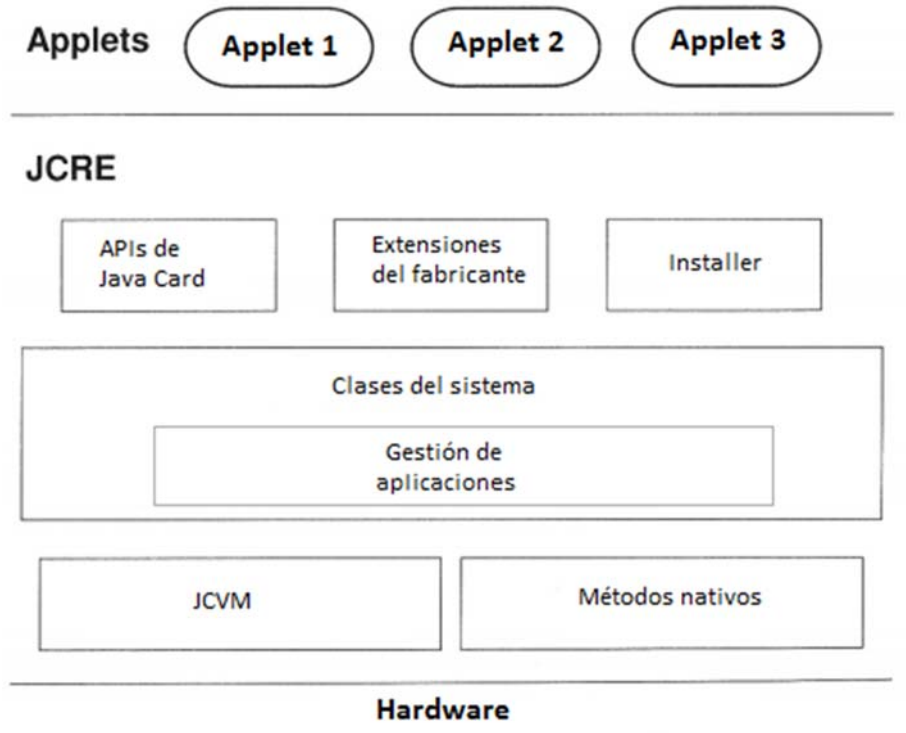

Figura 3.6 Java Card Runtime Environment

El JCRE funciona de forma continua en la tarjeta. Comienza su ciclo de vida cuando durante la fabricación es grabado en la ROM. Cuando la tarjeta pierde la alimentación de energía, la máquina virtual se queda en suspensión sin perder información importante como el estado del JCRE. Estos datos son almacenados en memoria no volátil para conservarlos. Cuando la tarjeta vuelve a recibir energía el JCRE se ejecuta desde el punto en que se encontraba anteriormente.

JCRE deriva de Java Runtime Environment (JRE) pero a este le añaden algunas funcionalidades nuevas:

- En JRE todos los objetos creados se almacenaban en memoria no volátil, mientras que en JCRE pueden ser creados en memoria volátil. Estos objetos reciben el nombre de objetos transitorios, y gracias a ellos se obtienen una mejora de rendimiento en cuanto a la velocidad y uso de la memoria, la cual en estos dispositivos en un recurso muy escaso.
- Para evitar problemas en la ejecución de las operaciones, éstas se realizan de una manera atómica, es decir, o se realiza todo o no se realiza nada. Restaurándose el estado inicial en caso de posibles errores de ejecución o circunstancias inesperadas (p.ej. corte de alimentación).
- En JCRE se introducen los firewall que evitan aplicaciones usen el espacio de memoria de otras y gestionan la compartición de datos entre aplicaciones de una manera segura.

3.6.3 Lenguaje Java Card

Las tarjetas inteligentes son dispositivos con unos recursos limitados. El lenguaje Java Card está adaptado a ello y es por eso que solo hace uso de un conjunto limitado de instrucciones y tipos de datos de todo el repositorio de Java. Este conjunto de instrucciones y tipos de datos se muestran en la siguiente Tabla 3.2 Comparando elementos.:

Elementos Soportados	Elementos No Soportados
• Tipos de datos pequeños: boolean, short, byte, int. • Arrays unidimensionales. • Paquetes, clases, interfaces y excepciones.	• Tipos de datos grandes: long, double, float. • Arrays multidimensionales. • Carga dinámica de clases. • Recolector de basura. • Multiproceso. • Serialización y clonación de objetos. • Gestor de seguridad.

Tabla 3.2 Comparando elementos.

En Java Card se eliminan todas aquellas opciones que se dedican a la salida gráfica, como las interfaces. Java Card elimina también el recolector de basura, por lo que la gestión de objetos dinámicos reside en el propio desarrollador. A pesar de que es posible emplear jerarquía de clases es recomendable que las aplicaciones no tengan muchos niveles de profundidad para simplificar las aplicaciones.

La API de Java Card contiene las diferentes clases para facilitar el desarrollo de aplicaciones:

- *javacard.lang*: paquete en el que se encuentran la mayor parte de las funcionalidades y excepciones derivadas de Java.

- *javacard.framework*: conjunto de clases e interfaces para la gestión de applets y su comunicación con el terminal según ISO 7816.
- *jacarcard.security* y *javacardx.crypto*: paquetes para dar soporte a las funcionalidades criptográficas.

3.7 GLOBAL PLATFORM

Global Platform (GP) [11] nace con la finalidad de conseguir una interoperabilidad entre tarjetas inteligentes multi-aplicación de diferentes fabricantes, pretendiendo además que se alcanzara de una manera eficiente y segura.

Gracias a GP se pueden gestionar las tarjetas inteligentes de una manera cómoda, instalando, eliminando y actualizando las aplicaciones. El control de las aplicaciones de las tarjetas antes quedaba única y exclusivamente reservado al fabricante de la tarjeta a la hora de grabarla. Sin embargo GP permite a los usuarios autorizados gestionar las aplicaciones alojadas en una tarjeta.

3.7.1 INTERACCIÓN CON GLOBAL PLATFORM

A la hora de interactuar con GP tenemos que diferenciar entre dos tipos de interacciones. Por un lado operaciones exportadas como API que se pueden usar en los aplicativos Java Card y permiten acceder a servicios GP relacionados con la autenticación del usuario, funciones criptográficas, etc. Por otro, GP expone una serie de comandos APDU gracias a los cuales se puede hacer una gestión del contenido de la tarjeta y asegurar el correcto acceso a las aplicaciones:

- **DELETE:** permite eliminar aplicaciones y paquetes almacenados en la tarjeta.
- **GET_DATA:** permite devolver un objeto asociado a una aplicación.
- **GET_STATUS:** retorna información acerca del estado del ciclo de vida de los elementos de la tarjeta.
- **INSTALL:** utilizado para instalar o instanciar una nueva aplicación en la tarjeta.
- **LOAD:** carga del binario de una aplicación en la tarjeta.
- **MANAGE CHANNEL:** permite la creación y uso de múltiples canales lógicos para comunicarse con las aplicaciones
- **PUT_KEY:** utilizado para añadir o modificar una clave a la tarjeta.
- **SELECT:** utilizado para seleccionar la aplicación residente en la tarjeta.
- **SET_STATUS:** cambia el ciclo de vida de la tarjeta o la aplicación seleccionada.
- **STORE_DATA:** almacenamiento de datos en una aplicación.

3.7.2 SEGURIDAD EN GLOBALPLATFORM

GP tiene como principal misión garantizar la integridad de los ejecutables de la tarjeta cuando se encuentran corriendo. Para garantizar lo anteriormente descrito GP incluye mecanismo de integridad, confidencialidad y autenticación.

GP también incorpora soporte para diversos algoritmos y procedimientos criptográficos. Las comunicaciones de la tarjeta inteligente con el dispositivo exterior se realizan de forma segura. Para ello se establece un canal seguro entre la tarjeta y el dispositivo externo empleando un procedimiento de autenticación mutua. El canal de comunicaciones seguro soporta confidencialidad e integridad en la transferencia de comandos, características que se negocian en función de las capacidades y versión de GP que disponga la tarjeta y el dispositivo externo.

Al finalizar la comunicación a través del canal seguro (p.ej. desconexión de la tarjeta del lector), las claves y parámetros de seguridad almacenados en la tarjeta son eliminados y deberán ser renegociados en la siguiente sesión que se establezca.

4 DISEÑO

A continuación se recoge la fase de diseño del sistema. En la primera parte se explica una visión global del funcionamiento del sistema, presentando un caso de uso que describa el funcionamiento. La segunda parte se muestra mediante diagramas las interacciones entre los diferentes componentes del sistema a implementar.

4.1 ARQUITECTURA.

El sistema está formado por un servidor, un teléfono móvil y una tarjeta inteligente. Todos estos elementos se encuentran comunicados entre sí como aparece en la Figura 4.1 Arquitectura de los diversos elementos interactuantes en el trabajo.y a continuación se detalla.

El servidor proporciona una página web en la que el usuario puede cargar un archivo el cual se encuentra alojado en el dispositivo desde el que se accede a la interfaz. Una vez cargado el archivo se indica el teléfono al cual se le enviará la solicitud de firma del archivo.

Mediante una notificación se avisará al usuario de las solicitudes pendientes de firma de archivos. Pulsando en la notificación se accede a la aplicación, en la que al acercar la tarjeta inteligente se procederá a la firma del fichero.

En la tarjeta inteligente se encuentran las claves pública y privada de la persona firmante con las que se va a realizar la firma del documento. Esta realiza la firma del hash del documento y se lo devuelve al teléfono, quien lo envía al servidor almacenándolo en una base de datos junto con el fichero original.

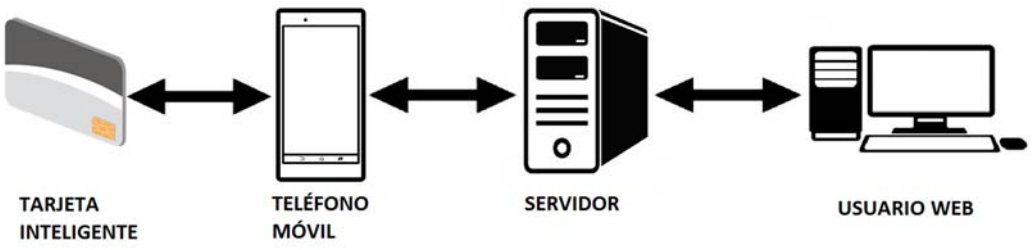

Figura 4.1 Arquitectura de los diversos elementos interactuantes en el trabajo.

4.2 DIAGRAMAS DE CLASES Y FLUJO.

Este apartado presenta la estructura de las aplicaciones realizadas. De un parte se presentan los diagramas de clases con las interacciones entre los diferentes componentes funcionales, para posteriormente mediante diagramas de flujo detallar más profundamente el comportamiento de la solución.

4.2.1 DIAGRAMAS DE CLASES

En primer lugar se muestra el diagrama de clases que implementan la lógica de la aplicación residente en la tarjeta inteligente. El programa que se desarrolla consiste únicamente de una clase por lo que el diagrama muestra únicamente una única caja con los diferentes atributos y métodos de los que se compone, tal y como aparece en la Figura 4.2 Clase del programa que corre en la tarjeta inteligente.. En esta clase de la aplicación Java Card cabe destacar como elementos principales los siguientes atributos y métodos.

Por un lado en cuanto a los atributos es obligatorio crear uno por cada clave generada, ya que estas van a ser creadas una única vez por un método y utilizadas a lo largo de la vida de la aplicación por el método firmar. Por otro lado también tenemos que crear un atributo en el que almacenar el código PIN de la tarjeta inteligente al objeto de verificar que este PIN almacenado en el atributo será el mismo que el introducido antes del uso de la tarjeta.

En cuanto a los métodos de la tarjeta inteligente hay que destacar los siguientes por ser los que den el uso específico a esta aplicación. Estos métodos son los a continuación relacionados getExpPriKeys, getModPriKeys, createKeys, sign, verifyPin, updatePin. Los dos primeros métodos son encargados de extraer el modulo y exponente de la clave privada al objeto de comprobarla en el exterior, el segundo método citado es el encargado de crear las claves pública y privada. El siguiente método sign, será el que realiza la firma y los dos últimos se encargaran de hacer las operaciones con el código pin.

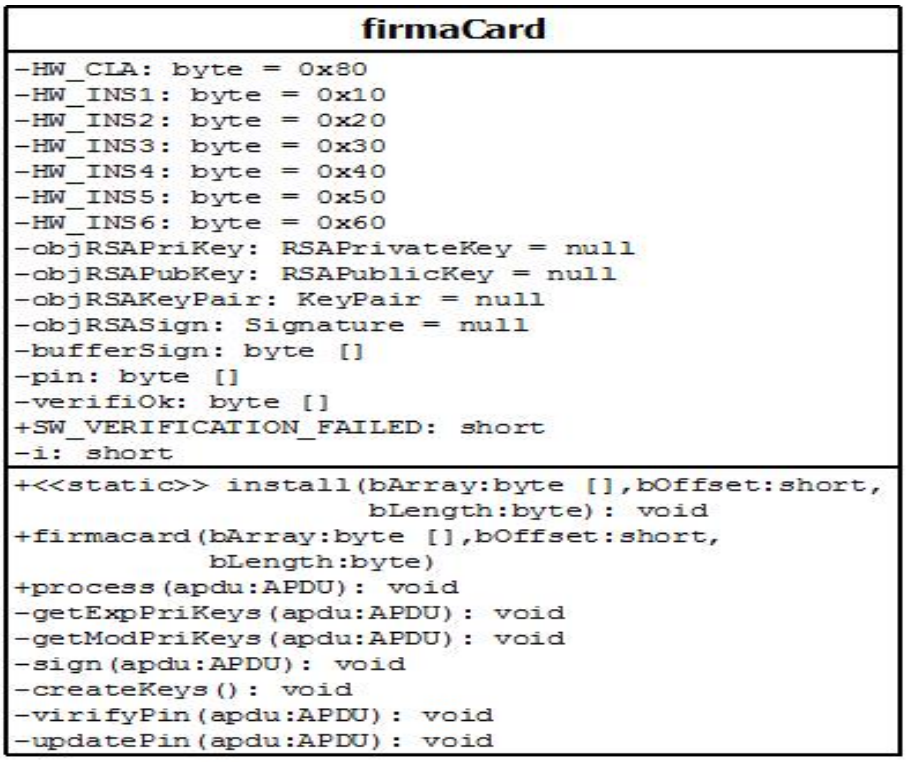

Figura 4.2 Clase del programa que corre en la tarjeta inteligente.

En segundo lugar, está el diagrama de clases de la aplicación Android, se compone de cuatro clases, como se ve en la Figura 4.3: MainActivity en que se realiza el primer registro del dispositivo en el sistema, GcmBroadCastReciever y GcmIntentService encargadas de tratar la notificación del sistema, y Second_NFC donde se gestiona la interacción entre la tarjeta y el dispositivo móvil al objeto de firmar el documento. La aplicación Android interactúa, como anteriormente se ha dicho tanto con la tarjeta inteligente, base principal del trabajo, como con el servidor web.

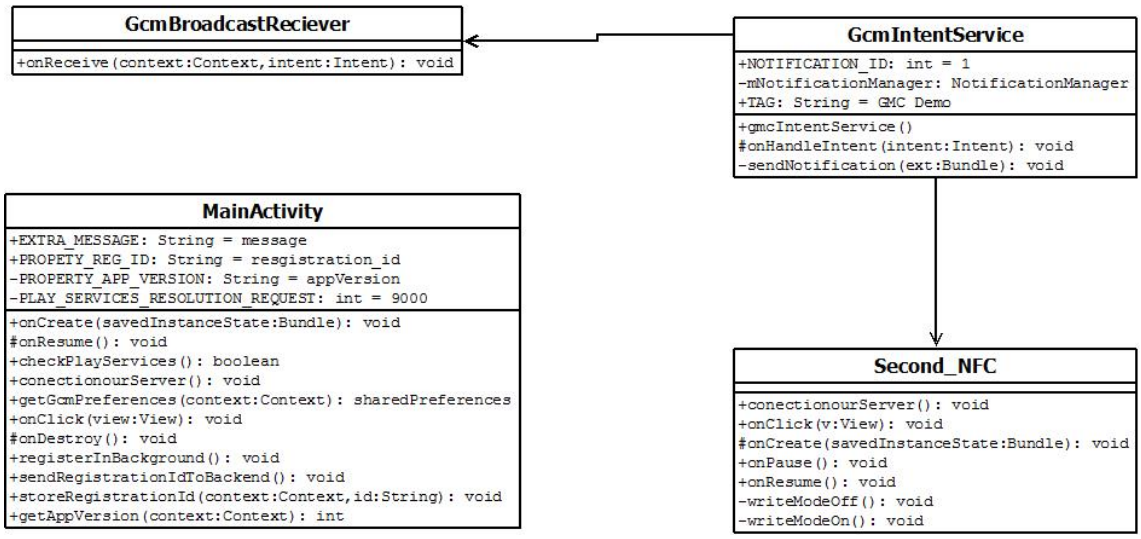

Figura 4.3 Diagrama de clases Aplicación Android.

Y por último, se encuentra el diagrama de clases del servidor web. Se trata de una implementación sencilla en el que ha buscado la funcionalidad frente a la estética. Este dota de los requisitos necesarios para interactuar y hacer completamente funcional

el trabajo. Aquí también se muestra el diagrama de clases de la base de datos creada en el servidor, ambos diagramas se muestran en las Figura 4.4 y Figura 4.5. El diagrama de clases del servidor se compone de tres partes: la principal es server.js, donde se encuentran todas las funciones pertenecientes al servidor de comunicación entre el servidor el navegador web del cliente, la página web de acceso encargada de mostrar una interfaz gráfica y por ultimo tenemos la parte firma.js encargada de ejecutar las operaciones pedidas por el dispositivo móvil de firmado del documento y registro del dispositivo. La base de datos consta de dos tablas: en primer lugar la de usuarios registrados con una relación de 1 a n a la segunda tabla documentos en la que se almacenaran los documentos a firmar correspondientes a un usuario ya registrado.

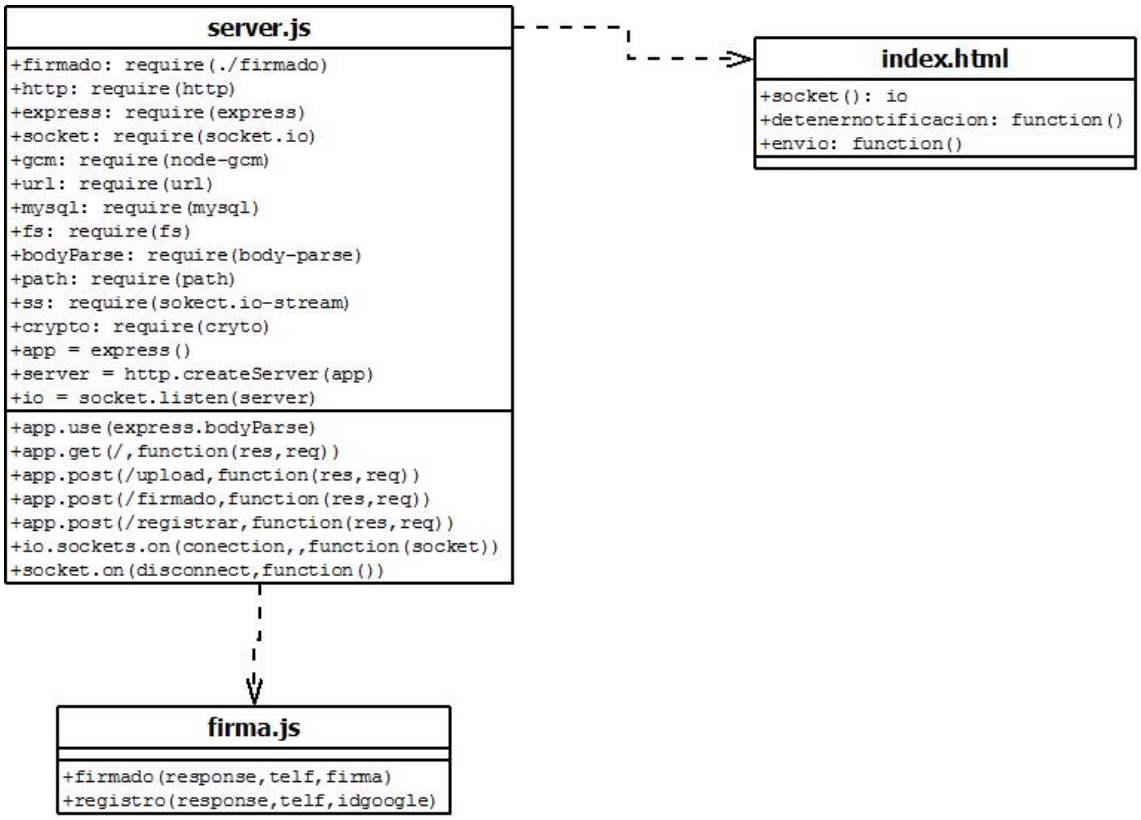

Figura 4.4 Diagrama de clases del servidor.

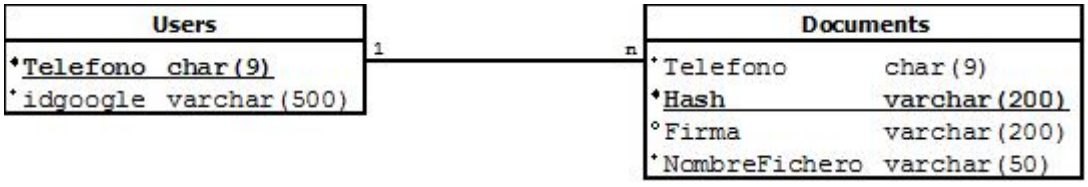

Figura 4.5 Diagrama de clases de la base de datos

4.2.2 Diagramas de flujo.

A continuación se proporciona una visión de la secuencia cronológica en la que se realizarán las operaciones. Se comienza con los intercambios entre el navegador web del usuario del sistema y el servidor web lo cual se aprecia en la Figura 4.6. Esta interacción se hace a través de websockets para conseguir que la interfaz web sea interactiva y fluida.

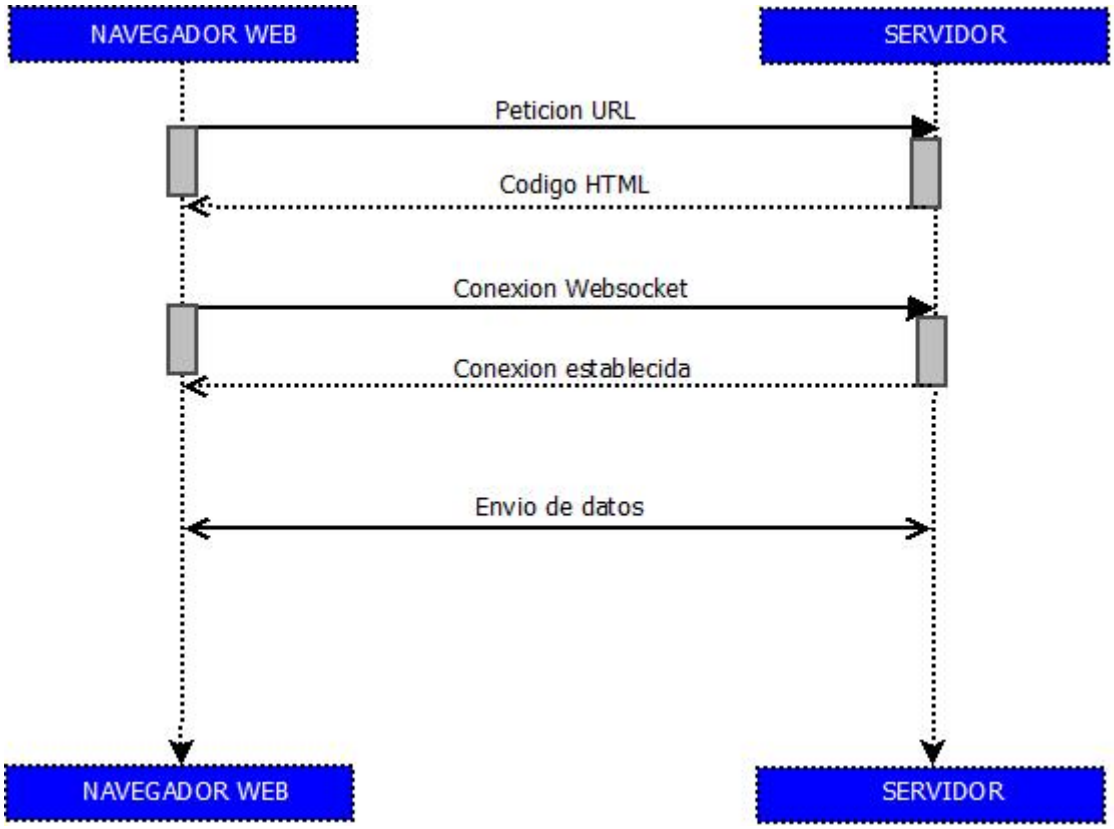

Figura 4.6 Comunicación entre el navegador del usuario y nuestro servidor.

Una vez finalizada la petición del usuario a través de la interfaz web proporcionada por el servidor, la siguiente interacción se produce entre este y el teléfono móvil destinatario, tal como se muestra en la Figura 4.7. El teléfono móvil debe estar previamente registrado en el sistema y con la aplicación desarrollada instalada. En este caso los intercambios entre los diferentes elementos se realizan mediante peticiones HTTP de tipo GET/POST.

Y por último y antes de comenzar a retornar el resultado del sistema, se produce la interacción entre el teléfono móvil, y la tarjeta inteligente, lo cual se aprecia en la última Figura 4.8. La tarjeta inteligente poseedora de las claves pública y privada, se encarga de firmar el documento. Para hacer la firma previamente se tiene que realizar una conexión contra la tarjeta seleccionando la aplicación a ejecutar y la posterior autenticación mediante el PIN correspondiente. Finalizado lo anterior se lanza la operación de firma del hash.

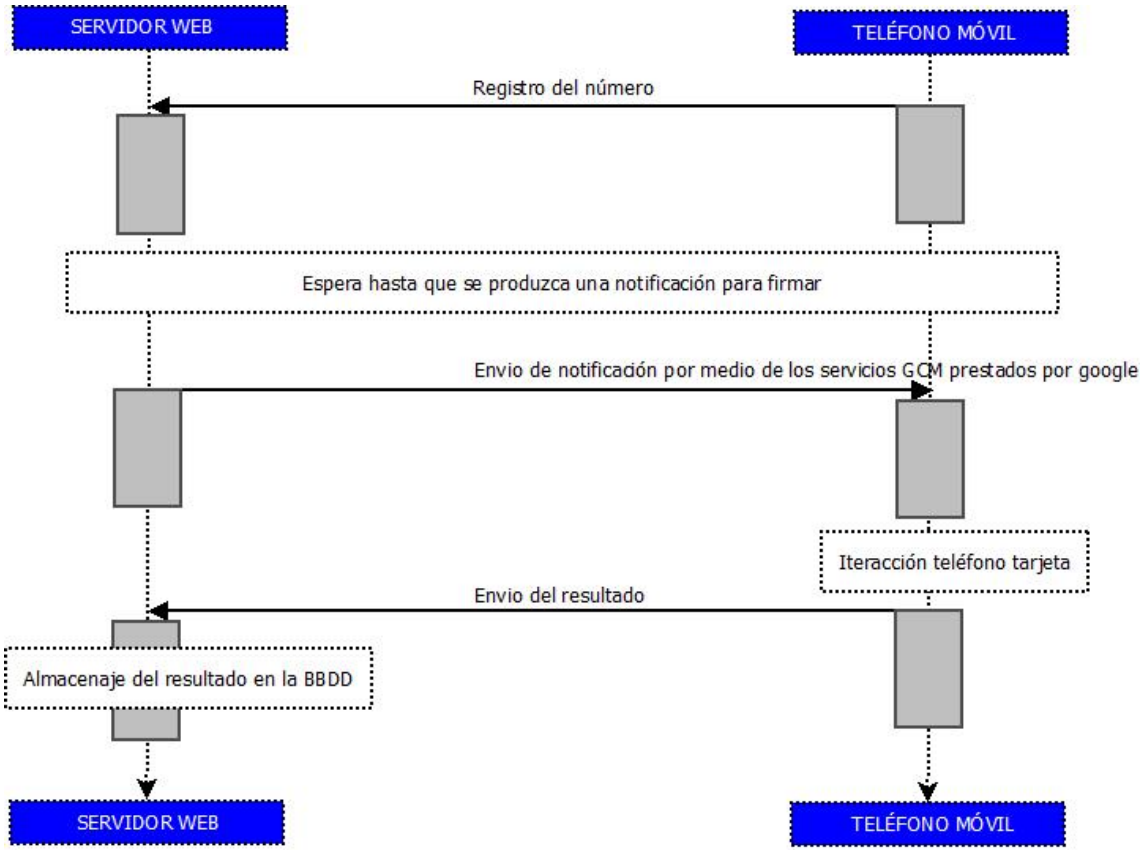

Figura 4.7 Diagrama de flujo entre el servidor web y la aplicación Android.

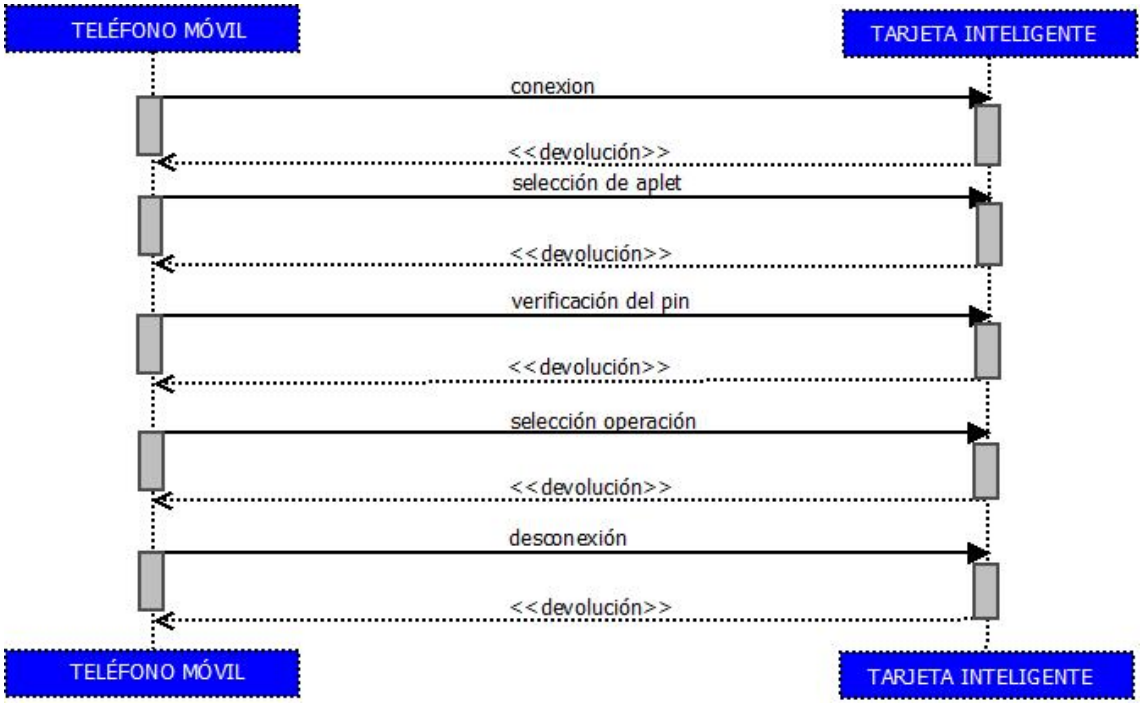

Figura 4.8 Diagrama de flujo entre la aplicación Android y la tarjeta inteligente.

DESARROLLO 5

A continuación se realiza la descripción del desarrollo del sistema planteado. Se divide en cuatro partes. Una primera en la que se introducen las tecnologías utilizadas en el sistema, para posteriormente en una segunda profundizar en la implementación de la solución sobre la tarjeta inteligente. La tercera parte describe la aplicación a ejecutar en el teléfono móvil, implementada para operar en el sistema operativo Android y por último el desarrollo del servidor usando en la tecnología NodeJs. En cada apartado se explica las diversas interacciones entre los diferentes componentes, tratando de ahondar en los puntos más relevantes del desarrollo.

Debido a la interacción existente entre tres entornos tan diversos como son la tarjeta inteligente, el teléfono móvil y el servidor, se han realizado numerosas pruebas de integración para lograr la operativa deseada, algunas de las cuales también se mostrarán.

Durante de la explicación del desarrollo del sistema se ira apoyando esta con diversas ilustraciones, las cuales hacen referencia a los hitos importantes en el desarrollo del sistema. Estas ilustraciones constan de trozos de código con alguna función importante y de capturas de pantalla referentes a la interfaz la cual el usuario final puede apreciar.

5.1 INSTALACIÓN DE HERRAMIENTAS.

En este primer punto se va a tratar de explicar los primeros pasos que se han de realizar en cuanto a la instalación y configuración de las diferentes herramientas antes de empezarse a desarrollar el código de las diferentes partes del sistema.

Para comenzar vamos a explicar con qué se desarrolló la aplicación Android. La aplicación se desarrolla con el entorno de programación Eclipse, por ser un entorno desarrollo muy utilizado y con el que la mayoría de desarrolladores Java trabajan. Al Eclipse se le instalaron un conjunto de plugins para facilitar la interacción con el SDK y aplicaciones del framework de Android. A día de hoy Eclipse está en desuso para el desarrollo Android desde el 2015 y el entorno utilizado es Android Studio.

Como dispositivo móvil se empleará un Sony Xperia Z, un terminal Android que incluye soporte de la tecnología NFC necesaria para la comunicación con tarjetas inteligentes sin contactos.

Configurado el entorno de desarrollo para poder depurar la aplicación directamente en el propio terminal móvil, se debe configurar y registrar el dispositivo en el sistema de notificaciones proporcionado por Google. Para ello se da de alta la aplicación en el sistema Google Cloud Messaging (GCM) de Google, gracias a lo cual se habilita la trasferencia de archivos y envío de notificaciones a la aplicación móvil. En el registro de la aplicación en [12], se proporciona un identificador único para el proyecto que permite el enlace con los servidores GCM para el envío de notificaciones a los usuarios que previamente se haya registrado en el sistema en desarrollo. Esta interacción se muestra en la Figura 5.1.

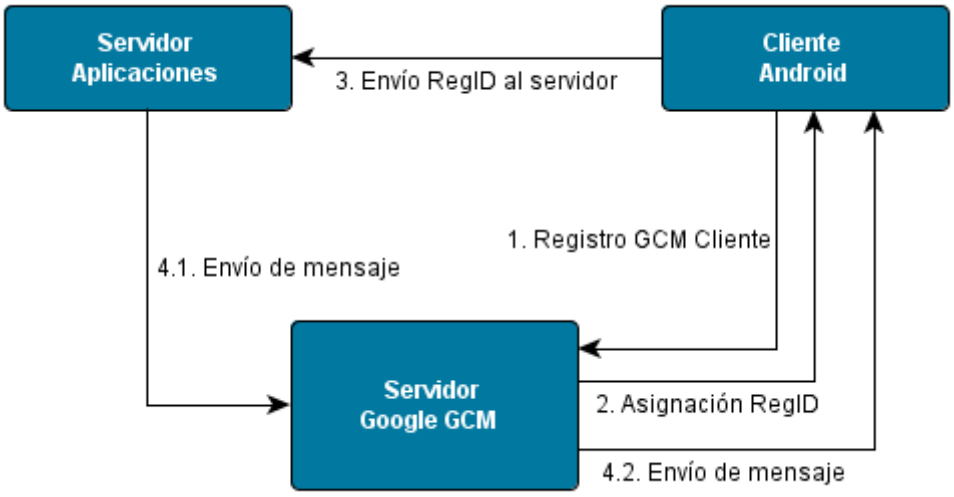

Figura 5.1 Interacción entre los servidores de Google y nuestras herramientas.

En segundo lugar trataremos la configuración del servidor utilizado. La plataforma hardware que soporta el servidor es en una Raspberry Pi Model B, la cual corre un sistema operativo Linux Raspberry 3.12.22 armv61 sobre el cual se han instalado el soporte para PHP y MySQL.

Se ha configurado el servidor y el router de acceso de tal manera que el tráfico HTTP en el puerto 80 esté accesible desde el exterior. El último paso en cuanto a la instalación y configuración de nuestro servidor será instalar NodeJs este es el motor principal del servidor, el cual va a permitir implementar la lógica de servidor en código JavaScript. Adicionalmente el habilitar una conexión a través de websockets permite mantener el enlace abierto entre el servidor y cualquier cliente, permitiendo con ayuda de código Javascript actualizar y mejorar la interactividad de la interfaz de acceso proporcionada al usuario.

Para el trabajo con la tarjeta inteligente se requiere también de la instalación de herramientas para la compilación e instalación de las aplicaciones o applets. Se ha empleado una tarjeta multiaplicación con interfaz dual y con la versión Java Card 2.2.1. Además de las herramientas incluidas en los paquetes pcsd y pcsd-tools, empleadas para habilitar la comunicación entre el lector y la tarjeta, se ha empleado la aplicación gpshell que incluye el soporte para la instalación, actualización, etc. de aplicaciones en la tarjeta.

5.2 TARJETA INTELIGENTE.

Como ya se explicó en el capítulo anterior, el applet Java Card se compone únicamente de la clase firmaCard. Seguidamente se describe la implementación realizada.

Por defecto en todo applet Java Card, el método process recoge todas las APDU remitidas por el lector, que en este caso será el teléfono móvil. Desde aquí gestiona las llamadas a las funcionalidades asociadas a las APDU. Tras comprobar el correcto formato de la APDU, ésta se analiza tomando el código INS como puntero que redirige a la función específica que implementa una determinada funcionalidad. En este trabajo con la tarjeta se han definido seis operaciones, cuya relación se incluye en la Tabla 5.1 Partes de una trama APDU.

Descripción	Código de la APDU	Función
Extraer Exponente de la clave privada	0x10	getExpPriKeys(apdu)
Extraer módulo de la claves privada	0x20	getModPriKeys(apdu)
Generar claves	0x30	createKeys()
Firmar	0x40	sign(apdu)
Verificar el PIN	0x50	verifyPin(apdu)
Actualizar PIN	0x60	updatePin(apdu)
No soportado	Otro	Excepción instrucción no soportada

Tabla 5.1 Partes de una trama APDU

Los métodos getExpPriKeys y getModPriKeys permiten extraer el exponente y el módulo de la clave privada y encapsularlo en una APDU respuesta. Se trata de una funcionalidad que únicamente tiene sentido en un entorno de depuración o de demostración. Se emplea para poder exportar los parámetros de las claves de usuario y poder validar el correcto funcionamiento de la implementación de los procedimientos criptográficos en la tarjeta mediante herramientas software/hardware disponibles en la comunidad (i.e. OpenSSL). Estas dos funciones serían eliminadas de la aplicación Java Card una vez comprobado que la firma se realiza de una forma correcta, al objeto de eliminar posible brecha de seguridad en el sistema.

El método createKeys genera la pareja de claves del usuario que se empleará para firmar los documentos. Este método permite seleccionar el tipo de algoritmo y

longitud de la clave. Para todas las operaciones se apoya en las funcionalidades incluidas en JavaCard 2.2.1. En la implementación de este método se podría haber optado por importar las claves, sin embargo dota de mayor robustez a la solución la generación interna y evitar que la clave privada esté disponible fuera del contenedor seguro que es la tarjeta.

En el método sign se realizan todas las operaciones que llevan a la firma del documento que se le ha transfiera a la tarjeta. Como ya se ha comentado, a la tarjeta se le remite únicamente el resumen del fichero original y no el fichero completo. Primeramente se extraen todos los datos de la APDU y se comprueba su correcto formato. Una vez comprobado que la APDU es correcta y validado el PIN de acceso a la tarjeta inteligente, se desbloquea el acceso a la clave privada y se realiza la firma. Para esto se va a apoyar en las operaciones que proporciona el paquete de javacard.security. En la Figura 5.2 se muestra un extracto del código fuente de este método.

```
private void sign( APDU apdu){

    byte[] buffer = apdu.getBuffer();

    if(buffer[ISO7816.OFFSET_P1] != 0x00 || buffer[ISO7816.OFFSET_P2]!= 0x00)
        ISOException.throwIt(ISO7816.SW_INCORRECT_P1P2);

    byte data =(byte) apdu.setIncomingAndReceive();

    if (buffer[ISO7816.OFFSET_LC] != data)
        ISOException.throwIt(ISO7816.SW_WRONG_DATA);

    if(verifyOk[0]!=(byte)1)
        ISOException.throwIt(SW_VERIFICATION_FAILED);

    objRSASign = Signature.getInstance(Signature.ALG_RSA_SHA_PKCS1, false);
    objRSASign.init(objRSAPriKey, Signature.MODE_SIGN);

    short length = objRSASign.sign(buffer, ISO7816.OFFSET_CDATA, ISO7816.OFFSET_LC, buffer,(short) 0);
    apdu.setOutgoingAndSend((short)0, length);

}
```

Figura 5.2 Código fuente del método sign

Los métodos verifyPin y updatePin permiten gestionar el valor del PIN tanto para cambiarlo como para verificarlo. En el primer caso se exige que el usuario antes haya acreditado ser el titular de dicha tarjeta introduciendo correctamente el PIN antiguo.

Generado el código, se compila y se genera el archivo CAP o JAR que instalar en la tarjeta. En la Figura 5.3 se muestra el script con el que tras cargar e instanciar el applet desarrollado, se evalúa su funcionamiento satisfactorio. Se incluyen las APDU correspondientes a la generación de las claves y extracción del módulo y exponente de la clave privada.

```
mode_211
enable_trace
establish_context
card_connect -readerNumber 1

// Select Applet
select -AID 01020304050601

//generateKeys
//send_apdu -sc 0 -APDU 80300000
//getKeysExpPri
//send_apdu -sc 0 -APDU 8010000099
//getKeysModPri
//send_apdu -sc 0 -APDU 8020000099
//verif Pin
//send_apdu -sc 0 -APDU 805000000400000000
//sign
//send_apdu -sc 0 -APDU 804000000811111111111111111
//update Pin
//send_apdu -sc 0 -APDU 806000000411111111
//verify Pin
//send_apdu -sc 0 -APDU 805000000411111111

card_disconnect
release_context
```

Figura 5.3 Código script para el envío de APDUs.

```
javier@javier-K55VD ~/smartcard/GPShell-Scripts $ gpshell test.gpshell
mode_211
enable_trace
establish_context
card_connect -readerNumber 1
* reader name SCM Microsystems Inc. SCR 355 [CCID Interface] 00 00
select -AID 01020304050601
Command --> 00A4040007010203040506 01
Wrapped command --> 00A4040007010203040506 01
Response <-- 9000
send_apdu -sc 0 -APDU 80300000
Command --> 80300000
Wrapped command --> 80300000
Response <-- 9000
send_APDU() returns 0x80209000 (9000: Success. No error.)
send_apdu -sc 0 -APDU 8010000099
Command --> 8010000099
Wrapped command --> 8010000099
Response <-- 41816DEE2BEF91BFE5B18E08ACD90E624E1058579A798B84325EE90E268CE1686DF88D
DF98B1109CBB4F8032C9C44C9688DD912C4136E5310D7A100C663094F7BA5148D86DBC57E7F99087ABD
E728EC7E011FD9CEA87184CA9C6DC6DD1F2AD6034B0EFB84BDFA06EDC90D7D58A45D4A7D696BE9A0540
E1C9983A94612424CE546CF5232190E0434C4F56969608E6911B3900B210AC3A7F8EE5615FCBC3B41F0
7D708CE4CADFC018ED832B3807932425B74D2435667857218E337B25C513ADF62F73164054F4BA75ABE
1FE8E7BCFC50F0660AF30E5BF0DBC688A5FF1A1BA36B02504F6588E7BC8F212B0617C029C4554DC00F8
C580BBCBCF6C25A01CC5F5619CD9000
send_APDU() returns 0x80209000 (9000: Success. No error.)
send_apdu -sc 0 -APDU 8020000099
Command --> 8020000099
Wrapped command --> 8020000099
Response <-- D72857F55968F59D1288546651A5CB4FAB0C0D7F24FA7857EFB84C39867C6D1FD96B66
31AF831A359A771549F722AF381FF95ABED0A20CCDC74E174E33EC627CFB6EC07FF9B28D7825C747EB0
97CA921F258D4F76B1E7B72D7FA0617195F207C04607CEE1F548AB340017E515832FFB9F00BF914DF6F
924EEE697F4F5C775A27A09F4AADA1CAD3F2E26E8F3DA8B4C74BF20B68E942F8D53BE107FD46965A4D4
C5623AD7E4F8C80941E559984013D233606EEDAFFB6215CEDE7E603F630A13D693739C9D50D70E15591
032B91DECCFF2EF10D95BAE16D1096CC2B34118B06EFF5C3FD8524D4F8BF95D3BD53846281DAE640DB1
8D4D7ABD1DB734F3F8E24AFE0759000
send_APDU() returns 0x80209000 (9000: Success. No error.)
card_disconnect
release_context
javier@javier-K55VD ~/smartcard/GPShell-Scripts $
```

Figura 5.4 Resultado de la primera prueba.

En la Figura 5.5 se muestra la verificación del proceso para la generación de la firma, previa verificación del PIN.

```
javier@javier-K55VD ~/smartcard/GPShell-Scripts $ gpshell test.gpshell
mode_211
enable_trace
establish_context
card_connect -readerNumber 1
* reader name SCM Microsystems Inc. SCR 355 [CCID Interface] 00 00
select -AID 01020304050601
Command --> 00A4040007010203040506 01
Wrapped command --> 00A4040007010203040506 01
Response <-- 9000
send_apdu -sc 0 -APDU 805000000400000000
Command --> 805000000400000000
Wrapped command --> 805000000400000000
Response <-- 9000
send_APDU() returns 0x80209000 (9000: Success. No error.)
send_apdu -sc 0 -APDU 80400000081D7C0D11704F08FD10
Command --> 80400000081D7C0D11704F08FD10
Wrapped command --> 80400000081D7C0D11704F08FD10
Response <-- 8E33A1D269E1ACFCFA5C0CC75B7214002D644ABDBAEE7B904CCF7E0648BEDB7AA48FC1
ADD4A013A00B2BC6311EA8C33536FC6152FB9D76E8A22FAF7C2FDCBF578B08D372008BE0D7FDD2E301B
ABF7D6DEAB520263F0C58D1BD19D2AF82A6E59A90123FF3AD49AA9993E702F1A9E9EF90374E74D6A38A
628B65C9C2ACBAF23C590D6130BCFD46BE2B2B59ACD6F93A5C800859F0A4E901BE2B3F25CC92165827A
75F4897DDFA353079D4D0E820D0B15FD2191BB138A6FDA4954048C103E767196C5A9611EA92F9229AFA
B2BC2DB97FFBF5D987825527F6E595FD46186E0DFF1EFE212ECD18E7CFB1FB660A0D96ED86726D8279B
07F1173C66B2F1840864DAD8B619000
send_APDU() returns 0x80209000 (9000: Success. No error.)
card_disconnect
release_context
javier@javier-K55VD ~/smartcard/GPShell-Scripts $
```

Figura 5.5 Resultado de la segunda prueba.

5.3 Aplicación Android.

La aplicación móvil se compone de cuatro clases cuya implementación se expone a continuación.

La clase principal MainActivity, proporciona la interfaz inicial de la aplicación, la cual se muestra en la Figura 5.6. Desde ésta el usuario se registra por un lado en el sistema para interactuar con el servidor documental y por otro en los servidores de GCM de google, para que se pueda realizar el envío de la notificación al teléfono empleado. La finalización satisfactoria o con errores de todo el proceso se gestiona y muestra al usuario.

Figura 5.6 Captura de la pantalla principal de nuestra aplicación.

Para la validación del correcto funcionamiento de este proceso se analiza el contenido en la base de datos, así como el potencial registro en GCM de Google. En la Figura 5.7 se muestra un ejemplo del resultado del registro del teléfono en el proyecto creado en los servicios GCM.

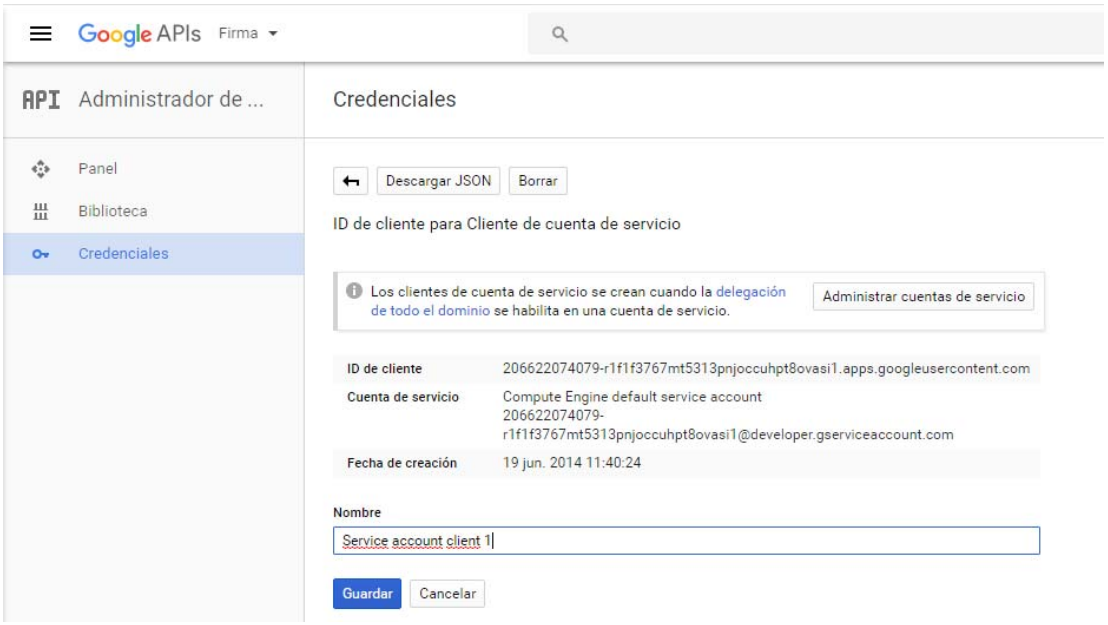

Figura 5.7 Cliente servicios GCM google.

Parte del proceso anterior también puede realizarse de forma manual mediante el uso del botón Registrarse que se ha incluido en la aplicación. Al pulsar el botón se realiza la llamada al método conectionourservidor(), que gestiona la comunicación con

el servidor de base de datos. Estas operaciones se realizan empleando un cliente HTTP que remite un objeto JSON y el identificador GCM. El objeto JSON es enviado por el cliente mediante una llamada POST. En cuanto a la clase MainActivity estos serían los métodos más relevantes.

```
HttpClient httpClient = new DefaultHttpClient();

HttpPost post = new HttpPost(
        "http://192.168.1.20:8888/registrar");

post.setHeader("content-type", "application/json");

try {
    // Construimos el objeto cliente en formato JSON
    JSONObject dato = new JSONObject();

    dato.put("telefono", etNumbTlf.getText().toString());
    dato.put("idgoogle", getRegistrationId(context));

    StringEntity entity = new StringEntity(dato.toString());
    post.setEntity(entity);

    HttpResponse resp = httpClient.execute(post);
```

Figura 5.8 Código encargado de realizar el registro contra nuestra base de datos-

A continuación se describen las clases GcmBroadcastReciever y GcmIntentService, encargadas de tratar las notificaciones enviadas por los servicios de Google GCM a la aplicación. La notificación enviada por los servicios GCM de Google es recogida por la clase GcmIntentService, esta clase tiene dos métodos sendNotification y OnhandlentIntent que a continuación se describirá su funcionalidad. El primer método es el encargado de dar el aspecto que el desarrollador desee a la notificación lanzándola posteriormente, haciendo que esta notificación nos envíe a la segunda activity de la aplicación, la cual conecta con la tarjeta inteligente. Por el otro lado el método OnhandlentIntent es el encargado de extraer la información que nos aporta la notificación, enviada por los servicios GCM y llamar a la otra clase citada anteriormente, GcmBroadcastReciever. En cuanto a esta clase es la encargada de hacer que la aplicación la cual no se encontraba ejecutando despierte y se ponga a trabajar.

Finalizada la implementación se comprueba de forma autónoma el correcto funcionamiento. La principal prueba de este código implementado en estas dos clases será comprobar que la notificación creada despierta la aplicación y salta a la segunda activity, encargada de conectar con la tarjeta que a continuación se explica. Para realizar esta prueba se ha generado un pequeño código JavaScript simulando el lanzamiento de un mensaje para la aplicación y que así genere una notificación. En la siguiente Figura 5.9 se muestra el código.

```
var gcm = require('node-gcm');

var sender = new gcm.Sender('AIzaSyDdorT4UV1YMXpTFb-syybhEPLsMxEC_9I');//API key

var registrationIds = [];

var message = new gcm.Message();
message.addData('mesagge',"NOTIFICACION RECIBIDA!!!!");
message.addData('title','Push Notification Sample' );
message.addData('msgcnt','3'); //Notificacion mostrada en la barra de tareas
message.addData('soundname','beep.wav'); //Sonido de la notificacion
message.timeToLive = 3000;// Duración en segundos que permanecerá en GCM e intentarlo de nuevo antes de finalizar.

registrationIds.push('APA91bG4mIRBfll7Bpl3SAAD3n5Jjjia1UDKca_07txCGfvvwF9UtYW-5EinW6xWFnbgV4SC_4-hgarW3UYw7XWysfxt
y0MYS6IyTAx8s-FdZjf1G_pc0nljOgK4ZGKokZoiPOgY9aMDtFm7jRNaMtEQXNWhOp-jkHrzi14IY5e3ImcIwm4nRwU'); // Id del dispositiv

sender.send(message, registrationIds, 4, function (result) {
    console.log(result);
});
```

Figura 5.9 Código encargado de generar una notificación para probar que se programó correctamente.

En la clase second_NFC se realiza la firma del hash, el cual ha sido recibido en la notificación. Para ello se hace uso de la tarjeta inteligente y el resultado, es decir, la firma se remite al servidor para su almacenaje en la base de datos. En esta clase cabe destacar los métodos onClick y conectionourserver. Este último método implementa una funcionalidad muy similar a la realizada en la clase Mainactivity. onClick por su parte comprueba si el dispositivo utilizado posee NFC, en cuyo caso crea un objeto ISODEP que gestionará la comunicación con la tarjeta. Las APDUs tanto comando como respuesta se transfieren en forma de array de bytes. Una vez finalizadas las comunicaciones con la tarjeta, se realiza el envío de la firma al servidor para su almacenaje. En las siguientes Figura 5.10 y Figura 5.11 se puede observar el código encargado de realizar tanto el envío de APDUs como la recepción de las respuestas generadas por la tarjeta.

```
    if(myTag == null){
        Toast.makeText(context, "Este dispositivo no contiene NFC", Toast.LENGTH_LONG).show();
}else{

    IsoDep isoDep = IsoDep.get(myTag);
    try{
        isoDep.connect();
         if(isoDep.isConnected()!=true) {
           System.out.println("Fallo en la conexion");
         }
        byte[] SelectApi={
           (byte) 0x00,//CLA
           (byte) 0xA4,//INS
           (byte) 0x04,//P1
           (byte) 0x00,//P2
           (byte) 0x06,
           (byte)0x01,(byte)0x02,(byte)0x03,(byte)0x04,(byte)0x05,(byte) 0x06
        };
        byte[] result = isoDep.transceive(SelectApi);

        System.out.println("API SELECT");
```

Figura 5.10 Código que selecciona la aplicación de nuestra tarjeta.

```
byte[] hash= text.getBytes();
int lengthhash =  hash.length;
String hex = Integer.toHexString(lengthhash);

int parsedResult = (int) Long.parseLong(hex, 16);

byte[] cabecera={
        (byte) 0x80,
        (byte) 0x20,
        (byte) 0x00,//P1
        (byte) 0x00,//p2
        (byte) parsedResult,
    };

byte[] sign = new byte[cabecera.length + hash.length];
System.arraycopy(cabecera, 0, sign, 0, cabecera.length);
System.arraycopy(hash, 0, sign, cabecera.length, hash.length);
```

Figura 5.11 Código encargado de generar la APDU que contiene el hash

Parte de este código se ha validado una vez se ha integrado todo el sistema. No obstante la comunicación entre móvil y tarjeta sí se testeó previamente. Para ello se generó un código JavaScript encargado de lanzar una notificación para poder disparar la firma. Así se despierta la comunicación NFC, se reconoce la existencia de la tarjeta y se realiza el intercambio de información.

Con esto se ha finalizado el desarrollo y pruebas individuales de la parte del sistema de la aplicación Android. A continuación se va a explicar el desarrollo del servidor.

5.4 Servidor NodeJs.

El servidor proporciona las funcionalidades de comunicación, gestión de la base de datos, la comunicación con el teléfono móvil y la interfaz de usuario. Al acceder a la página principal, mostrada en la Figura 5.12, se muestra la interfaz básica que solicita el documento a firmar y el número de teléfono de quien lo vaya a firmar. Igualmente se realiza la apertura de un websocket a través del cual se actualizará de manera dinámica y transparente para el usuario el contenido de la interfaz a la recepción de los resultados de la firma.

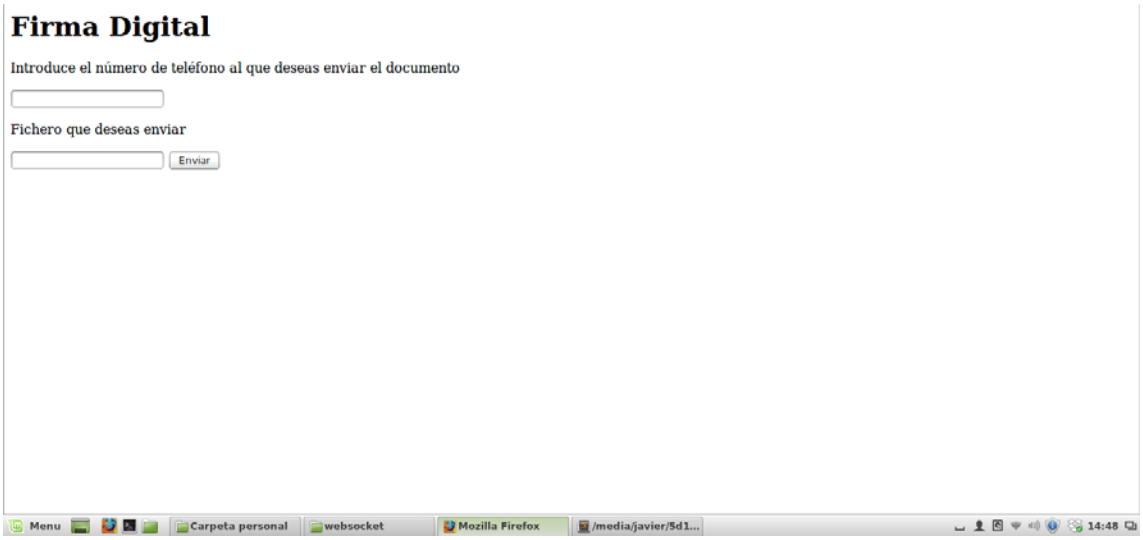

Figura 5.12 Interfaz web de nuestro servidor.

Por otro lado el servidor incluye varios procesos: por un lado se tiene la comunicación vía websocket con el navegador del usuario y por otro lado las comunicaciones HTTP GET/POST con la aplicación móvil. El motivo de no utilizar una conexión websocket con nuestro teléfono es la ausencia de necesidad en esta comunicación de una interacción continua.

En primer lugar el servidor recoge el fichero remitido a través de la interfaz junto con su nombre, el número de teléfono asociado y el resumen del fichero todo ello lo almacena en la base de datos. A continuación cuando ya se tiene todo almacenado en el servidor se lanza la notificación a la aplicación móvil para que el resumen del fichero sea firmado. En las Figura 5.13 y Figura 5.14 se muestra en primer lugar un ejemplo de notificación y un extracto del código encargado de lanzar la notificación a la aplicación.

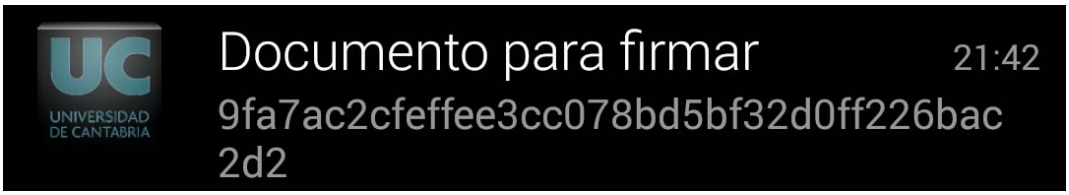

Figura 5.13 Notificación recibida en el teléfono.

```
var envio = new gcm.Sender('AIzaSyDdorT4UV1YMXpTFb-syybhEPLsMxEC_9I');

var registrationIds = [];

var mensaje = new gcm.Message;
mensaje.addData('mesagge',fichero);
mensaje.addData('title','Archivo para firmar');
mensaje.addData('msgcnt','3');
mensaje.addData('soundname','ncp01074.wav');

registrationIds.push(id);
envio.send(mensaje,registrationIds,4,function(result){
    console.log(result);
};
```

Figura 5.14 Código JavaScript del servidor encargado de enviar la notificación al teléfono.

Programadas estas funcionalidades se comprobó el correcto funcionamiento de ellas. Analizando el contenido de la base de datos mediante la instrucción SELECT, se observa cómo se encuentra almacenada la información. En la Figura 5.15 se aprecia el resultado del SELECT anteriormente citado.

```
mysql> select * from documents;
+-----------+------------------------------------------+-------+----------------
-+
| telefono  | hash                                     | firma | nombrefichero
 |
+-----------+------------------------------------------+-------+----------------
-+
| 675187080 | 9fa7ac2cfeffee3cc078bd5bf32d0ff226bac2d2 | NULL  | log6octubre.txt
 |
+-----------+------------------------------------------+-------+----------------
-+
1 row in set (0.00 sec)

mysql>
```

Figura 5.15 Consulta de la tabla documents.

Como se ha descrito anteriormente, el servidor soporta solicitudes HTTP GET y POST. Las primeras retornan el contenido de la página principal, mientras que las llamadas POST, realizadas desde la aplicación móvil, son tratadas para desempaquetar los datos recibidos y almacenar la firma asociándola con el fichero correspondiente. Por otro lado, también se incluye una gestión de otro modelo de datos que incluirá el ID proporcionado por GCM y el número de teléfono asociado. Esta última operación se realiza durante el registro en la plataforma.

Señalar que aunque se ha considerado el uso de conexiones HTTPS entre los diferentes elementos, para no incluir una mayor complejidad y permitir una más fácil evaluación del correcto funcionamiento del sistema, se ha optado por emplear HTTP.

5.5 Integración y evaluación

Seguidamente se describe y evalúa el comportamiento del sistema integrado.

Tras lanzar el servidor se accede al interfaz de usuario de la plataforma. En el primer acceso se realiza el registro del dispositivo móvil. Una vez registrado el dispositivo móvil se tiene que acceder desde el navegador web de un ordenador a la interfaz gráfica proporcionada por el servidor, rellenando en esta el número del usuario destinatario, seleccionando el archivo a firmar y pulsando el botón Enviar.

Tras esto, como ya se ha mostrado anteriormente, los datos se almacenan en la base de datos. Almacenados todos los datos se genera la notificación, que despertará la aplicación Android, la cual porta los datos necesarios para firmar el documento.

La notificación permite el acceso directo a la aplicación. Automáticamente se establece la comunicación con la tarjeta inteligente a través de la tecnología NFC. Acercando la tarjeta y pulsando sobre el botón de firma se genera el flujo de APDUs indicando a la aplicación cargada en la tarjeta inteligente que firme los datos que se le

envían en una APDU. Realizada la firma del hash por la tarjeta inteligente, ésta se retorna al teléfono móvil para su posterior almacenaje en el campo firma de la tabla documents de la base de datos del servidor.

Por último hay que comprobar que la firma se creó y almaceno correctamente en la base de datos. Para esto al igual que ya se realizó en el capítulo anterior, se realiza un SELECT de la tabla documents comprobando que todos los campos de esta tabla han sido rellenados de una forma correcta, como se puede apreciar en la Figura 5.16 a continuación mostrada.

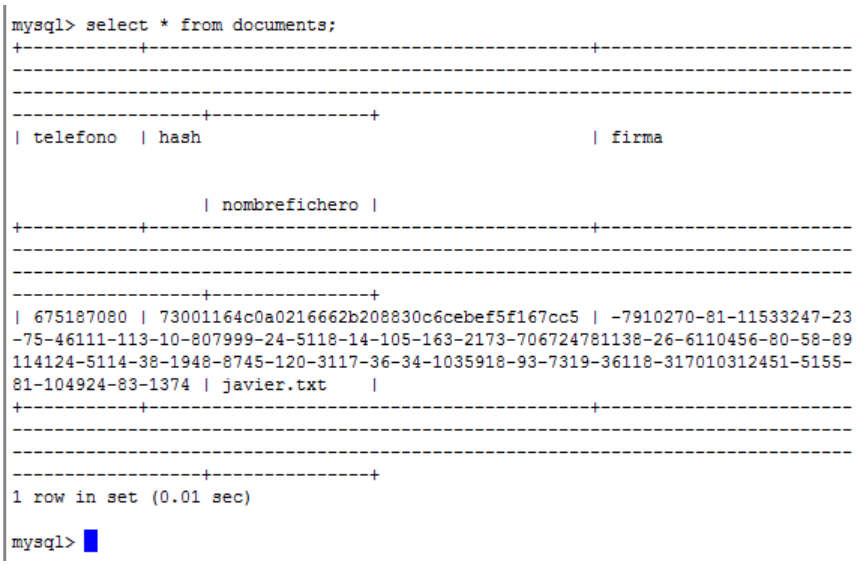

Figura 5.16 Captura de pantalla mostrando la base de datos tras ejecutar todo el sistema.

Estas pruebas se centran únicamente en comprobar el correcto funcionamiento del sistema, en ningún momento se han realizado pruebas comprobando el comportamiento del sistema, si se le pasan parámetros no esperados por el mismo.

CONCLUSIONES Y LÍNEAS FUTURAS

6

Terminado el trabajo de implementación de una plataforma móvil de firma electrónica, cabe destacar que se han conseguido los objetivos marcados cuando empezamos nuestro trabajo. Los cuales eran crear una forma sencilla e intuitiva, con la que acercar a los usuarios menos experimentados en las nuevas tecnologías, la importancia y el gran potencial de futuro que tiene la firma electrónica segura.

Para conseguir esto se comenzó con pequeño estudio del campo de la criptografía, ahondando tanto en la criptografía clásica como la moderna. El estudio se decanta por la criptografía moderna, dividida en simétrica y asimétrica. La simétrica se caracteriza por tener una única clave secreta para cifrar y descifrar, mientras que la asimétrica se basa en el empleo de dos claves, una pública y otra privada, con las que se va a cifrar y descifrar. En el estudio de criptografía también se hizo hincapié en la firma electrónica española la cual viene recogida en la Ley 59/2003, esta se basa en una criptografía asimétrica y tiene que ser utilizada en un dispositivo seguro, como puede ser una tarjeta inteligente.

Dado que para que la firma electrónica se considere reconocida debe ser realizada en un dispositivo seguro, en el presente proyecto se optó por usar tarjetas inteligentes. Estas se consideran dispositivos seguros y permiten generar y almacenar claves de forma robusta. Se ha realizado un estudio de la tecnología, centrándonos en las tarjetas inteligentes Java Card, pues son las que se han usado en este proyecto.

Los conocimientos adquiridos permiten plantear y diseñar el sistema. El sistema final se ha compuesto de tres elementos esenciales: una tarjeta inteligente sin contactos, un dispositivo móvil con sistema operativo Android y soporte NFC, y un servidor Linux creado con la tecnología NodeJs. Se ha desarrollado una aplicación móvil para la interacción con el usuario y la tarjeta inteligente, que a su vez transferirá la firma al servidor para su almacenaje. Se ha buscado en todo momento el facilitar el uso por parte del usuario, para lo cual se emplea un sistema de notificaciones móvil que enlazan directamente con la realización de la firma en la tarjeta.

Como objetivo secundario, indicar que el sistema creado se puede emplear también como sistema de autenticación de usuarios sin tener que realizar modificaciones al sistema. La comprobación de la firma realizada no solo dota de integridad sino también de autenticación.

Destacar también que gracias a este trabajo se ha podido conocer las grandes posibilidades de una tecnología usada en nuestro día a día pero técnicamente desconocida.

El presente trabajo presenta numerosas líneas futuras, enumerando a continuación aquellas que se consideran más relevantes:

- En primer lugar hacer del sistema un sistema seguro, tratando de evitar ataques Web como el SQL Injection, Cross-site scripting, etc. Por otro lado en el tema de la seguridad también realizar las llamadas entre el navegador, el teléfono y el servidor de manera cifrada bajo protocolo HTTPS.
- Dotar al sistema de una interfaz con un diseño más atractivo, ya que la actualmente proporcionada se ha centrado únicamente en la funcionalidad.
- Incluir un registro de usuario y restringir el uso de la plataforma mediante usuario y contraseña o usando certificados electrónicos. De esta forma se podrá personalizar la interfaz y permitir la gestión de los archivos pendientes de firmar, las listas de los ya firmados, etc.
- Implementar un procedimiento de comunicación segura extremo a extremo con la tarjeta inteligente, lo que asegure aún más la verificación de la identidad tanto del servidor, como del documento y el propio usuario firmante.

ACRÓNIMOS

APDU	Application Protocol Data Unit
API	Application Programming Interface
CLA	Class
CPU	Central Processing Unit
DES	Data Encryption Standard
DNI	Documento Nacional de Identidad
EEPROM	Electrically Erasable Programmable Read Only Memory
ETSI	European Telecommunications Standards Institute
GCM	Google Cloud Messaging
GP	Global Platform
GPL	General Public License
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
INS	Instruction
ISO	International Organization for Standardization
JCRE	Java Card Runtime Environment
JCVM	Java Card Virtual Machine
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JVM	Java Virtual Machine

NFC	Near Field Communication.
PIN	Personal Identification Number
RAM	Random Access Memory
ROM	Read Only Memory
RSA	Rivest Shamir Adleman
SIM	Subscriber Identity Module
SQL	Structured Query Language
SW	Status Word
URL	Uniform Resource Locator

BIBLIOGRAFÍA

[1] *Boletin Oficial Del Estado sobre Firma Electronica 59/2003*. Madrid 2003. https://www.boe.es/boe/dias/2003/12/20/pdfs/A45329-45343.pdf

[2] *Boletin Oficial Del Estado Ley Procedimiento Administrativo 39/15*. Madrid; 2015. https://www.boe.es/boe/dias/2015/10/02/pdfs/BOE-A-2015-10565.pdf

[3] Castillo Carlos. JavaCard Informe CC51B. Departamento ciencias de la computación Universidad de Chile. http://users.dcc.uchile.cl/~rbaeza/cursos/proyarq/ccastill/informe2.html

[4] Chen Zhiqun. *Java Card Technology For Smart Cards*. 18 septiembre 2000.

[5] Chirico U. *Smart Card Programming*. 2014.

[6] Cloud Messaging | Google Developers. *Google Developers*. https://developers.google.com/cloud-messaging

[7] ETSI - European Telecommunications Standards Institute. *ETSI*. http://www.etsi.org/

[8] Figueira Carlos. *Tarjetas Inteligentes*. Universidad Simon Bolivar; http://ldc.usb.ve/~figueira/cursos/Seguridad/Material/TarjetasInteligentes.pdf

[9] Foundation Node. Node.js. *Nodejsorg*. https://nodejs.org/en

[10] Gayoso Martínez Víctor, Implementación de tarjetas inteligentes Java Card de protocolos de cifrado y descifrado basado en curvas elípticas, Tesis Doctoral Universidad Politécnica de Madrid, 2010.

[11] *Global Platform Card Specification*. http://www.win.tue.nl/pinpasjc/docs/GPCardSpec_v2.2.pdf

[12] Google API Console. https://console.developers.google.com

[13] ISO International Organization for Standars. *Isoorg*. http://www.iso.org/iso/home.html

[14] Java Card Overview. *Oraclecom*. http://www.oracle.com/technetwork/java/embedded/javacard/overview/index.html

[15] Java Card 2.2 Runtime Environment (JCRE) Specification. https://www.informatik.uni-augsburg.de/lehrstuehle/swt/se/teaching/fruehere_semester/ws0708/javacard/Dokumentation/JCRESpec.pdf

[16] Java Card 2.2.1 Specification. https://www.informatik.uni-augsburg.de/lehrstuehle/swt/se/teaching/fruehere_semester/ws0506/javacard/Dokumentation/JavaCard211API.pdf

[17] MySQL : Developer Zone. *Devmysqlcom*. http://dev.mysql.com/

[18] Open Platform. http://downloads.acs.com.hk/technology/486-05-open-platform-smart-card.pdf

[19] Paredes Gibran. *INTRODUCCIÓN A LA CRIPTOGRAFÍA*. Revista Digital Electrónica; 2006. http://www.revista.unam.mx/vol.7/num7/art55/jul_art55.pdf