ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Proyecto Fin de Carrera

Implementación y análisis de clientes genéricos para el acceso seguro a Web Services

(Implementation and analysis of genereric client for secure access to Web Services)

Para acceder al Título de

INGENIERO DE TELECOMUNICACIÓN

Autor: Sergio Cosio del Olmo

Octubre – 2015

INGENIERÍA DE TELECOMUNICACIÓN

CALIFICACIÓN DEL PROYECTO FIN DE CARRERA

Realizado por: Sergio Cosio del Olmo
Director del PFC: Alberto Eloy Garcia Gutierrez
Título: "Implementación y análisis de clientes genéricos para el acceso seguro a Web Services"
Title: "Implementation and analysis of genereric client for secure access to Web Services"

Presentado a examen el día: 27-10-2015

para acceder al Título de

INGENIERO DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Marta García Arranz
Secretario (Apellidos, Nombre): Alberto Eloy García Gutiérrez
Vocal (Apellidos, Nombre): Pedro Corcuera Miró

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del PFC (sólo si es distinto del Secretario)
Vº Bº del Subdirector	Proyecto Fin de Carrera Nº (a asignar por Secretaría

Agradecimientos

Antes de finalizar mi etapa de estudiante con la redacción del presente Proyecto de Fin de Carrera me gustaría dedicar unas líneas de agradecimiento a todas aquellas personas que han sido claves para mí a lo largo de mi estancia en la universidad.

En primer lugar me gustaría agradecer a mi familia, especialmente mis padres Loli y Miguel, mi hermano Rubén y como no, a mi pareja Irene, el apoyo mostrado, no sólo durante estos últimos meses en los cuales he desarrollado este proyecto, sino durante toda mi época universitaria. Gracias por haberme escuchado y animado en los momentos más difíciles, y comprender que en ciertas ocasiones he sentido presión y mi carácter no ha sido el más adecuado.

A continuación me gustaría dar las gracias a mi tutor Alberto por darme la oportunidad de realizar este proyecto y por toda la ayuda y paciencia prestada durante su realización.

Por último agradecer a los todos los compañeros de la universidad, que siempre han estado ahí para solucionar dudas, prestarnos apuntes y también por qué no, para pasar un buen rato. En especial a Eugenio, por todos los momentos que hemos pasado juntos en estos últimos años, y a Pablo por su ayuda en la realización del presente proyecto.

A todos vosotros, muchas gracias.

RESUMEN

La aparición de los Smartphones o teléfonos inteligentes ha supuesto una auténtica revolución en la sociedad actual. Los dispositivos móviles han dejado de ser un aparato que únicamente se emplea para comunicarse por voz o texto, convirtiéndose en pequeños ordenadores de bolsillo personalizables mediante la instalación de aplicaciones que permiten realizar infinidad de tareas.

Sin embargo, los mismos fabricantes han desencadenado una guerra abierta para adueñarse del mercado, implantando sistemas operativos a medida, adaptados a cada dispositivo pero incompatibles con los de otros fabricantes. La guerra entre Android, IOS y Windows Phone obliga a los desarrolladores a generar códigos adaptados a cada sistema, multiplicando el esfuerzo y el coste de las denominadas Apps.

En este ámbito surge la idea de desarrollar aplicaciones cuyo código, y por tanto su desarrollo, sea independiente del sistema operativo. De forma integrada dentro del proyecto MobiWallet, cuyo desarrollo ya estaba iniciado, se procedió a implementar una aplicación hibrida similar a una nativa ya existente. Como condición fundamental debería cumplir todos los requerimientos definidos por el proyecto, y utilizarse en la mayor cantidad de dispositivos posibles, sin que el sistema operativo sea una limitación.

El proyecto MobiWallet tiene por objetivo unificar el transporte público en una única plataforma donde la interoperabilidad no sea un problema, aunque partía de una aplicación nativa para dispositivos Android. Este PFC implementa las funcionalidades de dicha aplicación y permite a los usuarios registrarse en la plataforma MobiWallet y acceder a todas las funcionalidades que se ofrecen en ésta. La compra de billetes, recargas de saldo, consulta de tarifas y movimientos, por ejemplo, son ahora compatibles con los principales sistemas operativos móviles del mercado.

ABSTRACT

Emergence of smartphones has brought a revolution in today's society. Mobile devices are no longer used to communicate by calls or sms only, becoming small handheld computers, which can be customized by app installations allowing many tasks

However, manufacturers have started an open warfare to control the market, deploying operating systems for each device, but fully incompatible with those of other manufacturers. The war between Android, iOS and Windows Phone requires that developers have to generate adapted code for each system, multiplying the effort and cost of new Apps

In this context arises the idea of developing applications whose code, and therefore its development would be independent of the operating system. Integrated within the project MobiWallet, whose development was begun, this thesis proceed to implement a hybrid application similar to an existing native application. As a fundamental condition should meet all the requirements defined by the project, wearable as many different devices as possible without the limitation based on operating system.

The aim of the MobiWallet project is to unify public transport in a single platform where interoperability would not be a problem. The starting point is a native application for Android devices. This thesis implements the functionality of the application allowing users to register on the MobiWallet platform with a complete access to all the offered features. Buying tickets, recharge balance, query tariffs and movements, for example, they are now compatible with all major mobile operating systems on the market.

Índice general

Índice de imágenes

1 INTRODUCCION Y OBJETIVOS

Este primer apartado sirve como introducción del presente proyecto. En primer lugar se expone brevemente el tema a tratar, y a continuación se explican las causas que han motivado su realización, así como los objetivos que se pretenden alcanzar.

Por último se hace un pequeño recorrido por el resto de apartados de la memoria explicando brevemente lo que se tratará en cada uno de ellos.

1.1 Introducción

Los dispositivos móviles de uso comercial aparecieron en la década de los años 80 con la idea de conseguir teléfonos sin cable, con una única función, comunicarse por medio de llamadas de voz. Las primeras innovaciones estaban destinadas a reducir el tamaño de los dispositivos y aumentar la duración de las baterías. Con el paso del tiempo estas innovaciones se centraron en incorporar en los dispositivos nuevas funcionalidades (mensajería instantánea, agenda, juegos, cámara fotográfica, acceso a internet, GPS, reproductor de video y audio…), hasta llegar a los actuales Smartphones o teléfonos inteligentes y sus millones de aplicaciones disponibles [1].

Esta constate evolución en la tecnología, ha estado ligada a una evolución en la sociedad, de tal forma que estos avances tecnológicos han ido apareciendo para resolver la necesidad de estar en continua conexión con la información y las comunicaciones, haciendo que los dispositivos móviles se conviertan en un dispositivo imprescindible en nuestra vida diaria.

En la actualidad los dispositivos móviles nos permiten realizar una gran cantidad de operaciones (compras, redes sociales, ocio y juegos, consulta de información…) y lo más habitual es que estas operaciones se efectúen empleando una aplicación móvil [2]. De hecho, en 2014 el uso de las aplicaciones móviles aumentó un 76% [3]. Debido a este crecimiento en el uso de aplicaciones en los últimos años, el objetivo del proyecto será estudiar, diseñar e implementar una aplicación móvil.

El principal problema a la hora de desarrollar una aplicación para un dispositivo móvil, es la gran variedad de sistemas operativos y la dependencia existente entre los recursos físicos del dispositivo (Hardware) y el propio sistema operativo (Software). Esto supone que si queremos una aplicación disponible para todas las plataformas, se deberán crear varias apps con el lenguaje de cada sistema operativo, lo que conlleva una gran inversión de recursos. Para evitar este problema, en este proyecto se estudia la posibilidad de desarrollar una aplicación híbrida multiplataforma y analizar sus posibles limitaciones.

Concretamente la aplicación se desarrollará dentro de la plataforma MobiWallet (Mobility and Transport Digital Wallet). MobiWallet es un proyecto financiado por la Comisión Europea dentro del Programa CIP (*Competitiveness and Innovation Framework Programme, as part of the Information and Communications Technology – Policy Support Programme, Call 7*), en el que la Universidad está enrolada junto con otro grupo de socios y entidades, tanto públicos como privados. Tiene como objetivo desarrollar soluciones sostenibles de transporte urbano basadas en la gestión de tarifas y cobros facilitando la interacción entre diferentes modos de transporte, todo ello en el ámbito de la Ciudad Inteligente [4]

La aplicación hibrida que se va a implementar, consiste en un prototipo, que tratará de acercase lo máximo posible a las prestaciones de la aplicación nativa que paralelamente está siendo desarrollada por Indra (Entidad Socia del proyecto MobiWallet). Esta aplicación permitirá el registro de nuevos usuarios y poder realizar operaciones de recarga, consulta de tarifas y compras en los principales servicios de transporte público de la ciudad de Santander (bus, taxi, bici, lancha y parking).

1.2 Motivación y objetivos del proyecto

La principal motivación a la hora de desarrollar este proyecto fue el tratar de evitar la fuerte dependencia que actualmente existe entre las aplicaciones nativas y los sistemas operativos de los dispositivos móviles. Concretamente dentro del proyecto MobiWallet se estaba llevando a cabo la implementación de una aplicación nativa "Android" por parte de la compañía Indra. Este desarrollo suponía que únicamente aquellos usuarios que dispusieran de dispositivos móviles con sistema operativo Android pudieran usarla. Con el fin de tratar de solucionar este problema, y conseguir una aplicación portable a la gran mayoría de dispositivos móviles surgió la idea de desarrollar este proyecto de forma paralela a la aplicación oficial.

Los principales objetivos que se pretendían alcanzar con el desarrollo de este proyecto fueron los siguientes:

- Investigar acerca de las principales tecnologías en el desarrollo de aplicaciones para dispositivos móviles, estudiando sus ventajas e inconvenientes, y analizando si la opción de diseñar una aplicación híbrida es la más acertada.
- Obtener un entorno de desarrollo que permita generalizar el código de tal forma que sea lo más reutilizable posible. Este entorno, además, debe permitir generar aplicaciones portables a la mayor parte de sistemas operativos existentes en los dispositivos móviles actuales.
- Desarrollar una aplicación que trate de cubrir la mayor parte de las funcionalidades planteadas dentro del proyecto MobiWallet, y que sea compatible con varios sistemas operativos.

  Derivados de este último objetivo, se definen otros más específicos:
  - La aplicación debe de contener las funcionalidades básicas de registro de usuarios y tarjetas, y consultas de estos datos.
  - Desarrollar las funciones de pago que permitan a los usuarios hacer recargas en sus tarjetas a través del TPV Virtual de Santander.
  - Trata de hacer las comunicaciones entre la aplicación y el servidor MobiWallet lo más seguras posibles.
  - Investigar hasta qué punto es posible manejar el Hardware de los dispositivos móviles mediante una aplicación desarrollada en un lenguaje diferente al definido por el sistema operativo.

1.3 Organización del documento

A lo largo de la memoria se tratará de explicar aspectos tanto prácticos como teóricos que ayuden a entender el desarrollo y funcionamiento de la aplicación diseñada. Para facilitar la comprensión, este documentos se ha divido en diferentes capítulos y en este punto describiremos los temas que se tratan en cada uno de ellos.

En primer lugar se describirán los **conceptos teóricos** del proyecto. Concretamente se definirá el término aplicación y se analizará su situación actual y sus tipos, así como los diferentes dispositivos móviles y sus sistemas operativos. También se estudiarán aspectos como la seguridad y la conectividad.

En el siguiente capítulo se explicaran los **aspectos prácticos** del proyecto. Este capítulo esta a su vez divido en dos grandes apartados. En el primero de ellos se explica el Proyecto MobiWallet y su situación actual, mientras que el segundo se centra en los requerimientos que debe cumplir la aplicación.

A continuación se describirá la **implementación** de la aplicación. En primer lugar se introducirá el Framework de desarrollo, así como las tecnologías empleadas. A continuación se irá explicando cómo está estructurada la aplicación y sus diferentes funcionalidades. Por último se describirán los problemas encontrados en aspectos como la seguridad, el manejo del chip NFC o la pasarela de pago.

La siguiente parte se dedicará a describir las diferentes utilidades empleadas para la **verificación** del correcto funcionamiento de la aplicación.

Por último, se concluirá con el apartado de **conclusiones y líneas futuras**, donde se analizaran si se han logrado los objetivos planteados, y se plantearán posibles mejoras que puedan llevarse a cabo.

2 CONCEPTOS TEÓRICOS

A lo largo de este capítulo se estudiarán diferentes aspectos teóricos relacionados con las aplicaciones de forma que se obtengan los conocimientos necesarios, para poder determinar cómo llevar a cabo el desarrollo de una aplicación.

2.1 Las aplicaciones

En primer lugar, ya que el proyecto se centra en el desarrollo de una aplicación móvil, es conveniente introducir el término aplicación, así como ciertos datos relevantes.

Una aplicación (también llamada app) es simplemente un programa informático creado para llevar a cabo o facilitar una tarea en un dispositivo [5]. En el caso de las aplicaciones móviles se diseñan para ser ejecutadas en teléfonos, tablets y otros dispositivos móviles. Las aplicaciones se desarrollan para satisfacer necesidades concretas de los usuarios, y su fin es facilitar o permitir la ejecución de tareas en las que un programador ha detectado una cierta necesidad, pueden ser necesidades lúdicas (los juegos son considerados aplicaciones), laborales…En definitiva, como se suele decir, que para cada problema hay una solución, en el campo de la informática podemos adaptar esta frase a que para cada problema existe una aplicación.

Las primeras aplicaciones aparecieron a finales de los años 90, estas estaban pre cargadas en los dispositivos móviles y eran muy básicas como agenda, contactos, ringtones, y en algunos casos email. La evolución comienza con la llegada de la tecnología EDGE y la posibilidad de conexión a internet, permitiendo de esta forma un mayor desarrollo de aplicaciones. Pero aún existía otra barrera que frenaba el desarrollo, se trataba de las restricciones de los fabricantes que hacían sus propios sistemas operativos y que no permitían desarrolladores externos. En esta época se prestaba más atención al hardware, y la evolución de la industria móvil era desordenada y sin rumbo. Pero en 2007 con el lanzamiento del iPhone todo cambia, Apple plantea una nueva estrategia, ofreciendo su dispositivo como una plataforma donde correr aplicaciones diseñadas por desarrolladores externos. Estas aplicaciones se ofrecían a través del “App Store”, inicialmente contaba con 500 aplicaciones y a finales de 2008 existía prácticamente una aplicación para todo. En septiembre de 2008 se lanzaba la primera versión de Android que copiando la estrategia de Apple, incluía una tienda de aplicaciones “Android market” con 50 aplicaciones inicialmente. El número de aplicaciones ha ido aumentando desde entonces, Android debido a que es una plataforma de código abierto, actualmente supera en número de aplicaciones a IOS [6].

En la siguiente gráfica podemos observar el crecimiento en número de aplicaciones ofrecidas en el App Store y Google Play (Actual Android Market) entre 2010 y 2014:

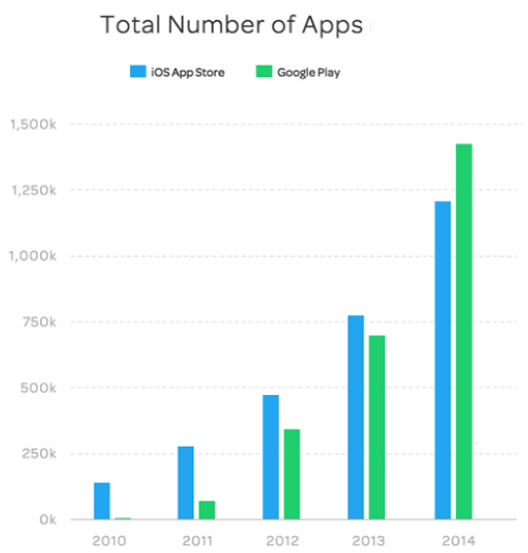

Imagen 1: Comparación del número de aplicaciones ofrecidas en App Store y Google Play [7]

Por otro lado, cabe destacar que el uso de las aplicaciones móviles experimento un crecimiento del 76% el año pasado con respecto al anterior [8]. En la siguiente gráfica podemos ver los resultados del crecimiento en el uso de aplicaciones móviles por categorías:

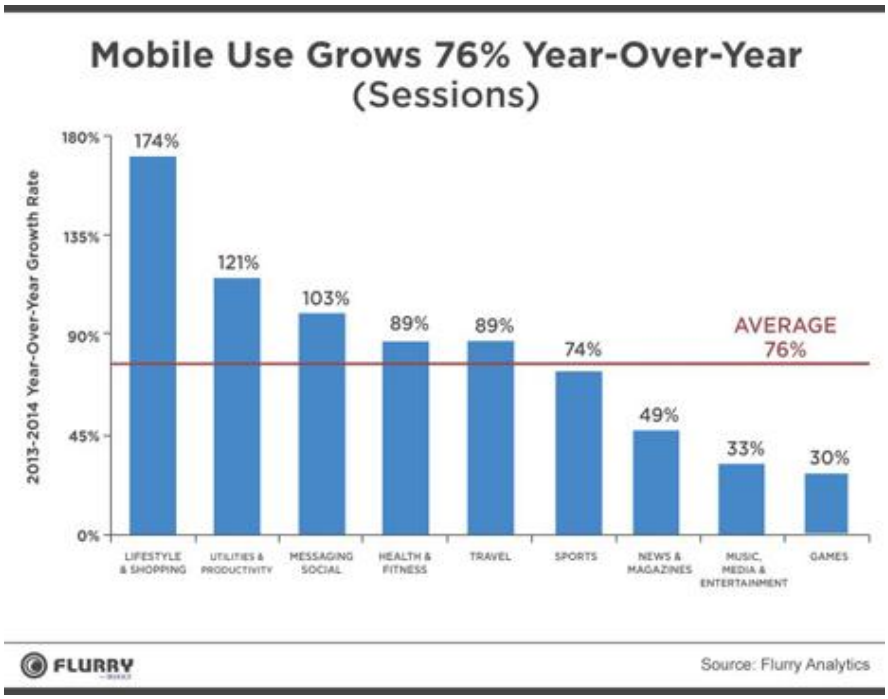

Imagen 2: Crecimiento en el uso de aplicaciones de 2013 a 2014 [8]

2.2 Tipos de aplicaciones

Una vez realizada una breve introducción a las aplicaciones, nos centraremos en analizar los tipos. Existen varias formas de clasificar las aplicaciones de acuerdo a diferentes factores. Algunas de estas clasificaciones son:

- Mercado para el que han sido desarrolladas:
  Se diferencian las aplicaciones atendiendo al tipo de dispositivo móvil para el que han sido diseñadas: teléfonos móviles, tablets, relojes inteligentes, televisiones inteligentes, todos los dispositivos…
- Lenguaje de programación:
  Se diferencian las aplicaciones atendiendo al lenguaje en el que han sido desarrolladas: Java, Objetive C, Bada, WebOS, C#, C++, HTML5, HTML/CSS/JavaScript…
- Tienda de aplicaciones:
  Se diferencian las aplicaciones dependiendo de la tienda de aplicaciones en las que se ofrezcan: Google Play, App Store, Windows Phone Store, BlackBerry World… [9].
- Coste
  Se diferencia entre aplicaciones gratuitas y aplicaciones con coste [10].
- Tipo de desarrollo
  Se diferencian atendiendo a la forma de desarrollar la aplicación, existen tres tipos: aplicaciones nativas, aplicaciones Web y aplicaciones híbridas

Esta última es la forma más habitual de clasificación, sobre todo desde el punto de vista del desarrollador, por ello será en la que nos centraremos.

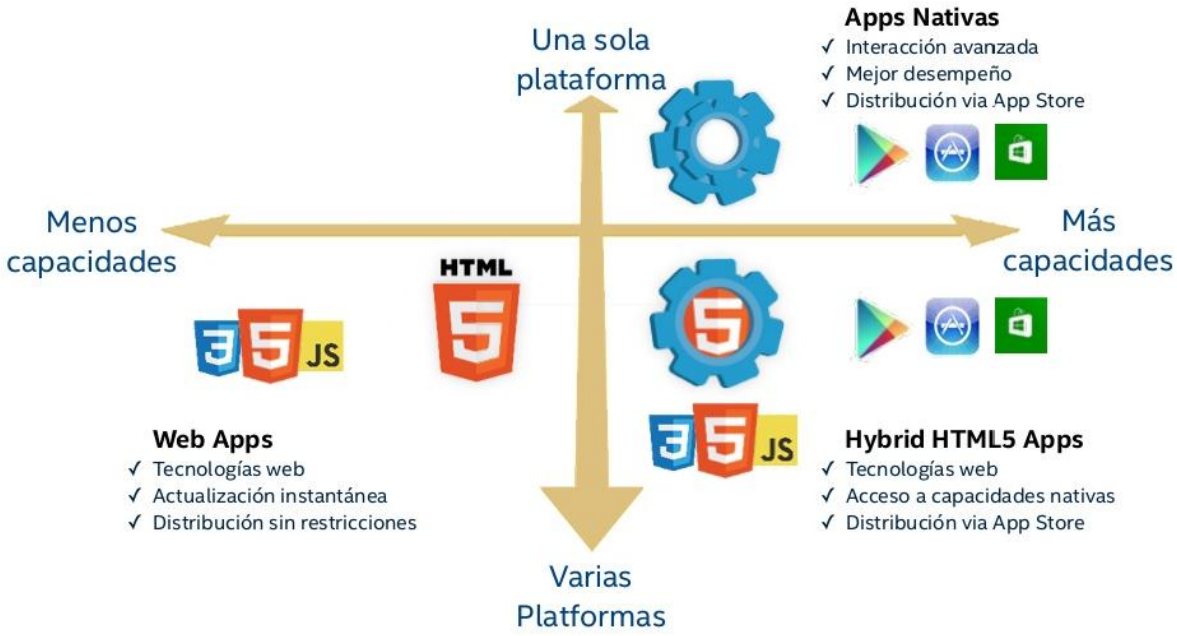

Imagen 3: Ventajas de cada tipo de aplicación en cuanto al tipo de desarrollo [11]

2.2.1 Aplicaciones nativas

Las aplicaciones nativas son aquellas que se implementan empleando el software específico que ofrece cada sistema operativo a los desarrolladores, este software recibe el nombre genérico de SDK (Software Development Kit) [12]. Las aplicaciones se diseñan y programan específicamente para cada plataforma, en el lenguaje utilizado por el SDK. De esta forma Android, iOS, Windows Phone, etc. tienen un SDK diferente, por lo que si queremos que la aplicación esté disponible en todas las plataformas, se deberán crear varias apps con el lenguaje del sistema operativo seleccionado. Por ejemplo las aplicaciones para iOS se desarrollan en lenguaje Objetive-C, para Android en Java y para Windows Phone en .Net [13].

El código fuente se compila a un ejecutable de forma similar a las tradicionales aplicaciones de escritorio. Todos los recursos (imágenes, iconos, etc.) que la aplicación utiliza en su ejecución quedan en el archivo compilado. De esta forma el archivo ya está listo para ser subido y distribuido a las tiendas de aplicaciones específicas del sistema operativo para el que estemos programando (App Store, Google Play…). Una vez que el ejecutable se ha subido, estas tiendas de aplicaciones tienen un proceso de auditoría, para evaluar si se adapta a los requerimientos del sistema.

Por tanto este tipo de aplicaciones se descargan e instalan a través de las tiendas de aplicaciones, permitiendo aplicar diferentes técnicas de promoción y marketing para potenciar su descarga. Destacar también, que este tipo de aplicaciones se actualizan frecuentemente para corregir errores o añadir mejoras, para ello el usuario tiene que volver a descargarlas para obtener la última versión.

La principal característica de este tipo de aplicaciones es que permiten acceder al Hardware del dispositivo móvil como la cámara, GPS, acelerómetro, dispositivos de almacenamiento… haciendo que la experiencia del usuario sea más positiva que con otro tipo de apps. Además este tipo de aplicaciones no necesitan conexión a internet para funcionar y suelen ser las más rápidas de los tres tipos. Otra característica es que pueden hacer uso de las notificaciones del sistema operativo para mostrar avisos importantes al usuario, aun cuando no se está usando la aplicación, como por ejemplo los mensajes de WhatsApp.

A nivel de diseño las aplicaciones nativas tienen una interfaz que se basa en las guías de cada sistema operativo, obteniendo una mayor coherencia con propio sistema operativo y el resto de aplicaciones. De esta forma se favorece la usabilidad, beneficiando de forma directa al usuario que se encuentra con interfaces familiares.

Podemos determinar por tanto, que si el coste no es una barrera a la hora de diseñar una aplicación la mejor opción es siempre la implementación de una aplicación nativa para cada plataforma (iOS, Android, Windows pone…), pero en caso contrario, lo mejor es decantarse por una de las otras opciones

2.2.2 Aplicaciones Web

Las aplicaciones Web, también conocidas como Webapps, se implementan en lenguajes de programación muy conocidos como HTML, Javascript y CSS empleando un framework de desarrollo de aplicaciones como por ejemplo jquery mobile, sencha, Kendo UI, etc. En este caso no se emplea un SDK, por lo que la programación es totalmente independiente al sistema operativo. Por tanto este tipo de aplicaciones se pueden usar en diferentes plataformas sin necesidad de desarrollar un código diferente para cada caso en particular.

Las Webapps se ejecutan, sin necesidad de instalación, dentro de propio navegador web del dispositivo a través de una URL, por ejemplo Safari si se trata de una plataforma iOS [14]. Por este motivo es necesario tener conexión a internet para que funcionen correctamente. Otra consecuencia es que no necesitan que el usuario reciba actualizaciones, ya que siempre va a estar viendo la última versión. Destacar que puesto que no necesitan de instalación, no se distribuyen en las tiendas de aplicaciones, sino que se comercializan y promocionan de forma diferente.

Debido a la independencia con el sistema operativo, no se pueden aprovechar al máximo las posibilidades de los diferentes componentes Hardware del teléfono. A nivel de diseño, las aplicaciones Web tienen una interfaz genérica e independiente del sistema operativo, por lo que el usuario se encuentra con interfaces que en muchos casos difieren de las del propio sistema operativo y del resto de aplicaciones.

Este tipo de aplicaciones son la mejor opción si nuestro objetivo es adaptar la web a formato móvil.

2.2.3 Aplicaciones híbridas

Como su propio nombre indica, este tipo de aplicaciones surge de una combinación de las dos anteriores, se podría decir que toma lo mejor de cada una de ellas [15]. Se desarrollan al igual que las aplicaciones Web usando HTML, CSS y javascript, lo que permite mantener el carácter multiplataforma, y una vez finalizadas, se compila o empaqueta usando un framework, de tal forma que el resultado final es un ejecutable como el que se genera en una aplicación nativa.

Es decir, prácticamente a partir de un mismo código se pueden obtener diferentes versiones para cada plataforma. Puesto que el resultado final es un instalador como en el caso de las aplicaciones nativas, este tipo de aplicaciones se distribuyen y comercializan a través de las tiendas de aplicaciones.

A diferencia de las aplicaciones web, las aplicaciones hibridas pueden acceder a la mayor parte de los recursos propios del dispositivo, como el GPS, la cámara, el acelerómetro, etc. empleando ciertas librerías [16].

Este tipo de aplicaciones también tienen un diseño visual que generalmente no se asimila al del sistema operativo, aunque en este caso, hay formas de utilizar botones y controles nativos en cada plataforma, obteniendo una estética similar a la del resto de aplicaciones y el sistema operativo.

2.2.4 Ventajas e inconvenientes de cada tipo

Una vez explicado cada tipo de aplicación. Vamos a realizar una comparativa entre ellas, para ello se ha elaborado la siguiente tabla, donde se muestran ventajas e inconvenientes: [17]

DESCRIPCIÓN	APLICACIONES NATIVAS	APLICACIONES WEB	APLICACIONES HÍBRIDAS
Acceso a los recursos Hardware del dispositivo	Acceso Completo	Acceso muy limitado	Acceso a la mayor parte
Conexión a Internet	No es necesaria	Requiere conexión	No es necesaria
Código reutilizable	No reutilizable entre plataformas	El mismo código reutilizable en múltiples plataformas	Mismo código base para múltiples plataformas
Coste desarrollo	Elevado (Requiere diferentes lenguajes/herramientas para cada plataforma	Bajo. Proceso de desarrollo sencillo y económico	Razonable con respecto a las aplicaciones nativas
Publicación en tiendas de aplicaciones	Posible	No es posible (requiere por tanto mayor esfuerzo de promoción)	Posible
Actualizaciones	Actualización constante	El usuario siempre dispone de la última versión	Actualización constante
Experiencia del usuario	La más completa	Menor que en una app nativa	Más propia de una app web que de una nativa

Ventaja

Inconveniente

Imagen 4: Tabla comparativa de los diferentes tipos de aplicaciones en cuanto a su desarrollo

2.3 Dispositivos móviles y sistemas operativos

Una vez realizada una breve introducción a las aplicaciones, es necesario conocer los diferentes dispositivos móviles que hacen uso de ellas, así como los diferentes sistemas operativos existentes en la actualidad.

- ***Dispositivos móviles***

Podemos definir dispositivo móvil como un aparato de tamaño pequeño, con ciertas capacidades de procesamiento, con conexión permanente/intermitente a la red, con memoria limitada, que ha sido diseñado específicamente para una función, pero que puede llevar a cabo otras más genérales [18]. Hoy en día existe una gran multitud de dispositivos móviles, pero el desarrollo de aplicaciones se centra en la telefonía móvil (Smartphone), tabletas (o también llamadas tablets), ordenadores (PC, Netbooks…) y recientemente en dispositivos como los relojes inteligentes (Smartwach), o la televisión inteligente (SmartTV)

Imagen 5: Dispositivos móviles que permiten la instalación de aplicaciones

En este proyecto nos centraremos únicamente en los teléfonos móviles y las tabletas, al ser los dispositivos que normalmente llevan asociado Hardware NFC y para los cuales MobiWallet resulta ser un servicio realmente funcional.

- ***Sistemas operativos***

Un sistema operativo es un software que se inicia al encender el dispositivo y que encarga de gestionar todos los recursos del sistema, permitiendo así la comunicación entre usuario y dispositivo. Los sistemas operativos de los dispositivos móviles son más sencillos que los de los PC y están más orientados a la conectividad inalámbrica. Son los fabricantes los encargados de elegir el sistema operativo se instalará en sus dispositivos [19].

Consultando la página de estadísticas “www.netmarketshare.com” podemos ver cuáles son los sistemas operativos más usados en los dispositivos móviles entre enero y septiembre de 2015:

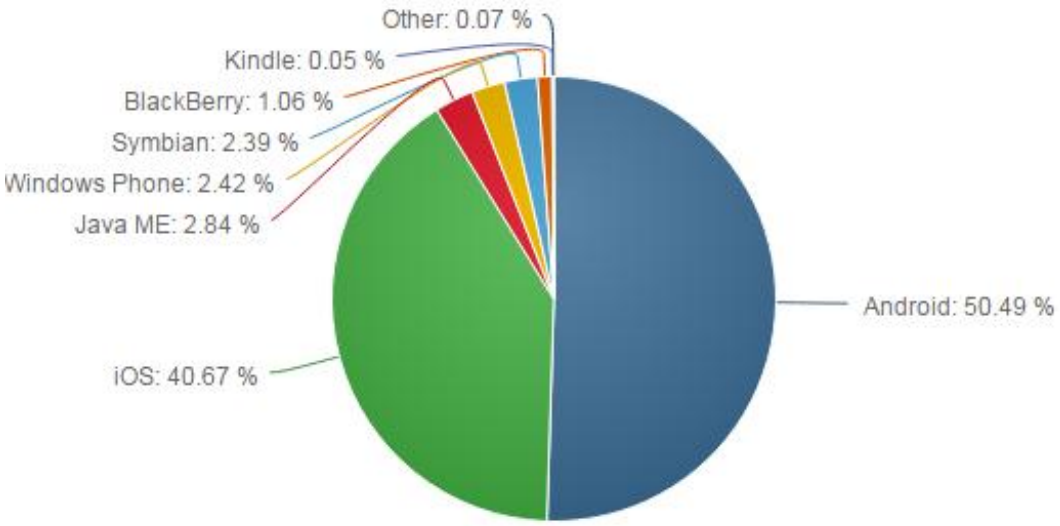

Imagen 6: Comparación de los sistemas operativos más usados en dispositivos móviles en 2015 [20]

Como se puede apreciar los sistemas operativos iOS y Android son los que predominan. Con un porcentaje mucho menor aparece Windows Phone, Symbian y BlackBerry. Destacar que Java ME no se menciona porque no es un sistema operativo propiamente dicho sino más bien una plataforma con la que se ejecutan aplicaciones.

2.4 Seguridad

Otro aspecto fundamental a la hora de desarrollar una aplicación, es la seguridad. Dentro de este capítulo se analizarán riegos y vulnerabilidades en las aplicaciones, así como los ataques que pueden sufrir. También se estudiaran las diferentes capas de seguridad, y los modelos y metodologías a seguir para desarrollar una aplicación segura.

2.4.1 Concepto de seguridad

En primer lugar se introducirá el concepto de seguridad. Se define seguridad, como la propiedad de algo en donde no se registran peligros, daños ni riesgos. En el área de las tecnologías y la información se puede definir la seguridad informática, como la disciplina que se encarga de minimizar los riesgos y vulnerabilidades, y dar respuestas eficientes ante los ataques contra los sistemas de comunicación y almacenamiento de información. Es decir se trata de proteger la integridad y la privacidad de la información almacenada o transmitida por un sistema informático.

2.4.2 Modelos y metodologías

Actualmente existen modelos, metodologías, estándares y certificaciones que se pueden utilizar para desarrollar una aplicación móvil de forma segura. A continuación se describen algunos de ellos:

- **OSSTMM** (*Open Source Security Testing Methology Manual o Manual de la Metodología Abierta de Testeo de Seguridad*). Se trata de un manual realizado por ISECOM (Institute for Security and Open Methodologies), que reúne las diversas pruebas y métricas de seguridad, utilizadas por los profesionales durante las Auditorías de Seguridad [21]. Se ha logrado gracias a un consenso entre más de 150 expertos de seguridad de todo el mundo y se encuentra en constante evolución. Se centra en los detalles técnicos de los elementos que deben ser probados. Se compone de 6 fases: Seguridad de la Información, Seguridad de los Procesos, Seguridad en las tecnologías de Internet, Seguridad en las Comunicaciones, Seguridad Inalámbrica y Seguridad Física [22].
- **OWASP** (*Open Web Application Security Project o Proyecto abierto de seguridad de aplicaciones web*). Es un organismo sin ánimo de lucro formado por empresas, organizaciones educativas y particulares, que proporciona guías de pruebas, manuales de referencias y metodologías para el análisis, revisión y evaluación de la seguridad en las aplicaciones web. Últimamente también se centra en la seguridad de las aplicaciones móviles. Los documentos con más éxito son: el top 10 de las vulnerabilidades, la guía de desarrollo, la guía de revisión del código y la guía de pruebas [23].
- **ISSAF** (*Information System Security Assessment Framework o Marco de Evaluación de Seguridad de Sistemas de Información*) es una metodología estructurada de análisis de seguridad cuyo objetivo es proporcionar procedimientos para el testing de sistemas de información. La metodología de test de penetración ISSAF está diseñada para evaluar su Red de trabajo, sistema y control de aplicaciones .Esta enfocada en tres fases: Planificación y Preparación, Evaluación, y por último Reportes, Limpieza y Destrucción de Objetos [24]

2.4.3 Vulnerabilidades y riesgos

Aunque estos estándares y metodologías sean útiles a la hora de desarrollar aplicaciones móviles, lo cierto es que es muy difícil conseguir aplicaciones 100% seguras, siempre existe la posibilidad de que aparezcan vulnerabilidades y riesgos.

Una ***vulnerabilidad*** es el fallo o mal funcionamiento de una aplicación o sistema operativo debido a errores de análisis, diseño y construcción, de tal forma que se expone la información del usuario a los posibles atacantes. Las vulnerabilidades pueden ser involuntarias o inducidas por los desarrolladores de software. Ejemplos de vulnerabilidades son: cierres inesperados, fugas de información sensible, almacenamiento y transmisión de datos no seguras, contraseñas violadas, etc.

Por otro lado podemos definir ***riesgo*** como la posibilidad de perder un recurso valioso, por ser vulnerado, robado y atacado por un usuario no autorizado. En general el riesgo en la ausencia de seguridad. En los dispositivos móviles el riesgo deriva de su principal virtud, la movilidad, dado que es difícil definir cuando el dispositivo se encuentra en un espacio seguro. El riego se presenta en tres niveles:

- A nivel de usuario: con el robo o la pérdida del dispositivo, el empleo de contraseñas frágiles, el almacenamiento de datos sensibles, el acceso al dispositivo de usuarios no autorizado, la descarga y utilización de aplicaciones no confiables.
- A nivel de proceso: utilización de dispositivos que no disponen de seguridad básica, intercambio de datos sensibles, perdida de información sensible, uso de servicios sin seguridad mínima
- A nivel tecnológico: defectos o fallos en sistemas operativos, plataformas, aplicaciones, navegadores de los dispositivos móviles, así como en los mecanismos de seguridad implementados [25].

A continuación se muestran el top 10 de riesgos identificados por OWASP en las aplicaciones, así como los controles y pautas a seguir durante el diseño para tratar de mitigarlos:

Imagen 7: Top 10 de riesgos identificados por OWASP y pautas para evitarlos [26]

2.4.4 Modelo de seguridad por capas

Una de las formas de afrontar el desarrollo de la seguridad de una aplicación tratando de evitar riesgos y vulnerabilidades, es seguir el modelo de capas, que propone realizar seguridad a diferentes niveles. Estos niveles son los siguientes:

Imagen 8: Niveles del modelo de seguridad por capas [27]

- **Nivel de Hardware**: En este nivel conviene realizar un inventario detallado de las características de la memoria, procesador, antenas y otros elementos del dispositivo de los que se va a hacer uso, para poder definir riesgos y vulnerabilidades.
  Destacar que el dispositivo móvil puede ser perdido o robado, convirtiéndose en el principal riesgo, ya que supone una vulnerabilidad muy grande, dado que se puede consultar la información accediendo al dispositivo, o extraerse de elementos de almacenamiento externos como tarjetas SD [27].

- **Nivel de sistema operativo:** Los principales riesgos y vulnerabilidades de este nivel son causados por errores en el aislamiento de los recursos, ya sea debido a sus diseños, a errores en el software o a una mala configuración de sus servicios.
  - *Principales riesgos:*
    - Virus y código malicioso
    - Ataques de denegación de servicio por saturación del sistema.
  - *Posibles contramedidas:*
    - Emplear sistemas de protección local como antivirus o firewalls.
    - Habilitar el uso de contraseñas para restringir el acceso al sistema o utilizar software de terceros para habilitar esta funcionalidad
    - Deshabilitar las acciones automáticas, como el redireccionamiento de mensajes.

- **Nivel de almacenamiento:** Dentro de este nivel, se debe tener en cuenta toda la información almacenada en el dispositivo móvil. Encontramos gran variedad de datos: archivos binarios almacenados en la memoria tanto interna como externa, archivos de usuario, archivos de bases de datos, etc.
  - *Principales riesgos:*
    - Acceso no autorizado a información confidencial
    - Modificación de datos de manera ilícita.
    - Obtención de credenciales de acceso a ciertos recursos.

- *Posibles contramedidas:*
  - Proteger el acceso al dispositivo solicitando contraseña
  - Emplear software que permita el cifrado de la memoria
  - Utilizar criptografía que permita cifrar la información almacenada
  - Efectuar controles de integridad de los datos mediante funciones de hashing

➢ **Nivel de comunicación:** En este nivel se debe tener en cuenta todos los procesos de comunicación del dispositivo móvil, como son las llamadas de voz, y el intercambio de datos mediante bluetooth, Wifi, NFC, etc.

- *Principales riesgos:*
  - Ataques de denegación de servicio a partir de inundación de paquetes, o mediante la degradación de la señal introduciendo interferencias
  - Intercepción del tráfico para monitorizarlo (Sniffing)
  - Suplantación de identidad para obtener recursos del sistema (masquearding)
  - Introducción de virus en redes corporativas
  - Ataques sobre redes TCP/IP como Spoofing, Man in the middle, sesión hijacking, etc.
- *Posibles contramedidas:*
  - Deshabilitar servicios innecesarios
  - Desactivar las diferentes opciones de conectividad del dispositivo cuando no se estén usando
  - Usar los mecanismos de cifrado más robustos para cada tecnología.
  - Emplear llaves en las comunicaciones, y efectuar su intercambio de forma segura.

➢ **Nivel de aplicación:** en el nivel de aplicación es muy importante revisar las medidas de seguridad que se han tomado para que las aplicaciones no puedan desestabilizar el sistema. En este nivel las vulnerabilidades son críticas, ya que pueden ser usadas directamente por las aplicaciones.

- *Principales riesgos:*
  - Denegación de servicio local por invocación indebida
  - Denegación de servicio sobre servidores remotos (aplicaciones WAP)
  - Descarga de aplicaciones que contengan código malicioso.
- *Posibles contramedidas:*
  - Proteger los recursos remotos que soportan la aplicación móvil frente a posibles ataques de denegación de servicio.
  - Descargar únicamente aplicaciones móviles desarrolladas por terceros, que se encuentren en las tiendas de aplicaciones (si es posible firmadas de forma digital)
  - Emplear las prácticas de programación seguras recomendadas. (ej. OWASP) [28]

2.4.5 Criptografía y Encriptación

Una vez analizados los riesgos y amenazas existentes en los distintos niveles de seguridad de las aplicaciones móviles, podemos determinar que en todos los niveles los datos y la información son el principal activo. De acuerdo a las buenas prácticas para el desarrollo de aplicaciones seguras, definidas por organizaciones como OWASP, es recomendable emplear técnicas de cifrado y encriptación de datos, así como implementar protocolos de comunicaciones seguras.

Los servicios básicos de criptografía para un dispositivo móvil son los siguientes:

- **Autenticación**: Es necesario asegurar y verificar la autenticidad de una comunicación entre el origen y el destino.
- **Confidencialidad**: Los datos transmitidos deben de protegerse frente a ataques mientras se está realizando un envió de información, de tal forma que se asegure que el destino recibirá la información desde el origen sin que terceros hayan podido acceder a la información enviada.
- **Integridad**: La información recibida en el destino debe ser la misma que la enviada por la fuente, sin que se produzcan modificaciones en la manipulación y envió de la misma.
- **Control de Acceso**: Verificar que el usuario que trata de acceder al dispositivo es identificado correctamente evitando una suplantación o un acceso no permitido.
- **Criptografía y Cifrado:** Protocolos, algoritmos y modelos matemáticos para transformar los datos de tal forma que no puedan ser leídos normalmente y que proporcionen seguridad a la información almacenada [29].

A la hora de desarrollar aplicaciones hay que tener en cuenta que no es necesario cifrar todos los datos, sino únicamente aquella información que pudiera ser sensible. Además, en el caso de que la aplicación vaya a realizar constantes intercambios de datos, es muy recomendable utilizar protocolos criptográficos como el SSL (Secure Sockets Layer) que no solamente cifra el flujo de datos, sino que también asegura la integridad del mensaje [30].

2.5 Conectividad.

Los dispositivos móviles generalmente disponen de múltiples tecnologías de comunicación inalámbrica, de tal forma que la información es transferida a través de diferentes métodos por el aire. Es común que las aplicaciones hagan uso del intercambio de datos, por lo que es necesario conocer las tecnologías que ofrecen los dispositivos móviles actuales. Vamos a estudiar las siguientes:

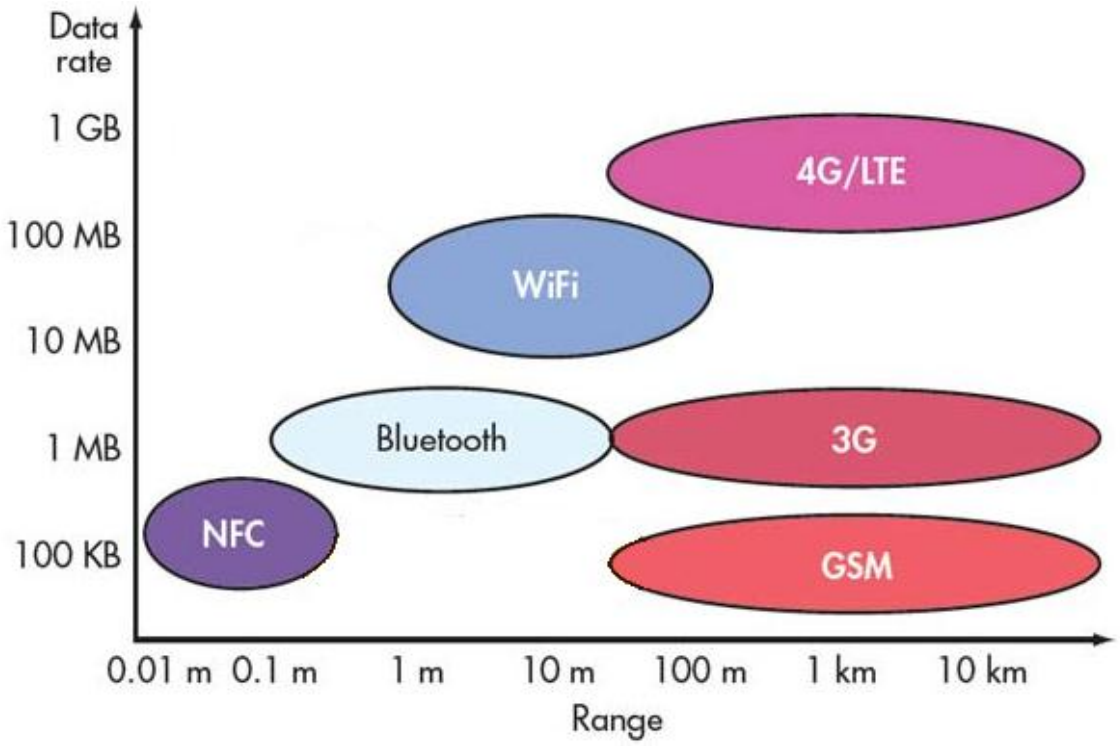

Imagen 9: Comparativa en cuanto alcance y velocidad de transmisión de las diferentes tecnologías de conectividad

- **Bluetooth (estándar IEEE 802.15.1)** Tecnología inalámbrica de corto alcance que opera a 2.4 GHz (rango no licenciado) que permite establecer comunicaciones de datos entre dispositivos móviles, ordenadores y periféricos, reemplazando el uso de cables serie, paralelo o USB. Puede alcanzar velocidades de transmisión de datos de hasta 24 Mbps, y lo normal es que su radio de acción sea de unos 10 metros [31].
  Aparte del intercambio de datos entre de dispositivos móviles, que está en decadencia, esta tecnología es ideal para la conexión de auriculares, teclados ratones y otros periféricos inalámbricos, que no necesitan transmitir una cantidad enorme de datos y además suelen estar cerca de los dispositivos móviles [32].

- **Wi-Fi (estándar IEEE 802.11)** mecanismo de comunicación principal de los dispositivos electrónicos actuales, que permite el intercambio de datos y el acceso a redes como Internet. Opera en rangos de frecuencia no licenciados, concretamente, 2,400-2,500 GHz (802.11b/g/n) o 4,915-5,825 GHz (802.11a/n/ac) y puede llegar a alcanzar una velocidad máxima de 1.3 Gbps.
  Las comunicaciones Wifi pueden establecerse de dos modos:
  - En *modo infraestructura*: los clientes se conectan a un punto de acceso
  - En *modo ad-hoc*: los clientes se conectan entre sí sin ningún punto de acceso [33].

- **Tecnologías de telefonía móvil (1G, 2G (GSM), 3G (UMTS), 4G (LTE)):** este tipo de tecnologías fueron concebidas inicialmente para la transmisión de voz, pero con el paso del tiempo han ido evolucionando permitiendo el intercambio de datos.
  - La *primera generación* se caracterizó por ser analógica y estrictamente de voz [34]. Estaba basada en un conjunto de celdas o células interconectadas, que daban servicio a los dispositivos que se encontraban dentro de su zona de cobertura. No todas las redes se basaban en los mismos protocolos, sino que dependían de sus fabricantes, por lo que no era fácil interconectarlas ni utilizar los mismos terminales en distintas redes. En la actualidad está en desuso y pronto desaparecerá definitivamente
  - La *segunda generación (GSM)* ya era digital y tenía como principales objetivos la interconexión de las redes y la posibilidad de conectarse a ellas con un mismo terminal, apareciendo el primer concepto de roaming. También trajo otras ventajas como una mejor calidad de voz, capacidad para transmitir datos a velocidades muy bajas, transmisión de faxes y los famosos SMS.
  - La *tercera generación (UMTS)* es una evolución de la anterior y mantiene uno de sus principios básicos: un estándar sobre el que continuar los desarrollos. Se mejora la potencia de las antenas, permitiendo más conexiones, mayor calidad de voz y mayor velocidad para transferir datos, alcanzándose hasta 2 Mbps [35]. Los servicios que ofrecen las tecnologías 3G son básicamente: acceso a Internet, servicios de banda ancha, roaming internacional e inter-operatividad. Gracias a ella surgen nuevos servicios y aplicaciones como videoconferencias o el comercio electrónico.
  - La *cuarta generación (LTE)* estará basada totalmente en tecnología IP, siendo un sistema que engloba a otros sistemas y una red de redes que se alcanza gracias a la convergencia entre las redes de cables y las inalámbricas [36]. Se alcanzaran velocidad de acceso de 100 Mbps en movimiento y 5 Gbps en reposo, manteniendo una calidad de servicio (QoS) de punta a punta de alta seguridad para permitir ofrecer servicios de cualquier tipo, en cualquier momento y en cualquier lugar. Aunque su uso ya está disponible en determinados países del mundo, no se espera su implantación global y definitiva hasta el año 2020.

- **NFC** (*Near Field Communication o Comunicación de campo cercano*) tecnología inalámbrica de corto alcance y alta frecuencia que permite el intercambio de datos entre dos dispositivos que se encuentran muy próximos. Concretamente opera a 13,56 MHz (rango no licenciado) con un ancho de banda que varía entre 106-424 Kbps y el alcance es de unos 10 cm [37]. La tecnología NFC puede funcionar en dos modos:
  - *Activo*, en el que ambos equipos con chip NFC generan un campo electromagnético e intercambian datos.
  - *Pasivo*, en el que solo hay un dispositivo activo y el otro aprovecha ese campo para intercambiar la información [38].

  Esta tecnología no existe en todos los dispositivos actuales, pero cada vez es más común, y puede recibir diferentes usos:

  - *Identificación/control*: acceder a lugares donde es precisa una identificación o un control de acceso, se puede hacer simplemente acercando el dispositivo móvil o tarjeta con chip NFC a un dispositivo de lectura.

- *Transferencia fotos/videos o música*: Aunque existen otras alternativas, esta tecnología también permite la transferencia de archivos multimedia de forma rápida.
- *Recogida/intercambio de datos*: Google es el principal protagonista de este uso, pues en combinación con otras tecnologías, se pueden obtener utilidades como marcar dónde estamos, recibir información de un evento o establecimiento.
- *Automatizar perfiles*: Es posible crear tareas q y escribirlas en etiquetas NFC, de tal forma, que al escanear estas etiquetas con el dispositivo móvil, se ejecuten dichas tareas.
- *Pago con el dispositivo móvil*: esta es la función estrella de los usos del NFC. Esta tecnología está camino de convertirse en el método de pago del futuro debido a su comodidad. El único requisito necesario es tener asociado el gasto a una cuenta bancaria [39].

En la siguiente imagen se muestran alguno de los posibles usos de esta tecnología.

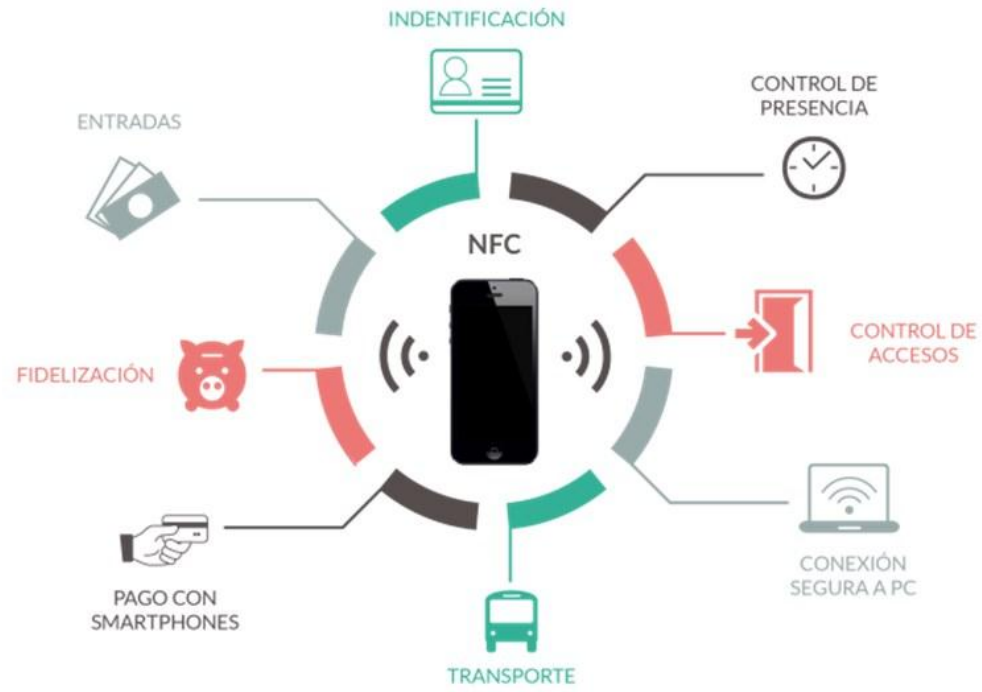

Imagen 10: Principales usos de la tecnología NFC [40]

3 ASPECTOS PRÁCTICOS

Este capítulo se divide en dos grandes apartados bien diferenciados. En el primero de ellos se presenta el proyecto MobiWallet y su escenario actual, mientras que en el segundo se describen los requerimientos que debe cumplir la aplicación a implementar

3.1 El proyecto MobiWallet

Dado que la aplicación a desarrollar se introducirá en la plataforma MobiWallet es necesario, conocer en primer lugar la motivación del proyecto MobiWallet y sus objetivos, y a continuación la situación actual de esta plataforma.

3.1.1 Definición

MobiWallet es un proyecto financiado por la Comisión Europea dentro del Programa CIP (Competitiveness and Innovation Framework Programme, as part of the Information and Communications Technology – Policy Support Programme, Call 7).

Tiene como objetivo implementar una visión distinta del transporte, donde la interoperabilidad no sea un problema a la hora pagar, y se pueda ofrecer un sistema de gestión de tarifas dotado de inteligencia. Para ello se combinaran estas tecnologías junto con un servicio de pago móvil y funcionalidades de viaje como un servicio de planificación de rutas. Los pilotos desplegados en Europa tratarán de demostrar los beneficios de una plataforma unificada capaz de procesar sin problemas los esquemas de pago de una gran variedad de operadores de transporte desde un único punto de venta: el dispositivo móvil del usuario [41].

Este proyecto apuesta por introducir soluciones interoperables de gestión de tarifas (componente esencial dentro de las ciudades y transportes inteligentes) a través de 4 áreas de impacto:

- ✓ Fomento del cambio de modo y facilidad de uso de múltiples opciones de transporte centradas en personas discapacitadas o de movilidad reducida.
- ✓ Reducir el consumo de energía mejorando la eficacia energética
- ✓ Movilidad mejorada y sostenible para todos los usuarios.
- ✓ Mejora de las capacidades de transporte entre países de la Unión Europea.

Los socios que forman MobiWallet incluirán la participación de cientos de usuarios en cuatro ciudades piloto en Europa (Santander(España), Toscana (Italia), West Midlands (Reino Unido), Novi Sad (Serbia) y se obtendrán y analizarán sus críticas y opiniones respecto al servicio para garantizar que la implementación de las soluciones tecnológicas dan respuesta a las necesidades de los ciudadanos y que las soluciones aportadas logran el máximo impacto en cuanto a facilitar y acercar el camino a los sistemas de transporte del futuro.

3.1.2 Objetivos

El proyecto MobiWallet trata de conseguir los siguientes objetivos:

- ➢ ***Obtener un sistema unificado e inteligente para el uso del transporte público.***
  Pretende alcanzar un sistema cuya principal característica sea la interoperabilidad entre los diferentes operadores de transporte de las ciudades, demostrando sus beneficios en los diferentes pilotos desplegados en Europa.
  Aunque ya existen soluciones al problema de la interoperabilidad, se desarrollaran mejoras como los servicios de pago ágiles basados en los pagos electrónicos con

smartphones, que unido a otros servicios como planificadores de rutas inteligentes, provean a los usuarios de un sistema útil para moverse por las ciudades.

- ***Cambiar el modo de viajar, facilitando el uso del transporte público***.
  Se centrará en varias áreas de operación que permitirán la explotación de aspectos innovadores de las tecnologías disponibles. Se integrarán servicios de información, servicios de pago y software de planificación de rutas con cálculo de tiempos, para dar facilidades y toda la información posible al usuario.
- ***Contribuir al medio ambiente.***
  Se pretende fomentar el uso del transporte púbico consiguiendo con ello de forma indirecta una reducción del consumo de energía, así como disminuir las emisiones de CO2 en las ciudades.
- **Llegar a todo el público posible**
  Tratará de simplificar y facilitar el uso del sistema a personas de otros países que viajen a cualquier ciudad europea, y ayudar a dar mayor disponibilidad del transporte público a personas de movilidad reducida.
- ***Contribución a los estándares.***
  MobiWallet se centrará en ampliar los estándares:
  - ISO 24014 (Public transport -- Interoperable fare management system)
  - EN 15320 (Identification card systems - Surface transport applications - Interoperable Public Transport Applications - Framework)

  De esta forma se pretende no sólo establecer el camino a seguir para un sistema de viaje común en la UE, sino también introducir ideas para facilitar la entrada en el mercado de más operadores de transporte y más empresas interesadas [42].

3.1.3 Piloto de Santander MobiWallet

El principal objetivo del piloto de Santander (único piloto de España) es ofrecer un campo de pruebas para validar con éxito los objetivos del proyecto MobiWallet. Este piloto se puede diferenciar por un conjunto de objetivos que lo distinguen de los pilotos restantes. Los dos objetivos principales son:

- ✓ Proporcionar una solución de gestión de tarifas interoperable que pueda cubrir toda la ciudad, incluyendo 5 modos diferentes de transporte.
- ✓ Explotar las sinergias entre los sistemas de pago contacless y los dispositivos móviles.

El piloto español se llevará a cabo en su mayoría en la zona urbana de la ciudad, cubriendo la mayor parte de transportes disponibles en esa área. También está incluidas zonas periféricas de la ciudad, particularmente aquellas cubiertas por las Pedreñeras, un pequeño ferry, que permite viajar a diferentes puntos de la bahía de la ciudad incluyendo el punto opuesto a la ciudad de Santander, que es el Puntal de Somo.

Aparte del ferry, en este piloto también se incluyen bicicletas públicas, autobuses, taxis y vehículos privados (a través de la incorporación en la solución del pago del parking de la ciudad) De esta forma el piloto Santander será referencia en la gestión de la interoperabilidad entre los diferentes operadores que gestionan los transportes comentados.

El piloto español incluye a Indra como director y principal proveedor tecnológico, el Ayuntamiento de Santander como proveedor de usuarios finales, la Universidad de Cantabria como proveedor de tecnología de innovación y soporte al usuario final, y TST Sistemas también como proveedor tecnológico[43].

3.1.4 Escenario de aplicación

Tras introducir ciertos aspectos teóricos de MobiWallet, que se han tenido en cuenta a la hora de realizar de este proyecto, en este apartado se presentará la situación del piloto en Santander sobre el cual se va a desarrollar la aplicación. Concretamente la plataforma MobiWallet está compuesta por 5 grandes bloques bien diferenciados: MobiWallet Web Service, MobiWallet Core, MobiWallet App, TPV Virtual y operadores.

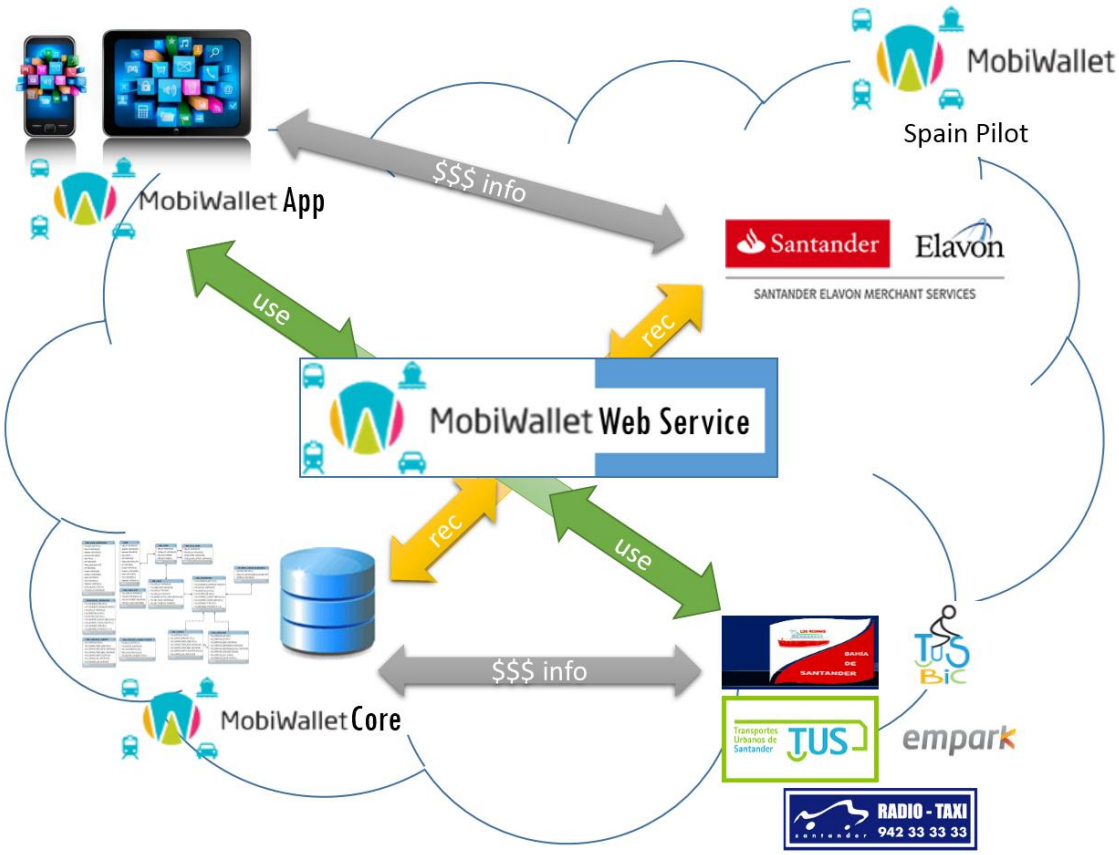

Imagen 11: Composición de la plataforma Mobiwallet en el piloto de Santander

En la imagen superior se puede distinguir cada una de las diferentes partes de la plataforma MobiWallet que a continuación se irán describiendo:

- **MobiWallet Core**

Es la parte de la plataforma donde reside la información de los usuarios y operadores de MobiWallet. Se trata de una base de datos desarrollada en lenguaje MySQL que se ha montado en un servidor interno de la red del laboratorio de telemática de la universidad de Cantabria. Es ajena al Web service para evitar que los datos se expongan directamente a internet, reduciendo el riesgo de ataques. Consta de diferentes tablas que albergan todos los datos de la plataforma, de tal forma que los usuarios y operadores puedan realizar todas las funciones requeridas. También contiene datos que se intercambian con el TPV cuando el usuario hace las recargas de saldo. Debida a la cantidad y complejidad de los datos que se almacenan, se generan backups diarios, que garantizan en caso de caída del servidor o pérdidas de datos, el poder restablecer la base de datos con todo su contenido.

- **MobiWallet Web Service**

El Web service tiene como principal función comunicar el resto de partes de la plataforma. La comunicación directa con la base de datos podría poner en riesgo la integridad y confidencialidad de los datos almacenados, aparte de la dificultad que entrañaría el acceso y la interpretación de esta información. Por tanto para abstraer a los usuarios de esta complejidad y problemas de seguridad, se desarrolla el Web Service. Al igual que la base de datos reside en la red LAN del laboratorio de Telemática de la Universidad. El Web Service es una aplicación web que está ejecutándose en el contenedor de aplicaciones Apache Tomcat. Realmente existen dos instancias de Tomcat, ya que el servicio está replicado, formando ambas instancias un clúster. De esta forma se consigue balancear la carga para que el sistema tenga una mayor capacidad de aceptación de conexiones y por tanto una alta disponibilidad. Destacar que el Web Service no está expuesto directamente a internet, sino que se han configurado ciertos componentes que se interponen entre el cliente y el servidor, con el objetivo de aumentar la seguridad tanto en el acceso al servicio como en las comunicaciones. La interfaz con la que se realizan la comunicación es mediante documentos XML que viajan encapsulados en cada petición a través del protocolo HTTP. Se ha desarrollado una API de usuario y otra de operador para definir las diferentes peticiones que se pueden enviar, mientras que las comunicaciones con el TPV se realizan mediante una API propia del banco Santander.

- **TPV virtual**

El TPV Virtual o más comúnmente Pasarela de Pago es la parte de la plataforma que permite a los usuarios recargar sus tarjetas MobiWallet (saldo virtual o físico), mediante un pago online. Este TPV es un proveedor de servicios de pago o pasarela contratado con el Banco Santander. Destacar que la mayor parte de comunicaciones con el TPV las realiza el Web service. Concretamente, cuando un usuario quiere realizar una recarga de saldo, se lo notifica al Web service mediante una petición, siendo este quien se comunica directamente con el TPV. El usuario únicamente mantiene comunicación directa con la pasarela de pago, a la hora de introducir los datos de su tarjeta, y en la posterior respuesta en la que se notifica el estado del pago realizado. El TPV permite además almacenar los datos de las tarjetas de crédito de los usuarios cuando estos realizan una recarga.

- **Operadores** (TUSBIC, TUS, Empark, Pedreñeras, Radio-Taxi)

Son los diferentes medios de transporte, en los cuales los usuarios van a poder hacer uso de la tarjeta NFC de MobiWallet para pagar las tarifas y obtener así los tickets de viaje. Los operadores emplean ticketadoras y lectores NFC para leer las tarjetas de los usuarios cuando realizan pagos. Estas herramientas se comunican con el Web service notificando los cobros realizados, para que se realicen los cargos correspondientes sobre los saldos asociados a las tarjetas de los usuarios. Existe una excepción en la que un operador no se comunica con el Web service, es el caso del autobús urbano (TUS),puesto que las ticketadoras que van a bordo dentro del autobús, no tienen conexión a internet, por lo que únicamente se puede pagar con el saldo físico de la tarjeta NFC. Para el resto de operadores existe una API para comunicarse con el Web service, pudiendo realizarse las siguientes funcionalidades:

  - *Solicitud de cobro en monedero virtua*l. Permite realizar el cobro a los usuarios leyendo únicamente el identificador de la pegatina NFC. El Web Service procesa la petición y en el caso de que fuera correcta descuenta del saldo virtual del usuario la cantidad correspondiente al pago del servicio.

- *Consulta de tarifas*. El operador puede consultar en todo momento las tarifas que tiene vigentes en la plataforma MobiWallet**.**
- *Consultado de tarjeta de usuario*. Permite al operador conocer la información de la tarjeta NFC de un usuario (si la tarjeta existe, si está activada y si tiene saldo disponible)
- *Consulta de tarjetas MobiWallet activas*. El operador puede obtener un listado de las tarjetas NFC de MobiWallet que podrían hacer uso de sus servicios.
- *Consulta de cobros realizados*. Permite poder obtener un listado de todas las operaciones de cobro realizadas por un operador sobre las tarjetas NFC de Mobiwallet.

- **MobiWallet App**

Esta es la parte de la plataforma donde se ubica el desarrollo de este proyecto. Como ya se ha mencionado en la introducción, la empresa Indra está desarrollando una aplicación nativa "Android" enfocada a aquellos usuarios que quieran disfrutar de los servicios ofrecidos en MobiWallet. El principal problema de esta aplicación es que está destinada únicamente a aquellos usuarios cuyos dispositivos móviles tengan sistema operativo Android. Por ello surge la idea de este proyecto, estudiar la posibilidad de desarrollar una aplicación híbrida de forma paralela a la aplicación oficial que permita llegar al mayor número de usuarios.

De este modo, resulta interesante conocer las funcionalidades que ofrece la aplicación oficial, para cubrirlas en la medida de los posible, así como para desarrollar aquellas que todavía no estén implementadas, pero que si estén planteadas dentro del proyecto MobiWallet. Actualmente la aplicación desarrollada por Indra permite únicamente las siguientes funcionalidades: Registro de usuario y Tarjeta MobiWallet, acceso a la aplicación y realizar recargas físicas y virtuales.

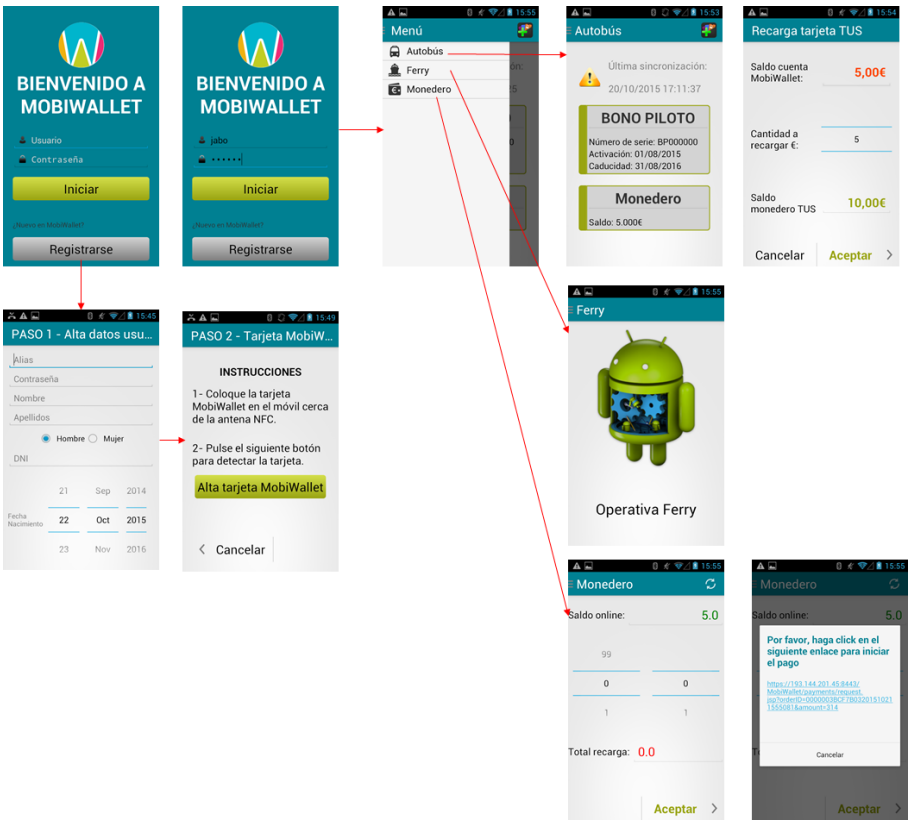

Imagen 12: Capturas de pantalla de las diferentes funcionalidades que ofrece la aplicación desarrollada por Indra

3.2 Requerimientos

Una vez presentada la situación en la que se encuentra la plataforma MobiWallet se procederá a describir los requerimientos que debe cumplir la aplicación híbrida que se va a implementar. Estos requerimientos son básicamente los mismos que se establecieron a la hora de desarrollar la aplicación oficial, aunque en esta, aún no se han desarrollado todos. Se distinguirá entre requerimientos funcionales y requerimientos de seguridad

3.2.1 Requerimientos funcionales

La interacción de los usuarios con la plataforma MobiWallet se realizará íntegramente mediante la aplicación para dispositivos móviles. Por ello hay que estudiar que funcionalidades van a poder realizar los usuarios, para tratar de implementarlas en la aplicación. Como se ha explicado en el punto anterior, las comunicaciones entre los usuarios, y el Web service se realizan mediante una API, que establece como deben llevarse a cabo los intercambios de datos. Hay que tener en cuenta, que dado que el Web service continua desarrollándose, puede que alguna de las funcionalidades ofrecidas en la aplicación no este aún implementada en la API de usuario, por lo que todavía no podrán ser utilizadas.

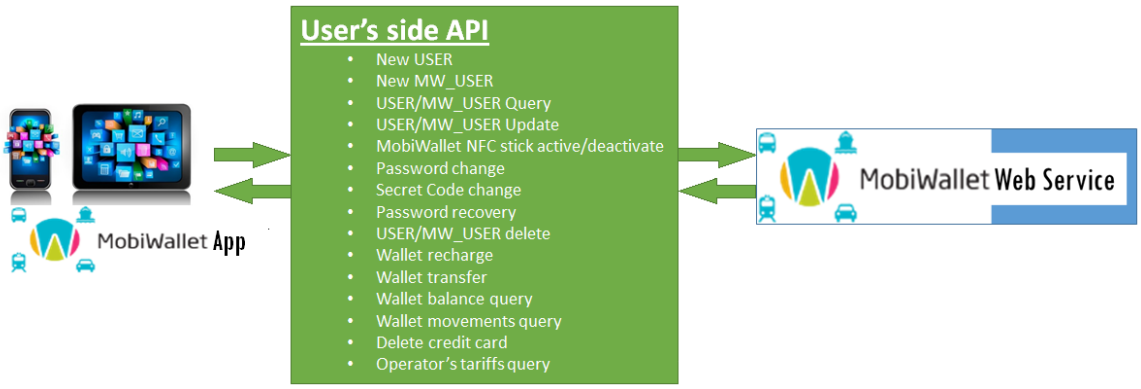

Imagen 13: API de usuario para establecer conexiones entre la aplicación y el Web service

Actualmente está API permite realizar las siguientes funcionalidades de usuario:

- ✓ Registro de usuario en la plataforma MobiWallet.
- ✓ Registro de tarjeta MobiWallet y asociarla al usuario.
- ✓ Actualizar los datos del usuario.
- ✓ Consulta de los datos del usuario.
- ✓ Asociar tarjetas MobiWallet a un usuario
- ✓ Eliminar tarjetas MobiWallet de un usuario.
- ✓ Activar/Desactivar tarjetas MobiWallet.
- ✓ Cambiar la contraseña.
- ✓ Cambiar el Pin Code.
- ✓ Generación de una nueva contraseña.
- ✓ Consulta de saldo virtual
- ✓ Consulta de movimientos
- ✓ Consulta de tarifas.
- ✓ Recarga del saldo físico.
- ✓ Recarga del saldo virtual.
- ✓ Transferencia de saldo virtual a físico.
- ✓ Eliminar datos bancarios (almacenados en el TPV Virtual).

A continuación se explicará cual es el funcionamiento que se pretende conseguir a la hora de desarrollar la aplicación.

En primer lugar, para que un usuario pueda disfrutar de los servicios de MobiWallet (piloto de Santander), ha de registrase en la plataforma. Este registro constará de una primera parte en la que se introducen los datos del usuario, y a continuación se solicitará una Tarjeta MobiWallet (la cual habrá obtenido el usuario apuntándose en el proyecto piloto en el Ayuntamiento de Santander), y se le pedirá que introduzca una contraseña y un Pin Code. De esta forma se asociara la tarjeta al usuario. MobiWallet permite que un usuario pueda tener asociadas más de una tarjeta.

A partir de aquí el usuario dispondrá de dos saldos, uno virtual y otro físico, asociados a su tarjeta:

- **Saldo físico**. Es el saldo del monedero físico integrado dentro del chip NFC de la tarjeta MobiWallet. Con este saldo el usuario puede pagar únicamente el transporte en autobús urbano (TUS).
- **Saldo Virtual**. Es el saldo virtual asociado a la tarjeta MobiWallet del usuario. Este saldo se almacena en la base de datos de la plataforma MobiWallet. Con el saldo virtual el usuario puede pagar todos los transportes incluidos dentro del proyecto piloto de Santander (Pedreñeras, taxis, parking, bicicletas) excepto el servicio de autobuses (TUS).

Después de que el usuario se ha registrado ya tiene que poder acceder a la aplicación, para ello se le solicitará el identificador de usuario y la contraseña, y en caso de tener varias tarjetas asociadas el identificador de tarjeta con la que se quiere acceder. En caso de que el usuario olvide su contraseña se le proporcionará la opción de recibir una nueva en la dirección de correo que introdujo a la hora del registro.

Una vez dentro de la aplicación, el siguiente paso es poder recargar la tarjeta. Aquí el usuario tendrá dos opciones, recargar saldo físico o virtual. Como se ha comentado el saldo físico únicamente se podrá usar en el servicio de autobuses TUS, mientras que el saldo virtual se usará en el resto de operadores, por lo tanto el tipo de recarga a realizar dependerá de que servicios se vayan a utilizar, aunque lo más lógico sería disponer tanto de saldo físico como de saldo virtual.

Estas recargas se realizarán mediante la pasarela de pago del Banco Santander (TPV Virtual Santander Elavon), de forma que el usuario deberá introducir los datos de su tarjeta de débito para hacer el pago. El TPV Virtual permitirá al usuario guardar los detalles de su tarjeta, de forma que en las próximas recargas, el usuario no tendrá que introducir de nuevo sus datos bancarios. En consecuencia, el usuario también va a tener la opción de eliminar del TPV los datos de su tarjeta en el caso de que los hubiera guardado.

Si el usuario selecciona la recarga del saldo virtual, directamente al realizar el pago en el TPV se le sumará el saldo en el servidor. En el caso de recarga física, el usuario deberá de disponer de un Smartphone que soporte la tecnología NFC (el chip NFC es del tipo Mifare Classic), con lo que simplemente acercando la tarjeta al móvil quedaría realizada la recarga. En el caso de que el usuario no disponga de un móvil compatible, no se le permitirá realizar recargas físicas. Para evitar este problema, los operadores tendrían repartidos por la ciudad unos Totem de recarga para que los usuarios pudieran realizar recargas físicas en sus tarjetas.

Otra funcionalidad que se le va a permitir al usuario es el realizar un traspaso de su saldo virtual a saldo físico, pero no a la inversa. Al igual que en las recargas físicas, solo será posible esta opción si el dispositivo del usuario dispone de la tecnología NFC, y evidentemente si su saldo virtual no es nulo.

Todas las recargas y traspasos de saldo van a estar almacenadas como movimientos del usuario en la base de datos de MobiWallet, por lo que el usuario va a tener que poder consultarlos en todo momento. Además el saldo virtual será mostrado en todo momento, mientras que el saldo físico, solo cuando se lea la tarjeta mediante la tecnología NFC.

Dentro de la aplicación el usuario también podrá consultar y modificar sus datos personales. Por otro lado se incluirán numerosas funcionalidades asociadas a las tarjetas MobiWallet, como activar/desactivar tarjeta, asociar/eliminar tarjeta, cambiar contraseña o cambiar Pin Code.

Por último, cada operador dispondrá de una serie de opciones, alguna de ellas comunes como las recargas de saldo o las consultas de tarifas, mientras que otras serian específicas como el poder realizar compras de tickets, o realizar pagos directos desde la aplicación.

3.2.2 Requerimientos de seguridad

Se puede determinar que todas las funcionalidades que se pretenden desarrollar en la aplicación necesitan de intercambio de información con el Web service. Puesto que es posible que en estas comunicaciones circulen datos sensibles (identificadores de tarjetas, claves de tarjeta, contraseñas de usuario…) es necesario que se realicen de forma segura. Para ello, el servidor debe de poder garantizar que el usuario es quien realmente dice ser, y por otro lado, se debe de proteger el flujo de datos, codificándolos de forma que no puedan ser descifrados. Por tanto los mecanismos de seguridad desarrollados en el servidor son los que hay que implementar en la aplicación

Las comunicaciones entre la aplicación y el servidor se realizan mediante el intercambio de ficheros XML, encapsulados en mensajes HTTP. Como se ha comentado existe una API de usuario, en la que se define el método de acceso HTTP y la plantilla XML requeridos en cada tipo de petición. Para dotar de seguridad a las comunicaciones, es necesario cifrar los mensajes mediante el protocolo SSL/TLS. Por tanto realmente el protocolo que se va a tratar de implementar en la aplicación a la hora de realizar intercambios de información es el HTPPS, de esta forma se logra proteger el flujo de datos [44].

Por otro lado para garantizar que quien trata de acceder es realmente un usuario MobiWallet, el Web service se ha desarrollado de forma que escucha por dos puertos, en ambos esta implementado el cifrado SSL, pero cada uno emplea un método de autenticación distinto.

En el uno de los puertos se emplea el cifrado SSL con autenticación de servidor (one-way authentication) o SSL ligero, de forma que cuando un usuario abre una conexión con ese puerto, únicamente el servidor presenta su certificado al cliente para verificar su identidad.

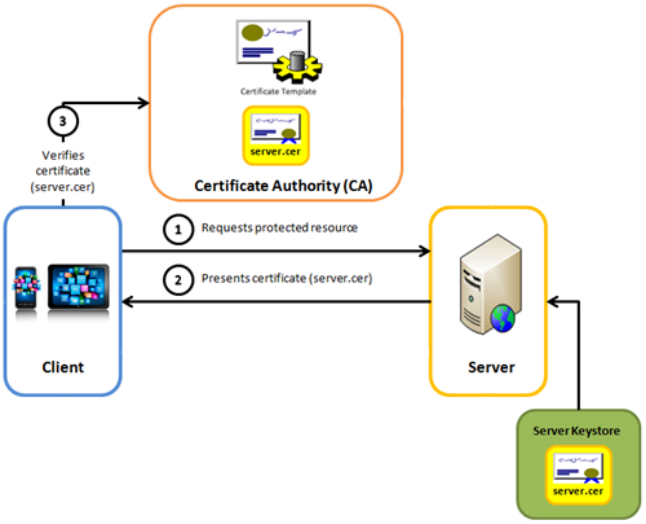

Imagen 14: Protocolo SSL con autenticación del servidor [45]

Por el contrario en el otro puerto, se implementa SSL con autenticación mutua (two-way authentication), en este caso tanto cliente como servidor tienen que intercambiar sus certificados digitales para autenticarse y poder establecer el canal de comunicaciones. De esta forma se realizaría una comunicación asegurada en los dos extremos, con lo que se podrían evitar o reducir posibles ataques como por ejemplo el famoso "man in the middle" o ataques de replay.

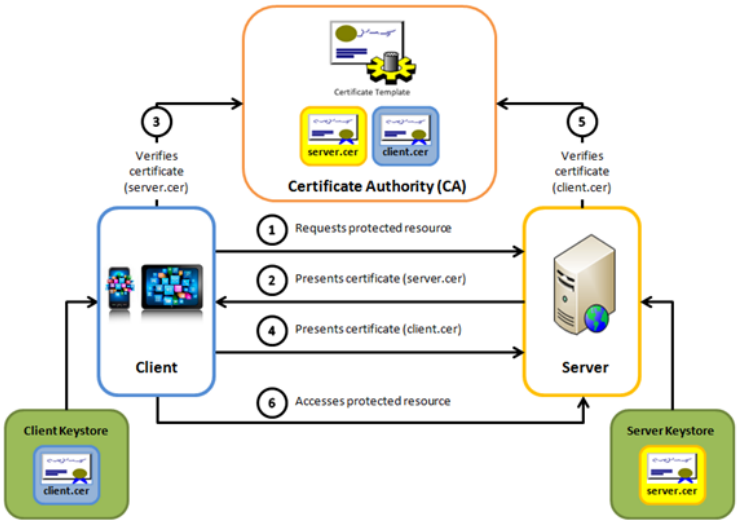

Imagen 15: Protocolo SSL con autenticación mutua [45]

Por tanto, la aplicación se tratará de implementar de forma que se pueda comunicar con el Web service mediante ambos métodos. Se accederá a través del puerto que tiene implementado SSL con autenticación del servidor, en el registro de nuevos usuarios en la plataforma MobiWallet. Cuando el registro se finalice de forma correcta, el servidor habrá generado un certificado para este usuario, y la aplicación deberá descargar y almacenar este certificado. De esta forma la aplicación podrá realizar las demás comunicaciones mediante SSL con autenticación mutua. El resto de conexión por tanto se realizara sobre el otro puerto, teniendo que presentar el usuario su certificado para poder establecer la comunicación.

4 IMPLEMENTACIÓN

En este apartado se explicará cómo se ha llevado a cabo el desarrollo de la aplicación hibrida en la que se centra este proyecto. Para ello se introducirá en primer lugar el entorno de desarrollo empleado, así como las diferentes tecnologías usadas. A continuación se presentará el contenido de la aplicación explicando su estructura y cada una de sus partes, pero sin profundizar en el código. En los puntos restantes sí que se mostrarán aspectos técnicos, como APIS y código utilizado para conseguir ciertas funcionalidades, así como problemas encontrados durante el desarrollo y las soluciones adoptadas.

4.1 Entorno de desarrollo

A la hora de implementar una aplicación, el primer paso es decidir el entorno de desarrollo, para ello hay que analizar entre las diferentes opciones, cual es la que mejor se adapta a las necesidades de cada caso.

En este proyecto el entorno de desarrollo que se va a utilizar es Intel XDK. Se trata de una herramienta (IDE) para desarrollar aplicaciones híbridas y juegos programando en tecnologías estándar como HTML5, CSS3 y JavaScript. Desde una misma base de código permite generar apps para distintas plataformas. Tiene la característica de que reúne todas las herramientas necesarias para desarrollar una aplicación desde su diseño inicial hasta su publicación. Estas herramientas están disponibles en 6 pestañas en la parte superior del programa, y son las siguientes [46]:

- Desarrollo
  Es la herramienta en la que más tiempo se empleará, pues es donde propiamente se desarrolla la aplicación. Permite utilizar dos métodos, el típico editor de código y una herramienta de diseño. Esta última opción es muy intuitiva ya que lo único que hay que hacer para diseñar es arrastrar los diferentes elementos (botones, imágenes, textos, etc.) hasta dar forma a la aplicación, aunque tiene el inconveniente de que se produce demasiado código innecesario. Dentro de este interfaz gráfico podemos elegir entre diferentes frameworks: App Framework, Bootstrap, JQuery Mobile, TopCoat, Ratchet, Ionic y Framework7.
- Simulación
  Se trata de un simulador basado en Apache Ripple, que permite emular la aplicación que se está desarrollando en una gran variedad de dispositivos (iPhone, Microsoft Surface, Google Nexus, entre otros).
- Test
  Permite realizar pruebas en dispositivos físicos antes de construir la aplicación, para ello es necesario descargar en el dispositivo una aplicación de Intel llamada “Intel App Preview” (disponible para iOS, Android y Windows phone). Existen dos formas diferentes para probar la aplicación que se está desarrollando. La primera de ellas es subir el proyecto a un servidor de testing que ofrece Intel, de forma que se mostrará en la aplicación “Intel App Preview”, desde donde podremos ejecutar la aplicación desarrollada. La segunda forma es conectar el dispositivo móvil a la misma red Wifi en la que se encuentra el ordenador desde el que se está desarrollando la aplicación.
- Depuración
  Se trata de una depuración USB, y únicamente está disponible para dispositivos con sistema operativo Android 4 o superior.

- Perfil
  Al igual que la depuración, solo está disponible para dispositivos con sistema operativo Android 4 o superior, permite obtener información sobre el rendimiento de la aplicación.
- Construcción
  Esta última herramienta es la que permite generar la aplicación para diferentes plataformas:
    - Aplicaciones móviles: iOS, Android (nativo, Cordova, Crosswalk), Windows 8 Store, Windows Phone 8, y Nook.
    - Aplicaciones web: Web, Chrome App, Facebook App [47].

  Es muy sencillo de utilizar únicamente hay que configurar alguna opción acerca de la aplicación. Las diferentes plataformas tienen diferentes requisitos de configuración. Por ejemplo, para construir una aplicación para iOS, se requiere alguna información sobre la cuenta de desarrollador de iOS. Tras elegir las opciones de la aplicación, el archivo se construye y obtendrá un cuadro de diálogo que permite descargar la aplicación.

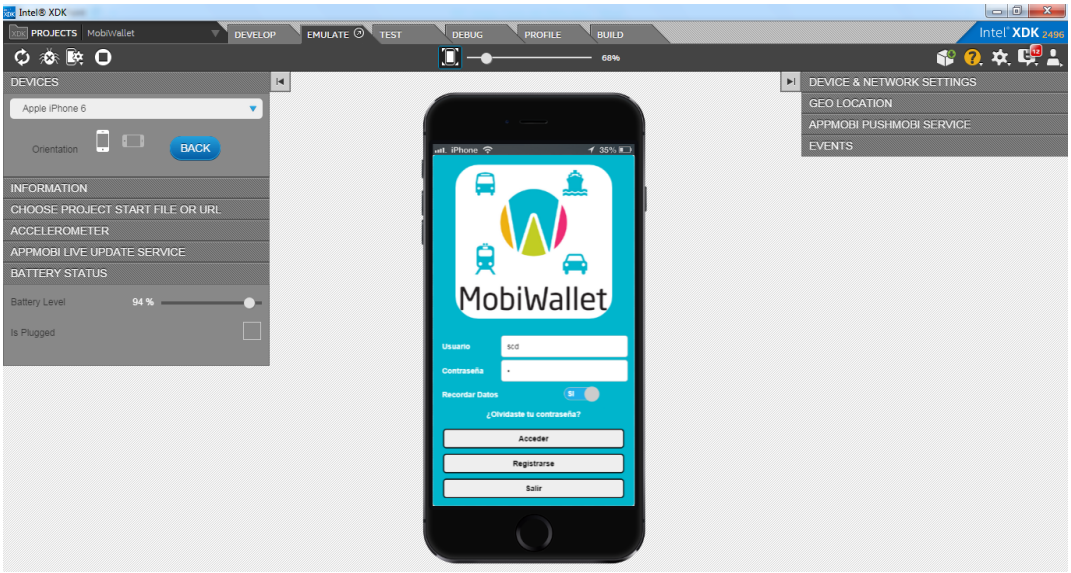

Imagen 16: Aspecto del entorno de desarrollo Intel XDK

Una vez explicado el funcionamiento de las diferentes utilidades que proporciona este entorno de desarrollo, se describirán las principales características que han hecho que esta sea la herramienta escogida para desarrollar la aplicación:

- Posibilidad de distribuir las aplicaciones realizadas a múltiples tiendas y formatos
- Desarrollo eficiente, diseño, prueba y armado de aplicaciones.
- Integra simuladores de diferentes dispositivos ya pre instalados.
- Integración de API y plugins de Apache Cordova
- Posibilidad de almacenar código en la nube de forma gratuita.
- Es gratuito
- Compatible con múltiples plataformas
- Posibilidad de arrastrar componentes a tu área de trabajo para realizar el armado de interfaces.
- Plantillas pre diseñadas para empezar tus proyectos.
- Potente editor de código al estilo Brackets ya que te permite ver en tiempo real los cambios realizados en tu proyecto ya sea en un dispositivo o en el navegador [48].

4.2 Tecnologías empleadas

Una vez definido el entorno de desarrollo se procederá a describir las diferentes tecnologías que se emplearan para desarrollar la aplicación.

- **HTML5** (*HyperText Markup Language*)

Es un lenguaje de programación que se utiliza para crear y representar visualmente páginas web, determinando el contenido de la página web, pero no su funcionalidad [49]. Está compuesto por una serie de etiquetas, también llamadas tags, que el navegador interpreta y da forma en la pantalla. HTML dispone de etiquetas para imágenes, hipervínculos que nos permiten dirigirnos a otras páginas, saltos de línea, listas, tablas, etc. [50]

HTML5 es la última versión de HTML e introduce nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Por ejemplo <nav> (bloque de navegación del sitio web) y <footer> son técnicamente similares a las etiquetas y , pero tienen un significado semántico. Otros elementos proporcionan nuevas funcionalidades a través de una interfaz estandarizada, como los elementos <audio> y <video>. También se ha conseguido una mejora del elemento <canvas>, capaz de renderizar elementos 3D en los navegadores más importantes (Firefox, Chrome, Opera, Safari e Internet Explorer). [51]

- **Javascript**

Se trata de un lenguaje de programación que permite a los desarrollares crear acciones en sus páginas web, obteniendo paginas dinámicas [52]. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Es decir mientras que con HTML5 se define la estructura y representación de la página web, con javascript se dota de funcionalidad. JavaScript es un lenguaje de programación interpretado, es decir, no es necesario compilar los programas para ejecutarlos.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Javascript es un lenguaje interpretado, basado en prototipos, mientras que Java es un lenguaje más orientado a objetos [53].

- **CSS3 (Cascading Style Sheets)**

Es un lenguaje de programación que sirve para organizar la presentación y aspecto de una página web. Ofrece multitud de opciones de presentación como colores, tipos y tamaños de letra, etc. [54] Por tanto, mientras que HTML permitía definir la estructura una página web y Javascript la funcionalidad, las hojas de estilo en cascada (CSS) definen las reglas y estilos de representación en diferentes dispositivos, ya sean pantallas de equipos de escritorio, portátiles, móviles, impresoras u otros dispositivos capaces de mostrar contenidos web [55]. La versión 3 es la última que se ha desarrollado y sus principales características se centran en abarcar un control más amplio sobre el estilo de los elementos de la página web y con un mayor número de efectos visuales.

- **App Framework**

Se trata de un Framework de interfaz de usuario multi-plataforma diseñado para desarrollar aplicaciones HTML5 en dispositivos móviles. Anteriormente se conocía como JQmobi pero en 2013 lo compró Intel y cambiaron el nombre.

Cuando se comienza a desarrollar una aplicación el entorno de desarrollo Intel XDK permite escoger entre diferentes frameworks. En este caso se ha elegido AppFramework por dos motivos:

- Los estilos de interfaz de usuario de esta biblioteca están diseñados para adaptarse fácilmente a la plataforma de destino (Android, iOS , Windows Phone, y RIM (Blackberry)) dando a la aplicación un aspecto específico.
- Aunque no es tan renombrado como jquery mobile o sencha touch, es mucho más veloz y ligero [56].

Imagen 17: Comparativa de tamaño y velocidad en diferentes Frameworks [57]

- **Cordova**

Cordova es un framework que permite empaquetar aplicaciones web dentro de una aplicación nativa, permitiendo acceder desde Javascript a funcionalidades del dispositivo [58].

Generalmente con las APIs disponibles en Javascript, no es posible acceder a características del dispositivo como la cámara, el acelerómetro, el uso de batería, el almacenamiento en el dispositivo, etc. Lo que proporciona Cordova es una capa intermedia que a través de sus propias APIs, se comunica con el dispositivo permitiendo el acceso a estas características. De esta forma, la aplicación queda finalmente empaquetada como una app nativa, y Cordova se encarga de traducir las llamadas hechas a los plugins de su sistema a llamadas en código nativo.

Cordova presenta una implementación única para todas las plataformas. Es decir, se desarrolla la funcionalidad de la aplicación en javascript y Cordova se encarga de traducir a los distintos sistemas operativos para los que se genere la aplicación [59].

El entorno de desarrollo Intel XDK permite al desarrollador seleccionar los plugins de Cordova que necesite para acceder a las diferentes características del dispositivo. Cuando la aplicación se construya, los plugins seleccionados se cargaran automáticamente junto con el código desarrollado [60].

4.3 Estructura de la aplicación

Este apartado es probablemente uno de los más importantes de esta memoria, puesto que se describe como se ha implementado la aplicación. El desarrollo de todo software conlleva una gran cantidad de código, pero no es oportuno ir explicándolo línea a línea. Por ello se irá describiendo a nivel funcional cada una de las partes de la aplicación, mostrándose sus aspectos más destacados, así como las diferentes acciones que se llevarán a cabo al pulsar cada elemento interactivo.

Resaltar que la mayor parte de funciones implementadas en la aplicación requieren de intercambios de información con el Web service, por tanto de envió/recepción de ficheros XML. En este aspecto se indicará cuando se produce una petición al Web service, pero no se irán detallando cada uno de los ficheros XML. Para ello se ha incluido un anexo con un ejemplo de las plantillas que se envían desde la aplicación y las posibles respuestas que ofrece el Web service. Por motivos de confidencialidad los detalles del resto de plantillas solo serán referenciados al documento interno de especificación de la API de MobiWallet [61].

A continuación se procederá describir cada una de las partes y/o funcionalidad de las que se compone la aplicación desarrollad

4.3.1 Pantalla de inicio

En primer lugar se diseñó la pantalla de inicio que se muestra al arrancar la aplicación.

Imagen 18: Pantalla que se muestra al iniciar la aplicación

Como se puede apreciar en la parte superior del dispositivo se muestra el logo de MobiWallet. La parte central consta de dos campos de entrada de texto para introducir el usuario y la contraseña respectivamente, y un botón de tipo “flip-switch” que permite recordar los datos introducidos de cara a futuras sesiones. Por último en la parte inferior aparecen 4 botones, el primero de ellos semi-oculto, permite al usuario recibir una nueva contraseña en su email, en caso de la haya olvidado. Los otros tres, permitirán el acceder, registrarse y salir de la aplicación respectivamente.

4.3.2 Registro de usuario y tarjeta MobiWallet

Para poder hacer uso de la aplicación es necesario disponer de un usuario con tarjeta MobiWallet asociada, por tanto la primera funcionalidad que se desarrollo fue el poder registrar usuarios y su correspondiente tarjeta MobiWallet.

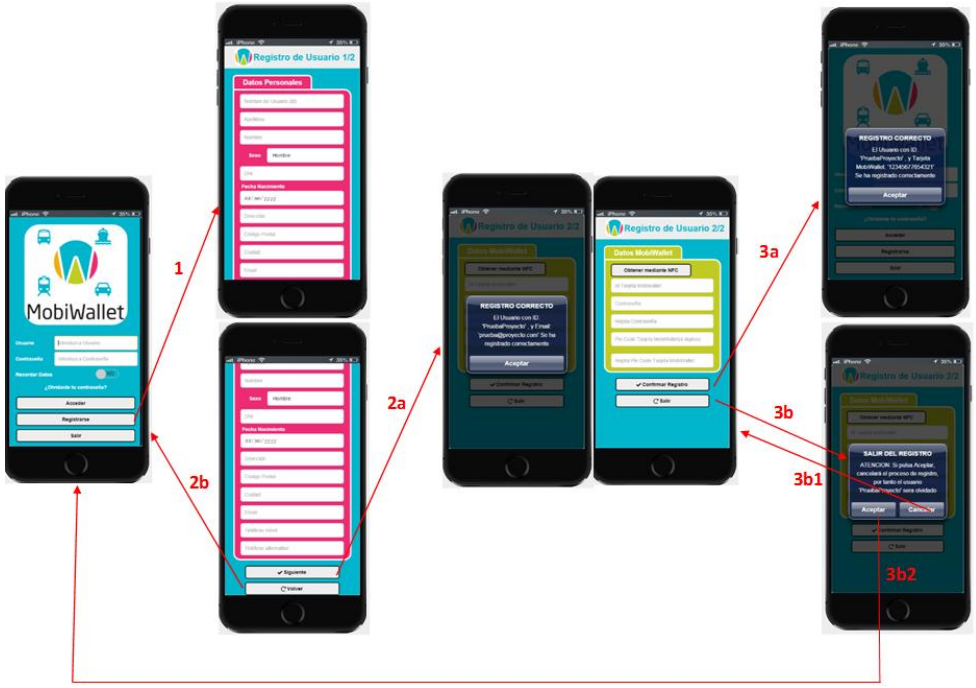

Imagen 19: Secuencia a seguir para el registro de usuario y tarjeta MobiWallet

El funcionamiento para registrar un usuario es el siguiente:

1. Pulsar el botón “Registrarse”. Se accede al menú de registro de usuario.
2. Se introducen los diferentes datos de usuario requeridos (Nombre de usuario, Apellidos, Nombre, Sexo, DNI, Fecha de nacimiento, Dirección, Código Postal, Ciudad, Email, Teléfono móvil, Teléfono Alternativo). Existen dos posibilidades:
    a. Pulsar el botón “siguiente”: Si todos los datos están completos y son correctos se envían en una petición “RegisterUser” [Plantilla Anexo]. Si el registro ha sido correcto se mostrará una alerta y se accederá al siguiente menú, en caso contrario se mostrará el error y se permanecerá en la misma pantalla.
    b. Pulsar el botón “volver”: Se regresará a la pantalla de inicio.
3. Se introducen los diferentes datos de Tarjeta MobiWallet (Identificador de Tarjeta, Contraseña, Repetir Contraseña, Pin Code, Repetir Pin Code). Existen dos posibilidades:
    a. Pulsar el botón “Confirmar Registro”: Si todos los datos están completos y son correctos se envían en una petición “RegisterMWUser” [61]. Si el registro ha sido correcto se mostrará una alerta y se volverá a la pantalla de inicio, en caso contrario se mostrará el error y se permanecerá en la misma pantalla.
    b. Pulsar el botón “salir”. Se mostrará una ventana de alerta, preguntando si realmente desea abandonar el registro.
        1. En caso de pulsar el botón “Aceptar”, se eliminará el usuario registrado en el menú anterior, puesto que no puede existir un usuario sin tarjeta asociada, para ello se envía una petición “DeleteUser” [61]. Se regresa a la pantalla de inicio.
        2. En caso de pulsar el botón “Cancelar”, se volverá al paso 3.

4.3.3 Recordar Contraseña

Es posible que los usuarios olviden la contraseña que establecieron, por tanto es necesario desarrollar un método que les permita obtener una nueva. Se ha optado por solicitar el identificador de la Tarjeta MobiWallet de la cual se desea recuperar la contraseña, y el email asociado al usuario propietario de dicha tarjeta, que será donde se envié la nueva contraseña.

Imagen 20: Secuencia a seguir para recordar contraseña

El funcionamiento para recuperar la contraseña de un usuario es el siguiente:

1. Pulsar el texto “¿Olvidaste tu contraseña?”. Se accede al menú de recuperación de contraseña.
2. Se introducen los diferentes datos de requeridos (Identificador de tarjeta, email del usuario al que pertenece la tarjeta). Existen dos posibilidades:
   a. Pulsar el botón “Recuperar” Si todos los datos están completos y son correctos se envían en una petición “DataRecovery” [61]. Si el proceso de recuperación ha sido correcto se mostrará una alerta y se volverá a la pantalla de inicio, en caso contrario se mostrará el error y se permanecerá en la misma pantalla.
   b. Pulsar el botón “volver”: Se regresará a la pantalla de inicio

4.3.4 Acceso a la aplicación

El siguiente paso, es desarrollar el método de acceso al contenido de la aplicación para los usuarios registrados. Como se puede ver en la pantalla de inicio, únicamente se solicitan el identificador de usuario y la contraseña, por lo que puede parecer sencillo. Pero estos dos parámetros no son suficientes puesto que puede darse el caso de que un usuario tenga asociadas varias tarjetas. Por tanto será necesario un tercer parámetro, el identificador de tarjeta. Pero el solicitar al usuario el identificador de tarjeta en cada acceso puede resultar pesado, puesto que en la mayoría de los casos los usuarios tendrán una única tarjeta. Por ello se ha planteado la opción de que el usuario pueda establecer como predeterminada una tarjeta MobiWallet, de forma que siempre que acceda con el identificador de usuario y la contraseña de esa tarjeta predeterminada, no se le solicitará el identificador de tarjeta.

Además se ha incluido un botón “flip-switch” que permite al usuario recordar sus datos. De esta forma el acceso para aquellos usuarios que utilicen a menudo la aplicación resultara sencillo puesto que no tendrán que introducir continuamente sus credenciales para acceder. Resaltar que tanto los datos predeterminados, como los datos recordados, se almacenan en la memoria del dispositivo.

En el siguiente diagrama se representa el proceso de acceso a la aplicación

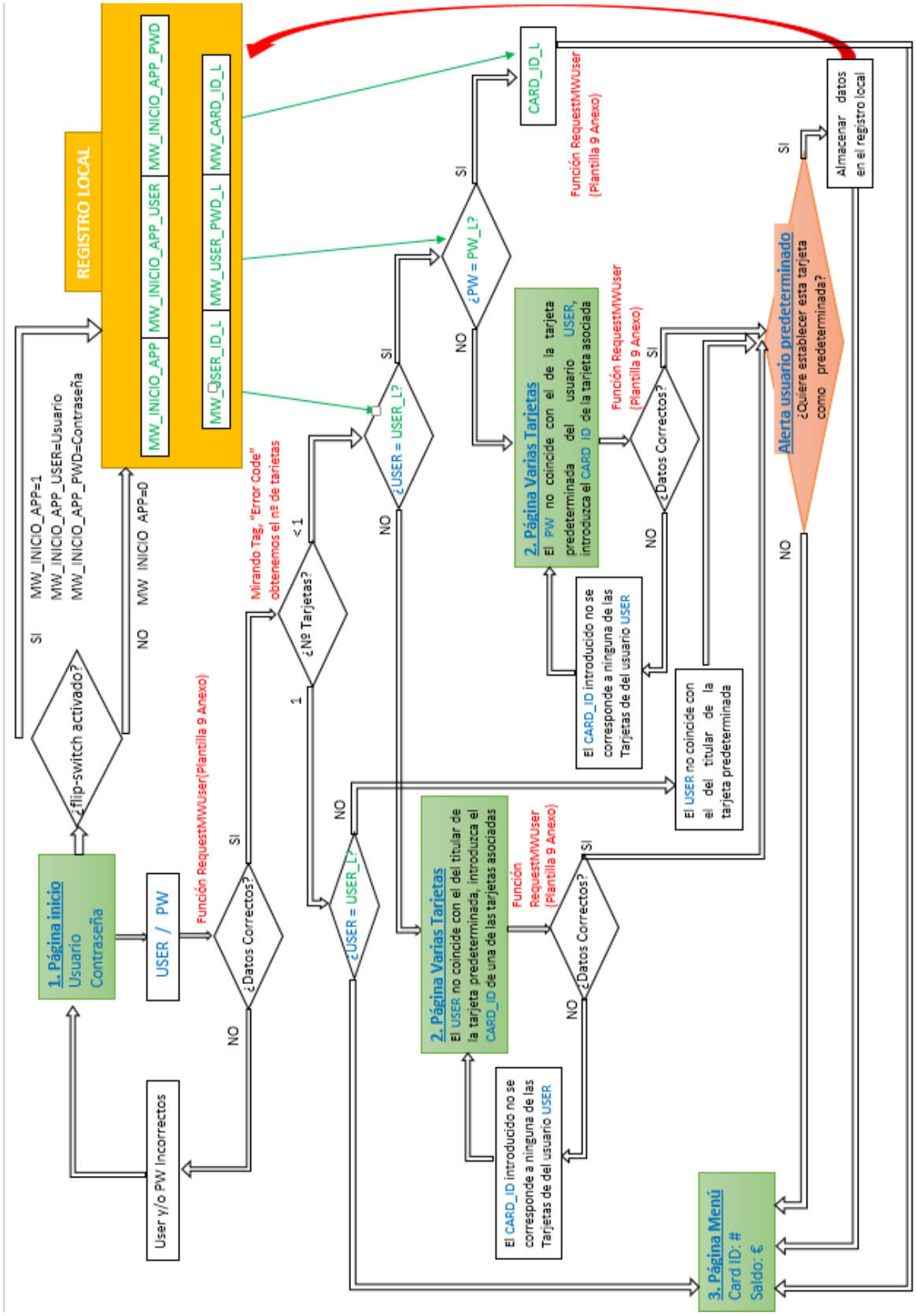

Imagen 21: Diagrama del proceso que sigue cuando se realiza un acceso

En el diagrama anterior, se ha resaltado en color verdoso, las páginas que intervienen al acceder en la aplicación. Destacar que únicamente aparecen tres páginas: la de inicio de la aplicación, la página en la que se solicita el identificador de tarjeta en el caso de que el usuario tenga varias, y por último cuando el acceso es exitoso, la página del menú principal. Por contra, se desarrolla una lógica de acceso compleja, en la que se realizan hasta 4 peticiones "RequestMWUser" [61] y varias lecturas y escrituras de los datos almacenados en el dispositivo móvil. Se puede concluir por tanto que para obtener el método de acceso se ha implementado un mayor desarrollo de JavaScript (funcionalidad) que de HTML (representación visual). A continuación se muestran las 3 páginas que intervienen en el acceso, y la alerta que pregunta al usuario si desea establecer la tarjeta con la que está iniciando sesión como predeterminada.

Imagen 22: Páginas que intervienen en el acceso a la aplicación

El menú principal de la aplicación consta de una parte superior en la que se muestran dos campos con el identificador de tarjeta MobiWallet y el saldo virtual de esta; además de un botón que permite actualizar el saldo en todo momento mediante una petición "ViewBalance" [61]. Por otro lado la parte inferior está formada por una serie de botones que permitirán acceder a las diferentes funcionalidades (Datos Usuario, Opciones de Cuenta, Consultar Movimientos y los submenús de cada operador) que se irán describiendo en los puntos siguientes.

4.3.5 Datos de usuario

Esta funcionalidad permite a los usuarios consultar sus datos personales, que introdujeron a la hora de realizar el registro. Además de consultarlos, también tienen la opción de modificarlos.

Imagen 23: Secuencia a seguir para consultar y/o modificar los datos de usuario

El funcionamiento para consultar y modificar los datos personales de usuario es el siguiente:

1. Pulsar el botón “Datos Usuario”. Se envía una petición “RequestMWUser” [61] y se accede a la página datos de usuario.
2. Se muestra la información personal del usuario. Existen dos posibilidades:
   a. Pulsar el botón “Modificar Datos Usuario”, se accede a la página “Modificar Datos”
   b. Pulsar el botón “Volver Menú”: Se regresará al menú principal
3. Se muestran los diferentes campos con la información personal del usuario, pero ahora con la posibilidad de editarlos.
   a. Pulsar el botón “Guardar Modificaciones”. Si todos los datos están completos y son correctos se envían en una petición “UpdateMWUser” [61]. Si la modificación de datos ha sido correcta se mostrará una alerta y se volverá al menú principal, en caso contrario se mostrará el error y se permanecerá en la misma pantalla.
   b. Pulsar el botón “Volver Menú”. Se regresará al menú principal sin guardar ninguna modificación.

4.3.6 Opciones de Cuenta

Al pulsar sobre el botón “Opciones Cuenta” se accede a un submenú que permite realizar diferentes operaciones con las tarjetas MobiWallet.

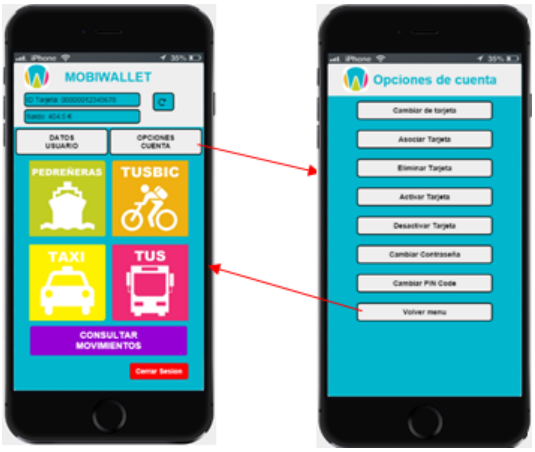

Imagen 24: Acceso al menú "Opciones Cuenta"

A continuación se ira describiendo en que consiste cada una de estas operaciones, y el procedimiento para llevarlas a cabo:

- **Cambiar de tarjeta**

  Esta funcionalidad se ha desarrollado para que aquellos usuarios que posean varias tarjetas tengan la posibilidad de cambiar de una a otra. Para ello únicamente deberán introducir el identificador y la contraseña de la tarjeta MobiWallet. Desde el punto de vista de la implementación es similar a realizar un acceso.

Imagen 25: Secuencia a seguir para cambiar de tarjeta

  El funcionamiento es muy sencillo, una vez en la ventana “Cambiar de Tarjeta”, se introducirá el identificador y la contraseña de la tarjeta MobiWallet a la que se quiere acceder. Si ambos campos están completos y son correctos se envían en una petición “RequestMWUser” [61]. Si la consulta es incorrecta se mostrará el error en esta misma ventana, si por el contario es correcta se mostrará una alerta, preguntando al usuario si desea que esta tarjeta sea la predeterminada. Independientemente de la opción escogida por el usuario, se mostrará el menú principal, pero ahora en una nueva sesión correspondiente a la tarjeta a la que se ha cambiado.

- **Asociar Tarjeta**

  Puesto que un usuario puede tener la posibilidad de poseer varias tarjetas, es necesario disponer de una función que permita a los usuarios asociar nuevas tarjetas. Desde el punto de vista de la implementación es similar a realizar un registro de tarjeta, por tanto los datos requeridos son: Identificador de tarjeta, contraseña, Repetir contraseña, Pin Code, Repetir Pin Code.

Imagen 26: Secuencia a seguir para asociar tarjeta

Una vez en la ventana “Asociar Tarjeta”, se introducirán los datos requeridos, si todos los campos están completos y son correctos se envían en una petición “RegisterMWUser” [61], si la consulta es incorrecta, se visualizaran los errores en la misma página, en caso contrario se mostrará una alerta, indicando que la tarjeta se ha asociado correctamente.

- **Eliminar Tarjeta**

  Del mismo modo que un usuario puede asociar tarjetas, debe de poder eliminarlas. La implementación de esta función puede parecer sencilla, pero no lo es, puesto que pueden darse varias situaciones dependiendo del número de tarjetas que tiene asociadas un usuario, y dependiendo de si la tarjeta que desea eliminar es aquella con la que ha iniciado sesión.

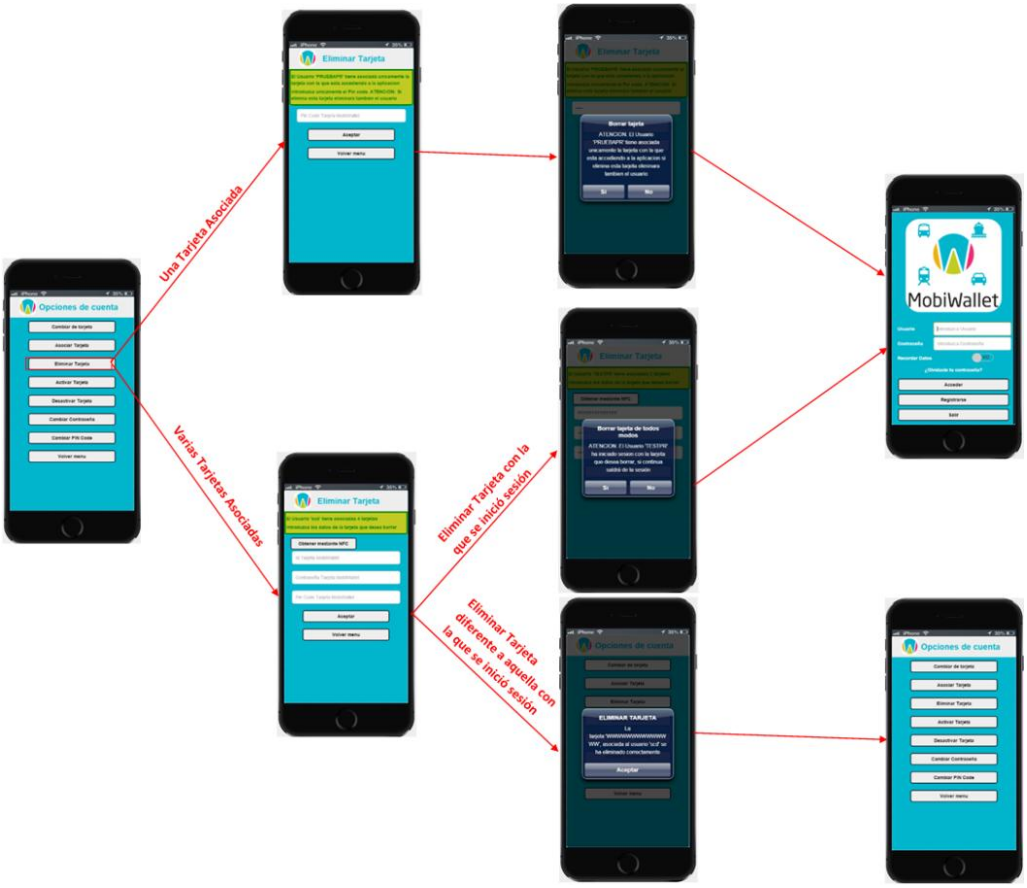

Imagen 27: Secuencia a seguir para eliminar tarjeta

Al pulsar el botón "Eliminar tarjeta" se lanza una petición "RequestMWUser" [61] consultando el número de tarjeta asociadas al usuario, y se accede a la ventana "Eliminar Tarjeta":

- Si el usuario tiene asociada únicamente la tarjeta con la que ha iniciado sesión, solo deberá introducir el Pin Code. Al pulsar el botón "Eliminar Tarjeta", se le advierte mediante una alerta que si elimina la tarjeta, también se eliminará el usuario asociado, puesto que no pueden existir usuarios sin tarjeta. Si la respuesta es afirmativa, se enviará una petición de tipo "deleteMWUser" [61], de forma que si el Pin Code es correcto se eliminará la tarjeta, y acto seguido se enviará una petición de tipo "deleteUser" [61] que eliminará el usuario. Por último se dará por finalizada la sesión de este usuario, ya inexistente, y se mostrará la pantalla de inicio.

- Si por el contrario el usuario tiene asociadas varias tarjetas, deberá introducir los datos de aquella que desea eliminar (identificador, contraseña y Pin Code). Al pulsar el botón "Eliminar Tarjeta" antes de eliminar se comprobará si la tarjeta introducida es aquella con la que ha iniciado sesión, con lo que se dan dos situaciones:
    - En caso de que se trate de la tarjeta con la que ha iniciado sesión, se advertirá mediante un mensaje de alerta que si continua con la eliminación de esa tarjeta se finalizará la sesión. Si confirma que desea continuar se enviará una petición de tipo "deleteMWUser" [61], de forma que si los datos introducidos son correctos se eliminará la tarjeta, y acto seguido se dará por finalizada la sesión de este usuario, mostrándose la pantalla de inicio.
    - En caso de que no se trate de la tarjeta con la que ha iniciado sesión, se envía directamente la petición "deleteMWUser" [61]. Si los datos son correctos se muestra una alerta confirmando la eliminación, y se muestra de nuevo el menú "Opciones de Cuenta"

- **Activar/Desactivar Tarjeta**

  Otra funcionalidad que se ofrece es el poder activar/desactivar las tarjetas MobiWallet, de forma que se pueden bloquear en el caso de pérdida o robo para evitar que se realicen pagos y recargas. Al igual que en el caso anterior no se requerirán los mismo campos a los usuarios que únicamente tengan una tarjeta asociada, que a aquellos que tengan varias.

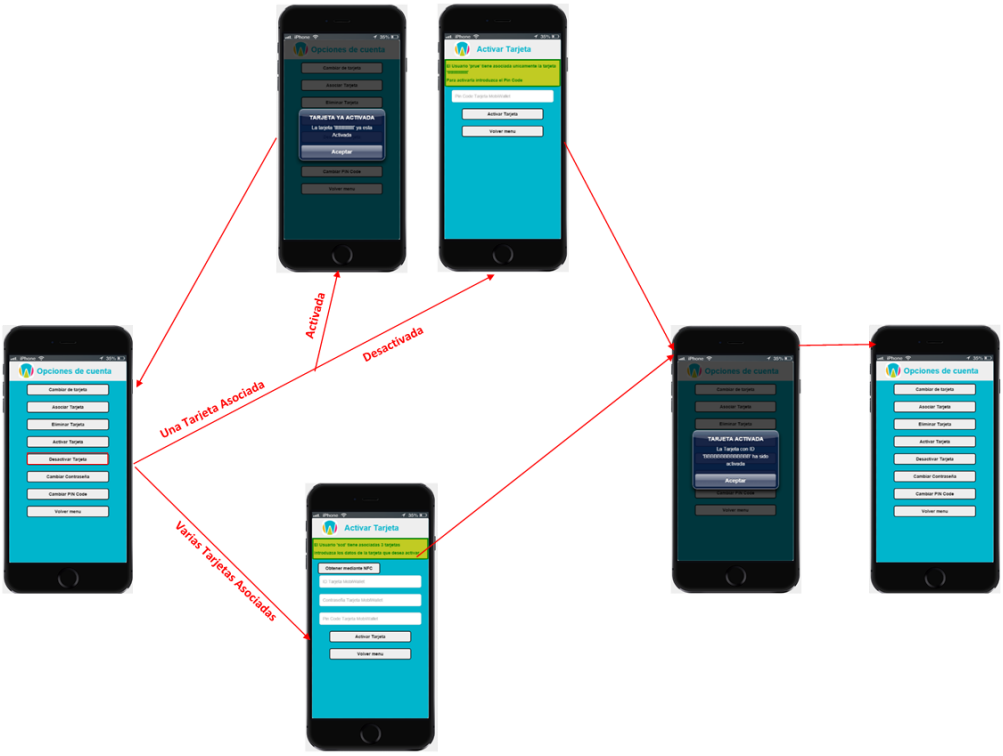

Imagen 28: Secuencia a seguir para Activar tarjeta

El funcionamiento para activar/desactivar tarjetas es el mismo, la única diferencia es que en la petición que se envía al Web service cambia uno de los parámetros (en caso de valor 1 se trata de una activación, y 0 una desactivación). Puesto que en la imagen superior se representa el proceso a seguir para realizar una activación, será este el que se explique, pero en el caso de la desactivación seria exactamente igual.

Al pulsar el botón "Activar Tarjeta" se envía una petición "RequestMWUser" [61] consultando el número de tarjetas asociadas al usuario, y en caso de solo tener una, se puede saber si esta está activada:

- Si el usuario tiene asociada únicamente la tarjeta con la que ha iniciado sesión, se podrá obtener el estado de esta. Si esta activada se mostrará una alerta indicándolo, y se volverá al menú "Opciones de Cuenta", en caso contrario se accederá a la ventana "Activar Tarjeta". Para realizar la activación solo deberá introducir el Pin Code. Al pulsar el botón "Activar Tarjeta" se enviará una petición de tipo "ChangeMWstatus" [61], de forma que si el Pin Code es correcto se activará la tarjeta, regresando al menú "Opciones de Cuenta".
- Si por el contrario el usuario tiene asociadas varias tarjetas, se accederá directamente a la ventana "Activar Tarjeta". En este caso se deberán introducir los datos de la tarjeta que se desea activar (identificador, contraseña y Pin Code). Al pulsar el botón "Activar Tarjeta" se enviará una petición de tipo "ChangeMWstatus" [61], de forma que si los datos son correctos se activará la tarjeta, y se volverá a mostrar el menú "Opciones de Cuenta".

- **Cambiar contraseña y Pin Code**

  Por último se ofrece la opción de modificar la contraseña y el Pin Code de la tarjeta MobiWallet.

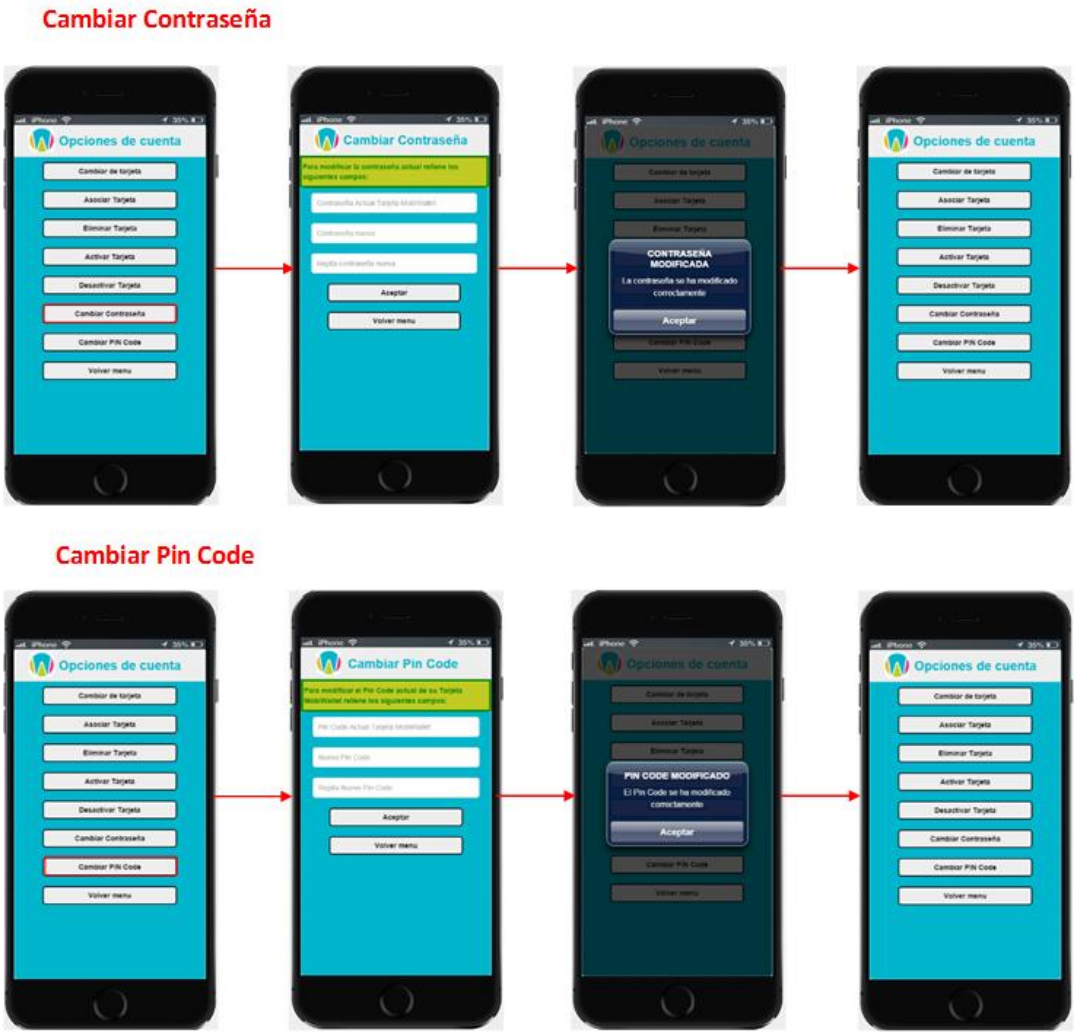

Imagen 29: Secuencia a seguir para modificar la contraseña y el Pin Code

El funcionamiento de ambas opciones es prácticamente el mismo, la única diferencia es la petición que se envía al Web service, en el caso de la modificación de contraseña es de tipo “changePass” [61] mientras que para la modificación del Pin Code es de tipo “changeCode” [61]. En ambos casos se solicita el valor actual, e introducir dos veces el nuevo valor. En caso de que las peticiones sean correctas, se mostrará una alerta notificándolo, y se regresará al menú “Opciones de Cuenta”

4.3.7 Consultar movimientos

Puesto que la aplicación va registrar pagos y recargas, en interesante proporcionar al usuario un método para poder visualizar los diferentes movimientos realizados. Esta funcionalidad se ha implementado de forma que permita al usuario filtrar por fecha y por tipo de movimiento (Recarga Física, Recarga Virtual, Compra de tickets, todos). Por defecto se mostrarán los movimientos de todo tipo realizados en el último mes.

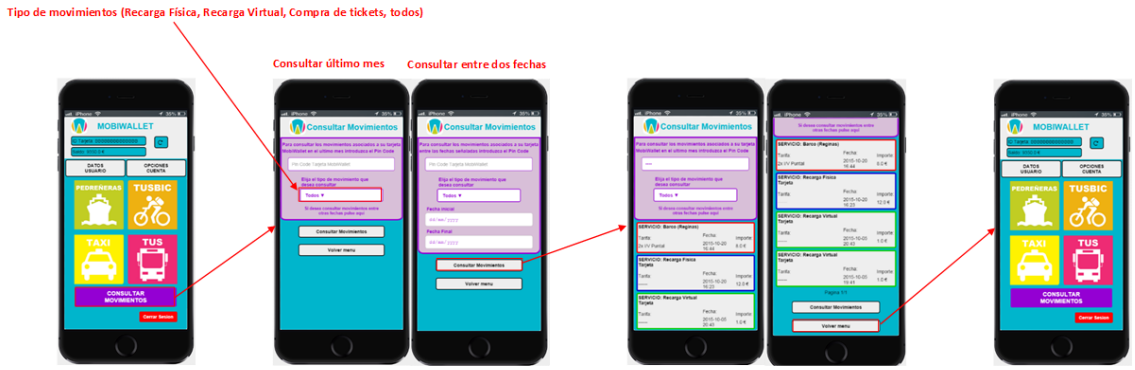

Imagen 30: Secuencia a seguir para consultar movimientos

El funcionamiento de esta opción es el siguiente:

Al pulsar el botón “Consultar Movimientos” se accederá a la ventana correspondiente. Para poder consultar los movimientos es necesario introducir el Pin Code de la tarjeta con la que se ha iniciado la sesión. Adicionalmente el usuario podrá establecer las fechas entre las que se mostrarán los movimientos, dado que al pulsar sobre el texto “Si desea consultar movimientos entre otras fechas pulse aquí”, se mostrarán dos campos ocultos donde introducir las fechas de inicio y fin de la consulta. También podrá seleccionar en un menú desplegable el tipo de movimiento que desea consultar.

Para visualizar los movimientos se pulsará sobre el botón “Consultar Movimientos” de forma que se enviará la plantilla “movements” [61]. La respuesta a esta plantilla contendrá los diferentes movimientos consultados, que se mostrarán en la misma ventana. De cada movimiento, se mostrará la fecha, el servicio, el importe y en el caso de las compras se incluirá la cantidad y el tipo de tickets. Además se diferenciará entre Compras, Recarga Físicas y Recargas virtuales, por el color asociado al movimiento (Rojo, azul y verde respectivamente). Para volver al menú principal se pulsará el botón “volver menú”

4.3.8 Servicios ofrecidos por los operadores

Por último queda describir las diferentes funcionalidades ofrecidas por cada operador. En este aspecto hay menos contenido desarrollado en el Web Service, por lo que la aplicación está limitada, aunque sí que se ha implementado alguna función que resultaba interesante, pese a que no se pueda comunicar con el Web Service (estas funciones se explicaran en el apartado 4.5).

En el menú principal se pueden observar 4 grandes iconos que se corresponden con autobús urbano, alquiler de bicicletas, taxis y pedreñeras.

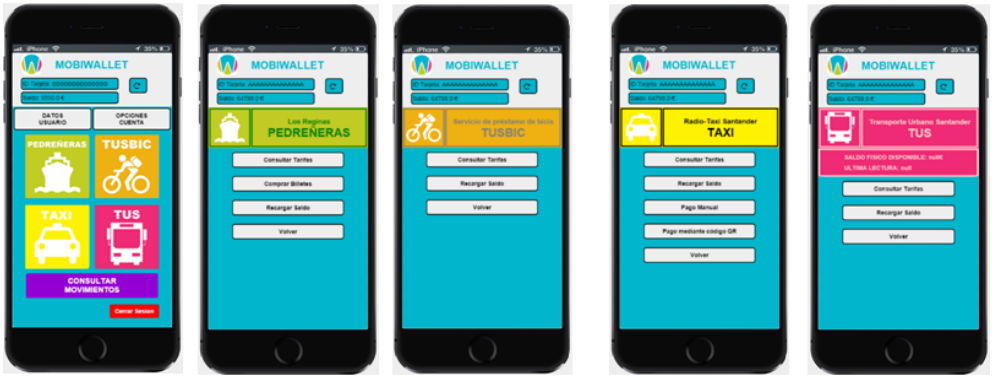

Imagen 31: Menú principal y submenú de cada uno de los operadores

Al pulsar en cada uno de estos iconos se accederá a un menú propio de cada servicio. Se puede observar que todos los operadores tienen opciones en común en sus menús, como la recarga de saldo y la consulta de tarifas. En este apartado describiremos el funcionamiento de las funciones en común de los operadores, describiéndose en los apartados 4.4 y 4.5 el resto.

- **Consultar Tarifas**

  Es interesante ofrecer al usuario la posibilidad de consultar las diferentes tarifas ofrecidas por cada operador. Estas tarifas están almacenadas en la base de datos de MobiWallet en vez de estar almacenadas en el propio dispositivo, permitiendo a los operadores poder modificar las tarifas en cualquier momento, sin necesidad de que la aplicación tenga que actualizarse.

Imagen 32: Secuencia a seguir para consultar tarifas

Puesto que las tarifas están almacenadas en la base de datos, al pulsar en el botón “consultar tarifas” se envía una petición al Web service de tipo “tariffs” [61] y se accede al menú “Consultar tarifas”. La respuesta del servidor contendrá las diferentes tarifas del operador consultado, y se visualizaran en esta misma ventana. De cada tarifa, se mostrará el nombre, una descripción y el precio. Por último, parar volver al menú del operador se pulsará el botón “volver menú”

- **Recargar Saldo**

Las operaciones de recarga de saldo son fundamentales para esta aplicación, ya que sin saldo, no se puede realizar pagos en los diferentes servicios, y por tanto la aplicación no tendría sentido. Dentro de las operaciones de recarga, tenemos dos comunes a todos los operadores: recargar saldo virtual y eliminar datos de tarjeta de crédito, y dos que únicamente se implementan para el servicio de autobuses urbanos: recargar saldo físico y transferir saldo virtual a físico.

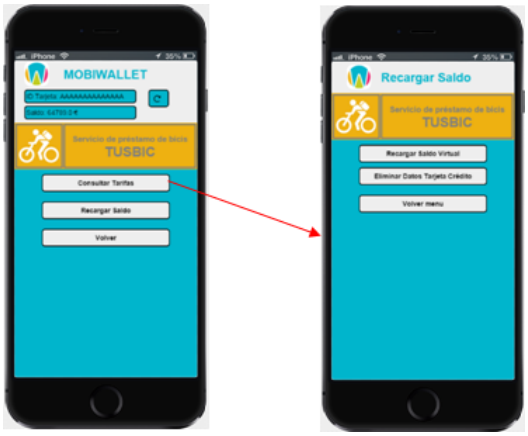

Imagen 33: Acceso al menú "Recargar Saldo"

Este apartado únicamente se centrará en explicar el funcionamiento de las operaciones comunes para todos los operadores:

- Recarga de saldo virtual

  Esta función permite recargar el saldo virtual asociado a la tarjeta MobiWallet del usuario. A la hora de realizar recargas entra en juego un factor que no se había tenido en cuenta en ninguna otra funcionalidad, la pasarela de pago o TPV. De forma que la aplicación a parte de comunicarse con el Web Service, lo hará también con el TPV.

  Destacar que existen dos formas de realizar la recarga.

  - Una de ellas es la que lleva a cabo al menos la primera vez que se desea recargar saldo.

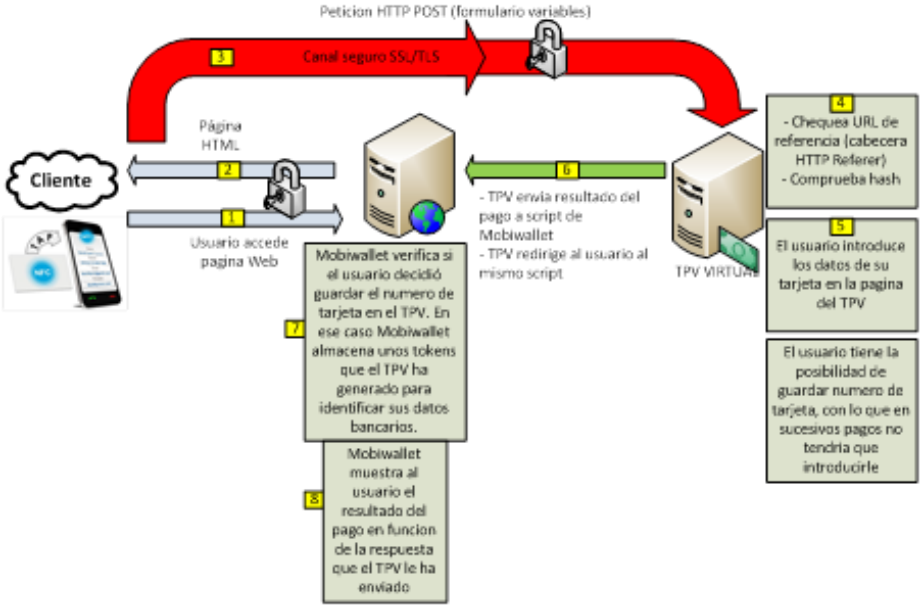

Imagen 34: Explicación detallada del proceso de conexión con el TPV al realizar recargas cuando los detalles de tarjeta no han sido almacenados

El usuario introduce el importe a recargar y el Pin Code de la Tarjeta MobiWallet con la que ha iniciado sesión. Al pulsar en el botón “confirmar recarga” se envía la petición de tipo “PaymentGW” [61].

Si los datos enviados son correctos, el Web service incluirá en su plantilla de respuesta una URL. Esta URL corresponde a una página de MobiWallet que actúa como pasarela (URL de referencia) para el TPV. Es decir el TPV solo recibirá peticiones HTTP que provengan de esta URL. En esta página se muestra el importe a recargar y el identificador de recarga, además sirve para incluir los términos y condiciones de uso, y se solicita al usuario que los acepte para continuar. En el momento que se pulsa el botón “continuar con el pago” se accede a una página de adquisición de datos de la tarjeta de crédito propia del TPV. Es decir a partir de este momento el usuario dialoga directamente con el TPV virtual, sin intervención de MobiWallet. Una vez introducidos los datos de la tarjeta de crédito, y si estos son correctos, el TPV responde al usuario. Si el usuario selecciona la opción “Guardar detalles de tarjeta” del TPV Virtual, éste almacenará dichos datos y entonces se podrá efectuar la segunda forma de recarga.

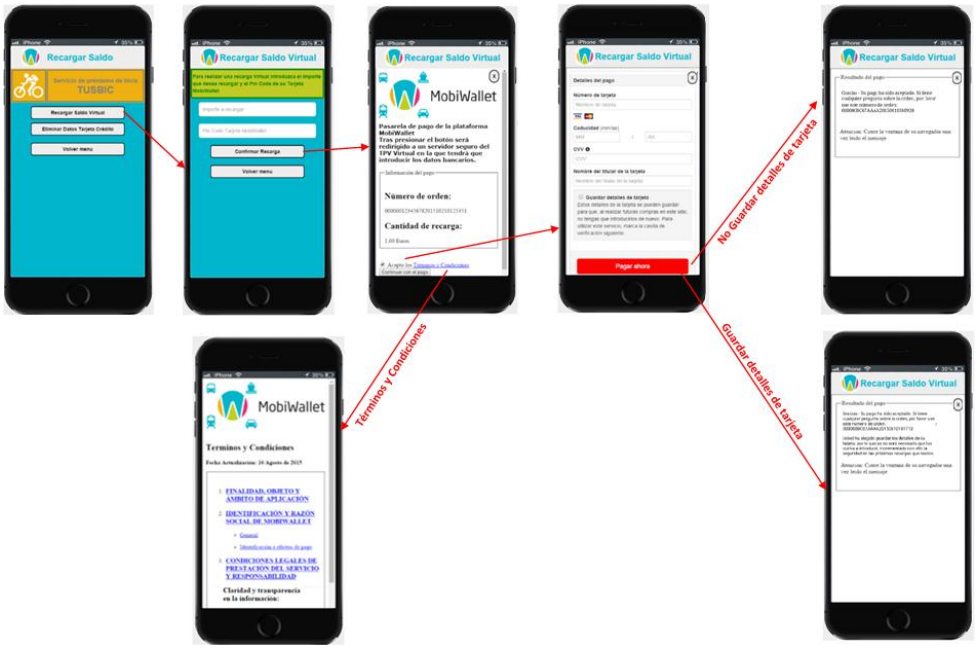

Imagen 35: Secuencia a seguir para realizar recargas cuando los detalles de tarjeta no han sido almacenados

- La segunda forma de realizar la recarga estará disponible, a partir de que en la primera forma, el usuario selecciona la opción “Guardar detalles de tarjeta” del TPV Virtual. Para conocer si el usuario tiene activada la opción “Guardar detalles de tarjeta”, basta consultar el campo MW_CARD_CC (True=activada, False=desactivada) de la repuesta del Web service a la petición “RequestMWUser” [61]. Dicha petición se enviará siempre que se pulse el botón “Recargar saldo Virtual”.

  Si el valor de MW_CARD_CC es “False” se seguirá la primera forma de descarga descrita. En caso contrario se solicita al usuario el importe a recargar, el Pin Code de la Tarjeta MobiWallet con la que ha iniciado sesión y el código de seguridad de la tarjeta de crédito (CVN).

Al pulsar en el botón “confirmar recarga” se envía la petición de tipo “PaymentGW” [61]. Si los datos enviados son correctos, el Web service retransmite de forma transparente la petición de pago hasta el TPV virtual, y devuelve los detalles de la operación.

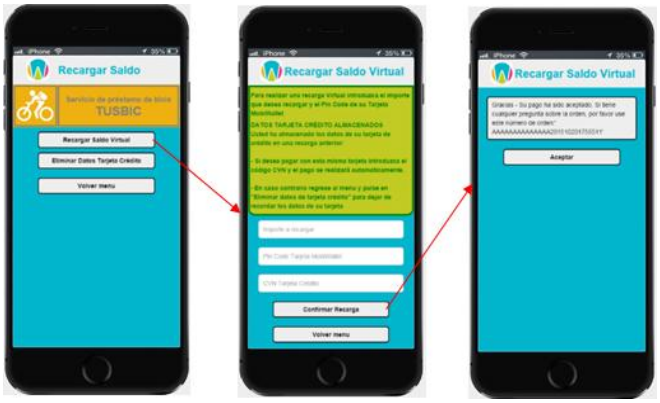

Imagen 36: Secuencia a seguir para realizar recargas cuando los detalles de tarjeta han sido almacenados

- Eliminar datos bancarios

Debido a la opción “Guardar detalles de tarjeta” ofrecida en el TPV Virtual se ha tenido que desarrollar una función adicional que permita olvidar los datos de tarjeta de crédito. De esta manera se permite al usuario seleccionar una tarjeta de crédito diferente a la hubiera utilizado en ocasiones anteriores.

Para eliminar los datos bancarios asociados a la tarjeta MobiWallet con la que se ha iniciado sesión, únicamente es necesario introducir el Pin Code de esta, Al pulsar en el botón “Eliminar datos”, se enviará la plantilla “deleteCCard” [61]. Si los datos enviados son correctos, pueden suceder dos cosas, que la tarjeta MobiWallet no tuviera datos bancarios asociados, o que si los tuviera y que hayan sido eliminados. Ambos caso se indican mediante una alerta.

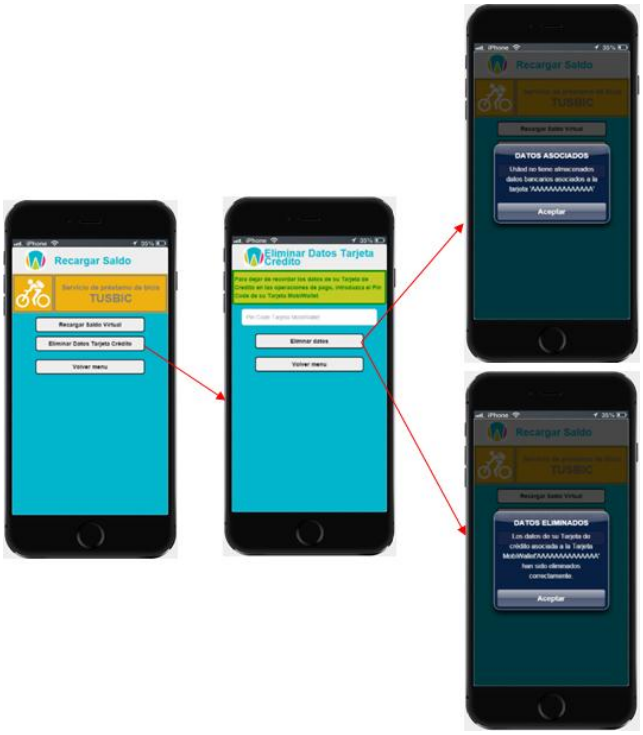

Imagen 37: Secuencia a seguir para eliminar datos bancarios almacenados

4.4 Uso tecnología NFC

Cuando se analizaron los tipos de aplicaciones comparándose sus prestaciones en el apartado 2.3, se concluyó que el desarrollar una aplicación de forma híbrida, era la opción más recomendable. Pero la menor capacidad de manejo del Hardware del dispositivo, que en las aplicaciones nativas, era uno de sus inconvenientes. El principal Hardware que necesita manejar la aplicación en este proyecto es el lector NFC. Por tanto en este apartado se estudia si es posible manejar este dispositivo en aplicaciones híbridas.

Recordemos que la necesidad de establecer un saldo físico al que se accede mediante tecnología NFC, se debe a que las ticketadoras que van a bordo de los autobuses, no disponen de conexión a internet, por lo que no se puede hacer uso del saldo virtual almacenado en la base de datos de MobiWallet.

Analizando los plugins ofrecidos por el entorno de desarrollo utilizado, ninguno servía para manejar el dispositivo NFC. Investigando un poco más, se descubrió que era posible incorporar plugins de terceros en Intel XDK. Por tanto se buscó información acerca de plugins que permitieran el manejo del dispositivo NFC en aplicaciones híbridas, y el único resultado que se obtuvo fue “PhoneGap NFC Plugin” que permite utilizar de ciertas formas el dispositivo NFC en dispositivos con sistema operativos Android, Windows Phone 8 y BlackBerry 10. [62]

Tras realizar varias pruebas, se logró leer el identificador de tarjeta MobiWallet mediante el dispositivo NFC, para ello le código empleado es el siguiente:

```
function init_AT() {
    document.addEventListener('deviceready_AT', enabled_AT, false);
}

function enabled_AT() {
    function onNfc_AT(nfcEvent) {
            // Se lee el taga de la tarjeta
            var tag = nfcEvent.tag;
            var tagId = nfc.bytesToHexString(tag.id);
            tagId = "000000"+tagId;
            // RESTO DE FUNCIONALIDAD QUE SE DESEE IMLEMENTAR//
    }
    function win_AT() {
        console.log("Listening for NFC Tags");
    }
    function fail_AT(error) {
        alert("Error adding NFC listener");
    }
    nfc.addTagDiscoveredListener(onNfc_AT, win_AT, fail_AT);
}
```

Imagen 38 : Código empleado para leer el Tag de las tarjetas MobiWallet mediante la tecnología NFC

Pero tras mucho investigar, no se encontró ningún método que incluyera las claves maestras necesarias para leer y escribir el saldo en las tarjetas Mobiwallet (tipo Mifarer Classic). Como última opción se decidió buscar ayuda dentro del grupo de ingeniería telemática en el que se desarrolla este proyecto. Concretamente se preguntó a Jorge Lanza Calderón, experto en tarjetas inteligentes, y tras analizar la situación, se llegó a la conclusión de que no existe actualmente ningún plugin para javascript que ofrezca un método o función que permitia manejar el chip NFC de la forma que exige MobiWallet. La única solución posible sería crear un nuevo plugin, pero se aleja de los objetivos del proyecto. Por tanto el empleo de esta tecnologia se intentará incorporar en futuras versiones de la aplicación.

El no poder utilizar la tecnología NFC, no es del todo crítico para el desarrollo del proyecto, puesto que hay una gran cantidad de dispositivos móviles que ni siquiera poseen esta tecnología. Para solucionar este problema MobiWallet repartirá por la ciudad unos Totem de recarga para que los usuarios puedan realizar recargas físicas en sus tarjetas.

Pese a no poder ofrecer un funcionamiento “real”, puesto que la tecnología actual no permite el control total del dispositivo NFC se optó por implementar aquellas funcionalidades que emplean el dispositivo NFC. Concretamente dos, la recarga de saldo Físico y el traspaso de saldo virtual a físico. Lo que se hace realmente es aprovechar la capacidad de leer el identificador de tarjeta MobiWallet mediante el dispositivo NFC, y simular la lectura/escritura del chip NFC en el almacenamiento interno del dispositivo. Esta solución no es muy sofisticada, puesto que el saldo físico va residir en el dispositivo, por lo que cuando se cambie de dispositivo, el saldo que se suponía que tenía la tarjeta será nulo. Pero hay que recordar que se trata de una solución provisional. Resaltar que en la aplicación oficial también están teniendo problemas a la hora de manejar el chip NFC y que en el Web Service no está implementada aun toda la funcionalidad de recarga de saldo físico.

Las funcionalidades incorporadas en la aplicación son las siguientes:

- Recarga de saldo físico.

  El proceso de recarga de saldo físico es exactamente igual que el de recarga de saldo virtual, la única diferencia existente, es que el campo MW_RECHARGE_TYPE de la petición “PaymentGW” [61] tiene valor 0 para las recargas físicas, y 1 para las virtuales.

- Traspaso de saldo virtual a físico

  Esta funcionalidad permite recargar el saldo físico, a partir del saldo virtual. Se diferencia de la recarga de saldo física, a parte de la procedencia del importe a recargar, en que únicamente es necesario introducir el importe a transferir, y el Pin Code de la tarjeta MobiWallet con la que se ha iniciado sesión, que será de donde se descuente el importe de la transferencia. Una vez introducidos estos datos, se pulsa el botón “Confirmar Transferencia” y se envía la plantilla de tipo “transfer” [61]. Si los datos enviados son correctos, el Web service responde con los detalles de la operación.

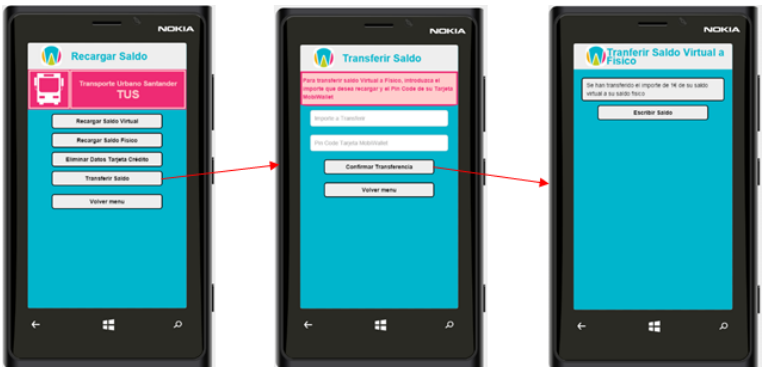

Imagen 39: Secuencia a seguir para realizar un traspaso de saldo virtual a físico

En ambos casos, una vez notificado el mensaje de que la recarga se ha realizado, se procede a escribir el importe recargado en el saldo físico de la tarjeta MobiWallet. Para ello se pulsa sobre el botón “Escribir Saldo”, a partir de este momento se dispone de un intervalo de tiempo para aproximar la tarjeta al dispositivo, en este caso se han fijado 8 segundos:

1. Si no se detecta ninguna tarjeta NFC, se muestra una ventana de alerta, que permite al usuario dos opciones:
   a. Volver a activar la lectura de tarjeta durante otros 8 segundos
   b. Cancelar el proceso de escritura. En este último caso, sería necesario almacenar en el Web service el importe pendiente de recargar, para que la próxima vez que se lea esta tarjeta se finalice el proceso de recarga. Puesto que esta funcionalidad no está implementada aun en el Web service, se simula el envío de información, y se almacena en la memoria del dispositivo

2. Si se detecta una tarjeta y su identificador coincide con el de la tarjeta con la que se ha iniciado sesión, se debería consultar en el Web service si para esta tarjeta existe alguna recarga pendiente. Pero puesto que esta función tampoco está implementada, se simula consultando la memoria del dispositivo donde estamos almacenando este tipo de información. En caso de existir una recarga pendiente, se suma el importe pendiente, al importe de la recarga actual y se escribiría en la tarjeta, en caso contrario únicamente se escribiría el importe de la recarga actual. Puesto que no es posible escribir en el chip de la tarjeta, debido a que la tecnología actual no lo permite, se simula esta escritura en la memoria local del dispositivo.
3. Si se detecta una tarjeta y su identificador no coincide con el de la tarjeta con la que se ha iniciado sesión, no se permitirá la escritura de saldo. En este caso se notificará error mediante una alerta y a continuación se reiniciará este proceso con otro intervalo de 8 segundos, para que se acerque la tarjeta correcta.

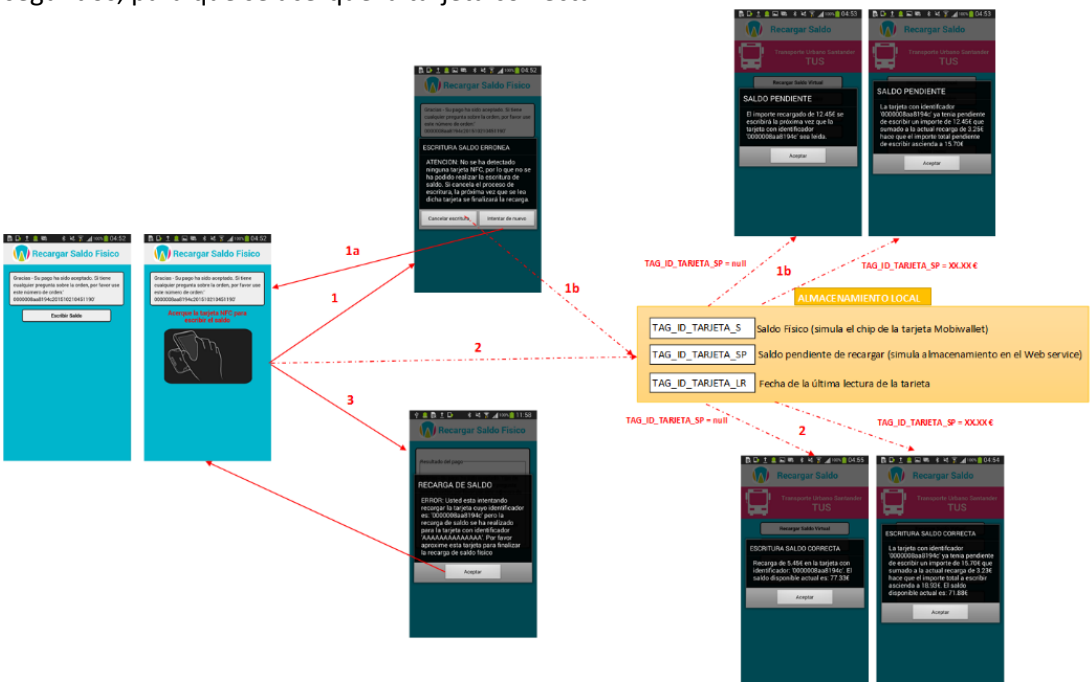

Imagen 40: Simulación del proceso de lectura/escritura de saldo físico en la tarjeta MobiWallet mediante la tencologia NFC

Por otro lado, se ha implementado en todas aquellas funciones que requieren la introducción del identificador de tarjeta, la posibilidad de hacerlo mediante la lectura con el chip NFC. Estas funciones son las siguientes: Registro tarjeta MobiWallet, Introducir tarjeta en el acceso a la aplicación, Recordar contraseña, Cambiar de tarjeta, Asociar tarjeta, Eliminar tarjeta, Activar/desactivar tarjeta. En la mayoría de estos caso, se consulta además si existe alguna recarga pendiente para esa tarjeta, de forma que si existe, se realiza. Al igual que ocurría en las funcionalidades descritas anteriormente, por no estar implementadas varias funciones en el Web service, y por el problema de no poder acceder al chip NFC de la tarjeta, estos procesos se simulan en el almacenamiento local del dispositivo.

Por último resaltar, que al acceder a la aplicación se comprueba si el dispositivo móvil posee tecnología NCF, de tal forma, que si no dispone de esta tecnología, todas las funcionalidades descritas en este apartado permanecen ocultas. Destacar también, que en este caso, las pruebas en el entorno de desarrollo se realizan en un Nokia Lumia 920 en vez de en un Iphone 6, esto se debe a que este plugin no permite el manejo del chip NFC en dispositivos iOS.

4.5 Funcionalidades desarrolladas en la aplicación, sin implementar aun en el web service

Como ya se ha citado en esta memoria, el desarrollo de la aplicación, se ha ido realizando paralelamente a la implementación del Web service. Esto ha supuesto, que solamente se puedan incluir en la aplicación aquellas funcionalidades que estén disponibles en la API de usuario del Web service, y actualmente existen las explicadas en los apartados anteriores.

Uno de los aspectos que contemplaba el proyecto MobiWallet, era la posibilidad de realizar compras y pagos de billetes desde la aplicación, pero en la API de usuario, aun no existen estas funcionalidades. Puesto que esta aplicación es un prototipo, se decidió incluir algunas de las funciones que se tiene previsto desarrollar en el Web service. Evidentemente su funcionamiento será nulo, ya que no existen las peticiones que se envían al Web service para que este pueda realizar los cargos oportunos en el saldo virtual del usuario almacenado en la base de datos. Las funcionalidades desarrolladas que permiten compras y pagos en la aplicación son las siguientes:

- Compra de billetes en el servicio pedreñeras

  El servicio de ferrys o pedreñeras, permite realizar compras de billetes desde la aplicación. Se deberá escoger el tipo de tarifa y la localización ofrecidos en dos menús desplegables, además se deberá introducir el número de billetes. Cuando se implemente la función en el Web service probablemente se requiera también el Pin Code de la tarjeta sobre la que se realizará el cargo.

Imagen 41: Secuencia a seguir para realizar compra de billetes en el servicio Pedreñeras (aun no implementado en el Web service)

Al pulsar sobre el botón “Aceptar” aparecerá un mensaje de alerta con los detalles de la compra, dando la posibilidad de cancelarla o seguir adelante. En el momento que se acepte el pago, será cuando se deba enviar la plantilla al Web service (cuando este servicio este implementado)

- Pago manual en el servicio de taxis

Esta funcionalidad permitirá a los usuarios del servicio taxi pagar las carreras desde su dispositivo, empleando el saldo virtual. El funcionamiento es sencillo, ya que únicamente se debe introducir el número de licencia del taxista y el importe de la carrera.

Al igual que en el caso anterior es posible que cuando se implemente la función en el Web service se requiera también el Pin code de la tarjeta sobre la que se realizará el cargo. Al pulsar sobre el botón "Aceptar" aparecerá un mensaje de alerta con los detalles de la compra, dando la posibilidad de cancelarla o seguir adelante. En el momento que se acepte el pago, será cuando se deba enviar la plantilla al Web service (cuando este servicio este implementado)

Imagen 42: Secuencia a seguir para realizar un pago manual en el servicio de Taxis (aun no implementado en el Web service)

- Pago mediante código QR en el servicio de taxis

En el caso de pago manual, es necesario, introducir el importe de la carrera y el número de licencia, y esto supone un riesgo, ya que pueden darse errores al insertar estos datos manualmente. Para evitar este problema, se diseña la funcionalidad de pago mediante código QR.

En este mecanismo los datos de la carrera y el taxista, se mostrarán automáticamente mediante un código QR en un dispositivo asociado al taxímetro. El usuario no tendrá más que acercar su móvil y leer dicho código para obtener el importe a pagar.

Será necesario establecer una estandarización en los códigos QR, de forma que todos se adapten a un mismo patrón. En este caso se ha decido que sigan la siguiente estructura:

Hora = hh:mm-DD-MM-AAAA

Numero Licencia = XXXXXXXX

Importe = XX.XX€

Imagen 43: Código QR de ejemplo y estructura de la información que contiene

Para poder emplear la cámara a la hora de leer códigos QR, es necesario incluir el plugin "Intel XDK Device" propio del entorno de desarrollo. La función que se ha implementado para leer códigos QR es la siguiente:

```
function leer_codigo_QR(){
    document.addEventListener("intel.xdk.device.barcode.scan",function(evt){
        intel.xdk.notification.beep(1);
        if (evt.success == true) {
            // Valor del texto contenido en el codigo QR
            var datos_QR = evt.codedata;

            // Codigo si la lectura es correcta
        }
        else {

            // Codigo si la lectura es incorrecta

        }
    },false);
    intel.xdk.device.scanBarcode();
}
```

Imagen 44: Código empleado para leer el contenido de un código QR

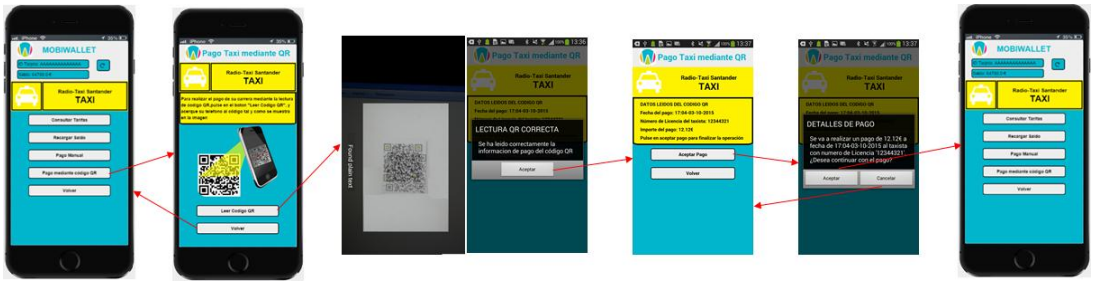

Imagen 45: Secuencia a seguir para realizar un pago mediante código QR en el servicio de Taxis (aun no implementado en el Web service)

Al pulsar el botón "Leer Código QR" se ejecuta la función de la imagen superior, de forma que se lanza la cámara del dispositivo para tratar de leer un Código QR. Independientemente de si la lectura es correcta o no, se mostrará una alerta notificándolo. En caso que la lectura sea errónea se vuelve a la página "Pago taxi mediante QR", por el contrario si la lectura es correcta, se mostrarán los datos de la carrera obtenidos de la lectura. El siguiente paso es pagar la carrera, para ello se pulsa el botón "Aceptar Pago", de forma que aparece un mensaje de alerta con los detalles de la compra, dando la posibilidad de cancelarla o seguir adelante. En el momento que se acepte el pago, será cuando se deba enviar la plantilla al Web service (cuando este servicio este implementado). Es posible que al igual que en las funcionalidades descritas anteriormente se requiera el Pin Code de la tarjeta sobre la que se realizará el cargo.

4.6 Simulación pasarela de pago

En este apartado se describirá como y porque se ha simulado la pasarela de pago o TPV. En primer lugar hay que aclarar que la pasarela de pago funciona correctamente, pero se decidió realizar una simulación por dos motivos provenientes de una misma causa. La causa es que se decidió poner a prueba la plataforma MobiWallet con usuarios y operadores reales, por lo que el TPV pasó de estar en versión de pruebas a operar en un entorno real. Esta transformación no afectó al correcto funcionamiento de la aplicación pero conllevo dos consecuencias:

- Cuando el TPV funcionaba en su versión de pruebas, el banco Santander proporcionaba tarjetas bancarias ficticias que permitían realizar pagos, pero en el momento que el TPV paso a funcionar en un entorno real, a la hora de realizar un pago desde la aplicación, se requerían datos bancarios reales, y por tanto se efectuarían cargos reales.

- Al habilitar la plataforma MobiWallet en un entorno real, los registros de usuario, contenían datos personales reales, por lo que por motivos de privacidad se decidió, que para el desarrollo de esta aplicación se habilitaría una base de datos de prueba diferente de la oficial. Esto supuso que al intentar acceder a la URL de la página pasarela de Mobiwallet, incluida en la respuesta devuelta por el Web service cuando se le envía una petición de pago “PaymentGW” [61], apareciera el siguiente error:

Imagen 46: Mensaje de error mostrado al acceder a la pasarela de pago

Este error se debe, a que cuando desde la aplicación se envía una petición de pago “PaymentGW” [61], el Web service almacena los datos del pago (Order Id e importe) en la base de datos de prueba, pero la página pasarela de MobiWallet hace uso de estos datos de pago y trata de buscarlos en la base de datos oficial, por lo que evidentemente no los encuentra y emite el error de la imagen superior.

Para evitar estos problemas y poder realizar pagos en la aplicación se realizó una simulación evitando la comunicación con el TPV. Recalcar de nuevo, que esta simulación se realiza por causas ajenas a la aplicación (privacidad de los datos de usuario, evitar pagos reales) y no por un funcionamiento incorrecto de esta.

Para lograr esta simulación, se determinó con los desarrolladores del Web service, modificar ligeramente la petición “PaymentGW” [61], incluyendo el campo MW_SIMULATION que indicará al Web service que se trata de un pago simulado cuando su valor sea 1, y por defecto será 0. El siguiente paso fue construirse en la propia aplicación cuatro páginas: La primera simulando el comportamiento de la página pasarela de MobiWallet, otra conteniendo los términos y condiciones de uso, la siguiente que simula la página de captación de datos bancarios del TPV y por último, otra que mostrará el mensaje de cómo ha finalizado el pago.

En la siguiente imagen se comparan las cuatro páginas simuladas frente a las reales:

Imagen 47: Comparación entre las páginas del entorno real y las simuladas, que intervienen al realizar un pago

Con esta simulación se consigue seguir manteniendo las dos formas de pago explicadas en el apartado 4.3.8:

- La primera de ellas es muy similar, en primer lugar se envía la petición "PaymentGW" [61] como si de un entorno real se tratara. Si la respuesta a esta petición es correcta, en vez de acceder a la URL de la página pasarela se accederá a su simulación, y tras aceptar los términos y condiciones, a la simulación de la página de captación de datos bancarios. Los valores introducidos en esta página se enviarán al Web service simulando la respuesta que enviaría el TPV en una situación real. La única diferencia, es que en este caso, al activar la función de "guardar detalles de tarjeta" lo único que supondrá es que el campo MW_CARD_CC se fije a uno, sin generarse, como en el caso real, los tokens, que permitirían la comunicación directa entre Web service y TPV. Por último el Web service responderá devolviendo a la aplicación el mensaje de cómo ha finalizado el pago.
- El segundo método es totalmente simulado, puesto que se envía una petición de pago "PaymentGW" [61] con el campo MW_SIMULATION a 1. Esto supone que el valor del CVN que introduzca el usuario no se tenga cuenta, ya que lo único que hará esta petición es comprobar que los datos de tarjeta MobiWallet y Pin Code son correctos, y en este caso, sumar al saldo virtual el importe indicado

4.7 Seguridad

En este apartado se comentará como se han afrontado los requisitos de seguridad propuestos en el punto 3.2.2 y los problemas que han surgido.

A la hora de empezar a desarrollar la aplicación, se buscó en primer lugar cumplir los requisitos funcionales, por lo que se diseñó toda la aplicación sin ningún tipo de seguridad. Es decir el intercambio de información con el Web service se realizaba a través de un puerto sin implementar ningún tipo de cifrado, por lo que se realizaban peticiones HTTP en abierto. El código empleado para realizar estas peticiones se muestra en la siguiente imagen:

```
//Plantilla xml
var sr = '<MW_USER_DATA_CHANGE>\r\n' +
         '<MW_VERSION>1.0</MW_VERSION>\r\n' +
         '<MW_USR_ID>' + MW_USR_ID + '</MW_USR_ID>\r\n' +
         '<MW_CARD_ID>' + MW_CARD_ID + '</MW_CARD_ID>\r\n' +
         '<MW_USR_OLD_PWD>' + MW_USR_PWD_O + '</MW_USR_OLD_PWD>\r\n' +
         '<MW_USR_NEW_PWD>' + MW_USR_PWD_N1 + '</MW_USR_NEW_PWD>\r\n' +
         '</MW_USER_DATA_CHANGE>';
// creamos la peticion http
peticionHttp = new XMLHttpRequest();
// Preparamos la funcion de respuesta
peticionHttp.onreadystatechange = muestraContenido;
// Realizamos la peticion HTTP
var url='http://XXX:XXX:XXX:XX:YY/Tlmat1Pruebas2Sergio3/rest/';
// Fijamos el tipo de peticion
var metodo='UserActions/changePass';
// Fijamos el metodo
peticionHttp.open("PUT", url+metodo, true);
// Definimos las cabeceras
peticionHttp.setRequestHeader('Data-Type','text/xml');
peticionHttp.setRequestHeader('Content-Type','application/xml');
// Se envia la plantilla xml
peticionHttp.send(sr);
// Obtenemos la respuesta que devuelve el servidor
function muestraContenido() {
    if(peticionHttp.readyState == 4) {
        // Si el status es 200 el xml se habra recibido correctamente en el servidor
        if(peticionHttp.status == 200) {
            //DESARROLLAMO FUNCIONALIDAD
        }
        // Si el status es 400 el formato del xml no sera el adecuado
        if(peticionHttp.status == 400) {
            //SE MUESTRAR EL ERROR
        }
        // Si el status es 403 el servidor este denegando el acceso a esa peticion
        if(peticionHttp.status == 403) {
            //SE MUESTRAR EL ERROR
        }
    }
}
```

Imagen 48: Código empleado para realizar una petición HTTP

Tras desarrollar la mayor parte de la funcionalidad de la aplicación se pasó a intentar hacer seguras las peticiones. Tal y como se especificó en el apartado 3.2.2, será necesario implementar el protocolo SSL de forma que se proporcione autenticación y privacidad de la información entre el Web service y la aplicación mediante el uso de criptografía. Se deberá desarrollar tanto SSL con autenticación del servidor como SSL con autenticación mutua.

Tras investigar varios métodos para desarrollar ambos protocolos mediante APIS o plugins de javascript volvieron a hacerse patentes las carencias de las aplicaciones híbridas puesto que es necesario descargarse e instalar los certificados en el “Truststore” del dispositivo, es decir, de nuevo se encuentra la limitación a la hora de manejar y acceder al Hardware del dispositivo.

Esta limitación sí que resultaba crítica, porque no poder encriptar los datos suponía que información delicada viajara en abierto entre los extremos de la comunicación.

Por este motivo se llegó al punto de plantearse abandonar el proyecto, pero tras mucho investigar se consiguió una API que permitía implementar al menos, el protocolo SSL con autenticación del servidor. Se trata de la API "API Security" ofrecida por el entorno de desarrollo Intel XDK. Esta API cuenta con los siguientes métodos:

API:

- open
  This API creates a new instance of secure transport with the provided parameters (indicated by url, method, serverKey and timeout).
- setURL
  This API modifies the URL (indicated by url) of a secure transport instance (indicated by instanceID).
- setMethod
  This API modifies the HTTP method (indicated by method) of a secure transport instance (indicated by instanceID).
- setHeaderValue
  This API adds/modifies a value (indicated by value) of a header key (indicated by key) of a secure transport instance (indicated by instanceID).
- sendRequest
  This API sends a request (indicated by requestBody) in the provided format (indicated by requestFormat) while embedding secure data elements into the sent request body defined within the descriptors (indicated by the secureDataDescriptors) using the secure transport instance (indicated by instanceID). The server response is received in the `success` callback.
- destroy
  This API releases the secure transport instance (indicated by instanceID).

Imagen 49: Métodos que ofrece la "API security" para realizar peticiones HTTPS [63]

Tras investigar y realizar múltiples pruebas, sin obtener resultados satisfactorios, se creó un post en el foro de desarrolladores de Intel preguntando acerca del funcionamiento de esta API. Después de varios intercambios de mensajes se llegó a una estructura que empleando varios métodos de los proporcionados por la API, permitía realizar peticiones HTTPS con autenticación del servidor. La estructura común en todas las funciones es la siguiente:

```
var options = { url: 'https://XXX:XXX:XXX:XX:YYYY/Tlmat1Pruebas2Sergio3/rest/UserActions/tariffs', 'method':'POST', serverKey: "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4Na+wc+lWweKrvDCdWJe\nzIX0Z/wgmBltmerIdMUpxiOIPpAzmzixv6mkIGiqrCRHWPeQZ2tgZfzq9XyJc1Pb\nsOtEqJIT+5AsU
NgBbtGhyo241PQjrDynekUQkVv1\nsMs1tE6GdMm1alkji5FhgheN\n0+QcaB5uaEjeQf61PZTAUgogHj+ou2YzoJbNptMYeN4n4gIrNiDmBNL9TgsXZkVX\niCjnUE/iMaq8Ps3r67OQycM4RE
Xywq5jaw5uK5MXO2hSh0Ooliwyv2IKLqJlEoMj\nq2ik9BVmrlRNj8V3Y4iCsxMtqSqI8LZtxpI4JuhLQws7h+1p37DgfyAHQdZwT4be\nYwIDAQAB\n-----END PUBLIC KEY-----" };
intel.security.secureTransport.open(
    function (instanceID) {
        var myInstanceID=instanceID;
        // Define the head Content-type
        intel.security.secureTransport.setHeaderValue(
            function(){
                intel.security.secureTransport.setHeaderValue(
                    function(){
                        // Send the request
                        var request_body ='<MW_USER_TARIFFS_QUERY>\r\n' +
                            '<MW_VERSION>1.0</MW_VERSION>\r\n' +
                            '<MW_USR_ID>' + MW_USR_ID_T +'</MW_USR_ID>\r\n' +
                            '<MW_USR_PWD>' + MW_USR_PWD_T + '</MW_USR_PWD>\r\n' +
                            '<MW_CARD_ID>' + MW_CARD_ID_T + '</MW_CARD_ID>\r\n' +
                            '<MW_OPERATOR_CODE>' + seleccion + '</MW_OPERATOR_CODE>\r\n' +
                            '<MW_LASTUSERUPDATE>2013-02-04</MW_LASTUSERUPDATE>\r\n' +
                            '</MW_USER_TARIFFS_QUERY>';
                        intel.security.secureTransport.sendRequest(
                            function(response){
                                //Creamos el objeto de tipo documento XML
                                var xml = response.responseBody
                                var parser = new DOMParser();
                                documentoXml = parser.parseFromString( xml, "text/xml" );
                            },
                            function(errorObj){
                                alert('Failed in sendRequest, code = '+errorObj.code+', message = '+errorObj.message);
                            },
                            {'instanceID':myInstanceID, 'requestBody':request_body }
                        );
                    },
                    function(errorObj){
                        alert('Failed in setHeaderValue, code = '+errorObj.code+', message = '+errorObj.message);
                    },
                    {'instanceID':myInstanceID, 'key':'Content-Type', 'value':'application/xml'}
                );
            },
            function(errorObj){
                alert('Failed in setHeaderValue, code = '+errorObj.code+', message = '+errorObj.message);
            },
            {'instanceID':myInstanceID, 'key':'Data-Type', 'value':'text/xml'}
        );
    },
    function (errorObj) {
        alert('Failed in open, code = '+errorObj.code+', message = '+errorObj.message);
    },
    options
);
```

Imagen 50: Código empleado para realizar una petición HTTPS

- En primer lugar (color rojo), se emplea el método intel.security.secureTransport.open(success, fail, options);
  Este método crea una instancia para realizar una petición HTTPS. Se incluyen en "options" los siguientes parámetros:
  - URL: En cada petición contendrá una dirección concreta
  - Method: Permite escoger el método que se empleara en la peticion HTTP
  - Server Key: Certificado del servidor en formato PEM

- A continuación (colores verde claro, y verde oscuro), se emplea dos veces el método intel.security.secureTransport.setHeaderValue(success, fail, options);
  Este método permite modificar las cabeceras de la peticion HTTPS. Se incluyen en "options" los siguientes parámetros:
  - InstanceID: nombre de la instancia creada en el método anterior
  - key: Clave de la cabecera (Data-Type , Content-Type)
  - value: Valor de la cabecera(text/xml, application/xml)

- Por ultimo (color azul), se emplea el método intel.security.secureTransport.sendRequest(success, fail, options);
  Este método establece una conexión segura y envía la petición. Se incluyen en "options" los siguientes parámetros:
  - InstanceID: nombre de la instancia creada en el método anterior
  - RequestBody: plantilla xml que se enviará

  Se puede observar que la respuesta se recoge mediante la sentencia "response.responseBody" y se parsea a xml.

Para finalizar este apartado, comentar que tras conseguir implementar peticiones HTTPS con autenticación del servidor, se preguntó a los desarrolladores de Intel si era posible desarrollar el protocolo de autenticación mutua, su respuesta fue la siguiente

Imagen 51: Repuesta de un desarrollador de Intel en este mismo foro [64]

Por tanto, es cuestión de tiempo que se desarrolle un plugin que permita realizar la autenticación mutua.

5 APLICACIÓN Y VERIFICACIÓN

En este apartado se describirá el procedimiento empleado para testear el correcto funcionamiento de la aplicación desarrollada. Destacar que no se ha empleado ningún software específico de testeo, sino que básicamente se han utilizado estas tres herramientas:

- SoapUI

Se trata de una herramienta que hace de cliente para conexiones a Web services, de forma que permite simular el comportamiento que debería tener la aplicación. Por tanto antes de implementar una función en la aplicación, se han realizado pruebas comprobando toda la casuística de respuestas por parte del Web Service.

Este testeo puede parecer innecesario puesto que actualmente en la API de usuario, se contemplan todas las posibles respuestas de cada petición. Pero hay que resaltar que el desarrollo de la aplicación ha sido paralelo al del Web service, y por lo tanto al de la API de usuario, de forma que mediante estas pruebas se ha obtenido alguna casuística de respuesta que no había sido tenida en cuenta pudiéndose corregir.

El manejo de esta herramienta es muy sencillo, únicamente hay que configurar lo siguientes parámetros:

- El método HTTP (POST o PUT) dependiendo de la petición a realizar
- La URL completa de acceso al recurso, por ejemplo
- Configurar la cabecera Content-Type de HTTP para ello fijar el campo Media type al valor *application/xml*, puesto que siempre se realizará la comunicación mediante el intercambio de documentos XML
- Por último definir el xml que contiene la información de la petición

Una vez definidos estos parámetros la petición se envía, y se obtiene la respuesta del servidor. Por un lado se obtendrán las cabeceras de la respuesta HTTP del servidor, y por otro el cuerpo del mensaje HTTP, en el que esta encapsulado el XML que el cliente leerá y parseará.

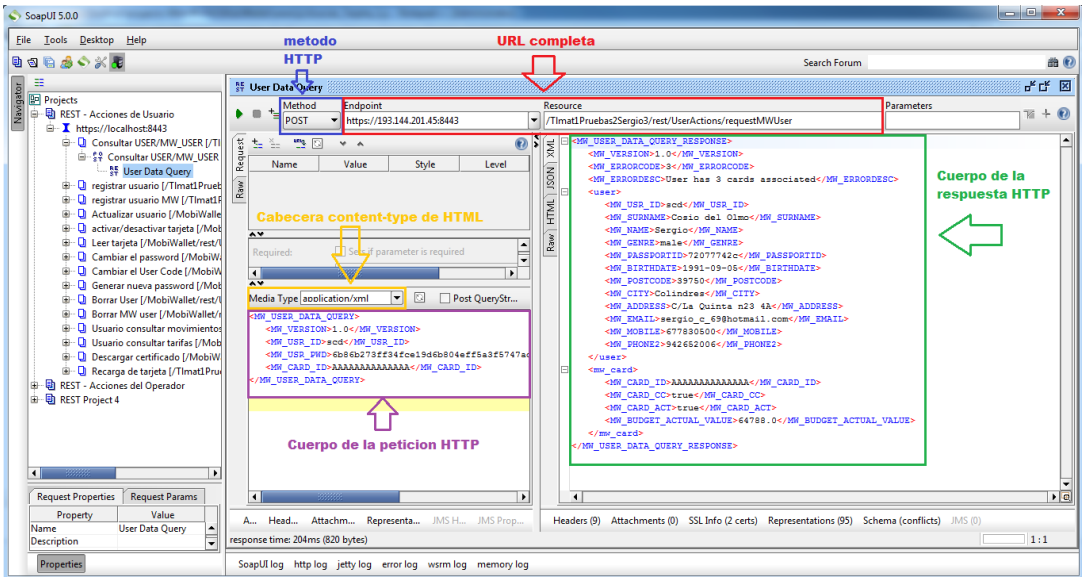

Imagen 52: Aspecto de la herramienta SoapUI detallando sus diferentes partes

- Intel XDK

Tal y como se explicó en el apartado 4.1, el entorno de desarrollo Intel XDK, aparte de permitir la implementación de la aplicación, también ofrece varias herramientas de simulación y testeo. La más usada ha sido la herramienta de simulación, para comprobar visualmente si el aspecto de la aplicación era el deseado, para asegurarse que el intercambio de datos con el Web service se realizaba correctamente, y por último, para visualizar el valor de las variables que se almacenan en el dispositivo.

En lo que respecta a las comprobaciones del aspecto de la aplicación, las pruebas han consistido en modificar el dispositivo con el que se simula, de esta forma se ha determinado si la aplicación se adapta adecuadamente a la pantalla del dispositivo y se comprueba si realmente funciona el framewok "App Framework", de forma que la aplicación se adapta a la interfaz del sistema operativo. En este aspecto la aplicación se ha simulado en varios dispositivos, tanto teléfonos móviles (iPhone 4s, iPhone 6, HTC Droid Incredible, Samsung Galaxy S, Nokia Lumia 920) como tablets (iPad, Samsung Galaxy tab, Microsoft Surface Pro)

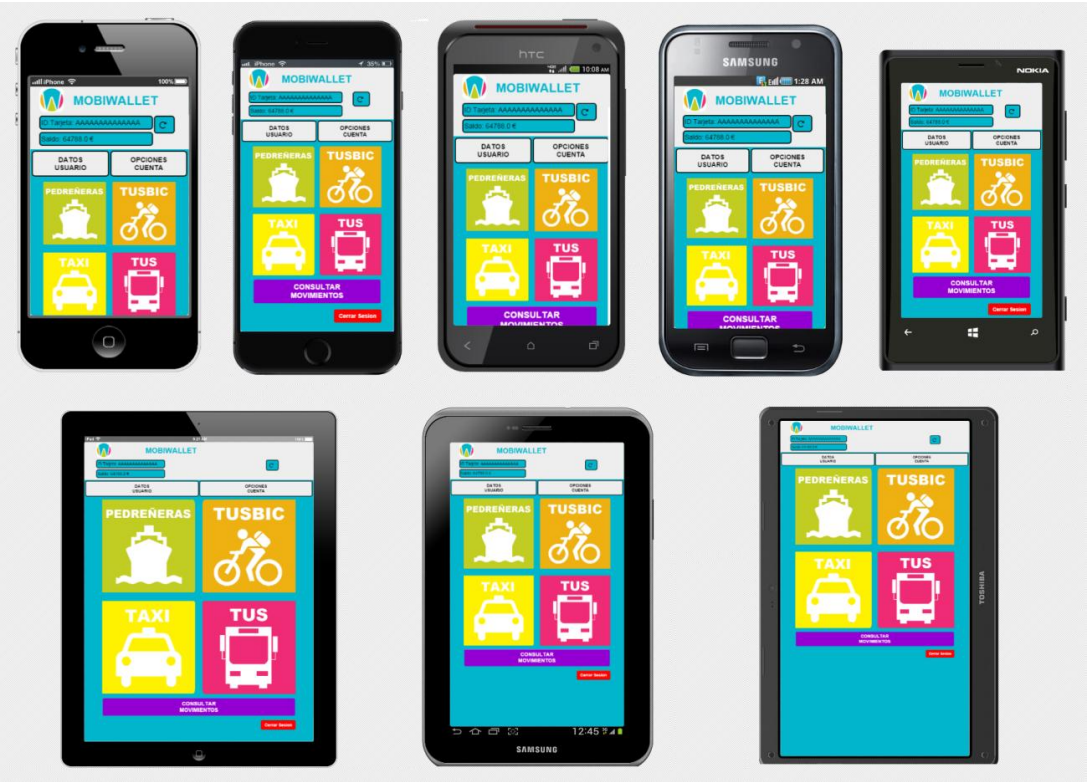

Imagen 53: Simulación de la aplicación en diferentes dispositivos móviles

Por otro lado para comprobar el proceso de comunicación con el Web service se emplea la utilidad debugger dentro de la simulación. De esta forma podemos observar la petición HTML enviada y la respuesta recibida al interactuar en la aplicación.

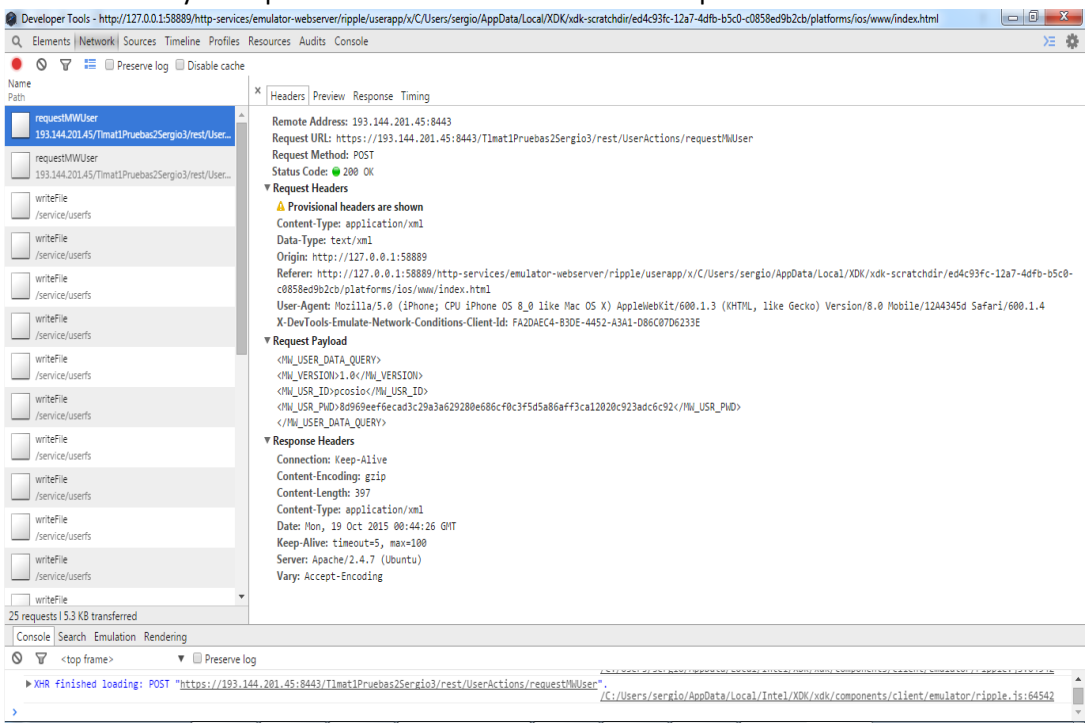

Imagen 54: Aspecto de la pestaña "Network" en herramienta de debug ofrecida en el entorno de desarrollo Intel XDK

Por último, con esta herramienta de debug también es posible visualizar el valor de las variables almacenadas en la memoria del dispositivo. En el caso de la aplicación desarrollada, los únicos valores almacenados, son el Identificador de usuario, la contraseña y el identificador de tarjeta, cuando un usuario decide almacenar sus datos como predeterminados; y por otro lado el identificador de usuario y la contraseña, cuando un usuario decide recordar sus datos de acceso.

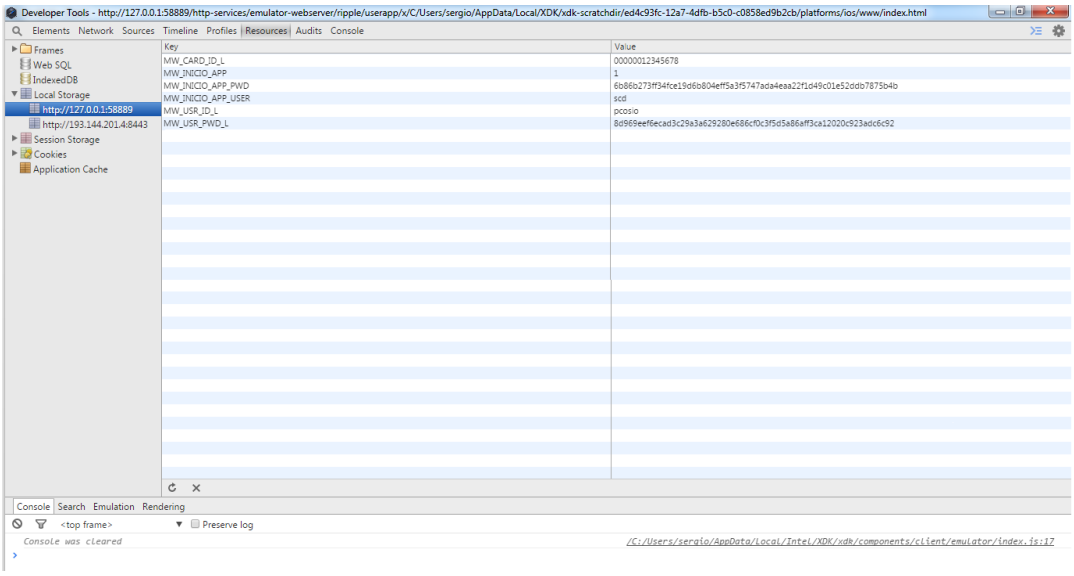

Imagen 55: Aspecto de la pestaña "Resources" en herramienta de debug ofrecida en el entorno de desarrollo Intel XDK

- Heidi SQL

Se trata de un programa que ofrece una interfaz amigable para administrar bases de datos MySQL. Esta herramienta más que para testear el funcionamiento de la aplicación en sí, se ha empleado para poder tener acceso a los datos de la plataforma MobiWallet. Por tanto realmente no ha sido una herramienta de testeo, aunque sí es cierto, que en muchos de los envíos de peticiones a parte de analizar la respuesta del web service, también se contemplaban los cambios en la base de datos. Por ejemplo al enviar una petición de modificación de los datos de usuarios, aparte de asegurar que la respuesta del Web service era correcta, se observaba si se producían estas modificaciones en la base de datos.

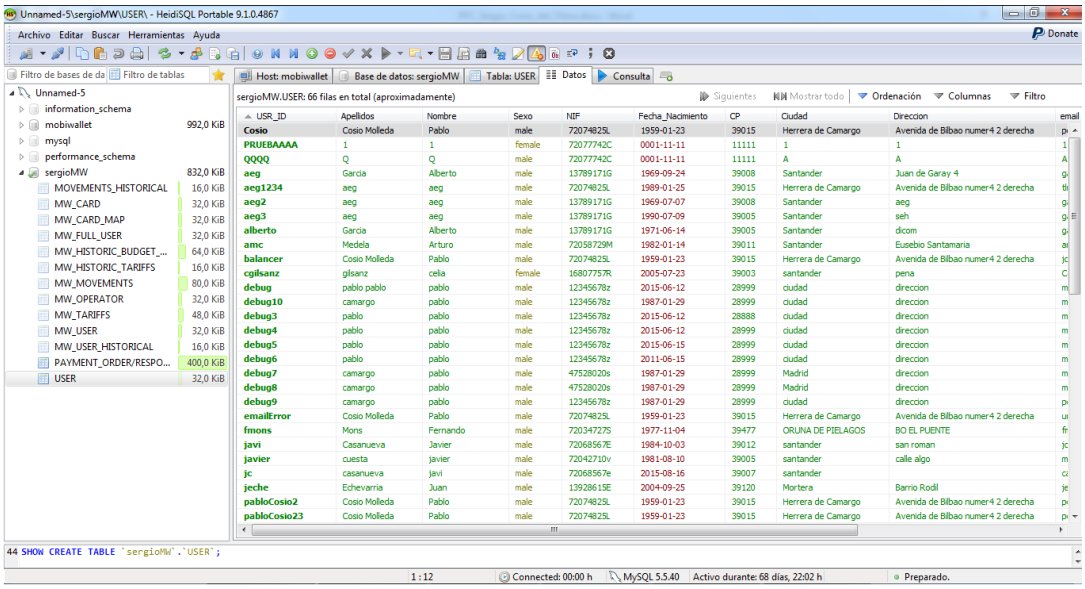

Imagen 56: Aspecto de la herramienta "Heidi SQL" empleada para visualizar la base de datos de MobiWallet

6 CONCLUSIONES Y LINEAS FUTURAS

Una vez finalizado el proyecto y logrados los objetivos planteados al inicio del mismo, en este apartado se presentan las conclusiones obtenidas de su desarrollo, así como las posibles líneas futuras a seguir para su mejora.

6.1 Conclusiones

La realización de este proyecto ha servido para conocer el mundo de las aplicaciones desde el punto de vista de desarrollador. Se han estudiado las tres tecnologías de implementación de aplicaciones existentes en la actualidad, y se ha podido comprobar las limitaciones de cada tipo, especialmente las restricciones en el control del hardware de los dispositivos en aquellas aplicaciones que no se desarrollan de forma nativa.

Pese a los problemas encontrados, se sigue defendiendo que la idea de desarrollar la aplicación de forma híbrida es la más viable, principalmente por los constantes intercambios de información que se producen entre esta y el Web service de MobiWallet. Otro de los puntos a favor a la hora de seguir apostando por el desarrollo hibrido, es la capacidad de aprovechar un mismo código para varios sistemas operativos, de forma que esta aplicación estará disponible para un mayor número de usuarios.

Por otro lado se puede decir que la aplicación obtenida mejora sustancialmente a la aplicación oficial implementada por Indra, sobre todo en el tema de la funcionalidad, puesto que se han implementado todas las funciones disponibles en la API de usuario, e incluso alguna más que aún no está disponible. Otra ventaja de la aplicación desarrollada es que presenta una interfaz mucho más intuitiva de cara al usuario.

Por otro lado destacar que se han realizado diversas comprobaciones para que el uso de esta aplicación sea lo más correcto posible. Para ello se han tenido en cuenta todos los casos de error que podían surgir a la hora de realizar las diferentes acciones, mostrando mensajes de alerta personalizados para mejorar la experiencia del usuario con la aplicación.

En lo respectivo a la seguridad, se puede concluir que sería posible realizar un sistema totalmente seguro para el pago a través de los dispositivos móviles, pero actualmente la mayor limitación estriba en los entornos de desarrollo y los sistemas operativos que no incluyen todas las opciones de seguridad posibles. Pero aun pudiendo implementar manualmente dichas opciones, el punto crítico de seguridad recae totalmente en el acceso implementado por los sistemas bancarios.

En lo personal, destacar que al inicio del presente proyecto, se carecía de conocimientos sobre programación HTML, CSS y JavaScript y más aún del entorno de desarrollo Intel XDK, sin embargo ha sido posible realizar la aplicación, cumpliendo con los requisitos especificados. Destacar que el desarrollo de este proyecto, ha servido para afianzar conceptos estudiados durante la carrera, y para adquirir nuevos conocimientos, no solo relacionados con el desarrollo de aplicaciones, sino también con tecnologías web, redes, seguridad, etc.

Una vez terminado el presente PFC se puede concluir que los resultados obtenidos son satisfactorios, ya que la aplicación desarrollada cumple con los objetivos impuestos al principio del desarrollo del mismo.

6.2 Líneas futuras

El desarrollo de este proyecto ha supuesto la implementación de la primera versión de una aplicación prototipo dentro de la plataforma MobiWallet. Puesto que inicialmente no se tenía experiencia en el campo de las aplicaciones móviles, los objetivos que se fijaron no eran muy complejos. A medida que se ha ido desarrollando la aplicación se ha ido adquiriendo experiencia y conocimientos que si se hubieran tenido al principio del proyecto, seguramente cambiarían el enfoque que se ha dado a este proyecto.

Resaltar que se ha conseguido implementar las funcionalidades, que se marcaban en los objetivos, pero a medida que se ha ido desarrollando el proyecto, han ido surgiendo ideas de cómo introducir mejoras en la aplicación. Algunas de las posibles nuevas mejoras que se podrían implementar en futuras versiones de la aplicación son las siguientes:

- ***Implementar plugin que permita realizar recargas físicas.***
  Para que las lecturas/escrituras de saldo físico se pudieran llevar a cabo, es necesario acceder al chip NFC de la tarjeta MobiWallet. Por tanto, se podría estudiar la forma de desarrollar un plugin que permitiera realizar estas operaciones que actualmente no están disponibles.

- ***Incorporar la versión de autenticación mutua del protocolo SSL.***
  El equipo de desarrolladores de Intel, en respuesta a una pregunta planteada en su foro, aseguro que estaban trabajando en el desarrollo de una API que permitiera implementar el protocolo SSL con autentificación única. En el caso de que este desarrollo se llevara a cabo y se implementara en la aplicación, supondría una mayor seguridad en las comunicaciones entre el Web service y la aplicación.

- ***Sustituir certificados de usuario, por certificados de persona física***
  Si se logra llevar a cabo la autenticación mutua tal y como se plantea en el punto anterior, sería interesante sustituir los certificados de usuario generados en el Web service, por el certificado de persona física asociado al DNI electrónico. De esta forma se puede saber en todo momento la verdadera identidad de quien accede al Web service.

- ***Facilitar la operación de cambiar de tarjeta***
  Sería interesante desarrollar una funcionalidad que permitiera mostrar todas las tarjetas asociadas a un usuario de forma que se pudiera cambiar de una a otra, sin necesidad de introducir los datos asociados a ellas.

- ***Sustituir las credenciales de usuario por tokens en los envíos de peticiones***
  Se trataría de modificar las peticiones que se envían al Web service para evitar que el usuario y la contraseña se envíen constantemente. Esto se podría evitar empleando tokens de acceso, de forma que el usuario hace una petición para obtener el token y después lo incluye en el resto de peticiones. Otra ventaja que supondría el uso de tokens, es que se disminuiría el riesgo de que alguien pudiera interceptar las credenciales de usuario, aunque no es un aspecto crítico ya que las comunicaciones con el Web service viajan encriptadas sobre un canal SSL.

- ***Mejorar la función de consulta de movimientos.***
  Actualmente cuando se realiza una consulta de movimientos, la respuesta del Web service, contiene una serie de campos, como el operador o el tipo de tarifa, cuyo valor es un número, y es la propia aplicación la que tiene que tener almacenada la información (tarifas y operadores) para poder realizar un mapeo entre cada número y su valor asociado. Pero esta información almacenada en la aplicación no es dinámica, mientras que las tarifas y los operadores almacenados en la base de datos pueden cambiar en todo momento. Por tanto un cambio en la base de datos supondría que el mapeo realizado por la aplicación sería erróneo. Una solución sencilla a este problema, es que el propio Web service envié la información asociada al movimiento en texto, de forma que se liberaría a la aplicación de tener que realizar mapeos.

- ***Establecer una duración máxima de sesión***
  De forma que cuando pase un terminado tiempo la sesión caducará y se debieran volver a tener que introducir las credenciales de usuario. Esto evitaría problemas a aquellos usuarios que inician sesión, y se olvidan de cerrarla, dado que pasado un tiempo esta sesión caducará.

- ***Deshabilitar los botones físicos del dispositivo***
  Actualmente los botones físicos del dispositivo pese a que carecen de funcionalidad en la aplicación, mantienen la funcionalidad que les aporta el sistema operativo. Por tanto sería interesante poder deshabilitarlos de forma que todo el control de la aplicación se lleve a cabo mediante la pantalla táctil del dispositivo.

- ***Ofrecer más opciones al usuario en la aplicación***
  Incorporando mejoras y funcionalidades planteadas dentro del proyecto MobiWallet como planificadores de rutas, o sistemas que permitan realizar pagos y compras desde la aplicación.

REFERENCIAS

[1] Historia de los dispositivos móviles

http://dispmovs.blogspot.com.es/2012/03/historia-de-los-dispositivos-moviles.html

[2] Uso de los dispositivos móviles

https://www.yeeply.com/blog/como-usamos-los-dispositivos-moviles/

[3] Uso de las aplicaciones

http://www.einnova.com/aplicaciones-moviles/diseno-aplicacion-movil-comercio-electronico

[4] Definición proyecto MobiWallet

http://www.tst-sistemas.es/id-2/mobiwallet/

[5] Definición de aplicación

http://www.mastermagazine.info/termino/3874.php

[6] Evolución de las aplicaciones móviles

https://upsasoyyo.wordpress.com/2013/09/17/aplicaciones-moviles-la-evolucion/

[7] Imagen 1

http://www.marketingdirecto.com/especiales/mobile-marketing-blog/el-uso-de-aplicaciones-crecio-un-76-en-2014-con-las-apps-de-mensajeria-como-lideres/

[8] Aumento del uso de las aplicaciones móviles. Imagen 2

http://www.elandroidelibre.com/2015/01/google-play-supera-la-appstore-en-cantidad-de-aplicaciones-y-desarrolladores.html

[9] Tiendas de aplicaciones

http://alejandraplicacionesmoviles.blogspot.com.es/2015_08_01_archive.html

[10] Aplicaciones gratis y de pago

http://deideaaapp.org/tipos-de-aplicaciones-moviles-y-sus-caracteristicas/

[11] Imagen 3

http://es.slideshare.net/RevistaSG/hibrida-vs-nativa-pedro-galvan

[12] Aplicaciones nativas

http://appdesignbook.com/es/contenidos/las-aplicaciones/

[13] Lenguajes de desarrollo de aplicaciones nativas

http://www.lancetalent.com/blog/tipos-de-aplicaciones-moviles-ventajas-inconvenientes/

[14] Aplicaciones web

http://geospatialtraininges.com/recursos-gratuitos/tipos-de-aplicaciones-moviles/

[15] Aplicaciones hibridas

http://duall.me/que-son-las-aplicaciones-moviles-y-cuales-son-sus-tipos/

[16] Manejo del Hardware con aplicaciones híbridas

http://blog.aplicacionesmovil.com/aplicaciones-celular/desarrollo-de-aplicaciones-hibridas/

[17] Ventajas e inconvenientes de los tipos de aplicaciones

http://www.avante.es/tipos-de-aplicaciones-moviles-ventajas-y-desventajas/

[18] Definición de dispositivo móvil

https://admsaludv.wordpress.com/59-2/

[19] Definición de sistema operativo

http://www.areatecnologia.com/informatica/sistemas-operativos-moviles.html

[20] Imagen 6

https://netmarketshare.com/

[21] Metodología OSSTMM

http://www.isecom.org/research/osstmm.html

[22] Metodología OSSTMM

http://www.securitybydefault.com/2010/03/metodologia-osstmm.html

[23] Proyecto OWASP

https://es.wikipedia.org/wiki/Open_Web_Application_Security_Project

[24] Metodología ISSAF

http://insecuredata.blogspot.com.es/2009/04/metodologia-de-test-de-intrusion-issaf.html

[25] Riesgos y vulnerabilidades

http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/leccin_8_fallos_de_seguridad.html

[26] Imagen 7

https://www.incibe.es/blogs/post/Seguridad/BlogSeguridad/Articulo_y_comentarios/desarr

[27] Modelo de seguridad en las capas. Imagen 8

http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/leccin_25_seguridad_segn_las_capas.html

[28] Modelo de seguridad en las capas

http://52.1.175.72/portal/sites/all/themes/argo/assets/img/Pagina/Fabian_Molina_VIJNSI.pdf

[29] Servicios básicos de criptografía

http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/leccin_24_criptografia_y_encriptar.html

[30] Protocolo SSL

https://es.wikipedia.org/wiki/Transport_Layer_Security

[31] Bluetooth

http://synnick.blogspot.com.es/2012/02/conectividad-en-los-dispositivos.html

[32] Uso del bluetooth

http://www.tuexperto.com/2012/04/11/bluetooth-%C2%BFque-es-y-para-que-sirve/

[33] Wifi

http://es.ccm.net/contents/791-modos-de-funcionamiento-wifi-802-11-o-wi-fi

[34] Tecnologías de telefonía móvil

http://www.monografias.com/trabajos14/celularhist/celularhist.shtml

[35] Tecnologías de telefonía móvil

http://blog.masmovil.es/la-evolucion-de-la-tecnologia-movil-1g-2g-3g-4g/

[36] Tecnologías de telefonía móvil

http://www.movilbak.es/blog/index.php/2012/11/14/la-evolucion-de-la-tecnologia-del-movil-y-sus-cuatro-generaciones/

[37] NFC

http://www.criptored.upm.es/crypt4you/temas/privacidad-proteccion/leccion4/leccion4.html

[38] Modos de funcionamiento del NFC

http://www.xataka.com/moviles/nfc-que-es-y-para-que-sirve

[39] Posibles usos del NFC

http://www.mibqyyo.com/articulos/2015/07/23/nfc-funcionamiento-utilidades/#/vanilla/discussion/embed/?vanilla_discussion_id=0

[40] Imagen 10

http://www.by.com.es/blog/que-es-la-tecnologia-nfc-y-que-usos-tiene/

[41] Definición proyecto MobiWallet

http://www.indracompany.com/sostenibilidad-e-innovacion/proyectos-innovacion/mobiWallet-mobility-and-transport-digital-wallet

[42] Objetivos proyecto MobiWallet

http://www.mobiwallet-project.eu/index.php?option=com_content&view=article&id=21&Itemid=118

[43] Piloto Santander proyecto MobiWallet

http://www.mobiwallet-project.eu/index.php?option=com_content&view=article&id=24&Itemid=122

[44] Protocolo SSL

https://es.wikipedia.org/wiki/Transport_Layer_Security

[45] Imagen 14. Imagen 15

http://blog.dreamix.eu/java/two-way-ssl-connection-with-tomcat-and-oc4j-2

[46] Entorno de desarrollo Intel XDK y sus utilidades

http://www.sitepoint.com/introduction-intel-xdk/

[47] Plataformas para la que se generan aplicaciones con Intel XDK

http://sg.com.mx/buzz/construye-apps-moviles-usando-html5-intel-xdk#.VihG8dLhBph

[48] Características principales de Intel XDK

http://byspel.com/intel-xdk/

[49] HTML

https://developer.mozilla.org/es/docs/Web/HTML

[50] HTML

http://aprenderaprogramar.es/index.php?option=com_content&view=article&id=435:iique-es-y-para-que-sirve-html-el-lenguaje-mas-importante-para-crear-paginas-webs-html-tags-cu00704b&catid=69:tutorial-basico-programador-web-html-desde-cero&Itemid=192

[51] Versión 5 de HTML

https://es.wikipedia.org/wiki/HTML5#Nuevas_APIs_y_Javascript

[52] Javascript

http://www.maestrosdelweb.com/que-es-javascript/

[53] Diferencia entre javscript y java

http://librosweb.es/libro/javascript/capitulo_1.html

[54] CSS3

http://www.aprenderaprogramar.com/index.php?option=com_content&id=546:que-es-y-para-que-sirve-el-lenguaje-css-cascading-style-sheets-hojas-de-estilo&Itemid=163

[55] CSS3

http://html5.dwebapps.com/que-es-css3/

[56] App Framework

http://www.phonegapspain.com/topic/app-framework/

[57] Imagen 17

https://ebaste.wordpress.com/2012/01/16/jq-mobi-jquery-reescrito-y-optimizado-para-html5-ios-y-android-gratis/

[58] Cordova

http://blog.koalite.com/2012/03/empaquetar-aplicaciones-html5-con-phonegapcordova/

[59] Cordova

http://voylinux.com/que-es-y-que-no-apache-cordova/

[60] Cordova en Intel XDK

https://software.intel.com/en-us/xdk/docs/adding-third-party-plugins-to-your-xdk-cordova-app

[61] "*Functional Specification of Web Service Commands"*, MobiWallet internal report, Agosto 2015.

[62] Plugin para el manejo del chip NFC

https://github.com/chariotsolutions/phonegap-nfc

[63] Imagen 49

https://software.intel.com/en-us/node/564377

[64] Imagen 51

https://software.intel.com/en-us/forums/intel-xdk/topic/562592

ANEXO

1º RegisterUser

URL:
https://XXX.XXX.XXX.XX:YYYY/Tlmat1Pruebas2Sergio3/rest/UserActions/mb/registerUser
METODO: POST
PETICION:

```
<MW_NEW_USER>
    <MW_VERSION>1.0</MW_VERSION>
    <MW_USR_ID>unicanSSL</MW_USR_ID>
    <MW_SURNAME>Cosio Molleda</MW_SURNAME>
    <MW_NAME>Pablo</MW_NAME>
    <MW_GENRE>male</MW_GENRE>
    <MW_PASSPORTID>72074825L</MW_PASSPORTID>
    <MW_BIRTHDATE>1989-01-25</MW_BIRTHDATE>
    <MW_POSTCODE>39015</MW_POSTCODE>
    <MW_CITY>Herrera de Camargo</MW_CITY>
    <MW_ADDRESS>Avenida de Bilbao numer4 2 derecha</MW_ADDRESS>
    <MW_EMAIL>tlmat_gestion@unican.es</MW_EMAIL>
    <MW_MOBILE>699858966</MW_MOBILE>
    <MW_PHONE2>666666666</MW_PHONE2>
</MW_NEW_USER>
```

POSIBLES RESPUESTAS:

a) Todo correcto

```
<MW_ERRORCODE>1</MW_ERRORCODE>
<MW_ERRORDESC>WITHOUT ERRORS. USER INSERTED</MW_ERRORDESC>
```

b) Usuario ya registrado

```
<MW_ERRORCODE>2</MW_ERRORCODE>
<MW_ERRORDESC>This MW_USER is already in the database</MW_ERRORDESC>
```

c) Error formato

```
<MW_ERRORCODE>-1</MW_ERRORCODE>
<MW_ERRORDESC>javax.xml.bind.UnmarshalException  - with linked
exception: [org.xml.sax.SAXParseException; lineNumber: 7; columnNumber:
43; cvc-pattern-valid: Value '72074825L12' is not facet-valid with
respect to pattern '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][a-zA-Z]'
for type 'stringDNI'.]</MW_ERRORDESC>
```