DIRECCIÓN COMERCIAL

DEPARTAMENTO DE DOCUMENTOS DE IDENTIFICACIÓN / TARJETAS

DESCRIPCIÓN TÉCNICA DE APLICACIONES ANDROID SOBRE DNIE V3.0

1 ÍNDICE DEL DOCUMENTO

2 INTRODUCCIÓN.

El Documento Nacional de Identidad es un documento con una antigüedad de más de 50 años, y está presente en la mayoría de las relaciones comerciales y administrativas, y su número figura en un altísimo porcentaje de las bases de datos de entidades y organismos públicos y privados.

El Documento Nacional de Identidad es el único documento de uso generalizado en todos los ámbitos a nivel nacional y referente obligado para la expedición de otros documentos (pasaporte, permiso de conducir, seguridad social, NIF, etc.).
De esta forma se puede afirmar que el Documento Nacional de Identidad goza de una plena aceptación en la sociedad española.

El DNI electrónico, además de la capacidad de identificación física de su titular, posee la capacidad de identificación en medios telemáticos y de firmar electrónicamente como si de una firma manuscrita se tratase. De esta forma garantiza que la personalidad del firmante no es suplantada.

En la actualidad, **ya se han expedido más de 45 millones de documentos** nacionales de identidad DNI electrónicos, y todos los ciudadanos españoles están obligados a tenerlo.

El actual DNI electrónico es un dispositivo con "Dual Interface":
El interfaz con contactos permite mantener la compatibilidad hacia atrás, es decir el uso del DNIe mediante un lector de tarjetas inteligentes, conectado a un puerto del ordenador, mientras que el interfaz sin contactos (contactless) es el que permite incluir las mismas funcionalidades, pero para dispositivos con tecnología NFC.

Además, el DNIe tiene una estructura de datos equivalente al pasaporte. Así que puede realizar funciones de Documento de Viaje, y se permite su uso en los Pasos Rápidos de Frontera (ABC systems) de forma totalmente equivalente a el pasaporte electrónico actual.

3 OBJETIVO DEL DOCUMENTO

En la actualidad se ha generalizado por completo el uso de los dispositivos móviles para el acceso a internet. Esto permite a cualquier usuario estar siempre conectado y tener disponibles multitud de servicios telemáticos. En función del nivel de seguridad que requieran dichos servicios, éstos pueden necesitar de la autenticación del usuario, lo que hace que el **Documento Nacional de Identidad (DNIe 3.0)** pueda convertirse en una pieza clave.

La incorporación de la **tecnología NFC** a los dispositivos móviles de última generación elimina las barreras del lector, drivers, etc. facilitando la conexión online y la autenticación del ciudadano. Por su parte, el **DNIe 3.0** tiene un chip *dual-interface* que permite su utilización tanto con contactos como en modo *contactless*.

Figura 1: Reverso DNIe 3.0 Dual Interface

El objetivo de este documento es detallar la arquitectura y el funcionamiento de una aplicación **Android con soporte al DNIe v3.0 mediante NFC e indicar los requisitos funcionales para su adaptación al sistema operativo iOS de Apple**.

El documento está dividido en dos partes. Por un lado incluye una descripción de la arquitectura necesaria y el diseño interno de la aplicación Android. Estas aplicaciones estarán siempre apoyadas en el **conector DNIeDroid**, un *middleware* diseñado para la comunicación con el DNI electrónico.

Por otro lado tenemos la descripción funcional de la **aplicación** "**MuniMadrid**", un portal de ejemplo de gestión de consultas con el ayuntamiento de Madrid. Esta aplicación requeriría de la autenticación mediante DNI electrónico. En esta parte del documento se puede comprobar el proceso completo de solicitud de acceso al documento, autenticación y conexión securizada con el servicio web.

Para información relativa a los certificados X509 incluidos en el DNIe 3.0, véase **[1] RFC 2459. Internet X.509 Public Key Infrastructure. Certificate and CRL Profile**. En cuanto a la conexión por radiofrecuencia entre dispositivos, ver **[2] ISO 14443 - Tarjetas de Identificación. Tarjetas sin contacto. Tarjetas de proximidad.**

4 REQUISITOS ANDROID

Los requisitos de funcionamiento para el entorno son los siguientes:

- **Dispositivo Android** con tecnología **NFC**,
- Aplicación "**MuniMadrid**".
- **DNIe v3.0** del ciudadano,

La autenticación del usuario para el acceso a servicios del Ayuntamiento se realiza mediante el uso del **DNI electrónico v3.0**. Este DNIe funciona mediante **NFC**, (*Near Field Communication*, ver **[3] NFC Forum Type Tags - White Paper v1.0. NXP Semiconductors**) lo que permite a cualquier dispositivo que disponga de esta tecnología, actuar como lector de tarjetas.

Las comunicaciones entre el dispositivo y el DNIe se realizarán siempre cifradas según la norma **[4] CWA 14890-1: Application Interface for smart cards used as Secure Signature Creation Devices**.

En este documento nos centraremos en la versión inalámbrica del DNIe, en la que la interacción desde el punto de vista físico se realiza por medio de NFC.

5 MIDDLEWARE DNIeDROID

5.1 ARQUITECTURA LÓGICA

El **DNIeDroid** es un *middleware* encargado de gestionar la conexión entre dispositivos móviles y el DNIe. Ofrece un API básico de operaciones que permite a los desarrolladores gestionar de manera transparente las conexiones NFC con el DNIe.

De esta manera se pretende facilitar la implementación de aplicaciones, ya que no hará falta conocer en detalle el funcionamiento de la tarjeta sino que bastará con incluir el DNIeDroid en la aplicación para poder acceder a las funcionalidades del DNIe.

A continuación se muestra un diagrama en el que se describe la arquitectura lógica actual del componente, desarrollada en Java para Android.

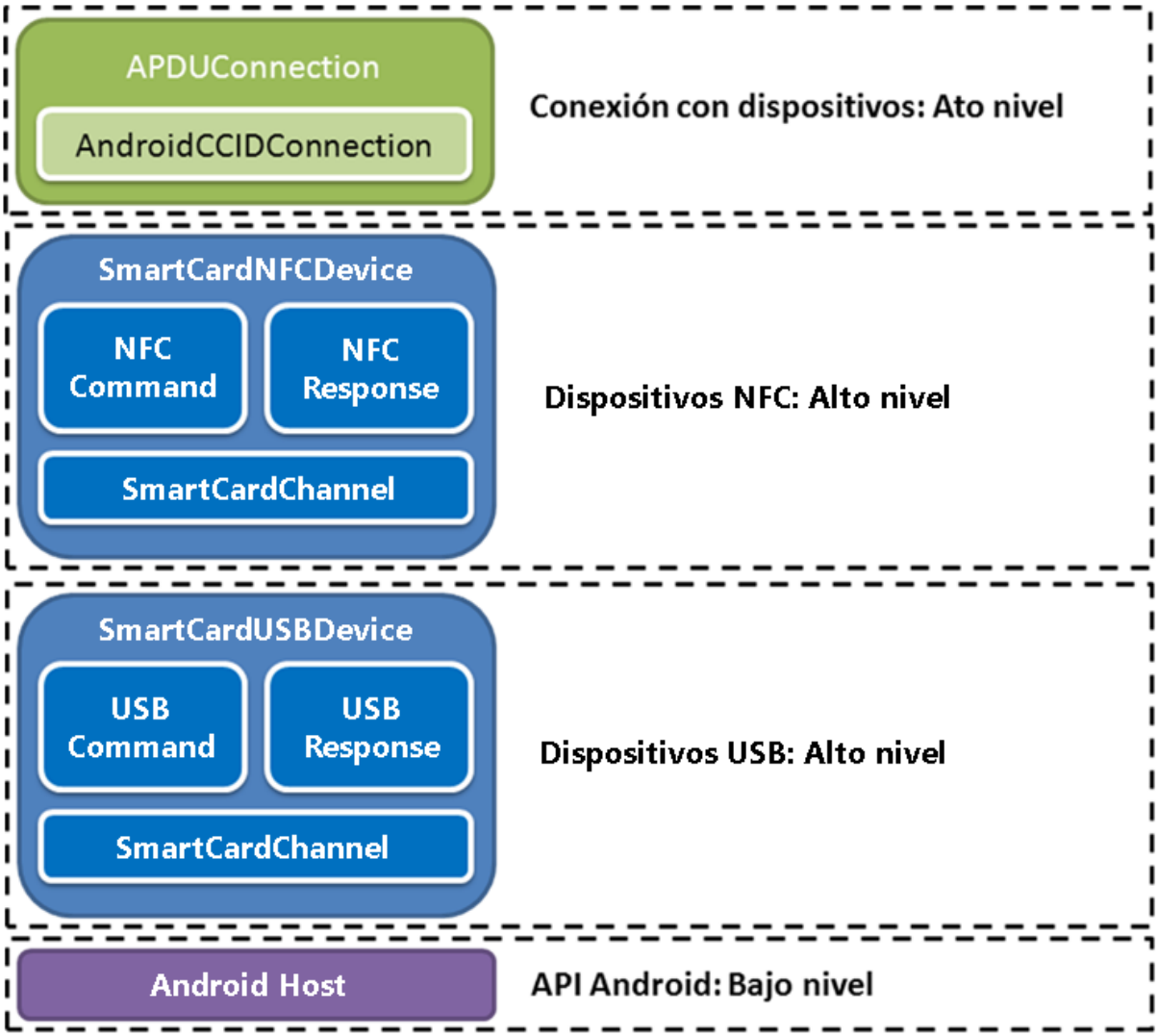

Figura 2: Arquitectura lógica DNIe Droid

DNIeDroid ofrece una capa de alto nivel para la gestión de la conexión con dispositivos lectores de tarjetas. Esta capa, se apoya en una representación de "alto nivel" de los dispositivos, gestionando el envío y recepción de comandos a través del canal de comunicación (USB o NFC). Esta gestión se lleva a cabo gracias a la capa de abstracción de bajo nivel proporcionada por Android, y que en su caso debería proporcionar también iOS.

El objetivo del componente DNIeDroid es interactuar con el DNIe en dispositivos basados en Java (como *smartphones y tablets Android*), ya sea mediante interfaz USB o mediante NFC. De esta manera se facilita el acceso desde esas plataformas a los servicios que hacen uso del DNIe para autenticación y firma electrónica.

En el siguiente diagrama de describe el modo de integración del DNIeDroid en DD4J.

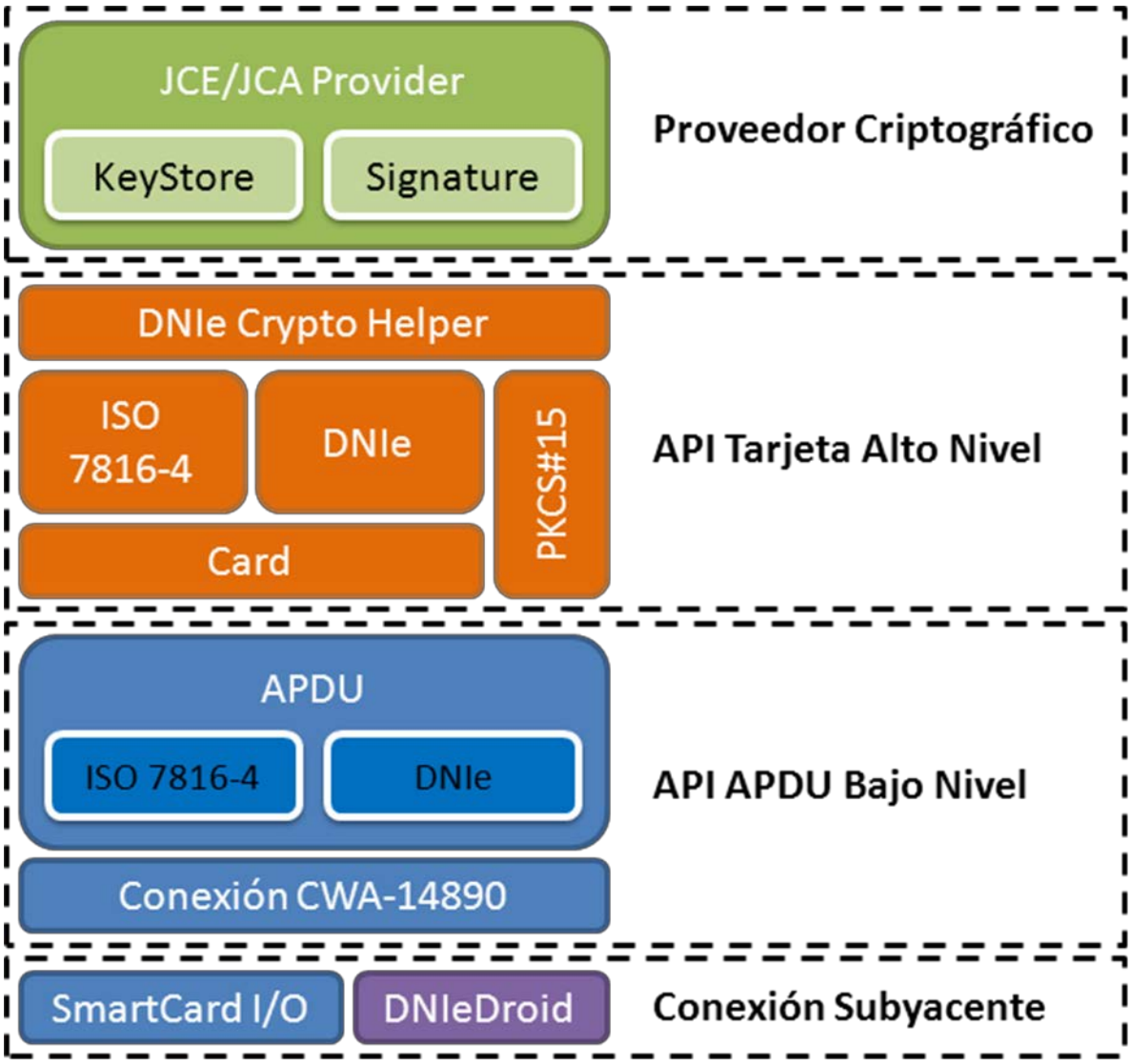

Figura 3: Integración en Android

Tal como se observa en el diagrama, el componente **DNIeDroid** se integra como una conexión subyacente más dentro de la arquitectura del Controlador Java para el DNIe.

Se abstrae así a las capas superiores de la interacción con los lectores de tarjetas conectados a los dispositivos Android (ya sea vía USB o NFC) y la gestión de intercambio de comandos de bajo nivel. Es esta última capa la que da acceso al API de NFC en Android y que nos permite el envío de comandos y respuestas entre DNIe y dispositivo.

5.2 ARQUITECTURA FÍSICA

A pesar de que el componente es puramente lógico, está diseñado para la comunicación de dispositivos Android mediante dos posibles interfaces.

En primer lugar con **lectores de tarjetas USB** conectados a él mediante un adaptador USB-OTG (On The Go), ya que la mayoría de dispositivos Android disponen de un puerto micro USB en lugar de USB y no hay, a día de hoy, lectores de SmartCard micro USB. En caso de que el dispositivo disponga de un puerto USB, no haría falta dicho adaptador.

En segundo lugar mediante la conexión por proximidad con un DNIe v3.0 mediante **tecnología NFC**. Esta tecnología permite la comunicación entre el dispositivo Android y el DNIe sin necesidad de lector de tarjetas. Para más información véase **[3] NFC Forum Type Tags - White Paper v1.0. NXP Semiconductors**.

6 APLICACIÓN ANDROID

6.1 ARQUITECTURA LÓGICA

En esta parte del documento describiremos la arquitectura lógica de una aplicación *Android* y su utilización del middleware *DNIeDroid*. Como ejemplo tomaremos la app ***MuniMadrid***, desarrollada por la FNMT-RCM. Esta aplicación corresponde a un portal de ejemplo del ayuntamiento de Madrid, desde el que podemos acceder no sólo a información general sino principalmente a servicios para el ciudadano que requieren autenticación.

A continuación se muestra un diagrama en el que se muestra la arquitectura lógica de la aplicación. Aunque está enfocado a esta app *de ejemplo,* el diseño y la arquitectura sería idéntica en cualquier otra aplicación que hiciese uso del *DNIeDroid*.

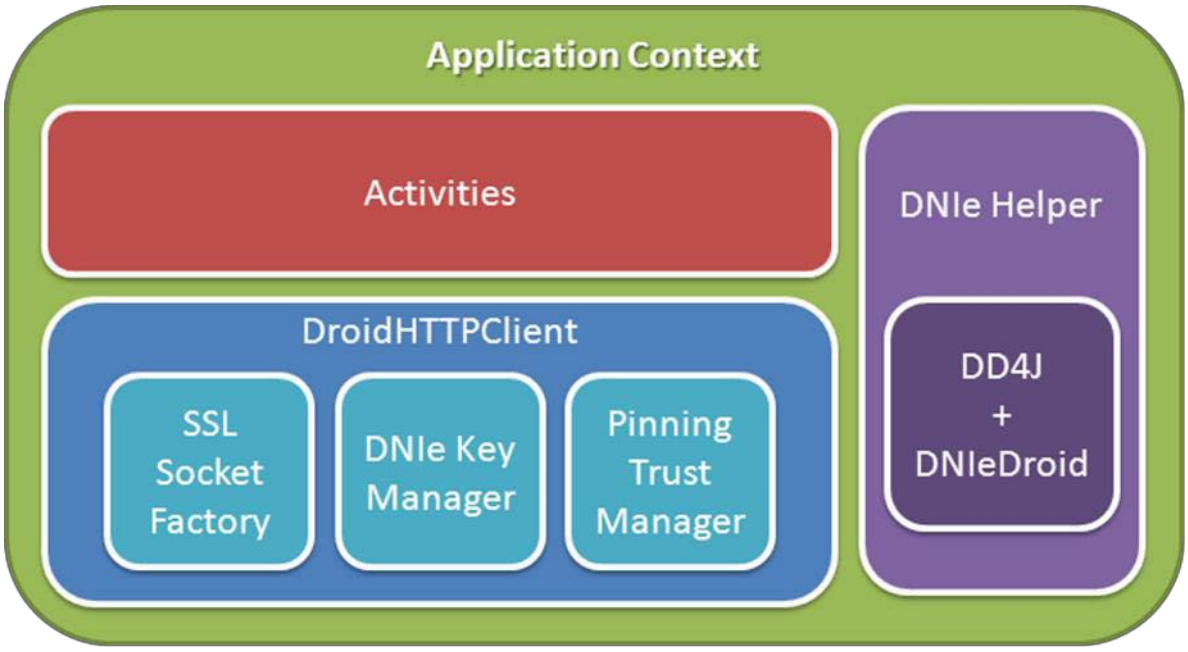

Figura 4: Arquitectura lógica App sobre DNIeDroid

En esta figura podemos ver las partes que integran la aplicación. Dado que nuestro ejemplo se trata de una aplicación Android, el punto de contacto con el usuario son las *Activities*. El componente **DNIeHelper** está disponible para todas ellas, y gestiona la interacción con el DNIe y los cuadros de diálogo que se muestran al usuario durante esta interacción.

Por otro lado, aquellas *Activities* que requieran el uso del DNIe para la autenticación en servidores de terceros, se pone a su disposición el componente **DroidHTTPClient,** que gestiona esta problemática y que también se apoya en el componente de interacción con el DNIe.

6.2 API DE COMANDOS NFC (ANDROID)

A continuación incluimos el API que ofrece Android para la comunicación vía NFC. Básicamente consiste en una serie de clases que gestionan la conexión (**NFCManager, NFCAdapter, IsoDep**…) y un método para el envío de comandos y recepción de respuestas (**IsoDep::transceive**).

Para obtener una información más detallada sobre el API proporcionado en Android para el soporte NFC puede consultarse la documentación online del enlace **[8] android.NFC. http://developer.android.com/reference/android/nfc/package-summary.html** "Portions of this page are reproduced from work created and shared by the Android Open Source Project and used according to terms described in the Creative Commons"

6.2.1 Package Android.nfc

Provides access to Near Field Communication (NFC) functionality, allowing applications to read NDEF message in NFC tags. A "tag" may actually be another device that appears as a tag.
For more information, see the Near Field Communication guide.

Here's a summary of the classes:
NfcManager
This is the high level manager, used to obtain this device's NfcAdapter. You can acquire an instance using getSystemService(String).
NfcAdapter
This represents the device's NFC adapter, which is your entry-point to performing NFC operations. You can acquire an instance with getDefaultAdapter(), or getDefaultAdapter(android.content.Context).

6.2.2 Public final class NfcManager

extends Object
java.lang.Object

↳ android.nfc.NfcManager
Class Overview
High level manager used to obtain an instance of an NfcAdapter.
Use getSystemService(java.lang.String) with NFC_SERVICE to create an NfcManager, then call getDefaultAdapter() to obtain the NfcAdapter.
Alternately, you can just call the static helper getDefaultAdapter(android.content.Context).

6.2.3 Public final class NfcAdapter

extends Object
java.lang.Object

↳ android.nfc.NfcAdapter
Class Overview
Represents the local NFC adapter. Use the helper getDefaultAdapter(Context) to get the default NFC adapter for this Android device.

6.2.3.1 Public final class IsoDep

extends Object
implements TagTechnology
java.lang.Object

↳ android.nfc.tech.IsoDep
Class Overview
Provides access to ISO-DEP (ISO 14443-4) properties and I/O operations on a Tag.
Acquire an IsoDep object using get(Tag).
The primary ISO-DEP I/O operation is transceive(byte[]). Applications must implement their own protocol stack on top of transceive(byte[]).
Tags that enumerate the IsoDep technology in getTechList() will also enumerate NfcA or NfcB (since IsoDep builds on top of either of these).

6.2.3.2 Public byte[] transceiver (byte[] data)

Added in API level 10
Send raw ISO-DEP data to the tag and receive the response.
Applications must only send the INF payload, and not the start of frame and end of frame indicators. Applications do not need to fragment the payload, it will be automatically fragmented and defragmented by transceive(byte[]) if it exceeds FSD/FSC limits.
Use getMaxTransceiveLength() to retrieve the maximum number of bytes that can be sent with transceive(byte[]).
This is an I/O operation and will block until complete. It must not be called from the main application thread. A blocked call will be canceled with IOException if close() is called from another thread.
Requires the NFC permission.
Parameters
data command bytes to send, must not be null

Returns
response bytes received, will not be null
Throws
if the tag leaves the field
IOException if there is an I/O failure, or this operation is canceled

7 APLICACIÓN IOS

7.1 ARQUITECTURA LÓGICA

La arquitectura lógica de una aplicación en iOS sobre NFC sería análoga a la versión de Android, utilizando en su caso el API que proporcionase Apple. Bastaría con que el sistema operativo **iOS ofrezca un interfaz de comunicación** con el lector NFC que permita:

- **gestionar la comunicación** con el NFC (conexiones, *timeouts*, etc.)

- **enviar y recibir bytes** en modo raw.

Esto último es importante de cara a que el hardware no filtre los mensajes, ya que debemos soportar tanto comandos con formato NFC Forum (ver [3] NFC Forum Type Tags - White Paper v1.0. NXP Semiconductor) y formatos tipo ISO ([6] ISO/IEC 7816-4. Identification cards – Integrates circuit cards. Part 4: Organization, security and commands for interchange).

Hay que indicar que la tarjeta DNIe se basa en comandos ISO-7816, un conjunto más amplio que el incluido en NFC Forum, y que da soporte a operaciones criptográficas, firmas, autenticaciones, etc.

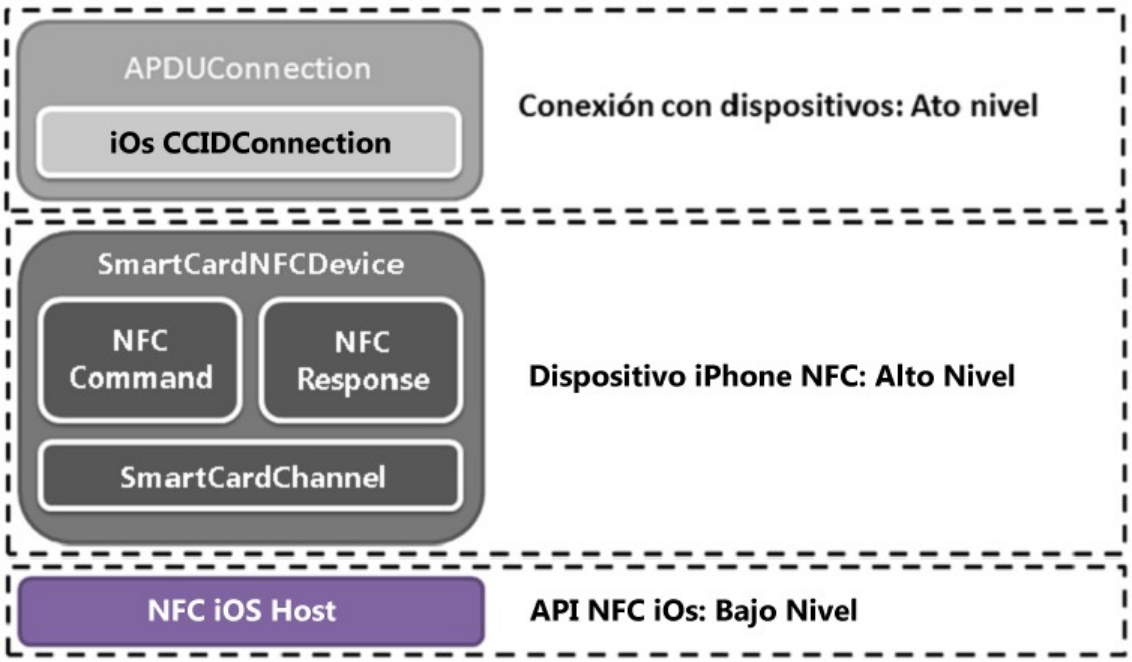

Figura 5: Posible arquitectura lógica sobre iOS.

Una vez que tuviésemos el **API de conexión NFC**, los cambios que se requieren en el middleware *DNIeDroid* para su funcionamiento en iOS serían mínimos. Esto nos permitiría desarrollar rápidamente multitud de aplicaciones para el DNIe 3.0.

8 ARQUITECTURA FÍSICA APP (ANDROID/iOS)

En cuanto a la arquitectura física, se mostrarán los componentes físicos necesarios para la interacción de la aplicación con el DNIe. Esa interacción permitirá la conexión del DNIe con sitios web externos para la consulta de datos del ciudadano, realización de firmas electrónicas, etc.

La aplicación utilizada como ejemplo, ***MuniMadrid,*** debe estar instalada en un dispositivo Android con tecnología NFC (Smartphone o Tablet). De esta manera la conexión entre el smartphone o tablet con el DNIe se realizará mediante proximidad.

También sería posible desarrollar la aplicación para trabajar con DNIe mediante un lector de tarjetas USB conectado a él mediante un adaptador USB-OTG. En este documento nos centraremos sólo en la versión para DNIe v3.0 y **tecnología NFC**.

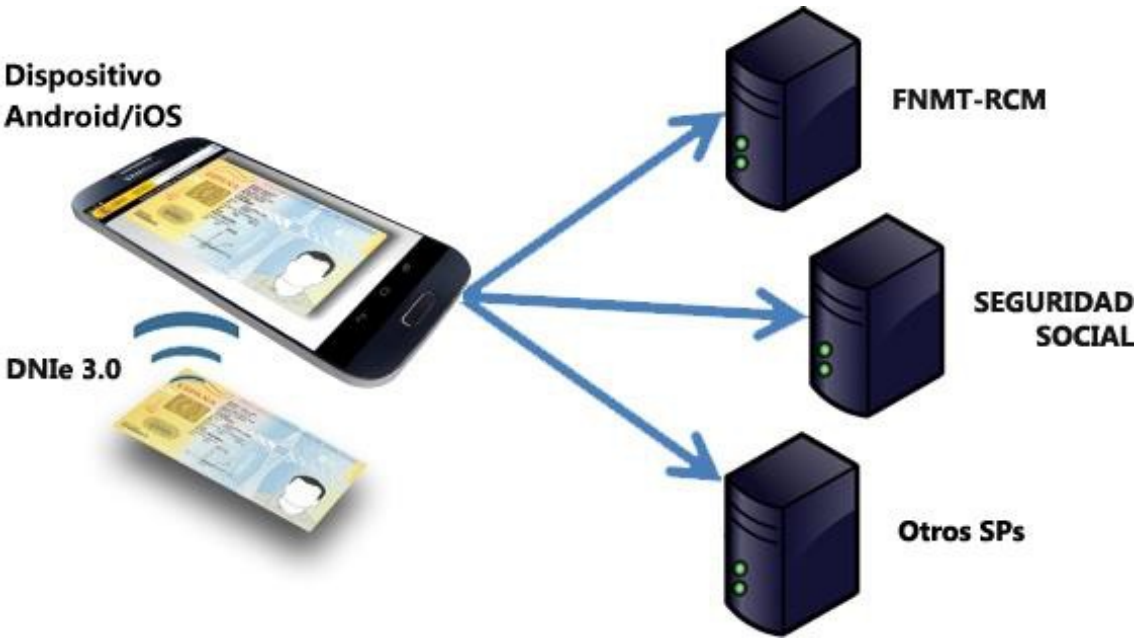

Figura 6: Arquitectura física del sistema

Para el acceso a los servicios al ciudadano que ofrece el la aplicación **MuniMadrid** será necesaria la conexión segura mediante protocolo *https* con los servidores del Ayuntamiento.

Esa conexión requerirá la autenticación con el DNIe del ciudadano mediante el uso de la tecnología NFC.

9 EJEMPLO DE FUNCIONAMIENTO: MUNIMADRID

9.1.1 Menú Principal

La pantalla principal de la aplicación muestra las opciones básicas de funcionamiento. Aquí podemos ver todos los servicios accesibles para el ciudadano, ya sean de información general o de acceso protegido mediante DNIe.

Figura 7: Pantalla principal App MuniMadrid

Las opciones disponibles son:

- Consulta de multas,
- Datos censales,
- Tributos,
- Ayuntamiento
- Acceso al portal web “¡Madrid!”

9.1.2 Selección de CAN

Una vez elegida la operación a realizar, la aplicación nos solicita el **CAN (*Card Access Number*)** del documento con el que nos vamos a autenticar. Este identificador es un número de 6 dígitos que aparece en el anverso del documento físico del DNIe y se utiliza para establecer canal cifrado PACE entre el documento y el dispositivo.

Figura 8: Card Access Number

La aplicación nos muestra un listado de los CAN's ya utilizados, evitando así tener que introducirlo cada vez.

Figura 9: Selección del CAN

Para más información sobre el establecimiento del canal **PACE (*Password Authenticated Connection Establishment*)** ver **[5] Technical Guideline TR-03110-2.Advanced Security Mechanisms for Machine Readable Travel Documents –P2**.

9.1.3 Autenticación del DNIe mediante PIN

Cuando ya hemos indicado el *CAN (Card Access Number)* del documento que vamos a utilizar, la aplicación nos solicitará que aproximemos el DNIe al dispositivo.

Figura 10: Lectura del DNIe por proximidad

Al aproximar el documento al dispositivo, se establecerá la comunicación entre ambos permitiendo el paso de comandos y respuestas entre ambos. Por motivos de seguridad la distancia máxima soportada por NFC es, dependiendo del dispositivo, de alrededor de un centímetro.

Ya hemos comentado anteriormente que las comunicaciones entre DNIe y el dispositivo irán siempre cifradas según **[4] CWA 14890-1: Application Interface for smart cards used as Secure Signature Creation Devices.**

Para acceder a las operaciones de firma del DNIe es necesario que el ciudadano se autentique presentando el ***PIN (Personal Identification Number)*** de su documento.

Figura 11: Inserción del PIN

Una vez que éste se ha autenticado se realizará la conexión segura con el servidor del prestador de servicios (Ayuntamiento de Madrid, en este caso), realizándose las operaciones criptográficas necesarias para establecer el canal SSL y obtener la información personal.

Si la operación se ha completado satisfactoriamente, la aplicación nos mostrará la información solicitada. En caso contrario nos devolverá un mensaje de error.

Figura 12: Resultado correcto de la consulta

10ESQUEMA DE FUNCIONAMIENTO: MUNIMADRID

10.1 CONSULTA DE MULTAS. CORRECTA.

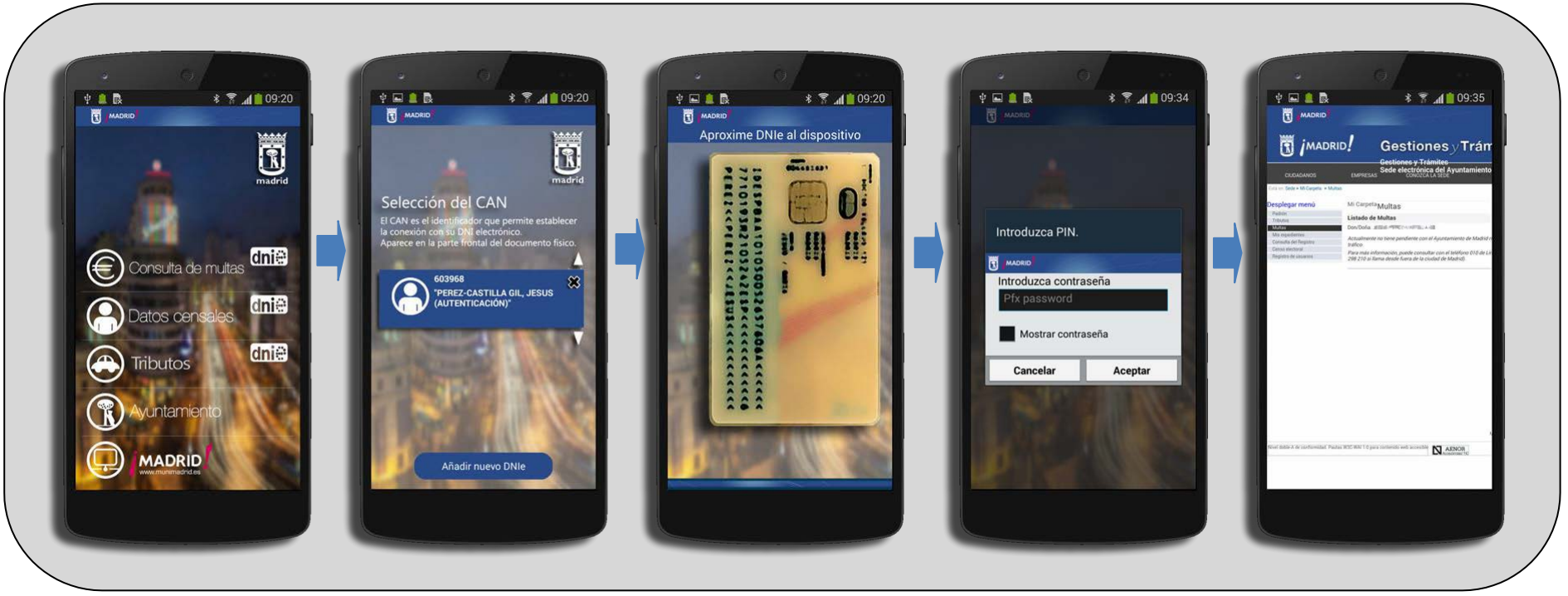

Figura 13: Consulta de multas del Ayuntamiento de Madrid

10.2 CONSULTA DE MULTAS. ERRÓNEA.

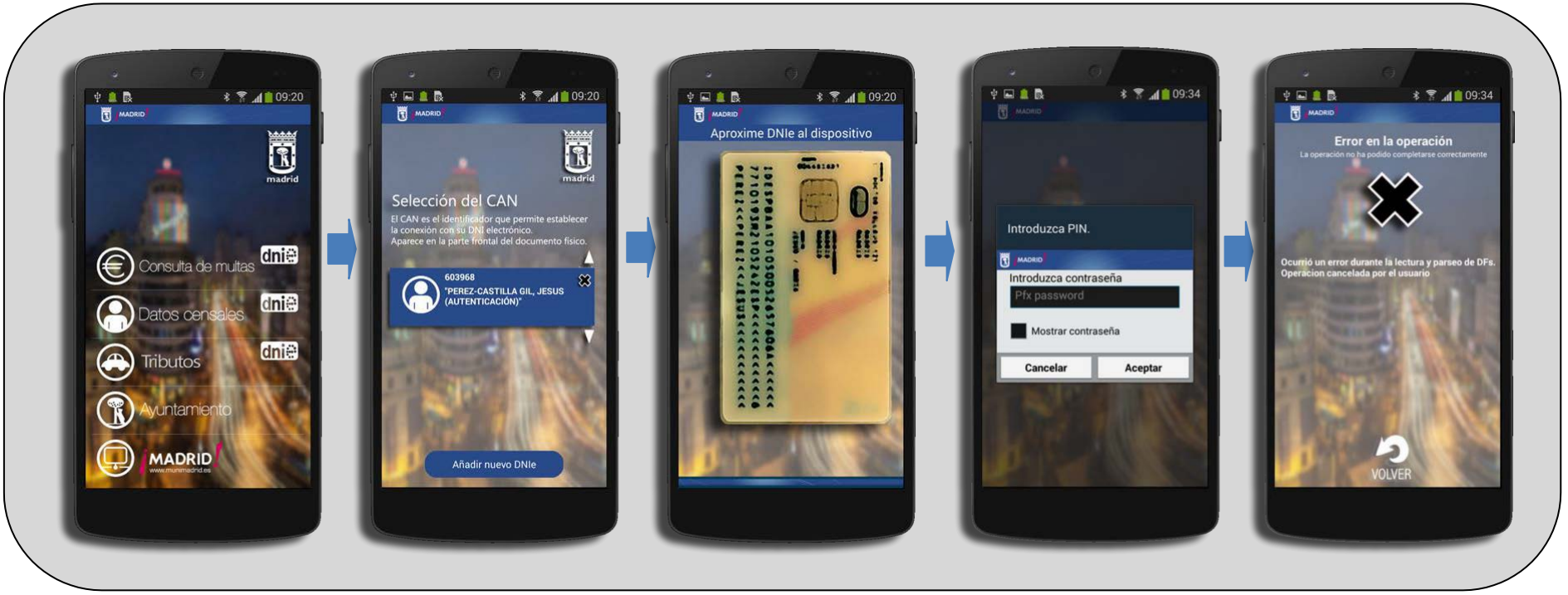

Figura 14: Error en el acceso al DNIe

11REFERENCIAS

[1] RFC 2459. Internet X.509 Public Key Infrastructure. Certificate and CRL Profile

[2] ISO 14443 - Tarjetas de Identificación. Tarjetas sin contacto. Tarjetas de proximidad

[3] NFC Forum Type Tags - White Paper v1.0. NXP Semiconductors

[4] CWA 14890-1: Application Interface for smart cards used as Secure Signature Creation Devices

[5] Technical Guideline TR-03110-2.Advanced Security Mechanisms for Machine Readable Travel Documents –Part 2

[6] ISO/IEC 7816-4. Identification cards – Integrates circuit cards. Part 4: Organization, security and commands for interchange.

[7] ISO/IEC18092 Information technology — Telecommunications and information exchange between systems — Near Field Communication — Interface and Protocol (NFCIP-1)

[8] Android.NFC. http://developer.android.com/reference/android/nfc/package-summary.html

[9] UIT-T X.509 (11/2008) – Open systems interconnection: Public-key and attribute certificate frameworks

12 TABLA DE FIGURAS