ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Casos prácticos de su uso de la Tarjeta Universitaria Inteligente basados en QR y otros usos

AUTOR: Inmaculada Concepción García Machado
DIRECTORES: José Juan Sánchez Manzanares
Juan Carlos Sánchez Aarnoutse
Mayo / 2016

Autor	Inmaculada Concepción García Machado
E-mail del Autor	icgarciamachado@gmail.com
Directores	José Juan Sánchez Manzanares Juan Carlos Sánchez Aarnoutse
E-mail de los Directores	pepe.manzanares@si.upct.es juanc.sanchez@upct.es
Título del PFC	Casos prácticos de su uso de la Tarjeta Universitaria Inteligente basados en QR

Resumen

Análisis sobre la Tarjeta Universitaria Inteligente sobre todas sus funciones y servicios, sus características físicas y evolución. Principalemente profundiza sobre la firma electrónica mediante la TUI con todas sus ventajas y oportunidades que ofrece a la comunidad universitaria.

Se analiza también sobre los códigos QR y su nuevo uso e implantación en la TUI llegando a desarrollar tres casos prácticos para aplicaciones móviles con el fin de utilizar el código QR de la TUI en distintos aspectos universitarios.
El primero de los casos prácticos realiza una identificación electrónica del usuario para la asistencia a eventos.
El segundo realiza un registro del dispositivo móvil del usuario para su posterior uso, inicio de sesión y enviar notificaciones.
Y el tercero realiza transacciones monetarias con la TUI usando una cuenta virtual.

Titulación	Ingeniería de Telecomunicaciones
Intensificación	
Departamento	Tecnologías de la Información y la Comunicaciones
Fecha de Presentación	Mayo - 2016

Índice

Índice de figuras

Capítulo 1. Introducción

La Tarjeta Universitaria Inteligente (TUI) es el medio de identificación para todo el colectivo universitario (estudiantes, personal docente e investigador (PDI) y personal de administración y servicios (PAS)); además de la identificación "física" o presencial, esta tarjeta debe ser también el medio para la identificación de los usuarios en los servicios telemáticos, servir para la firma electrónica reconocida, como medio de pago y muchos servicios más.

Hay todo un gran potencial en la Tarjeta Universitaria Inteligente. Al estar provista de un chip de contacto que permite almacenar la información sobre el titular con una gran capacidad y también de un chip de proximidad, permite ofrecer al usuario una gran cantidad de servicios impensables con los sistemas antiguos. Además, actualmente con la nueva TUI R7 de la cual se hablará en este proyecto, también entra la posibilidad de poder utilizar todas las aplicaciones de la TUI en modo con contacto o en modo sin contacto haciendo que la TUI sea cada vez más útil y cada vez más cómoda para cualquier servicio en el que se utilice.

A continuación se muestra todo sobre la TUI como su funcionalidad, sus características, su tecnología, su seguridad, su presente y su futuro, ya que actualmente, el avance tecnológico está haciendo que evolucione cada vez más rápido y ofreciendo a sus usuarios más servicios y prestaciones. Dicha evolución me llevó al desarrollo de tres aplicaciones desarrolladas al final del proyecto las cuales usan un código QR como identificación, debido a que ya se está implantando en la TUI un código QR personal y puede servir de mucha utilidad.

El presente proyecto tiene su origen en mis años colaborando en el Servicio de Informática de la Universidad Politécnica de Cartagena como becaria de Desarrollo y Mantenimiento del Carnet Universitario Inteligente de la UPCT.

En estos años he podido apreciar el valor de la TUI y que utilidades tiene. Parte de este trabajo lo desarrollé para la Universidad Politécnica de Cartagena y el haber podido comprobar tanto su utilidad como el uso que se le da me ha generado una gran satisfacción.

Capítulo 2. Tarjeta Universitaria Inteligente (TUI)

La importancia de este proyecto reside en que la Tarjeta Universitaria Inteligente (TUI) tiene un alto componente tecnológico y permite acceder a multitud de servicios universitarios. Además sirve como identificador para la comunidad universitaria, es decir, para los alumnos, el personal de administración y servicios (PAS) y el personal docente e investigador (PDI). En un principio veremos la parte física (el hardware) de la Tarjeta Inteligente Universitaria, después nos centraremos en las características, funcionalidades y servicios que ofrece a sus usuarios y se terminará con un pequeño resumen sobre la evolución que ha experimentado la TUI en la UPCT desde sus comienzos.

2.1 Hardware de la TUI

Actualmente en la UPCT se encuentra la TUI R5 de Gemalto, pero ya se está empezando a implantar y a desarrollar aplicaciones para la TUI R7 la cual implementa la última tecnología existente relacionada con las smart cards, aportando nuevas funcionalidades, mayor capacidad y robustez para sus aplicaciones ya existentes y, además se encuentra preparada para añadirle nuevos servicios y funciones. Como la TUI R7 es el futuro próximo de la TUI, en este proyecto se va a profundizar sobre ella.

La R7 está compuesta por los microprocesadores de la tarjeta para su uso con contactos y sin contactos, una memoria EEPROM y la antena para el modo sin contactos. Estos componentes están incorporados en un sustrato de plástico del tamaño de una tarjeta de crédito como puede observarse en la figura 2.1.

Figura 2.1 Tarjeta Inteligente Universitaria desfragmentada

Esta tarjeta inteligente no contiene una fuente de alimentación, o un display ni un teclado. Para poder realizar la comunicación, la tarjeta debe colocarse dentro de un lector (modo con contacto) o estar cerca de un dispositivo que lea estas tarjetas y que esté conectado a un ordenador (modo sin contacto o *contactless*).

Para el modo con contactos, cada smart card tiene 8 puntos de contacto en la superficie del sustrato de plástico. En la figura 2.2 se puede ver la funcionalidad y la localización de cada

punto. Las dimensiones y la localización de los contactos se encuentran especificadas en la parte 2 de la ISO 7816.

Figura 2.2 Contactos en el chip WG10

La funcionalidad de cada contacto[1]:

- C1 – Vcc: Suministra energía al chip. El valor para las distintas smart card suelen ser de 3 ó 5 voltios. La TUI R7 necesita 5 V y, por ejemplo, las smart cards de los teléfonos móviles usan una Vcc de 3 V.
- C2 – Rst: Se usa para enviar una señal de reset al microprocesador.
- C3 – Clk: Este contacto provee un reloj externo al microprocesador. De este reloj externo se deriva el reloj interno porque las tarjetas inteligentes no poseen un generador de reloj interno.
- C4: Está reservado para un uso futuro.
- C5 – Gnd: Se usa como punto de referencia de voltaje. Se considera que su valor es 0.
- C6 – Vpp: Se usa solamente en tarjetas antiguas y es opcional. Cuando se usa se suministra el voltaje de programación, que tiene dos niveles (uno bajo y otro alto). El nivel alto de voltaje se utiliza para programar la EEPROM de algunos chips antiguos de smart card.
- C7 – I/O: Se usa para transferir datos y comandos entre la tarjeta inteligente y el lector en modo half-duplex, es decir, que los comandos o los datos solo se pueden transmitir en una sola dirección en un determinado instante.
- C8: Está reservado para un uso futuro.

Para el modo sin contactos se usa una antena RFID (Radio Frequency IDentification) la cual se verá profundamente más adelante en el apartado de La TUI sin contacto.

2.1.1 Unidad Microprocesadora de la Tarjeta Inteligente (MPU)

Los microprocesadores de la TUI ejecutan instrucciones programadas. En ella se usa Java Card que es una tecnología subconjunto del lenguaje de programación Java con un entorno de ejecución optimizado para las tarjetas inteligentes. Ofrece un entorno seguro para las distintas aplicaciones que se ejecutan en las tarjetas inteligentes y otros dispositivos de confianza que tienen capacidad de memoria y de procesamiento muy limitadas. Sobre Java Card se profundizará más adelante.

Todo esta estructura se implementa en un sistema de archivos en el que hay DFs (*dedicated files*, equivalente a directorios) y EFs (*elementary files*, equivalente a ficheros). Estos archivos (DFs y EFs) pueden ser identificados por nombre, por id o por un path (concatenación de los ids de los DFs padre de la ruta hasta el DF o EF en cuestión).

[1] ISO 7816-2. http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-2.aspx

En la TUI del Banco Santander tenemos los siguientes campos relacionados con la UPCT alojados en distintos directorios[2]:

- Nº Identificación Académica (NIA). Protegido ante actualización.
- Cód. Documento Identidad (D.N.I./Pasaporte) Protegido ante actualización.
- Nombre y apellidos. Protegido ante actualización.
- Categoría (estudiante, PDI, PAS, antiguo alumno, otros). Protegido ante actualización.
- Centro. Protegido ante actualización. Filler de uso interno Santander. Relleno a ceros o espacios, según formato.
- Año Curso Actual. Protegido ante actualización. Formato AAAA.
- Indicador de tarjeta activa/inactiva. Protegido ante actualización. 0 Inactiva; 1 Activa.
- PIN universitario. Protegido ante lectura y escritura.
- Reserva control de accesos. Protegido ante actualización.
- Campos de uso interno. Filler de uso interno Santander. Relleno a ceros o espacios, según formato.
- Fecha de fin de validez del monedero. Protegido ante actualización.

El acceso a los ficheros puede ser restringido en función de varias políticas, entre ellas que el usuario esté o no identificado mediante su PIN. Esto lleva a tener dos zonas:

- Zona pública: Accesible en lectura sin restricciones. Por ejemplo el NIF, el nombre y apellidos y la fecha de validez de la TUI.
- Zona privada: Accesible en lectura por el usuario, mediante la utilización de la Clave Personal de Acceso o PIN. Por ejemplo el monedero electrónico.

2.1.2 Memoria EEPROM

Electrical Erasable Programmable Read-Only Memory, o Memoria de sólo lectura programable y eléctricamente borrable. Esta memoria preserva los datos cuando no hay fuente de alimentación y además el contenido de la memoria se puede modificar durante un uso normal de la tarjeta. Se usa para guardar los datos y además también se pueden escribir en ella las aplicaciones de usuario después de la fabricación de la tarjeta. En el caso de la TUI R7, es de 72 KB para aplicaciones estándar y puede estar limitada a 45 KB cuando se instala M/Chip Avance (para uso financiero).

2.1.3 Antena RFID

La antena usada en la TUI (Fig.2.3), también llamada "tag" RFID, es una antena pasiva, es decir, que no tiene alimentación eléctrica propia. Se alimentan de la señal magnética que les llega de los lectores, de ahí su forma en espiral plana, la cual induce una pequeña corriente eléctrica suficiente para que la tarjeta funcione. Opera en alta frecuencia (13.56

[2] Información cedida por el Observatorio de la Tarjeta Inteligente

MHz). Más adelante se explicará más en profundidad cómo funciona el modo sin contacto de la TUI.

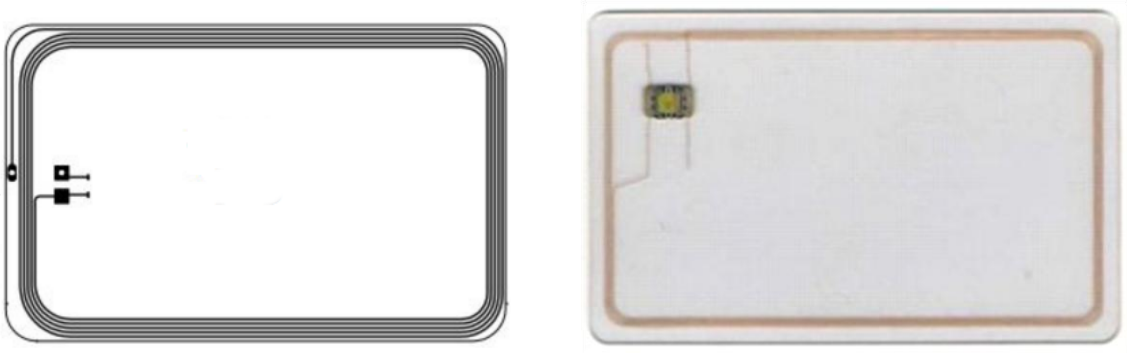

Figura 2.3 Antena RFID

2.2 Personalización de la TUI

La personalización es distinta para cada universidad e incluso en algunas tarjetas es distinta en función de la relación del titular con la entidad bancaria.

En todas las tarjetas se imprimen la fotografía, el nombre y los apellidos para que sean válidas como documento identificativo. También suelen imprimirse la vinculación del titular con la universidad, para completar así la identificación[3].

En la TUI se incluye un código de barras debido al uso extendido de la tarjeta en las bibliotecas universitarias y al uso de esta por los sistemas de reservas de libros. Actualmente también se está implantando el código QR que puede usarse con la misma utilidad. Un ejemplo de la nueva TUI R7 con el código QR añadido se puede observar en la figura 2.4.

Figura 2.4 Ejemplo de TUI con QR

Un aspecto importante sobre la personalización de una tarjeta es que debe cumplir con una serie de requisitos a la hora de situar logotipos, la zona de la firma, la banda magnética o la

[3] Implantación de la TUI en el Sistema Universitario Español. Estudio realizado por el Observatorio TUI (2014).

foto. Estos requisitos los valida la Fábrica Nacional de Moneda y Timbre y la TUI cumple con la normativa.

En los comienzos de la TUI, algunas universidades no emitían las tarjetas con fecha de caducidad impresa, pero más tarde comenzaron a imprimirla debido a la exigencia de terceros para aplicar los descuentos ofrecidos.

Actualmente en la UPCT, no se imprime la fecha de caducidad ni la relación entre el titular y la universidad debido a que pueden cambiar al día siguiente de estar impresa la TUI, lo que daría lugar a una "tarjeta errónea". Por tanto, es preferible que sólo la información permanente aparezca impresa en la tarjeta y que la variable se obtenga desde dispositivos de lectura, como móviles, PCs o similares, a través del código de barras, el identificador de usuario o el código QR.

2.2.1 Circuito de emisión

Hay dos modos de conseguir la TUI: solicitándola a través del Portal de Servicios de la UPCT o la emisión instantánea. En el caso de que el usuario no use la emisión instantánea, el circuito de emisión de las tarjetas es más laborioso. La emisión instantánea acaba de ser implantada en la UPCT y se habla de ella más adelante.

Si se solicita la TUI a través del Portal de Servicios, en la UPCT se dispone de un circuito de emisión prácticamente automatizado. Este dato es relevante porque el número de incidencias relacionadas con el circuito de emisión, en este caso, está inversamente relacionado con su nivel de automatización (Baixauli Soler, 2014).

La mayoría de solicitudes de la TUI se concentra en el periodo de matrícula. La emisión de una TUI requiere de una foto del usuario y de datos de la universidad que se le envían a la entidad correspondiente a través de ficheros de intercambio. La recogida de la fotografía identificativa del titular sigue siendo un proceso manual porque es el usuario el que añade digitalmente su fotografía en el Portal de Servicios y tiene que ser validada por una persona de administración y servicios de la UPCT. Y después, en el caso de que sea la fotografía válida, se envía a la entidad financiera colaboradora.

La TUI cuando es solicitada pasa por unos estados:

- **Inicial (I)**. Primer estado de una solicitud. En este estado están las solicitudes de TUI pendientes de validar la fotografía. La solicitud aún no ha salido de la Universidad.
- **Mecanizado (M)**. En este estado se encuentran las solicitudes que tienen una fotografía válida y que están esperando a ser enviadas a la entidad correspondiente. Las solicitudes se envían una vez a la semana a la entidad. A un grupo de solicitudes enviadas a la vez se le llama lote.
- **Comunicado (C)**. Cuando una entidad recibe correctamente una solicitud cambia el estado de la solicitud a 'Comunicado' y envía un fichero a la UPCT para informarle de que ha sido recibido.
- **Personalización (P).** En caso de que todos los datos del usuario sean correctos se pasa la TUI al estado de personalización. La entidad manda los datos a la estampadora para que imprima la TUI con el diseño de la UPCT y los datos del usuario.

- **Distribución (D)**. Una vez que está la tarjeta terminada se manda al domicilio del usuario mediante correo postal.
- **En la Universidad (U)**. Cuando las TUIs se envían al usuario la entidad cambia el estado de la solicitud a en la Universidad pero no es posible saber qué día han recibido la TUI exactamente.
- **Entregado (E).** TUI entregada satisfactoriamente al usuario. Similar al estado anterior debido a que no sabemos si una TUI ha sido entregada a no ser que la utilice y conste en la entidad o la universidad de ello.
- **Fallido (F)**. En caso de que los datos no sean correctos o haya algún error, la entidad no envía la solicitud a la estampadora, en su lugar, le cambia el estado y comunica a la UPCT el error para que sea corregido.
- **Anulado (A)**. La UPCT le comunica al usuario el motivo de la anulación de la solicitud y cómo puede modifica los datos erróneos. Cambia la solicitud a estado anulado para que el usuario, una vez corregidos sus datos, pueda volver a realizar una nueva solicitud de la TUI.

En la figura 2.5 se puede observar el diagrama de estados del circuito de emisión.

En la UPCT, el proceso manual en el circuito es la contrastación de datos. Este proceso es fuente de múltiples incidencias y ralentizan el circuito de emisión y demoran la entrega de la TUI al usuario, sin contar con problemas postales en la entrega por correo. Otros procesos que en la UPCT ya no son manuales serían la composición de ficheros de intercambio y el envío del cartón foto los cuales se han automatizado. En la figura 2.6 se muestra un esquema del funcionamiento actual del circuito de emisión con el Banco Santander (Baixauli Soler, 2014).

El tiempo de entrega es uno de los factores más importantes que condicionan el nivel de satisfacción con el circuito de emisión. La demora en la entrega implica que el titular tiene que esperar para acceder a ciertos servicios universitarios, lo que hace que la impresión inmediata sea la opción más rápida y práctica.

Los principales aspectos a mejorar respecto al circuito de emisión están relacionados con los tiempos de espera entre que se tramita la solicitud y se entrega la tarjeta al titular, los procesos manuales existentes y la difícil trazabilidad de la TUI durante el proceso de emisión. La mayor parte de las incidencias y consultas tienen que ver con estos aspectos.

Además, también se producen errores en la personalización de las tarjetas con los datos del titular debidos a inconsistencias entre los datos que tiene registrados la universidad y los datos que disponen las entidades bancarias en sus sistemas informáticos. Estos errores pueden ser debidos a diferentes motivos, errores de transcripción, que los usuarios indiquen diferentes direcciones para uso universitario o bancario o, algunas veces, a limitaciones en los campo de los formularios; por ejemplo, el tamaño del campo *Calle y número* que en algunas ocasiones es muy largo y no cabe toda la información necesaria para el envío.

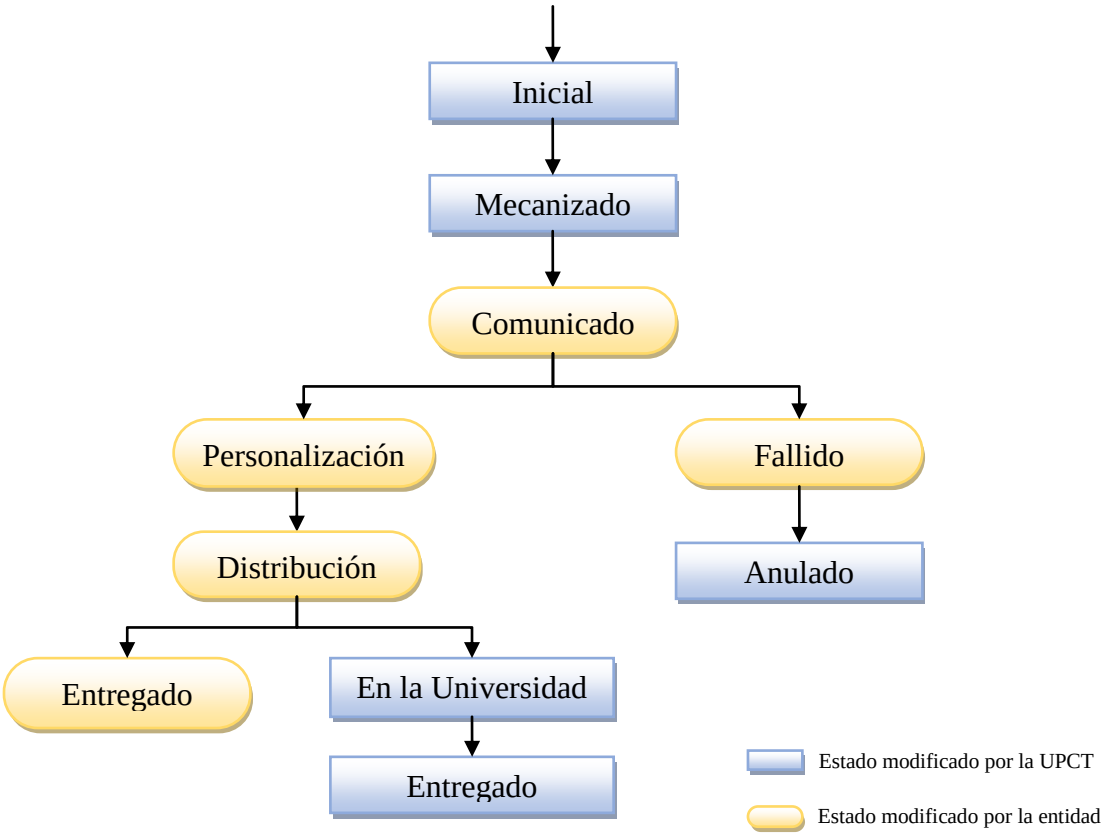

Figura 2.5 Diagrama de estados de la solicitud de una TUI

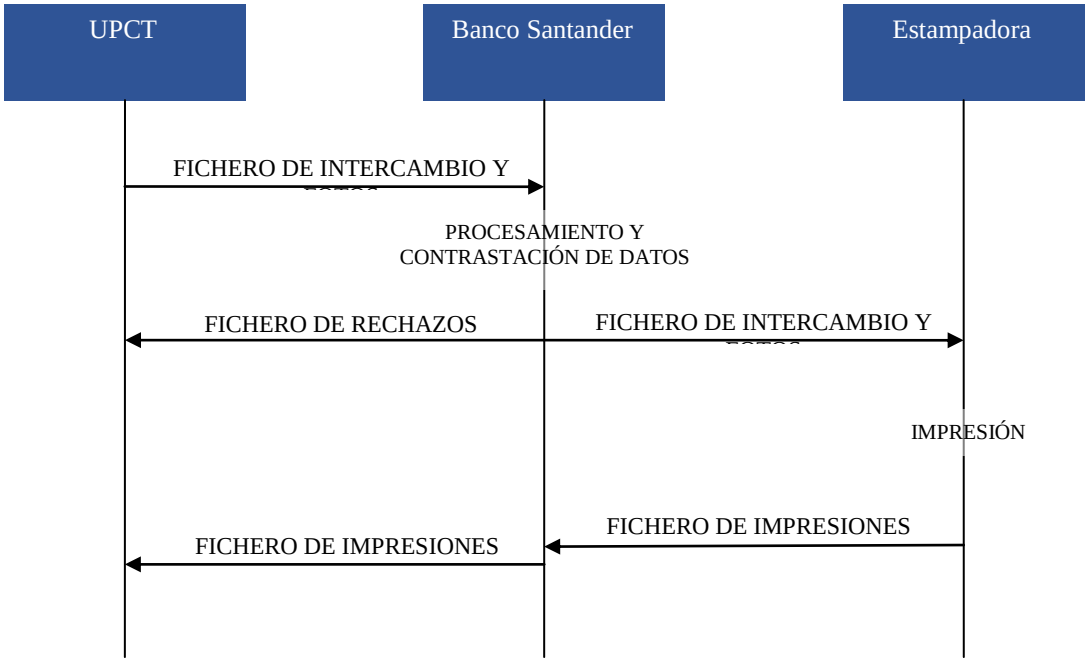

Figura 2.6 Esquema de funcionamiento del Sistema de Emisión actual

2.2.2 NexUSC

Actualmente está en marcha un proyecto llamado **NexUSC** desarrollado por el Observatorio TUI del Banco Santander el cual consiste en un sistema libre para que todas las universidades que utilizan la TUI puedan gestionar de forma más eficiente el circuito de emisión de tarjetas con las estampadoras y el Banco Santander.

Consiste en un Web Service con autenticación sobre SSL que se encargará del envío y la recepción de los datos entre las universidades y el Banco Santander. Permitirá automatizar completamente las tareas de solicitud y descarga de toda la información referente a una TUI[4].

El proyecto tiene como objetivos optimizar y resolver estas diferentes situaciones:

- Pérdida de tiempo en operaciones manuales en el proceso de intercambio de fichero con la empresa estampadora y con el Banco Santander.
- Pérdida de tiempo en operaciones manuales del Banco Santander o las estampadoras al procesar los datos que recibe de las universidades.
- Disminuir los errores en los procesos manuales de procesamiento de datos.
- Retraso en el tiempo de entrega de TUIs a los usuarios de las universidades.
- Homogeneización de los datos recibidos por el Banco Santander
- Imposibilidad de compartir soluciones relacionadas entre las universidades, debido a implementaciones diferentes en cada una de ellas.

La UPCT enviará a NexUSC dos ficheros: uno con todas las altas de solicitudes de TUIs y otro con todas las fotos de las solicitudes de ese lote. Actualmente se envía un correo electrónico al Banco Santander con un fichero el cual contiene todas las fotos y un archivo de texto con las altas. Con el nuevo procedimiento, después el Banco Santander recoge los ficheros pendientes en NexUSC y procesa las solicitudes. Si hay alguna incidencia, el Banco Santander sube a NexUSC un fichero con los registros rechazados el cual la UPCT recogerá desde NexUSC.

Una vez impresas las TUIs, el Banco Santander envía por correo postal las TUIs a los destinatarios a la vez que le informa a NexUSC los valores de cada uno de los plásticos que procesa (número de monedero, número de serie y número de MIFARE) subiendo un fichero de impresiones.

Por último, la UPCT recogerá el fichero de impresiones de NexUSC y podrá actualizar el estado de las solicitudes informando así al usuario del estado de su solicitud.

[4] http://www.observatoriotui.com/nexusc

El nuevo circuito funcionará bajo el esquema de la figura 2.7.

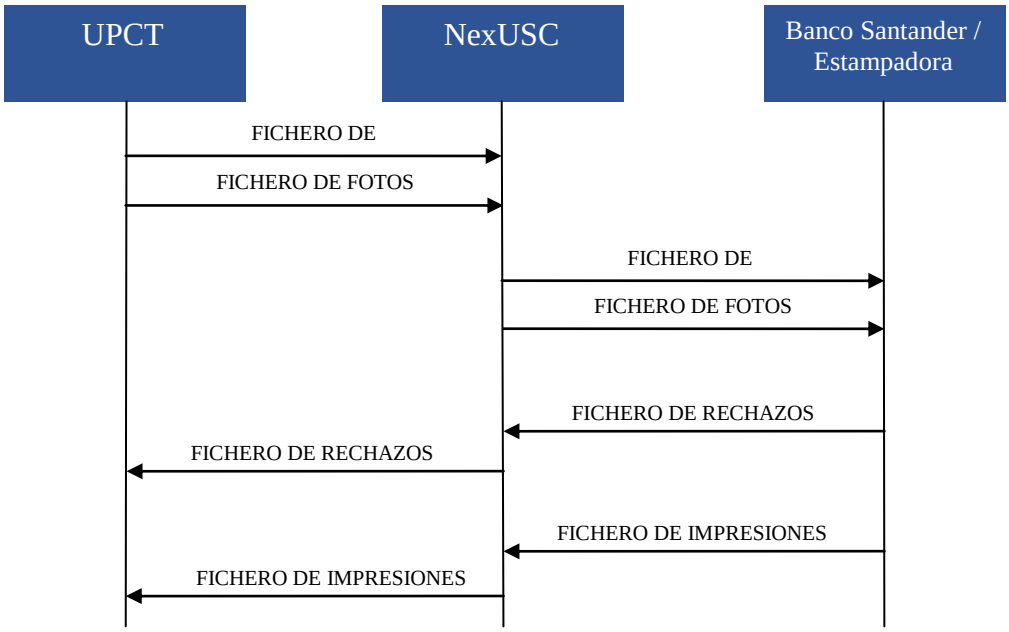

Figura 2.7 Esquema de funcionamiento NexUSC

2.2.3 Emisión instantánea

Desde septiembre del 2015 se puede realizar en la UPCT la impresión instantánea de la TUI. Una de las ventajas que tiene es la de obtener instantáneamente la TUI. Sin embargo, la calidad fotográfica no es la misma que si se solicita la TUI mediante el Portal de Servicios ya que las máquinas de emisión instantánea de la TUI sólo tienen un color para imprimir. Estas máquinas tienen tarjetas semipersonalizadas, es decir, hay campos en blanco para los datos personales, la fotografía, el código de barras y el código QR (Baixauli Soler, 2014).

Actualmente las máquinas de emisión instantánea únicamente inyectan los datos básicos del usuario en la TUI, pero está en desarrollo poder además introducir en la TUI más datos, como por ejemplo, el certificado electrónico en el mismo momento de la impresión inmediata.

2.3 Java Card

En los microprocesadores de la TUI se usa Java Card que es una tecnología subconjunto del lenguaje de programación Java con un entorno de ejecución optimizado para las tarjetas inteligentes. Ofrece un entorno seguro para las distintas aplicaciones que se ejecutan en las tarjetas inteligentes y otros dispositivos de confianza que tienen capacidad de memoria y de procesamiento muy limitadas, por ejemplo, también se usan en las tarjetas SIM de los móviles.

En la TUI se pueden implementar múltiples aplicaciones y se pueden añadir más incluso después de que se haya emitido la tarjeta al usuario final. Estas aplicaciones son llamadas applets, como por ejemplo el applet de transporte, monedero electrónico o firma electrónica. [5]

El objetivo de Java Card es ser un estándar de tarjetas inteligentes que permita a una misma applet funcionar en diferentes tarjetas inteligentes. Al igual que ocurre con el lenguaje de programación Java, se consigue utilizando la combinación de una máquina virtual, en este caso la Máquina Virtual de Java Card, y unas librerías cuya API[6] está especificada. Cualquiera puede desarrollar una aplicación Java Card, pero un tema a tener en cuenta es la seguridad. La seguridad en las tarjetas inteligentes es una de las partes más importantes y la más difícil de programar correctamente y se deben seguir unas normas estrictas de seguridad para ello.

Gracias a la última actualización de Java Card, es posible acceder a nuevas funcionalidades antes sólo disponibles mediante la interfaz de contacto y también ahora permite la implementación de aplicativos para el acceso a la memoria MIFARE en modo con contactos, es decir, con la TUI R7 podemos acceder a todas las aplicaciones sin importar si estamos accediendo con contacto o sin contacto con el lector. Un ejemplo de esto sería pagar con el monedero electrónico sin tener que insertar la tarjeta en un lector, sólo pasándola cerca del lector, siempre y cuando el lector permita el modo sin contacto.[7]

2.4 La TUI con contacto

Antes de comenzar, hay que aclarar que la TUI y las smart cards en general se insertan en el interior de un dispositivo de aceptación (CAD), que puede estar conectado a un ordenador y estos CAD`s se dividen en dos grupos: los lectores y los terminales.[8]

- Un lector se conecta a un ordenador a través de un puerto serie, paralelo o USB. Existen para tarjetas con contacto o sin contacto. Los lectores no tienen la inteligencia para procesar datos pero muchos de ellos tienen funciones de detección y corrección de errores si los datos transmitidos no cumplen con el protocolo de transporte subyacente.
- Los terminales, en cambio, son ordenadores que integran un lector de smart cards como uno de sus componentes. Tienen la capacidad de procesar los datos intercambiados.

A las aplicaciones que se comunican con la smart card, estén en el ordenador conectado a un lector o en el terminal, son llamadas aplicaciones del host. Estas aplicaciones son las que dirigen el proceso de comunicación con la smart card y en el siguiente punto se explica cómo es el proceso.

[5] https://jcardsim.org/es/blogs/jcardsim-java-card-simple
[6] API. Application Programming Interface
[7] Implantación de la TUI en el Sistema Universitario Español. Estudio realizado por el Observatorio TUI (2014).
[8] Introducción a las Java Cards.http://bibing.us.es/proyectos/abreproy/11458/fichero/PFC%252FCapitulo03+-+Introduccion+a+las+Java+Cards.pdf

2.4.1 Comunicación con la TUI

La comunicación entre la TUI y el ordenador (host) es half-duplex, es decir, solo uno de ellos puede transmitir datos en un instante de tiempo.

Cuando una tarjeta inteligente se comunica con un terminal usan sus propios paquetes de datos llamados APDU's (application protocol data units). Una unidad APDU puede ser un mensaje de orden o un mensaje de respuesta.

El modelo de comunicación entre la tarjeta y el host es de maestro y esclavo. La tarjeta siempre será el esclavo, esperando APDU's de comando procedentes del host (C-APDU). Una vez recibido una orden, la tarjeta ejecuta la instrucción especificada y responde al host (R-APDU). La tarjeta y el host intercambian APDU's de comando y respuesta de forma alternada.

El protocolo APDU

El protocolo APDU se especifica en la ISO 7816-4 y es un protocolo de nivel de aplicación entre una tarjeta inteligente y una aplicación del host. Los mensajes APDU poseen dos estructuras:

- APDU de comando (C-APDU): Estructura usada por la aplicación del host para mandar órdenes.

C-APDU						
Cabecera				Cuerpo (opcional)		
CLA	INS	P1	P2	Lc	Data field	Le

Figura 2.8 Estructura de una C-APDU

Siendo:

- **CLA**: Clase de instrucción. Byte que identifica una categoría de APDU's de comando y respuesta. Pueden tener los siguientes valores: 0x0X, 0x8X, 0x9X, 0xAX. Donde la X sirve para determinar canales lógicos y asegurar la codificación del mensaje.
- **INS**: Código de instrucción. Especifica la instrucción del comando.
- **P1**: Parámetro de la instrucción.
- **P2**: Parámetro de la instrucción.
- **Lc**: Especifica la longitud del Data Field en bytes.
- **Data field**: Contiene información adicional para llevar a cabo la instrucción. Es opcional.
- **Le**: Especifica el número de bytes esperados en la respuesta de la tarjeta.

- APDU de respuesta (R-APDU): Estructura usada por la tarjeta para enviar las respuestas a la aplicación del host.

R-APDU		
Cuerpo (opcional)	Cola	
Data field	SW1	SW2

Figura 2.9 Estructura de una R-APDU

Siendo:

- **Data field**: Es opcional y su longitud coincide con la especificada en el campo Le del C-APDU al que responde.
- **SW1 (status word)**: Indica el estado de procesado en la tarjeta después de la ejecución del comando (por ejemplo, el éxito se marca con el valor 0x9000).
- **SW2 (status word)**: Indica el estado de procesado en la tarjeta después de la ejecución del comando (por ejemplo, el éxito se marca con el valor 0x9000).

Siempre una C-APDU va emparejada con una R-APDU.

Las APDU's se transmiten gracias al protocolo de transporte TPDU definido en la ISO 7816-3. En este caso, a las estructuras de datos intercambiadas entre un host y una tarjeta usando este protocolo se les llama unidades de datos del protocolo de transporte o TPDU's.

Protocolo TPDU

Actualmente, los dos protocolos de transporte más usados en los sistemas de tarjetas inteligentes son el protocolo T=0 y el protocolo T=1.

El protocolo T=0 está orientado a bytes ó caracteres, esto quiere decir que la mínima unidad procesada transmitida por el protocolo es un byte. Este protocolo también es semidúplex, asíncrono y la conversación siempre la inicia el host siguiendo un esquema maestro-esclavo. La TUI usa el protocolo T=0[9].

En cambio, el protocolo T=1 está orientado a bloques de datos, esto significa que los bloques (secuencia de bytes) son la mínima unidad de datos que se transmiten entre una tarjeta y el host. La comunicación es entre iguales, alternándose el permiso para enviar datos.

ATR

Automáticamente, después de encenderse una tarjeta inteligente y antes de empezar cualquier comunicación, una tarjeta inteligente envía un mensaje "answer to reset" (ATR) al host. Esto le transmite al host los parámetros requeridos por la tarjeta para establecer una comunicación de datos correcta. El ATR está compuesto por más de 33 bytes. Entre otros, está compuesto por parámetros de transmisión, como el protocolo de transporte que soporta la tarjeta (T=0 ó T=1); la tasa de transmisión; parámetros de hardware de la tarjeta (número de serie del chip y número de la versión) y otra información que el host necesita conocer acerca de la tarjeta.

[9] Para más información sobre el protocolo T=0: http://www.info-ab.uclm.es/labelec/Solar/Otros/ISO_7816/#2_3_6_T0

2.4.2 Acceso al chip de la tarjeta

Acceder al chip de la tarjeta no es tan fácil como se puede pensar. No sirve tener sólo un lector de tarjetas. La TUI no sólo destaca por sus aplicaciones, sino también por su seguridad.

Dependiendo de donde se quiera acceder se necesitará un lector apropiado y cada lector será distinto. La diferencia entre ellos es el módulo SAM que utilice ya que es el encargado de toda la gestión de claves y la criptografía de una manera segura. Gracias a los módulos SAM es posible la autenticación y la comunicación cifrada.

Un módulo SAM (Módulo de Acceso Seguro) es físicamente otra tarjeta inteligente que puede estar conectada a una ranura lectora o puede estar insertada en un circuito integrado fijo como se muestra en la figura 2.10.

Existen los módulos SAM para la gestión de acceso e identificación. También hay otro tipo de módulos SAM de carga que sirven para cobrar/recargar dinero para el transporte cada vez que se utiliza. Los módulos SAM de aplicación son los más importantes ya que con este se pueden hacer transacciones financieras. Sólo las tienen las entidades bancarias y contienen aplicaciones criptográficas que permiten negociar las claves oportunas con la tarjeta del usuario. Estos son los más seguros y más difícil de decodificar.

Figura 2.10 Módulo SAM en tarjeta y módulo SAM insertada en un circuito integrado.

2.5 La TUI sin contacto

En el modo sin contacto, para que se realice la comunicación, la TUI no necesita estar en contacto con el lector. Se comunica a través de una antena enroscada en la tarjeta, como las figura 2.3 muestra y se transmiten los datos al lector a través de campos electromagnéticos.

Estas tarjetas sin contactos hacen que operaciones sencillas sean más rápidas que en modo con contacto. No necesitan ser insertadas en un lector para realizar la operación. Al no necesitar los contactos estos no se estropean por un uso excesivo.

Uno de sus inconvenientes es que las tarjetas deben estar a cierta distancia para el intercambio de datos. También como la tarjeta puede salirse rápidamente del rango de cobertura, sólo se puede transmitir una cantidad limitada de datos en un espacio de tiempo y, como en toda comunicación inalámbrica, es posible que las transacciones sean interceptadas sin que el titular de la tarjeta lo sepa.

A continuación se verá en qué consiste este sistema sin contactos, como es el protocolo que utiliza la TUI y algunos de sus usos.

2.5.1 RFID

En el modo sin contacto, la TUI usa el sistema RFID (*Radio Frequency IDentification*). Este sistema está compuesto por tres elementos principales (Figura 2.11) (Acosta, 2014):

- **La etiqueta RFID:** Está compuesto por un chip, un transductor y una antena la cual opera a 13,56 MHz. En el chip se encuentra la memoria de lectura o lectura/escritura. La etiqueta RFID es la parte que se encuentra en la tarjeta. En el caso de nuestra TUI, es de tipo pasivo porque no contiene ninguna fuente de energía. La señal obtenida desde los lectores induce una carga de corriente eléctrica suficiente para activar el chip y suele ser lo más económico en términos de implantación. En este sistema la TUI es la etiqueta RFID.
- **El lector RFID:** Está compuesto por una antena, un transceptor y un decodificador. Necesita una fuente de energía continua porque está continuamente activo en búsqueda de etiquetas. Los lectores mantienen a su alrededor un campo electromagnético de modo que al acercarse una tarjeta al campo, ésta se alimenta eléctricamente de esta energía inducida y puede establecerse la comunicación lector- etiqueta.
- **El sistema de procesamiento RFID:** Contiene la lógica para el procesamiento de los datos obtenidos por el lector de la etiqueta.

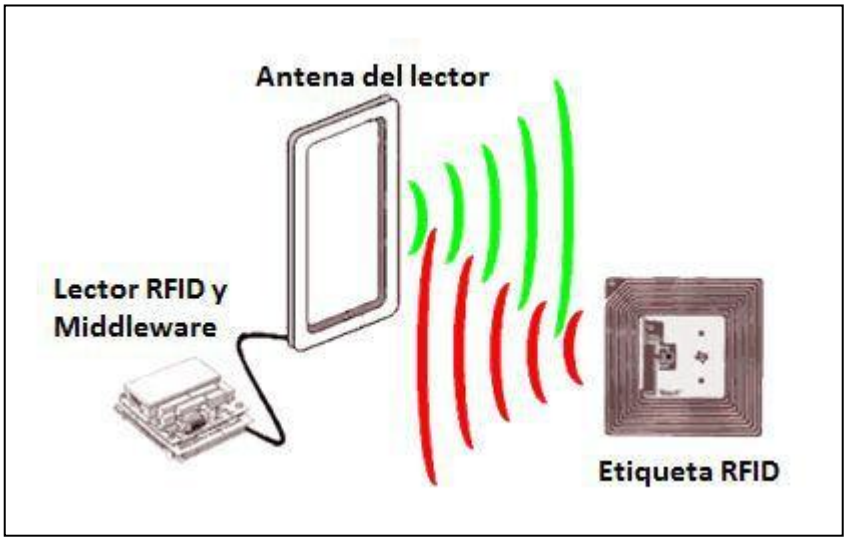

Figura 2.11 Sistema RFID. Fuente: (Acosta, 2014)

El estándar que utiliza es el ISO 14443 el cual consta de cuatro partes y se describen dos tipos de tarjetas: tipo A y tipo B. Las diferencias entre estos tipos están en los métodos de modulación, en la codificación de los planes (ISO 14443-2) y el protocolo de inicialización de los procedimientos (ISO 14443-3). Las tarjetas de ambos tipos utilizan el mismo protocolo de alto nivel llamado T=CL (ISO 14443-4). Este protocolo especifica los bloques de datos y los mecanismos de intercambio. Las tarjetas MIFARE cumplen con las partes 1, 2 y 3 del tipo A y las tarjetas Calypso cumplen con las partes 1, 2 y 3 de tipo B.

La TUI R7 cumple el estándar de tipo A, es decir, es una tarjeta MIFARE.

2.5.1.1 MIFARE

La tecnología MIFARE permite leer y escribir en una tarjeta sin necesidad de estar en contacto. Estas tarjetas son como tarjetas de memoria en algunas de sus versiones como la Classic y en otras versiones, además de ser como una tarjeta de memoria, también ofrece mayor flexibilidad, mayor seguridad (certificación EAL 4+) y mayor rapidez, como ocurre en la DESFire EV1 que también se incluye en la TUI R7.

La TUI R7 puede usar MIFARE Classic 4K y DESFire EV1 4K. La diferencia principal entre las dos es que la MIFARE DESFire EV1 tiene un nivel de seguridad mediante un motor de cifrado 3DES o AES para cifrar los datos de transmisión y también es aún más rápida llegando a conseguir una transferencia de datos de 848 kbits/s.[10]

2.5.1.2 Funcionamiento

Las MIFARE 4K ofrecen 3KB los cuales están divididos en 64 sectores de 4 bloques cada uno. De los 4 bloques, 3 pueden contener información del usuario. La información es de formato libre, y se puede modificar con comandos simples de lectura y escritura. MIFARE también tiene un formato especial llamado 'bloque de valor' en el que los bloques que tienen información guardada en este formato se comportan de una manera diferente, incluyendo operaciones de descuento e incremento.[11]

Los sectores usan dos claves de acceso 'A' y 'B'. Estas claves se almacenan en el cuarto bloque junto con los permisos de acceso a cada uno de los otros tres bloques. Los permisos pueden ser de lectura, escritura, descuento o incremento (para bloques de valor).

Cuando una tarjeta se acerca a un lector, se activa e inicia un proceso de intercambio para establecer una comunicación cifrada y está diseñado de tal forma para ofrecer protección contra escuchas y no tanto para autenticar la tarjeta o el lector en un principio.

Después de conseguir la comunicación cifrada, la tarjeta inteligente envía un código de identificación de conexión el cual generalmente es el número de serie de la tarjeta. La norma ISO 14443 indica que este número puede ser aleatorio. Una vez transmitido el

[10] MIFARE. https://www.mifare.net/en/products/chip-card-ics/mifare-classic/
https://www.mifare.net/en/products/chip-card-ics/mifare-desfire/

[11] Funcionamiento MIFARE. http://www.securityartwork.es/2010/01/29/hacking-rfid-rompiendo-la-seguridad-de-mifare-i/

código de identificación de la conexión, se envían las claves de acceso a los respectivos sectores y el lector ya está preparado para realizar cualquier operación en la tarjeta.

También utiliza distintos métodos de detección de errores para garantizar la integridad de los datos como la comprobación de redundancia cíclica (CRC) de 16/32 bits y bit de paridad.

2.5.1.3 Aplicaciones

Las aplicaciones que usan las tarjetas inteligentes en modo sin contacto suelen ser aplicaciones donde se requieren transacciones rápidas, por ejemplo, en el transporte urbano y para el control de acceso a espacios deportivos.

Actualmente, como se ha explicado anteriormente, también se puede utilizar el monedero electrónico en modo sin contactos y otras funciones que antes sólo se podía con la interfaz con contactos, gracias a la actual versión de Java Card (2.2.2) en la TUI R7, que como se ha visto anteriormente, permite acceder a nuevas funcionalidades sólo disponible antes mediante contacto y viceversa.

Las aplicaciones más usadas en la UPCT son las de acceso a espacios deportivos y el transporte urbano, pero se están proyectando otras aplicaciones para poder usar la TUI en modo sin contacto, como por ejemplo, el control de acceso a determinadas dependencias (edificios, zona de despachos, laboratorios) y el control de presencia. De este modo dependiendo del usuario puede acceder a determinados espacios y también se puede llevar un control de la asistencia, por ejemplo, a clase sin necesidad de perder tiempo en firmar en una hoja o poder entrar en una sala de estudio en horario nocturno sin necesidad de que una persona de administración se encuentre en el lugar, simplemente con el personal de seguridad habitual. Todo esto requeriría una instalación de tornos y lectores en las puertas. Más adelante se profundiza en el control de acceso en el apartado 5.4.

2.6 Evolución tecnológica del chip de la TUI

Las características que acabamos de ver son de la última versión de la Tarjeta Inteligente Universitaria, pero la TUI no siempre ha sido así. Ha estado en constante evolución, adoptando las últimas soluciones tecnológicas y adelantándose a las necesidades de los usuarios. A continuación se verá la evolución que ha seguido la TUI a lo largo de su vida en la Universidad Politécnica de Cartagena.

Las tarjetas inteligentes universitarias más destacadas que se han usado en la UPCT han sido la TUI R4, R5, R6 y R7. En los siguientes puntos se hace un pequeño resumen de cada una y como han ido cambiando tecnológicamente.

2.6.1 TUI R4

La TUI R4 (2008-2011) de Gemalto era una tarjeta híbrida la cual disponía de dos microprocesadores y memorias independientes. Esto significa que un conjunto de recursos era accesible en modo con contactos y otros servicios eran accesibles en modo sin contactos sin posible interacción entre los distintos servicios.

Figura 2.12 TUI R4

Además, el conjunto de recursos en modo con contactos sólo era accesible mediante su interfaz (Java Card). Las aplicaciones disponibles con contactos era la gestión de datos, aplicaciones financieras, de firma electrónica y de identificación. Este microprocesador de (WG10) tiene una alta seguridad homologada por la Fábrica Nacional de Moneda y Timbre. Usa criptografía simétrica (algoritmo DES) y es idóneo en aplicaciones donde se requiera capacidad y seguridad para los datos grabados en la tarjeta.

En modo sin contactos, usaba el chip MIFARE Classic de 1K y sólo era accesible mediante la interfaz IOS 14443-A, es decir, usando la frecuencia 13.56 MHz y para distancias de comunicación de hasta 10 cm. Este chip está dividido en bloques agrupados en diferentes sectores gestionados por dos claves (Key A y Key B). La aplicación de este chip para la TUI era su uso en el transporte público principalmente.

Los servicios qué ofrecía y cómo se implementaban están relacionados en la figura 2.13.

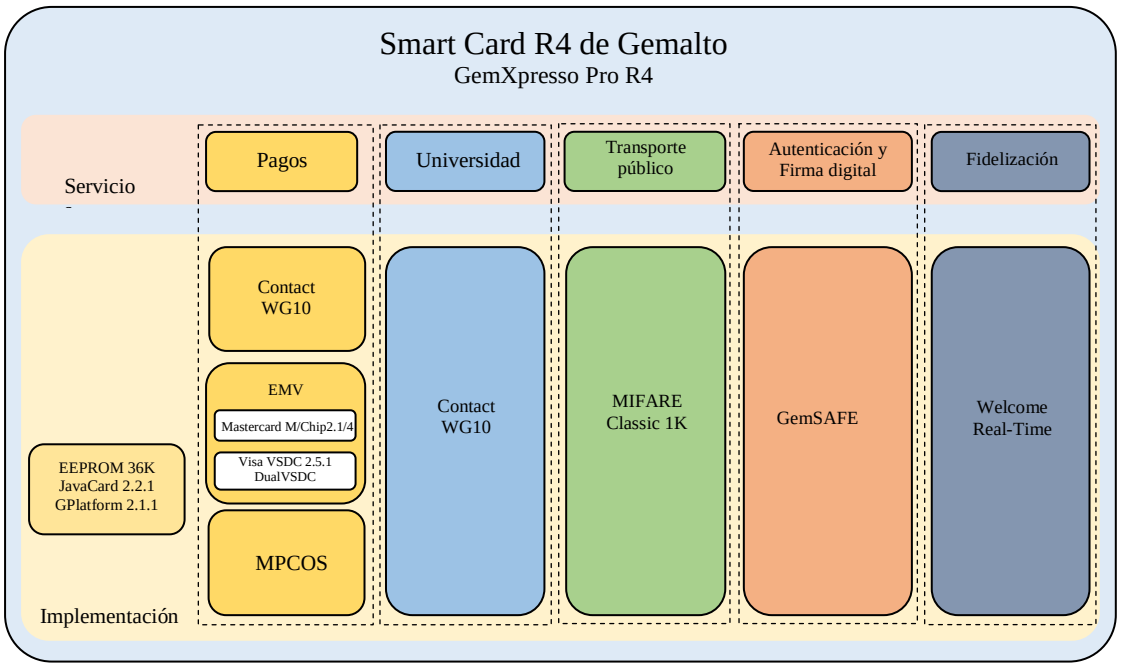

Figura 2.13 Servicios e implementación de la TUI R4

2.6.2 TUI R5/R6

La TUI R5 es la primera que integra una interfaz dual, es decir, todas las aplicaciones se sustentan en un único procesador, ya que parte de los recursos de la tarjeta (memoria y capacidad de cómputo) son accesibles tanto a través de la interfaz de contacto como de la interfaz sin contactos. Esto significa que en modo sin contactos también se dispone de las capacidades Java Card y se puede tener acceso a las utilidades que hasta entonces no se podían utilizar sin contacto con procedimientos de seguridad estándar.

Figura 2.14 TUI R5/R6

Esta versión de la TUI mejora con respecto a la R4 en términos de rendimiento y memoria disponible. También se mejoran, se complementan y amplían las aplicaciones disponibles de la anterior TUI como los usos universitarios y de pago, en el modo sin contacto el pago del transporte público y la firma electrónica. En la figura 2.15 podemos ver los servicios que ofrecía y como se implementaban en el caso de la TUI R5.

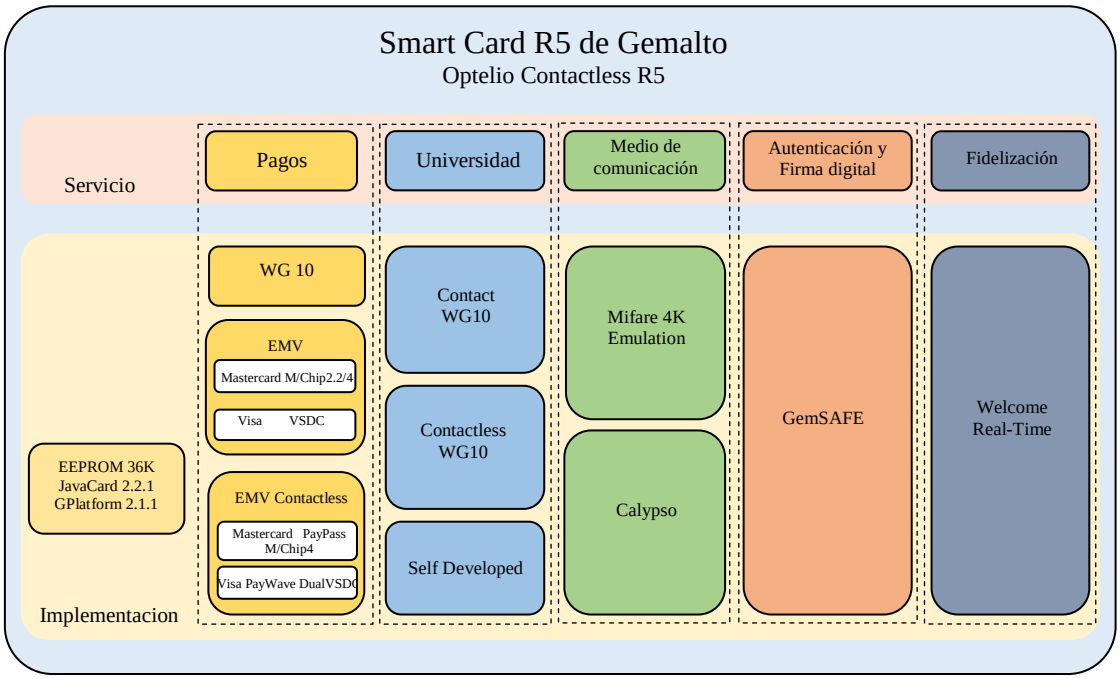

Figura 2.15 Servicios e implementación de la TUI R5/R6

La diferencia entre la R5 y la R6 es la superficie que abarca la antena en el modo sin contactos. La R5 es una "half antenna" donde la antena cubre media tarjeta y la R6 es una "full antenna" donde la antena cubre toda la superficie de la tarjeta como puede observarse en la figura 2.16. Esta diferencia era significativa debido a que la TUI R5 daba un porcentaje mayor de fallos en la transferencia de datos *contactless* y necesitaba estar más

cerca del lector. Ambas TUI R5 y R6 estuvieron distribuyéndose en la UPCT desde 2011 hasta el 2015.

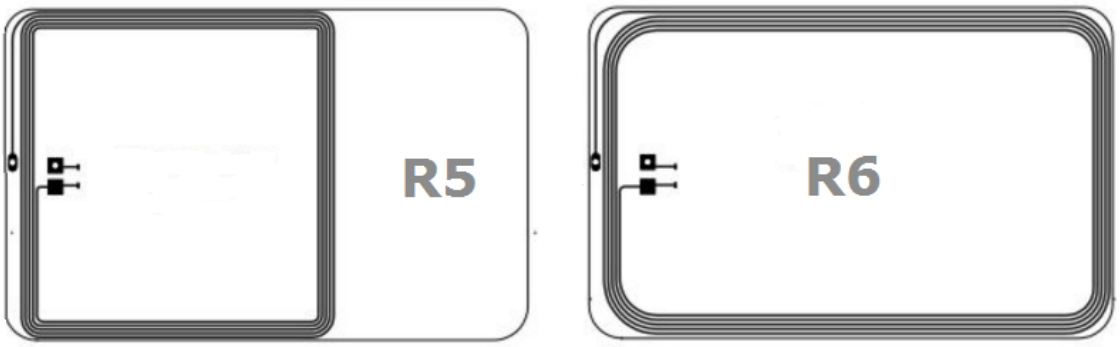

Figura 2.16 Diferencias entre las antenas de la TUI R5 y R6.

2.6.3 TUI R7

La evolución de la R5/R6 es la R7 la cual implementa la última tecnología existente relacionada con las tarjetas inteligentes, actualizando y mejorando los servicios que aporta la TUI. En la R7 se añade mayor capacidad y robustez a las funcionalidades existentes y da el soporte para nuevos servicios y funciones. En estos momentos es la tarjeta que se está implantando en la UPCT.

En la TUI R7 se actualizan aspectos de la plataforma hardware, doblando su memoria y aumentando la capacidad de procesamiento (relacionado con las funcionalidades criptográficas). También incluye una versión más actual de Java Card (2.2.2) que permite acceder a nuevas funcionalidades antes sólo disponible mediante contacto (como el monedero electrónico) y también da acceso a la memoria MIFARE en modo con contactos. Ahora el acceso a todos los recursos es total y no depende del modo en el que te comuniques con la tarjeta.

Los servicios qué ofrece y cómo se pueden implementar están relacionados en la figura 2.17.

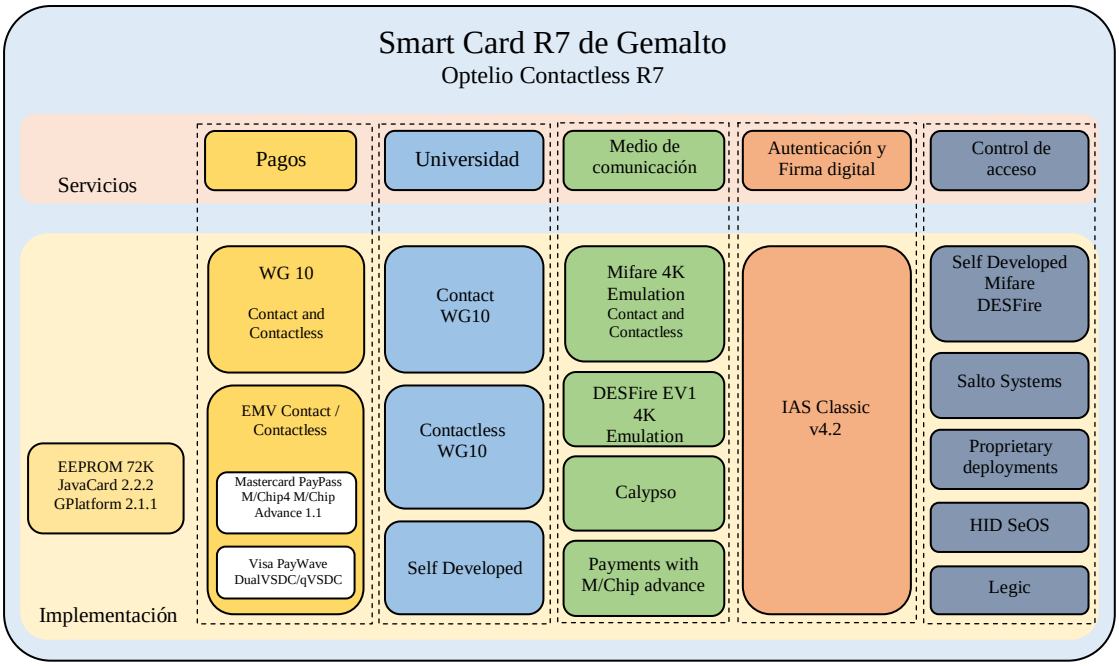

Figura 2.17 Servicios e implementación de la TUI R7

Capítulo 3. Autenticación y firma electrónica mediante la TUI

La evolución experimentada por el marco normativo exige a las Administraciones Públicas la modernización de los procedimientos en los que intervienen para adaptarse a la nueva realidad social. Es el desarrollo de las tecnologías relacionadas con la información y la comunicación (TICs) el que permite a las universidades, como al resto de organizaciones, modernizar los modelos de gestión utilizados. Esta obligación está plasmada en la Ley 11/2007, de 22 de junio, de acceso electrónico de los ciudadanos a los Servicios Públicos y "reconoce el derecho de los ciudadanos a relacionarse con las Administraciones Públicas por medios electrónicos y regula los aspectos básicos de la utilización de las tecnologías de la información en la actividad administrativa, en las relaciones entre las Administraciones Públicas, así como en las relaciones de los ciudadanos con las mismas con la finalidad de garantizar sus derechos, un tratamiento común ante ellas y la validez y eficacia de la actividad administrativa en condiciones de seguridad jurídica"[12], es decir, autenticándose de manera segura utilizando la firma electrónica.

La TUI también viene preparada para la firma electrónica y la autenticación del usuario y se convierte en '**algo que posee el usuario**' de los factores de seguridad para la autenticación (Rouse, 2014). Estos factores se clasifican en tres y son:

- Lo que **sabe** el usuario (por ejemplo, una contraseña, una frase o un PIN).
- Lo que **posee** (como la TUI o el teléfono móvil cada vez más usado para la identificación).
- Lo que **es** (la huella digital, el patrón retiniano o el rostro por ejemplo).

Utilizar la TUI y además un PIN se convierte en una autenticación de dos factores siendo más segura que si se utilizara sólo un factor. Al igual que si se utilizan tres factores como por ejemplo, la TUI, el PIN y la huella dactilar del usuario da lugar a una autenticación multifactor inequívoca en el dispositivo.

Además, en los próximos años, se va a notar la entrada en vigor de la Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Común de las Administraciones Públicas la cual pretende implantar una Administración totalmente electrónica, interconectada y transparente, mejorando la agilidad de los procedimientos administrativos y reduciendo los tiempos de tramitación[13].

Por este motivo, se analiza en qué medida la TUI contribuye a que las universidades sustituyan los medios de tramitación convencionales basados en el uso masivo de papel por una tramitación realizada completamente a través de medios electrónicos. Para ello, a continuación vamos a ver el certificado electrónico y la firma electrónica usando la TUI.

[12] España. Ley 11/2007, de 22 de junio, de acceso electrónico a los ciudadanos a los Servicios Públicos. Boletín Oficial del Estado, 23 de junio de 2007, nº 150, pp. 27150-27166

[13] España. Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Común de las Administraciones Públicas. Boletín Oficial del Estado, 2 de octubre de 2015, nº236, pp. 89343-89410.

3.1 Certificado electrónico

El certificado digital es un documento digital que contiene, entre otros, los datos identificativos del propietario. Permite realizar trámites de forma segura a través de Internet[14]. Gracias al certificado electrónico el usuario no necesita desplazarse para realizar trámites y se olvida de esperas innecesarias.

Un certificado electrónico tiene dos claves, una pública y una privada. La pública es la única que viaja en las transacciones telemáticas y la privada nunca sale de la tarjeta. Con la clave pública se puede autenticar al usuario y con la clave privada se pueden realizar las firmas electrónicas.

En la Universidad Politécnica de Cartagena continuamente se están desarrollando servicios que permitan el uso de la firma electrónica. Para firmar un documento es necesario disponer de un certificado electrónico emitido por un proveedor de servicios de certificación y en la UPCT se obtienen los certificados digitales de la Autoridad de Certificación de la Comunidad Valenciana (ACCV).

La TUI se constituye como una tarjeta criptográfica que ofrece la posibilidad de almacenar este certificado electrónico, convirtiéndose en un dispositivo que proporciona una mayor seguridad a la hora de desarrollar el proceso de firma electrónica. También para la identificación ante servicios telemáticos.

Al contrario que poseer un certificado digital en soporte software, como en un ordenador, en una tarjeta criptográfica el certificado (clave privada) no puede salir de la tarjeta ni copiarlo y, por lo tanto, es más seguro.

Una de las desventajas de usar las tarjetas criptográficas con el certificado almacenado es que para hacer uso del certificado se requiere utilizar dispositivos adiciones, como es un lector de tarjetas.

3.2 Firma electrónica

La firma electrónica es una herramienta imprescindible para poder llevar a cabo la tramitación electrónica de los procedimientos. Posibilita la total movilidad de los agentes que intervienen en la gestión administrativa, ya que no es necesaria su presencia física. Según el artículo 3.1) de la Ley 59/2003, de 19 de diciembre, de la firma electrónica, "La firma electrónica es el conjunto de datos en forma electrónica, consignados junto a otros o asociados con ellos, que pueden ser utilizados como medio de identificación del firmante"[15]. Y distingue entre dos tipos de firmas electrónicas:

- Firma electrónica avanzada: (Art.3.2) "La firma electrónica avanzada es la firma electrónica que permite identificar al firmante y detectar cualquier cambio ulterior de los datos firmados, que está vinculada al firmante de manera única y a los datos a que se refiere y que ha sido creada por medios que el firmante puede mantener bajo su exclusivo control"[16].

[14] ESPAÑA. Ley 59/2003, de 19 de diciembre, de Firma Electrónica. Boletín Oficial del Estado, de 20 de diciembre de 2003, nº 304, pp. 45329-45343
[15] Ibid. p. 45332.
[16] Ibid. p. 45332.

- Firma electrónica reconocida: (Art.3.3) "Se considera firma electrónica reconocida la firma electrónica avanzada basada en un certificado reconocido y generada mediante un dispositivo seguro de creación de firma"[17]. Y según el artículo 3.4), "La firma electrónica reconocida tendrá respecto de los datos consignados en forma electrónica el mismo valor que la firma manuscrita en relación con los consignados en papel"[18].

El número medio de firmas electrónicas en el curso académico 2012-2013 supera las 12.000 en el 15% de las universidades españolas que informó sobre su uso en el Estudio sobre la TUI del Banco Santander. Además la mayoría de las universidades informó que les gustaría implantar más servicios en los que se pueda usar la firma electrónica, lo que permite afirmar que es un uso de la TUI en pleno proceso de crecimiento.

3.2.1 Firma electrónica reconocida frente avanzada

Se ha hablado sobre dos tipos de firmas electrónicas y, a continuación, se van a ver las diferencias entre ellas y cómo saber que firma electrónica estamos usando.

Ambas firmas se realizan a través del ordenador pero para que una firma electrónica sea considerada reconocida debe cumplir las siguientes propiedades o requisitos[19]:

- Identificar al firmante.
- Verificar la integridad del documento firmado.
- Garantizar el no repudio en el origen.
- Contar con la participación de un tercero de confianza.
- Estar basada en un certificado electrónico reconocido.
- Debe ser generada con un dispositivo seguro de creación de firma.

Los cuatro primeros puntos son posibles gracias al uso de las claves criptográficas contenidas en el certificado digital y a la existencia de las Autoridades de Certificación que ofrecen confianza en la entrega de los certificados digitales. Estos cuatro primeros puntos sólo nos ofrecen una firma electrónica avanzada.

Para que una firma electrónica sea considerada equivalente a la manuscrita, es decir, firma electrónica reconocida, debe además:

- **Estar basada en un Certificado Reconocido** y estos son aquellos reconocidos por el Ministerio de Industria y Comercio como habilitado para crear firmas reconocidas.
  Se les llama certificados reconocidos porque tanto el prestador que los emite como el contenido mismo del certificado, cumplen con los requisitos declarados en el Capítulo II de la Ley 59/2003 de firma electrónica sobre Certificados reconocidos.

- **Debe ser generada con un dispositivo seguro de creación de firma (SSCD)**, en nuestro caso la TUI. Las características de un dispositivo seguro de creación de firma principalmente son:

[17] Ibid. p. 45332.
[18] Ibid. p. 45332.
[19] Portal de Administración Electrónica, Gobierno de España. Base legal de la Firma Electrónica. http://firmaelectronica.gob.es/Home/Ciudadanos/Base-Legal.html#firma_electronica

- Que el dispositivo debe garantizar que las claves sean únicas y secretas.
- Que la clave privada no se pueda deducir de la pública y viceversa.
- Que el firmante pueda proteger de forma fiable las claves.
- Que no se altere el contenido del documento original.
- Que el firmante pueda ver qué es lo que va a firmar.

Todas estas características están recogidas en el artículo 24 de la Ley 59/2003 de Firma Electrónica y la TUI cumple con todas ellas ya que:

- Los pares de claves para los certificados se generan en la tarjeta criptográfica del usuario y nunca abandonan la misma.
- Las claves privadas para los certificados se encuentran contenida en la tarjeta criptográfica que se entrega al suscriptor con su certificado en el momento de su registro.
- La clave pública a ser certificada es generada en el interior de la tarjeta criptográfica y es entregada a la Autoridad de Certificación por la Autoridad de Registro mediante el envío de una solicitud de certificación, firmada digitalmente por el Operador de la Autoridad de Registro.

En cambio, las firmas generadas en el ordenador con un certificado software instalado en el navegador no se consideran firma electrónica reconocida porque el ordenador no es un dispositivo seguro de creación de firma. Esas firmas serían sólo avanzadas según la definición de la ley.

3.3 Applet IAS Classic V4.2

Para la autenticación y la firma electrónica la TUI utiliza la aplicación IAS[20] Classic V4.2. Esta aplicación se puede utilizar para la autenticación fuerte, las firmas electrónicas y el almacenamiento de certificados. Es la aplicación Java Card que hace que sea la TUI un dispositivo seguro de creación de firma como se define en la Directiva 1999/93/EC del Parlamento Europeo y del Consejo del 13 de diciembre de 1999, sobre un Marco comunitario para la firma electrónica.

El applet es compatible con PKCS#15[21] y todas las aplicaciones que usen el PKCS#11[22] y/o las de Microsoft CAPI. El PKCS#15 define un estándar que permite a los usuarios de dispositivos criptográficos identificarse con aplicaciones independientemente de la implementación del PKCS#11 u otro API. Siendo de este modo aún más compatible.

Este applet ofrece:

- Servicios de canal de confianza para asegurar las comunicaciones con las entidades generadoras de certificados y las aplicaciones creadoras de firmas.
- Generación del par de claves.
- Exportación de la clave de verificación de firma (necesario si se genera la clave privada dentro de la TUI) y la recepción/almacenamiento de certificados.

[20] IAS: Identification Authentication Signature
[21] Public-Key Cryptography Standards#15. Estándar de formato de información de dispositivo criptográfico.
[22] Public-Key Cryptography Standards#11. Interfaz de dispositivo criptográfico. Define un API genérico de acceso a dispositivos criptográficos.

- Inicialización de los datos de autenticación (por medio de PIN).
- Creación de firma electrónica.
- La verificación de los certificados.

Esta aplicación es capaz de gestionar simultáneamente varios pares de claves generados dentro de la TUI o importados desde fuera, o una combinación de ambos. En estos casos, la TUI le permite al usuario elegir entre las que tenga antes de firmar.

También este applet funciona para el modo de contacto y sin contacto.

Capítulo 4. Medio de pago electrónico

Usar la TUI como soporte de diferentes medios de pago es una funcionalidad que va más allá de la identificación. La TUI en estos momentos permite el uso de la misma como monedero electrónico o como tarjeta financiera (opcional). También es posible asociarla a una cuenta virtual como veremos más adelante.

Estos métodos de pago permiten que se reduzcan los costes de transacción y de tiempo, garantizando así mismo una mayor eficiencia, eficacia y transparencia del sistema. Del mismo modo se facilitan las tareas de conciliación y tanto el usuario como el prestador de bienes o servicios dispondrían de mayor información.

Hoy en día, el monedero electrónico cada vez está más en desuso debido a que es necesario disponer de un terminal adecuado para el cobro y esto es un gasto a considerar. También influye la diversidad de sistemas de monedero (monedero 4B, monedero Euro 6000, monedero VisaCASH,…) porque no son interoperables entre sí, es decir, o dispone el comerciante varios terminales distintos o el usuario debe tener varias tarjetas monedero de distintos sistemas. Esta última desventaja llevó a crear unas especificaciones comunes para un monedero electrónico. Los más destacables son EMV (Europay-Mastercard-Visa) y CEPS (Especificaciones Comunes del Monedero Electrónico). De todos modos, el monedero electrónico no ha tenido el éxito esperado. En cambio, se está hablando más sobre el monedero virtual pero este tema se verá más adelante.

Ambos servicios, monedero electrónico y tarjeta financiera, pueden ser usados con la TUI en modo con contactos y sin contactos gracias a la versión más actual de Java Card que, como se indicó anteriormente, permite acceder a nuevas funcionalidades antes sólo disponibles mediante la interfaz de contacto o en la interfaz sin contacto.

4.1 Monedero electrónico de la TUI

El monedero electrónico intenta reproducir en formato electrónico las características del dinero en efectivo, como el anonimato en los pagos, los costes de transacción son nulos y es difícil de copiar. La operación se realiza off-line. Evita que cada pago precise de una conexión a la red bancaria para ser autorizado, de este modo la transacción es más rápida.

4.1.1 Funcionamiento

Los pagos individuales se realizan entre un tipo de sistema de cuentas “privadas” y el dinero en estas cuentas puede utilizarse sólo en este sistema.

El usuario carga el dinero en su tarjeta chip. Internamente, cajero y tarjeta se autentican mutuamente y la tarjeta permite que el cajero incremente el saldo (escritura en fichero). Puede hacerse en un cajero de la entidad o también se podría realizar en máquinas destinadas para la recarga de las tarjetas monedero contra efectivo pero en la UPCT no se disponen de ellas.

Al realizar el pago no hay conexión al banco. El vendedor tiene un lector de tarjetas. Este lector y la tarjeta se autentican mutuamente y la tarjeta permite que su saldo sea decrementado. El lector de tarjetas incrementa el saldo del vendedor y periódicamente el lector se comunica con el banco y actualiza la cuenta del vendedor, por ejemplo, una vez al día.

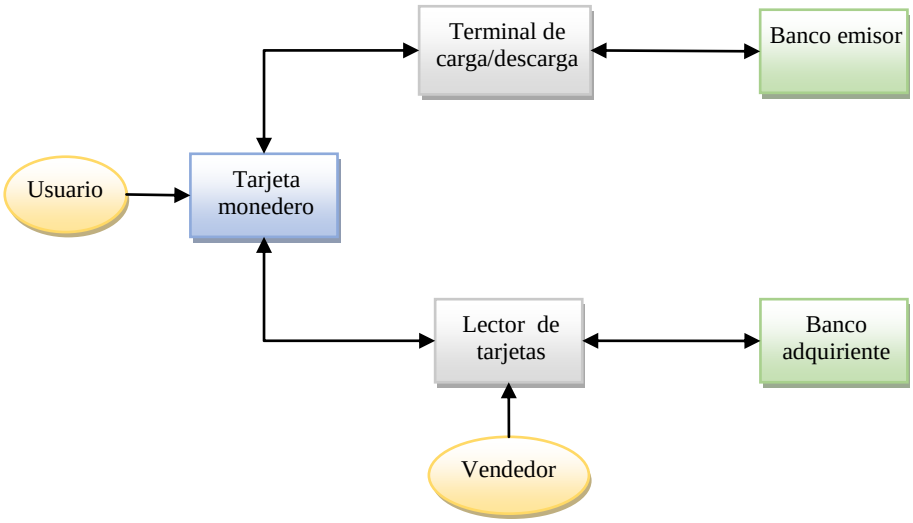

Figura 4.1 Esquema de conexiones de la tarjeta monedero

4.2 Tarjeta financiera

Actualmente la TUI también puede ser emitida por el Banco Santander como una tarjeta financiera común. Permitiendo al titular de la TUI usarla como medio de pago vinculado a su cuenta bancaria. Este servicio es opcional.

La TUI ya no usa la banda magnética en la que se almacenaban los datos de la tarjeta de pago (en texto claro). Ahora todos los datos para esta función se almacena en el chip de forma cifrada debido a que estos circuitos permiten la gestión de una lógica específica permitiendo el uso de diferentes algoritmos de encriptación (DES, TripleDES, RSA y SHA) siendo más seguro el pago.

4.2.1 EMV

Para estandarizar este nuevo modo de pago con chip, EuroPay, MasterCard y Visa conformaron EMVCo, consorcio al que posteriormente se unirían Discover, JCB, American Express y UnionPay.

El objetivo de este grupo fue crear una serie de elementos comunes (físicos, eléctricos, datos y aplicaciones) que permitieran la interoperabilidad entre las tarjetas de pago con chip y los lectores, usando como base el estándar ISO/IEC 7816.

4.3 Monedero virtual

El monedero virtual es una nueva modalidad de pago. Es un sistema de pago electrónico que permite a cada titular disponer de una cuenta virtual con la que abonar unos servicios prestados por una empresa o universidad y no requiere la colaboración de entidades financieras. Aunque hay entidades financieras que también ofrecen este servicio muy similar a un monedero electrónico pero sin el obstáculo de estar vinculada la cuenta sólo a la tarjeta inteligente como ocurre con el monedero electrónico.

En la UPCT se encuentra implantado este servicio y hay muchos más ejemplos de cuentas virtuales como PayPal, Google Wallet, agencias de viajes como PriceTravel, varias entidades financieras como, por ejemplo, el Banco Santander.

4.3.1 Funcionamiento

El cliente necesita tener una cuenta virtual. En ella se ingresa dinero mediante un banco, tarjeta o un ingreso al contado, dependiendo del funcionamiento de la cuenta virtual. Una vez que en la cuenta se dispone de saldo, se puede usar para hacer pagos al contado en medios on-line no presenciales y en establecimientos que admitan este medio de pago por Internet siendo servicios prestados por la empresa o universidad de la cuenta virtual. Para estas transacciones siempre se necesita internet para comprobar que la cuenta virtual dispone de saldo.

En la actualidad, la Ley 21/2011, de 26 de julio, de dinero electrónico regula la emisión de dinero electrónico. Sin embargo, según el artículo 1 apartado 3, esta Ley no se aplica a los monederos virtuales que pueden vincular las universidades a la TUI y que difieren de las monedas virtuales que en los últimos años han proliferado. Ver anexo.

En la UPCT ya se encuentra este servicio implantado pero su uso está destinado únicamente al pago de actividades deportivas o pago de reservas de las instalaciones deportivas. Como futuro proyecto, se podría usar la TUI como identificador en este servicio de cuentas virtuales usando el código QR o se podría usar el modo sin contacto de la TUI. Hay un abanico de posibilidades para realizar este servicio y relacionar la TUI con las cuentas virtuales de la UPCT. También se desea implantar el pago de más servicios como cursos, pago de certificados, la impresión de documentos y la compra de entradas para eventos de la UPCT.

4.3.1.1 Google Wallet

Un ejemplo de cuenta virtual a destacar es la Google Wallet. En septiembre de 2011 Google lanzó su servicio de monedero virtual. Esta cuenta virtual almacena de forma segura las tarjetas de crédito, las tarjetas de débito, cheques regalo, tarjetas de socio, ofertas y más. Se puede comprar en tiendas, comprar online y enviar dinero mediante correo electrónico. También los usuarios pueden solicitar de forma gratuita una tarjeta de débito prepago, Google Wallet Card, vinculada a la cuenta de monedero virtual y con ella

se puede comprar en tiendas y retirar dinero en efectivo en cajeros automáticos en los sitios que admiten las tarjetas MasterCard.

Puedes comprar y aprovecharte de ofertas en las tiendas con el almacenamiento de cheques regalo, tarjetas de socio y ofertas en la aplicación de Google Wallet. También se puede comprar en las tiendas que utilizan la Tarjeta Monedero Google o pagar mediante NFC si tienes un dispositivo Android con NFC. Esta función sólo está disponible en Estados Unidos.

También sólo está aún disponible en Estados Unidos el servicio de envío de dinero a través de email. Este servicio se puede realizar con cualquier persona con una dirección de correo electrónico con la aplicación de Google Wallet activa. También se puede solicitar dinero a amigos o familiares con la misma condición anterior.

Dentro y fuera de los Estados Unidos (en más de 125 países) se pueden hacer compras en línea en Google Play y otros productos de Google, a través de Google Wallet, donde se vea un botón de *Comprar con Google.* Por ejemplo: comprar aplicaciones, dispositivos o películas.

En estos momentos, la aplicación para el móvil de Google Wallet sólo está disponible en los Estados Unidos.

Capítulo 5. Usos de la TUI

La TUI se puede integrar con multitud de servicios además de los vistos en una universidad y de manera diferente según la universidad de que se trate. Esto se debe al hecho de que en la integración de la TUI con los servicios intervienen multitud de factores, como son la tipología de la universidad (pública/privada, presencial/a distancia), su tamaño o la estrategia TI de la universidad.

Hay que distinguir también entre servicios universitarios y servicios externos (descuentos en comercios,…), en cuanto a tipos de servicios se refiere, según estén relacionados con la actividad interna de la universidad o con el acceso a servicios proporcionados por empresas externas.

A continuación, se van a analizar algunos servicios universitarios y no externos ya implantados y otros en desarrollo relacionados con la TUI en la Universidad Politécnica de Cartagena.

Algunos de estos servicios estarán vinculados exclusivamente a la TUI (sin ella no se puede usar el servicio) y otros tendrán una vinculación compartida (existen otros canales de acceso al servicio). Por ejemplo el control de horario y marcajes tiene una vinculación exclusiva y el servicio de préstamo de libros tiene una vinculación compartida. En el servicio de préstamo de libros únicamente se necesita identificar a la persona. Si el usuario tiene su DNI también puede hacer uso del servicio de préstamos. En cambio, con el servicio de control de horario sólo se puede usar la TUI porque se hace con el chip de la TUI en los puntos de marcaje instalados.

5.1 Préstamo de libros

Los alumnos usan la TUI principalmente para el préstamo y recogida de libros en las bibliotecas. Sin embargo, como se limita a una función de identificación, en algunos casos la TUI está siendo sustituida por el DNI como se ha dicho anteriormente (vinculación compartida). Esto es debido a que el código de barras de la TUI coincide con el número de DNI.

El modo de identificar al usuario se realiza a través de la lectura del código de barras del reverso de la TUI. Hay que señalar que no se utiliza el código de barras para otro servicio actualmente. Próximamente también se podrá realizar el préstamo de libros con el código QR implantado en septiembre del 2015 en la UPCT.

En la mayoría de las universidades españolas, tener la TUI es de carácter obligatorio para el préstamo de libros o mostrar la matrícula del curso académico actual. En muy pocas universidades españolas, la identificación del usuario en las bibliotecas se realiza con el chip de la TUI. Hacerlo de este modo supondría tener los lectores apropiados en cada biblioteca y sustituir a los lectores de código de barras actuales.

5.2 Control horario

El control horario es el segundo servicio universitario más integrado con la TUI. Los

sistemas de control de horario se aplican al colectivo de personal de administración y servicios (PAS), pero podría integrarse también en el resto de la comunidad universitaria. Por ejemplo, mediante la TUI se podría realizar un seguimiento de la asistencia a clase, lo que permitiría cuantificar ratios de aprovechamiento basados en la asistencia a clase o la intensidad de uso de espacios.

Actualmente, el registro se hace mediante el chip de la TUI exclusivamente pero también está preparada para hacer esta función sin contacto como hemos explicado anteriormente, gracias a la última versión de la TUI y la versión de Java Card 2.2.1. Simplemente se pasaría la tarjeta cerca de un terminal para uso sin contacto. Todo esto se debe a las características antes mencionadas del modo sin contacto de la TUI. En estos momentos no hay ningún lector sin contacto para este servicio.

5.3 Gestión de la impresión

Otro servicio universitario sería la gestión de la impresión. Este servicio en la Universidad Politécnica de Cartagena no se ofrece. Las copisterías de la UPCT son las encargadas sin dar opción a imprimir en las aulas de informática. Esta solución pospone la necesidad de integrar la TUI en la gestión de la impresión.

Actualmente, el 40% de las universidades españolas usan la TUI en la gestión de la impresión teniendo lectores en las fotocopiadoras y dando la posibilidad de pagar con el monedero electrónico o con la propia cuenta virtual de la universidad.

5.4 Control de accesos

Las universidades tienen numerosas infraestructuras de muy diversa naturaleza destinadas a la comunidad universitaria. Debido a sus características, algunas de ellas requieren el acceso restringido y/o controlado. El uso de la TUI por parte de la comunidad universitaria para el acceso a las diferentes infraestructuras de que disponen las Universidades es otra de sus principales funcionalidades. La TUI permite aglutinar en un soporte una identificación y una llave que permite el acceso a las infraestructuras como se ha visto en el apartado de autenticación y firma electrónica. Esto contribuye al control automatizado para alcanzar niveles de seguridad mayores en las instalaciones.

De su implantación también se pueden generar ventajas importantes posibilitando, por ejemplo, recabar información acerca de la intensidad de uso de las infraestructuras, como frecuencia o niveles de ocupación, características de los usuarios como la pertenecía o no a un determinado colectivo y procesar la información para interactuar con otras aplicaciones.

Las principales infraestructuras de la Universidad Politécnica de Cartagena que usará la TUI en un futuro muy cercano para el control de acceso son las puertas de acceso a edificios y algunas instalaciones deportivas que todavía no disponen de este servicio. Se podrán usar en tornos o con lectores en las puertas y la TUI se podrá usar mediante contacto mediante el chip o sin contacto.

En resumen, se puede concluir que la UPCT se encuentran dentro de un proceso de transición, intentando dotar a su personal y alumnos de las herramientas necesarias para que se pueda llevar a cabo una gestión completa de los trámites administrativos utilizando medios electrónicos. Se trata de un campo de actuación en el que la TUI puede ofrecer

soluciones y funcionalidades que faciliten a la UPCT su adaptación a la nueva realidad social, potenciando su papel de integrador de servicios dentro del mundo universitario.

5.5 Control de asistencia a clase

Este servicio aún no se ofrece en la UPCT pero está en creciente auge en las universidades españolas. Esto se debe a que el Espacio Europeo de Educación Superior implica una evaluación continua de los alumnos y para ello necesitan saber la asistencia de los alumnos y para ello en estos momentos se hace con una hoja de firmas.

Este es un futuro proyecto de la UPCT que pronto se llevará a cabo. Su funcionamiento sería muy similar al del control de accesos, permitiendo controlar la asistencia y de este modo evitando la pérdida de tiempo invertido en el tradicional proceso de firma (además del consecuente ahorro en papel).

5.6 Valoración sobre la integración de la TUI en los servicios universitarios

Como conclusión a este apartado de servicios universitarios, el grado de integración de la TUI se puede considerar como medio, a excepción del uso generalizado de la TUI para la identificación del usuario en el préstamo de libros en las bibliotecas de la UPCT y el control de horario para el personal de administración y servicios (PAS).

Con la nueva Ley 39/2015 pronto su uso será mayor para poder autenticarse, firmar y realizar las gestiones oportunas en la UPCT y de este modo evitar desplazamientos y pérdidas de tiempo en esperas.

También se está trabajando actualmente para que la TUI esté vinculada a más servicios que respondan a las necesidades requeridas por los usuarios, como por ejemplo, para los alumnos con el Plan Bolonia y la asistencia obligatoria a clase. De esta manera se ofrecería el valor añadido de eliminar cualquier proceso manual asociado ahorrando tiempo y papel. Otro uso que sería muy popular es el de instalar tornos en las aulas de estudio para acceder a ellas en horarios no habituales, de este modo siempre se sabría qué usuarios están disfrutando de las instalaciones y en qué horas, evitando además el acceso a personas ajenas a la universidad.

5.7 Sinergias con otros entes

El valor añadido de la TUI consiste en los servicios que ofrece a sus titulares y en su ampliación así como en la búsqueda de sinergias con terceras entidades. Por ejemplo, algunos proyectos que se están llevando a cabo en algunas universidades es la de utilizar la TUI en algunos servicios municipales uniendo la TUI con las tarjetas del ayuntamiento, como por ejemplo, su uso en el transporte público, la retirada de bicicletas o acceso a las piscinas públicas y centros deportivos municipales entre muchos otros servicios.

La unión de tarjetas no solo amplia los servicios disponibles sino que también complementa y aumenta las infraestructuras a disposición de la comunicada universitaria como son bicicletas, instalaciones deportivas, etc.

Capítulo 6. Los códigos QR

6.1 Introducción

Hoy en día los códigos QR están cada vez más introducidos en nuestra vida cotidiana y con multitud de aplicaciones. Tal avance incluso ha hecho que en cada TUI se introduzca un código QR para usarlo para diversos servicios.

En este capítulo, se hablará sobre los códigos QR (qué son, cuáles son sus principales aplicaciones) y se desarrollarán tres casos prácticos para usar en la UPCT donde se utilizan códigos QR.

En primer lugar, un código QR (*quick response code*, código de respuesta rápida) es un módulo útil para almacenar información en una matriz de puntos o un código de barras bidimensional (Estebanell, 2012).

Se caracteriza por los tres cuadrados que se encuentran en las esquinas y que permiten detectar la posición del código al lector (ver figura 6.1). Son tres patrones idénticos de 7x7 módulos que le dicen al lector la posición, las dimensiones y la orientación del código QR. Se muestra en verde en la figura 6.1.

También contiene un cuadrado más pequeño que los anteriores el cual sirve como patrón de alineación. Un cuadrado de 5x5 módulos que ayudan al lector a detectar deformaciones de la superficie. Su cantidad depende del tamaño de QR, los más sencillos tienen sólo uno. Se muestra en azul en la siguiente figura.

Se puede observar también un patrón de sincronía. Son dos reglas de módulos blancos y negros alternándose que le indica al lector la densidad de datos y le sirven como ejes de coordenadas. Se muestra en amarillo en la siguiente figura.

Figura 6.1 Ejemplo de código QR

En otros de sus componentes podemos encontrar información de formato, bloques de datos e información de la versión, bloques de corrección de errores y zonas de silencio. Las zonas de silencio, según la especificación, deben de estar alrededor de un código QR. Deben tener cuatro módulos de ancho y ser de color blanco. Esto garantiza que el código es legible.

Fue creado en 1994 por la compañía japonesa Denso Wave, una filial de Toyota. Su nombre <<Quick Response>> se debe a su origen, pues los creadores (un equipo de dos personas en Denso Wave, dirigido por Masahiro Hara) tenían como objetivo que el código permitiera que su contenido se leyera a alta velocidad. Inicialmente se usó para registrar repuestos en el área de la fabricación de vehículos. [23]

Hoy en día, los códigos QR se usan para administración de inventarios en una gran variedad de industrias y, además, con la inclusión de software que lee códigos QR en teléfonos móviles, se han permitido usos más orientados al consumidor, que se manifiestan en comodidades como el dejar de tener que introducir datos de forma manual en los teléfonos.

6.2 Aplicaciones

En las revistas y anuncios publicitarios se están volviendo más comunes para ingresar la página web o una URL sin necesidad de escribir. También se está haciendo muy común agregar un código QR en las tarjetas de presentación simplificando el ingreso de datos de un nuevo contacto, por ejemplo, un cliente en tu agenda de contactos.

Una de sus mayores utilidades es la de sustituir una tarjeta de embarque, una entrada de cine, cupones descuento, etc. Cada vez más empresas incluyen la posibilidad de recibir sus billetes o entradas en los smartphones usando aplicaciones llamadas carteras virtuales, por ejemplo, PassBook (iPhone), Pass Wallet y Pass2U (Android). Estas carteras virtuales te permiten organizar los billetes y tarjetas de embarque, las entradas de cine, teatro y conciertos, tarjetas de socio, cupones, ofertas, etc.

Un detalle importante respecto a los códigos QR es que, a diferencia de otros formatos bidimensionales como el BIDI, su código es abierto y sus derechos de patente (propiedad de Denso Wave) no son ejercidos.[24]

6.2.1 Autenticación con código QR

La aplicación SQRL (Secure Quick Reliable Login) es un nuevo método de autenticación en el que no es necesario usar usuario y contraseña para identificarse en un sitio web. Ideado por Steve Gibson. Simplemente debes tener instalada la aplicación SQRL.[25]

El funcionamiento de este sistema para un usuario es muy simple:

[23] http://www.qrcode.com/en/history/ (Consultado en junio 2015)
[24] Estebanell, M., Ferrés, J. y Gamboa, J.L. (2012), *Tendencias emergentes en educación con TIC*. Barcelona: Asociación Espiral, Educación y Tecnología. Págs. 197-209.
[25] https://www.grc.com/sqrl/sqrl.htm (Consultado en junio 2015)

- Un usuario ejecuta en un dispositivo la aplicación SQRL, y escanea el código mostrado.
- La aplicación SQRL muestra el nombre del dominio contenido en el código SQRL para que el usuario se asegure de que es correcto.
- Después el usuario permite a la aplicación SQRL que lo autentique.
- Y sin poner ni usuario ni contraseña ya está autenticado.

Figura 6.2 Pantalla de autenticación con SQRL

Es bastante simple pero es mucho más seguro este método y ahora veremos el porqué. El funcionamiento internamente es:

- El código QR situado en la página de autenticación contiene la URL del servicio de autenticación para el sitio. La URL incluye un número aleatorio largo generado de forma segura (nonce) por lo que cada vez el código QR será distinto.
- La aplicación SQRL del dispositivo del usuario encripta el nombre de dominio por la clave maestra del usuario para producir un par de claves pública/privada site-specific.
- La aplicación firma criptográficamente la URL completa escaneada del código QR con la clave privada.
- La aplicación envía una consulta HTTPS POST a la URL del código QR, la cual es al servicio de autenticación del sitio. El POST lleva la clave pública especifica obtenida y la firma criptográfica de la URL del código QR que se ha encriptado antes.
- El sitio web recibe y afirma la llegada del POST devolviendo una respuesta estándar HTTP “200 OK” sin otro contenido. La aplicación SQRL acepta la solicitud exitosa del código QR del usuario identificado.
- El servicio de autenticación tiene la URL que contiene el nonce. También tiene la firma criptográfica de esa URL, y la clave pública site-specific del usuario. Con esto, el servicio de autenticación usa la clave pública para verificar que la firma es válida. Esto confirma que el usuario que produjo la firma usó la clave privada que corresponde a la clave pública. Después de verificar la firma, se reconoce al usuario, ya autenticado, por la clave publica site-specific.

Una de las ventajas de usar SQRL es que no se necesita ni usuario ni contraseña para cada lugar al que se acceda. Estos pueden ser robados o pueden olvidarse debido a la cantidad de contraseñas diferentes para cada sitio que hay que recordar. Tampoco tendría servicio de acceso de terceros a partir de Facebook, Twitter, Google y Microsoft debido a que estos sistemas también tienen una debilidad, por ejemplo, si tu cuenta de Facebook es

comprometida, todos los sitios que se conecten con ese nombre de usuario también se verán comprometidos. Otra de las ventajas es que este método es inmune a un ataque de fuerza bruta para contraseñas.

El SQRL está comenzando a ganar fuerza y desde el lado del usuario no podría ser más sencillo.

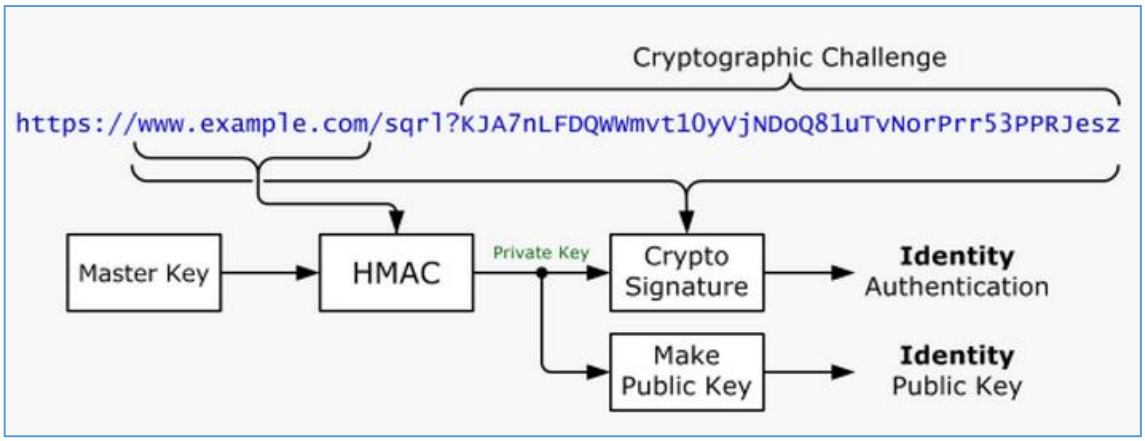

Figura 6.3 Esquema SQRL

6.3 Aplicaciones y servicios basados en QR en las universidades

6.3.1 Implantación de un código QR en la TUI

En la UPCT se ha implantado un código QR identificativo en la TUI. Además del código QR también se ha añadido soporte criptográfico para certificados electrónicos mejorando la seguridad. Todo esto con la colaboración del Banco Santander.

En la TUI, por un lado el certificado electrónico del usuario es almacenado y por otro, a través de la lectura del código QR, se puede determinar la caducidad de la tarjeta así como las credenciales y filiación del usuario sin necesidad de tener impresa información variable en el tiempo en la tarjeta.

El código QR indica una URL, donde se incluye la universidad, así como un dato identificativo único de la tarjeta o del usuario. En la TUI de la UPCT es https://autentica.upct.es/apps/identidad?qr=xxxxxxxxxxxxxxxxxx, donde 'qr' es una cadena de caracteres único para cada usuario. El acceso a esta URL invoca a un servicio implementado por la universidad que devuelve datos personales del usuario, la relación con la universidad e informa de si está activo ese usuario.

Figura 6.4 Parte trasera de una TUI con QR

Esto facilita su identificación, en todos los lugares y países, en los idiomas requeridos, y fundamentalmente, el acceso directo al ERP universitario garantiza la información actualizada.

6.3.2 Observatorio TUI App

Gracias al código QR implantado en la TUI, el Observatorio de la TUI del Banco Santander ha desarrollado una aplicación para smartphones la cual ofrece los servicios universitarios que a continuación se exponen:

- Consultar las reservas de las pistas deportivas.
- Consultar la disponibilidad de plazas en tiempo real en los aparcamientos de la Universidad.
- Consultar los descuentos en comercios por disponer de una TUI (Club TUI).
- Consultar el saldo y movimientos de la cuenta virtual EUNIS.
- Consultar toda la información de la TUI.
- Consultar los puntos de interés de la universidad.

Figura 6.5 Logotipo de Observatorio TUI App

Además irán incluyendo paulatinamente más servicios en la App como, por ejemplo, la posibilidad de reservar libros, consultar reservas y recibir avisos desde el móvil cuando un libro en el que estamos interesados está disponible, o cuando va a llegar la fecha de devolución de una reserva.

Para usar todo esto, además de tener un smartphone, sólo es necesario asociar la TUI con la App y hay dos alternativas para ello: usar el QR de la TUI o si es una TUI antigua, usar el PAN (Personal Account Number).

Figura 6.6 Capturas de pantalla de Observatorio TUI App

6.3.3 Portal Académico UMA App

Desarrollado por la Universidad de Málaga. Es una aplicación para dispositivos móviles

que permite a los estudiantes acceder a su información académica generando avisos inmediatos al móvil cuando se publica una calificación en su tablón de notas.

Esta aplicación, cuando se utiliza por primera vez, se debe autenticar en el portal web de la Universidad de Málaga como miembro de la misma y leer un código QR que le aparecerá en pantalla. Es imprescindible ser alumno de la UMA para poder hacer uso de esta aplicación.

A partir de ese momento la aplicación ya está activa y se pueden recibir avisos de notas. La aplicación dispone de un pin opcional por si se desea añadir un control de acceso adicional.

Actualmente, con el proceso de autenticación mediante código QR, sólo está disponible para Android. Puedes encontrarla en Google Play:

Figura 6.7 Portal Académico UMA

6.3.4 CIGES. Sistema de citas previas y gestión de colas

CIGES es un sistema que permite a los usuarios la posibilidad de concertar con las Secretarías de los Centros de la Universidad de Granada una cita previa a través de Internet y gestionar de manera eficiente el tiempo de espera usando los códigos QR, mejorando la calidad del servicio prestado.

Puede usarlo cualquier persona que necesite acceder a un servicio de un Centro Académico.

Al pedir una cita previa por vía web o presencialmente, se genera un código QR de confirmación que enviarán al correo electrónico o se imprime en un tótem instalado para pedir las citas presenciales. Después, para ser atendido, se debe validar el código QR en el tótem (impreso o desde la pantalla de un Smartphone) el mismo día de la cita y antes de la hora asignada para que el sistema sepa que estás esperando a ser atendido.

Este sistema gestiona de manera eficiente el tiempo de espera mejorando la calidad del servicio. Crea una espera correctamente implementada para proporcionar un trato equitativo a los usuarios. Los usuarios estarán más tranquilos al saber cuándo serán atendidos y sin necesidad de traslados innecesarios.

6.3.5 Uso del código QR en el campus

Un ejemplo de cómo usar los códigos QR en el campus se encuentra en la Universidad de Jaén. Ellos han lanzado un nuevo servicio destinado a ofrecer la información más relevante a través del teléfono móvil, utilizando códigos QR. Así, a través del teléfono móvil se dispone de información en tiempo real sobre la Universidad como eventos que se están desarrollando, actualidad universitaria, servicios o ubicación de dependencias.

Los códigos QR están ubicados en distintas zonas de la universidad y pueden ser capturados con un teléfono móvil para que la aplicación lo interprete y se traslade a la página web donde se encuentre la información.

Figura 6.8 Uso del QR en el campus de la Universidad de Jaén. Fuente: (Universidad de Jaén, 2010)

6.3.6 Uso del código QR en las bibliotecas

El primer uso que se le dio a los códigos QR en el ámbito universitario fue el uso en las bibliotecas, servicios de documentación y archivos.

Algunas de las ventajas de su uso en bibliotecas son:

- Facilita el acceso rápido a las versiones adaptadas para móvil de sitios webs o perfiles sociales de la biblioteca (blogs, Twitter, Facebook, etc.),
- Complementa la información de carteles, paneles explicativos, directorios (con enlace a un mapa de situación), derivando a la dirección web sobre el evento o servicio en cuestión.
- Acceso a sistemas de ayuda o referencia bibliográfica mediante SMS.

- Se puede localizar físicamente en las estanterías, permite acceder a los libros electrónicos que se encuentran disponibles sobre una materia concreta en esa estantería.
- Insertados en los registros bibliográficos del catálogo en línea, puede facilitar el manejo de esta información. Y al contrario, pegándolos al libro físico, a través del móvil se puede tener acceso a la información en el catálogo sobre esa obra,
- Descargar archivos de sonido con audio-guías de cada planta de la biblioteca y diseñar nuevos estilos de visitas guiadas.

Un ejemplo sería este código QR en una estantería de la Biblioteca de la Universidad de Sevilla:

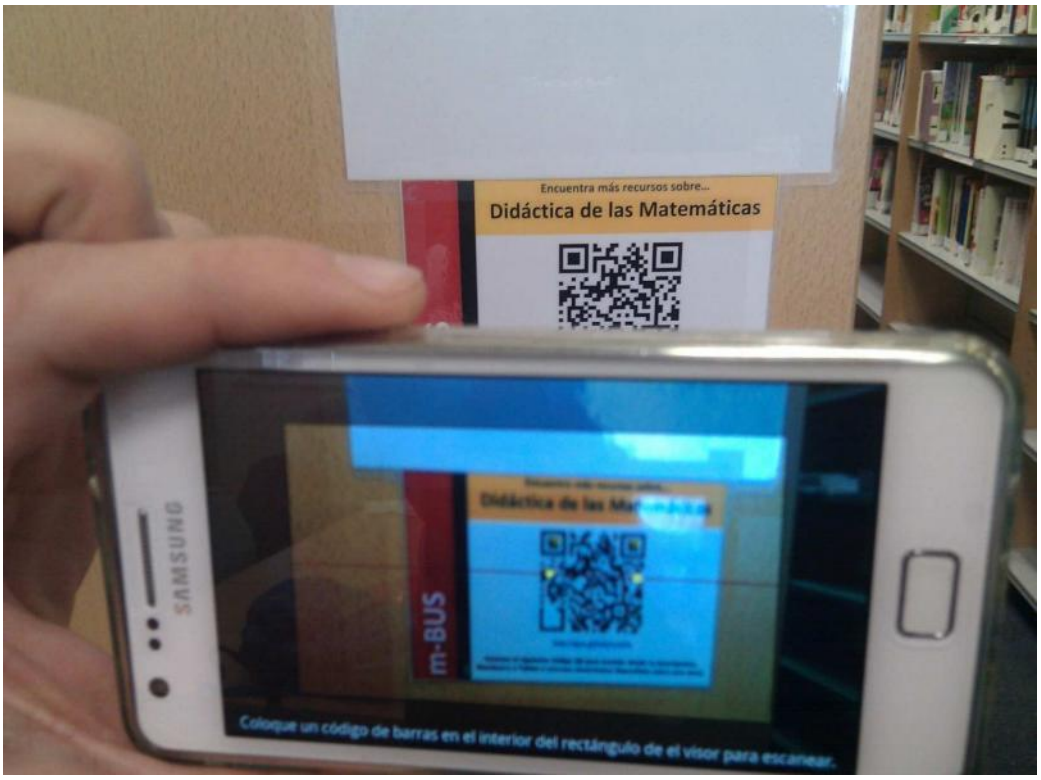

Figura 6.9 Ejemplo de QR en una biblioteca. Fuente: (Universidad de Sevilla, 2014)

Cada vez hay más bibliotecas que añaden los códigos QR en sus estanterías, libros y tablones y no sólo como enlace directo para páginas webs. Algunos ejemplos de universidades españolas son:

- Biblioteca y Archivo de la Universidad Autónoma de Madrid
- Biblioteca de la Universidad de Sevilla.
- Bibliotecas de la Universitat Politècnica de València.
- Biblioteca de Cappont de la Universitat de Lleida.

6.3.7 QR-Where App

Esta aplicación consiste en un sistema de mensajería sencillo en el cual cada usuario dispone de un código QR el cual al leerlo indica el estado y/o el lugar en el que está el

usuario. También indica el tiempo que durará el estado. El código QR del usuario es siempre el mismo, no se necesita cambiar el código QR cada vez que cambie el estado.

La utilidad de esta aplicación es que clientes, compañeros, amigos, alumnos, profesores o cualquier persona que intenten localizar al usuario, por ejemplo, en el despacho, puedan leer el QR que se encuentre en la puerta y que les muestre que se encuentra en una reunión o en clase o en casa o fuera de la empresa o de viaje, y también se puede indicar cuándo volverá al despacho. Es la alternativa perfecta a los post-it en las puertas.

La persona que lea el código QR del usuario no necesita tener instalada esta aplicación, simplemente necesitará un lector de códigos QR.

Figura 6.10 Pasos a seguir en QR-Where. Fuente: http://qr-where.com/#!/?page_id=2, visitada junio 2014

6.4 Casos prácticos

Vistas todas las características y propiedades de la TUI y los usos que pueden tener los códigos QR, se ha llegado a la conclusión que ellos pueden ser muy útiles y más aún si en cada TUI encontramos un código QR personal. Por ese motivo en este proyecto se han desarrollado tres casos prácticos para el uso del QR y de la TUI en la UPCT. En dos de ellos no se ha utilizado el propio QR de la TUI de la UPCT porque aún no estaba implantado en la TUI de la UPCT cuando se desarrollaron pero el cambio para poder usar ese QR sería muy fácil.

Los casos prácticos siguientes son tres aplicaciones móviles desarrolladas en Android y las tres aplicaciones tienen como objetivo ayudar a los usuarios, agilizar ciertos procesos en determinados momentos y dar comodidad tanto a usuarios como para el personal de la UPCT. Y todo eso usando la TUI y un smartphone.

6.4.1 Caso 1) Control de asistencia a eventos basadas en QR

En este ejemplo práctico se ha creado un sistema para el control de asistencia a eventos basado en QR con la finalidad de obtener un registro de las personas que han asistido a un evento privado. La diferencia de este caso práctico y los siguientes es que este caso se llevó a cabo el 20 de marzo del 2014 en el evento Charla Magistral de Ferran Adrià que organizó conjuntamente la Universidad Politécnica de Cartagena y Telefónica con exitoso resultado.

El caso práctico consta de una aplicación móvil y de dos servlet en el servidor de la UPCT.

El funcionamiento básico consiste en que los invitados deben asistir con una invitación generada por el servidor con anterioridad, la cual lleva un código QR y en la entrada del evento, los encargados llevarán en un dispositivo móvil la aplicación móvil la cual registrará y validará las invitaciones. De este modo se sabrá quienes asistieron al evento de todos los invitados.

En un principio, el servidor generará las invitaciones a los usuarios que lo soliciten, enviándoles un archivo PDF en el cual hay un código QR y más datos como el nombre, el NIF, el lugar y la hora del evento. Todo esto personalizable. El invitado puede llevar la invitación en papel o incluso en su dispositivo móvil al evento.

En el momento del evento, en la entrada habrá una o más personas que se encarguen del control de asistencia. Ellas llevarán unos dispositivos móviles los cuales tienen esta aplicación móvil que leerá los códigos QR de las invitaciones y comprobarán, conectándose con el servidor, si esa invitación es válida o no.

En el caso de que el dispositivo no tuviera conexión a Internet o el servidor no funcionara correctamente se ha desarrollado un modo offline con el que la app puede seguir funcionando.

El esquema de funcionamiento se muestra a continuación:

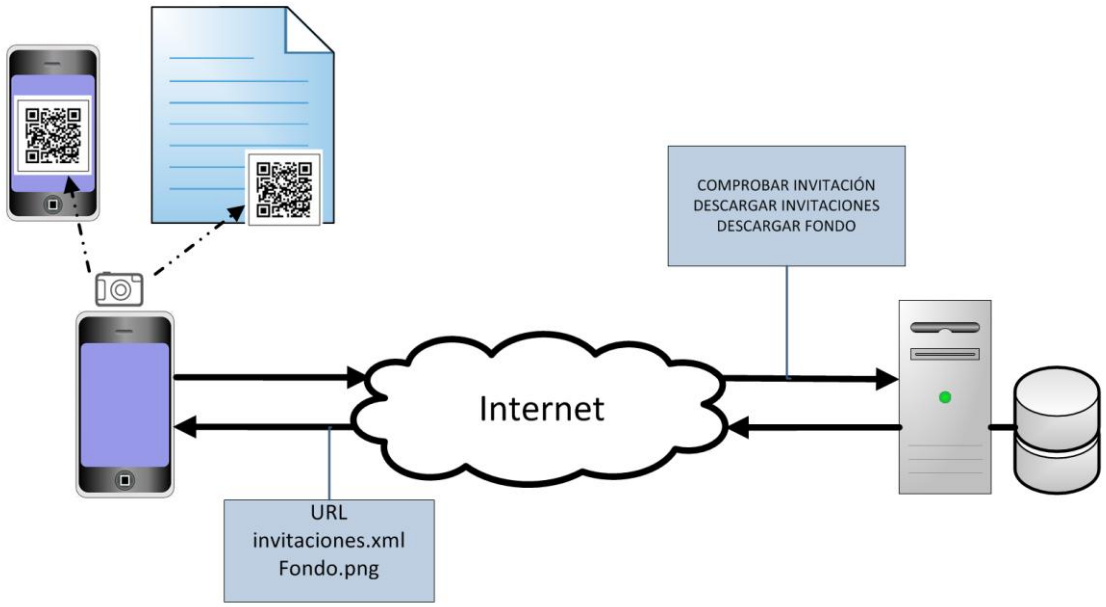

Figura 6.11 Esquema de funcionamiento del caso 1.

Comenzaremos explicando la aplicación móvil y continuaremos con la parte del servidor.

El objetivo de la app móvil es tener una aplicación capaz de leer el código de las invitaciones y verificar si es una invitación válida o no en la entrada del evento. También que sea capaz de funcionar en modo offline por si hubiera problemas de conexión. A continuación veremos cómo funciona la aplicación móvil. La explicación del código y el código completo se puede encontrar en los anexos.

Al ejecutar la aplicación lo primero que se encuentra es la pantalla que se encuentra en la figura 6.12 la cual tiene un botón para escanear los códigos QR de las invitaciones. También como se puede observar hay un texto que muestra si la app se encuentra en modo online o en modo offline.

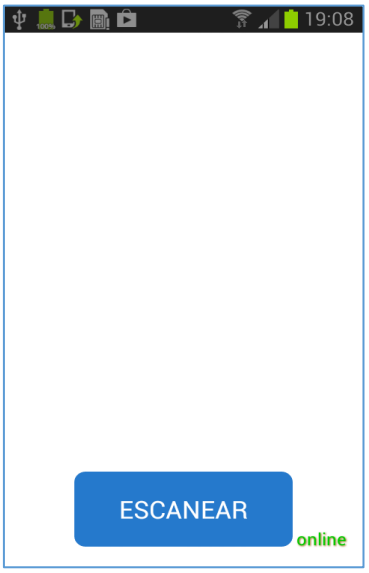

Figura 6.12 Caso 1 – Pantalla principal

Por defecto la app comenzará en modo online. Si se produjera algún fallo en la conexión habría que cambiar a modo offline la aplicación para poder trabajar con la aplicación. Este modo puede cambiarse en el menú. Para usar el modo offline, antes del evento, se debe descargar desde el servidor un archivo XML con la lista de invitados y la app podrá comprobar si los datos que tiene en dicha lista son los mismos que en el código QR de la invitación. Para poder descargar la lista de invitados se debe tener autorización y los encargados de preparar la aplicación para el evento son los únicos que deben conocer la contraseña para realizar tal descarga.

En la figura 6.12 también podemos observar que el fondo es blanco, pero eso es temporal. En el menú de la aplicación podemos descargar una imagen del evento para usarla como fondo.

Al pulsar el botón la aplicación entra en un lector de QR el cual nos extrae los campos que necesitamos para posteriormente conectarnos al servidor y poder verificar la validez de la invitación.

Si todo es correcto, aparecerá una pantalla con los datos del invitado y el evento al que entra como se puede observar en la figura 6.13. También aparecerán dos botones los cuales uno es para escanear otra invitación para que el control de acceso al evento sea más rápido y el otro es para volver a la pantalla inicial anteriormente mencionada.

En el caso de que esté funcionando en modo offline, la aplicación comparará los datos del código QR con la lista de invitados descargada con anterioridad al evento y la pantalla que mostrará se puede ver en la figura 6.14 para el caso de una invitación válida.

Este caso práctico está preparado para que pueda usarse en más de un evento a la vez pero desde distintos dispositivos. De los dos servlet que se han desarrollado, uno es empleado para descargar la lista de invitados de cada evento en el dispositivo móvil con seguridad, y el otro (bastante más complejo) es el encargado de verificar cada invitación. Todo lo referente a los servlets se explica con mayor profundidad en los códigos adjuntados en los anexos.

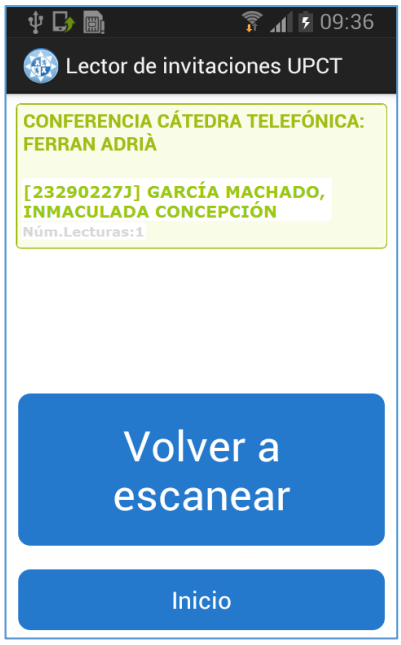

Figura 6.13. Caso 1 - Validación modo online

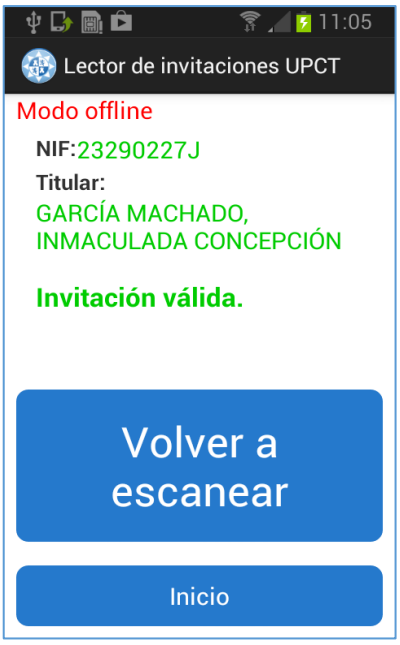

Figura 6.14. Caso 1 – Validación modo offline

6.4.1.1 Cambios si se usa el QR de la TUI

En el caso de que se usara también el QR de la TUI en los próximos eventos habría que hacer una serie de cambios.

La información del QR sería algo así:

https://autentica.upct.es/apps/identidad?qr=xxxxxxxxxxxxxxxxxxxx

, donde el 'qr' es una cadena de caracteres única para cada usuario de la UPCT.

Para poder verificar la validez tendríamos que tener una lista de las cadenas de las TUIs de la UPCT y así se verificaría sin importar si la aplicación está en modo online u offline.

En el caso de poder usarse el QR de la TUI para entrar en eventos, también se tendría que tener el acceso con invitación, dado que no todos los invitados tienen que tener relación con la UPCT y disponer de la TUI.

6.4.2 Caso 2) Registro de dispositivos móviles de un usuario basado en QR

En este caso se ha desarrollado una aplicación universitaria en la cual el usuario sólo necesita autenticarse una vez para poder usarla. Para ello, en esa primera autenticación el dispositivo queda registrado en la base de datos vinculado a ese usuario. Todo ello, para tener más seguridad, se realiza leyendo un código QR creado por el Portal de Servicios del usuario desde un ordenador seguro desde el que sí se ha autenticado el usuario. De este modo, en la app no ha sido necesario introducir la contraseña evitando ser interceptada.

Al igual que el dispositivo se puede vincular a ese dispositivo también se puede desvincular desde la aplicación.

El funcionamiento consiste en:

- Un usuario se descarga la app y la ejecuta. Inmediatamente la aplicación le pregunta al servidor si ese dispositivo está registrado.
- En el caso de que sí esté registrado, la aplicación mostrará el menú principal de ese usuario asociado a ese dispositivo.
- En el caso de que no esté registrado ese dispositivo, se procederá a autenticar al usuario y para ello necesitará tener un ordenador y acceder al Portal de Servicios. En el Portal, una vez autenticado el usuario, se accederá a la opción de 'Otros Servicios' – 'Acceso QR' y ahí se mostrará un código QR personal con una validez de pocos minutos para poder leerlo con la aplicación y de este modo ese dispositivo móvil quede vinculado a ese usuario.

En la figura 6.15 podemos ver un esquema reducido del funcionamiento.

Este caso práctico se ha hecho en un servidor local, pero está pensado para que los usuarios encuentren el QR personal en el Portal de Servicios como se hizo en el primer caso.

El dispositivo móvil debe llevar cámara para poder escanear el QR desde un ordenador una vez autenticado el usuario en el Portal de Servicios.

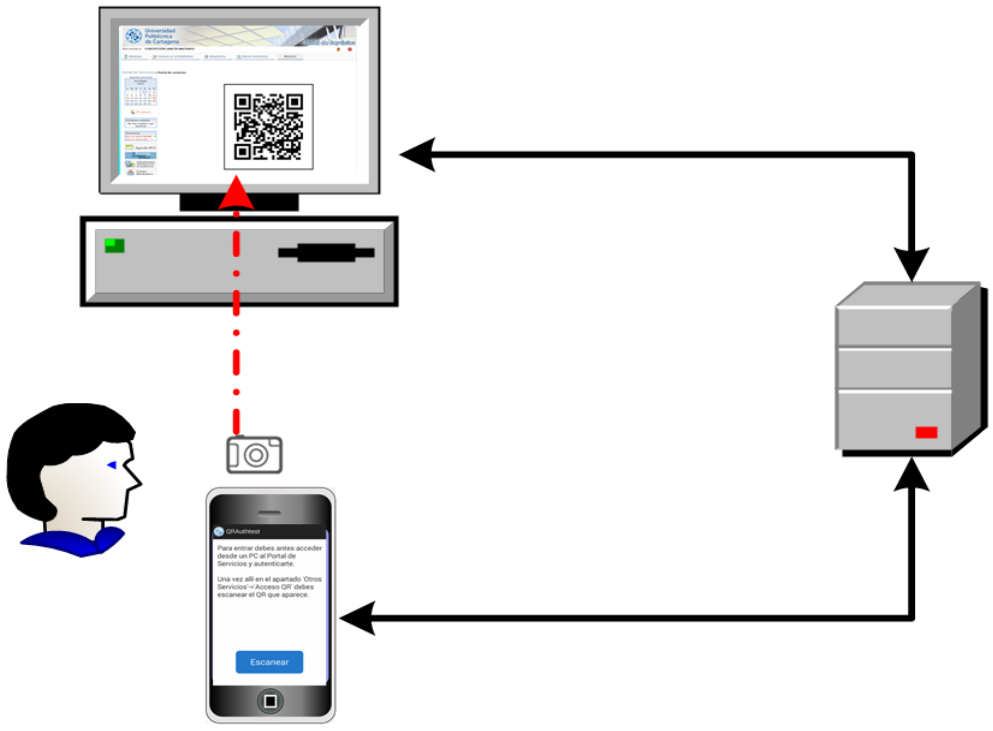

Figura 6.15 Caso 2 – Esquema del funcionamiento

Este modo de autenticación y registro podría usarse para varias aplicaciones que requieran un acceso continuo y en las que se quieran enviar notificaciones al usuario, como por ejemplo, notificar a un alumno la nota de una asignatura. En este caso, se ha hecho un

pequeño ejemplo de aplicación para ver las asignaturas de cada año académico indicando también las calificaciones si las tiene.

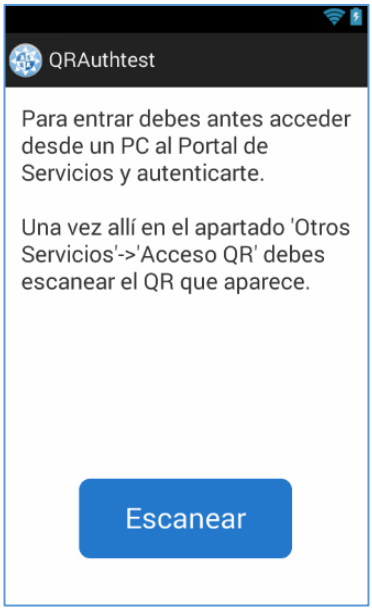

Figura 6.17. Caso 2 – Instrucciones para el registro

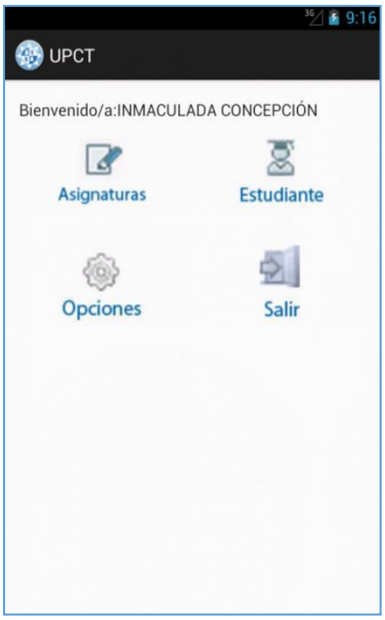

Figura 6.16. Caso 2 – Menú principal

Al igual que el caso práctico 1, este caso está explicado más profundamente y se pueden encontrar los códigos en los anexos.

6.4.3 Caso 3) Aplicación para pagar en los establecimientos universitarios a través de una cuenta virtual

Con esta aplicación la comunidad universitaria podrá pagar en los establecimientos universitarios como las cantinas o las copisterías sólo usando su TUI, pero no como tarjeta financiera sino como una tarjeta identificativa para la cuenta virtual de la UPCT. Se ha creado una aplicación móvil la cual se conectará con la base de datos de las cuentas virtuales de la UPCT haciendo las modificaciones necesarias en ella. Estas cuentas virtuales personales son prepago y son muy cómodas y fáciles de usar y en este caso sólo se necesitará disponer de la TUI.

Es una aplicación que permite cobrar, devolver o mirar el historial de transacciones de los usuarios a través del código QR de cada TUI. La figura 6.18 muestra el menú principal de la aplicación.

Deberá estar instalada en un dispositivo autorizado y registrado por la UPCT y, además, el personal que la utilice deberá también estar registrado y en cada

Figura 6.18. Caso 3 – Menú principal

sesión autenticarse.

Cuando se vaya a realizar una transacción, el empleado autorizado debe introducir la cantidad a cobrar o devolver (Fig. 6.20). Después la aplicación pedirá realizar la lectura del código QR de la TUI del cliente (Fig. 6.21). Si es correcta, pedirá la contraseña de la cuenta virtual la cual el cliente debe introducir y se enviarán los datos al servidor. El servidor comprobará si es posible la transacción y, si todo es correcto, se realizará la transacción y se enviará un mensaje de confirmación. En el caso de que no hubiera saldo suficiente o la contraseña introducida fuera incorrecta se mostraría un error. El esquema de funcionamiento se muestra en la figura 6.19.

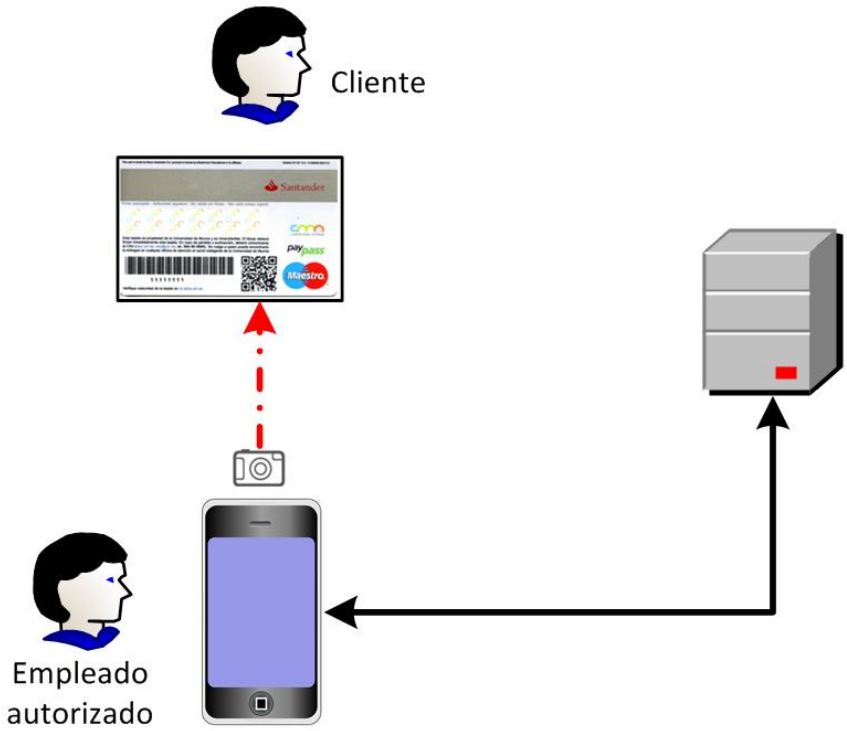

Figura 6.19 Caso3 – Esquema del funcionamiento

En el servidor se registrará cada movimiento y el saldo de los usuarios será automáticamente actualizado. En la base de datos se guardarán para cada movimiento todos los datos como la fecha, la hora, el lugar, el dispositivo móvil utilizado, el empleado, el cliente, el importe y el tipo de transacción para que no se produzcan incidencias ni errores.

En el caso de que un cliente quiera mirar su historial de transacciones, el sistema funciona de modo similar. La aplicación hace la lectura del código QR del cliente y el cliente debe introducir su contraseña (Fig.6.22). Si todo es correcto se mostrará una lista con los movimientos realizados en esa cuenta virtual indicando la fecha, hora, lugar, importe y tipo de transacción (compra o devolución) de cada movimiento (Fig. 6.23).

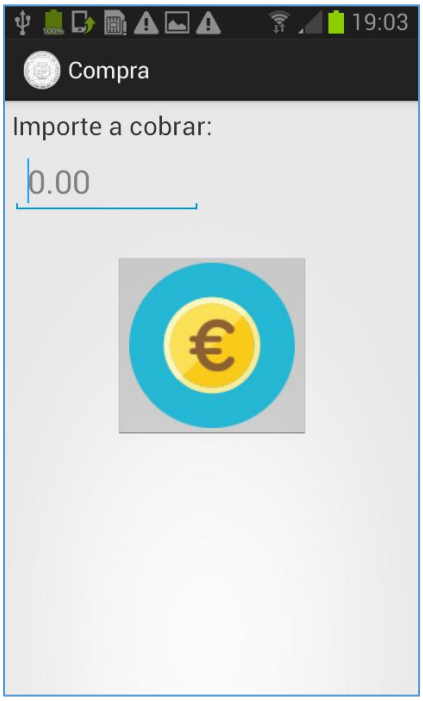

Figura 6.20. Caso 3 - Compra

Figura 6.21 – Lectura del QR

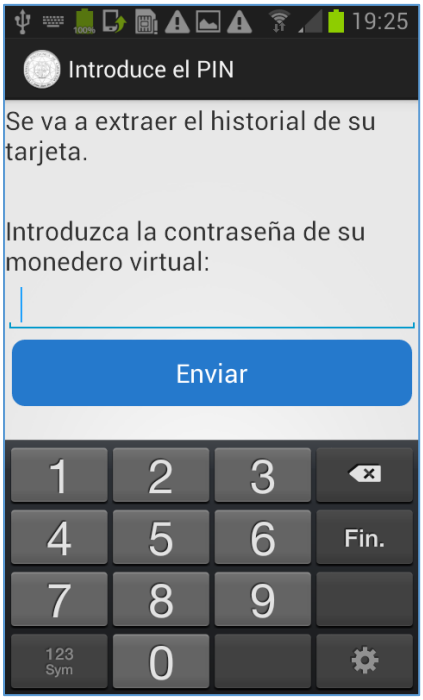

Figura 6.22. Caso 3 – Introducción del PIN

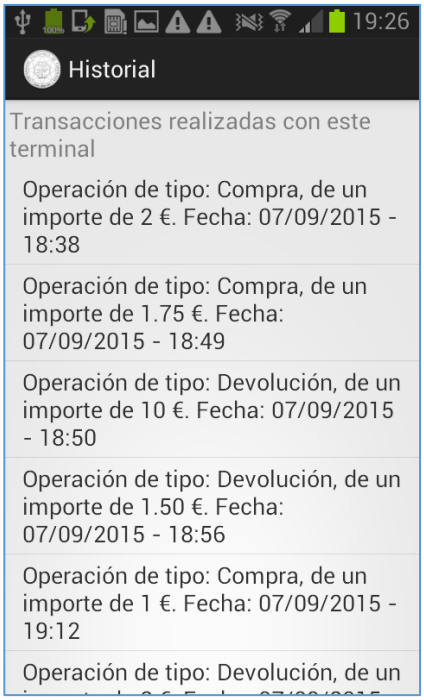

Figura 6.23. Caso 3 - Historial

6.5 Fortalezas, carencias y mejoras de los casos prácticos

En el caso 1, respecto a la seguridad de la entrada del evento tenemos un código QR generado con unos datos personales del invitado pero también una clave (hash) que verifica la invitación del invitado, esto hace que es invitación sea imposible de falsear. Una persona podría generar un código QR con sus datos pero no sería válida la invitación porque no ha sido generada por el servidor. Por lo tanto, la seguridad es alta.

Además, en la entrada del evento es aconsejable que se pida un carnet de identificación para que nadie pueda suplantar a otra persona, por ejemplo, si un alumno tiene que asistir obligatoriamente a una conferencia y lo suplanta un amigo. Debido a esta debilidad como es la suplantación física, sería aconsejable usar el código QR de la TUI. La tarjeta sería la entrada y la identificación a la vez, siendo más rápido que si se tiene que mostrar una invitación y un carnet.

Respecto al caso 2, hay un montón de ventajas respecto a la autenticación mediante códigos QR, y es por eso que ha comenzado a ganar fuerza.

Es un sistema simple, lo que significa que es más fácil para el usuario que no necesita recordar tantas contraseñas, y hay menos posibilidades de intrusismo. Los usuarios sólo deben recordar la contraseña de la UPCT.

Además en el Portal de Servicios se podría implementar una lista con los dispositivos activos que tiene el usuario, así podría ver si hay algún dispositivo ajeno y dar de baja el que desee.

El código QR en estos momentos para cada usuario es el mismo, no cambia. Esto es una debilidad respecto a seguridad se refiere. Sin embargo podría modificarse y así proporcionaría mayor protección ya que el código QR cambiaría para cada nuevo dispositivo, y no sería ese QR nunca más reutilizado, parecido al sistema SQRL del que se habla en el apartado 6.2.1 Autenticación con código QR.

Respecto a la aplicación en sí, pueden añadirse multitud de mejoras: poder ver más información, añadir la reserva de instalaciones deportivas, añadir notificaciones PUSH para notificar la subida de notas, un examen, o cambios realizados por un profesor.

En el caso 3 se debe indicar que al usar la TUI como medio de pago las transacciones son bastante más rápidas y más cómodas. Tampoco cabe la posibilidad de suplantación o de que otra persona pueda usar ese dinero de la cuenta virtual porque al usar la TUI como medio de pago se está usando ‘**algo que posee’** el usuario y que además lleva su fotografía, y al tener que introducir el PIN de la cuenta virtual estamos usando **‘algo que sabe’**. De este modo se está usando una identificación multifactor muy segura y fiable.

Otra ventaja a tener en cuenta es la de que todo movimiento queda reflejado. El lugar desde donde se hace, con qué dispositivo, quién del personal autorizado es el que ha hecho ese movimiento, la hora y el día, todos los datos del movimiento queda reflejado para posibles incidencias o errores.

Capítulo 7. Conclusiones

La TUI se puede convertir en un recurso muy valioso siempre y cuando se aprovechen sus propiedades. Además de la identificación física del usuario y de poder hacer pagos con ella, la TUI también sirve para autenticarse y firmar electrónicamente documentos y estas dos funciones la convierten principalmente en un objeto muy útil para su usuario.

Las Administraciones Públicas están evolucionando y en este desarrollo es muy importante disponer de las herramientas básicas para poder utilizar sus nuevos recursos. La TUI permite a su usuario la posibilidad de alojar sus certificados electrónicos dándole la oportunidad de firmar con ella con mayor seguridad utilizando la firma electrónica reconocida. Esta firma se considera idéntica a la manuscrita debido a que ha sido creada en la TUI que se considera un dispositivo seguro de creación de firma (SSCD). Al contrario, cuando usas un ordenador para generar una firma, no se trataría de una firma electrónica reconocida porque un ordenador no es considerado un dispositivo seguro de creación de firma y en este caso se realizaría una firma electrónica avanzada.

Actualmente, el código QR en la TUI permite obtener información del usuario de la TUI que no se encuentra plasmada en ella físicamente, pero en este proyecto se ha mostrado que otras utilidades se le pueden dar a este recurso. Gracias a la evolución de las tecnologías y al gran uso de los smartphones se han desarrollado tres aplicaciones que muestran como usando el código QR, por ejemplo, hacen una identificación electrónica rápida, registra un dispositivo móvil para poder obtener rápidamente información relevante y también se puede pagar en una cuenta virtual vinculada a ese código QR. De este modo, la TUI se convierte en una herramienta muy útil e indispensable para el día a día de la comunidad universitaria.

Todas estas características, funciones de la tarjeta, los servicios que ofrece y que pueden ofrecer, muestran como la Tarjeta Inteligente Universitaria puede ser no sólo una credencial identificativa, sino un recurso con un gran valor para sus usuarios.

7.1 Planes futuros

El futuro de la TUI está más cerca de lo que se cree. Gracias al gran equipo que se encuentra detrás que siempre está trabajando y adelantándose a los tiempos, se están llevando a cabo grandes proyectos los cuales van a hacer que la TUI se convierta en una herramienta muy útil y necesaria en el futuro.

En el IV Congreso Internacional de la TUI el cual se ha llevado a cabo los días 7 y 8 de abril de 2016 en Lisboa, han salido a la luz grandes proyectos de los cuales cabe destacar la próxima TUI la cual no dispondrá de chip de contacto porque la TUI ya es totalmente accesible en su modo sin contacto, cada vez se usa el modo con contacto y por los inconvenientes como el desgaste y la necesidad de lectores específicos. Además, el ahorro que supone este cambio.

Otro de los proyectos de los que se hablaron es la posibilidad de poder firma electrónicamente con la TUI a través del móvil. Cada vez los smartphone son más seguros y útilies y la TUI no se quedará atrás adaptándose a estos cambios. Por eso pronto será

posible poder realizar la firma eléctronica reconocida de la TUI a través de nuestro dispositivo móvil.

Y por último, destacar la posibilidad de llevar la TUI en un wearable, por ejemplo en una pulsera, capaz de comunicarse a poca distancia y poder pagar con ella, acceder a determinadas dependencias e incluso usar el transporte urbano. De este proyecto ya se está llevando a cabo un piloto con usuarios reales y pronto estará a la disposición de toda la comunidad. Además en un futuro no muy lejano, además de pagar en máquinas expendedoras, usar el transporte urbano o para el control de acceso, se conseguirá que estos wearables tengan las mismas funciones que la TUI ofrece.

Capítulo 8. Bibliografía

Tarjeta Universitaria Inteligente y su tecnología:

ABC.es, (2013). El monedero virtual Google Wallet ya está disponible para iOS. [online] Disponible en: http://www.abc.es/tecnologia/moviles-aplicaciones/20130919/abci-google-wallet-201309191816.html

Acosta, D. (2014). *¿Cómo funcionan las tarjetas de pago?* Recuperado en julio de 2015, en http://www.pcihispano.com/como-funcionan-las-tarjetas-de-pago-parte-vi-tarjetas-contactless-rfid-nfc/

Agencia de Tecnología y Certificación Electrónica. *Certificados reconocidos en dispositivo seguro para ciudadanos*. Recuperado en julio de 2015, en http://www.accv.es/ciudadanos/certificados/certificados-en-tarjeta-criptografica/

Amado, R. (2010). *Hacking RFID, rompiendo la seguridad de Mifare*. Recuperado el 21 de septiembre de 2015, de http://www.securityartwork.es/2010/01/29/hacking-rfid-rompiendo-la-seguridad-de-mifare-i/

A3M, (2016). *Tarjeta chip WG10*. [online] Disponible en: http://www.a3m.eu/es/tarjetas-plasticas/tarjetas-chip-de-contacto/tarjeta-chip-wg10.html

Baixauli Soler, J.S. y otros. (2014). *Implantación de la TUI en el Sistema Universitario Español*. Murcia: Observatorio Internacional de la TUI de la Universidad de Murcia.

Benita Crespo, J. (2007). *Cifrado de la Información con Algoritmos Simétricos Usando Claves de Identificación Única de la Java Card para la Java ME*. Sevilla: Universidad de Sevilla.

España. Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Común de las Administraciones Públicas. Boletín Oficial del Estado, 2 de octubre de 2015, nº236, pp. 89343-89410

España. Ley 21/2011, de 26 de julio, de Dinero Electrónico. Boletín Oficial del Estado, 27 de julio de 2011, nº 179, pp. 84235-84254.

España. Ley 16/2009, de 13 de noviembre, de Servicios de Pago. Boletín Oficial del Estado, 14 de noviembre de 2009, nº 275, pp. 96887-96918.

España. Ley 11/2007, de 22 de junio, de Acceso Electrónico a los Ciudadanos a los Servicios Públicos. Boletín Oficial del Estado, 23 de junio de 2007, nº 150, pp. 27150-27166.

España. Ley 59/2003, de 19 de diciembre, de Firma Electrónica. Boletín Oficial del Estado, de 20 de diciembre de 2003, nº 304, pp. 45329-45343.

Eumed.net, (2011). *Monederos virtuales | Uso de tarjetas en Internet*. [online] Disponible en: http://www.eumed.net/cursecon/ecoinet/seguridad/mon-vir.htm

Gemalto, (2012). *IAS Classic V3 on MultiApp ID V2.1*. Recuperado en noviembre de 2015, en http://www.commoncriteriaportal.org/files/epfiles/anssi-cc-cible_2013-30en.pdf

Google, (2013). *Google Wallet*. [online] Google.com. Disponible en: https://www.google.com/wallet/

Henríquez, C. (2010). Sistema de control de acceso basado en Java Cards y Hardware libre. *Prospectiva*, [online] 8(2), pp.63-68. Disponible en: http://dialnet.unirioja.es/servlet/articulo?codigo=3634650.

Medina Vargas, Y. (2015). Comparación de Algoritmos Basados en la Criptografía Simétrica DES, AES y 3DES. *Revista Mundo FESC*, (9), pp.14-21.

MIFARE. Visitado en febrero de 2014, en https://www.mifare.net/en/

Observatorio Internacional de la TUI. (2015). *Aplicación Móvil*. Recuperado el 12 de diciembre de 2015, de http://www.observatoriotui.com/apptui

Observatorio Internacional de la TUI, (2015). *Evolución de la TUI R4 a TUI R5/R6 y TUI R7*. Recuperado el 15 de septiembre de 2015, de http://www.observatoriotui.com/evolucin-a-tui-r7/tui-r7

Observatorio Internacional de la TUI, (2015). *NexUSC*. Disponible en: http://www.observatoriotui.com/nexusc

Portal de Administración Electrónica, Gobierno de España. *Base legal de la Firma Electrónica*. Recuperado el 5 de octubre de 2015, de http://firmaelectronica.gob.es/Home/Ciudadanos/Base-Legal.html

Redondo, M. (2013). El *monedero virtual entra en nuestras vidas*. [online] Cinco Días. Disponible en: http://cincodias.com/cincodias/2013/11/14/empresas/1384445419_657778.html [Último acceso en febrero de 2014]

Rouse, M. (2014). *Autenticación multifactor (MFA): Definición*. Recuperado en diciembre del 2015, de http://searchdatacenter.techtarget.com/es/definicion/Autenticacion-multifactor-MFA

Códigos QR:

Alcalde, A. (2013). *SQRL y la idea de eliminar el uso de usuario y contraseña en internet*. Visitado en marzo de 2015, en http://elbauldelprogramador.com/sqrl-y-la-idea-de-eliminar-el-uso-de-usuario-y-contrasena-en-internet/#more-1929

Barrera, J.A. y Moya, V. (2011). *Tendiendo puentes entre el mundo real y el virtual: Códigos QR en la Biblioteca de Ciencias de la Educación*. Recuperado de http://www.slideshare.net/jabarrera/cdigos-qr-en-la-biblioteca-de-ciencias-de-la-educacin-de-la-universidad-de-sevilla

Biblioteca ETSID UPV, (2011). *Códigos QR en la Biblioteca*. [imagen] Disponible en: https://www.flickr.com/photos/62718242@N07/sets/72157626751904685/

Denso Wave. *QR Code*. Visitado en enero de 2015, en http://www.qrcode.com/en/ y http://www.codigos-qr.com/

Gibson Research Corporation, (2013). *Secure Quick Reliable Login*. Recuperado en marzo de 2015, en https://www.grc.com/sqrl/sqrl.htm

Jiménez García, T. (2013). *Video: Implantación de la Tarjeta Universitaria Inteligente (TUI) en la Universidad de Murcia*. Visitado en abril de 2015, en http://www.observatoriotui.com/ier-congreso-internacional-tui

Lambert, P. (2013). SQRL: A new method of authentication with QR codes. [Blog] TechRepublic. Disponible en: http://www.techrepublic.com/blog/it-security/sqrl-a-new-method-of-authentication-with-qr-codes/ .

QR-Where. Visitado en 2013 en http://qr-where.com

Renfe. *Billete en formato PassBook / Pass Wallet*. Visitado en noviembre de 2015, en http://www.renfe.com/viajeros/movilidad/billete_passbook.html

Sánchez, M.A. (2012). *¿Qué son los códigos QR?* Recuperado en febrero 2015, en http://realidadaumentadadesdecero.blogspot.com.es/2012/07/que-son-los-codigos-qr.html

Ube González, J.M. (2014). *Visita la Biblioteca de Cappont (Universidad de Lleida)*. Recuperado de http://blog.biblioteca.unizar.es/general/visita-a-la-biblioteca-de-cappont-universitat-de-lleida/

Universidad Autónoma de Madrid, (2011). *Código QR en la biblioteca*. Recuperado de http://biblioteca.uam.es/sc/codigoqr_en_biblioteca.html

Universidad de Granada. *CIGES Sistema de Cita previa y Gestión de colas*. Recuperado en septiembre de 2014, en https://ciges.ugr.es/

Universidad de Jaén, (2010). *La Universidad de Jaén lanza un nuevo servicios de información a través de teléfono móvil basado en códigos QR*. Recuperado en septiembre de 2014, en http://diariodigital.ujaen.es/node/16446

Universidad de Málaga, (2013). *Aplicación móvil: Portal Académico UMA*. Recuperado en septiembre de 2014, en https://play.google.com/store/apps/details?id=es.uma.sci.android.academico&hl=es

Universidad de Sevilla, (2014). *Códigos QR en la biblioteca*. Recuperado en junio de 2014,http://bib.us.es/sobre_la_biblioteca/gestion_y_organizacion/common/jornadas4/codigos_qr.pdf

Programación Android:

Arroyo, F.J. (2012). Integrar *Barcode Scanner en nuestra aplicación Android | Adictos al Trabajo*. [online] Adictosaltrabajo.com. Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=qr_reader_android.

Cipolat, S. (2012). Login en Android usando PHP y MySQL. [Blog] *Androideity*. Disponible en: http://androideity.com/2012/07/05/login-en-android-usando-php-y-mysql/

Developer.android.com. *AsyncTask | Android Developers*. [online] Disponible en: http://developer.android.com/reference/android/os/AsyncTask.html

Gómez Oliver, S. (2011). Ficheros en Android (I): Memoria Interna. [Blog] *Sgoliver.Net*. Disponible en: http://www.sgoliver.net/blog/ficheros-en-android-i-memoria-interna/#.

Medina, D. (2013). Tip Android #20: el servicio Download Manager. [Blog] *Danielme.com*. Disponible en: http://danielme.com/tip-android-20-el-servicio-download-manager/

Rodríguez, A. (2011). Trabajar con códigos QR en tus aplicaciones Android. [Blog] *Androideity*. Disponible en: http://androideity.com/2011/11/23/trabajar-con-codigos-qr-en-tus-aplicaciones-android/

Sean, O. (2012). *Official ZXing ("Zebra Crossing") project home*. Software alojado en: https://github.com/zxing/zxing

Sony Mobile Communications Inc. (2012). Flash Tool for Sony Xperia E. Software disponible en: http://developer.sonymobile.com/services/flash-tool/how-to-download-and-install-the-flash-tool/

Tomás Gironés, J. (2014). *El gran libro de Android avanzado*. Barcelona: Marcombo.

Ghanem, H. (2011). *Send and Receive JSON between Android and PHP Web Service - CodeProject*. [online] Codeproject.com. Disponible en: http://www.codeproject.com/Articles/267023/Send-and-receive-json-between-android-and-php

Webalys, (2015). Iconset: Kameleon Icons. [Imágenes] Disponibles en: http://www.iconarchive.com/show/kameleon.pics-icons-by-webalys.html

Capítulo 9. Anexos

9.1 Servicios implantados en la TUI de la UPCT

Datos de impresión	
Tarjetas de diferentes proveedores	No
Nombre y apellidos	Sí
Identificador del usuario	Sí
Fotografía	Sí
Caducidad	No
Relación con la universidad	Sí
Varias tarjetas por varias relaciones u otros	No
Código de barras	Sí
Código QR	Sí
Código de seguridad	No
Emisión y gestión	
Aplicación de gestión de tarjetas	Sí
Circuito de emisión automatizado	Sí
Fotomatón para recogida de fotografía	No
Impresión inmediata de tarjetas	Sí
Infraestructuras	
Control de Acceso. Puertas edificios	No
Control de Acceso. Puertas despachos	No
Control de Acceso. Aparcamientos	No
Control de Acceso. Tornos	No
Control de Acceso. Con NFC	No
Préstamo/Regida de bicicletas	No
Taquillas de apertura con tarjeta	No
Sistemas de ahorro de energía	No
Pago	
Tarjeta financiera	Opcional
Monedero electrónico	Sí
Monedero virtual	Sí
Máquinas de carga contra efectivo	No
Pago en servicios universitarios	Sí
Pago en reprografía/fotocopiadoras/impresión	No
Pago en cantinas/ comedores	No
Pago en máquinas de vending	No
Bibliotecas	
Préstamo de libros en biblioteca	Sí
Máquinas automáticas para recogida/devolución de libros	No
TPS-PIU-Secretarías virtuales	No
Control horario PAS o PDI	Sí
Control de asistencia a clase	No
Gestión de impresión	No
Cita previa	No
Voto electrónico	No

Descuentos	Sí
Transporte en la ciudad	No
Inscripciones en actividades deportivas	Sí
Acceso a ordenadores	
Acceso a puestos de informática de libre acceso	Sí
Acceso a ordenadores para PAS y PDI	No
Acceso al Campus Virtual	No
Acceso al sistema de generación de recibos (certificados, compulsas,…)	No
Acceso al portal del empleado	No
Acceso al sistema de validación de facturas por sus responsables	No
Acceso al sistema de reservas de aulas/salas de reunión	No

9.2 Datasheet de la TUI R7

Estado del producto	
Nombre del producto	Optelio Contactless R7 for Santander
Estado de vida del producto	Para certificado – muestras de I+D disponibles (cantidad limitada)
Características principales	
JavaCard / Global Platform	JavaCard 2.2.2 / GP 2.1.1
EEPROM	Hasta 72 kB para aplicaciones estándar. Se limitará a 45 kB cuando se instala M/Chip Avance
Sistema de comunicación de contacto	
Cumplimiento de estándares	ISO7816
T=0 (protocolo de transmisión a nivel de carácter, definido en la norma ISO / IEC 7816-3)	Sí
Sistema de comunicación sin contacto	
Cumplimiento de estándares	ISO 14443 – 2, -3, -4
T=CL (protocolo de transmisión para contactless)	Tipo A & Mifare 4K emulation & Desfire EV 1 4K emulation
Velocidad de transmisión	106kb en ambos sentidos
Frecuencia de operación	13,56MHz
Antena	Antena FULL Size
Algoritmos criptográficos	
Mensajería segura	Card Manager configurado con SCP02 (i=55)
Algoritmos soportados	• DES / 3DES • RSA (la tarjeta soporta hasta 2048bits en CRT – la longitud máxima de la clave depende de cada aplicación EMV) • AES: 128, 192 y 256 bytes • SHA1 • SHA256
Aplicaciones	
Aplicaciones EMV (sólo una puede ser seleccionada)	• M/Chip Paypass v2E • VSDC281f1 • M/Chip Advance v1.1
Otras aplicaciones	MPCOS V4.1

	WG10 IAS Classic V4.0.2c – Autenticación y firma electrónica Dual PSE 1.2
Mifare	Mifare 4K disponible – Opción para soportar Defire EV1 4K simultáneamente – UID: 7 bytes o 4 bytes NUID (derivado de 7 bytes)
Multi-instancia de applets	Sí (Verificar con su consultor técnico de Gemalto por los límites de tamaño de la memoria RAM y EEPROM)
Certificados	
MASTERCARD	PayPass M/Chip4 M/Chip Advance v1.1
VISA	VSDC2.8.f1
Criterios comunes de la Firma Digital	IAS Classic v4.0.2c – PPSSCD KI&KG
ICP Brasil	IAS Classic v4.0.2c
Cuerpo de la tarjeta	
Tipo	Interfaz dual
Estándar	ISO 14443-1 compatibles
Material	PVC de tipo banca con exterior personalizable
Tecnología	Laminación caliente
Banda magnética	ISO 7816 -4, -5
Antena	Tamaño completo (permitiendo 5 líneas de gofrado)

Application supported
M/Chip Paypass v2E (Mastercard) • Applet basado en la última versión de las Especificaciones V1.3 Mastercard PayPass M/Chip. Es totalmente compatible con el anterior producto R5.
M/Chip Advance v1.1 (Mastercard) • Aplicación de interfaz dual incluyendo parámetros de gestión avanzados Card Risk, así como la opción de DataStorage (identificación leal, aplicación de tránsito específico basado en el pago Open Loop, etc.)
VSDC281f1 (Visa) • Applet desarrollado por Visa en la última versión lanzado en febrero de 2014. Cumple con VIS1.5.3 y VCPS2.1.1. Esto incluye VSDC (Visa Smart Debit Card), qVSDC (Quick VSDC) y aplicaciones MSD (Magnetic Stripe Data).
Classic IAS V4.2 • Este applet puede usarse para autenticación fuerte, firmas digitales y cifrado, y el almacenamiento de certificados digitales. El esquema de la autenticación y la firma utilizado por el applet se basa en el estándar E-Sign. El applet es compatible con PKCS#15 y todas las aplicaciones PKI posteriores a las APIs PKCS#11 y/o Microsoft CAPI puede usar el applet Classic IAS (Internet Authentication Service). • Esto significa que las universidades Santander pueden utilizar el software Classic IAS o el software desde un third-party. • Este applet funciona ahora tanto en el modo de contacto como en el sin contacto.
WG10 • Gemalto utilizará el applet WG10 estándar ya utilizado por el Banco Santander • WG10 es una tarjeta con microprocesador ideada para aplicaciones de identificación de alta seguridad. Homologada por la Fábrica Nacional de Moneda y Timbre, su potente

sistema de seguridad con criptografía simétrica (algoritmo DES) hace que se use en documentos de identidad y tarjetas de corporaciones.
Mifare Classic / Desfire EV1 • En esta nueva generación, Mifare 4K y/o Desfire EV1 4K pueden ser habilitados durante la pre-personalización. Debe señalarse que opciones deben ser activadas al mismo tiempo para asegurar la máxima compatibilidad con los sistemas de acceso y transporte.
Aplicación de acceso a Mifare (versión TBD) • Aún TBD

9.3 Artículo 1, de la Ley 21/2011, de 26 de julio, de dinero electrónico

Artículo 1. Objeto y ámbito de aplicación.

1. El objeto de esta Ley es la regulación de la emisión de dinero electrónico, incluyendo el régimen jurídico de las entidades de dinero electrónico y la supervisión prudencial de estas entidades.

2. Se entiende por dinero electrónico todo valor monetario almacenado por medios electrónicos o magnéticos que represente un crédito sobre el emisor, que se emita al recibo de fondos con el propósito de efectuar operaciones de pago según se definen en el artículo 2.5 de la Ley 16/2009, de 13 de noviembre, de servicios de pago, y que sea aceptado por una persona física o jurídica distinta del emisor de dinero electrónico.

3. Esta Ley no se aplicará a aquel valor monetario:

a) almacenado en instrumentos que puedan utilizarse para la adquisición de bienes o servicios únicamente en las instalaciones del emisor o, en virtud de un acuerdo comercial con el emisor, bien en una red limitada de proveedores de servicios o bien para un conjunto limitado de bienes o servicios, de acuerdo con las condiciones que se establezcan reglamentariamente;

b) utilizado para realizar operaciones de pago exentas en virtud del artículo 3.l) de la Ley 16/2009, de 13 de noviembre, de servicios de pago.

9.4 Artículo 2.5 de la Ley 16/2009, de 13 de noviembre, de servicios de pago

Artículo 2. Definiciones.

A efectos de esta Ley, se entenderá por:

5. «Operación de pago»: una acción, iniciada por el ordenante o por el beneficiario, consistente en situar, transferir o retirar fondos, con independencia de cualesquiera obligaciones subyacentes entre ambos;

9.5 Artículo 3.I de la Ley 16/2009, de 13 de noviembre, de servicios de pago

Artículo 3. Excepciones a la aplicación de la Ley.

Esta Ley no se aplicará a las siguientes actividades:

i) las operaciones de pago relacionadas con la gestión de carteras, con inclusión de dividendos, réditos u otras distribuciones, o con amortizaciones o ventas, realizadas por personas mencionadas en la letra h) del presente artículo o por empresas de servicios de inversión, entidades de crédito, instituciones de inversión colectiva y sus Gestoras, Planes y Fondos de Pensiones y sus Gestoras y cualquier otra entidad autorizada a custodiar instrumentos financieros;

9.6 Artículos 3.2, 3.3 y 3.4 de la Ley 59/2003, de 19 de diciembre, de Firma Electrónica

Artículo 3. Firma electrónica, y documentos firmados electrónicamente.

2. La firma electrónica avanzada es la firma electrónica que permite identificar al firmante y detectar cualquier cambio ulterior de los datos firmados, que está vinculada al firmante de manera única y a los datos a que se refiere y que ha sido creada por medios que el firmante puede mantener bajo su exclusivo control.

3. Se considera firma electrónica reconocida la firma electrónica avanzada basada en un certificado reconocido y generada mediante un dispositivo seguro de creación de firma.

4. La firma electrónica reconocida tendrá respecto de los datos consignados en forma electrónica el mismo valor que la firma manuscrita en relación con los consignados en papel.

9.7 Ciclo de vida de una actividad en Android

Para profundizar un poco más en Android, a continuación se explica cómo es el ciclo de vida de una actividad, sus estados y métodos.

Todas las clases activities deben llevar por lo menos el método onCreate. Una actividad en Android tiene un ciclo de vida, por lo tanto, se necesita comprender y manejar los eventos relacionados con el ciclo de vida.

Los estados de una actividad son:

- Activa (running)
- Visible (paused)
- Parada (stopped)
- Destruida (destroyed)

Cada vez que una actividad cambia de estado se generan eventos que podrán ser capturados por ciertos métodos de la actividad. Por ejemplo, el método onCreate se llama en la creación de la actividad. Los métodos son:

- onCreate (Bundle): Se utiliza para realizar todo tipo de inicializaciones, como la creación de la interfaz de usuario o la inicialización de estructuras de datos. Puede recibir información de estado de la actividad (en una instancia de la clase Bundle), por si se reanuda desde una actividad que ha sido destruida y vuelta a crear.
- onStart(): Nos indica que la actividad está a punto de ser mostrada al usuario.
- onResume(): Se llama cuando la actividad va a comenzar a interactuar con el usuario. Es un buen lugar para lanzar las animaciones y la música.
- onPause(): Indica que la actividad está a punto de ser lanzada a segundo plano, normalmente porque otra actividad es lanzada. Es el lugar adecuado para detener animaciones o almacenar los datos que se quieren mantener o los datos que estaban en edición.
- onStop(): La actividad ya no va a ser visible para el usuario. En el caso de tener muy poca memoria, es posible que la actividad se destruya sin llamar a este método.
- onRestart(): Indica que la actividad va a volver a ser representada después de haber pasado por onStop().
- onDestroy(): Se llama antes de que la actividad sea totalmente destruida. Por ejemplo, cuando el usuario pulsa el botón ‘volver’ o cuando se llama al método finish(). En el caso de tener muy poca memoria, es posible que la actividad se destruya sin llamar a este método.

A continuación se muestra un diagrama en el cual se ilustran los métodos que capturan estos eventos:

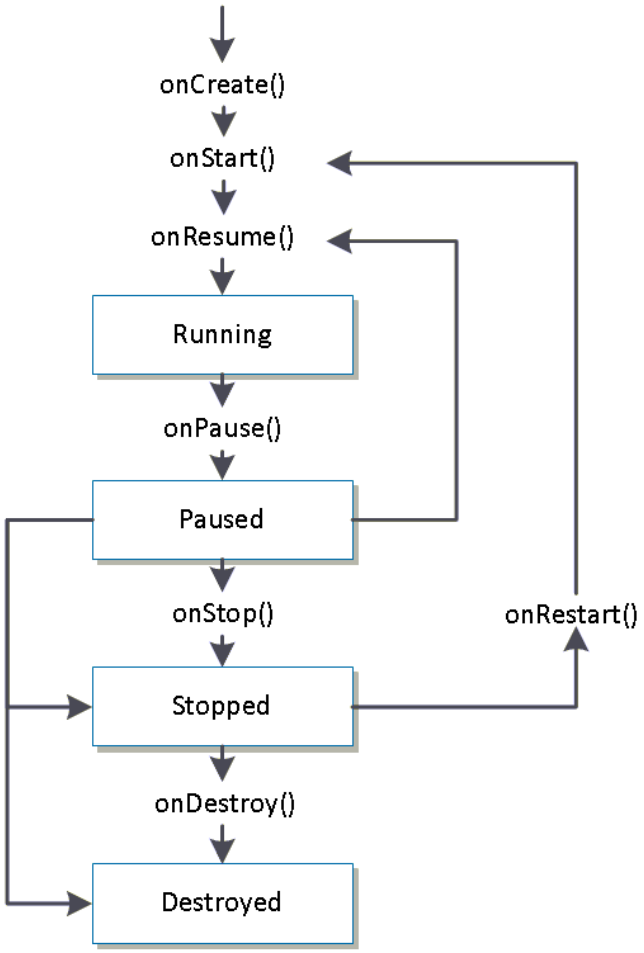

Figura 9.1. Esquema del ciclo de vida de una actividad en Android.

9.8 Explicación del código del caso 1

Como se ha explicado anteriormente, este caso consta de una aplicación móvil y dos servlet.

9.8.1 Aplicación móvil

Comenzaremos explicando el desarrollo de la aplicación móvil y continuaremos con la parte del servidor.

Con la app móvil el objetivo es tener una aplicación capaz de leer el código de las invitaciones y verificar si es una invitación válida o no en la entrada del evento. También que sea capaz de funcionar en modo offline. A continuación veremos cómo funciona la aplicación móvil explicando las partes importantes. El código completo se puede encontrar en los anexos.

MainActivity

Al ejecutar la app, lo primero que se ejecuta es la clase MainActivity.

Esta actividad muestra la pantalla principal de la aplicación. Mostrará una imagen de fondo según el evento. Esta imagen nos la descargaremos desde el servidor. Y para utilizarla se accede a un directorio interno de la aplicación en el dispositivo.

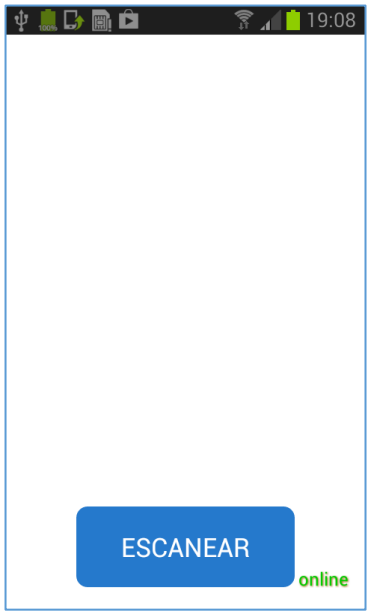

Figura 9.2. Layout activity_main.xml

En el caso de que se haya descargado la imagen del evento (siempre estará en la misma carpeta) se relaciona con el ImageView del layout y se establece como imagen en el MainActivity. En el caso de que no exista la imagen porque aún no la hemos descargado, se representará el ImageView vacío y se mostrará el color de fondo de la actividad que en este caso es blanco.

Como se puede ver, también hay un campo que mostrará si la app se encuentra en **modo online** o **modo offline**. Este estado se elige manualmente desde el menú. Se ha diseñado para poder trabajar con la app aunque no tengamos internet o incluso si el servidor se ha caído.

Si se pulsa el botón **ESCANEAR** inmediatamente se pasa a la actividad que contiene el lector para escanear una invitación llamada Lector.class que describiremos más adelante. También se pasa el valor de la variable `opcion` con el valor de si se queremos un funcionamiento online u offline de la app. Este string llamado *opcion* puede tiener el valor "`online`" u "`offline`". En el caso de ejecutarse la aplicación por primera vez, el string estará vacío y se le asigna el modo online por defecto. Para el resto de ejecuciones el modo se mantiene igual que cuando se sale de la aplicación porque se guarda el valor de la variable.

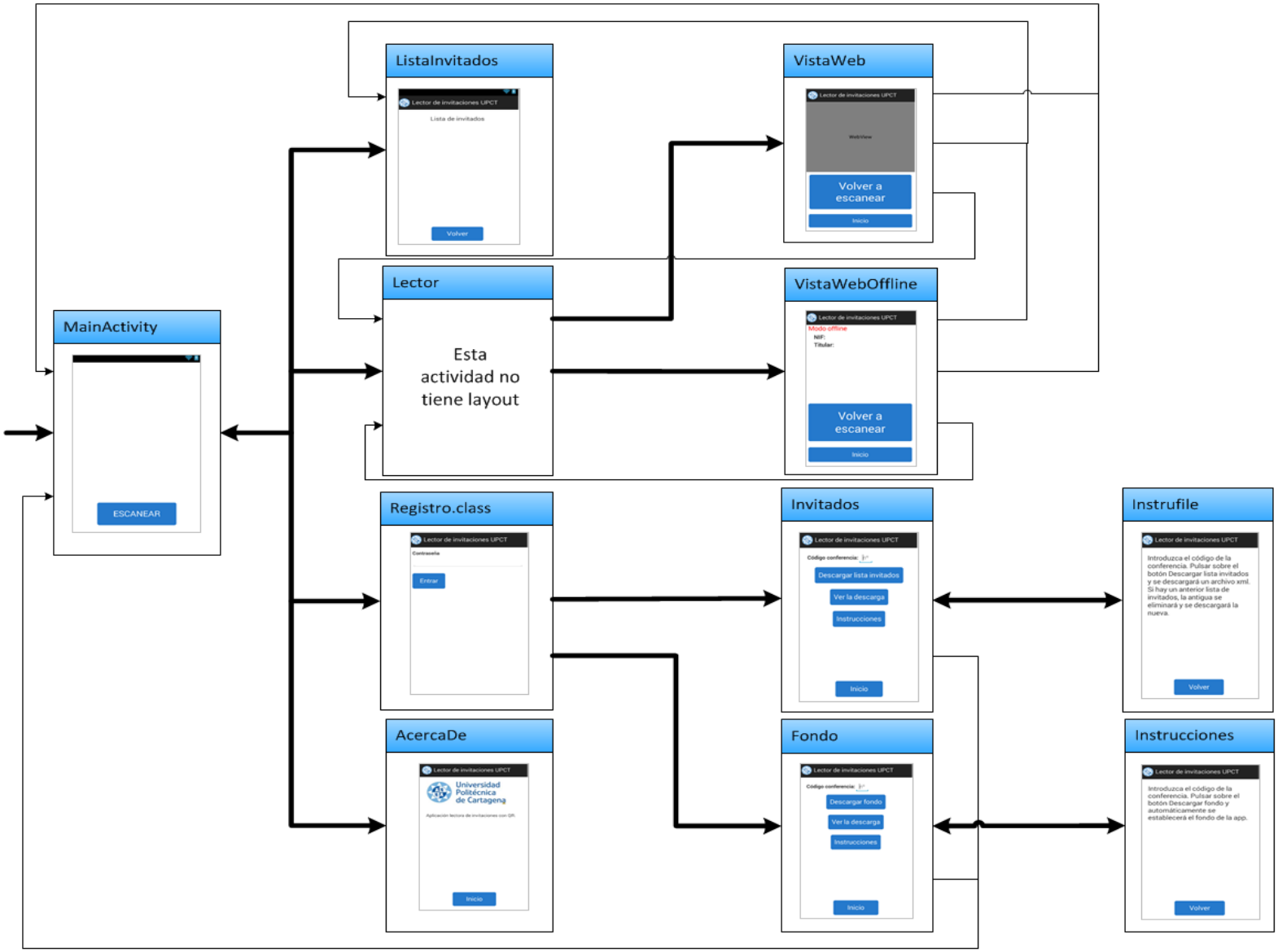

Figura 9.3.Diagrama de flujo de actividades. Caso

Para guardar una variable en Android se han usado los métodos **onPause()** y **onResume()** porque es la forma más fiable. En Android, al moverse de actividad en actividad y luego regresa a la primera actividad, es posible que el sistema haya eliminado el proceso que ejecutaba la actividad. En este caso, el proceso será creado de nuevo, pero se habrá perdido su estado, es decir, se habrá perdido el valor de sus variables y el puntero de programa. Por este motivo guardamos el valor de la variable. Primero en el método onPause() creamos un fichero con un nombre y un modo de acceso. Este método se ejecuta cuando la actividad MainActivity pasa a segundo plano porque es llamada otra actividad.

```
FileOutputStream fos;
try {
	fos = openFileOutput(FILENAME, Context.MODE_WORLD_WRITEABLE);
	fos.write(opcion.getBytes());		//Escribimos en el fichero
	fos.close();				//Cerramos el fichero
} catch (FileNotFoundException e) {
	e.printStackTrace();
} catch (IOException e) {
	e.printStackTrace();
}
```

Android guarda por defecto los ficheros creados en una ruta determinada, que sigue el siguiente patrón:

`/data/data/paquete.java/files/nombre_fichero`

Para recuperar los datos guardados, en el método onResume() simplemente tenemos que leer el fichero. Esto se realiza usando el método openFileInput() para abrir el fichero y usando los métodos de lectura de `java.io` para leer el contenido. En este caso:

```
FileInputStream fis;
try {
	fis = openFileInput(FILENAME);
	int c;
	String temp="";
	while( (c = fis.read()) != -1){
		temp = temp + Character.toString((char)c);  //Guardamos en temp lo que
	}                                                    //hay en el fichero
	fis.close();
	opcion = temp;
	text.setText(opcion);                              //Se muestra en pantalla
} catch (FileNotFoundException e) {
	e.printStackTrace();
} catch (IOException e) {
	e.printStackTrace();
}
```

En los anexos se encuentra una breve explicación de cómo es el ciclo de vida de una actividad en Android.

En esta actividad se ha añadido un método público llamado onBackPressed(). Este método es el que se ejecuta cuando pulsamos el botón 'Atrás' de los dispositivos Android. En esta actividad se ha deshabilitado el botón.

Y por último, en el MainActivity se encuentra el menú. Las opciones que tiene son:

- Online: Cambia el estado a modo online
- Offline: Cambia el estado a modo offline
- Configuración

  - Establecer fondo: Se descarga la imagen para el fondo de pantalla y se pone automáticamente. Se necesita contraseña.
  - Descargar lista de invitados. Se descarga la lista de invitados. Se necesita contraseña.
- Lista de invitados: Va a otra actividad que muestra la lista de invitados.
- Acerca de
- Salir

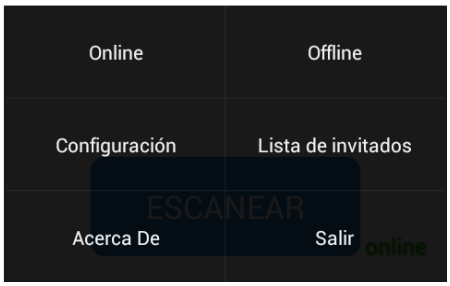

Figura 9.4. Menú MainActivity

Lector

Lector es una actividad pero que no tiene layout. Esta actividad se utiliza para llamar al lector QR que hay instalado en el dispositivo. También se puede añadir el lector en la app y no se necesita instalar uno pero de este modo la app ocupará mayor espacio y prácticamente todos los dispositivos disponen de un lector QR siendo innecesario tener otro lector en nuestra app.

Sin embargo, en el Caso 2) Registro de dispositivos móviles de un usuario basado en QR se puede encontrar un ejemplo de cómo implementar el lector de códigos QR en la propia aplicación.

En la actividad se llama al lector, en nuestro caso Barcode Scanner, con las líneas:

```
//Utilizamos un lector QR que hay instalado en el dispositivo
Intent intent = new Intent ("com.google.zxing.client.android.SCAN");
intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
startActivityForResult(intent,0); //El 0 es para indicar que cuando finalice la
actividad debe volver a ésta y funciona como código identificativo de la solicitud
```

Invocamos el componente de ZXing para que vaya a un lector QR que tengamos ya instalado en el dispositivo móvil y nos devuelva los resultados.

Una vez llamado el lector y leído el QR, vuelve a Lector.java con el resultado y dependiendo de si estamos en modo online u offline iremos a las actividades VistaWeb o VistaWebOffline respectivamente.

```
public void onActivityResult (int requestCode, int resultCode, Intent intent) {
              if (requestCode == 0) {
                     if (resultCode == RESULT_OK) {
                            String contents = intent.getStringExtra("SCAN_RESULT");
                            if (modos.equals("offline")){
                                   Intent i = new Intent (Lector.this,
VistaWebOffline.class);
                                   i.putExtra( "contenido", contents.toString());
                                   startActivity(i);
                                   finish(); //Con esta sentencia eliminamos lo que
quede en caché de esta actividad.
                            }else if (modos.equals("online")){
                                   Intent i = new Intent (Lector.this,
VistaWeb.class);
                                   i.putExtra( "contenido", contents.toString());
                                   startActivity(i);
```

```
                                        finish();//Con esta sentencia eliminamos lo que
quede en caché de esta actividad.
                                }
                        }else if (resultCode == RESULT_CANCELED) {
                                finish();
                        }
                }
        }
```

VistaWeb

Esta actividad recibe el contenido del código QR el cual consiste en una cadena de texto con este formato http://www.upct.es/***evento***##**NIF**##**apellidos, nombre**##**hash**##**codigoevento** . De la cadena extraemos los tokens separados por "##" y se guardan los parámetros que más interesan como son el código de identificación (hash) y el código de conferencia (cada evento tiene un código distinto). Los campos NIF, apellidos y nombre del QR se usan en el modo offline que veremos más adelante. Para extraer los datos separados por "##" se hace así:

```
StringTokenizer tokens=new StringTokenizer(contents, "##");
while (tokens.hasMoreTokens()){                 //Si quedan más tokens
        tokens.nextToken();                     //Siguiente token
        contador ++;                            //Incrementamos el contador

        if(contador == 4){                      //Los primeros tokens no los usamos
                codid = tokens.nextToken();     //Guardamos el cód. identificación
                codconf = tokens.nextToken();   //y el código de conferencia
                contador = 0;
        }
}
```

Una vez que se tienen estos parámetros, se cargan en el WebView (se puede apreciar en la figura del layout) la URL de nuestro servidor con el hash y con el código de conferencia:

```
myWebView.loadUrl("https://upctportal.upc
t.es/upct/conferenciaQrServlet?id="+codid
+"&codConferencia="+codconf);
```

Y el servidor es el encargado de mostrar si es válida o no la invitación. En el caso de no tener en ese mismo instante internet o se produce algún error, el WebView mostrará una página de error.

En el caso de pulsar el botón *Inicio*, la actividad se cierra y vuelve a MainActivity. Y si se pulsa *Volver a escanear*, la actividad se cierra y vamos a la clase Lector de nuevo. Este último botón es para que sea más rápido leer las invitaciones y los invitados fluyan prácticamente igual que si no necesitaran invitación.

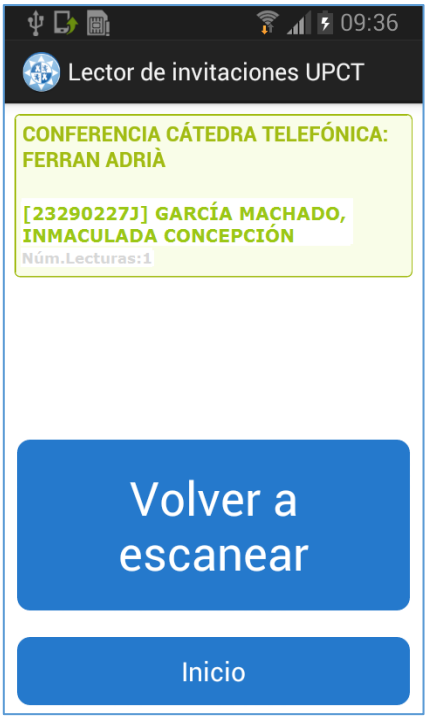

Figura 9.5. Layout vistaweb.xml

En esta actividad podemos encontrar en su menú la opción de mostrar la lista de invitados. Esto significa ir a otra actividad que muestra la lista de invitados llamada ListaInvitados.class que se explicará más adelante.

VistaWebOffline

Esta actividad realiza la misma función que VistaWeb pero en esta actividad no se trabaja con internet, en este caso se trabaja con un fichero que contiene la lista de invitados y que se ha tenido que descargar con anterioridad cuando sí se disponía de internet. Esto se usaría en el caso de que el servidor se caiga o que el dispositivo móvil no disponga de ningún tipo de acceso a la red.

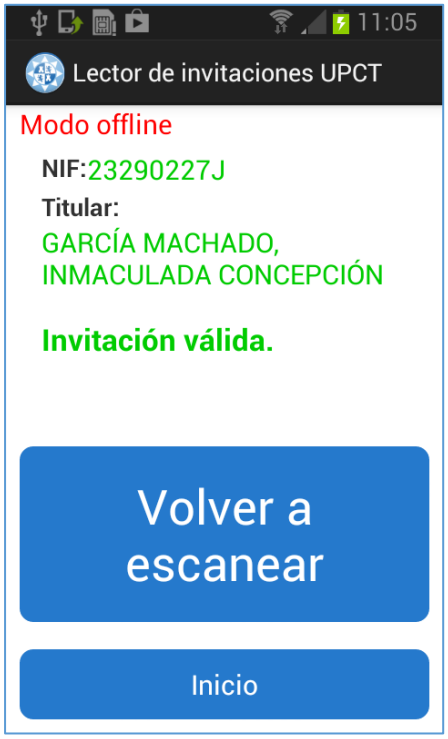

Figura 9.6. Layout vistaweboffline.xml

Para saber si la invitación es válida o no sin tener conexión en ese momento se necesita haber descargado antes un fichero el cual contiene una lista de invitados con sus datos y los hash de cada invitado.

En esta clase lo primero que se realiza es obtener el NIF, nombre y apellidos y el hash del código QR. Esto se realiza igual que en la clase VistaWeb.java.

A continuación, hay que leer el fichero de invitados. Es un fichero XML creado por nuestro servidor y que se ha descargado con anterioridad cuando sí había conexión y que se verá como se descarga en otra actividad.

Para poder leer un fichero XML se necesita hacerlo en una tarea asíncrona para cargarlo en segundo plano. Esto se debe a que desde la versión 3 de Android se estableció una política que no permite a las aplicaciones ejecutar operaciones de larga duración en el hilo principal que puedan bloquear temporalmente la interfaz de usuario, es decir, se hace el trabajo duro en segundo plano dejando libre el hilo principal de la aplicación y, de este modo, no afecta a la interfaz.

Para realizar esto la forma más sencilla es creando una clase privada dentro de nuestra clase que extienda de AsyncTask. En su método `doInBackground()` se parsea el archivo XML y guardamos los datos en una lista.

```
protected Boolean doInBackground(String... params){
        RssParserDom dom = new RssParserDom();
        clasepruebas = dom.parse();
        return true;
}
```

Una vez que doInBackground() ha terminado se pasa al método onPostExecute() para tratar la lista de invitados y poder averiguar si la invitación es válida o no.

Para averiguarlo se busca en la lista el hash que se encuentra en el código QR de la invitación. Si está en la lista se muestran los datos del invitado que hay en la lista en el layout y se indica que es una invitación válida. Todos estos campos en color verde como se puede apreciar en la figura anterior.

En el caso de que no coincidieran los hash de la lista con el de la invitación, mostramos los datos del invitado que hemos obtenido del código QR y se indicaría que es una invitación no válida. Todo esto en color rojo.

```
protected void onPostExecute(Boolean result){
	textfile.setText("");

	for (int i=0; i<clasepruebas.size(); i++){
		String hashs = clasepruebas.get(i).getHash();
		String nifs = clasepruebas.get(i).getDni();
		if (hashs.indexOf(tokenhash)!=-1){
		//Si el hash del qr y el de la invitación coinciden
			edtdni.setText(nifs);
			edttitu.setText( clasepruebas.get(i).getNombre());
			textfile.setText("Invitación válida.");
			edtdni.setTextColor(Color.parseColor("#00CC00"));//Color verde
			edttitu.setTextColor(Color.parseColor("#00CC00"));//Color verde
			textfile.setTextColor(Color.parseColor("#00CC00"));//Color verde
			break;
		}else {//Si los hash no coinciden lo indicamos y mostramos el nombre y
NIF del titular
			edtdni.setText(tokendni);
			edttitu.setText(tokentitu);
			textfile.setText("Invitación NO válida.");
			edtdni.setTextColor(Color.parseColor("#FF0000"));//Color rojo
			edttitu.setTextColor(Color.parseColor("#FF0000"));//Color rojo
			textfile.setTextColor(Color.parseColor("#FF0000"));//Color rojo
		}
	}
}
```

En esta actividad podemos encontrar en su menú la opción de mostrar la lista de invitados. Esto significa ir a otra actividad que muestra la lista de invitados llamada ListaInvitados.class que se explicará a continuación.

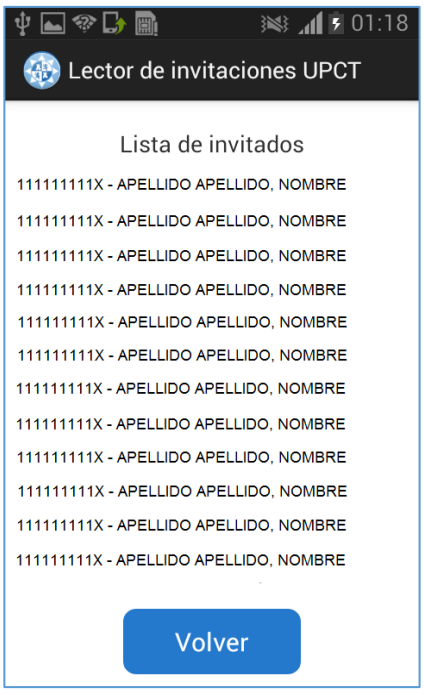

Figura 9.7. Layout listainvitados.xml

ListaInvitados

Esta actividad muestra la lista de invitados del evento cargada desde un archivo XML descargado desde el servidor. Lo primero es comprobar si existe el archivo invitaciones.xml que es el archivo que contiene la lista de invitados.

Si no se ha descargado ninguna lista de invitados muestra un mensaje “No hay lista de invitados” y se sale de la actividad. En caso contrario, muestra la lista de invitados con el NIF y el nombre y apellidos de cada invitado con el método mostrarlista().

El método mostrarlista() ejecuta una clase privada asíncrona la cual parseará el archivo XML usando otra

clase llamada RssParserDom.java y a continuación mostrará la lista que contiene el archivo con un formato determinado (NIF – APELLIDO APELLIDO, NOMBRE) en el TextView destinado a la lista. El código más significativo de esta clase es:

```
private void mostrarlista() {

                //List<ClasePruebas> clasepruebas;
                //Tarea Asíncrona para cargar un XML en segundo plano
                CargarXmlTask tarea = new CargarXmlTask();
                tarea.execute();
        }

        //Tarea Asíncrona para cargar un XML en segundo plano
        private class CargarXmlTask extends AsyncTask<String, Integer,Boolean>{

                protected Boolean doInBackground(String... params){
                        RssParserDom dom = new RssParserDom();
                        clasepruebas = dom.parse();
                        return true;
                }

                protected void onPostExecute(Boolean result){
                        txtView5.setText("");
                        for (int i=0; i<clasepruebas.size(); i++){
                                txtView5.setText(txtView5.getText().toString()+
clasepruebas.get(i).getDni()+" - "+ clasepruebas.get(i).getNombre()+
System.getProperty("line.separator")+System.getProperty("line.separator"));
                        }
                }
        }
```

A esta actividad se puede acceder desde el MainActivity, VistaWeb y VistaWebOffline.

Registro

Esta actividad es para ingresar una contraseña. La contraseña es necesaria para poder descargar el fondo de pantalla del evento y para poder descargar la lista de invitados. Se ha querido hacer así para que sean los administradores los encargados de descargar lo necesario y comprobar si hay algún problema y preparar el dispositivo para el evento.

Esta actividad aparecerá cada vez que se pulse en el menú de la actividad MainActivity en ‘Configurar’ -> ‘Descargar fondo’ o ‘Configurar’ -> ‘Descargar lista’. Está entre la pantalla principal y las actividades *Invitados* y *Fondo (*se explicarán más adelante). Para poder acceder a ambas actividades necesitamos una contraseña y en vez de tener dos actividades iguales donde te piden la contraseña se ha hecho una. Para saber si la siguiente actividad es *Invitados* o *Fondo*, la actividad *MainActivity* pasará un parámetro a *Registro* donde indicará que actividad es la que se quiere.

Cuando la siguiente actividad es *Invitados*, se pasa un parámetro a esta actividad que es la contraseña. Esto se hace para poder usarla como semilla en una encriptación que se verá más adelante.

Figura 9.8. Submenú Configuración

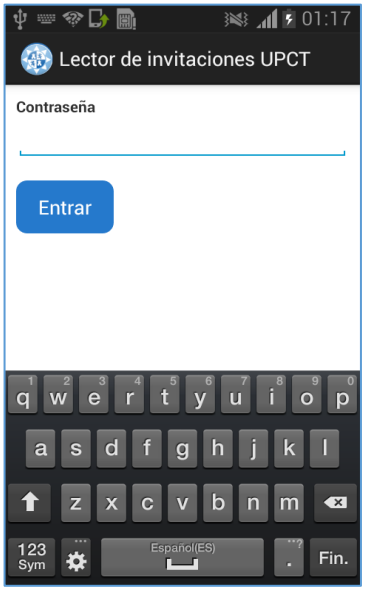

Figura 9.9. Layout registro.xml

Invitados

Esta actividad es una de las más complejas de la aplicación. En ella se puede descargar la lista de invitados. A continuación se explica poco a poco como se hace:

Cada evento de la UPCT tendrá un código de conferencia para poder distinguirlos y así podrá haber dos eventos en el mismo momento.

En esta clase lo más destacable es la obtención del parámetro semilla de la actividad Registro y la puesta en marcha del gestor de descargas **Download Manager**. Download Manager es un servicio del sistema que se encarga de descargas HTTP de larga duración. El Download Manager llevará a cabo la descarga en segundo plano y volverá a intentar la descarga si falla o si se producen cambios de conectividad o si se reinicia el sistema. La aplicación debe tener el permiso de acceder a internet para usar esta clase.

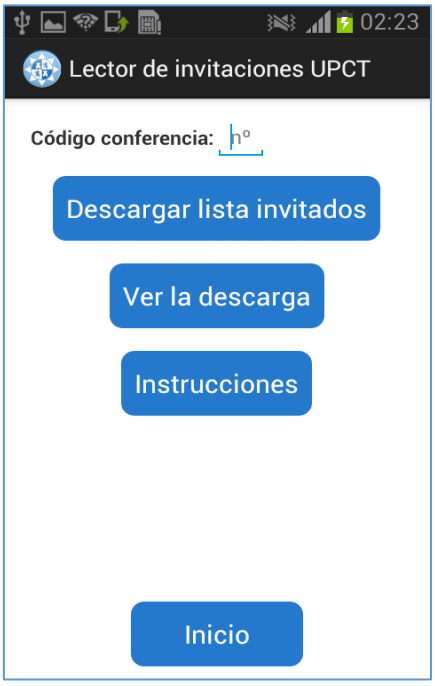

Figura 9.10. Layout invitados.xml

Al pulsar en 'Descargar lista invitados' entramos en el método descargar(). Se encripta la semilla con el código de conferencia para poder obtener un identificador.

```
id = UtilidadesCifrado.encrypt(semilla, codcon);
```

La clase UtilidadesCifrado se encuentra en el anexo. En ella se encuentran los métodos para encriptar y desencriptar usando el algoritmo AES que es un esquema de cifrado por bloques.

Una vez obtenido el identificador, este se usará en la URL que se usa para descargar el archivo XML de invitados. En el servidor veremos que también se realizará esta operación teniendo el archivo en la misma URL.

```
intent.putExtra("url",
"https://upctportal.upct.es/upct/conferenciaInscritosServlet?id=" + id);
```

Se dan valor a los parámetros obligatorio para realizar la descarga, como el directorio en el que se guardará y el nombre que tendrá el fichero descargado. También mostramos una rueda giratoria muestras se realiza la descarga. A esto se le llama ProgressDialog.

Gracias al DownloadManager, pulsando en el botón 'Ver la descarga', podemos ver cómo va el progreso de la descarga y los archivos descargados anteriormente.

Pulsando sobre el botón 'Instrucciones', la aplicación mostrará la actividad Instrufile la cual veremos a continuación.

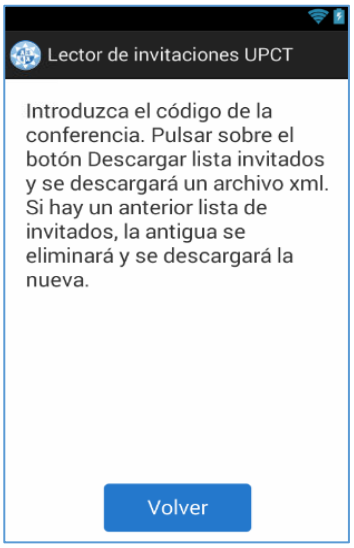

Figura 9.11. Layout instrufile.xml

Instrufile

Esta actividad es muy simple. Únicamente muestra las instrucciones para descargar el archivo de la lista de invitados.

Fondo

Esta actividad es muy similar a la actividad *Invitados*. En ella se descarga un fondo de pantalla relacionado con el evento.

El código es muy similar exceptuando que en este caso no se necesita semilla. No encriptamos el código de conferencia para sacar la URL de la imagen. En este caso no necesitamos seguridad porque es una simple imagen. En la clase Invitados sí debido a que contiene datos e información de los invitados.

Para indicar la URL donde se encuentra la imagen a descargar, en este caso, hemos usado otra forma pero realiza lo mismo:

```
DownloadManager.Request request = new DownloadManager.Request
(Uri.parse("https://uxxiportal.upct.es/portlets/images/conferencias/"+codcon+".png"))
;
```

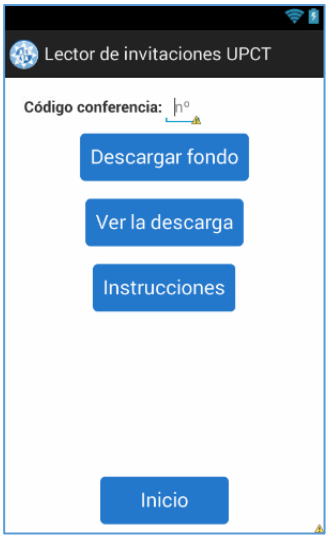

Figura 9.12. Layout fondo.xml

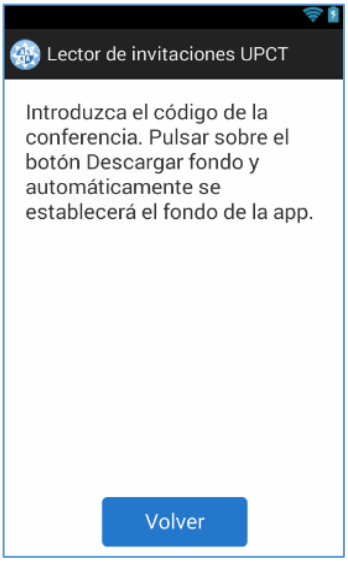

Figura 9.13. Layout instrucciones.xml

Instrucciones

Esta actividad es muy simple y es muy similar a Instrufile. Únicamente muestra las instrucciones para descargar el fondo de pantalla.

Esta clase es muy sencilla similar a Instrufile.java. Se indica cual es el layout de esta actividad, se declaran y se les da valor a sus componentes visuales (TextView y Button) y se indica que se debe hacer en el caso de que se pulse el botón 'Volver'.

Acercade

En esta actividad se muestra por pantalla información sobre la aplicación.

El código de esta actividad es muy sencillo. Simplemente indica el layout que corresponde a esta actividad y que se debe hacer en el caso de que se pulse el botón. Si se pulsa el botón Inicio la actividad finaliza y vuelve al a pantalla principal.

Figura 9.14. Layout acercade.xml

AndroidManifest.xml

El AndroidManifest.xml es un archivo de configuración donde se presenta la información esencial acerca de la aplicación para el sistema Android.

En él se describen los componentes de la aplicación como las actividades, servicios, etc. También declara los permisos que la aplicación tiene para poder acceder a las partes protegidas de la API e interactuar con otras aplicaciones.

En este caso hemos necesitado estos permisos:

- Acceso a la cámara. Este permiso se necesita para poder leer el código QR.

```
<uses-permission android:name="android.permission.CAMERA" />
```

- Acceso a Internet. Para conectarnos con el servidor y realizar las descargar.

```
<uses-permission android:name="android.permission.INTERNET" />
```

- Acceso al estado de la red. Permite a la aplicación a acceder a la información sobre las redes.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

- Acceso a leer el estado del teléfono.

```
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

- Acceso a leer en memoria externa. Para poder leer el archivo de la lista de invitados o la imagen de fondo de pantalla.

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

- Acceso a escribir en memoria externa. Para poder guardar, eliminar o sobrescribir el archivo de la lista de invitados o la imagen de fondo.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

- Acceso para desactivar el estado de inactividad. El dispositivo no debe ponerse en estado de inactividad, ni apagar la pantalla en mitad de una descarga.

```
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

9.8.2 Servidor

Este caso como se ha llevado a cabo en la UPCT se han creado dos servlet:

- ConferenciaInscritosServlet.java
- ConferenciaQrServlet.java

ConferenciaInscritosServlet.java

Este es el servlet para descargar el fichero con los inscritos al evento.

En el apartado de Invitados de la aplicación se vio como para poder descargar el archivo XML de la lista de invitados, primero encriptabamos el código del evento y como semilla usábamos la contraseña para acceder. Y ese resultado se lo pasamos al servlet en la URL:

```
intent.putExtra("url",
"https://upctportal.upct.es/upct/conferenciaInscritosServlet?id=" + id);
```

En este servlet se recupera el parámetro id, se desencripta y obtenemos el código de conferencia.

Después obtenemos los datos y la respuesta del servidor será un archivo XML con los datos llamado invitaciones.xml. Las líneas son:

```
byte[] data = null;
//Obtenemos los invitados
try {
	Clob file = serv.getInscripciones(new BigDecimal(1));
	data = IOUtils.toByteArray(file.getAsciiStream());;
} catch (SQLException e) {
	e.printStackTrace();
}
//la respuesta del servidor es un archivo XML de nombre invitaciones.xml
response.setContentType("application/xml");
response.setHeader("Content-Disposition", "attachment;filename=invitaciones.xml");
response.setContentLength(data.length);
//enviamos la respuesta
OutputStream output = response.getOutputStream();
output.write(data);
output.flush();
output.close();
```

ConferenciaQrServlet.java

Este servlet es para la lectura del código QR.

Se ha visto en la clase VistaWeb de la aplicación como cargábamos la URL de este servlet pasándole dos parámetros, uno el código de identificación (hash que nos da el código QR) y el otro el código de conferencia.

```
myWebView.loadUrl("https://upctportal.upct.es/upct/conferenciaQrServlet?id="+
codid+"&codConferencia="+codconf);
```

Este servlet obtiene los parámetros y comprueba que está inscripto el usuario con ese hash en ese evento. Si el evento aun no ha caducado se actualiza el estado de la inscripción aumentando en 1 la lectura de esa invitación y se muestra en el WebView de la aplicación una página llamada invitación.jsp que muestra si la invitación es válida o no.

Para relacionar la vista del servlet Qr con la clase java (lógica de negocio), se añaden las siguientes líneas de código en el fichero de configuración struts-config.xml

```
<action path="/conferenciaQrServlet" scope="request"
type="upct.uxxiportal.util.servlet.ConferenciaQrServlet">
	<forward name="irInvitacion" path="/htdocs/conferencia/invitacion.jsp"/>
</action>
```

La página invitación.jsp mostrará dos tipos de pantalla, uno para una invitación válida y otro para una invitación no válida:

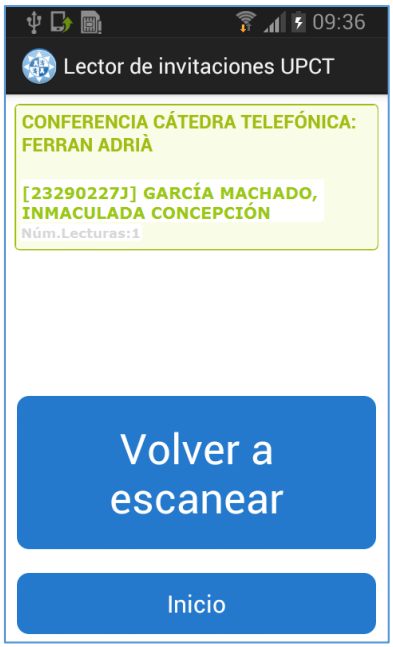

Figura 9.15. Pantalla invitación válida

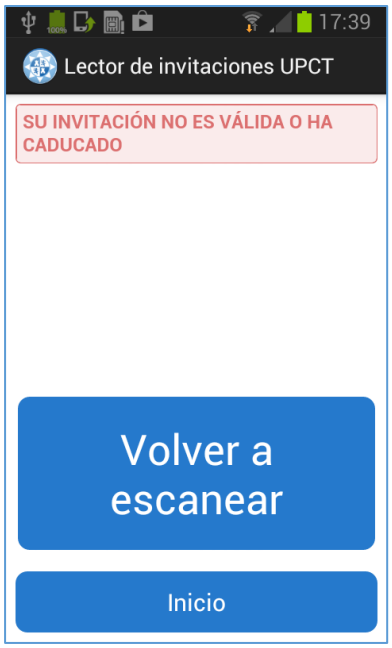

Figura 9.16. Pantalla invitación no válida

9.9 Explicación del código del caso 2

Este caso consiste en el registro de dispositivos móviles de un usuario basado en QR.

El funcionamiento consiste en:

Un usuario se descarga la app y la ejecuta. Inmediatamente hace una consulta al servidor para saber si ese dispositivo está registrado en la base de datos. Envía el ID del dispositivo que es el código de identificación alfanumérico especifico asociado al dispositivo móvil. Y también envía el nombre del modelo para hacer una doble comprobación.

El servidor devolverá unos parámetros en los cuales se encuentra un estado que indica si el dispositivo está registrado o no en la base de datos. Si está registrado continuamos en la pantalla principal de la aplicación. Pero si no está registrado se va a otra actividad donde aparecerá un mensaje de las instrucciones que debe hacer para registrarse y un botón para escanear el QR personal. Este QR lo genera el servidor y para cada usuario es distinto.

Para realizar el proyecto se ha hecho en un servidor local, pero está pensado para que los usuarios encuentren el QR personal en el Portal de Servicios como se hizo en el primer caso.

El dispositivo móvil debe llevar cámara para poder escanear el QR desde un ordenador una vez autenticado el usuario en el Portal de Servicios.

La aplicación lee el QR y enviamos los datos extraídos del QR, el identificador y el modelo del dispositivo y lo envía al servidor.

El servidor verifica los datos y si todo es correcto registra ese dispositivo y lo vincula a ese usuario. Y entraría en la pantalla principal.

Podría usarse para varias aplicaciones que requieran autenticación. En este caso, se ha hecho un pequeño ejemplo de aplicación para ver las asignaturas.

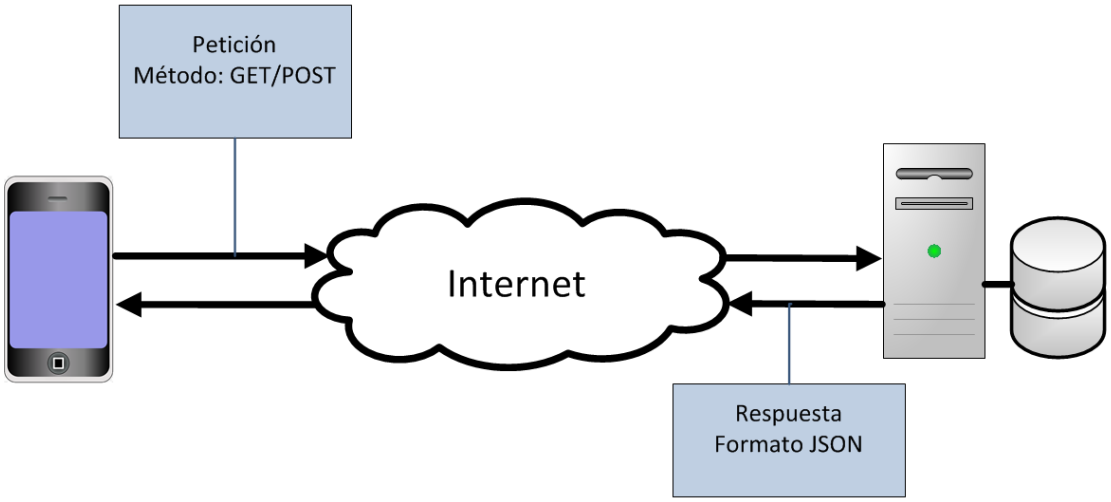

Figura 9.17. Esquema del funcionamiento

Diagrama de flujo del funcionamiento:

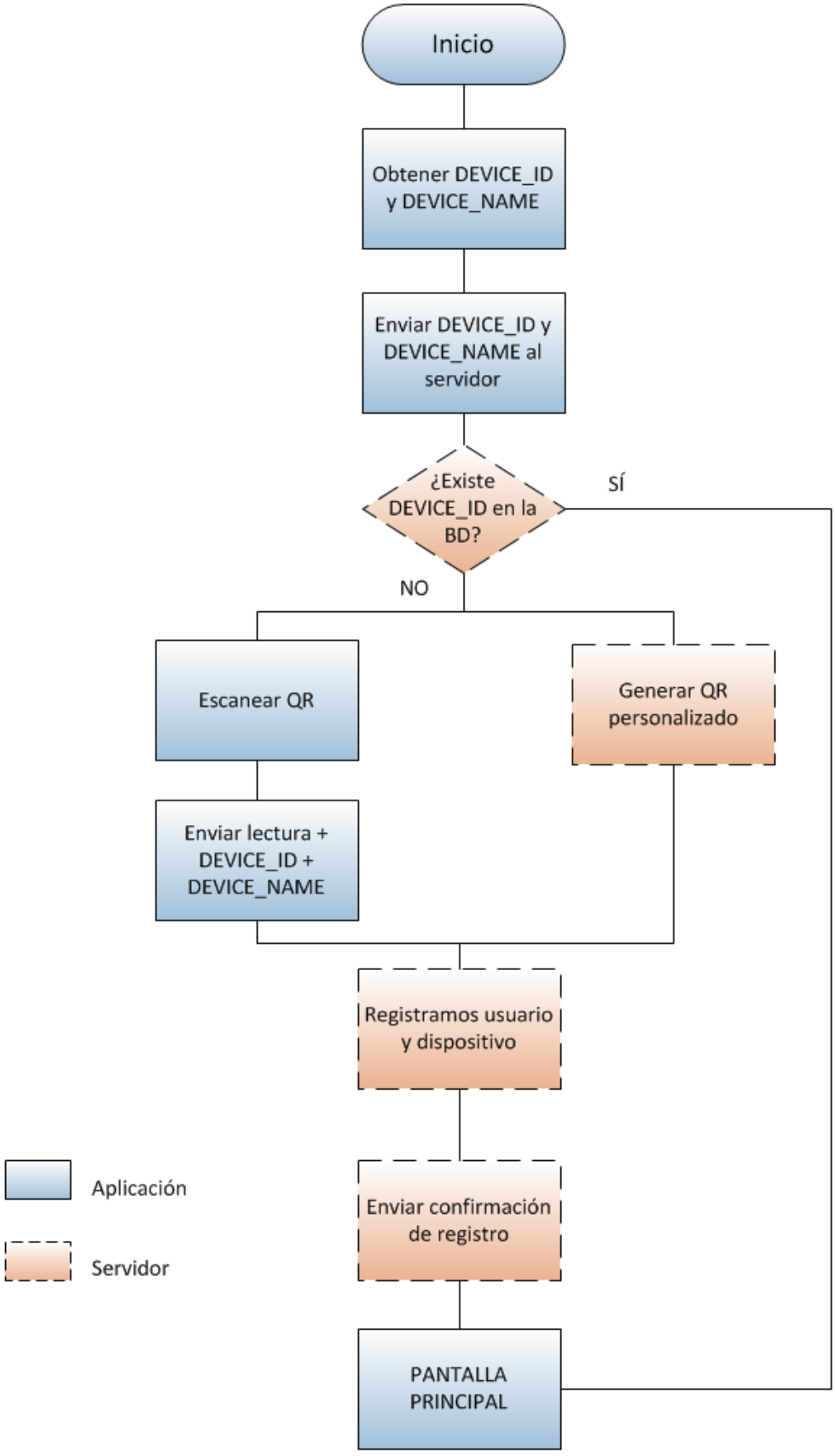

Figura 9.18. Diagrama de flujo. Caso 2.

Se va a empezar explicando la aplicación móvil y después se verá la parte del servidor local.

9.9.1 Aplicación móvil

En este proyecto disponemos de 10 actividades o clases: Inicio.java, MainActivity.java, Registro.java, Principal.java, ImageAdapter.java, Expediente.java, Asignaturas.java, Estudiante.java, Configuracion.java y Httppostaux.java.

Inicio

Comenzaremos explicando la actividad Inicio porque es la primera actividad que se ejecuta como se puede observar en el AndroidManifest.xml

Esta actividad comienza obteniendo el DEVICE_ID y el DEVICE_NAME.

El DEVICE_ID es el IMEI del dispositivo móvil o es el número de serie si no dispone de IMEI como es el caso para una tablet sin conexión a la red telefónica. El DEVICE_ID es un número único para cada dispositivo, aunque puede ser que al cambiar el sistema de arranque al dispositivo se convierta en uno genérico. Por esta razón también se comprueba el DEVICE_NAME.

Inmediatamente después envía al servidor los datos para saber si este dispositivo está registrado por un usuario o no.

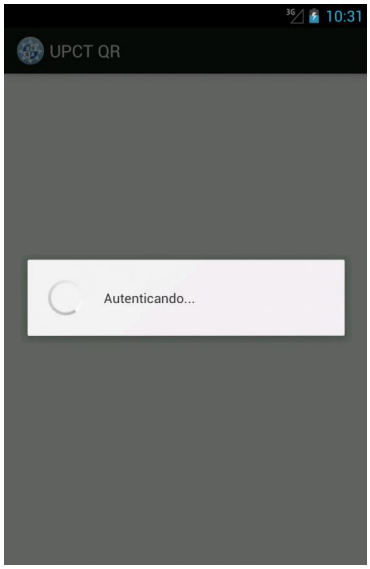

Figura 9.19. Layout l_inicio.xml

Para enviar los datos al servidor se realiza en una clase embebida subclase de AsyncTask. Esta clase se caracteriza por tener tres métodos: onPreExecute(), doInBackground y onPostExecute(). Como sus nombres indican se ejecutan en orden.

En el método doInBackground() definimos lo que se hará en segundo plano. El envío de datos y la espera de la respuesta se harán en este método.

```
protected String doInBackground(String... params) {
        //obtenemos el device_id y el device_name
        dev_id =params[0];
        dev =params[1];

        //enviamos, recibimos y analizamos los datos en segundo plano
        if(envio(dev_id, dev)==true){
                return "ok"; //device ya registrado
        }else if(conexion == false){ // fallo de conexión
                return "falloconexion";
        }else{
                return "no"; //device no registrado. Hay que registrarlo
        }
}
```

Se realiza el método envio() que devuelve un boolean. Si devuelve **true** significará que está registrado y si devuelve **false** puede ser porque hay un fallo de conexión o no está registrado. El método envio():

```
public Boolean envio (String device_id , String device ){
	int devstatus = -1;
	//Enviamos los parámetros al servidor
	ArrayList<NameValuePair> postparameters2send = new ArrayList<NameValuePair>();
	postparameters2send.add(new BasicNameValuePair("device_id",device_id));
	postparameters2send.add(new BasicNameValuePair("device",device));
	JSONArray jdata = post.getserverdata(postparameters2send, URL_connect);

	SystemClock.sleep(950); //Para simular el tiempo de espera. Eliminar

	if (jdata!=null && jdata.length() > 0){
		JSONObject json_data;

		try{
			json_data = jdata.getJSONObject(0);
			devstatus = json_data.getInt("devstatus");
			json_data = jdata.getJSONObject(1);
			nombre = json_data.getString("NOMALU");

		}catch (JSONException e){
			e.printStackTrace();
		}

		if(devstatus == 0){
			Log.e("devistatus ", "no registrado");
			return false;
		}else {
			Log.e("devistatus ", "registrado");
			return true;
		}
	}else {
		Log.e("JSON ", "ERROR");
		Log.e("Error" , "Error en la conexión");
		conexion = false;
		return false;
	}
}
```

Se crea un ArrayList del tipo nombre valor para agregar los datos DEVICE_ID y DEVICE_NAME y se envían mediante POST a nuestro servidor. Se realiza una petición y como respuesta se obtiene un array JSON.

A continuación paramos el proceso unos segundos para dar realismo al tiempo que tardaría en recibir la respuesta. Ya que las pruebas se han realizado en un servidor local la respuesta es inmediata. Se elimina esa línea en el caso de que se lleve a la práctica.

Las líneas siguientes analizan el array JSON recibido. Primero se obtiene el objeto 0 que corresponde con el estado del dispositivo (si está registrado o no) y el segundo objeto es el nombre del usuario. Si no estuviera registrado el dispositivo este objeto estará vacío. Si devstatus es igual a 0 significa que no está registrado el dispositivo y devolveremos a doInBackground() **false**. Si devstatus es igual a 1 significa que si está registrado y devolveremos **true**. En el caso de que no se recibiera ningún dato significa que hay un error de conexión y devolvemos **false** y además cambiamos el estado de un boolean creado llamado conexión a **false** para que el método doInBackground sepa diferenciar los estados.

En el método onPostExecute() se analiza el resultado del método doInBackground() e indicamos que hacer según el resultado.

```
protected void onPostExecute(String result) {

        pDialog.dismiss(); //ocultamos el progress dialog.

        if(result.equals("no")){
                //Device NO registrado
                //Invocamos MainActivity para leer el QR
                Intent i = new Intent(Inicio.this, MainActivity.class);
                i.putExtra("device_id", dev_id);
                i.putExtra("device", dev);
                startActivity(i);
                finish();

        }else if (result.equals("falloconexion")){
                fallodeconexion();
        }else{
                //Device registrado
                Intent i = new Intent(Inicio.this, Principal.class);
                i.putExtra("nombre", nombre);
                i.putExtra("dnialu", nif);
                startActivity(i);
                finish();
        }
}
```

En el caso de que esté registrado se invoca a la actividad Principal.class y si no está registrado se invoca la actividad MainActivity.class.

Si no hubiera conexión a Internet o el servidor no responde se utiliza el método fallodeconexion(). En estos casos se mostrará un mensaje que dice: “En estos momentos es imposible la conexión. Por favor, busque una buena conexión a Internet o inténtelo de nuevo más tarde". Y se saldrá de la aplicación.

Mientras se comprueba si está registrado o no aparecerá un ProgressDialog y mostrará una rueda de progreso y un mensaje: “Autenticando…”.

MainActivity

Si el dispositivo no está registrado hay que registrarlo y vincularlo a un usuario. Para la autenticación no usamos el NIF ni la contraseña de la UPCT en esta aplicación pero sí para acceder al Portal de Servicios. Usamos un modo de autenticación mediante códigos QR en el cual el usuario se autentica pero en el Portal de Servicios de la UPCT desde un ordenador y ahí aparecerá un QR personalizado. Como se ha dicho, nunca se ha introducido esta aplicación al Portal de Servicios, esto sería en el caso de que se llevara a cabo.

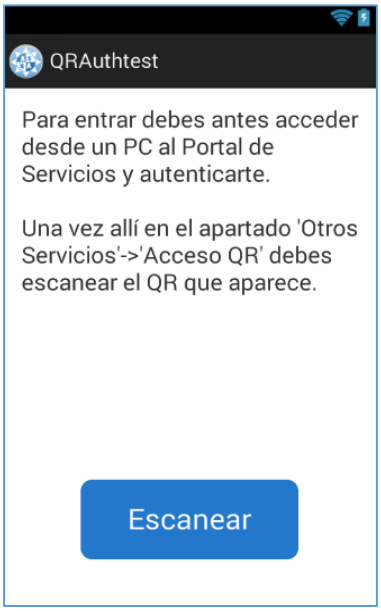

Figura 9.20. Layout activity_main.xml

En esta actividad escaneamos el QR personalizado y si la lectura es correcta invocamos a Registro.class.

Esta actividad es similar a la actividad Lector.java del

caso anterior pero en esta actividad el lector si va implementado en la aplicación.

Para implementar el lector en la aplicación hemos realizado una serie de pasos antes como descargarnos la librería de ZXing e importarla como un proyecto en el workspace indicando que es una librería. También se necesita incluir en el proyecto el archivo core.jar de esta librería.

Una vez hecha la librería se tiene que incluir en el proyecto.

Al hacer click en el botón invocamos a la clase SCAN que se encuentra en la librería de ZXing y esto nos llevará al lector de la librería incluida.

```
btnQR = (Button) findViewById(R.id.btn_qr);
btnQR.setOnClickListener(new OnClickListener() {
	@Override
	public void onClick(View v) {
		Intent intent1 = new Intent("com.google.zxing.client.android.SCAN");
		intent1.putExtra("SCAN_MODE", "QR_CODE_MODE");
		startActivityForResult(intent1, 0);
			}
		});
```

Figura 9.21. Captura de pantalla del lector de QR

Se escanea el código e inmediatamente devuelve el resultado a la actividad MainActivity. También devuelve unos parámetros que son los que indican si todo ha salido bien o ha ocurrido algún problema. Si todo ha salido correctamente invocamos a la actividad Registro y le enviamos los parámetros necesarios como la lectura del QR, el DEVICE_ID y el DEVICE_NAME.

Registro

Para realizar el registro del dispositivo obtenemos todos los datos necesarios de MainActivity. Ahora se tienen que enviar al servidor para que realice la operación. Si se registra correctamente se dirige a la actividad Principal.java donde encontramos la pantalla principal.

Pero antes de enviar los datos al servidor tenemos que adaptar la información obtenida por el QR: El resultado del lector de QR es un string el cual tenemos que dividir en substrings o tokens. Están separados por "##" como en el caso 1. Si esto cambia, como se puede observar es muy fácil su modificación.

```
StringTokenizer tokens=new StringTokenizer(contents, "##");
while (tokens.hasMoreTokens()){
       nif = tokens.nextToken();
       usuario = tokens.nextToken();
       hash = tokens.nextToken();
}
```

Una vez hecho esto, se ejecuta el método enviardatos(). Este método recibe los parámetros para el registro y si ninguno está vacío ejecuta la clase embebida asyncregistro(). Esta clase está encargada de enviar en segundo plano los parámetros al servidor, similar a la actividad Inicio con sus tres métodos de ejecución: onPreExecute(), doInBackground y onPostExecute().

Mientras se espera la respuesta se muestra en pantalla un ProgressDialog como en otras pantallas. La respuesta puede ser que se ha registrado correctamente o no. Si se ha registrado correctamente invocamos a la actividad Principal y si no se ha registrado se muestra un mensaje: "No se puede registrar. Inténtelo de nuevo más tarde".

Principal

Esta actividad consiste en un menú en el cual se puede encontrar un acceso para ver las asignaturas, los datos personales del estudiante, la configuración y un botón para salir de la aplicación.

En este layout se pueden encontrar dos TextView y un GridView. Un GridView muestra datos a modo de rejilla bidimensional. Los datos provienen de un Adapter personalizado, en este caso, un ImageAdapter del cual se hablará después.

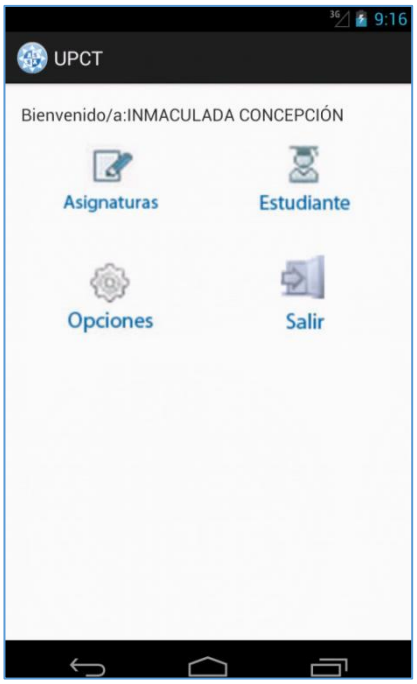

Figura 9.22. Layout principal.xml

Podemos acceder a esta actividad desde dos actividades distintas. Una es desde *Registro.java* porque es la siguiente actividad después del registro y la otra actividad es desde *Inicio.java* cuando iniciamos la aplicación e *Inicio* comprueba que este dispositivo ya está registrado por un usuario.

Lo siguiente más importante es montar el GridView. Para ello le indicamos que use el ImageAdapter antes nombrado. Con este adapter se llena el GridView con datos. Un nuevo objeto de la clase ImageAdapter, definido ahí, detectará la pulsación (con un listener) y la posición. Dependiendo de la posición de la pulsación se hará una cosa u otra.

```
GridView gv = (GridView)findViewById(R.id.gridView1);
gv.setAdapter(new ImageAdapter(this));

gv.setOnItemClickListener(new OnItemClickListener() {
	public void onItemClick(AdapterView<?> parent, View v, int position, long id)
{
		if (position == 0){ // Asignaturas
			Intent intent = new Intent(Principal.this, Expediente.class);
			intent.putExtra("nif", dnialu);
			startActivity(intent);
		}else if(position == 1){ // Estudiante
			Intent i = new Intent (Principal.this, Estudiante.class);
			i.putExtra("dnialu",dnialu);
			startActivity(i);
		}else if (position == 3){  // Salir
			Intent intent2 = new Intent(Intent.ACTION_MAIN);
			intent2.addCategory(Intent.CATEGORY_HOME);
			intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
			startActivity(intent2);
			finish();
		}else{  // Configuración
			Intent i = new Intent(Principal.this, Configuracion.class);
			i.putExtra("dnialu",dnialu);
			startActivity(i);
		}
	}
});
```

ImageAdapter

Esta clase se encarga de obtener todas las imágenes que cargaremos en el GridView. La clase ImageAdapter extiende a BaseAdapter. Un BaseAdapter puede usarse para un Adapter en un listview o un gridview. En él hay que implementar algunos métodos heredados de la clase Adapter porque BaseAdapter es una subclase de Adapter. Estos métodos son: getCount(), getItem(), getItemId() y getView(). El código se puede encontrar en el anexo.

De los métodos que contiene el más importante es el getView(). Este método crea un nuevo ImageView para cada elemento añadido al ImageAdapter.

Al final de la clase encontramos las referencias a las imágenes. En un array de enteros se guardan los número que son los id de todos los recursos guardados en drawable. Dentro de la carpeta res/drawable están todas las imágenes con esos nombres.

```
private Integer[] mThumbIds = {

		R.drawable.asignaturas,
		R.drawable.estudiante,
		R.drawable.opciones,
		R.drawable.salir
	};
```

Expediente

A continuación vamos a ver la clase Expediente. A esta clase se accede pulsando sobre

Asignaturas en la clase Principal. En esta clase encontraremos los planes del alumno y el número de expediente del alumno de ese plan en forma de lista. Pulsando sobre uno de ellos se accede a las asignaturas cursadas o matriculadas del alumno.

Un ListView visualiza una lista deslizable verticalmente de varios elementos, donde cada elemento puede definirse como un Layout. En nuestro caso se llamará simple_list.xml. Para crear un ListView se tienen que seguir unos pasos:

- Diseñar un layout que contenga al ListView (expediente.xml).
- Diseñar un layout individual para cada fila de la lista que se repita (simple_list.xml).
- Implementar la actividad que visualice el layout con el ListView (en Expediente.java).

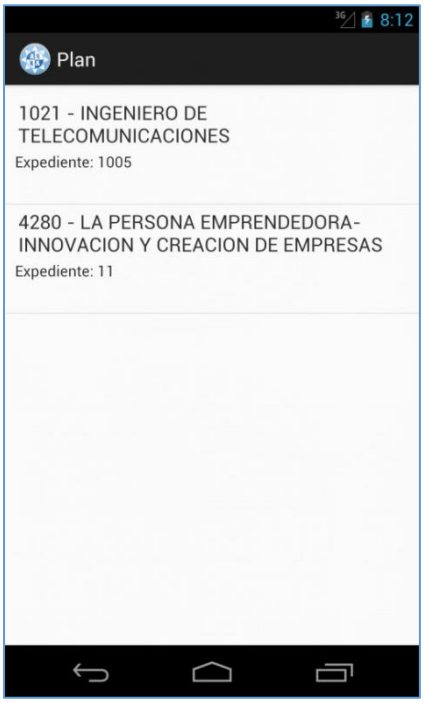

Figura 9.23. Layout expediente.xml

Una vez definidos los layouts tenemos que implementar la actividad y relacionarlo con el ListView. Toda actividad que visualice un ListView ha de heredar de ListActivity. También tiene que llamar al método setListAdapter() para indicar el adaptador con la lista de elementos a visualizar. Se crea un adaptador SimpleAdapter para crear las vistas del ListView a partir de los datos almacenados en un array. El constructor de SimpleAdapter tiene cinco parámetros:

- El primero es un Context con información sobre el entorno de la aplicación. Se utiliza como contexto la misma actividad que hace la llamada.
- El segundo parámetro es un ArrayList que contiene todos los datos.
- El tercero es el layout utilizado para representar cada elemento de la lista, en nuestro caso simple_list.xml.
- El cuarto es un array de strings que contiene una lista de los nombres de las columnas, que se añadirá al mapa asociado a cada elemento.
- Y por último un array de enteros que indica los TextView que deben mostrar las columnas del parámetro anterior.

```
String[] from=new String[] {"Plan","Expediente"};
int [] to=new int[]{R.id.plan,R.id.exp};

                        ……

 SimpleAdapter ListadoAdapter = new SimpleAdapter(this, Expedi,
R.layout.simple_list, from, to);
        setListAdapter(ListadoAdapter);
```

Después de tener la lista desarrollada, se tiene que crear un método para detectar la pulsación sobre un elemento de la lista.

En la clase ListActivity hay un método que es invocado cada vez que se pulsa sobre un elemento de la lista y es el que utilizamos.

```
@Override
```

```
protected void onListItemClick(ListView l, View v, int position, long id) {
    super.onListItemClick(l, v, position, id);

    Object o = getListAdapter().getItem(position);

     String objpul = o.toString();

     switch (position){
          case 0:
                 Intent intent = new Intent(this,Asignaturas.class);
            intent.putExtra("expe",exped2[0]);
            intent.putExtra("plan", exped2[1]);
            startActivity(intent);
            break;
          case 1:
                 Intent i = new Intent(this,Asignaturas.class);
            i.putExtra("expe",exped2[3]);
            i.putExtra("plan", exped2[4]);
            startActivity(i);
            break;
          case 2:
                 Intent inte = new Intent(this,Asignaturas.class);
            inte.putExtra("expe",exped2[6]);
            inte.putExtra("plan", exped2[7]);
            startActivity(inte);
            break;
     }
}
```

En este método obtenemos un objeto que contiene la posición de la pulsación y dependiendo de la posición sabrá qué plan y expediente porque están ordenados en un array de strings. Y automáticamente al pulsar, invocamos a la clase Asignaturas pasándole los parámetros expediente y plan para realizar la búsqueda de asignaturas del alumno que tiene en su expediente.

El resto del código de Expediente.java también contiene información relevante. En él enviamos al servidor el NIF del estudiante y recibimos los planes en los que ha estado o está matriculado el alumno y los números de expediente. Estos datos los guardamos en un ArrayList y son los que usamos para rellenar la lista.

La forma de enviar y recibir los datos al servidor es la misma forma que se ha usado con el resto de actividades, con una tarea asíncrona encargada de realizar la petición y recibir el resultado (AsyncTask).

Figura 9.24. Layout asignaturas.xml

Asignaturas

Esta actividad es muy similar a la anterior Expediente. En este caso también obtenemos los datos que necesitamos del servidor mediante una tarea asíncrona y los usamos para rellenar una lista.

Obtenemos el código de asignatura, el nombre, el curso académico de la matrícula de esa asignatura, el curso al que pertenece y la nota si es que ya se ha superado esa asignatura.

Su layout es similar al de la anterior actividad. También hay otro archivo XML llamado simple_list_asignatura.xml también similar a simple_list.xml pero con unas dimensiones distintas.

Respecto a Asignaturas.java es también muy similar a Expediente.java. En él se aprecian los mismos métodos pero con los cambios apropiados para recibir otros datos.

La única diferencia es a la hora de pulsar en uno de los ítems de la lista que no se hace nada. Pero se podría realizar igual que se realizó en la otra actividad.

Estudiante

Esta actividad muestra al usuario sus datos personales. El nombre, apellidos y fecha de nacimiento. Pueden añadirse más datos pero en este proyecto sólo hemos incluido estos datos.

Esta actividad recibe el NIF del alumno. Después se comunica con el servidor y recibe los datos personales del alumno. Si se han obtenido los datos correctamente los muestra en la pantalla y si no muestra un mensaje: No se encuentran datos que mostrar.

Para comunicarse con el servidor utiliza la clase AsyncTask vista anteriormente.

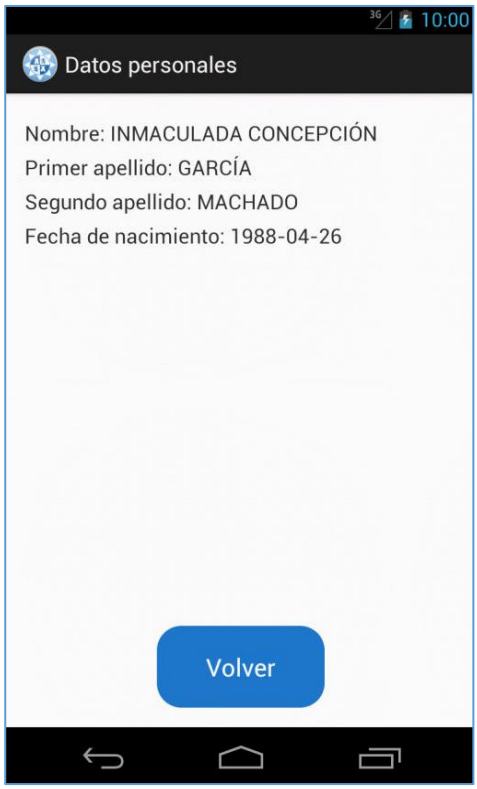

Figura 9.25. Layout estudiante.xml

Configuración

En esta actividad se va a realizar la desvinculación del dispositivo, es decir, este dispositivo no estará registrado en las bases de datos y no estará vinculado a ningún alumno. Por lo tanto, si se procede a desvincular el dispositivo, la próxima vez que intentemos entrar en la app tendremos que volver a registrar el dispositivo con la lectura del QR incluido.

En esta actividad, si procedemos a desvincular el dispositivo, nos aparecerá un mensaje de confirmación como muestran las siguientes figuras. Este AlertDialog está diseñado para obtener una respuesta positiva o negativa.

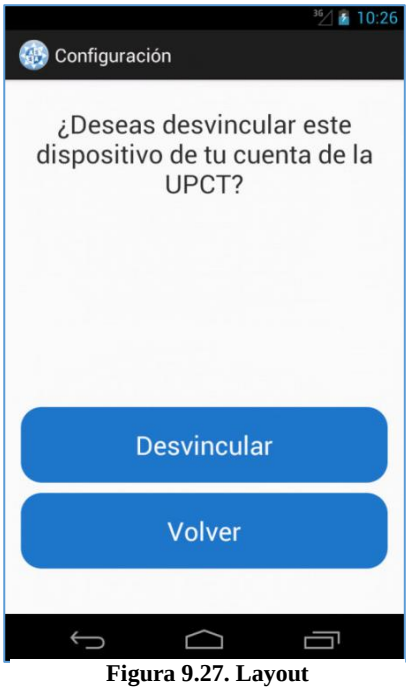

Figura 9.27. Layout configuracion.xml

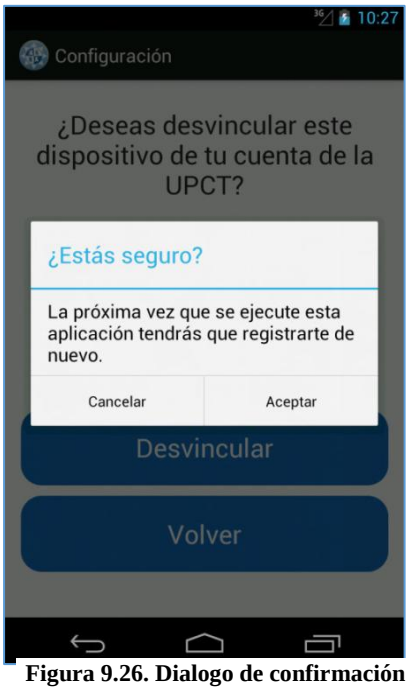

Figura 9.26. Dialogo de confirmación

Para desvincular un dispositivo de un usuario, lo primero que hay que realizar si se pulsa el botón Aceptar, es obtener el DEVICE_ID y el DEVICE_NAME como se realizó en la actividad Inicio. Una vez obtenidos estos datos, se procede a realizar una petición al servidor enviándole el DEVICE_ID, el DEVICE_NAME y el NIF del alumno. La comunicación se realiza como en las anteriores actividades con una tarea asíncrona AsyncTask que lo realiza en segundo plano.

El servidor nos enviará un parámetro y dependiendo de su valor, la desvinculación habrá sido realizada con éxito o no.

AndroidManifest.xml

En el AndroidManifest.xml se describen los componentes de la aplicación como las actividades, servicios, etc. También se declaran los permisos que la aplicación tiene para poder acceder a las partes protegidas de la API e interactuar con otras aplicaciones.

En este caso hemos necesitado tres permisos:

- Acceso a la cámara. Este permiso se necesita para poder leer el código QR.

```
<uses-permission android:name="android.permission.CAMERA" />
```

- Acceso a Internet. Para conectarnos con el servidor y realizar las descargar.

```
<uses-permission android:name="android.permission.INTERNET" />
```

- Acceso a leer el estado del teléfono. Para poder leer el IMEI, el número de serie o el nombre del dispositivo.

```
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

9.9.2 Servidor

Como se ha dicho anteriormente, para hacer esta aplicación se ha usado un servidor local con una base de datos con características parecidas a la de la UPCT. Para poder conectarnos a la base de datos y ejecutar una consulta se ha decidido utilizar un web service. A él se puede acceder pasando diversos parámetros y nos devuelve la respuesta ya sea en formato XML o JSON. En este ejemplo hemos usado JSON.

Para realizar esta parte se ha utilizado como lenguaje en la parte del servidor PHP (versión 5.4.22), como Base de Datos MySQL (versión 5.5.39) y como servidor web Apache (versión 2.4.10). Se ha utilizado XAMPP (versión 1.8.2) ya que dispone lo anterior mencionado, está disponible para varios sistemas operativos y es muy fácil instalarlo y usarlo.

Comencemos con la Base de Datos: Se ha creado una BD llamada android_login. En ella se han creado 5 tablas:

- usuarios: Con los datos de usuarios que usa la app.
- alu_expediente: Contiene los números de expediente, el plan, el NIF, el año académico y el nombre del plan.
- alu_alumno: Contiene el NIF del alumno, el nombre, el primer y segundo apellido y la fecha de nacimiento.
- alu_matricula: Contiene el año académico de matrícula, el plan, el número de expediente, el código de asignatura, el tipo, el curso en el que se imparte y el nombre de la asignatura
- Alu_acta: Contiene el plan, número de expediente, el año académico, el código de asignatura, la nota de la asignatura y la convocatoria.

En la parte web de nuestro servidor, en la carpeta htdocs de XAMPP, tenemos una carpeta llamada androidlogin. En ella se guardan los archivos que contiene la parte web. Podemos encontrar 10 archivos PHP: config.php, connectbd.php, funciones_bd.php, adduser.php, deleteuser.php, seedevice.php, getexpediente.php, getasignaturas.php y getpersonal.php.

A continuación los describiremos de uno en uno:

config.php

En este archivo se define el nombre de la base de datos, el usuario, la contraseña y la IP del servidor.

```
<?php
/**
 * Database config variables
 */
define("DB_HOST", "127.0.0.1"); //hace referencia a donde se encuentra la BD
define("DB_USER", "root");//nombre de usuario definido en la configuración de la BD.
```

```
define("DB_PASSWORD", "");//password elegido
define("DB_DATABASE", "android_login");//Nombre de la base de datos
?>
```

connectbd.php

Este archivo provee los métodos para conectarse y desconectarse a la BD. Podemos ver el código en el Anexo.

funciones_bd.php

Este archivo provee los métodos para conectarse y desconectarse a la BD y los métodos para realizar las consultas.

Vamos a poner un ejemplo de función que se encuentra en funciones_bd.php, la función que verifica si el dispositivo ya está registrado o no:

```
public function isdeviceexist($device_id , $device) {

        $result = mysql_query("SELECT device_id from usuarios WHERE device = 
'$device' and device_id='$device_id'");

        $num_rows = mysql_num_rows($result); //número de filas retornadas

        if ($num_rows > 0) {
            // el dispositivo ya está registrado
            return true;
        } else {
            // no está registrado el dispositivo
            return false;
        }
               mysql_free_result($result);
    }
```

Como podemos ver, la función recibe el `$device_id y $device` y la query se guarda en `$result`. Obtenemos el número de filas retornadas y si es mayor que 1 es que hemos obtenido un dispositivo con ese `$device_id y $device`, es decir, que si se encuentra en la BD. Se retorna true o false dependiendo del resultado.

En este archivo funciones_bd.php encontramos todos los métodos como la obtención de los datos personales del usuario, los expedientes y planes del usuario, las asignaturas, etc. Se puede encontrar el código completo en el Anexo.

adduser.php

Permite realizar alta de dispositivos. En la app, en la clase Registro, tenemos el NIF del usuario, el DEVICE_ID, DEVICE_NAME, el nombre y el HASH que hemos recibido del QR. Hacemos un POST enviándole estos datos a adduser.php. Y con estos datos accedemos a la función adduser() del archivo funciones_bd.php y se realiza el alta del dispositivo. Devolvemos a la clase que ha hecho la petición el resultado en formato JSON. En las siguientes líneas podemos ver adduser.php:

```
<?php
```

```
$nif = $_POST['nif'];
$user = $_POST['user'];
$device_id = $_POST['device_id'];
$device = $_POST['device'];
$hash = $_POST['hash'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

if($db->adduser($nif,$user,$device_id,$device,$hash)){
	$resultado[]=array("regstatus"=>"1");
}else{
	$resultado[]=array("regstatus"=>"0");
}
echo json_encode($resultado);
?>
```

Y la función adduser() que está en funciones_bd.php es:

```
/**
 * agregar nuevo dispositivo de un usuario
 */
public function adduser($nif,$user,$device_id,$device,$hash) {

    $result = mysql_query("INSERT INTO usuarios (username, device_id, device, DNIALU,
HASH) VALUES ('$user', '$device_id','$device','$nif','$hash')");
     // check for successful store
    if ($result) {
         return true;
    } else {
         return false;
    }
}
```

deleteuser.php

En el caso de tener que dar de baja un dispositivo. Por ejemplo, se usa en la aplicación móvil en Configuracion.java. Es muy similar a dar de alta un usuario pero en este caso en vez de insertar una fila en la tabla lo que se hace es borrar una fila de la tabla que coincida con el DEVICE_ID y el DEVICE_NAME. Si ha sido borrada la fila devolverá `"regstatus"=>"1"` y si no `"regstatus"=>"0"`.

```
<?php
$nif = $_POST['nif'];
$device_id = $_POST['device_id'];
$device = $_POST['device'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

if($db->deleuser($nif,$device_id,$device)){
	$resultado[]=array("regstatus"=>"1");
}else{
	$resultado[]=array("regstatus"=>"0");
}

echo json_encode($resultado);
?>
```

Y el método que utiliza está en funciones_bd.php y es esta función:

```
/**
 * eliminar dispositivo de un usuario
 */
public function deleuser($nif,$device_id,$device) {

	$result = mysql_query("DELETE FROM `usuarios` WHERE DNIALU='$nif' AND
device_id ='$device_id' and device ='$device'");
// check for successful delete

	if ($result) {
		return true;
	} else {
		return false;
	}
}
```

seedevice.php

En este archivo lo que se hace es comprobar si el dispositivo ya está en la base de datos o no. Para ello hemos recibido el DEVICE_ID y DEVICE_NAME con un POST y lo comprobamos en la función isdeviceexist() del archivo funciones_BD.php. En el caso de que sí estuviera registrado, obtenemos el NIF y el nombre del usuario en otra función llamada datospersonal() de funciones_BD() y los enviamos en la respuesta. El código de seedevice() es:

```
<?php

/*Busca el dispositivo en la base de datos*/
$devi_id = $_POST['device_id'];
$devi = $_POST['device'];

require_once 'funciones_bd.php';
$db = new funciones_BD();
	/*Comprobamos que existe este dispositivo en la bd*/
	if($db->isdeviceexist($devi_id, $devi) ){

		$resultado[]=array("devstatus"=>"1");
		/* Obtenemos el nombre del alumno y su NIF */
		$result=($db->datospersonal($devi_id , $devi));

		$nom = $result[0];
		$nif = $result[1];

		$resultado[]=$nom;
		$resultado[]=$nif;

	}else{
		$resultado[]=array("devstatus"=>"0");
	}
echo json_encode($resultado);
?>
```

Y el código del método isdeviceexist() lo hemos visto anteriormente y el del método datospersonal() es:

```
/* Función para obtener el nombre y el NIF del usuario de un dispositivo*/
public function datospersonal($device_id , $device) {
	$resu = mysql_query("SELECT a.NOMALU AS NOM, a.DNIALU AS DNI
				FROM alu_alumne a, usuarios u
				WHERE device = '$device'
```

```
                        AND device_id='$device_id'
                        AND u.DNIALU=a.DNIALU");

        while ($fila = mysql_fetch_array($resu)) {

                $tempo1 =$fila["NOM"];
                $tempo = utf8_encode($tempo1);
                $result []= array("NOMALU"=>"$tempo");

                $aux1 =$fila["DNI"];
                $aux = utf8_encode($aux1);
                $result []= array("NIF"=>"$aux");
        }
        return ($result);
        mysql_free_result($resu);
  }
```

getexpediente.php

En este archivo recibimos el POST con el NIF del alumno y obtiene los planes y los números de expediente del alumno en esos planes.

Primero comprueba que existe el alumno con la función isuserexist() de funciones_BD.php. En el caso de que sí exista obtiene los planes y expedientes de la función verexpedientes() y devolvemos el resultado en JSON [{“NUMORD”:$expe1}, {“PLA_CODALF”:$plan1}, {“TITU”:$titu1}, {“NUMORD”:$expe2} , {…}]. Y el código es:

```
<?php
/*Buscamos la información personal del estudiante*/
$dnialu = $_POST['dnialu'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

        if($db->isuserexist($dnialu)){
                $expe=($db->verexpedientes($dnialu));
                if($expe!=null){
                        $cuenta = count($expe);
                        for ($i = 0; $i< $cuenta;$i++){
                                $j = $i % 3;
                                switch($j){
                                        case 0:
                                                $num = $expe[$i];
                                                $resultado[]=$num;
                                                break;
                                        case 1:
                                                $pla = $expe[$i];
                                                $resultado[]=$pla;
                                                break;
                                        case 2:
                                                $titu = $expe[$i];
                                                $resultado[]=$titu;
                                                break;
                                }
                        }
                }
        }else{
                $resultado[]="";
        }
echo json_encode($resultado);
 ?>
```

El código de los métodos isuserexist() y verexpedientes() podemos encontrarlos más adelante.

getasignaturas.php

Este archivo es muy similar a getexpedientes.php. En él lo que se hace es buscar las asignaturas de un plan y un expediente. Para ello tenemos el método verasignaturas() de funciones_BD.php que devuelve el código de la asignatura, el año académico en el que se matriculó, el nombre de la asignatura, el curso y la nota en esa asignatura. En el caso de que no esté aún aprobada aparecerá en blanco este campo. El código podemos encontrarlo más adelante en el código fuente.

getpersonal.php

Con este archivo obtenemos con un POST los datos personales del estudiante enviándole el NIF. Datos como el nombre, los apellidos y la fecha de nacimiento. Es muy similar a los archivos anteriores pero en este caso obtiene los datos de la función datosestudiante() del fichero funciones_BD.php. El código podemos encontrarlo en más adelante en el código fuente de la aplicación.

9.10 Explicación del código del caso 3

Con esta aplicación la comunidad universitaria podrá pagar en los establecimientos universitarios como las cantinas o las copisterías sólo usando su TUI, pero no como tarjeta financiera sino como una tarjeta identificativa para la cuenta virtual de la UPCT. Se ha creado una aplicación móvil la cual se conectará con la base de datos de las cuentas virtuales de la UPCT haciendo las modificaciones necesarias en ella. Estas cuentas virtuales personales son prepago y son muy cómodas y fáciles de usar y en este caso sólo se necesitará disponer de la TUI.

Es una aplicación que permite cobrar, devolver o mirar el historial de transacciones de los usuarios a través del código QR de cada TUI.

Deberá estar instalada en un dispositivo autorizado y registrado por la UPCT y, además, el personal que la utilice deberá también estar registrado y en cada sesión autenticarse.

Funcionamiento:

Primero para poder usar la aplicación el empleado autorizado deberá autenticarse introduciendo su número de identificación y una contraseña. Cuando se vaya a realizar una transacción, el empleado debe introducir la cantidad a cobrar o devolver. Después la aplicación pedirá realizar la lectura del código QR de la TUI del cliente. Si es correcta, pedirá la contraseña de la cuenta virtual la cual el cliente debe introducir y se enviarán los datos al servidor. El servidor comprobará si es posible la transacción y, si todo es correcto, se realizará la transacción y se enviará un mensaje de confirmación. En el caso

de que no hubiese saldo suficiente o la contraseña introducida fuera incorrecta se mostraría un error. El esquema de funcionamiento se muestra en la figura 9.28.

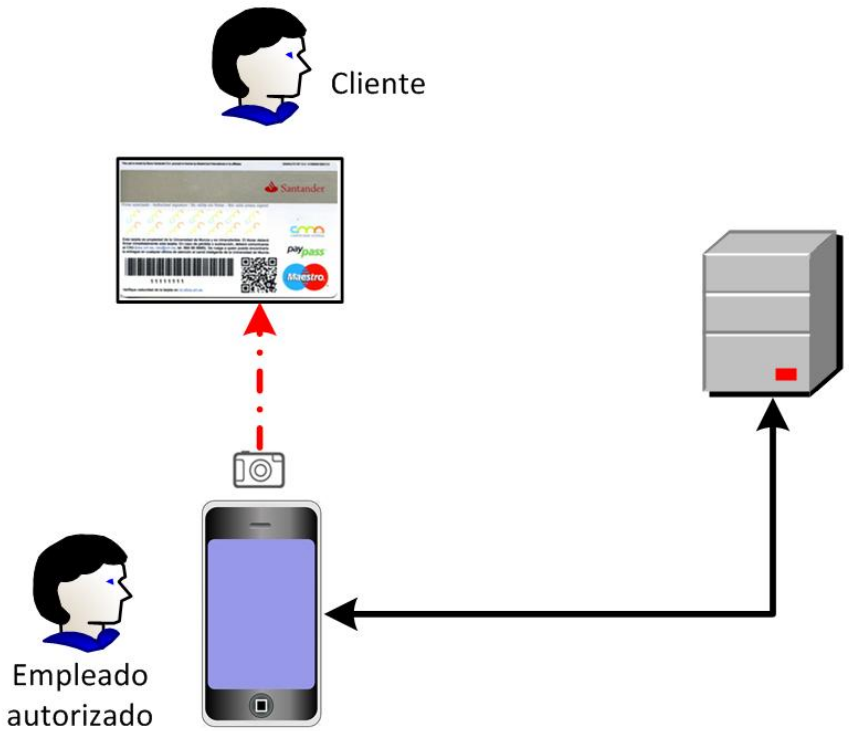

Figura 9.28 Caso3 – Esquema del funcionamiento

En el servidor se registrará cada movimiento y el saldo de los usuarios será automáticamente actualizado. En la base de datos se guardarán para cada movimiento todos los datos como la fecha, la hora, el lugar, el dispositivo móvil utilizado, el empleado, el cliente, el importe y el tipo de transacción para que no se produzcan incidencias ni errores o para poder comprobarlos.

En el caso de que un cliente quiera mirar su historial de transacciones, el sistema funciona de modo similar. La aplicación hace la lectura del código QR del cliente y el cliente debe introducir su contraseña. Si todo es correcto se mostrará una lista con los movimientos realizados en esa cuenta virtual indicando la fecha, hora, lugar, importe y tipo de transacción (compra o devolución) de cada movimiento.

En la siguiente figura se muestra un diagrama de flujo del funcionamiento de la aplicación:

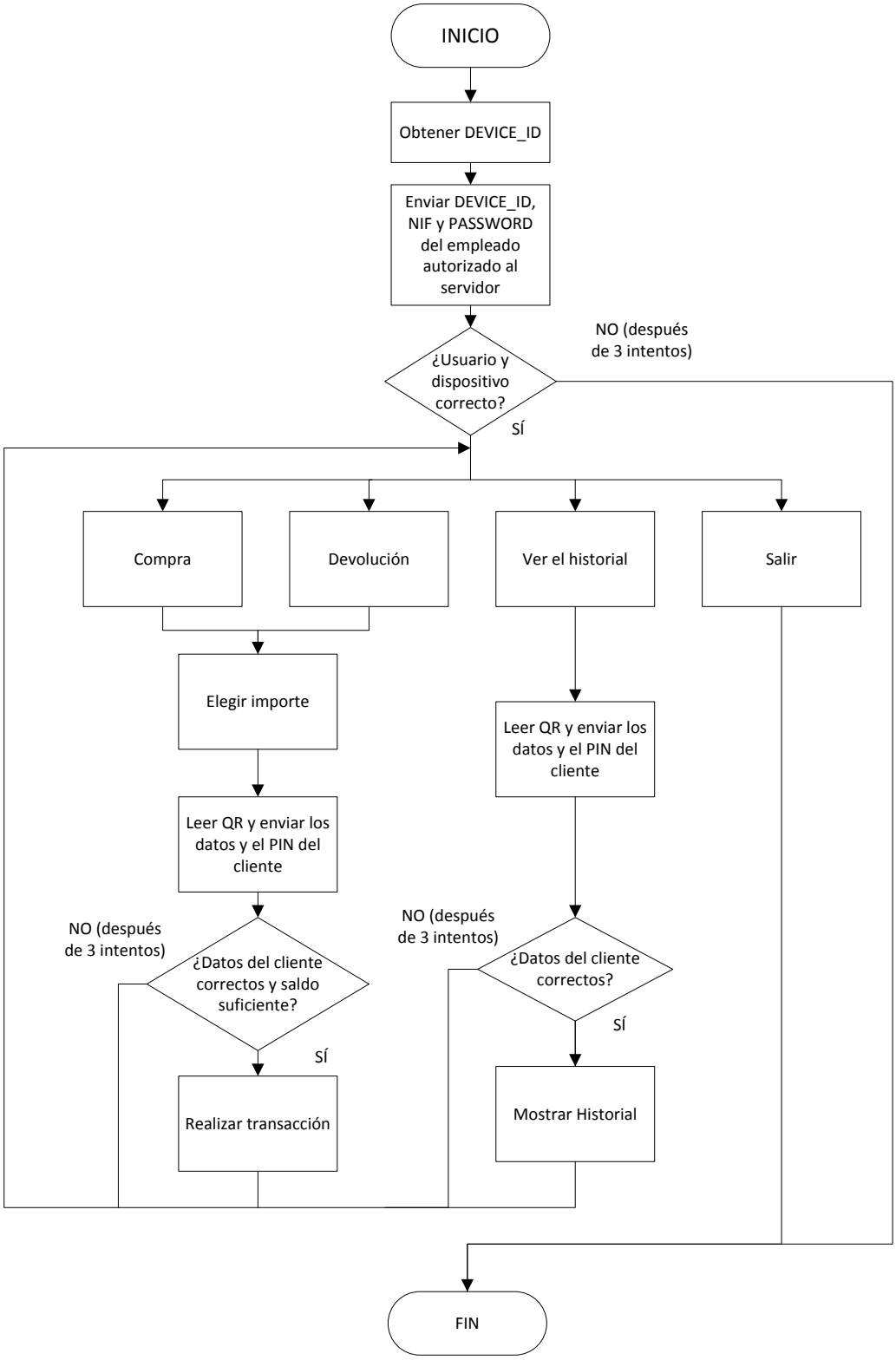

Figura 9.29 Diagrama de flujo del caso 3

En ella se puede apreciar un inicio donde se trata de autenticar al empleado autorizado que será el responsable de las operaciones hechas en su sesión.

Una vez autenticado, la aplicación le lleva al menú principal donde, en este caso, sólo puede hacer 4 acciones: compra, devolución, ver el historial o salir de la aplicación.

En el caso de la compra o de la devolución se realizan los mismos pasos simplemente cambia el signo del importe. En la aplicación hay más diferencias entre compra y devolución, como veremos más adelante, pero para simplificar en el diagrama de flujo se han puesto las mismas acciones para los dos.

En la aplicación siempre que se requiere de una contraseña se da la oportunidad de equivocarse hasta 2 veces. Al equivocarse 3 veces introduciendo una contraseña la aplicación se cerrará o le llevará al empleado autorizado al menú principal en el caso de que sea el cliente el que se ha equivocado. También hay que destacar que en ningún momento se envían contraseñas sin una previa encriptación para mayor seguridad.

Se va a empezar explicando la aplicación móvil y después se verá la parte del servidor local que como se verá es muy similar al servidor del caso 2.

9.10.1 Aplicación móvil

Para comenzar vamos a dividir la aplicación en dos partes. Primero se va a comenzar explicando la parte de la autenticación del empleado autorizado y segundo la parte de las transacciones con el cliente.

Esta primera parte de la autenticación del empleado es muy sencilla y ya se ha visto en el primer caso un ejemplo similar.

Autorización

En esta actividad se declara y se define la IP del servidor. También se muestran los campos donde el empleado autorizado introducirá su NIF y su contraseña para iniciar sesión como puede observarse en la figura 9.30. Además, se averigua cual es el ID del dispositivo. Estos tres últimos datos son enviados al servidor (la contraseña encriptada) y el servidor responderá si ese usuario es el correcto y si está autorizado a trabajar con ese dispositivo.

En el caso de obtener una respuesta positiva, el servidor nos responderá también con el centro o el lugar en el que ese dispositivo está destinado a trabajar y se continuará en la siguiente actividad siendo la del menú principal.

En el caso de que la respuesta sea negativa, se da la oportunidad de intentarlo 2 veces más y al tercer fallo la aplicación se cerrará mostrando un mensaje.

MainActivity

Aquí se muestra el menú principal de la aplicación. En ella se puede ver un menú realizado con una herramienta de Android llamada GridView. Se han usado imágenes para representar cada opción como puede apreciarse en la figura 9.31.

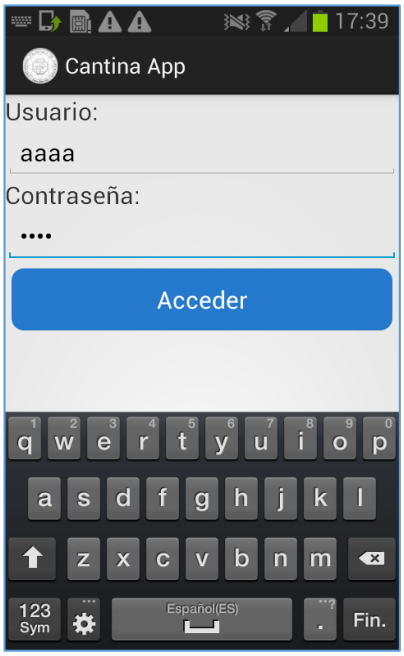

Figura 9.30 Layout autorizacion.xml

Figura 9.31 Layout activity_main.xml

Además se mostrará el NIF o el nombre del empleado autorizado de la sesión activa y el lugar que hemos recibido del servidor asociado a ese dispositivo.

Dependiendo en que zona del GridView pulsemos se realizarán unas acciones u otras:

- Compra: Se avanza hacia la actividad *Compra* y se le envía a la actividad el valor de la variable **user**, la cual contiene el NIF del empleado autorizado.
- Devolución: Se avanza hacia la actividad *Devolucion* y se le envía el valor de la variable **user**.
- Historial: Se avanza a la actividad *LectorQR* donde se leerá el código QR de la TUI del cliente para que después de introducir su PIN podemos observar las transacciones realizadas con anterioridad. Se le envía el valor de la variable **user**, **importe** y **tipo_oper**. La variable importe estará vacía pero a la variable tipo_oper se le da el valor de tipo_oper = 3 indicando este valor que se trata de una visualización del historial de transacciones.
- Salir: Como su nombre indica se cierra la aplicación.

En esta actividad además del menú principal, internamente se guarda en un fichero el valor de la variable espacio. Esta variable es la que guarda el lugar asociado a ese dispositivo y persona autorizada. Este tipo de código donde creamos, escribimos y guardamos un fichero interno para no perder el valor de una variable ya se ha explicado anteriormente en el caso 1.

Compra

Esta actividad es la primera que se verá para realizar una compra, es decir, descontar de la cuenta virtual del cliente el importe indicado por el empleado autorizado. En ella sólo se especifica el saldo que se va a cobrar y al pulsar el botón a continuación se avanza

hacia la actividad *LectorQR* para leer la TUI enviando el valor de las variables **user, importe** y **tipo_oper**.

En esta actividad se define la variable tipo_oper = 1 para indicar que se está realizando una compra.

En la figura 9.32 se puede observar el layout de esta actividad.

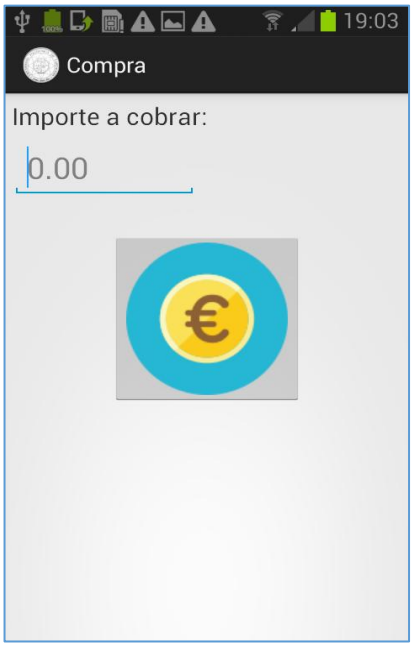

Figura 9.32 Layout compra.xml

Figura 9.33 Layout devolucion.xml

Devolución

Para realizar una devolución los pasos a seguir son muy similares a los de la compra. El empleado autorizado indicará que cantidad es la de reembolso y al pulsar el botón se avanza hacia la actividad *LectorQR* para leer la TUI del cliente enviando el valor de las variables **user, importe** y **tipo_oper**.

En este caso la variable del tipo de operación será tipo_oper = 2 indicando que se está realizando una devolución.

En la figura 9.33 se puede observar el layout de esta actividad.

LectorQR

Esta actividad es la encargada de escanear el QR de la TUI y si la lectura es correcta invocamos a la actividad Tarjetapin que la veremos a continuación. Se le enviará el valor de las variables recibidas **user, importe** y **tipo_oper** y también se le enviará lo leído en el QR.

En la figura 9.34 se encuentra su layout el cual muestra el botón para iniciar el lector. Esta actividad es similar a Lector.java del caso 1 y a MainActivity del caso 2.

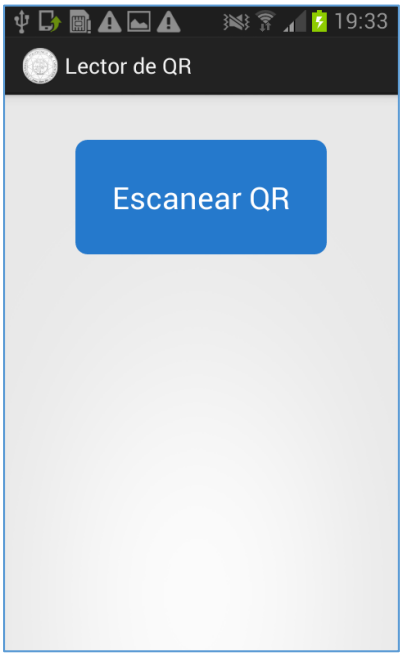

Figura 9.34 Layout lectorqr.xml

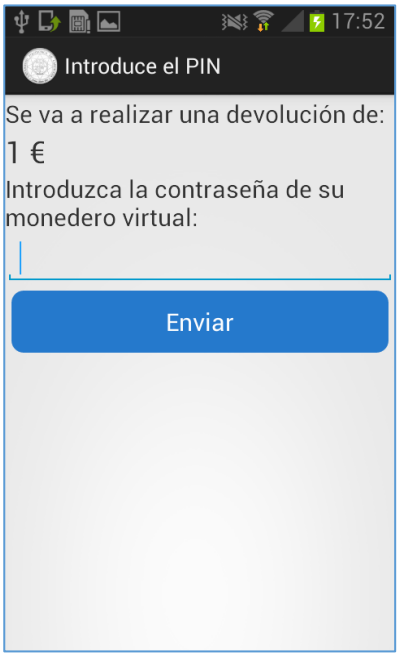

Figura 9.35 Layout tarjetapin.xml

Tarjetapin

Una vez leído el código QR de la TUI se le pide al usuario de la tarjeta el PIN para poder acceder a su cuenta virtual.

En este layout dependiendo del tipo de operación que se vaya a realizar se mostrará un mensaje u otro:

- Compra: Se mostrará un mensaje indicando que se va a realizar un cargo a su cuenta virtual de X cantidad.
- Devolución: Se mostrará un mensaje indicando que se va a realizar una devolución a su cuenta virtual de X cantidad (Figura 9.35).
- Historial: Se mostrará un mensaje indicando que se va a extraer el historial de su cuenta para su visualización.

También se muestra un teclado numérico para que el cliente introduzca su PIN para que después de encriptarlo se envíe al servidor.

El servidor nos responderá si el PIN es el correcto o no. En el caso de ser el correcto, dependiendo del valor de tipo_operacion se avanzará hasta una actividad u otra:

- Compra o devolución → Pago
- Ver historial → Historial

En caso de ser incorrecto se podrá intentar 2 veces más y si no se consigue la aplicación volvería al menú principal indicando que el PIN no es el correcto.

El envío al servidor de los datos es similar al caso 2 de este proyecto.

Pago

Para realizar una compra o una devolución se utiliza esta misma actividad. A partir de aquí la diferencia se realiza en el servidor sumando o restando el importe enviado. La variable que indicará la operación es tipo_operación que como hemos visto en cada caso tiene un valor distinto.

Esta actividad comienza enviando al servidor las variables **id_operacion, cod_tarjeta, nif_autorizado, tipo_operacion, importe, cod_concepto** y **terminal**. La variable **cod_tarjeta** es la cadena de caracteres leída por el *LectorQR*. **nif_autorizado** es el NIF del empleado autorizado, la variable **user** usada en las anteriores actividades. **tipo_operacion** e **importe** se reciben de la anterior actividad también y son conocidas. **cod_concepto** es una variable que en este proyecto está vacía siempre pero que se ha creado para futuros usos en el caso de ser llevada a cabo. Y **terminal** es el código de identificación del dispositivo obtenido en la actividad *Autorización.* Todos estos datos se envían al servidor, el servidor realiza la transacción y guarda todos los valores en su registro. Este registro es el que se visualiza cuando queremos ver el historial de un usuario. Más adelante se explicará.

Una vez enviados los datos, el servidor responde con una transacción exitosa o fallida dependiendo del saldo disponible del usuario. En las figuras 9.36 y 9.37 se pueden ver los distintos layouts para cada caso.

Figura 9.36 Layout pago.xml - Éxito

Figura 9.37 Layout pago.xml - Fallida

Como se puede observar, el resultado de la transacción se encuentra en un WebView y debajo hay dos botones, uno para acceder rápidamente a una nueva compra y el otro para acceder al menú principal.

Historial

Aquí se muestra una lista donde se muestran todas las transacciones realizadas en la cuenta virtual del usuario mostrando el tipo de operación que se realizó, el importe, la fecha y la hora a la que se realizó. Actividad muy similar a la actividad ListaInvitados del caso práctico 1.

Al servidor se le envían el código de la tarjeta y el identificador del dispositivo y nos responde con todas las transacciones realizadas con la TUI. El código fuente se encuentra más adelante. En la figura 9.38 podemos ver cómo sería la lista resultante.

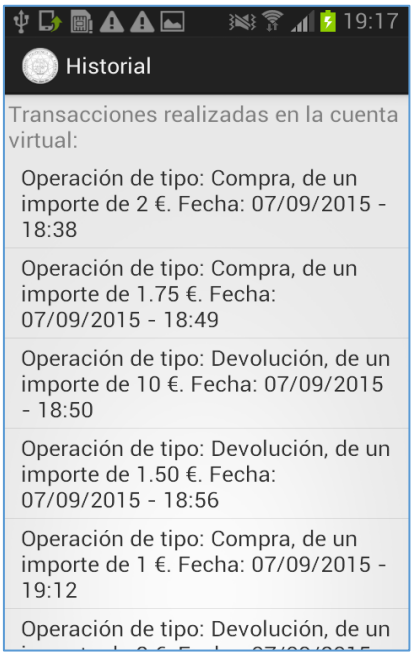

Figura 9.38 Layout historial.xml

9.10.2 Servidor

Para hacer esta aplicación se ha usado un servidor local con una base de datos nueva. Para poder conectarnos a la base de datos y ejecutar una consulta se ha decidido utilizar un web service. A él se puede acceder pasando diversos parámetros y nos devuelve la respuesta ya sea en formato XML o JSON. En este ejemplo hemos usado JSON.

Para realizar esta parte, al igual que en el caso práctico 2, se ha utilizado como lenguaje en la parte del servidor PHP (versión 5.4.22), como Base de Datos MySQL (versión 5.5.39) y como servidor web Apache (versión 2.4.10). Se ha utilizado XAMPP (versión 1.8.2) ya que dispone lo anterior mencionado, está disponible para varios sistemas operativos y es muy fácil instalarlo y usarlo.

Comencemos con la Base de Datos: Se ha creado una BD llamada monedero_virtual. En ella se han creado 5 tablas:

- espacios: Contiene los códigos de los lugares o centros y su descripción.

- historial: En esta tabla se registran todas las operaciones que se realizan. Contiene los códigos identificativos de la operación, el código de la tarjeta, el NIF del empleado autorizado que ha realizado la operación, el tipo de operación, el importe, el código de concepto y el terminal que ha realizado la operación.
- monederos: Contiene los códigos de la tarjeta, el PIN encriptado y el saldo que dispone cada cuenta.
- nif_auto: Contiene el código de identidad de cada terminal, los nif_autorizados que pueden usar ese dispositivo y el PIN de cada usuario autorizado a usar la aplicación.
- tipo_operaciones: Contiene el código de cada tipo de operación y su descripción.

En la parte web de nuestro servidor, en la carpeta htdocs de XAMPP, tenemos una carpeta llamada monedero_virtual. En ella se guardan los archivos que contiene la parte web. Podemos encontrar 7 archivos PHP: autenticar.php, config.php, connectbd.php, funciones_bd.php, historial.php, pagar.php y validarpin.php.

A continuación los describiremos de uno en uno:

config.php

En este archivo se define el nombre de la base de datos, el usuario, la contraseña y la IP del servidor.

```
<?php
/**
 * Database config variables
 */
define("DB_HOST", "127.0.0.1");
define("DB_USER", "root");
define("DB_PASSWORD", "");
define("DB_DATABASE", "monedero_virtual");
?>
```

connectbd.php

Este archivo provee los métodos para conectarse y desconectarse a la BD. Podemos ver el código en el anexo.

funciones_bd.php

Este archivo provee los métodos para conectarse y desconectarse a la BD y los métodos para realizar las consultas.

Vamos a poner un ejemplo de función que se encuentra en funciones_bd.php, la función que verifica si el usuario/empleado autorizado es válido para ese dispositivo o no:

```
public function isuserexist($nif,$pass,$deviceid) {

        $select = mysql_query("SELECT e.desc_espacio AS espacio from nif_auto n, 
espacios e WHERE n.nif_autorizados = '$nif' AND n.pin = '$pass' AND n.terminal = 
'$deviceid' AND n.espacio=e.cod_espacio");

        $num_rows = mysql_num_rows($select); //numero de filas retornadas
```

```
        if ($num_rows > 0) {
            // el usuario es correcto
                    while ($fila = mysql_fetch_array($select)) {
                           $tempo = $fila["espacio"];
                           $tempo2 = utf8_encode($tempo);
                           $result [] = array("espacio"=>"$tempo2");
                    }
        } else {
            // no existe
            return false;
        }
               return ($result);
               mysql_free_result($select);
    }
```

Como podemos ver, la función recibe el `$nif, $pass y $deviceid` y la query se guarda en `$select`. Obtenemos el número de filas retornadas y si es mayor que 1 es que hemos obtenido un dispositivo con ese `$nif` con esa contraseña `$pass` y para ese `$deviceid`, es decir, que sí se encuentra en la BD. Se retorna la descripción del espacio/lugar/centro si él se encuentra en la base de datos o false si no se ha encontrado.

En este archivo funciones_bd.php encontramos todos los métodos como la comprobación del PIN de una cuenta virtual, escribir en el historial, mirar el historial de una cuenta virtual, hacer un cobro y hacer una devolución. Se puede encontrar el código completo en el anexo.

autenticar.php

Permite comprobar y autenticar a los usuarios/empleados autorizados. En la app, en la clase Autorizacion, tenemos el NIF del usuario, la contraseña y el identificador del dispositivo. Se hace un POST enviándole estos datos a autenticar.php Y con estos datos accedemos a la función `isuserexist()` del archivo funciones_bd.php que se ha visto en el punto anterior y se realiza la comprobación de la contraseña y si está autorizado a usar ese dispositivo. Devolvemos a la clase que ha hecho la petición el valor del centro/lugar/espacio en el que está autorizado ese dispositivo y usuario en formato JSON para después la app mostrarlo. En las siguientes líneas podemos ver autenticar.php:

```
<?php

$nif = $_POST['nif'];
$pass = $_POST['pass'];
$deviceid = $_POST['deviceid'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

        $res=($db->isuserexist($nif,$pass,$deviceid));

        if($res!=null){

                $resultado[]=array("regstatus"=>"1");

                $centro = $res[0];
                $resultado[] = $centro;
```

```
        }else{
                $resultado[]=array("regstatus"=>"0");
        }

echo json_encode($resultado);

?>
```

validarpin.php

Este archivo lo que hace es validar el PIN de un usuario de una cuenta virtual. La cuenta virtual de un usuario estará asociada a su TUI y de este modo, con el código de la tarjeta y el PIN (llamado `$hash` en el ejemplo) podemos comprobar su validez y seguir con las transacciones. El código de este archivo es:

```
<?php

$codtar = $_POST['codtarjeta'];
$hash = $_POST['hash'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

        $res=($db->ispincorrect($codtar,$hash));

        if($res!=null){

                $resultado[]=array("regstatus"=>"1");

        }else{
                $resultado[]=array("regstatus"=>"0");
        }

echo json_encode($resultado);

?>
```

En funciones_bd.php podemos encontrarnos una función que comprueba si existe ese usuario con ese PIN encriptado y devuelve un resultado si sí existe en la base de datos y devuelve un false si no existe.

pagar.php

Este archivo es más complejo que los otros anteriores. En él primero se distingue entre si se va a pagar o si se va a devolver, para realizar una suma o una resta.

En el caso de que sea pagar, se ejecutará la función cobrar de funciones_bd.php la cual primero obtiene el saldo del que dispone y comprueba si es superior al importe que se desea cobrar. En el caso afirmativo, resta el importe al saldo y modifica el saldo. Y por último escribe en el registro. Y en el caso negativo, devuelve false y envía un false a la aplicación indicando que no se ha podido realizar la operación.

```
<?php

$id_operacion = $_POST['id_operacion'];
```

```
$cod_tarjeta = $_POST['cod_tarjeta'];
$nif_autorizado = $_POST['nif_autorizado'];
$tipo_operacion = $_POST['tipo_operacion'];
$importe = $_POST['importe'];
$cod_concepto = $_POST['cod_concepto'];
$terminal = $_POST['terminal'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

	if($tipo_operacion == '1'){		//Pagar

		if($db->cobrar($id_operacion, $cod_tarjeta, $importe)){

			if($db->write_history($id_operacion, 		$cod_tarjeta,
$nif_autorizado, $tipo_operacion, $importe, $cod_concepto, $terminal))
			{
				$resultado[]=array("regstatus"=>"1");
			}else{
				$resultado[]=array("regstatus"=>"0");
			}
		}else{
			$resultado[]=array("regstatus"=>"0");
		}
	}else{			//Devolución

		if($db->devolver($id_operacion, $cod_tarjeta, $importe)){

			if($db->write_history($id_operacion, 		$cod_tarjeta,
$nif_autorizado, $tipo_operacion, $importe, $cod_concepto, $terminal))
			{
				$resultado[]=array("regstatus"=>"1");
			}else{
				$resultado[]=array("regstatus"=>"0");
			}
		}else{
			$resultado[]=array("regstatus"=>"0");
		}
	}
echo json_encode($resultado);

?>
```

historial.php

En este archivo simplemente se hace una consulta a la base de datos y esta consulta nos devuelve los parámetros que necesitamos para mostrarlos por la pantalla de la aplicación.

```
<?php

/*Buscamos el historial de la tarjeta y con ese terminal móvil*/
$codtar = $_POST['codtar'];
$terminal = $_POST['terminal'];
/*
$codtar = '5901000421958215';
$nifaut = 'aaaa';*/
require_once 'funciones_bd.php';
$db = new funciones_BD();

	$hist=($db->verhistorial($codtar, $terminal));

	if($hist!=null){
```

```
		$cuenta = count($hist);
		for ($i = 0; $i< $cuenta;$i++){
			$j = $i % 8;
			switch($j){
				case 0:
					$id_oper = $hist[$i];
					$resultado[]=$id_oper;
					break;
				case 1:
					$cod_tarjeta = $hist[$i];
					$resultado[]=$cod_tarjeta;
					break;
				case 2:
					$nif_autorizado = $hist[$i];
					$resultado[]=$nif_autorizado;
					break;
				case 3:
					$tipo_operacion = $hist[$i];
					$resultado[]=$tipo_operacion;
					break;
				case 4:
					$importe = $hist[$i];
					$resultado[]=$importe;
					break;
				case 5:
					$cod_concepto = $hist[$i];
					$resultado[]=$cod_concepto;
					break;
				case 6:
					$terminal = $hist[$i];
					$resultado[]=$terminal;
					break;
				case 7:
					$pin = $hist[$i];
					$resultado[]=$pin;
					break;
			}
		}
	}

echo json_encode($resultado);
 ?>
```

Todos los códigos fuente podemos encontrarlos completos en el apartado de anexos del caso 3.

9.11 Código fuente

9.11.1 Caso 1) Control de asistencia a eventos basada en QR

9.11.1.1 MainActivity.java

```
/**
 * Actividad principal de la aplicación.
 *
 * @author Inmaculada C. García Machado
 * @version 1.0
 */
package com.qrtest;

import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import android.os.Bundle;
import android.os.Environment;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.Drawable;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.view.View.OnClickListener;

public class MainActivity extends Activity {
   Button boton;
   TextView text;
   String opcion = "";
   RssParserDom dom;
   String FILENAME = "hola_archivo";
   /**
    * Método que se ejecuta al iniciar.
    * Inserta una imagen de fondo si la hay.
    * Se declaran los controles como el botón y un campo de texto que pondrá online u
offline
    * @param savedInstanceState Estado de la actividad
    */
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);

       //Fondo de pantalla. Sólo actividad principal
       File file = new File(
Environment.getExternalStorageDirectory().getAbsolutePath()+
"/Android/data/com.qrtest/files/mnt/sdcard", "fondo.png");
       if (file.exists()) {
          Drawable mDrawable = Drawable.createFromPath(file.getAbsolutePath());
          ImageView fondopantalla =
(ImageView)findViewById(R.id.BackgroundImageView);
          fondopantalla.setBackgroundDrawable(mDrawable);
       }

       //Declaración de controles
       boton = (Button)findViewById(R.id.boton);
       text = (TextView)findViewById(R.id.text);

       //Si es la primera vez que se ejecuta la app, elegimos un modo por defecto
       if (opcion.equals("")){
          opcion = "online";
          text.setText(opcion);
          text.setTextColor(Color.parseColor("#00CC00"));
       }

       //Al pulsar escanear. Depende de la opción luego será online u offline
       boton.setOnClickListener(new OnClickListener(){
          public void onClick(View v) {
              Intent intent = new Intent (MainActivity.this, Lector.class);
              intent.putExtra("modo", opcion);
              startActivity(intent);
          }
```

```
        });
    }

    /**
     * Deshabilitamos el botón de atrás con este método porque no hace nada si se
pulsa el botón de atrás
     */
    @Override
    public void onBackPressed() {
        return;
    }

    /**
     * Al salir de la actividad, creamos un fichero en el que guardamos el estado
(modo)
     * Esta forma de guardar el valor de un parámetro es el más fiable en Android.
     */
    @Override
    protected void onPause() {
        //La primera vez que se ejecuta será online.
        if (opcion.equals("")){
            opcion = "online";
            text.setText(opcion);
            text.setTextColor(Color.parseColor("#00CC00"));
        }
        FileOutputStream fos;
        try {
            fos = openFileOutput(FILENAME, Context.MODE_WORLD_WRITEABLE);
            fos.write(opcion.getBytes());
            fos.close();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        super.onPause();
    }
    /**
     * Cuando regresemos a esta actividad, recuperamos su estado (el modo anterior que
había, online u offline)
     */
    @Override
    protected void onResume() {
        FileInputStream fis;
        try {
            fis = openFileInput(FILENAME);
            int c;
            String temp="";
            while( (c = fis.read()) != -1){
                temp = temp + Character.toString((char)c);
            }
            fis.close();
            opcion = temp;
            text.setText(opcion);
            if (opcion.equals("online")){
                text.setTextColor(Color.parseColor("#00CC00"));
            }else if (opcion.equals("offline")){
                text.setTextColor(Color.parseColor("#FF0000"));
            }
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }
        super.onResume();
    }

    /**
     * Menú. Con este método indicamos que menú es el de esta pantalla
     * @param menu
     */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    /**
     * Menú. Con este método indicamos que se debe hacer cuando se pulsa un
     * elemento del menú
     * @param item Elemento del menú
     * @return Si existe esa opción devuelve true
     */
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.AcercaDe:
                Intent intent4 = new Intent (MainActivity.this, AcercaDe.class);
                startActivity(intent4);
                return true;
            case R.id.Online:
                 opcion = "online";
                 text.setText(opcion);
                 text.setTextColor(Color.parseColor("#00CC00"));          //color verde
                 return true;
            case R.id.Offline:
                opcion = "offline";
                text.setText(opcion);
                text.setTextColor(Color.parseColor("#FF0000"));          //color rojo
                return true;
           case R.id.Descargarfichero:
                Intent intent = new Intent (MainActivity.this, Registro.class);
                //Opción con contraseña
                intent.putExtra("funcion", "fichero");
                startActivity(intent);
                return true;
           case R.id.Establecerfondo:
                Intent intent5 = new Intent (MainActivity.this, Registro.class);
                //Opción con contraseña
                 intent5.putExtra("funcion", "fondo");
                 startActivity(intent5);
                 return true;
            case R.id.MostrarList:
                 Intent intent6 = new Intent (MainActivity.this, ListaInvitados.class);
                 startActivity(intent6);
                 return true;
            case R.id.Salir:
                 Intent intent2 = new Intent(Intent.ACTION_MAIN);
                 intent2.addCategory(Intent.CATEGORY_HOME);
                 intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                 startActivity(intent2);
                 return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

9.11.1.2 activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="#FFFFFF"
    tools:context=".MainActivity" >

    <ImageView
        android:id="@+id/BackgroundImageView"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_marginBottom="95dp"
        android:layout_marginLeft="40dp"
        android:layout_marginRight="40dp"
        android:scaleType="center" />

       <TextView
           android:id="@+id/text"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:layout_alignParentBottom="true"
           android:layout_alignParentRight="true"
           android:layout_toRightOf="@+id/boton"
           android:shadowColor="#8a6603"
           android:shadowDx="1"
           android:shadowDy="0.5"
           android:shadowRadius="1.5"
           android:textColorHint="#000000"
           android:textSize="16sp" />

       <Button
           android:id="@+id/boton"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_alignBottom="@+id/text"
           android:layout_centerHorizontal="true"
           android:layout_margin="3dp"
           android:background="@drawable/redondeo"
           android:text="@string/escanear"
           android:textColor="#ffffff"
           android:textSize="23sp"
           android:typeface="sans" />

</RelativeLayout>
```

9.11.1.3 Lector.java

```
/**
 * Actividad que invoca al lector de QR
 * y envía el resultado a la siguiente actividad
 *
 * @author Inmaculada García Machado
 * @version 1.0
 */
package com.qrtest;

import android.os.Bundle;
import android.app.Activity;
```

```
import android.content.Intent;

public class Lector extends Activity {
	Bundle modo;
	String modos;
	/**
	 * Método inicial.
	 * Se obtiene el parámetro que pasa la actividad anterior, el modo en el que
está trabajando.
	 * Se invoca al lector de QR.
	 * @param savedInstanceState
	 */
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		//Esta actividad no tiene layout
		//setContentView(R.layout.activity_main);
		//Obtenemos la variable modo (online u offline)
		modo = getIntent().getExtras();
		modos = modo.getString("modo");

		//Utilizamos un lector QR que hay instalado en el dispositivo
		Intent intent = new Intent ("com.google.zxing.client.android.SCAN");
		intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
		startActivityForResult(intent,0);
	}
	/**
	 * Método para obtener el resultado del lector y lo pasamos a la siguiente
actividad
	 * que dependerá del valor de 'modo'
	 * @param requestCode Código de la solicitud
	 * @param resultCode Código del resultado. Puede ser RESULT_OK o
RESULT_CANCELLED
	 * @param intent De aquí obtenemos el resultado del lector y otros muchos
	 * parámetros como el tipo de código leído (QR, de barras)...
	 */

	public void onActivityResult (int requestCode, int resultCode, Intent intent){
		if (requestCode == 0) {
			if (resultCode == RESULT_OK) {
				String contents = intent.getStringExtra("SCAN_RESULT");
				if (modos.equals("offline")){
					Intent i = new Intent (Lector.this,
VistaWebOffline.class);
					i.putExtra( "contenido", contents.toString());
					startActivity(i);
					finish();//Con esta sentencia eliminamos lo que
quede en caché de esta actividad.
				}else if (modos.equals("online")){
					Intent i = new Intent (Lector.this,
VistaWeb.class);
					i.putExtra( "contenido", contents.toString());
					startActivity(i);
					finish();//Con esta sentencia eliminamos lo que
quede en caché de esta actividad.
				}
			}else if (resultCode == RESULT_CANCELED) {
				finish();
			}
		}
	}
}
```

9.11.1.4 VistaWeb.java

```
/**
 * Actividad que muestra si una invitación es válida o no.
 * @author Inmaculada García Machado
 * @version 1.0
 */
package com.qrtest;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Button;
import java.util.StringTokenizer;

public class VistaWeb extends Activity {
	Button botonvolver;
	Button botonescanear;
	WebView myWebView;
	/**
	 * Método inicial.
	 * Declaramos los controles. Obtenemos los parámetros de la
	 * actividad anterior. El resultado del QR lo dividimos para una
	 * fácil comprensión. Cargamos en el WebView la página que muestra
	 * la respuesta del servidor sobre si es válida o no la invitación
	 * @param savedInstanceState
	 */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.vistaweb);

		int contador = 1;
		String codid = "";    //Código de identificación. Hash
		String codconf = "";  //Código de conferencia

		botonvolver = (Button)findViewById(R.id.botonvolver);
		botonescanear = (Button)findViewById(R.id.botonescanear);
		myWebView = (WebView) this.findViewById(R.id.webView);
		myWebView.setWebViewClient(new WebViewClient());

		//Obtenemos los parámetros recibidos (resultado de la lectura del QR)
		Bundle reicieveParams = getIntent().getExtras();
		String contents = reicieveParams.getString("contenido");

		StringTokenizer tokens=new StringTokenizer(contents, "##");
		while (tokens.hasMoreTokens()){
			tokens.nextToken();
			contador ++;

			if(contador == 4){
				codid = tokens.nextToken();
				codconf = tokens.nextToken();
				contador = 0;
			}
		}
```

```
myWebView.loadUrl("https://upctportal.upct.es/upct/conferenciaQrServlet?id="+codid+"&
codConferencia="+codconf);
              myWebView.setWebViewClient(new WebViewClient() {
              //Si hay algún error o no hay conexión a internet, se muestra una
página de error.
                     @Override
                     public void onReceivedError(WebView view, int errorCode, String
description, String failingUrl) {
                          myWebView.loadUrl ("file:///android_asset/errorpage.html");
                     }
              });

              //Pulsar el botón inicio cerramos la ventana y volveremos a la anterior
              botonvolver.setOnClickListener(new OnClickListener() {
                /**
                 * Al pulsar el botón inicio cerramos la ventana y volveremos a la
                 * actividad principal
                 * @param v View
                 */
                public void onClick(View v) {
                       Intent in = new Intent(VistaWeb.this, MainActivity.class);
                       startActivity(in);
                       finish();
                }
              });

              //Al pulsar el botón volver a escanear, vuelve al lector de qr
              botonescanear.setOnClickListener(new OnClickListener() {
                     /**
                      * Al pulsar el botón volver a escanear, vuelve al lector de qr
                      * y le pasamos el modo en el que estamos trabajando
                      * @param v View
                      */
                     public void onClick(View v) {
                            Intent intent = new Intent (VistaWeb.this, Lector.class);
                            intent.putExtra("modo", "online");
                            startActivity(intent);
                            finish();
                     }
              });
       }

       /**
        * Deshabilitamos el botón de atrás en esta actividad
        */
       @Override
       public void onBackPressed() {
              return;
       }

       /**
        * Menú. Con este método indicamos que menú es el de esta pantalla
        * @param menu
        */
       @Override
       public boolean onCreateOptionsMenu(Menu menu) {
           getMenuInflater().inflate(R.menu.offlinemenu, menu);
           return true;
       }
       /**
        * Menú. Con este método indicamos que se debe hacer cuando se pulsa un
        * elemento del menú
        * @param item Elemento del menú
        * @return Si existe ese item se realiza lo que se debe y se devuelve true
        */
```

```
        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
               case R.id.MostrarLista:
                       Intent intent4 = new Intent (VistaWeb.this,
ListaInvitados.class);
                       startActivity(intent4);
                              return true;
               default:
                    return super.onOptionsItemSelected(item);
            }
        }
}
```

9.11.1.5 *vistaweb.xml*

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFFFF"
    tools:context=".VistaWeb" >

    <Button
        android:id="@+id/botonvolver"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_margin="8dp"
        android:layout_marginBottom="8dp"
        android:background="@drawable/redondeo2"
        android:gravity="center_horizontal"
        android:text="@string/inicio"
        android:textColor="#ffffff"
        android:textSize="20sp"
        android:typeface="sans" />

    <Button
        android:id="@+id/botonescanear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/botonvolver"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_margin="10dp"
        android:background="@drawable/redondeo"
        android:gravity="center_horizontal"
        android:text="@string/volverescanear"
        android:textColor="#ffffff"
        android:textSize="35sp"
        android:typeface="sans" />

       <WebView
        android:id="@+id/webView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/botonescanear"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" />
</RelativeLayout>
```

9.11.1.6 ListaInvitados.java

```
/** Actividad que muestra la lista de invitados cargada desde el archivo xml
descargado.
 * Si no hay lista muestra un mensaje y se sale de la actividad.
 * En caso contrario, te muestra una lista de invitados mostrando sólo el NIF,
 * apellidos y nombre de cada invitado.
 * @author Inmaculada García Machado
 * @version 1.0
 */
package com.qrtest;

import java.io.File;
import java.util.List;

import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.text.method.ScrollingMovementMethod;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class ListaInvitados extends Activity {

	List<ClasePruebas> clasepruebas;
	TextView txtView5;
	File file;
	/**
	 * Método inicial.
	 * Se declaran los controles (textview, button)
	 * Se comprueba si existe un archivo llamado invitaciones.xml y si existe se
va a mostrarlista().
	 * @param savedInstanceState
	 */
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.listainvitados);

		txtView5 = (TextView)findViewById(R.id.textView5);
		txtView5.setMovementMethod(new ScrollingMovementMethod());
		Button btninicio = (Button)findViewById(R.id.botonvolver5);

		file = new File(
Environment.getExternalStorageDirectory().getAbsolutePath()+
"/Android/data/com.qrtest/files", "invitaciones.xml");
		if ( file.exists())
			mostrarlista();
		else{
			Toast.makeText(ListaInvitados.this,"No hay lista de
invitados",Toast.LENGTH_LONG).show();
			finish();
		}

		btninicio.setOnClickListener(new OnClickListener() {
		  public void onClick(View v) {

			finish();
		  }
	    });
	}
	/**
	 * Método que llama a una tarea asíncrona para cargar un XML en segundo plano
```

```
         */
        private void mostrarlista() {

                //List<ClasePruebas> clasepruebas;
                //Tarea Asíncrona para cargar un XML en segundo plano
                CargarXmlTask tarea = new CargarXmlTask();
                tarea.execute();
        }

        /**
         * Tarea asíncrona para cargar un XML en segundo plano
         */
        private class CargarXmlTask extends AsyncTask<String, Integer,Boolean>{
                /**
                 * Parsea los datos xml a una lista
                 * @param params
                 * @return Devuelve true y se continua en el método onPostExecute()
                 */
                protected Boolean doInBackground(String... params){
                        RssParserDom dom = new RssParserDom();
                        clasepruebas = dom.parse();
                        return true;
                }
                /**
                 * Para mostrar los datos. Recorre la lista clasepruebas y va añadiendo
                 * a la lista de invitados
                 * @param result No lo usamos
                 */
                protected void onPostExecute(Boolean result){
                        txtView5.setText("");
                        for (int i=0; i<clasepruebas.size(); i++){
                                txtView5.setText(txtView5.getText().toString()+
clasepruebas.get(i).getDni()+" - "+ clasepruebas.get(i).getNombre()+
System.getProperty("line.separator")+System.getProperty("line.separator"));
                        }
                }
        }
}
```

9.11.1.7 RssParserDom.java

```
/**
 * Parsea el archivo XML de la lista de invitados
 * @author Inmaculada García Machado
 * @version
 */
package com.qrtest;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import android.os.Environment;

public class RssParserDom {

        //Depende del dispositivo el directorio será distinto
```

```
        //File file = new File(
Environment.getExternalStorageDirectory().getAbsolutePath()+
"/Android/data/com.qrtest/files/storage/sdcard0", "invitacion.xml");    //Samsung
Galaxy SII
        File file = new File(
Environment.getExternalStorageDirectory().getAbsolutePath()+
"/Android/data/com.qrtest/files", "invitaciones.xml");    //Sony Xperia E
        /**
         * Este método parsea el archivo XML devolviendo una lista con los invitados
         * @return Lista con los datos de los invitados
         */
    public List<ClasePruebas> parse()
    {
        //Instanciamos la fábrica para DOM
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        List<ClasePruebas> clasepruebas = new ArrayList<ClasePruebas>();

        try{
            //Creamos un nuevo parser DOM
            DocumentBuilder builder = factory.newDocumentBuilder();
            //Realizamos lalectura completa del XML
            Document dom = builder.parse(file);  //this.getInputStream());
            //Nos posicionamos en el nodo principal del árbol (<qr>)
            Element root = dom.getDocumentElement();
            //Localizamos todos los elementos <invitacion>
            NodeList items = root.getElementsByTagName("invitacion");
            //Recorremos la lista de invitaciones
            for (int i=0; i<items.getLength(); i++){
                ClasePruebas claseprueba = new ClasePruebas();
                //Obtenemos la invitación actual
                Node item = items.item(i);
                //Obtenemos la lista de datos de la invitación actual
                NodeList datosPersona = item.getChildNodes();
                //Procesamos cada dato de la invitación
                //System.out.println("Procesamos cada dato de la invitacion");
                for (int j=0; j<datosPersona.getLength(); j++)
                {
                    Node dato = datosPersona.item(j);
                    String etiqueta = dato.getNodeName();

                    if (etiqueta.equals("nif")){
                        String texto = obtenerTexto(dato);
                        claseprueba.setNif(texto);
                        //claseprueba.setNombre(dato.getFirstChild().getNodeValue());
Para obtener hijos de un item
                    }else if (etiqueta.equals("nombre")){
                      String texto = obtenerTexto(dato);
                        claseprueba.setNombre(texto);
                    }else if (etiqueta.equals("hash")){
                        String texto = obtenerTexto(dato);
                        claseprueba.setHash(texto);
                    }else if (etiqueta.equals("estado")){
                      String texto = obtenerTexto(dato);
                        claseprueba.setEstado(texto);
                    }else if (etiqueta.equals("numLecturas")){
                      String texto = obtenerTexto(dato);
                        claseprueba.setLecturas(texto);
                    }
                }
                clasepruebas.add(claseprueba);
            }
        }catch (Exception ex){
            throw new RuntimeException(ex);
        }
        return clasepruebas;
    }
```

```
    /**
     * Convierte el nodo obtenido en un string
     * @param dato
     * @return string
     */
    private String obtenerTexto(Node dato)
    {
        StringBuilder texto = new StringBuilder();
        NodeList fragmentos = dato.getChildNodes();

        for (int k=0;k<fragmentos.getLength();k++)
        {
            texto.append(fragmentos.item(k).getNodeValue());
        }
        return texto.toString();
    }
}
```

9.11.1.8 listainvitados.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFFFF" >

       <Button
           android:id="@+id/botonvolver5"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_alignParentBottom="true"
           android:layout_centerHorizontal="true"
           android:layout_marginBottom="14dp"
           android:layout_margin="10dp"
           android:background="@drawable/redondeo2"
           android:gravity="center_horizontal"
           android:paddingLeft="40dp"
           android:paddingRight="40dp"
           android:textColor="#ffffff"
           android:textSize="20sp"
           android:text="Volver"
           android:typeface="sans" />

       <TextView android:id="@+id/textView5"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_below="@+id/textView1"
           android:layout_above="@+id/botonvolver5"
           android:layout_centerHorizontal="true"
           android:layout_margin="10dp"
           android:layout_marginTop="50dp"></TextView>

       <TextView
           android:id="@+id/textView1"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_alignParentTop="true"
           android:layout_centerHorizontal="true"
           android:layout_marginTop="21dp"
           android:text="Lista de invitados"
           android:textSize="18sp" />
</RelativeLayout>
```

9.11.1.9 Registro.java

```
/**
 * Actividad que pide una contraseña para poder acceder
 * a la configuración de la aplicación como descargar la lista
 * de invitados y modificar el fondo de la pantalla principal de la app
 * @author Inmaculada García Machado
 * @version 1.0
 */
package com.qrtest;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Button;
import android.widget.Toast;

public class Registro extends Activity {

	String user = "";
	String password = "";
	String contents = "";
	/**
	 * Método inicial.
	 * Obtenemos el parámetro de entrada e indicamos que hacer
	 * cuando se pulsa el botón, que será ir a un método para comprobar
	 * la contraseña
	 * @param savedInstanceState
	 */
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.registro);

		//Obtenemos el parámetro de entrada 'funcion'
		Bundle reicieveParams = getIntent().getExtras();
		contents = reicieveParams.getString("funcion");

		final EditText TxtPassword = (EditText)findViewById(R.id.TxtPassword);
		Button entrar = (Button)findViewById(R.id.BtnAceptar);
		entrar.setOnClickListener(new OnClickListener() {
			public void onClick(View v) {
				password = TxtPassword.getText().toString();
				comprobacion();
			}
		});
	}
	/**
	 * Se comprueba si la contraseña es correcta.
	 * En caso afirmativo se mira el valor del parametro de entrada
	 * y dependiendo de su valor invoca a la actividad para descargar la lista
	 * o invoca la actividad para descarga la imagen de fondo.
	 * Si la contraseña es incorrecta se sale y vuelve a MainActivity
	 */
	public void comprobacion(){
		if ( password.equals("admin")){ //Si la contraseña es correcta

			if (contents.equals("fichero")){ //a Invitados.class
				Intent inte = new Intent (Registro.this,
Invitados.class);
				inte.putExtra("semi", password);
				startActivity(inte);
```

```
                        }else if ( contents.equals("fondo")) { //a Fondo.class
                                Intent inten = new Intent (Registro.this, Fondo.class);

                                startActivity(inten);
                        }
                }else { //Si la contraseña es incorrecta
                        Toast.makeText(Registro.this,"Contraseña
incorrecta",Toast.LENGTH_LONG).show();
                        finish();
                }
        }
}
```

9.11.1.10 registro.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="vertical"
  android:padding="10dip"
  android:background="#FFFFFF">

    <TextView android:id="@+id/TextView02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/contrasena"
        android:textStyle="bold" />

    <EditText android:id="@+id/TxtPassword"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:inputType="textPassword" />

    <LinearLayout android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <TextView android:id="@+id/LblMensaje"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="10dip"
            android:textStyle="bold" />
    </LinearLayout>

    <Button
        android:id="@+id/BtnAceptar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/redondeo2"
        android:paddingLeft="20dip"
        android:paddingRight="20dip"
        android:text="@string/login"
        android:textColor="#ffffff" />
</LinearLayout>
```

9.11.1.11 Invitados.java

```
/**
 * Actividad para descargar la lista de invitados
 * @author Inmaculada García Machado
```

```
 * @version 1.0
 */
package com.qrtest;

import java.io.File;
import java.io.FileNotFoundException;

import android.app.Activity;
import android.app.DownloadManager;
import android.app.ProgressDialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.ContextWrapper;
import android.content.Intent;
import android.content.IntentFilter;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.ParcelFileDescriptor;
import android.os.ResultReceiver;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class Invitados extends Activity {

	EditText codConfe;
	String codcon;
	String semilla;
	private long id;
	Uri uri;
	private DownloadManager downloadManager;
	protected ProgressDialog mProgressDialog;

	/**
	 * Método inicial. Recibe de la anterior actividad un parámetro que
	* usaremos como semilla en la encriptación del código de conferencia
	* @param savedInstanceState
	*/
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.invitados);

		IntentFilter filter = new
IntentFilter(DownloadManager.ACTION_DOWNLOAD_COMPLETE);
		registerReceiver(downloadReceiver, filter);

		Bundle reicieveParams = getIntent().getExtras();
		semilla = reicieveParams.getString("semi");

		codConfe = (EditText)findViewById(R.id.codConfe);
		Button btninicio = (Button)findViewById(R.id.botonvolver4);
		btninicio.setOnClickListener(new OnClickListener() {
		  public void onClick(View v) {
			Intent intent = new Intent (Invitados.this,
MainActivity.class);
			startActivity(intent);
		  }
		});
	}
```

```
	/**
	 * Deshabilitamos el botón de atrás.
	 */
	@Override
	public void onBackPressed() {
		return;
	}
	/**
	 * Método por si la descarga se quedó a medias cuando se salió de la
actividad.
	 */
	@Override
	protected void onResume() {
		super.onResume();
		IntentFilter intentFilter = new
IntentFilter(DownloadManager.ACTION_DOWNLOAD_COMPLETE);
		registerReceiver(downloadReceiver, intentFilter);
	}
	/**
	 * Método que se ejecuta si se sale de la actividad para guardar el estado de
	 * la descarga.
	 */
	@Override
	protected void onPause(){
		super.onPause();
		unregisterReceiver(downloadReceiver);
	}
	/**
	 * Método que inicia la descarga.
	 * Primero encripta el código de conferencia y usa como semilla el parámetro
	 * recibido semilla.
	 * Después durante la descarga se muestra un ProgressDialog.
	 * Indicamos los parámetros necesarios para realizar la descarga y se inicia.
	 */
	public void descargar(View button) {
		String id="";
		codcon = codConfe.getText().toString();
		if (codcon.isEmpty()){
			Toast.makeText(this, "Debe poner un código",
Toast.LENGTH_LONG).show();
		} else {
			try {
				id = UtilidadesCifrado.encrypt(semilla, codcon);
			} catch (Exception e) {
				e.printStackTrace();
			}

			mProgressDialog = new ProgressDialog(this);
			mProgressDialog.show();
			Intent intent = new Intent(this, DownloadService.class);
			String directory = new
ContextWrapper(this).getApplicationContext().getExternalFilesDir(DOWNLOAD_SERVICE).ge
tAbsolutePath();

			intent.putExtra("url",
"https://upctportal.upct.es/upct/conferenciaInscritosServlet?id=" + id);
			intent.putExtra("file", "invitaciones.xml");
			intent.putExtra("directory", directory.substring(0,
directory.lastIndexOf("/"))); //this.getApplicationContext().getFilesDir());
			intent.putExtra("receiver", new DownloadReceiver(new
Handler()));
			startService(intent);
		}
	}
	/**
	 * Método que comprueba si existe ya un archivo llamado invitaciones.xml
```

```
         * @param filename Nombre del archivo para comprobar si existe uno igual
         * en un directorio
         */
        public boolean isFileExists(String filename){
            File folder1 = new
File(Environment.getExternalStorageDirectory().getAbsolutePath()+"/Android/data/com.q
rtest/files/mnt/sdcard/"+ filename);
            return folder1.exists();
        }
        /**
         * Borra un archivo situado en un directorio
         * @param filename Nombre del archivo que se quiere eliminar
         */
        public boolean deleteFile(String filename){
            File folder1 = new
File(Environment.getExternalStorageDirectory().getAbsolutePath()+"/Android/data/com.q
rtest/files/mnt/sdcard/"+ filename);
            return folder1.delete();
        }

        /**
         * Muestra las últimas descargas realizadas con el servicio DownloadManager
         * @param button Botón pulsado
         */
        public void ver(View button){
               Intent intent = new Intent();
               intent.setAction(DownloadManager.ACTION_VIEW_DOWNLOADS);
               startActivity(intent);
        }
        /**
         * Invoca a otra actividad que muestra las instrucciones para descargar
         * el fichero.
         * @param button Botón pulsado
         */
        public void instrucciones(View button) {
               Intent intent = new Intent(Invitados.this, Instrufile.class);
               startActivity(intent);
        }

        private BroadcastReceiver downloadReceiver = new BroadcastReceiver() {
          /**
          * Gestionamos la finalización de la descarga
          * @param context
          * @param intent
          */
               @Override
               public void onReceive(Context context, Intent intent) {
                      Log.d(this.getClass().getName(), "Recepcion de datos (1)");
                      DownloadManager.Query query = new DownloadManager.Query();
                      query.setFilterById(id, 0);
                      Cursor cursor = downloadManager.query(query);
                      Log.d(this.getClass().getName(), "Recepcion de datos (2)");

                      if(cursor.moveToFirst()){
                             int status =
cursor.getInt(cursor.getColumnIndex(DownloadManager.COLUMN_STATUS));
                             int reason =
cursor.getInt(cursor.getColumnIndex(DownloadManager.COLUMN_REASON));

                             if(status == DownloadManager.STATUS_SUCCESSFUL){
                              //podemos recuperar el fichero descargado
                                ParcelFileDescriptor file = null;
                                Log.d(this.getClass().getName(), "download begining --
----------------------------------------> (4)");
                                try{
                                     file = downloadManager.openDownloadedFile(id);
```

```
                                    Toast.makeText(Invitados.this,"Fichero obtenido
con éxito",Toast.LENGTH_LONG).show();
                                }catch (FileNotFoundException ex){
                                    Toast.makeText(Invitados.this,"Exception: " +
ex.getMessage(),Toast.LENGTH_LONG).show();
                                }
                            }else if(status == DownloadManager.STATUS_FAILED){
                             Toast.makeText(Invitados.this,"FAILED: " +
reason,Toast.LENGTH_LONG).show();
                            }else if(status == DownloadManager.STATUS_PAUSED){
                             Toast.makeText(Invitados.this, "PAUSED: " + reason,
Toast.LENGTH_LONG).show();
                            }else if(status == DownloadManager.STATUS_PENDING){
                             Toast.makeText(Invitados.this, "PENDING: " + reason,
Toast.LENGTH_LONG).show();
                            }else if(status == DownloadManager.STATUS_RUNNING){
                             Toast.makeText(Invitados.this, "RUNNING: " + reason,
Toast.LENGTH_LONG).show();
                            }
                        }
                }
        };
        /**
       * Manejador del resultado.
       * Si se ha descargado con éxito mostramos un mensaje
       */
        private class DownloadReceiver extends ResultReceiver{
            public DownloadReceiver(Handler handler) {
                super(handler);
            }
            @Override
            protected void onReceiveResult(int resultCode, Bundle resultData) {
                super.onReceiveResult(resultCode, resultData);

                if (resultCode == DownloadService.UPDATE_PROGRESS) {
                    int progress = resultData.getInt("progress");
                    mProgressDialog.setProgress(progress);
                    if (progress == 100) {
                        mProgressDialog.dismiss();
                        Toast.makeText(Invitados.this, "Descarga realizada
correctamente", Toast.LENGTH_LONG).show();
                    }
                }
            }
    }
}
```

9.11.1.12 invitados.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFFFF" >
    <TextView
        android:id="@+id/codigo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="20dp"
        android:text="@string/codConferencia"
        android:textStyle="bold"
        android:textSize="15sp" />
```

```
    <EditText
        android:id="@+id/codConfe"
        android:layout_width="40dp"
        android:layout_height="35dp"
        android:layout_alignBaseline="@+id/codigo"
        android:layout_toRightOf="@+id/codigo"
        android:ems="10"
        android:hint="nº"
        android:textSize="15sp"
        android:inputType="number" >
        <requestFocus></requestFocus>
    </EditText>
    <Button
        android:id="@+id/descargarfichero"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:background="@drawable/redondeo2"
        android:text="@string/descargar"
        android:textSize="20sp"
        android:textColor="#ffffff"
        android:onClick="descargar"
        android:layout_below="@+id/codigo" />
    <Button
            android:id="@+id/botonvolver4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerHorizontal="true"
            android:layout_marginBottom="30dp"
            android:layout_margin="10dp"
            android:background="@drawable/redondeo2"
            android:gravity="center_horizontal"
            android:paddingLeft="40dp"
            android:paddingRight="40dp"
            android:text="@string/inicio"
            android:textColor="#ffffff"
            android:textSize="20sp"
            android:typeface="sans" />
    <Button
        android:id="@+id/ver"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/descargarfichero"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="18dp"
        android:background="@drawable/redondeo2"
        android:onClick="ver"
        android:text="@string/ver"
        android:textColor="#ffffff"
        android:textSize="20sp" />
    <Button
        android:id="@+id/instrufile"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/ver"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="18dp"
        android:background="@drawable/redondeo2"
        android:onClick="instrucciones"
        android:text="@string/instrucciones"
        android:textColor="#ffffff"
        android:textSize="20sp" />
</RelativeLayout>
```

9.11.1.13 Instrufile.java

```
/**
 *  Actividad que muestra un string con las instrucciones para descargar
 *  el archivo de la lista de invitados y tiene un botón para volver.
 *  @author Inmaculada García Machado
 *  @version 1.0
 */
package com.qrtest;

import android.app.Activity;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class Instrufile extends Activity {
	/**
	 * Método inicial.
	 * Se declaran los controles como el botón y los textview.
	 * @param savedInstanceState
	 */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.instrufile);

		Button btnvolver7 = (Button)findViewById(R.id.botonvolver7);
		TextView textView7 = (TextView)findViewById(R.id.textView7);
		textView7.setMovementMethod(new ScrollingMovementMethod());
		btnvolver7.setOnClickListener(new OnClickListener() {
			/**
			 * Si se pulsa el botón la actividad se cierra y volvemos a
			 * la actividad anterior
			 * @param v
			 */
			public void onClick(View v) {
			   finish();
			 }
		});
	}
}
```

9.11.1.14 instrufile.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFFFF" >
       <TextView
           android:id="@+id/textView7"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_marginTop="20dp"
           android:layout_marginLeft="20dp"
           android:layout_marginRight="20dp"
           android:layout_marginBottom="10dp"
           android:text="@string/txtinstfile"
           android:textSize="20sp" />
       <Button
```

```
            android:id="@+id/botonvolver7"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerHorizontal="true"
            android:layout_marginBottom="14dp"
            android:layout_margin="10dp"
            android:background="@drawable/redondeo2"
            android:gravity="center_horizontal"
            android:paddingLeft="40dp"
            android:paddingRight="40dp"
            android:text="@string/volver"
            android:textColor="#ffffff"
            android:textSize="20sp"
            android:typeface="sans" />
</RelativeLayout>
```

9.11.1.15 Fondo.java

```
/**
 * Actividad para descargar la imagen que será el fondo en la pantalla principal
 * @author Inmaculada García Machado
 * @version 1.0
 */
package com.qrtest;

import java.io.File;
import java.io.FileNotFoundException;

import android.app.Activity;
import android.app.DownloadManager;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.os.ParcelFileDescriptor;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class Fondo extends Activity {
	private long id;
	EditText codConf;
	String codcon;
	private DownloadManager downloadManager;
	/**
	 * Método inicial. Declaramos los controles del botón y el textview
	 * e indicamos que hacer en caso de ser pulsado el botón
	 * @param savedInstanceState
	 */
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.fondo);

		IntentFilter filter = new
IntentFilter(DownloadManager.ACTION_DOWNLOAD_COMPLETE);
		registerReceiver(downloadReceiver, filter);
```

```
		codConf = (EditText)findViewById(R.id.codConf);
		Button btninicio = (Button)findViewById(R.id.botonvolver5);
		btninicio.setOnClickListener(new OnClickListener() {
		  /**
		  * Método que se usan en caso de que se pulse el botón.
		  * Si es pulsado invocará a la actividad MainActivity para volver a la
		  * pantalla principal.
		  * @param v View
		  */
		  public void onClick(View v) {
			Intent intent = new Intent (Fondo.this, MainActivity.class);
				startActivity(intent);
		  }
		});
	}

	/**
	 * Deshabilitamos el botón de atrás en esta actividad.
	 * Para volver ya está el botón Volver.
	 */
	@Override
	public void onBackPressed() {
		return;
	}
	/**
	 * Método por si la descarga se quedó a medias cuando se salió de la
actividad.
	 */
	@Override
	protected void onResume() {
		super.onResume();
		IntentFilter intentFilter = new
IntentFilter(DownloadManager.ACTION_DOWNLOAD_COMPLETE);
		registerReceiver(downloadReceiver, intentFilter);
	}
	/**
	 * Método que se ejecuta si se sale de la actividad para guardar el estado de
	 * la descarga.
	 */
	@Override
	protected void onPause(){
		super.onPause();
		unregisterReceiver(downloadReceiver);
	}
	/**
	 * Método que inicia la descarga. Si se pulsa el botón Descargar fondo entra
en este
	 * método.
	 * Si el cod. de conferencia no está vacío, se comprueba que no exista ya una
imagen
	 * con el mismo nombre. Si existe se elimina.
	 * Después durante la descarga se muestra un ProgressDialog.
	 * Indicamos los parámetros necesarios para realizar la descarga y se inicia.
	 * En este caso no se necesita semilla ni encriptar el codigo de conferencia
	 * porque es una imagen y no necesitamos seguridad.
	 * @param button View
	 */
	public void descargar(View button) {
		codcon = codConf.getText().toString();
		if (codcon.isEmpty()){
			Toast.makeText(Fondo.this, "Debe poner un código",
Toast.LENGTH_LONG).show();
		} else {
			downloadManager =
(DownloadManager)getSystemService(DOWNLOAD_SERVICE);
			//Para sobreescribir si existe otro fondo
```

```
                        String filename = "fondo.png";
                        if (isFileExists(filename)){
                                Boolean correcto = deleteFile(filename);
                                if (correcto){
                                        Toast.makeText(Fondo.this, "Se va a sobreescribir
el fondo", Toast.LENGTH_LONG).show();
                                }
                        }
                        DownloadManager.Request request = new
DownloadManager.Request(Uri.parse("https://uxxiportal.upct.es/portlets/images/confere
ncias/"+codcon+".png"));

//podemos limitar la descarga a un tipo de red (opcional)
//request.setAllowedNetworkTypes(DownloadManager.Request.NETWORK_WIFI);
                //Esta información  se mostrará en el área de notificaciones
                        request.setTitle("Descarga");
                        request.setDescription("Fondo imagen app.");
                        //Guardamos el fondo
                        if
(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)){
                           request.setDestinationInExternalFilesDir(this,
Environment.getExternalStorageDirectory().getAbsolutePath()+ File.separator ,
"fondo.png");
                        }
                        //Iniciamos la descarga
                        id = downloadManager.enqueue(request);
                }
        }
        /**
         * Método que comprueba si existe ya un archivo llamado fondo.png
         * @param filename Nombre de la imagen para comprobar si existe uno igual
         * en un directorio
         */
        public boolean isFileExists(String filename){
            File folder1 = new
File(Environment.getExternalStorageDirectory().getAbsolutePath()+"/Android/data/com.q
rtest/files/mnt/sdcard/"+ filename);
            return folder1.exists();
        }
        /**
         * Borra un archivo situado en un directorio
         * @param filename Nombre del archivo que se quiere eliminar
         */
        public boolean deleteFile(String filename){
            File folder1 = new
File(Environment.getExternalStorageDirectory().getAbsolutePath()+"/Android/data/com.q
rtest/files/mnt/sdcard/"+ filename);
            return folder1.delete();
        }

        /**
         * Muestra las últimas descargas realizadas con el servicio DownloadManager.
         * Este método se ejecuta cuando se pulsa 'Ver la descarga'
         * @param button Botón pulsado
         */
        public void ver(View button) {
                Intent intent = new Intent();
                intent.setAction(DownloadManager.ACTION_VIEW_DOWNLOADS);
                startActivity(intent);
        }
        /**
         * Invoca a otra actividad que muestra las instrucciones para descargar
         * el fichero. Se ejecuta cuando se pulsa 'Instrucciones'
         * @param button Botón pulsado
         */
        public void instrucciones(View button) {
```

```
                Intent intent = new Intent(Fondo.this, Instrucciones.class);
                startActivity(intent);
        }

        private BroadcastReceiver downloadReceiver = new BroadcastReceiver() {

         /**
          * Gestionamos la finalización de la descarga
          * @param context
          * @param intent
          */
           @Override
           public void onReceive(Context context, Intent intent) {
              DownloadManager.Query query = new DownloadManager.Query();
              query.setFilterById(id, 0);
              Cursor cursor = downloadManager.query(query);
             if(cursor.moveToFirst()) {
               int status =
cursor.getInt(cursor.getColumnIndex(DownloadManager.COLUMN_STATUS));
               int reason =
cursor.getInt(cursor.getColumnIndex(DownloadManager.COLUMN_REASON));

               if(status == DownloadManager.STATUS_SUCCESSFUL){
                  //podemos recuperar el fichero descargado
                  ParcelFileDescriptor file = null;
                  try{
                      file = downloadManager.openDownloadedFile(id);
                      Toast.makeText(Fondo.this,"Fichero obtenido con
éxito",Toast.LENGTH_LONG).show();
                  }catch (FileNotFoundException ex){
                      Toast.makeText(Fondo.this,"Exception: " +
ex.getMessage(),Toast.LENGTH_LONG).show();
                  }
               }else if(status == DownloadManager.STATUS_FAILED){
                  Toast.makeText(Fondo.this,"FAILED: " +
reason,Toast.LENGTH_LONG).show();
               }else if(status == DownloadManager.STATUS_PAUSED){
                  Toast.makeText(Fondo.this, "PAUSED: " + reason,
Toast.LENGTH_LONG).show();
               }else if(status == DownloadManager.STATUS_PENDING){
                  Toast.makeText(Fondo.this, "PENDING: " + reason,
Toast.LENGTH_LONG).show();
               }else if(status == DownloadManager.STATUS_RUNNING){
                  Toast.makeText(Fondo.this, "RUNNING: " + reason,
Toast.LENGTH_LONG).show();
               }
            }
         }
    };
}
```

9.11.1.16 fondo.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFFFF" >
    <TextView
        android:id="@+id/codConferencia"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/codConferencia"
        android:textStyle="bold"
```

```
        android:layout_marginTop="20dp"
        android:layout_marginLeft="20dp"
        android:textSize="15sp" />
    <EditText
        android:id="@+id/codConf"
        android:layout_width="40dp"
        android:layout_height="35dp"
        android:layout_alignBaseline="@+id/codConferencia"
        android:layout_toRightOf="@+id/codConferencia"
        android:ems="10"
        android:hint="nº"
        android:textSize="15sp"
        android:inputType="number" >
       <requestFocus></requestFocus>
    </EditText>
    <Button
       android:id="@+id/botonvolver5"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_alignParentBottom="true"
       android:layout_centerHorizontal="true"
       android:layout_marginBottom="30dp"
       android:layout_margin="10dp"
       android:background="@drawable/redondeo2"
       android:gravity="center_horizontal"
       android:paddingLeft="40dp"
       android:paddingRight="40dp"
       android:text="@string/inicio"
       android:textColor="#ffffff"
       android:textSize="20sp"
       android:typeface="sans" />"
    <Button
       android:id="@+id/descargarfondo"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_centerHorizontal="true"
       android:layout_marginTop="17dp"
       android:background="@drawable/redondeo2"
       android:onClick="descargar"
       android:text="@string/descfondo"
       android:textColor="#ffffff"
       android:textSize="20sp"
       android:layout_below="@+id/codConferencia" />
    <Button
       android:id="@+id/ver"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_below="@+id/descargarfondo"
       android:layout_centerHorizontal="true"
       android:layout_marginTop="18dp"
       android:background="@drawable/redondeo2"
       android:onClick="ver"
       android:text="@string/ver"
       android:textColor="#ffffff"
       android:textSize="20sp" />
    <Button
       android:id="@+id/instrucciones"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_below="@+id/ver"
       android:layout_centerHorizontal="true"
       android:layout_marginTop="18dp"
       android:background="@drawable/redondeo2"
       android:onClick="instrucciones"
       android:text="@string/instrucciones"
       android:textColor="#ffffff"
```

```
        android:textSize="20sp" />
</RelativeLayout>
```

9.11.1.17 UtilidadesCifrado.java

```
/**
 * Clase para encriptar y desencriptar un texto usando AES.
 * Necesita una semilla
 * @author Inmaculada García
 * @version 1.0
 */
package com.qrtest;

import java.security.MessageDigest;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class UtilidadesCifrado {

	/**
	 * Método para encriptar. A la semilla le aplica el MD5
	 * @param seed Semilla
	 * @param cleartext Texto a encriptar
	 * @return toHex(result)
	 */
	public static String encrypt(String seed, String cleartext)
			throws Exception {
		MessageDigest md5 = MessageDigest.getInstance("MD5");
		byte[] result = encrypt(md5.digest(seed.getBytes()),
cleartext.getBytes());
		return toHex(result);
	}

	/**
	 * Método que desencripta. A la semilla le aplica el MD5
	 * @param seed Semilla
	 * @param encrypted texto encriptado
	 * @return result Texto desencriptado
	 */
	public static String decrypt(String seed, String encrypted)
			throws Exception {
		byte[] enc = toByte(encrypted);
		byte[] result =
decrypt(MessageDigest.getInstance("MD5").digest(seed.getBytes()), enc);
		return new String(result);
	}

	/**
	 * Encriptación AES
	 * @param raw  Bloque de 128 bits (semilla  MD5)
	 * @param clear      Texto a encriptar
	 * @throws Exception
	 * @return encrypted Array de byte
	 */
	private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
		SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
		Cipher cipher = Cipher.getInstance("AES");
		cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
		byte[] encrypted = cipher.doFinal(clear);
		return encrypted;
	}
	/**
	 * Desencriptación AES
```

```
         * @param raw Bloque de 128 bits (semilla MD5)
         * @param encrypted Texto encriptado a desencriptar
         * @return decrypted Texto desencriptado
         */
        private static byte[] decrypt(byte[] raw, byte[] encrypted)
                        throws Exception {
                SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
                Cipher cipher = Cipher.getInstance("AES");
                cipher.init(Cipher.DECRYPT_MODE, skeySpec);
                byte[] decrypted = cipher.doFinal(encrypted);
                return decrypted;
        }
        /**
         * Método para convertir a hexadecimal
         * @param txt String a convertir
         * @return String en hexadecimal
         */
        public static String toHex(String txt) {
                return toHex(txt.getBytes());
        }
        /**
         * Método para convertir un hexadecimal a texto
         * @param hex Texto hexadecimal a convertir
         * @return Texto resultado
         */
        public static String fromHex(String hex) {
                return new String(toByte(hex));
        }
        /**
         * Para convertir a Byte
         * @param hexString
         * @return result
         */
        public static byte[] toByte(String hexString) {
                int len = hexString.length() / 2;
                byte[] result = new byte[len];
                for (int i = 0; i < len; i++){
                        result[i] = Integer.valueOf(hexString.substring(2 * i, 2 * i +
2),
                                        16).byteValue();
                }
                return result;
        }
        /**
         * Método para convertir a hexadecimal un array de byte
         * @param buf
         * @return result.toString();
         */
        public static String toHex(byte[] buf) {
                if (buf == null){
                        return "";
                }
                StringBuffer result = new StringBuffer(2 * buf.length);
                for (int i = 0; i < buf.length; i++) {
                        appendHex(result, buf[i]);
                }
                return result.toString();
        }
        private final static String HEX = "0123456789ABCDEF";
        private static void appendHex(StringBuffer sb, byte b) {
                sb.append(HEX.charAt((b >> 4) & 0x0f)).append(HEX.charAt(b & 0x0f));
        }
}
```

9.11.1.18 AcercaDe.java

```
/**
 * En esta actividad solo mostramos por pantalla un AcercaDe con
 * una imagen de la UPCT
 * @author Inmaculada García
 * @version 1.0
 */
package com.qrtest;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class AcercaDe extends Activity {
	/**
	 * Método inicial de la actividad.
	 * Si se pulsa el botón se sale de la actividad y va a
	 * la anterior.
	 * @param savedInstanceState
	 */
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.acercade);

		Button btninicio = (Button)findViewById(R.id.botonvolver3);
		btninicio.setOnClickListener(new OnClickListener() {
		  public void onClick(View v) {
			finish();
		  }
		});
	}
}
```

9.11.1.19 acercade.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFFFF" >
    <ImageView
            android:id="@+id/imageView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_marginLeft="18dp"
            android:layout_marginTop="16dp"
            android:src="@drawable/escudo" />
        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignLeft="@+id/imageView1"
            android:layout_below="@+id/imageView1"
            android:layout_marginTop="30dp"
            android:text="@string/txtacerca" />
        <Button
            android:id="@+id/botonvolver3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
```

```
            android:layout_centerHorizontal="true"
            android:layout_marginBottom="14dp"
            android:layout_margin="10dp"
            android:background="@drawable/redondeo2"
            android:gravity="center_horizontal"
            android:paddingLeft="40dp"
            android:paddingRight="40dp"
            android:text="@string/inicio"
            android:textColor="#ffffff"
            android:textSize="20sp"
            android:typeface="sans" />
</RelativeLayout>
```

9.11.1.20 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.qrtest"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="9"
        android:targetSdkVersion="10" />

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
       <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
       <uses-permission android:name="android.permission.READ_PHONE_STATE" />
       <uses-permission android:name="android.permission.WAKE_LOCK" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <service android:name="com.qrtest.DownloadService"/>
        <activity
            android:name="com.qrtest.MainActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
            android:theme="@android:style/Theme.NoTitleBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.qrtest.VistaWeb"
            android:screenOrientation="portrait">
        </activity>
        <activity
            android:name="com.qrtest.Lector"
            android:screenOrientation="portrait">
        </activity>
        <activity
            android:name="com.qrtest.VistaWebOffline"
            android:screenOrientation="portrait">
        </activity>
        <activity
            android:name="com.qrtest.AcercaDe"
            android:screenOrientation="portrait">
```

```
        </activity>
        <activity
            android:name="com.qrtest.Invitados"
            android:screenOrientation="portrait">
        </activity>
        <activity
            android:name="com.qrtest.Fondo"
            android:screenOrientation="portrait" >
        </activity>
        <activity
            android:name="com.qrtest.Registro"
            android:screenOrientation="portrait">
        </activity>
        <activity
            android:name="com.qrtest.Instrucciones"
            android:screenOrientation="portrait">
        </activity>
        <activity
            android:name="com.qrtest.ListaInvitados"
            android:screenOrientation="portrait">
        </activity>
        <activity
            android:name="com.qrtest.Instrufile"
            android:screenOrientation="portrait">
        </activity>
    </application>

</manifest>
```

9.11.1.21 ConferenciaInscritosServlet.java

```
/**
 * Servlet para descargar el fichero con la lista de los invitados
 *
 * @author Inmaculada C. García Machado
 * @version 1.0
 */
package upct.uxxiportal.util.servlet;

import java.io.IOException;
import java.io.OutputStream;
import java.math.BigDecimal;
import java.sql.Clob;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.io.IOUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import upct.uxxiportal.catalogo.ServiciosConferencia;
import upct.uxxiportal.portlets.conferencia.Conferencia;
import upct.uxxiportal.util.UtilidadesCifrado;

@SuppressWarnings("serial")
public class ConferenciaInscritosServlet extends HttpServlet {
        private static final Log LOG =
LogFactory.getLog(ConferenciaInscritosServlet.class);

        private ServiciosConferencia serv = ServiciosConferencia.getInstance();

    @Override
```

```
        public void init() throws ServletException { }

    @Override
        public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException{}

    /**
     * Se procesa la petición GET. En ella se desencripta el id recibido en request y
si es todo correcto
     * obtenemos los datos de la lista de invitados y los enviamos en un archivo XML
     * @param request Solicitud
     * @param response Respuesta
     * @throws ServletException
     * @throws IOException
     */
    @Override
        public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException{
        // Parámetros del usuario
         String id = request.getParameter("id");
         String codConferencia = null;
               //Obtenemos el código de conferencia desencriptando el id.
         try {
               codConferencia = UtilidadesCifrado.decrypt(serv.getClaveAcceso(), id);
               } catch (Exception e1) {
                     LOG.error("Error al obtener código de conferencia", e1);
               }

         Conferencia conferencia = serv.getConferenciaByCodigo(codConferencia);

         if(conferencia == null){
               throw new ServletException("Foto no encontrada");
         }
         //Si el cód. conferencia es correcto obtenemos los datos de la lista de
invitados y lo enviamos en
         //un archivo XML llamado invitaciones.xml
         byte[] data = null;

               try {
                     Clob file = serv.getInscripciones(new BigDecimal(1));
                     data = IOUtils.toByteArray(file.getAsciiStream());;
               } catch (SQLException e) {
                     e.printStackTrace();
               }

               response.setContentType("application/xml");
               response.setHeader("Content-Disposition",
"attachment;filename=invitaciones.xml");
               response.setContentLength(data.length);

               OutputStream output = response.getOutputStream();
               output.write(data);
               output.flush();
               output.close();
    }
}
```

9.11.1.22 ConferenciaQrServlet.java

```
/**
 * Servlet para la lectura del código QR de la invitación
 *
 * @author Inmaculada C. García Machado
 * @version 1.0
 */
package upct.uxxiportal.util.servlet;
```

```
import java.math.BigDecimal;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import upct.uxxiportal.catalogo.ServiciosConferencia;
import upct.uxxiportal.portlets.conferencia.ConferenciaInscripcion;

public class ConferenciaQrServlet extends Action {
    private static final Log LOG = LogFactory.getLog(ConferenciaQrServlet.class);

    /**
     * Obtenemos los parámetros necesarios para después mostrar la página
invitacion.jsp
     * @param mapping  ActionMapping
     * @param form    ActionForm
     * @param request Solicitud con los parámetros id que es el hash del QR del
invitado y codConferencia
     * @param response       Respuesta
     * @throws Exception
     * @return ActionForward mapping.findForward("irInvitacion")
     */
    @Override
       public ActionForward execute(ActionMapping mapping, ActionForm form,
                                    HttpServletRequest request,
                                    HttpServletResponse response) throws Exception {

       ServiciosConferencia serv = ServiciosConferencia.getInstance();

        String id = request.getParameter("id");
        String codConferencia = request.getParameter("codConferencia");

        LOG.debug("Recuperando valores sesión (" + id + ", " + codConferencia + ")");
        ConferenciaInscripcion inscripcion =
serv.getInscripcionByCodInvitacion(codConferencia, id);
        //Actualizamos la inscripción si existe y si no ha caducado el evento
        if(inscripcion != null && !inscripcion.getConferencia().getEventoCaducado())
{

       inscripcion.setNumLecturas(inscripcion.getNumLecturas().add(BigDecimal.ONE));
              serv.updateInscripcion(inscripcion);

              request.setAttribute("inscripcion", inscripcion);
        }

        return mapping.findForward("irInvitacion");

    }
}
```

9.11.1.23 *invitación.jsp*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
```

```
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://xmlns.oracle.com/portal/pdk/struts/tags-html" prefix="html"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<html>
    <head>
        <title>Portal de Servicios - Universidad Polit&eacute;cnica de
Cartagena</title>
        <!-- <link href="css/estilo_fuera_portal.css" rel="stylesheet"
type="text/css"></link> -->
        <style>
              .info-box, .alert, .ok, .warning {
                           -moz-border-radius: 4px;
                           -webkit-border-radius: 4px;
                           border-radius: 4px;
                           display: block;
                           margin: 1px 0;
                           padding: 5px;
                           text-align: left;
                           font-weight: bold;
                     }

                     .alert {
                           background-color: #faebeb;
                           border: 1px solid #dc7070;
                           color: #dc7070;
                     }

                     .ok {
                           background-color: #f9fde8;
                           border: 1px solid #a2bc13;
                           color: #a2bc13;
                     }

                     .warning {
                           background-color: #fbf9e9;
                           border: 1px solid #e3cf57;
                           color: #e3cf57;
                     }

                     body {
                           font-size: 12pt;
                     font-family: Verdana,Arial,Helvetica,sans-serif;
                         padding-left: 0px;
                     }
       </style>
      </head>
      <body>
             <c:choose>
                    <c:when test="${empty inscripcion}">
                           <div class="alert">
                                  SU INVITACIÓN NO ES VÁLIDA O HA CADUCADO
                           </div>
                    </c:when>
                    <c:otherwise>
                           <c:choose>
                                  <c:when test="${inscripcion.numLecturas > 1}">
                                         <c:set var="class" value="ok" />
                                  </c:when>
                                  <c:otherwise>
                                         <c:set var="class" value="ok" />
                                  </c:otherwise>
                           </c:choose>
```

```
                            <div class="${class}">
                                    SU INVITACIÓN ES VÁLIDA
                                    <br/><br/>
                                    ${inscripcion.conferencia.titulo}
                                    <br/><br/>
                                    [${inscripcion.id.nif}]
${inscripcion.persona.nombreCompleto}
                                    <br/>
                                    <span style="font-size: 10pt; color:
#D6D6D2;">Núm.Lecturas: ${inscripcion.numLecturas}</span>
                            </div>

                    </c:otherwise>
            </c:choose>
        </body>
</html>
```

9.11.2 Caso 2) Registro de dispositivos móviles de un usuario basado en QR

9.11.2.1 Inicio.java

```
/**
 * Actividad inicial de la aplicación. En ella comprueba si está registrado
 * el dispositivo en la BD.
 * @author Inmaculada García Machado
 * @version 1.0
 */
package com.qrauthtest;

import java.util.ArrayList;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import com.qrauthtest.library.Httppostaux;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.os.SystemClock;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.widget.TextView;

public class Inicio extends Activity{

        Httppostaux post;
        TelephonyManager tm;
        ProgressDialog pDialog;

        String IP_Server =  "192.168.1.101";
        String URL_connect = "http://"+IP_Server+"/androidlogin/seedevice.php";
        String Device_name;
```

```
        String Device_id;

        String nombre;
        String nif;

        TextView texto;
        Boolean conexion = true;
        /**
         * Método inicial de la actividad. Se obtiene el DEVICE_ID y el
         * DEVICE_NAME y se comprueba si está ya registrado el dispositivo
         * @param savedInstanceState
         */
        @Override
        public void onCreate(Bundle savedInstanceState) {

                super.onCreate(savedInstanceState);
                setContentView(R.layout.l_inicio);

                post = new Httppostaux();

                texto = (TextView)findViewById(R.id.text);

                //------      OBTENGO  DEVICE_ID Y EL NOMBRE      ------
                Device_id = getDeviceId();
                Device_name = getDeviceName();

                //------      SEND_ID      ------
                devexist(Device_id, Device_name);
        }



        //--------------------------------------------------------------------------
--------------------
        /**
         * Obtenemos el Device_ID (IMEI) o el número de serie si no tiene
         * IMEI (por ejemplo para tablets)
         * @return DEVICE_ID
         */
        public  String getDeviceId() {

            final String deviceId = ((TelephonyManager)
this.getSystemService(Context.TELEPHONY_SERVICE)).getDeviceId();

            if (deviceId != null) {
                System.out.println("deviceId");
                 return deviceId;
            } else {
                System.out.println("serial");
                 return android.os.Build.SERIAL;

            }
        }
        /**
         * Obtenemos el Device_name que es el modelo del dispositivo
         * @return DEVICE_NAME
         */
        public static String getDeviceName() {
                  String manufacturer = Build.MANUFACTURER;
                  String model = Build.MODEL;
                  if (model.startsWith(manufacturer)) {
                    return capitalize(model);
                  } else {
                    return capitalize(manufacturer) + " " + model;
                  }
        }
```

```
        /**
         * Capitalizamos el DEVICE_NAME
         * @param s
         * @return DEVICE_NAME
         */
        private static String capitalize(String s) {
                if (s == null || s.length() == 0) {
                  return "";
                }
                char first = s.charAt(0);
                if (Character.isUpperCase(first)) {
                   return s;
                } else {
                   return Character.toUpperCase(first) + s.substring(1);
                }
        }
        /**
         * Se comprueba si existe el dispositivo en la BD
         * @param device_id
         * @param device
         */
        public void devexist (String device_id , String device ) {

                if( device_id!= null && device != null){

                        new asyncdevice().execute(device_id, device);
                }else{
                        texto.setText("En estos momentos no es posible. Inténtelo de
nuevo más tarde");
                }

        }
        /**
         * Método para realizar el POST. Enviamos los parámetros y recibimos la
         * respuesta
         * @param device_id
         * @param device
         * @return
         */
        public Boolean envio (String device_id , String device ){
                int devstatus = -1;
                //Enviamos los parámetros al servidor
                ArrayList<NameValuePair> postparameters2send = new
ArrayList<NameValuePair>();
                postparameters2send.add(new BasicNameValuePair("device_id",device_id));
                postparameters2send.add(new BasicNameValuePair("device",device));

                JSONArray jdata = post.getserverdata(postparameters2send, URL_connect);

                SystemClock.sleep(950); //Para simular el tiempo de espera. Eliminar en
real

                if (jdata!=null && jdata.length() > 0){
                        JSONObject json_data;

                        try{
                                json_data = jdata.getJSONObject(0);
                                devstatus = json_data.getInt("devstatus");
                                //nombre = jdata.optJSONArray("NOMALU");
                                json_data = jdata.getJSONObject(1);
                                nombre = json_data.getString("LL1ALU");
                                json_data = jdata.getJSONObject(2);
                                nif = json_data.getString("NIF");

                        }catch (JSONException e){
```

```
                        e.printStackTrace();

                }

                if(devstatus == 0){
                        Log.e("devistatus ", "no registrado");
                        return false;
                }else {
                        Log.e("devistatus ", "registrado");
                        return true;
                }
        }else {
                Log.e("JSON ", "ERROR");
                Log.e("Error" , "Error en la conexión");
                conexion = false;
                return false;
        }
    }

    /**
     * Método en caso de que no se produzca la conexión con el servidor
     */
    private void fallodeconexion() {
        texto.setText("En estos momentos es imposible la conexión. Por favor,
inténtelo de nuevo más tarde.");
        SystemClock.sleep(950);
        Intent intent2 = new Intent(Intent.ACTION_MAIN);
        intent2.addCategory(Intent.CATEGORY_HOME);
        intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent2);
        finish();
    }

    /**
     * Tarea asíncrona para realizar el POST en Android
     * @author Inmaculada García
     *
     */
    class asyncdevice extends AsyncTask<String, String, String> {

        String dev_id;
        String dev;
        private volatile boolean running = true;
        /**
         * Se configura el ProgressDialgo
         */
        protected void onPreExecute() {
                pDialog = new ProgressDialog(Inicio.this);
                pDialog.setMessage("Autenticando...");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(false);
                pDialog.show();
        }

        /**
         * Enviamos el POST y esperamos la respuesta
         */
        protected String doInBackground(String... params) {
                //obtenemos el user y pass
                dev_id =params[0];
                dev =params[1];

                //enviamos, recibimos y analizamos los datos en segundo plano
                if(envio(dev_id, dev)==true){
                        return "ok"; //device registrado
                }else if(conexion == false){ // fallo de conexión
```

```
                        return "falloconexion";
                    }else{
                        return "no"; //device no registrado
                    }
                }

                /**
                 * Despues de terminar el doInBackground, dependiendo del resultado,
                 *  pasamos a otra actividad o mostramos error
                 *  @param result Resultado del POST
                 */
                protected void onPostExecute(String result) {

                    pDialog.dismiss(); //ocultamos el progress dialog.

                    if(result.equals("no")){

                        Intent i = new Intent(Inicio.this, MainActivity.class);
                        i.putExtra("device_id", dev_id);
                        i.putExtra("device", dev);
                        startActivity(i);
                        finish();

                    }else if (result.equals("falloconexion")){
                        fallodeconexion();
                    }else{
                        Intent i = new Intent(Inicio.this, Principal.class);
                        i.putExtra("nombre", nombre);
                        i.putExtra("dnialu", nif);
                        startActivity(i);
                        finish();
                    }
                }
            }
}
```

9.11.2.2 l_inicio.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin" >

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:textSize="20dp" />
</RelativeLayout>
```

9.11.2.3 MainActivity.java

```
package com.qrauthtest;

import com.google.zxing.client.android.CaptureActivity;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.app.ActionBar;
import android.support.v4.app.Fragment;
```

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.os.Build;

public class MainActivity extends Activity {

	private Button btnQR;
	TextView textview;
	String dev_id;
	String dev;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		Bundle reicieveParams = getIntent().getExtras();
		dev_id = reicieveParams.getString("device_id");
		dev = reicieveParams.getString("device");

		textview = (TextView) findViewById(R.id.text);
		textview.setText("Para entrar debes antes acceder desde un PC al Portal
de Servicios. n\nUna vez allí en el apartado \'Otros Servicios\'->\'Acceso QR\' debes
escanear el QR que aparece.");

		//Al pulsar el botón entramos en el lector
		btnQR = (Button) findViewById(R.id.btn_qr);
		btnQR.setOnClickListener(new OnClickListener() {
			@Override
			public void onClick(View v) {
				Intent intent1 = new
Intent("com.google.zxing.client.android.SCAN");
				intent1.putExtra("SCAN_MODE", "QR_CODE_MODE");
				startActivityForResult(intent1, 0);
			}
		});
	}
	//Deshabilitamos el botón de atrás de esta actividad
	@Override
	public void onBackPressed() {
		return;
	}
	//El resultado del lector lo extraemos y vamos a Registro.class para registrar
al usuario
	public void onActivityResult(int requestCode, int resultCode, Intent intent){
		if(requestCode == 0) {
			if (resultCode == RESULT_OK) {
				String capturedQRValue =
intent.getStringExtra("SCAN_RESULT");
				Intent i = new Intent(MainActivity.this, Registro.class);
				i.putExtra("contenido", capturedQRValue);
				i.putExtra("device_id", dev_id);
				i.putExtra("device", dev);
				startActivity(i);
				finish();
```

```
                        }else if ( resultCode == RESULT_CANCELED) { //Si cancela el
usuario la lectura volvemos a Inicio
                                Intent in = new Intent(MainActivity.this, Inicio.class);
                                startActivity(in);
                                finish();
                        }
                }else {
                        textview.setText("Lo sentimos. Inténtelo de nuevo más tarde");
                }
        }
}
```

9.11.2.4 activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin" >

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"/>
    <Button
        android:id="@+id/btn_qr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="28dp"
        android:background="@drawable/redondeo"
        android:text="@string/escanear"
        android:textColor="#FFF"
        android:textSize="25sp" />
</RelativeLayout>
```

9.11.2.5 Registro.java

```
/**
 * Registramos el dispositivo y lo vinculamos a un usuario
 * @author Inmaculada García Machado
 * @version 1.0
 */
package com.qrauthtest;

import java.util.ArrayList;
import java.util.StringTokenizer;
import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import com.qrauthtest.library.Httppostaux;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.SystemClock;
import android.util.Log;
```

```
import android.widget.TextView;

public class Registro extends Activity {

	TextView estado;
	String nif="";
	String usuario="";
	String hash="";

	String IP_Server =  "192.168.1.101";
	String URL_registro = "http://"+IP_Server+"/androidlogin/adduser.php";

	ProgressDialog pDialog;
	Httppostaux post;
	/**
	 * Método inicial de la actividad. Recibimos todos los datos necesarios
	 * para el registro. Se divide el string leido del QR.
	 * @param savedInstanceState
	 */
	@Override
	protected void onCreate(Bundle savedInstanceState) {

		super.onCreate(savedInstanceState);
		setContentView(R.layout.registro);

		estado = (TextView)findViewById(R.id.registro1);

		post = new Httppostaux();

		//Obtenemos todos los datos necesarios para el registro
		Bundle reicieveParams = getIntent().getExtras();
		String contents = reicieveParams.getString("contenido");
		String device_id = reicieveParams.getString("device_id");
		String device = reicieveParams.getString("device");
		//Obtenemos los datos del QR
		StringTokenizer tokens=new StringTokenizer(contents, "##");
		while (tokens.hasMoreTokens()){
			nif = tokens.nextToken();
			usuario = tokens.nextToken();
			hash = tokens.nextToken();

		}

		enviardatos(nif, usuario, device_id, device, hash);
	}
	/**
	 * Método inicial para enviar el POST. Comprueba que
	 * ningún campo está vacío.
	 * @param nif
	 * @param user
	 * @param dev_id
	 * @param dev
	 * @param hash
	 */
	public void enviardatos(String nif, String user, String dev_id, String dev,
String hash) {

		if(nif!=null && user!=null && dev_id!=null && dev!=null && hash!=null){
			new asyncregistro().execute(nif, user, dev_id, dev, hash);
		}else{
			System.out.println("Algo ha salido mal");
		}
	}

	/**
	 * Deshabilitamos el botón de atrás de esta actividad
```

```
         */
        @Override
        public void onBackPressed() {
                return;
        }

        /**
         * Método que realiza el POST
         * @param nif
         * @param user
         * @param device_id
         * @param device
         * @param hash
         * @return
         */
        public Boolean enviartodo (String nif, String user, String device_id, String
device, String hash ){
                int regstatus = -1;

                ArrayList<NameValuePair> postparameters2send = new
ArrayList<NameValuePair>();
                postparameters2send.add(new BasicNameValuePair("nif",nif));
                postparameters2send.add(new BasicNameValuePair("user",user));
                postparameters2send.add(new BasicNameValuePair("device_id",device_id));
                postparameters2send.add(new BasicNameValuePair("device",device));
                postparameters2send.add(new BasicNameValuePair("hash",hash));
                JSONArray jdata = post.getserverdata(postparameters2send,
URL_registro);

                SystemClock.sleep(950);

                if (jdata!=null && jdata.length() > 0){
                        JSONObject json_data;

                        try{
                                json_data = jdata.getJSONObject(0);
                                regstatus = json_data.getInt("regstatus");
                                System.out.println("Registrado: "+regstatus);

                        }catch (JSONException e){
                                e.printStackTrace();
                        }

                        if(regstatus == 0){
                                Log.e("regstatus ", "no registrado");
                                return false;
                        }else {
                                Log.e("regstatus ", "registrado");
                                return true;
                        }
                }else {
                        Log.e("JSON ", "ERROR");
                        return false;
                }
        }

        /**
         * Tarea asíncrona para poder enviar el POST en Android
         * @author Inma
         *
         */
class asyncregistro extends AsyncTask<String, String, String> {

                String nif;
                String user;
                String dev_id;
```

```
		String dev;
		String hash;
		/**
		 * Se configura el ProgressDialgo
		 */
		protected void onPreExecute() {
			pDialog = new ProgressDialog(Registro.this);
			pDialog.setMessage("Registrando...");
			pDialog.setIndeterminate(false);
			pDialog.setCancelable(false);
			pDialog.show();
		}
		/**
		 * Enviamos el POST y esperamos la respuesta
		 */
		protected String doInBackground(String... params) {
			//obtenemos los datos
			nif = params[0];
			user = params[1];
			dev_id =params[2];
			dev =params[3];
			hash = params[4];

			//enviamos, recibimos y analizamos los datos en segundo plano
			if(enviartodo(nif,user,dev_id, dev,hash)==true){
				return "ok"; //device registrado
			}else{
				return "no"; //device no registrado
			}
		}

		/**
		 * Despues de terminar el doInBackground, dependiendo del resultado,
		 *  pasamos a otra actividad o mostramos error
		 *  @param result Resultado del POST
		 */
		protected void onPostExecute(String result) {

			pDialog.dismiss(); //ocultamos el progress dialog.

			if(result.equals("ok")){

				estado.setText("Registrado");
				Intent i = new Intent(Registro.this, Principal.class);
				i.putExtra("nif", nif);
				i.putExtra("nombre", user);
				startActivity(i);
				finish();

			}else{
				System.out.println("Usuario no registrado");
				estado.setText("No se puede registrar. Inténtelo de nuevo
más tarde");
			}
		}
	}
}
```

9.11.2.6 registro.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:id="@+id/registro1"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="26dp"
        android:text="@string/registro" />

</RelativeLayout>
```

9.11.2.7 Principal.java

```
/**
 * Actividad de la pantalla principal
 * @author Inmaculada García
 * @version 1.0
 */
package com.qrauthtest;

import java.sql.Array;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.Button;
import android.widget.GridView;
import android.widget.TextView;

public class Principal extends Activity {

	Button expedi;
	Array expedien;
	/**
	 * Método inicial de la actividad.
	 * Mosstramos un GridView que actuará de menú
	 */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
	    setContentView(R.layout.principal);

	    Bundle reicieveParams = getIntent().getExtras();
		final String nomb = reicieveParams.getString("nombre");
		final String dnialu = reicieveParams.getString("dnialu");

		TextView txtnombre = (TextView)findViewById(R.id.textView2);
		txtnombre.setText(nomb);

		GridView gv = (GridView)findViewById(R.id.gridView1);
		gv.setAdapter(new ImageAdapter(this)); //Con el setAdapter se llena
		//el gridview con datos. En este caso con un objeto de la clase
		//ImageAdapter, que estÃ¡ definido en la clase ImageAdapter.java.
		//Para que detecte la pulsaciÃ³n se le aÃ±ade un listener de itemClick

		gv.setOnItemClickListener(new OnItemClickListener() {
			//Aqui dentro definimos la funciÃ³n que se ejecuta en el caso de
hacer click
			// en algunos de los datos. Esta funciÃ³n recibe el objeto
padre, que es un
			//adapterview en el que se ha pulsado, una vista, una posiciÃ³n,
que es la posiciÃ³n
			//del elemento dentro del adapter, y un idde la fila del item
que se ha pulsado
```

```
                        public void onItemClick(AdapterView<?> parent, View v, int
position, long id) {

                                if (position == 0){ //Asignaturas
                                        Intent intent = new Intent(Principal.this,
Expediente.class);
                                        intent.putExtra("nif", dnialu);
                                        startActivity(intent);

                                }else if(position == 1){ //Estudiante
                                        Intent i = new Intent (Principal.this,
Estudiante.class);
                                        i.putExtra("dnialu",dnialu);
                                        startActivity(i);

                                }else if (position == 3){  //Salir
                                        Intent intent2 = new Intent(Intent.ACTION_MAIN);
                                intent2.addCategory(Intent.CATEGORY_HOME);
                                intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                                startActivity(intent2);
                                finish();

                                }else{  //ConfiguraciÃ³n
                                        Intent i = new Intent(Principal.this,
Configuracion.class);
                                        i.putExtra("dnialu",dnialu);
                                        startActivity(i);
                                }
                        }
                });
        }
}
```

9.11.2.8 principal.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="#ffffff"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin" >

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="2dp"
        android:layout_toRightOf="@+id/textView1"
        android:hint="INMACULADA CONCEPCIIIIION"
        android:textSize="16dp" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/textView2"
        android:layout_alignBottom="@+id/textView2"
        android:layout_alignParentLeft="true"
        android:layout_marginLeft="2dp"
        android:text="@string/bienvenido"
        android:textSize="16dp" />
```

```
    <GridView
        android:id="@+id/gridView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView1"
        android:numColumns="auto_fit"
        android:verticalSpacing="10dp"
        android:horizontalSpacing="10dp"
        android:columnWidth="150dp"
        android:stretchMode="columnWidth"
        android:gravity="center" >
    </GridView>

</RelativeLayout>
```

9.11.2.9 ImageAdapter.java

```
/**
 * Adaptador de las imagenes del GridView
 */
package com.qrauthtest;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.GridView;

public class ImageAdapter extends BaseAdapter {

        private Context mContext;

        /**
         * El constructor necesita el contexto de la actividad
         * donde se utiliza el adapter
         */
        public ImageAdapter(Context c){
                mContext = c;
        }

        /**
         * Devuelve el n. de elementos que se introducen en el adapter
         */
        @Override
        public int getCount() {
                // TODO Auto-generated method stub
                return mThumbIds.length;
        }
        /**
         * Este debería devolver el objeto que está en esa posición del
         * adapter. En este caso no es neceario. Sólo se devuelve un objeto null
         * @param position
         */
        @Override
        public Object getItem(int position) {
                return null;
        }
        /**
         * Este método debería devolver el id de la fila del item que está en esa
         * posición del adapter. No es necesario en este caso. Devolvemos 0.
         * @param position
         */
        @Override
        public long getItemId(int position) {
```

```
                return 0;
        }

        /**
         * Creamos un nuevo ImageView para cada item referenciado por el Adapter
         * @param position
         * @param convertView
         * @param parent
         */
        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
                //Se le pasa el View en el que se ha pulsado, convertview. Si
convertview
                //es null, se instancia y configura un ImageView con las propiedades
                //deseadas para la presentaciónn de la imagen.
                //Si convertView no es null, el imageView local es inicializado con
                //este objeto view
                ImageView imageView;
                if(convertView == null) {
                        imageView = new ImageView(mContext);
                        imageView.setLayoutParams(new GridView.LayoutParams(200,200));
//Ancho y alto
                        imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
                        imageView.setPadding(10, 10, 10, 10);
                }else {
                        imageView = (ImageView) convertView;
                }

                imageView.setImageResource(mThumbIds[position]);
                return imageView;
        }

        //En un array de integer se guardan los números que son los id de todos los
        //recursos guardados en drawable. Dentro de res/drawable están todas las
        //imágenes con esos nombres.
        private Integer[] mThumbIds = {

                R.drawable.asignaturas,
                R.drawable.estudiante,
                R.drawable.opciones,
                R.drawable.salir
        };

}
```

9.11.2.10 Expediente.java

```
/**
 * Muestra los expedientes del estudiante
 * @author Inmaculada García Machado
 * @version 1.0
 */
package com.qrauthtest;

import java.util.ArrayList;
import java.util.HashMap;
import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import com.qrauthtest.library.Httppostaux;
import android.app.ListActivity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
```

```
import android.os.Bundle;
import android.os.SystemClock;
import android.util.Log;
import android.view.View;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;

public class Expediente extends ListActivity {
TextView estado;

	String nif="";
	String IP_Server =  "192.168.1.101";
	String URL_registro = "http://"+IP_Server+"/androidlogin/getexpediente.php";

	ProgressDialog pDialog;
	Httppostaux post;

	String num, plan,titu, num2, plan2,titu2;

	ArrayList<String[]> lista = new ArrayList<String[]>();
	ArrayList<String[]> lista2 = new ArrayList<String[]>();
	ArrayList<HashMap<String,String>> Expedi;
	String[] from=new String[] {"Plan","Expediente"};
	int [] to=new int[]{R.id.plan,R.id.exp};
	String exped2[] = new String[15] ;

	/**
	 * Método inicial de la actividad.
	 * Se recibe el NIF de la actividad anterior y se procede
	 * a realizar el POST para obtener los expedientes
	 */
	@Override
	protected void onCreate(Bundle savedInstanceState) {

		super.onCreate(savedInstanceState);
		setContentView(R.layout.expediente);

		post = new Httppostaux();

		Bundle reicieveParams = getIntent().getExtras();
		nif = reicieveParams.getString("nif");

		enviardatos(nif);

	}
	/**
	 * Método para obtener el objeto seleccionado de la lista. Cuando se pulsa
	 * en un expediente enviamos los datos del pulsado a la siguiente actividad
	 * Asignaturas.
	 * @param l Lista
	 * @param v Vista
	 * @param position
	 * @param id
	 */
	@Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
	    super.onListItemClick(l, v, position, id);

	    Object o = getListAdapter().getItem(position);

	     String objpul = o.toString();
	     System.out.println("El objecto pulsado es: "+ objpul);

	     switch (position){
		case 0:
```

```
                    Intent intent = new Intent(this,Asignaturas.class);
                intent.putExtra("expe",exped2[0]);
                intent.putExtra("plan", exped2[1]);
                startActivity(intent);
                break;
              case 1:
                    Intent i = new Intent(this,Asignaturas.class);
                i.putExtra("expe",exped2[3]);
                i.putExtra("plan", exped2[4]);
                startActivity(i);
                break;
              case 2:
                    Intent inte = new Intent(this,Asignaturas.class);
                inte.putExtra("expe",exped2[6]);
                inte.putExtra("plan", exped2[7]);
                startActivity(inte);
                break;
         }
    }

       /**
        * Método que crea la lista con los datos recibidos del POST usando
        * SimpleAdapter
        */
       public void crearlista(){

              Expedi = new ArrayList<HashMap<String,String>>();
        for(String[] expedi:lista){
              HashMap<String,String> datosExpe = new HashMap<String, String>();

              datosExpe.put("Plan",expedi[0]);
              datosExpe.put("Expediente", expedi[1]);
              Expedi.add(datosExpe);
        }

        SimpleAdapter ListadoAdapter = new SimpleAdapter(this, Expedi,
R.layout.simple_list, from, to);
        setListAdapter(ListadoAdapter);
       }

       /**
        * Método que se inicia para realizar el POST. Si el nif no está
        * vacío procedemos a realizar la tare asíncrona que realiza
        * el POST.
        * @param nif
        */
       public void enviardatos(String nif){

              if(nif!=null ){
                    new asyncregistro().execute(nif);
              }else{
                    System.out.println("Algo ha salido mal");
              }
       }

       /**
        * Se hace el POST enviando el NIF del usuario. Recibimos los expedientes
        * y planes del usuario.
        * @param nif
        * @return True si todo ha ido bien. False si algo ha fallado.
        */
       public boolean enviartodo (String nif){
              int regstatus = -1;

              ArrayList<NameValuePair> postparameters2send = new
ArrayList<NameValuePair>();
```

```
		postparameters2send.add(new BasicNameValuePair("dnialu",nif));
		JSONArray jdata = post.getserverdata(postparameters2send,
URL_registro);

		SystemClock.sleep(950);

		if (jdata!=null && jdata.length() > 0){
			JSONObject json_data;
			try{
				int i;
				for( i=0; i<jdata.length();i++){ //recorremos el
resultado
					int j ;
					j= i%3; //dividimos los datos entre 3 para
distinguir entre planes
					switch(j){
						case 0:
							json_data = jdata.getJSONObject(i);
							num =
json_data.getString("NUMORD");
							exped2[i]=num; //Guardamos el
expediente y los demÃ¡s datos
							//por separado para luego poder
acceder a ellos mÃ¡s fÃ¡cilmente
							//para enviarlos a la siguiente
actividad. No para la ListView
							break;
						case 1:
							json_data = jdata.getJSONObject(i);
							plan =
json_data.getString("PLA_CODALF");
							exped2[i]= new String(plan);
//Guardamos el plan despuÃ©s
							break;
						case 2:
							json_data = jdata.getJSONObject(i);
							titu = json_data.getString("TITU");
							exped2[i]= new String(titu);
//Guardamos el nombre del plan
							//En este array guardamos los datos
tal y como queremos que se
							//muestren en la lista
							String[] exped1= {plan+" -
"+titu,"Expediente: "+num};
							 lista.add(exped1);
							break;
					}
				}
				regstatus = 1;

			}catch (JSONException e){
				e.printStackTrace();
			}
			if(regstatus == 0){
				Log.e("regstatus ", "No hay expediente");
				return false;
			}else {
				Log.e("regstatus ", "Todo correcto");
				return true;
			}
		}else {
			Log.e("JSON ", "ERROR");
			return false;
		}
	}
```

```
	/**
	 * Tarea asíncrona que realiza el POST
	 * @author Inmaculada García
	 *
	 */
	class asyncregistro extends AsyncTask<String, String, String> {

		String nif;
		/**
		 * Se configura el ProgressDialgo y se muestra
		 */
		protected void onPreExecute() {
			pDialog = new ProgressDialog(Expediente.this);
			pDialog.setMessage("Cargando...");
			pDialog.setIndeterminate(false);
			pDialog.setCancelable(false);
			pDialog.show();
		}
		/**
		 * Enviamos el POST y esperamos la respuesta
		 */
		protected String doInBackground(String... params) {
			//obtenemos el nif
			nif = params[0];

			//enviamos, recibimos y analizamos los datos en segundo plano
			if(enviartodo(nif)==true){
				return "ok"; //datos recibidos
			}else{
				return "no"; //datos no recibidos
			}
		}

		/**
		 * Despues de terminar el doInBackground, dependiendo del resultado,
		 *  creamos la lista o mostramos un mensaje
		 *  @param result Resultado del POST
		 */
		protected void onPostExecute(String result) {

			pDialog.dismiss(); //ocultamos el progress dialog.

			if(result.equals("ok")){
				crearlista();
			}else{
				System.out.println("Usuario sin expedientes");
			}
		}
	}
}
```

9.11.2.11 expediente.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
	android:orientation="vertical"
	android:layout_width="match_parent"
	android:layout_height="match_parent"
	android:layout_marginTop="10dp">
	<ListView
		android:id="@android:id/list"
		android:layout_width="match_parent"
		android:layout_height="match_parent"
		android:drawSelectorOnTop="false" />
	<TextView
		android:id="@android:id/empty"
```

```
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="No tienes expediente." />

</LinearLayout>
```

9.11.2.12 simple_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="100dp">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingRight="10px"
        android:paddingLeft="10dp">
        <TextView android:id="@+id/plan"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18dp"
            android:padding="3dp" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingLeft="10dp"
        android:paddingBottom="5dp">
        <TextView android:id="@+id/exp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="14dp" />
    </LinearLayout>
</LinearLayout>
```

9.11.2.13 Asignaturas.java

```
package com.qrauthtest;

import java.util.ArrayList;
import java.util.HashMap;
import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import com.qrauthtest.library.Httppostaux;
import android.app.ListActivity;
import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.SystemClock;
import android.util.Log;
import android.view.View;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.Toast;
/**
 * Clase para obtener las asignaturas de un expediente y mostrarlas
 * en una lista
 * @author Inma
 * @version 1.0
 *
 */
public class Asignaturas extends ListActivity {
```

```
	ProgressDialog pDialog;
	Httppostaux post;
	String nif;
	String plan;//="1021";
	String expe;//="1005";

	String IP_Server =  "192.168.1.101";
	String URL_asignatura =
"http://"+IP_Server+"/androidlogin/getasignaturas.php";

	String asig, any, cur, qua, nom;

	ArrayList<String[]> asslist = new ArrayList<String[]>();
	ArrayList<HashMap<String,String>> Asign;
	String[] from=new String[] {"Asignatura","Datos"};
	int [] to=new int[]{R.id.asignatura,R.id.asignatura2};
	String[] exped2;
	/**
	 * Método inicial de la actividad.
	 * Se reciben el plan y el numero de expediente y se procede
	 * a realizar el POST para obtener las asignaturas y sus datos
	 */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.asignaturas);

		post = new Httppostaux();

		Bundle reicieveParams = getIntent().getExtras();
		plan = reicieveParams.getString("plan");
		expe = reicieveParams.getString("expe");

		enviardatos(plan,expe);//, usuario, device_id, device, hash);
	}
	/**
	 * Método para obtener el objeto seleccionado de la lista. Cuando se pulsa
	 * en una asignatura no hacemos nada.
	 * @param l Lista
	 * @param v Vista
	 * @param position
	 * @param id
	 */
	@Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
	    super.onListItemClick(l, v, position, id);

	    Object o = getListAdapter().getItem(position);

	     Toast.makeText(this, "Selección: " + Integer.toString(position)
			+  " - " + o.toString(),Toast.LENGTH_LONG).show();
    }

	/**
	 * Método que crea la lista con los datos recibidos del POST usando
	 * SimpleAdapter
	 */
	public void crearlista(){

	 Asign = new ArrayList<HashMap<String,String>>();
	 for(String[] expedi:asslist){
		HashMap<String,String> datosExpe = new HashMap<String, String>();

		datosExpe.put("Asignatura", expedi[0]);
		datosExpe.put("Datos",expedi[1]);
```

```
                Asign.add(datosExpe);
        }

        SimpleAdapter ListadoAdapter = new SimpleAdapter(this, Asign,
R.layout.simple_list_asignatura, from, to);
        setListAdapter(ListadoAdapter);

       }

       /**
        * Método que se inicia para realizar el POST. Si el plan o el expediente
        * no están vacíos procedemos a realizar la tarea asíncrona que realiza
        * el POST
        * @param plan
        * @param expe
        */
       public void enviardatos(String plan, String expe){ //, String user, String
dev_id, String dev, String hash) {

               if(plan!=null && expe!=null ){////&& user!=null && dev_id!=null &&
dev!=null && hash!=null){
                       new asyncregistro().execute(plan,expe); // , user, dev_id, dev,
hash);
               }else{
                       System.out.println("Algo ha salido mal");
               }
       }

       /**
        * Se hace el POST enviando el PLAN y el EXPEDIENTE. Recibimos las
        * asignaturas con sus datos como el nombre, el año académico de la matricula
        * de esa asignatura, el curso y la nota.
        * @param plan
        * @param expe
        * @return  True si todo ha ido bien. False si algo ha fallado.
        */
       public boolean enviartodo (String plan, String expe){
               int regstatus = 0;

               ArrayList<NameValuePair> postparameters2send = new
ArrayList<NameValuePair>();
               postparameters2send.add(new BasicNameValuePair("plan",plan));
               postparameters2send.add(new BasicNameValuePair("expe",expe));
               JSONArray jdata = post.getserverdata(postparameters2send,
URL_asignatura);

               SystemClock.sleep(950);

               if (jdata!=null && jdata.length() > 0){
                       JSONObject json_data;

                       try{
                               int i;
                               for( i=0; i<jdata.length();i++){
                                       int j ;
                                       j= i%5;
                                       switch(j){
                                               case 0:
                                                       json_data = jdata.getJSONObject(i);
                                                       asig =
json_data.getString("ASS_CODNUM");

                                                       break;
                                               case 1:
                                                       json_data = jdata.getJSONObject(i);
```

```
                                                any =
json_data.getString("ANY_ANYACA");
                                                break;
                                        case 2:
                                                json_data = jdata.getJSONObject(i);
                                                cur =
json_data.getString("CUR_NUMCUR");
                                                break;

                                        case 3:
                                                json_data = jdata.getJSONObject(i);
                                                qua =
json_data.getString("QUANUM");
                                                break;
                                        case 4:
                                                json_data = jdata.getJSONObject(i);
                                                nom =
json_data.getString("ASS_NOM");

                                                String[] asig2 = {"Asignatura:
"+asig+" - "+nom , "C. Académico: "+any+"  Curso: "+cur+" Nota: "+qua};
                                                asslist.add(asig2);

                                                break;
                                        }
                                }
                                regstatus = 1;

                        }catch (JSONException e){
                                e.printStackTrace();
                        }

                        if(regstatus == 0){
                                Log.e("regstatus ", "Algo ha salido mal");
                                return false;
                        }else {
                                Log.e("regstatus ", "Correcto");
                                return true;
                        }
                }else {
                        Log.e("JSON ", "ERROR");
                        return false;
                }
        }

        /**
         * Tarea asíncrona que realiza el POST
         * @author Inmaculada García
         *
         */
        class asyncregistro extends AsyncTask<String, String, String> {

                String plan;
                String expe;;
                /**
                 * Se configura el ProgressDialgo y se muestra
                 */
                protected void onPreExecute() {
                        pDialog = new ProgressDialog(Asignaturas.this);
                        pDialog.setMessage("Cargando...");
                        pDialog.setIndeterminate(false);
                        pDialog.setCancelable(false);
                        pDialog.show();
                }
                /**
                 * Enviamos el POST y esperamos la respuesta
```

```
		 */
		protected String doInBackground(String... params) {
			//obtenemos el user y pass
			plan = params[0];
			expe = params[1];

			//enviamos, recibimos y analizamos los datos en segundo plano
			if(enviartodo(plan,expe)==true){  //,user,dev_id,
dev,hash)==true){
				return "ok"; //device registrado
			}else{
				return "no"; //device no registrado
			}
		}

		/**
		 * Después de terminar el doInBackground, dependiendo del resultado,
		 * creamos la lista
		 * @param result Resultado del POST
		 */
		protected void onPostExecute(String result) {

			pDialog.dismiss(); //ocultamos el progress dialog.

			if(result.equals("ok")){
				crearlista();
			}else{
				System.out.println("Sin asignaturas");
			}
		}
	}
}
```

9.11.2.14 asignaturas.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
	android:orientation="vertical"
	android:layout_width="match_parent"
	android:layout_height="match_parent"
	android:layout_marginTop="10dp">
	<ListView
		android:id="@android:id/list"
		android:layout_width="match_parent"
		android:layout_height="match_parent"
		android:drawSelectorOnTop="false" />
	<TextView
		android:id="@android:id/empty"
		android:layout_width="match_parent"
		android:layout_height="match_parent"
		android:text="No tienes asignaturas." />

</LinearLayout>
```

9.11.2.15 simple_list_asignatura.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="100dp">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
```

```
        android:paddingRight="10dp"
        android:paddingLeft="10dp">
        <TextView
            android:id="@+id/asignatura"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="3dp"
            android:hint="MAtematicas"
            android:textSize="18sp" />

        </LinearLayout>
        <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingLeft="10dp"
        android:paddingBottom="5dp">
        <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:id="@+id/asignatura2"
                        android:textSize="14sp"
                        android:padding="3dp"
                        android:hint="C.Academico: 2012-13"/>
    </LinearLayout>
</LinearLayout>
```

9.11.2.16 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.qrauthtest"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="9"
        android:targetSdkVersion="18" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.qrauthtest.Inicio"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.qrauthtest.MainActivity"
            android:label="@string/app_name">
        </activity>
        <activity
           android:name="com.google.zxing.client.android.CaptureActivity"
           android:configChanges="orientation|keyboardHidden"
           android:screenOrientation="landscape"
           android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
           android:windowSoftInputMode="stateAlwaysHidden" >
```

```
            <intent-filter>
              <action android:name="android.intent.action.MAIN" />
              <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
            <intent-filter>
              <action android:name="com.google.zxing.client.android.SCAN" />
              <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.qrauthtest.Registro" ></activity>
        <activity
            android:name="com.qrauthtest.Principal"
            android:label="@string/upct"
            android:screenOrientation="portrait" >
        </activity>
        <activity
            android:name="com.qrauthtest.Estudiante"
            android:label="@string/datospersonales"
            android:screenOrientation="portrait" >
        </activity>
        <activity
            android:name="com.qrauthtest.Expediente"
            android:label="Plan"
            android:screenOrientation="portrait">
        </activity>
        <activity
            android:name="com.qrauthtest.Asignaturas"
            android:label="Asignaturas"
            android:screenOrientation="portrait">
        </activity>
        <activity
            android:name="com.qrauthtest.Configuracion"
            android:label="Configuración"
            android:screenOrientation="portrait" >
        </activity>
    </application>

</manifest>
```

9.11.2.17 config.php

```
<?php
/**
 * Database config variables
 */
define("DB_HOST", "127.0.0.1"); //hace referencia a donde se encuentra la BD
define("DB_USER", "root");//nombre de usuario definido en la configuracion de la BD.
define("DB_PASSWORD", "");//password elegido
define("DB_DATABASE", "android_login");//Nombre de la base de datos
?>
```

9.11.2.18 connectbd.php

```
<?php

class DB_Connect {

    // constructor
    function __construct() {
    }
    // destructor
    function __destruct() {
        // $this->close();
```

```
    }
    // Connecting to database
    public function connect() {
        require_once 'config.php';
        // connecting to mysql
        $con = mysql_connect(DB_HOST, DB_USER, DB_PASSWORD);
        // selecting database
        mysql_select_db(DB_DATABASE);
        // return database handler
        return $con;
    }
    // Closing database connection
    public function close() {
        mysql_close();
    }
}
?>
```

9.11.2.19 funciones_bd.php

```
<?php

class funciones_BD {

    private $db;

    // constructor

    function __construct() {
        require_once 'connectbd.php';
        // connecting to database

        $this->db = new DB_Connect();
        $this->db->connect();

    }

    // destructor
    function __destruct() {

    }

    /**-------------------------------------------------------------------------
     * agregar nuevo usuario
     */
    public function adduser($nif,$user,$device_id,$device,$hash) {

    $result = mysql_query ("INSERT INTO usuarios (username ,device_id ,device,
DNIALU, HASH) VALUES ('$user', '$device_id' ,'$device', '$nif', '$hash')");

       // check for successful store
        if ($result) {
            return true;

        } else {
            return false;
        }
    }

    /**-------------------------------------------------------------------------
     * eliminar dispositivo de un usuario
     */
    public function deleuser($nif,$device_id,$device) {

    $result = mysql_query("DELETE FROM `usuarios`
```

```
                        WHERE DNIALU='$nif'
                        AND device_id='$device_id'
                        AND device='$device'");
        // check for successful delete
        if ($result) {
            return true;

        } else {
            return false;
        }
    }


     /**-------------------------------------------------------------------------
     * Verificar si el usuario ya existe por el nif
     */

    public function isuserexist($username) {

        $result = mysql_query("SELECT DNIALU
                               FROM usuarios
                               WHERE DNIALU = '$username'");

        $num_rows = mysql_num_rows($result); //numero de filas retornadas

        if ($num_rows > 0) {
            // el usuario existe
            return true;
        } else {
            // no existe
            return false;
        }
    }

    /**-------------------------------------------------------------------------
     * Verificar si el dispositivo ya está registrado
     */
    public function isdeviceexist($device_id , $device) {

        $result = mysql_query("SELECT device_id
                               FROM usuarios
                               WHERE device = '$device'
                               AND device_id='$device_id'");

        $num_rows = mysql_num_rows($result); //numero de filas retornadas

        if ($num_rows > 0) {
            // el dispositivo ya está registrado
            return true;
        } else {
            // no está registrado el dispositivo
            return false;
        }
              mysql_free_result($result);
    }

    /**-------------------------------------------------------------------------
     * Funcion para obtener el nombre y el NIF del usuario de un dispositivo
     */
   public function datospersonal($device_id , $device) {

        $resu = mysql_query("SELECT a.NOMALU AS NOM, a.DNIALU AS DNI
                             FROM alu_alumne a, usuarios u
                             WHERE device = '$device'
                             AND device_id='$device_id'
                             AND u.DNIALU=a.DNIALU");
```

```
      while ($fila = mysql_fetch_array($resu)) {

              $tempo1 =$fila["NOM"];
              $tempo = utf8_encode($tempo1);
              $result []= array("NOMALU"=>"$tempo");

              $aux1 =$fila["DNI"];
              $aux = utf8_encode($aux1);
              $result []= array("NIF"=>"$aux");
      }
      return ($result);
      mysql_free_result($resu);
   }

 /**-------------------------------------------------------------------------
  * Funcion para obtener los planes y los números de expediente del alumno
  */

 public function verexpedientes($dnialu){

      $consulta = mysql_query("SELECT NUMORD, PLA_CODALF, TITU
                               FROM `alu_expedient`
                               WHERE ALU_DNIALU='$dnialu'");

      while ($fila = mysql_fetch_array($consulta)) {

              $tempo1 =$fila["NUMORD"];
              $tempo = utf8_encode($tempo1);
              $result []= array("NUMORD"=>"$tempo");

              $aux1 =$fila["PLA_CODALF"];
              $aux = utf8_encode($aux1);
              $result []= array("PLA_CODALF"=>"$aux");

              $aux2 = $fila["TITU"];
              $aux3 = utf8_encode($aux2);
              $result []=array("TITU"=>"$aux3");

      }
      return ($result);
      mysql_free_result($consulta);

 }

/**-------------------------------------------------------------------------
 * Funcion para obtener los datos personales del alumno
 */
 public function datosestudiante($dnialu) {

      $resu = mysql_query("SELECT NOMALU, LL1ALU, LL2ALU, DNIALU, DATNAI
                           FROM alu_alumne
                           WHERE DNIALU = $dnialu");

      while ($fila = mysql_fetch_array($resu)) {

              $tempo1 =$fila["NOMALU"];
              $tempo = utf8_encode($tempo1);
              $result []= array("NOMALU"=>"$tempo");

              $aux1 =$fila["LL1ALU"];
              $aux = utf8_encode($aux1);
              $result []= array("LL1ALU"=>"$aux");

              $aux1 =$fila["LL2ALU"];
              $aux = utf8_encode($aux1);
```

```
                $result []= array("LL2ALU"=>"$aux");

                $aux1 =$fila["DNIALU"];
                $aux = utf8_encode($aux1);
                $result []= array("NIF"=>"$aux");

                $aux1 =$fila["DATNAI"];
                $aux = utf8_encode($aux1);
                $result []= array("DATNAI"=>"$aux");
        }
        return ($result);
        mysql_free_result($resu);
    }

 /**--------------------------------------------------------------------------
  * Funcion para obtener las asignaturas del expediente seleccionado del alumno
  */
   public function verasignaturas($plan, $expe){

        $selec = mysql_query("SELECT DISTINCT l.ASS_CODNUM AS ASS_CODNUM, l.ANY_ANYACA
AS    ANY_ANYACA,    l.CUR_NUMCUR    AS    CUR_NUMCUR,    l.ASS_NOM    AS    ASS_NOM,
SUBSTRING(a.QUANUM,1,3) AS QUANUM
                            FROM `alu_linmatricula` l , `alu_linacta` a
                            WHERE l.PLA_CODALF='$plan'
                            AND l.EXP_NUMORD='$expe'
                            AND l.PLA_CODALF=a.PLA_CODALF
                            AND l.EXP_NUMORD=a.EXP_NUMORD
                            AND l.ANY_ANYACA=a.ANY_ANYACA
                            AND l.ASS_CODNUM=a.ASS_CODNUM
                            ORDER BY l.ANY_ANYACA DESC, a.QUANUM DESC");

        while ($fila = mysql_fetch_array($selec)) {
                $tempo1 =$fila["ASS_CODNUM"];
                $tempo = utf8_encode($tempo1);
                $result []= array("ASS_CODNUM"=>"$tempo");

                $aux1 =$fila["ANY_ANYACA"];
                $aux = utf8_encode($aux1);
                $result []= array("ANY_ANYACA"=>"$aux");

                $aux1 =$fila["CUR_NUMCUR"];
                $aux = utf8_encode($aux1);
                $result []= array("CUR_NUMCUR"=>"$aux");

                $aux1 =$fila["QUANUM"];
                $aux = utf8_encode($aux1);
                $result []= array("QUANUM"=>"$aux");

                $aux1 =$fila["ASS_NOM"];
                $aux = utf8_encode($aux1);
                $result []= array("ASS_NOM"=>"$aux");
        }
        return ($result);
        mysql_free_result($selec);
    }
}
?>
```

9.11.2.20 adduser.php

```
<?php

$nif = $_POST['nif'];
$user = $_POST['user'];
$device_id = $_POST['device_id'];
```

```
$device = $_POST['device'];
$hash = $_POST['hash'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

if($db->adduser($nif,$user,$device_id,$device,$hash)){
	$resultado[]=array("regstatus"=>"1");
}else{
	$resultado[]=array("regstatus"=>"0");
}
echo json_encode($resultado);

?>
```

9.11.2.21 deleteuser.php

```
<?php
$nif = $_POST['nif'];
$device_id = $_POST['device_id'];
$device = $_POST['device'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

if($db->deleuser($nif,$device_id,$device)){
	$resultado[]=array("regstatus"=>"1");
}else{
	$resultado[]=array("regstatus"=>"0");
}

echo json_encode($resultado);
?>
```

9.11.2.22 seedevice.php

```
<?php
/*Busca el dispositivo en la base de datos*/
$devi_id = $_POST['device_id'];
$devi = $_POST['device'];

require_once 'funciones_bd.php';
$db = new funciones_BD();
	/*Comprobamos que existe este dispositivo en la bd*/
	if($db->isdeviceexist($devi_id, $devi) ){

		$resultado[]=array("devstatus"=>"1");
		/* Obtenemos el nombre del alumno y su NIF */
		$result=($db->datospersonal($devi_id , $devi));

		$nom = $result[0];
		$nif = $result[1];

		$resultado[]=$nom;
		$resultado[]=$nif;

	}else{
		$resultado[]=array("devstatus"=>"0");
	}

echo json_encode($resultado);
?>
```

9.11.2.23 getexpedientes.php

```
<?php
```

```
/*Buscamos la información personal del estudiante*/
$dnialu = $_POST['dnialu'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

	if($db->isuserexist($dnialu)){
		$expe=($db->verexpedientes($dnialu));
		if($expe!=null){
			$cuenta = count($expe);
			for ($i = 0; $i< $cuenta;$i++){
				$j = $i % 3;
				switch($j){
					case 0:
						$num = $expe[$i];
						$resultado[]=$num;
						break;
					case 1:
						$pla = $expe[$i];
						$resultado[]=$pla;
						break;
					case 2:
						$titu = $expe[$i];
						$resultado[]=$titu;
						break;
				}
			}
		}
	}else{
		$resultado[]="";
	}
echo json_encode($resultado);
 ?>
```

9.11.2.24 getasignaturas.php

```
<?php
/*Buscamos la información personal del estudiante*/
$plan = $_POST['plan'];
$expe = $_POST['expe'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

	$asig=($db->verasignaturas($plan, $expe));

	if($asig!=null){

		$cuenta = count($asig);
		for ($i = 0; $i< $cuenta;$i++){
			$j = $i % 5;
			switch($j){
				case 0:
					$ass = $asig[$i];
					$resultado[]=$ass;
					break;
				case 1:
					$any = $asig[$i];
					$resultado[]=$any;
					break;
				case 2:
					$cur = $asig[$i];
					$resultado[]=$cur;
					break;
				case 3:
					$qua = $asig[$i];
```

```
                                        $resultado[]=$qua;
                                        break;
                                case 4:
                                        $nom = $asig[$i];
                                        $resultado[]=$nom;
                                        break;
                        }
                }
        }
echo json_encode($resultado);
 ?>
```

9.11.2.25 getpersonal.php

```
<?php

/*Buscamos la información personal del estudiante*/
$dni = $_POST['dnialu'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

        if($db->isuserexist($dni)){

                $datos=($db->datosestudiante($dni));

                if($datos!=null){
                        $nom = $datos[0];
                        $ape1 = $datos[1];
                        $ape2 = $datos[2];
                        $nif = $datos[3];
                        $datna = $datos[4];

                        $resultado[]=$nom;
                        $resultado[]=$ape1;
                        $resultado[]=$ape2;
                        $resultado[]=$nif;
                        $resultado[]=$datna;
                }
        }else{
                $resultado[]="";
        }
echo json_encode($resultado);
```

9.11.3 Caso 3) Aplicación para pagar en los establecimientos universitarios a través de una cuenta virtual

9.11.3.1 Autorización.java

```
package com.comedorcv;

import java.util.ArrayList;

import library.Httppostaux;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
```

```
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.SystemClock;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

/**
 * Clase donde el personal que utilice la aplicación se autenticará para poder
usarla.
 * Introducirá su DNI y su contraseña y la app se conectará con la BBDD para saber
 * si esa persona está autorizada. Además del DNI y de la contraseña (la cual será
enviada
 * encriptada) la app también enviará el ID del dispositivo porque cada usuario debe
estar
 * asociado a uno o varios dispositivos.
 * @author Inmaculada García Machado
 * @version 1
 *
 */

public class Autorizacion extends Activity {

	private EditText etnif;
	private EditText etpass;

	ProgressDialog pDialog;
	Httppostaux post;

	/*
	 * Dirección IP del servidor que usarán todas las clases que lo necesiten y
dirección
	 * del archivo php que autentica al personal
	 */
	static String IP_Server = "192.168.1.101";
	String URL_registro = "http://"+IP_Server+"/monedero_virtual/autenticar.php";

	int contador = 3;
	static String deviceid;
	String espacio = "";

	/**
	 * Método que se llama en la creación de la actividad.
	 * Se inicializan e identifican todos los campos y parámetros para
	 * realizar la autenticación.
	 */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.autorizacion);

		etnif = (EditText)findViewById(R.id.eTusuario);
	    etpass = (EditText)findViewById(R.id.etpassword);
		Button b_acceso = (Button)findViewById(R.id.btnacceder);

		post = new Httppostaux();

		b_acceso.setOnClickListener(new Button.OnClickListener(){

			/**
			 * Si se pulsa el botón de la actividad se obtienen los
carácteres
```

```
                         * del usuario y contraseña, se obtiene el ID del dispositivo,
se
                         * encripta la contraseña y se envían los datos al servidor para
                         * saber si ese usuario está autorizado y con ese dispositivo
                         */
                        public void onClick(View arg0) {

                                String nif = etnif.getText().toString();
                                String pass = etpass.getText().toString();
                                deviceid = getDeviceId();

                                //La contraseña se encripta mediante MD5 en la clase MD5
                                String hash = MD5.md5(pass);

                                autenticar(nif, hash, deviceid);
                        }
                });
        }

        //----------------------------------------------------------------------------
-------------------
        /**
         * Obtenemos el Device_ID (IMEI) o el número de serie si no tiene
         * IMEI (por ejemplo para tablets)
         * @return DEVICE_ID
         */
        public  String getDeviceId() {

            final String deviceId = ((TelephonyManager)
this.getSystemService(Context.TELEPHONY_SERVICE)).getDeviceId();

            if (deviceId != null) {
                System.out.println("deviceId");
                 return deviceId;
            } else {
                System.out.println("serial");
                 return android.os.Build.SERIAL;

            }
        }

        /**
         * Deshabilitamos el botón de atrás con este método porque no queremos
         * que haga nada si se pulsa el botón de atrás
         */
        @Override
        public void onBackPressed() {
                return;
        }

        /**
         * Antes de enviar los datos al servidor comprobamos que no son nulos.
         * @param nif
         * @param pass
         * @param deviceid
         */
        public void autenticar(String nif, String pass, String deviceid){

                if(nif!=null && pass!=null && deviceid!=null){
                        new asyncautent().execute(nif, pass, deviceid);
                }else{
                        Toast toast3 = Toast.makeText(getApplicationContext(), "El
usuario y/o la contraseña están vacíos", Toast.LENGTH_LONG);
                        toast3.show();
                }
        }
```

```
	/**
	 * Método para realizar el POST. Enviamos los parámetros y recibimos la
respuesta
	 * @param nif
	 * @param pass
	 * @param deviceid
	 * @return
	 */
	public boolean enviartodo (String nif, String pass, String deviceid){
		int regstatus = -1;

		System.out.println("nif "+nif+" pass "+ pass+" deviceid "+deviceid);

		ArrayList<NameValuePair> postparameters2send = new
ArrayList<NameValuePair>();
		postparameters2send.add(new BasicNameValuePair("nif",nif));
		postparameters2send.add(new BasicNameValuePair("pass", pass));
		postparameters2send.add(new BasicNameValuePair("deviceid",deviceid));

		JSONArray jdata = post.getserverdata(postparameters2send,
URL_registro);

		SystemClock.sleep(950); //simula el tiempo de conexión. Eliminar en
real

		if (jdata!=null && jdata.length() > 0){
			JSONObject json_data;

			try{
				json_data = jdata.getJSONObject(0);
				regstatus = json_data.getInt("regstatus");
				System.out.println("Transacción: "+regstatus);

				json_data = jdata.getJSONObject(1);
				espacio = json_data.getString("espacio");
				System.out.println("El espacio es "+espacio);

			}catch (JSONException e){
				e.printStackTrace();
			}


			if(regstatus == 0){
				Log.e("regstatus ", "no transaccion");

				return false;
			}else {
				Log.e("regstatus ", "Transacción exitosa");


				return true;
			}
		}else {
			Log.e("JSON ", "ERROR");
			return false;
		}
	}



	/**
	 * Tarea/clase asíncrona para realizar el POST en Android.
	 * @author Inmaculada García Machado
	 *
	 */
```

```
	class asyncautent extends AsyncTask<String, String, String> {

		String nif, pass, deviceid;
		/**
		 * Método antes de la ejecución del post donde se inserta una
		 * barra de progreso donde indica que espere.
		 */
		protected void onPreExecute() {
			pDialog = new ProgressDialog(Autorizacion.this);
			pDialog.setMessage("Por favor, espere.");
			pDialog.setIndeterminate(false);
			pDialog.setCancelable(false);
			pDialog.show();
		}
		/**
		 * Enviamos el POST y esperamos la respuesta
		 */
		protected String doInBackground(String... params) {
			//Obtenemos los datos a enviar
			nif = params[0];
			pass = params[1];
			deviceid =params[2];

			//enviamos, recibimos y analizamos los datos en segundo plano
			if(enviartodo(nif, pass, deviceid)==true){
				return "ok"; //operacion exitosa
			}else{
				return "no"; //operacion fallida
			}
		}

		/**
		 * Después de terminar el doInBackground, dependiendo del resultado,
		 *  pasamos al menú principal o mostramos error de login (después de
		 *  3 intentos en caso de error salimos de la aplicación)
		 */
		protected void onPostExecute(String result) {

			pDialog.dismiss(); //ocultamos el progress dialog

			if(result.equals("ok")){

				/*
				 * Vamos al menú principal y enviamos el NIF y el espacio
donde está el
				 * dispositivo registrado porque se utilizarán en otras
clases.
				 */
				Intent i = new Intent(Autorizacion.this,
MainActivity.class);
				i.putExtra("nif_autorizado", nif);
				i.putExtra("espacio", espacio);
				startActivity(i);
				finish();

			}else{

				contador = contador -1;

				if (contador == 0){
					Toast toast2 =
Toast.makeText(getApplicationContext(), "Lo sentimos no es posible la
identificación", Toast.LENGTH_LONG);
					toast2.show();
					Intent intent3 = new Intent(Intent.ACTION_MAIN);
				intent3.addCategory(Intent.CATEGORY_HOME);
```

```
                            intent3.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                            startActivity(intent3);
                            finish();
                            }else if(contador == 1){
                                    Toast toast =
Toast.makeText(getApplicationContext(), "Usuario y contraseña incorrectos. Le queda
"+contador+" intento.", Toast.LENGTH_LONG);
                                    toast.show();
                            }else{
                                    Toast toast =
Toast.makeText(getApplicationContext(), "Usuario y contraseña incorrectos. Le quedan
"+contador+" intentos.", Toast.LENGTH_LONG);
                                    toast.show();
                            }

                    }
            }
        }
}
```

9.11.3.2 autorizacion.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >


    <TextView
        android:id="@+id/tVusuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="@string/usuario" />

    <EditText
        android:id="@+id/eTusuario"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:ems="10" >

        <requestFocus />
    </EditText>

    <TextView
        android:id="@+id/tVpassword"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="@string/contrasena" />

    <EditText
        android:id="@+id/etpassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:ems="10"
        android:inputType="textPassword" />

    <Button
```

```
        android:id="@+id/btnacceder"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/redondeo2"
        android:textColor="#ffffff"
           android:textSize="20sp"
           android:typeface="sans"
           android:layout_margin="5dp"
        android:text="@string/acceder" />

</LinearLayout>
```

9.11.3.3 MainActivity.java

```
package com.comedorcv;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.GridView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.TextView;

/**
 * Actividad principal que muestra el menú principal de la app. En ella se dan las
 * opciones de comprar, devolver, ver el historial y salir.
 * @author Inmaculada García Machado
 * @version 1
 *
 */
public class MainActivity extends Activity {

	String user, espacio;
	String importe, tipo_oper;   //Vacías

	String FILENAME = "hola_archivo";
	TextView tV3;
	static final int READ_BLOCK_SIZE = 100;

	/**
	 * Método que se llama en la creación de la actividad.
	 * Se inicializan e identifican todos los campos y parámetros.
	 */
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		Bundle reicieveParams = getIntent().getExtras();
		user = reicieveParams.getString("nif_autorizado");
		espacio = reicieveParams.getString("espacio");
```

```
		TextView tV2 = (TextView)findViewById(R.id.textView2);
		tV2.setText(" "+user);

		tV3 = (TextView)findViewById(R.id.textView3);
		tV3.setText("Lugar: "+espacio);

		GridView gv = (GridView)findViewById(R.id.gridView1);
		gv.setAdapter(new ImageAdapter(this));

		gv.setOnItemClickListener(new OnItemClickListener() {

			/**
			 * Se ha creado un gridview donde dependiendo de donde se pulse iremos a una
			 * actividad o a otra.
			 */
			public void onItemClick(AdapterView<?> parent, View v, int position, long id) {

				if (position == 0){   //Compra
					Intent intent = new Intent(MainActivity.this, Compra.class);
					intent.putExtra("user", user);
					startActivity(intent);
					finish();

				}else if(position == 1){     //Devolución
					Intent i = new Intent (MainActivity.this, Devolucion.class);
					i.putExtra("user",user);
					startActivity(i);
					finish();
				}else if (position == 2){    //Ver el historial
					tipo_oper = "3";
					Intent in = new Intent (MainActivity.this, LectorQR.class);
					in.putExtra("user", user);
					in.putExtra("importe", importe);
					in.putExtra("tipo_oper", tipo_oper);
					startActivity(in);
					finish();
				}else if (position == 3){    //Salir
					Intent intent2 = new Intent(Intent.ACTION_MAIN);
				intent2.addCategory(Intent.CATEGORY_HOME);
				intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
				startActivity(intent2);
				finish();
				}else{

				}
			}
		});
	}

	/**
	 * Deshabilitamos el botón de atrás con este método porque no hace nada si se pulsa el botón de atrás
	 */
	@Override
	public void onBackPressed() {
		return;
	}

	/**
	 * Al salir de la actividad, creamos un fichero en el que guardamos el espacio de la persona autorizada
```

```
        * Esta forma de guardar el valor de un parámetro es el más fiable en Android.
Se hace así porque no
        * se necesita el valor de esta variable en otras actividades y queremos
seguir teniendola si se vuelve
        * al menú en la misma sesión.
        */
       @Override
       protected void onPause() {

              FileOutputStream fos;
              try {
                     fos = openFileOutput(FILENAME, Context.MODE_WORLD_WRITEABLE);
                     fos.write(espacio.getBytes());
                     fos.close();
              } catch (FileNotFoundException e) {
                     // TODO Auto-generated catch block
                     e.printStackTrace();
              } catch (IOException e) {
                     // TODO Auto-generated catch block
                     e.printStackTrace();
              }
              super.onPause();
       }
       /**
        * Cuando regresemos a esta actividad, recuperamos su estado (el modo anterior
que había, online u offline)
        */
       @Override
       protected void onResume() {

              System.out.println("dentro de on Resume");
              if (espacio == null || espacio.isEmpty()){
                     System.out.println("dentro de on Resume si espacio = null");

                     try {
                            FileInputStream fis;
                            fis = openFileInput(FILENAME);
                            InputStreamReader isr = new InputStreamReader(fis);

                            char[] inputBuffer = new char[READ_BLOCK_SIZE];
                            String temp="";
                            int c;
                            while((c = isr.read(inputBuffer))>0){
                                   String readString =
String.copyValueOf(inputBuffer, 0, c);
                                   temp += readString;

                                   inputBuffer = new char[READ_BLOCK_SIZE];
                            }

                            fis.close();
                            espacio = temp;
                            tV3.setText("Lugar: "+espacio);

                     } catch (FileNotFoundException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                     } catch (IOException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                     }
              }
              super.onResume();

       }
```

```
	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the action bar if it is
present.
		getMenuInflater().inflate(R.menu.main, menu);
		return true;
	}

	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		// Handle action bar item clicks here. The action bar will
		// automatically handle clicks on the Home/Up button, so long
		// as you specify a parent activity in AndroidManifest.xml.
		int id = item.getItemId();
		if (id == R.id.action_settings) {
			Intent intent3 = new Intent(Intent.ACTION_MAIN);
		intent3.addCategory(Intent.CATEGORY_HOME);
		intent3.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
		startActivity(intent3);
		finish();
		}
		return super.onOptionsItemSelected(item);
	}
}
```

9.11.3.4 activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="#ffffff"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.comedorcv.MainActivity" >

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="2dp"
        android:layout_toRightOf="@+id/textView1"
        android:textSize="16dp" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/textView2"
        android:layout_alignBottom="@+id/textView2"
        android:layout_alignParentLeft="true"
        android:layout_marginLeft="2dp"
        android:text="@string/bienvenido"
        android:textSize="16dp" />

    <TextView
        android:id="@+id/textView3"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView2"
        android:hint="CANTINA X"
        android:textSize="16dp" />

    <GridView
        android:id="@+id/gridView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView3"
        android:layout_marginTop="20dp"
        android:numColumns="auto_fit"
        android:verticalSpacing="20dp"
        android:horizontalSpacing="1dp"
        android:columnWidth="142dp"
        android:stretchMode="columnWidth"
        android:gravity="center" >
    </GridView>



</RelativeLayout>
```

9.11.3.5 ImageAdapter.java

```
package com.comedorcv;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;

public class ImageAdapter extends BaseAdapter {

        private Context mContext;

        //Se ponen las imágenes en un array
        public Integer[] mThumbIds = {
                        R.drawable.comprar, R.drawable.caja,
                R.drawable.historial, R.drawable.salir
            };

        // Constructor
    public ImageAdapter(Context c){
        mContext = c;
    }

        @Override
        public int getCount() {
                return mThumbIds.length;

        }

        @Override
        public Object getItem(int position) {
                 return mThumbIds[position];
        }

        @Override
```

```
	public long getItemId(int position) {
		return 0;
	}

	@Override
	public View getView(int position, View convertView, ViewGroup parent) {
		  ImageView imageView = new ImageView(mContext);
		imageView.setImageResource(mThumbIds[position]);
		imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
		imageView.setLayoutParams(new GridView.LayoutParams(140, 140));
		return imageView;
	}

}
```

9.11.3.6 Compra.java

```
package com.comedorcv;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;

/**
 * Esta actividad es específica sólo para cuando se va a realizar una compra.
 * Se introduce la cantidad a cobrar y se pulsa el botón que llevará al
 * lector del QR de la TUI para pagar.
 *
 * @author Inmaculada García Machado
 * @version 1.0
 */
public class Compra extends Activity {

	String user;
	String importe;
	String tipo_oper;

	private EditText cantidad;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.compra);

		Bundle reicieveParams = getIntent().getExtras();
		user = reicieveParams.getString("user");

		cantidad = (EditText)findViewById(R.id.cantidad);

		tipo_oper = "1"; //Tipo de operación=1 => Compra

		ImageButton b_pagar = (ImageButton)findViewById(R.id.botonpagar);
		b_pagar.setOnClickListener(new ImageButton.OnClickListener(){

			public void onClick(View arg0) {

				importe = cantidad.getText().toString(); //obtenemos la
cantidad a cobrar

				Intent i = new Intent(Compra.this, LectorQR.class);
```

```
                        i.putExtra("user", user);
                        i.putExtra("importe", importe);
                        i.putExtra("tipo_oper", tipo_oper);
                        startActivity(i);
                        finish();
                    }
                });
        }

        @Override
        public void onBackPressed() {
                Intent i = new Intent(Compra.this, MainActivity.class);
                i.putExtra("nif_autorizado", user);
                startActivity(i);
                finish();
        }

}
```

9.11.3.7 compra.xml

```
<?xml version="1.0" encoding="utf-8"?><RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

     <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:textSize="20sp"
        android:layout_marginTop="5dp"
        android:layout_marginLeft="5dp"
        android:text="@string/cantidad" />

     <EditText
        android:id="@+id/cantidad"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="38dp"
        android:layout_marginLeft="5dp"
        android:textSize="25sp"
        android:ems="5"
        android:hint="0.00"
        android:inputType="numberDecimal" />

    <ImageButton
        android:id="@+id/botonpagar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/cantidad"
        android:layout_marginTop="30dp"
        android:src="@drawable/pagar" />

</RelativeLayout>
```

9.11.3.8 Devolucion.java

```
package com.comedorcv;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class Devolucion extends Activity {

	String user;
	String importe;
	String tipo_oper;

	private EditText cantdevol;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.devolucion);

		Bundle reicieveParams = getIntent().getExtras();
		user = reicieveParams.getString("user");

		cantdevol = (EditText)findViewById(R.id.eTdevolver);

		tipo_oper = "2";

		Button btndevolver = (Button)findViewById(R.id.btndevolver);
		btndevolver.setOnClickListener(new Button.OnClickListener(){

			public void onClick(View arg0) {
				importe = cantdevol.getText().toString();

				Intent i = new Intent(Devolucion.this, LectorQR.class);
				i.putExtra("user", user);
				i.putExtra("importe", importe);
				i.putExtra("tipo_oper", tipo_oper);
				startActivity(i);
				finish();
			}
		});
	}

	@Override
	public void onBackPressed() {
		Intent i = new Intent(Devolucion.this, MainActivity.class);
		i.putExtra("nif_autorizado", user);
		startActivity(i);
		finish();
	}

}
```

9.11.3.9 devolucion.xml

```
<?xml version="1.0" encoding="utf-8"?><RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:orientation="vertical" >

     <TextView
        android:id="@+id/tVdevolver"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:textSize="20sp"
        android:layout_marginTop="5dp"
        android:layout_marginLeft="5dp"
        android:text="@string/cantidaddevolver" />

     <EditText
         android:id="@+id/eTdevolver"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignParentLeft="true"
         android:layout_below="@+id/tVdevolver"
         android:layout_marginTop="17dp"
         android:layout_marginLeft="5dp"
         android:textSize="25sp"
         android:ems="5"
         android:hint="0.00"
         android:inputType="numberDecimal" />

     <Button
         android:id="@+id/btndevolver"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:layout_centerHorizontal="true"
         android:layout_below="@+id/eTdevolver"
         android:layout_marginTop="22dp"
         android:layout_marginLeft="5dp"
         android:layout_marginRight="5dp"
         android:background="@drawable/redondeo2"
         android:textColor="#ffffff"
            android:textSize="20sp"
            android:typeface="sans"
            android:padding="10dp"
         android:text="@string/devolver" />

</RelativeLayout>
```

9.11.3.10 LectorQR.java

```
package com.comedorcv;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.zxing.client.android.CaptureActivity;


public class LectorQR extends Activity {
```

```
	private Button btnQR;
	TextView textview;
	String user, importe, tipo_oper;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.lectorqr);

		Bundle reicieveParams = getIntent().getExtras();
		user = reicieveParams.getString("user");
		importe = reicieveParams.getString("importe");
		tipo_oper = reicieveParams.getString("tipo_oper");


		btnQR = (Button) findViewById(R.id.btn_qr);
		btnQR.setOnClickListener(new OnClickListener() {
			@Override
			public void onClick(View v) {
				Intent intent1 = new
Intent("com.google.zxing.client.android.SCAN");
				intent1.putExtra("SCAN_MODE", "QR_CODE_MODE");

				startActivityForResult(intent1, 0);
			}
		});
	}

	//Deshabilitamos el botón de atrás de esta actividad
	@Override
	public void onBackPressed() {
		return;
	}

	public void onActivityResult(int requestCode, int resultCode, Intent intent) {
		if(requestCode == 0) {
			if (resultCode == RESULT_OK) {
				String capturedQRValue =
intent.getStringExtra("SCAN_RESULT");

				System.out.println("LEido "+ capturedQRValue);

				Intent i = new Intent (LectorQR.this, Tarjetapin.class);
				i.putExtra("datosqr", capturedQRValue.toString());
				i.putExtra("user", user);
				i.putExtra("importe", importe);
				i.putExtra("tipo_oper", tipo_oper);
				startActivity(i);
				finish();

			}else if ( resultCode == RESULT_CANCELED) {

				Toast toast = Toast.makeText(getApplicationContext(), "Ha
ocurrido un error. Inténtelo de nuevo.", Toast.LENGTH_LONG);
				toast.show();

				Intent in = new Intent(LectorQR.this,
MainActivity.class);
				in.putExtra("user", user);
				startActivity(in);
				finish();
			}
		}else {
			textview.setText("Lo sentimos. Inténtelo de nuevo más tarde");
		}
	}
```

```
}
```

9.11.3.11 lectorqr.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:id="@+id/btn_qr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="37dp"
        android:background="@drawable/redondeo2"
        android:text="@string/escanear"
        android:textColor="#ffffff"
        android:textSize="25sp"
        android:padding="30dp"
        android:typeface="sans" />

</RelativeLayout>
```

9.11.3.12 Tarjetapin.java

```
package com.comedorcv;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.StringTokenizer;

import library.Httppostaux;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import com.comedorcv.Autorizacion.asyncautent;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.SystemClock;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class Tarjetapin extends Activity {
```

```
	String contents, nif_autorizado, importe, tipo_operacion, codtarjeta;
	int contador = 3;

	EditText pintarjeta;
	TextView importepago, infopago;

	String URL_registro =
"http://"+Autorizacion.IP_Server+"/monedero_virtual/validarpin.php";
	ProgressDialog pDialog;
	Httppostaux post;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.tarjetapin);

		Bundle reicieveParams = getIntent().getExtras();
		contents = reicieveParams.getString("datosqr");
		nif_autorizado = reicieveParams.getString("user");
		importe = reicieveParams.getString("importe");
		tipo_operacion = reicieveParams.getString("tipo_oper");
		System.out.println("tipo_operacion: "+ tipo_operacion);

		post = new Httppostaux();

		infopago = (TextView)findViewById(R.id.tVinfopago);
		if ("1".equals(tipo_operacion)){
			infopago.setText("Se va a realizar un pago de:");

			importepago = (TextView)findViewById(R.id.tVimportepago);
			importepago.setText(importe + " €");
		} else if("2".equals(tipo_operacion)){
			infopago.setText("Se va a realizar una devolución de:");

			importepago = (TextView)findViewById(R.id.tVimportepago);
			importepago.setText(importe + " €");
		} else {
			infopago.setText("Se va a extraer el historial de su tarjeta.");
		}

		Codigotarjeta cotarjeta = new Codigotarjeta();
		codtarjeta = cotarjeta.codigo(contents);

		pintarjeta = (EditText)findViewById(R.id.eTtarjetapin);

		Button btntarjetapin = (Button)findViewById(R.id.btntarjetapin);
		btntarjetapin.setOnClickListener(new Button.OnClickListener(){

			public void onClick(View arg0) {

				String pin = pintarjeta.getText().toString();
				MD5 cript = new MD5();
				String hash = cript.md5(pin);

				validar(codtarjeta,hash);

				System.out.println("Hash: "+hash);


			}
		});
	}



	public void validar(String codtarjeta, String hash){
```

```
		if(codtarjeta!=null && hash!=null){
			new asyncautent().execute(codtarjeta, hash);
		}else{
			System.out.println("Algo ha salido mal");
		}

	}

	/**
	 *
	 * @param nif
	 * @param pass
	 * @return
	 */
	public boolean enviartodo (String codtarjeta, String hash){
		int regstatus = -1;

		System.out.println("nif "+codtarjeta+" pass "+hash);

		ArrayList<NameValuePair> postparameters2send = new
ArrayList<NameValuePair>();
		postparameters2send.add(new
BasicNameValuePair("codtarjeta",codtarjeta));
		postparameters2send.add(new BasicNameValuePair("hash", hash));

		JSONArray jdata = post.getserverdata(postparameters2send,
URL_registro);

		SystemClock.sleep(950); //simula el tiempo de conexión. Eliminar en
real

		if (jdata!=null && jdata.length() > 0){
			JSONObject json_data;

			try{
				json_data = jdata.getJSONObject(0);
				regstatus = json_data.getInt("regstatus");
				System.out.println("Transacción: "+regstatus);


			}catch (JSONException e){
				e.printStackTrace();
			}


			if(regstatus == 0){
				Log.e("regstatus ", "no transaccion");

				return false;
			}else {
				Log.e("regstatus ", "Transacción exitosa");


				return true;
			}
		}else {
			Log.e("JSON ", "ERROR");
			return false;
		}
	}



	/**
	 *
```

```
         * @author Inma
         *
         */
        class asyncautent extends AsyncTask<String, String, String> {

                String codtarjeta, hash;


                protected void onPreExecute() {
                        pDialog = new ProgressDialog(Tarjetapin.this);
                        pDialog.setMessage("Por favor, espere.");
                        pDialog.setIndeterminate(false);
                        pDialog.setCancelable(false);
                        pDialog.show();
                }

                protected String doInBackground(String... params) {
                        //obtenemos los datos a enviar
                        codtarjeta = params[0];
                        hash = params[1];

                        //enviamos, recibimos y analizamos los datos en segundo plano
                        if(enviartodo(codtarjeta, hash)==true){
                                return "ok"; //operacion exitosa
                        }else{
                                return "no"; //operacion fallida
                        }
                }

                /*Despues de terminar el doInBackground, dependiendo del resultado,
                 *  pasamos al menú principal o mostramos error de login (después de
                 *  3 intentos salimos de la aplicación)
                 */
                protected void onPostExecute(String result) {

                        //pDialog.dismiss(); //ocultamos el progress dialog.
                        Log.e("onPostExecute=",""+result);

                        if(result.equals("ok")){
                                pDialog.dismiss(); //ocultamos el progress dialog.
                                System.out.println("Autenticacion correcta");

                                if("3".equals(tipo_operacion)){
                                        Intent in = new Intent(Tarjetapin.this,
Historial.class);

                                        in.putExtra("codtarjeta", codtarjeta);
                                        in.putExtra("user", nif_autorizado);
                                        startActivity(in);
                                        finish();

                                }else{
                                        Intent i = new Intent(Tarjetapin.this,
Pago.class);

                                        i.putExtra("codtarjeta", codtarjeta);
                                        i.putExtra("user", nif_autorizado);
                                        i.putExtra("importe", importe);
                                        i.putExtra("tipo_oper", tipo_operacion);
                                        startActivity(i);
                                        finish();
                                }

                        }else{

                                pDialog.dismiss(); //ocultamos el progress dialog.

                                contador = contador -1;
```

```
                            if (contador == 0){
                                    Toast toast2 =
Toast.makeText(getApplicationContext(), "PIN incorrecto. Inténtelo de nuevo más
tarde.", Toast.LENGTH_LONG);
                                    toast2.show();
                                    Intent intent3 = new Intent(Tarjetapin.this,
MainActivity.class);
                            intent3.putExtra("nif_autorizado", nif_autorizado);
                            startActivity(intent3);
                            finish();
                            }else if(contador == 1){
                                    Toast toast =
Toast.makeText(getApplicationContext(), "PIN incorrecto. Le queda "+contador+"
intento.", Toast.LENGTH_LONG);
                                    toast.show();
                            }else{
                                    Toast toast =
Toast.makeText(getApplicationContext(), "PIN incorrecto. Le quedan "+contador+"
intentos.", Toast.LENGTH_LONG);
                                    toast.show();
                            }

                    }
            }
      }


}
```

9.11.3.13 tarjetapin.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/tVinfopago"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_marginTop="2dp" />

    <TextView
        android:id="@+id/tVimportepago"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="25sp"  />

    <TextView
        android:id="@+id/tVtarjetapin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="@string/introduce" />

    <EditText
        android:id="@+id/eTtarjetapin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
```

```
        android:ems="10"
        android:inputType="numberPassword" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/btntarjetapin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/redondeo2"
        android:textColor="#ffffff"
           android:textSize="20sp"
           android:typeface="sans"
           android:layout_margin="5dp"
        android:text="@string/enviar" />

</LinearLayout>
```

9.11.3.14 Pago.java

```
package com.comedorcv;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.StringTokenizer;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import library.Httppostaux;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.SystemClock;
import android.telephony.TelephonyManager;
import android.text.format.Time;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Button;

public class Pago extends Activity {

       Button btnnuevacompra;
       Button btnprincipal;
       WebView myWebView;

       String id_operacion, cod_tarjeta, nif_autorizado, tipo_operacion, importe,
cod_concepto, terminal;
```

```
	String URL_registro =
"http://"+Autorizacion.IP_Server+"/monedero_virtual/pagar.php";
	Httppostaux post;
	ProgressDialog pDialog;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.pago);

		post = new Httppostaux();

		int contador = 1;

		btnnuevacompra = (Button)findViewById(R.id.btnnuevacompra);
		btnprincipal = (Button)findViewById(R.id.btnprincipal);
		myWebView = (WebView) this.findViewById(R.id.myWebView);
		myWebView.setWebViewClient(new WebViewClient());

		//Obtenemos los parámetros recibidos
		Bundle reicieveParams = getIntent().getExtras();
		cod_tarjeta = reicieveParams.getString("codtarjeta");
		nif_autorizado = reicieveParams.getString("user");
		importe = reicieveParams.getString("importe");
		tipo_operacion = reicieveParams.getString("tipo_oper");

		id_operacion = obtenerIdOperacion();

		cod_concepto="1";

		enviardatos(id_operacion, cod_tarjeta, nif_autorizado, tipo_operacion,
importe, cod_concepto, terminal);

		btnprincipal.setOnClickListener(new OnClickListener() {

			public void onClick(View v){
				Intent in = new Intent(Pago.this, MainActivity.class);
				in.putExtra("nif_autorizado", nif_autorizado);
				startActivity(in);
				finish();
			}
		});

		btnnuevacompra.setOnClickListener(new OnClickListener() {

			public void onClick(View v){
				Intent inte = new Intent(Pago.this, Compra.class);
				inte.putExtra("user", nif_autorizado);
				startActivity(inte);
				finish();
			}
		});

	}

	//-------------------------------------------------------------------------
-------------------

	public String obtenerIdOperacion(){

		SimpleDateFormat tiempo = new SimpleDateFormat("yyyyMMddHHmmss");
		String currentDateandTime = tiempo.format(new Date());
		System.out.println("Tiempo: "+currentDateandTime);

		terminal = Autorizacion.deviceid;
```

```
		String idoperacion = currentDateandTime+terminal;

		return idoperacion;

	}

	public void enviardatos(String id_operacion, String cod_tarjeta, String
nif_autorizado, String tipo_operacion, String importe, String cod_concepto, String
terminal) {

		if(id_operacion!=null && cod_tarjeta!=null && nif_autorizado!=null &&
tipo_operacion!=null && importe!=null && terminal!=null ){
			new asyncllamada().execute(id_operacion, cod_tarjeta,
nif_autorizado, tipo_operacion, importe, cod_concepto, terminal);
		}else{
			System.out.println("Algo ha salido mal");
		}

	}

	/**
	 *
	 * @param nif
	 * @param user
	 * @param device_id
	 * @param device
	 * @param hash
	 * @return
	 */
	public boolean enviartodo (String id_operacion, String cod_tarjeta, String
nif_autorizado, String tipo_operacion, String importe, String cod_concepto, String
terminal){
		int regstatus = -1;
		System.out.println("Va bien");
		ArrayList<NameValuePair> postparameters2send = new
ArrayList<NameValuePair>();
		postparameters2send.add(new
BasicNameValuePair("id_operacion",id_operacion));
		postparameters2send.add(new
BasicNameValuePair("cod_tarjeta",cod_tarjeta));
		postparameters2send.add(new
BasicNameValuePair("nif_autorizado",nif_autorizado));
		postparameters2send.add(new
BasicNameValuePair("tipo_operacion",tipo_operacion));
		postparameters2send.add(new BasicNameValuePair("importe",importe));
		postparameters2send.add(new
BasicNameValuePair("cod_concepto",cod_concepto));
		postparameters2send.add(new BasicNameValuePair("terminal",terminal));

		JSONArray jdata = post.getserverdata(postparameters2send,
URL_registro);

		//SystemClock.sleep(950);

		if (jdata!=null && jdata.length() > 0){
			JSONObject json_data;

			try{
				json_data = jdata.getJSONObject(0);
				regstatus = json_data.getInt("regstatus");
				System.out.println("Transacción: "+regstatus);

			}catch (JSONException e){
				e.printStackTrace();
			}
```

```
                        if(regstatus == 0){
                                Log.e("regstatus ", "no transaccion");

                                return false;
                        }else {
                                Log.e("regstatus ", "Transacción exitosa");


                                return true;
                        }
                }else {
                        Log.e("JSON ", "ERROR");
                        return false;
                }
        }

        /**
         *
         * @author Inma
         *
         */
        class asyncllamada extends AsyncTask<String, String, String> {

                String id_operacion, cod_tarjeta, nif_autorizado, tipo_operacion,
importe, cod_concepto, terminal;


                protected void onPreExecute() {
                        pDialog = new ProgressDialog(Pago.this);
                        pDialog.setMessage("Enviando... Por favor, espere.");
                        pDialog.setIndeterminate(false);
                        pDialog.setCancelable(false);
                        pDialog.show();
                }

                protected String doInBackground(String... params) {
                        //obtenemos los datos a enviar
                        id_operacion = params[0];
                        cod_tarjeta = params[1];
                        nif_autorizado =params[2];
                        tipo_operacion = params[3];
                        importe = params[4];
                        cod_concepto = params[5];
                        terminal = params[6];

                        //enviamos, recibimos y analizamos los datos en segundo plano
                        if(enviartodo(id_operacion, cod_tarjeta, nif_autorizado,
tipo_operacion, importe, cod_concepto, terminal)==true){
                                return "ok"; //operacion exitosa
                        }else{
                                System.out.println("Algo ha salido mal 2");
                                return "no"; //operacion fallida
                        }
                }

                /*Despues de terminar el doInBackground, dependiendo del resultado,
                 *  pasamos a otra actividad o mostramos error
                 */
                protected void onPostExecute(String result) {

                        //pDialog.dismiss(); //ocultamos el progress dialog.
                        Log.e("onPostExecute=",""+result);

                        if(result.equals("ok")){
                                pDialog.dismiss(); //ocultamos el progress dialog.
```

```
                                System.out.println("Despues de pagar. Mensaje de ok");


        myWebView.loadUrl("file:///android_asset/transaccionok.html");

                                myWebView.setWebViewClient(new WebViewClient() {
                                        //Si hay algún error o no hay conexión a internet,
se muestra una página de error.
                                        @Override
                                        public void onReceivedError(WebView view, int
errorCode, String description, String failingUrl) {

        myWebView.loadUrl("file:///android_asset/errorpage.html");
                                        }
                                });

                        }else{

                                System.out.println("Transacción fallida");
                                pDialog.dismiss(); //ocultamos el progress dialog.

        myWebView.loadUrl("file:///android_asset/transaccionno.html");

                                myWebView.setWebViewClient(new WebViewClient() {
                                        //Si hay algún error o no hay conexión a internet,
se muestra una página de error.
                                        @Override
                                        public void onReceivedError(WebView view, int
errorCode, String description, String failingUrl) {

        myWebView.loadUrl("file:///android_asset/errorpage.html");
                                        }
                                });
                        }
                }
        }
}
```

9.11.3.15 pago.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <WebView
        android:id="@+id/myWebView"
        android:layout_width="match_parent"
        android:layout_height="256dp" />

    <Button
        android:id="@+id/btnnuevacompra"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0.88"
        android:background="@drawable/redondeo2"
        android:text="Nueva compra"
        android:layout_margin="5dp"
        android:textColor="#ffffff"
        android:textSize="20sp"
        android:typeface="sans" />

    <Button
```

```
        android:id="@+id/btnprincipal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/redondeo2"
        android:textColor="#ffffff"
           android:textSize="20sp"
           android:typeface="sans"
           android:layout_marginLeft="5dp"
           android:layout_marginRight="5dp"
           android:layout_marginBottom="5dp"
        android:text="@string/menu" />

</LinearLayout>
```

9.11.3.16 Historial.java

```
package com.comedorcv;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.StringTokenizer;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import library.Httppostaux;
import android.app.ListActivity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.SystemClock;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.Toast;

public class Historial extends ListActivity {

        String datosqr, nif_autorizado, terminal;
        int contador = 1;
        String codtarjeta ="";

        ProgressDialog pDialog;
        Httppostaux post;
        String URL_registro =
"http://"+Autorizacion.IP_Server+"/monedero_virtual/historial.php";

        String id_oper, cod_tar, nif_aut, tipo_operacio, importe, cod_concepto,
terminal2, pin;

        ArrayList<String[]> histlist = new ArrayList<String[]>();
        ArrayList<HashMap<String,String>> Historialn;
        String[] from=new String[]{"Historial"};
```

```
        int [] to = new int[]{R.id.linea};

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.historial);

                post = new Httppostaux();

                //Obtenemos los parámetros recibidos (resultado de la lectura del QR
también)
                Bundle reicieveParams = getIntent().getExtras();
                codtarjeta = reicieveParams.getString("codtarjeta");
                nif_autorizado = reicieveParams.getString("user");

                terminal = Autorizacion.deviceid;
                System.out.println("Terminal " +terminal);
         obtenerdatos(codtarjeta, terminal);

        }

        @Override
        public void onBackPressed() {
                Intent i = new Intent(Historial.this, MainActivity.class);
                i.putExtra("nif_autorizado", nif_autorizado);
                startActivity(i);
                finish();
        }

        //--------------------------------------------------------------------------
-------------------
        /**
         * Obtenemos el Device_ID (IMEI) o el número de serie si no tiene
         * IMEI (por ejemplo para tablets)
         * @return DEVICE_ID
         */
                public  String getDeviceId() {

                    final String deviceId = ((TelephonyManager)
this.getSystemService(Context.TELEPHONY_SERVICE)).getDeviceId();

                    if (deviceId != null) {
                       System.out.println("deviceId");
                 return deviceId;
            } else {
                System.out.println("serial");
                 return android.os.Build.SERIAL;

            }
        }
        //--------------------------------------------------------------------------
-------------------

        /**
         * Método para obtener el objeto seleccionado de la lista. Cuando se pulsa
         * en algún item no hacemos nada.
         * @param l Lista
         * @param v Vista
         * @param position
         * @param id
         */
        @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
         super.onListItemClick(l, v, position, id);

         Object o = getListAdapter().getItem(position);
```

```
        //Para futuros usos

    }

       public void crearlista(){

              Historialn = new ArrayList<HashMap<String,String>>();
        for(String[] histo:histlist){
              HashMap<String,String> datosHistorial = new HashMap<String, String>();

              datosHistorial.put("Historial", histo[0]);


              Historialn.add(datosHistorial);

        }

        SimpleAdapter ListadoAdapter = new SimpleAdapter(this, Historialn,
R.layout.simple_list, from, to);
        setListAdapter(ListadoAdapter);

       }

       /**
        * Método que se inicia para realizar el POST. Si el cod_tarjeta y el
nif_autorizado
        * no están vacíos procedemos a realizar la tarea asíncrona que realiza
        * el POST
        * @param codtarjeta
        * @param terminal
        */
       public void obtenerdatos(String codtarjeta, String terminal){

              if(codtarjeta!=null && terminal!=null){
                     new asyncregistro().execute(codtarjeta,terminal);
              }else{
                     System.out.println("Algo ha salido mal");
              }
       }

       /**
        * Se hace el POST enviando el PLAN y el EXPEDIENTE. Recibimos las
        * asignaturas con sus datos como el nombre, el año académico de la matricula
        * de esa asignatura, el curso y la nota.
        * @param plan
        * @param expe
        * @return  True si todo ha ido bien. False si algo ha fallado.
        */
       public boolean enviartodo (String codtar, String terminal){
              int regstatus = 0;

              ArrayList<NameValuePair> postparameters2send = new
ArrayList<NameValuePair>();
              postparameters2send.add(new BasicNameValuePair("codtar",codtar));
              postparameters2send.add(new BasicNameValuePair("terminal",terminal));


              JSONArray jdata = post.getserverdata(postparameters2send,
URL_registro);
              System.out.println("3");
              SystemClock.sleep(950);

              if (jdata!=null && jdata.length() > 0){
                     JSONObject json_data;
                     String ano,mes,dia, hora, minu;
                     String fecha = "";
```

```
                    try{
                            int i;
                            for( i=0; i<jdata.length();i++){
                                    int j ;
                                    j= i%8;
                                    switch(j){
                                            case 0:
                                                    json_data = jdata.getJSONObject(i);
                                                    id_oper =
json_data.getString("id_operacion");

                                                    ano = id_oper.substring(0, 4);
                                                    mes = id_oper.substring(4, 6);
                                                    dia = id_oper.substring(6, 8);
                                                    hora = id_oper.substring(8, 10);
                                                    minu = id_oper.substring(10, 12);
                                                    fecha = dia+"/"+mes+"/"+ano+" -
"+hora+":"+minu;
                                                    break;
                                            case 1:
                                                    json_data = jdata.getJSONObject(i);
                                                    cod_tar =
json_data.getString("cod_tarjeta");
                                                    break;
                                            case 2:
                                                    json_data = jdata.getJSONObject(i);
                                                    nif_aut =
json_data.getString("nif_autorizado");
                                                    break;
                                            case 3:
                                                    json_data = jdata.getJSONObject(i);
                                                    tipo_operacio =
json_data.getString("tipo_operacion");
                                                    break;
                                            case 4:
                                                    json_data = jdata.getJSONObject(i);
                                                    importe =
json_data.getString("importe");
                                                    break;
                                            case 5:
                                                    json_data = jdata.getJSONObject(i);
                                                    cod_concepto =
json_data.getString("cod_concepto");
                                                    break;
                                            case 6:
                                                    json_data = jdata.getJSONObject(i);
                                                    terminal2 =
json_data.getString("terminal");
                                                    break;
                                            case 7:
                                                    json_data = jdata.getJSONObject(i);
                                                    pin = json_data.getString("pin");

                                                    String[] histo2 = {"Operación de
tipo: "+tipo_operacio+", de un importe de "+importe+" €. Fecha: "+fecha};
                                                    histlist.add(histo2);
                                                    break;
                                    }
                            }
                            regstatus = 1;

                    }catch (JSONException e){
                            System.out.println("Error en el try");
                            e.printStackTrace();
                    }
```

```
                if(regstatus == 0){
                        Log.e("regstatus ", "Algo ha salido mal");

                        return false;
                }else {
                        Log.e("regstatus ", "Correcto");


                        return true;
                }
        }else {
                Log.e("JSON ", "ERROR");
                return false;
        }
}

/**
 * Tarea asíncrona que realiza el POST
 * @author Inmaculada García
 *
 */
class asyncregistro extends AsyncTask<String, String, String> {

        String codtar, nifaut;

        /**
         * Se configura el ProgressDialgo y se muestra
         */
        protected void onPreExecute() {
                pDialog = new ProgressDialog(Historial.this);
                pDialog.setMessage("Cargando...");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(false);
                pDialog.show();
        }
        /**
         * Enviamos el POST y esperamos la respuesta
         */
        protected String doInBackground(String... params) {
                //obtenemos el cod_tarjeta y el nif_autorizado
                codtar = params[0];
                nifaut = params[1];


                //enviamos, recibimos y analizamos los datos en segundo plano
                if(enviartodo(codtar,nifaut)==true){
                        return "ok";
                }else{
                        return "no";
                }
        }

        /**
         * Después de terminar el doInBackground, dependiendo del resultado,
         * creamos la lista
         * @param result Resultado del POST
         */
        protected void onPostExecute(String result) {

                pDialog.dismiss(); //ocultamos el progress dialog.

                if(result.equals("ok")){

                        crearlista();
```

```
                    }else{

                    }
                }
            }
}
```

9.11.3.17 historial.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/linea"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="3dp"
        android:hint="Transacciones realizadas en la cuenta virtual:"
        android:textSize="18sp"/>

    <ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>

</LinearLayout>
```

9.11.3.18 simple_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingRight="10dp"
        android:paddingLeft="10dp"
        >

        <TextView
            android:id="@+id/linea"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="3dp"
            android:hint="Linea de historial"
            android:textSize="18sp" />

    </LinearLayout>

</LinearLayout>
```

9.11.3.19 MD5.java

```
package com.comedorcv;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class MD5 {

	public static String md5(String pin) {

		MessageDigest m = null;

		try{

			m = MessageDigest.getInstance("MD5");

		}catch(NoSuchAlgorithmException e){
			e.printStackTrace();
		}

		m.update(pin.getBytes(),0,pin.length());
		String hash = new BigInteger(1, m.digest()).toString(16);
		return hash;
	}
}
```

9.11.3.20 string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Cantina App</string>
    <string name="nombre">Cantina App</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="acceder">Acceder</string>
    <string name="acceso">Acceso:</string>
    <string name="bienvenido">Bienvenido </string>
    <string name="contrasena">Contraseña:</string>
    <string name="usuario">Usuario:</string>
    <string name="cantidad">Importe a cobrar:</string>
    <string name="cantidaddevolver">Importe a devolver:</string>
    <string name="escanear">Escanear QR</string>
    <string name="devolver">Devolver</string>
    <string name="devolucion">Devolución</string>
    <string name="salir">Salir</string>
    <string name="introduce">Introduzca la contraseña de su monedero
virtual:</string>
    <string name="enviar">Enviar</string>
    <string name="infopago">Se va a realizar un pago de: </string>
    <string name="identificacion">Identificación</string>
    <string name="menu">Menú principal</string>
    <string name="historial">Historial</string>
    <string name="compra">Compra</string>
    <string name="pago">Pago</string>
    <string name="intropin">Introduce el PIN</string>
    <string name="lector">Lector de QR</string>
```

```
</resources>
```

9.11.3.21 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.comedorcv"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="9"
        android:targetSdkVersion="18" />

     <uses-permission android:name="android.permission.CAMERA" />
     <uses-permission android:name="android.permission.INTERNET" />
     <uses-permission android:name="android.permission.READ_PHONE_STATE" />
     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
     <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/nombre"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/menu" >

        </activity>
        <activity
            android:name="com.comedorcv.Autorizacion"
            android:label="@string/nombre">
             <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
           android:name="com.comedorcv.Compra"
           android:label="@string/compra"
           android:screenOrientation="portrait" >
       </activity>
       <activity
            android:name="com.comedorcv.Devolucion"
            android:label="@string/devolucion"
            android:screenOrientation="portrait" >
       </activity>
        <activity
           android:name="com.comedorcv.ImageAdapter"
           android:label="@string/app_name"
           android:screenOrientation="portrait" >
       </activity>
       <activity
            android:name="com.comedorcv.LectorQR"
            android:label="@string/lector"
            android:screenOrientation="portrait" >
       </activity>
       <activity
            android:name="com.google.zxing.client.android.CaptureActivity"
            android:configChanges="orientation|keyboardHidden"
            android:screenOrientation="portrait"
```

```
            android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
            android:windowSoftInputMode="stateAlwaysHidden" >
            <intent-filter>
              <action android:name="android.intent.action.MAIN" />
              <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
            <intent-filter>
              <action android:name="com.google.zxing.client.android.SCAN" />
              <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
         <activity
            android:name="com.comedorcv.Pago"
            android:label="@string/pago"
            android:screenOrientation="portrait" >
        </activity>
        <activity
            android:name="com.comedorcv.Tarjetapin"
            android:label="@string/intropin"
            android:screenOrientation="portrait" >
        </activity>
        <activity
            android:name="com.comedorcv.Historial"
            android:label="@string/historial"
            android:screenOrientation="portrait" >
        </activity>
    </application>

</manifest>
```

connectbd.php

```
<?php

class DB_Connect {

    // constructor
    function __construct() {

    }

    // destructor
    function __destruct() {
        // $this->close();
    }

    // Connecting to database
    public function connect() {
        require_once 'config.php';
        // connecting to mysql
        $con = mysql_connect(DB_HOST, DB_USER, DB_PASSWORD);
        // selecting database
        mysql_select_db(DB_DATABASE);

        // return database handler
        return $con;
    }

    // Closing database connection
    public function close() {
        mysql_close();
    }

}
```

```
?>
```

config.php

```
<?php
/**
 * Database config variables
 */
define("DB_HOST", "127.0.0.1");
define("DB_USER", "root");//cambiar por el nombre de usuario definido en la
configuracion de la BD.
define("DB_PASSWORD", "");//Modificar por el password elegido
define("DB_DATABASE", "monedero_virtual");//Nombre de la base de datos reemplazar si
se utilizo otro diferente al mencionado en el tutorial.
?>
```

funciones_bd.php

```
<?php

class funciones_BD {

    private $db;

    // constructor

    function __construct() {
        require_once 'connectbd.php';
        // connecting to database

        $this->db = new DB_Connect();
        $this->db->connect();

    }

    // destructor
    function __destruct() {

    }

    /**
     * Verificar si el usuario es válido
     */

    public function isuserexist($nif,$pass,$deviceid) {

        $select = mysql_query("SELECT e.desc_espacio AS espacio from nif_auto n,
espacios e WHERE n.nif_autorizados = '$nif' AND n.pin = '$pass' AND n.terminal =
'$deviceid' AND n.espacio=e.cod_espacio");

        $num_rows = mysql_num_rows($select); //numero de filas retornadas

        if ($num_rows > 0) {

            // el usuario es correcto
                    while ($fila = mysql_fetch_array($select)) {

                            $tempo = $fila["espacio"];
                            $tempo2 = utf8_encode($tempo);
                            $result [] = array("espacio"=>"$tempo2");

                    }

        } else {
```

```
            // no existe
            return false;
        }

              return ($result);
              mysql_free_result($select);
    }

    /**
     * Verificar si el pin de la tarjeta es correcta
     */

    public function ispincorrect($codtar,$hash) {

        $select = mysql_query("SELECT * from monederos WHERE cod_tarjeta = '$codtar'
AND pin = '$hash'");

        $num_rows = mysql_num_rows($select); //numero de filas retornadas

        if ($num_rows > 0) {

            $result []=array("regstatus"=>"1");

        } else {
            // no existe
            return false;
        }

              return ($result);
              mysql_free_result($select);
    }


       /**
        * escribir en el historial
        */
       public function write_history($id_operacion, $cod_tarjeta, $nif_autorizado,
$tipo_operacion, $importe, $cod_concepto, $terminal) {

              $result             =             mysql_query("INSERT             INTO
historial(id_operacion,cod_tarjeta,nif_autorizado,tipo_operacion,importe,cod_concepto
,terminal)
VALUES('$id_operacion','$cod_tarjeta','$nif_autorizado','$tipo_operacion','$importe',
'$cod_concepto','$terminal')");

              // check for successful store
              if ($result) {
                     return true;
              } else {
                     return false;
              }
       }


       public function cobrar($id_operacion, $cod_tarjeta, $importe){

              $saldoantiguo   =   mysql_query("SELECT   saldo   FROM   monederos   WHERE
cod_tarjeta='$cod_tarjeta'");

              $num_rows = mysql_num_rows($saldoantiguo);
              if ($num_rows = 1){
                     while ($fila = mysql_fetch_array($saldoantiguo)){
                            $tempo1 =$fila["saldo"];
                            $tempo = utf8_encode($tempo1);
                            $saldoant = (float)$tempo;
                     }
```

```
                    $impor = (float)$importe;


                    if (($saldoant == 0) OR ($saldoant < $impor)){
                           return false;
                    }else{

                           $saldonuevo= $saldoant - $impor;
                           $saldonue = number_format($saldonuevo,2,'.','');

                           $result    =    mysql_query("UPDATE    `monederos`    SET
`saldo`='$saldonue' WHERE `cod_tarjeta`='$cod_tarjeta'");

                           if($result){
                                  return true;
                           }else{
                                  return false;
                           }
                    }
             }else{
                    return false;
             }
      }

      public function devolver($id_operacion, $cod_tarjeta, $importe){

             $saldoantiguo   =   mysql_query("SELECT   saldo   FROM   monederos   WHERE
cod_tarjeta='$cod_tarjeta'");

             $num_rows = mysql_num_rows($saldoantiguo);
             if ($num_rows =1){
                    while ($fila = mysql_fetch_array($saldoantiguo)){
                           $tempo1 =$fila["saldo"];
                           $tempo = utf8_encode($tempo1);
                           $saldoant = (float)$tempo;
                    }
                    $impor = (float)$importe;

                    $saldonuevo= $saldoant + $impor;
                    $saldonue = number_format($saldonuevo,2,'.','');

                    $result      =      mysql_query("UPDATE      `monederos`      SET
`saldo`='$saldonue' WHERE `cod_tarjeta`='$cod_tarjeta'");

                    if($result){
                           return true;
                    }else{
                           return false;
                    }
             }else{
                    return false;
             }
      }

      public function verhistorial($codtar, $terminal){

             $selec   =   mysql_query("SELECT   h.id_operacion   AS   id_operacion,
h.cod_tarjeta AS cod_tarjeta, h.nif_autorizado AS nif_autorizado,
             t.descripcion AS tipo_operacion, h.importe AS importe, h.cod_concepto
AS  cod_concepto,  h.terminal  AS  terminal,  h.pin  AS  pin  FROM  `historial`  h  ,
tipos_operaciones t
             WHERE    h.cod_tarjeta='$codtar'    AND    h.terminal='$terminal'    AND
h.tipo_operacion=t.tipo_operacion");

             while ($fila = mysql_fetch_array($selec)) {
```

```
                        $tempo1 =$fila["id_operacion"];
                        $tempo = utf8_encode($tempo1);
                        $result []= array("id_operacion"=>"$tempo");

                        $aux1 =$fila["cod_tarjeta"];
                        $aux = utf8_encode($aux1);
                        $result []= array("cod_tarjeta"=>"$aux");

                        $aux1 =$fila["nif_autorizado"];
                        $aux = utf8_encode($aux1);
                        $result []= array("nif_autorizado"=>"$aux");

                        $aux1 =$fila["tipo_operacion"];
                        $aux = utf8_encode($aux1);
                        $result []= array("tipo_operacion"=>"$aux");

                        $aux1 =$fila["importe"];
                        $aux = utf8_encode($aux1);
                        $result []= array("importe"=>"$aux");

                        $aux1 =$fila["cod_concepto"];
                        $aux = utf8_encode($aux1);
                        $result []= array("cod_concepto"=>"$aux");

                        $aux1 =$fila["terminal"];
                        $aux = utf8_encode($aux1);
                        $result []= array("terminal"=>"$aux");

                        $aux1 =$fila["pin"];
                        $aux = utf8_encode($aux1);
                        $result []= array("pin"=>"$aux");

                }

                //echo $result;
                return ($result);
                mysql_free_result($selec);


        }
    }

?>
```

autenticar.php

```
<?php

$nif = $_POST['nif'];
$pass = $_POST['pass'];
$deviceid = $_POST['deviceid'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

        $res=($db->isuserexist($nif,$pass,$deviceid));

        if($res!=null){

                $resultado[]=array("regstatus"=>"1");

                $centro = $res[0];
                $resultado[] = $centro;
```

```
        }else{
                $resultado[]=array("regstatus"=>"0");
        }

echo json_encode($resultado);

?>
```

validarpin.php

```
<?php

$codtar = $_POST['codtarjeta'];
$hash = $_POST['hash'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

        $res=($db->ispincorrect($codtar,$hash));

        if($res!=null){

                $resultado[]=array("regstatus"=>"1");

        }else{
                $resultado[]=array("regstatus"=>"0");
        }

echo json_encode($resultado);

?>
```

pagar.php

```
<?php

$id_operacion = $_POST['id_operacion'];
$cod_tarjeta = $_POST['cod_tarjeta'];
$nif_autorizado = $_POST['nif_autorizado'];
$tipo_operacion = $_POST['tipo_operacion'];
$importe = $_POST['importe'];
$cod_concepto = $_POST['cod_concepto'];
$terminal = $_POST['terminal'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

        if($tipo_operacion == '1'){         //Pagar

                if($db->cobrar($id_operacion, $cod_tarjeta, $importe)){

                        if($db->write_history($id_operacion,                $cod_tarjeta,
$nif_autorizado, $tipo_operacion, $importe, $cod_concepto, $terminal))
                        {
                                $resultado[]=array("regstatus"=>"1");
                        }else{
                                $resultado[]=array("regstatus"=>"0");
                        }
                }else{
                        $resultado[]=array("regstatus"=>"0");
                }
        }else{                  //Devolución
```

```
		if($db->devolver($id_operacion, $cod_tarjeta, $importe)){

			if($db->write_history($id_operacion, 		$cod_tarjeta, 
$nif_autorizado, $tipo_operacion, $importe, $cod_concepto, $terminal))
			{
				$resultado[]=array("regstatus"=>"1");
			}else{
				$resultado[]=array("regstatus"=>"0");
			}
		}else{
			$resultado[]=array("regstatus"=>"0");
		}
	}
echo json_encode($resultado);

?>
```

historial.php

```
<?php

/*Buscamos el historial de la tarjeta y con ese terminal móvil*/
$codtar = $_POST['codtar'];
$terminal = $_POST['terminal'];
/*
$codtar = '5901000421958215';
$nifaut = 'aaaa';*/
require_once 'funciones_bd.php';
$db = new funciones_BD();

	$hist=($db->verhistorial($codtar, $terminal));

	if($hist!=null){

		$cuenta = count($hist);
		for ($i = 0; $i< $cuenta;$i++){
			$j = $i % 8;
			switch($j){
				case 0:
					$id_oper = $hist[$i];
					$resultado[]=$id_oper;
					break;
				case 1:
					$cod_tarjeta = $hist[$i];
					$resultado[]=$cod_tarjeta;
					break;
				case 2:
					$nif_autorizado = $hist[$i];
					$resultado[]=$nif_autorizado;
					break;
				case 3:
					$tipo_operacion = $hist[$i];
					$resultado[]=$tipo_operacion;
					break;
				case 4:
					$importe = $hist[$i];
					$resultado[]=$importe;
					break;
				case 5:
					$cod_concepto = $hist[$i];
					$resultado[]=$cod_concepto;
					break;
				case 6:
					$terminal = $hist[$i];
```

```
                                        $resultado[]=$terminal;
                                        break;
                                case 7:
                                        $pin = $hist[$i];
                                        $resultado[]=$pin;
                                        break;
                        }
                }
        }

echo json_encode($resultado);
 ?>
```