ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

SISTEMA PARA LA AMPLIACIÓN DE CONTEXTO BASADO EN RFID PARA TERMINALES MÓVILES

AUTOR: Antonio Rubio Martínez
DIRECTOR: Javier Vales Alonso
Septiembre/2013

ÍNDICE

FIGURAS Y TABLAS

1. Introducción

1.1 Resumen del proyecto

En la actualidad, la tecnología está muy presente en la vida de todas las personas. Se ha convertido en una parte importante de la vida de cada una de esas personas que pueden interactuar con ella. Desde que se inventase el primer computador hasta hoy, se ha avanzado mucho en este camino, pues en aquellos días un ordenador ocupaba toda una sala. Gracias al avance de la investigación y de las técnicas de fabricación se ha hecho posible la creación de dispositivos con mejores prestaciones, en menos espacio, y a un precio más asequible.

Un ejemplo de esto son los Smartphone, dispositivos móviles inteligentes que incorporan todo tipo de prestaciones, así como GPS, Internet, Wifi, Bluetooth, etc. Actualmente se encuentran en auge, y cada día se sigue investigando para crear un Smartphone un poco más potente, con mejores prestaciones. Disponer de uno de estos dispositivos móviles resulta todo un gozo por las numerosas posibilidades que ofrece.

En este proyecto hemos querido unir los Smartphones con la tecnología RFID para crear un sistema que detecte una etiqueta RFID y que la información de esta sea llevada hasta un terminal móvil, el cual, utilizará dicha información para mostrar el contenido requerido por dicha etiqueta.

1.2 Objetivos del proyecto

Partiendo de la base del planteamiento inicial se pueden definir una serie de objetivos que el Proyecto tiene que cumplir para su correcta consecución, y son los siguientes:

- Creación de un programa en lenguaje de programación java el cual sea capaz de interactuar con el lector de RFID para, a raíz de comprobar que etiquetas están activas, actualizar la base de datos cada cierto tiempo.

- Creación de un servidor UDP en java para recibir peticiones de la aplicación móvil y enviar los datos requeridos por dicha aplicación para tratarlos apropiadamente.
- Creación de una Base de Datos MySql para guardar todas las etiquetas RFID de que dispone el sistema y una descripción de cada una de ellas. Esta descripción es lo que leerá el servidor cuando realice una llamada a la base de datos.
- Creación de una aplicación cliente sobre tecnología Android, la cual, se conectará al servidor y este le devolverá la información a mostrar por pantalla.

Para conseguir todos estos objetivos, se consideran los siguientes objetivos adicionales:

- Aprender sobre el sistema operativo *Android*, el modo de programación en ese sistema operativo en dispositivos móviles y su entorno de desarrollo, ya que está en verdadero auge.
- Indagar en la programación sobre dispositivos móviles inteligentes o *Smartphone,* pues ofrecen un mundo de posibilidades y funcionalidades al alcance de la mano.
- Conocer las características de la tecnología RFID y su alcance, ya que cada vez se utiliza mas sobre todo en el campo de la logística mejorando claramente a elementos como el código de barras.
- Aprender a crear y modificar bases de datos MySql donde guardar la información de forma segura y de acceso rápido.

Este proyecto se crea con la idea de dar un soporte genérico para el desarrollo de cualquier tipo de aplicación práctica que necesite de la tecnología RFID, dotándole de una interfaz mucho más amena y de rápido acceso como es un terminal móvil, ya que no sería necesario estar sentado delante de un ordenador

para poder controlar y gestionar el proceso de uso de etiquetas RDID. Se podría disponer de toda la información necesaria desde un móvil y en cualquier parte del mundo ya que la conexión con el servidor se realiza a través de wifi o internet.

1.3 Estructura de la memoria.

A continuación se explica brevemente el contenido de cada uno de los capítulos:

- Introducción: Se explica brevemente el planteamiento inicial del proyecto, así como sus objetivos principales.

- Trabajos relacionados: Buscaremos proyectos del mundo real que hayan buscado objetivos similares al nuestro.

- Descripción técnica de equipos y herramientas: Aquí detallaremos las características del equipo utilizado así como los simuladores y programas necesitados durante el proyecto.

- Arquitectura y desarrollo: Comentaremos detalladamente el desarrollo y funcionamiento del sistema creado.

- Pruebas: Se comentarán los experimentos realizados y sus resultados.

- Conclusiones: se reflexiona sobre los objetivos conseguidos y las dificultades encontradas, y se indican nuevas líneas de desarrollo.

2. Trabajos relacionados.

En este apartado vamos a comentar algunos proyectos llevados a cabo con la tecnología RFID para conocer su funcionalidad y posibles aplicaciones en el mundo real. A pesar de llevar muchos años en el mercado, es en los últimos tiempos cuando se le está dando auge debido, sobre todo, a la reducción de costes y abaratamiento de la tecnología, es por esto, que muchas empresas de diversos campos han comenzado a utilizarla. Exponemos a continuación algunos ejemplos:

- Una empresa de ropa para niños aplica etiquetas RFID a todos los productos, lo que permite que el almacén de la empresa pueda hacer el seguimiento de la llegada y la partida de las mercaderías; sus tiendas que operan por el sistema de franquicias emplean RFID para el seguimiento del inventario, en el punto de venta y para seguridad. Con este sistema dicho inventario permanece en todo momento actualizado en tiempo real.

- Media maratón de Xirivella controla a los corredores con Tecnología RFID. En el año 2011 utilizaron RFID como proyecto piloto para controlar la carrera. El sistema se basa en la utilización de etiquetas RFID que llevan los corredores en sus zapatillas y en una alfombra sintética inteligente diseñada especialmente para este tipo de eventos.

  Cada etiqueta RFID está asociada a un único corredor y se entrega a este junto con su dorsal. La alfombra contiene antenas lectores que detectan cada vez que los corredores pasan por encima de ella y les identifican de manera unívoca y automática.

  Una vez el sistema se puso en funcionamiento, se identificaron cada uno de los corredores que pasaban por la meta y el instante mismo en el que

lo había hecho, diferenciando con gran precisión los tiempos de llegada entre uno y otro.

- Algunas autopistas, como por ejemplo El carril de Telepeaje IAVE en las autopistas de CAPUFE En México la FasTrak deCalifornia, el sistema I-Pass de Illinois, el telepeaje TAG en las autopistas urbanas en Santiago de Chile, la totalidad de las autopistas pagas argentinas y la *Philippines South Luzon Expressway E-Pass* utilizan etiquetas RFID para recaudación con peaje electrónico. Las tarjetas son leídas mientras los vehículos pasan; la información se utiliza para cobrar el peaje en una cuenta periódica o descontarla de una cuenta prepago. El sistema ayuda a disminuir el entorpecimiento del tráfico causado por las cabinas de peaje.

3. Descripción técnica de equipos y herramientas

3.1 Tecnología RFID

Vamos a comenzar explicando en que consiste RFID en un ámbito general y a continuación particularizaremos en el lector concreto que hemos usado en nuestro proyecto.

En términos generales, la tecnología RFID (“*Radio Frequency IDentification”*) permite la identificación de objetos de forma inalámbrica, sin necesidad de que exista entre el lector y el objeto contacto o línea de visión directa, requisito indispensable para otras tecnologías como la lectura laser de códigos de barras. Esta identificación se realiza mediante la incorporación o fijación de un transpondedor al objeto (“*tag*”), el cual transmite los datos que contiene cuando detecta que está siendo interrogado por un lector RFID.

Aunque la tecnología no es nueva, los avances técnicos en aspectos tales como alcance, seguridad, almacenamiento o velocidad de lectura entre otros, han suscitado el interés de la industria por ella, considerándola como el sustituto natural del código de barras dada la importante oportunidad que RFID ofrece para conseguir una importante reducción de costes en las cadenas de producción y logística. Grandes empresas internacionales con una importante carga logística o de producción han comenzado a implantar la tecnología o han exigido a sus proveedores que la incorporen, motivadas por las notables mejoras que supone su introducción para sus procesos productivos. Algunos casos ampliamente documentados son los de las empresas Wal-Mart, Metro Group, Tesco, Mark&Spencer, Departamento de Defensa de Estados Unidos, Michelin, BMW, Volvo, Hewlett-Packard, Best-Buy o Nokia.

Sin embargo, aunque la aplicación natural de esta tecnología sea dentro de la cadena de producción y distribución, diariamente aparecen nuevas aplicaciones

y oportunidades de negocio alrededor de las distintas variantes de esta tecnología de identificación y su combinación con otras tecnologías. Aplicaciones sobre las que se puede encontrar una amplia bibliografía e implantaciones en distintos sectores de actividad son:

- Control de acceso: peajes de carreteras, aparcamientos, acceso a edificios, acceso a zonas restringidas.
- Prepago: peajes de carreteras, transportes (autobús, metro), pago con teléfono móvil.
- Identificación, localización y monitorización de personas, animales o materiales: en combinación con sensores (temperatura, humedad), tecnología inalámbrica (wlan) o tecnología de localización (GPS).
- Autenticidad de productos ("anti-counterfeiting") o documentos.

Son tantas las posibilidades de utilización de la tecnología RFID en todos los sectores de actividad que, hoy día, se la considera uno de los pilares basicos de la siguiente evolución de las redes de comunicación, la cual ha recibido varias denominaciones ("*Internet of things*", "*Ambient Intelligence*", "*Ubiquitous Computing*") aunque todas ellas se refieren al mismo concepto: la interacción automática e inteligente entre dispositivos en cualquier circunstancia o ubicación, y su comunicación con sistemas remotos de datos a través de las redes de telecomunicación.

En un primer acercamiento, un sistema RFID básico se puede definir en los siguientes puntos:

- El objetivo de la tecnología es la identificación de objetos a distancia, via radio, sin necesidad de contacto ni línea de visión directa.
- Una solución básica basada en RFID se compone de un lector con una o más antenas, etiquetas de identificación ("*tags*") y un software que realice el tratamiento de la información recogida por los lectores.

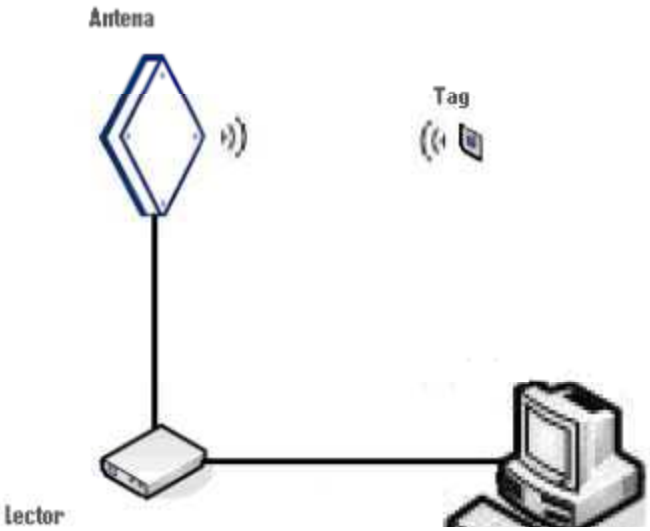

Figura 1: Sistema RFID básico "standalone"

Hay que tener en cuenta que la potencia de la tecnología RFID reside tanto en su bajo coste como en la universalidad ("*serialization*") y unicidad del código identificador del tag (EPC, "*Electronic Product Code*"), fundamentales para las aplicaciones de la cadena de suministro.

Por tanto, la estandarización a nivel mundial tanto del código EPC (concebido como evolución del código UPC, "*Universal Product Code*" de los códigos de barras) como de los mecanismos para su asignación y para garantizar la interoperabilidad de los distintos sistemas es vital cuando se habla de RFID.

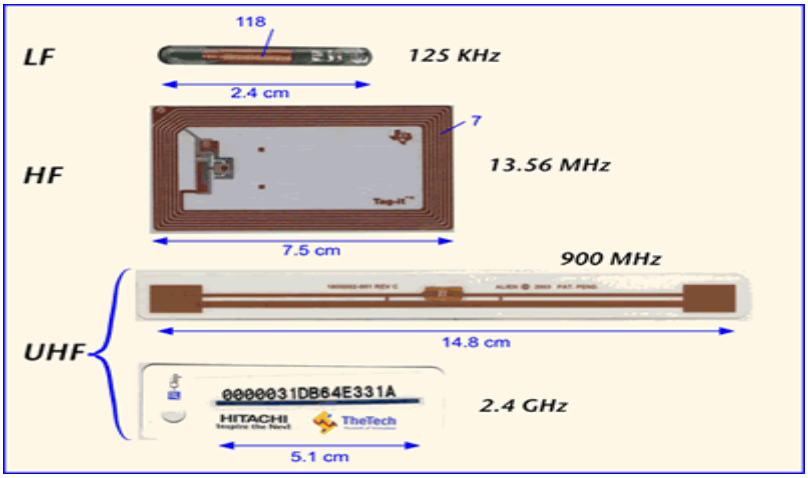

Figura 2: Estructura etiquetas RFID.

3.2 Lector RFID Alien ALR-8800

Un lector o reader RFID es realmente como una radio, como la que podemos llevar en nuestro coche, que capta o produce señales analógicas. Su funcionamiento es sencillo, el reader produce electricidad que viaja por cable a un ratio determinado, normalmente hacia una antena que radia la misma senal en el espacio a una frecuencia determinada para que otros elementos lo escuchen. No solo genera la señal que a través de las antenas se transmite en el aire, sino que también escucha las respuestas de los tags procedentes también de estas antenas. Transmite y recibe ondas analógicas que transforma en cadenas de bits de ceros y unos, bits de información digital. A veces cuando se habla de lectores, ya se entiende que también se habla de las antenas, ya que existen lectores con antenas integradas y otros que necesitan su conexión.

Cada lector es conectado a una o más antenas (máximo según tipo de lectores). Estas tienen una ciencia propia, pero es importante conocer como el reader crea la señal electromagnética y la antena realiza la difusión en su zona de interrogación (campo de radio frecuencia).

Además el reader también se conecta a una red de datos LAN o WAN o a un PC mediante varios tipos de interfaz como pueden ser RS-232 o Ethernet. Hay multitud de tipos de lectores: simples (un solo estándar y frecuencia), multiregionales, multifrecuencias (trabajan a diferentes frecuencias), multiprotocolos, etc.

En nuestro caso concreto hemos utilizado el lector de la marca **Alien,** se trata de un fabricante líder en la fabricación de etiquetas RFID, en concreto su modelo **ALR-8800**. Se trata de un modelo fácil de instalar y de operar. Se configura localmente vía puerto serie o vía remota mediante la red de área local LAN mediante el protocolo TCP/IP (RJ-45). Es capaz de utilizar los Protocolos de Red: DHCP, TCP/IP, SNTP, DNS, SNMP.

Este lector utiliza los protocolos RFID EPC Class 1 Gen 2, ISO 18000-6c.

Dispone de 4 puertos para conectar hasta 4 antenas a la vez, cada una con hasta 6 metros de cable para abarcar un amplio espectro de espacio. El software que hemos usado para controlar y manejar el lector ha sido Java pero también se podrían haber usado otros como por ejemplo .NET .

Figura 3: Lector RFID Alien ALR-8800

El protocolo habitual de actuación suele ser conectarle las antenas al lector y a continuación conectarlo mediante el cable de red al puerto RJ-45 a un pc. Una vez hecho esto conectaremos el lector a la red eléctrica y comenzaros con su uso. A pesar de traer su propio software “alien RFID gateway”, lo mejor es usar las API’s proporcionadas por el fabricante para manejarlo mediante Java. Una vez hecho todo esto estaremos en disposición de tratar la información proporcionada por el lector de la manera que nos sea necesaria.

3.3 Etiquetas RFID o Tags.

Cuando el lector transmite en el espacio, espera normalmente una respuesta de otro elemento para mantener la comunicación, en los sistemas RFID es el tag quien responde también denominado como etiqueta o transpondedor. Un tag RFID está compuesto por tres partes: el chip o circuito integrado (IC), la antena y el sustrato. El chip es un minúsculo microprocesador que almacena una serie de información. También contiene la lógica de lo que hay que hacer para responder a un lector. La antena permite al chip recibir la energía y la comunicación procedente del lector para emitir la suya y poder intercambiar flujos de datos entre ellos. Destacar que los tags, opcionalmente, pueden estar encapsulados para incrementar su protección mediante plásticos, siliconas, etc.

Hay una gran variedad de tipos de tags, siendo este elemento es más difícil de decidir según la aplicación que vayamos a realizar. Podemos diferenciarlos según su fuente de energía como activos, semi activos y pasivos, según su memoria como tags de solo de lectura, tags de múltiple lectura y una sola escritura, tags de múltiples lecturas y escrituras, etc., o bien clasificarlos según los estándares que cumplen, su ciclo de vida, su tamaño, su distancia de lectura, etc.

En nuestro caso hemos usado tags pasivos. Este tipo de tags no requieren de batería, ni interna ni externa, ya que toda la energía la recoge del campo electromagnético creado por el lector.

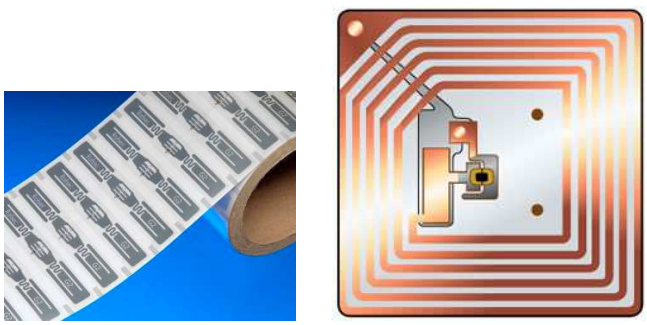

Figura 4: Etiquetas pasivas.

Este tipo de tag son los más económicos aunque, por otra parte, son los que tienen menor rango de comunicación, pero por su relación entre comportamiento y precio son los más utilizados.

El EPCglobal es el órgano encargado de la estandarización para la tecnología RFID y han organizado los tags en 6 clases dependiendo de las veces de lectura y escritura, de si tienen fuente de alimentación…

En nuestro caso utilizamos etiquetas de clase 1, este tipo de tags son de una sola escritura y lecturas indefinidas. Se fabrican sin código EPC y se incorpora al tag mas tarde.

3.4 Software RIFIDI Emulador

Rifidi Emulator es una herramienta de desarrollo para simular rápidamente lecturas y eventos tag. Se trata de un software desarrollado por rifidi.org para favorecer y hacer más rápida y cómoda la labor del programador. Es un programa muy simple gracias al cual se puede simular un lector Alien y crear las etiquetas que consideremos oportunas introduciéndoles el código EPC que deseemos. No será necesario llevar el lector y las antenas, lo cual es bastante pesado y aparatoso, ni perder tiempo en conectarlo todo para realizar pruebas de desarrollo.

Su uso es muy simple. Se comenzará x ejecutar el programa, una vez cargado deberemos agregar un reader de la lista de readers compatibles, en nuestro caso el Alien.

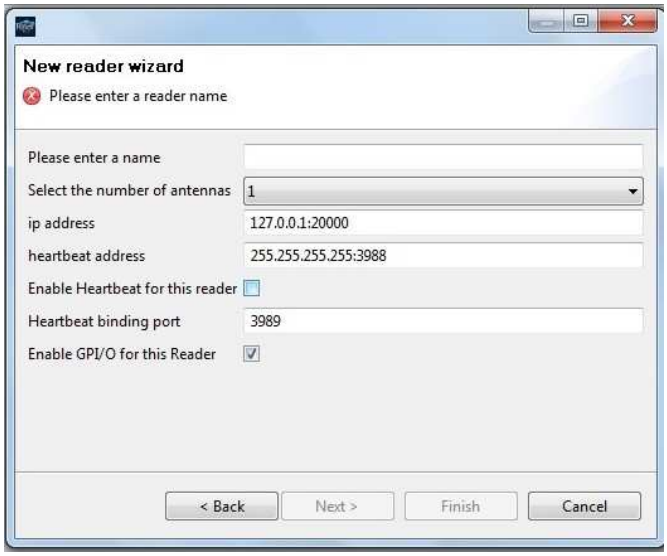

Figura 5: Captura crear reader.

A continuación nos pedirá un nombre para el reader, el número de antenas que queremos simular, la dirección IP, que usaremos la dirección de localhost, y el puerto a través del cual se conectará.

Una vez creado el reader lo siguiente será crear las etiquetas, para ello nos iremos a la parte de “TagsView” y agregaremos las etiquetas necesarias.

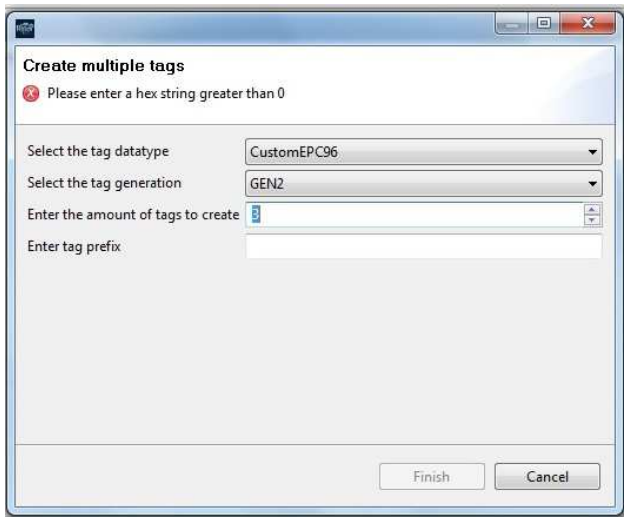

Figura 6: Captura crear etiquetas.

Nos pedirá el tipo de etiqueta, en nuestro caso elegiremos el modelo CustomEPC96, la generación del tag, las etiquetas que queremos crear y el código que queremos que lleven.

Y con esto ya tendremos todo el programa configurado listo para ejecutarlo. Para ello deberemos activar el reader.

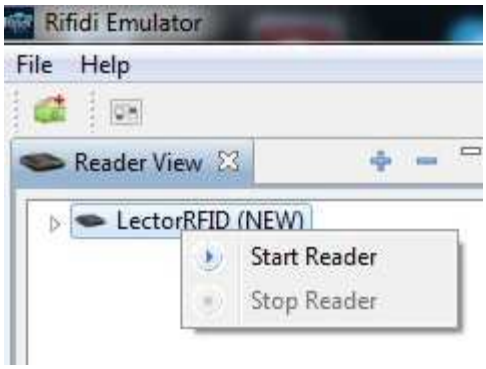

Figura 6: Captura activar reader.

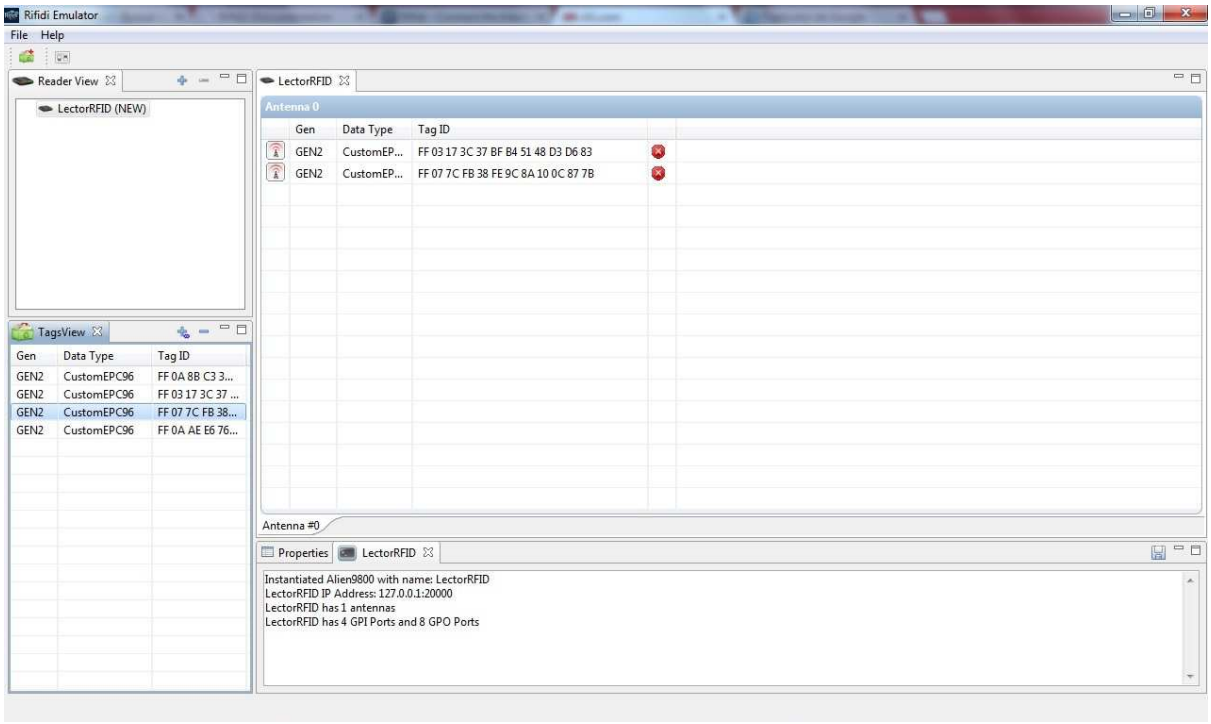

Figura 7: Captura pantalla principal.

En esta pantalla vemos el lector y las etiquetas creadas en el lado izquierdo. Para activar una etiqueta lo que debemos hacer es pinchar sobre ella y arrastrarla a la parte derecha donde esta la pestaña de la antena. Y con esta sencilla y rápida configuración ya estaremos en disposición de utilizar el emulador.

3.5 Android

Android es una plataforma de software y un sistema operativo (SO) diseñado para dispositivos móviles, es decir, teléfonos inteligentes o *Smartphone*, aunque su desarrollo se ha ampliado para soportar otros dispositivos tales como *tablets*, *netbooks*, PC'S, lectores de libros electrónicos, reproductores de audio MP3, … Este SO está basado en *GNU/Linux*, la mayor parte de la plataforma está disponible bajo la licencia de software libre *Apache* (y otras licencias de código abierto), y en la actualidad pertenece a *Google*. Tras de sí tiene una gran comunidad de desarrolladores de aplicaciones para extender la funcionalidad de los dispositivos que incorporan este SO, haciendo, en el ejemplo de los dispositivos móviles, que lideren las listas de ventas.

La estructura del SO Android se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de bibliotecas de Java en una máquina virtual con compilación en tiempo de ejecución (*Dalvik*). Las bibliotecas, escritas en lenguaje de programación C, incluyen un administrador de interfaz gráfica, un *framework OpenCore*, una base de datos relacional *SQLite*, una API gráfica *OpenGL ES 2.0 3D*, un motor de renderizado *WebKit*, un motor gráfico SGL, SSL y una biblioteca estándar de *C Bionic*.

Las aplicaciones para este SO se desarrollan en Java, gracias a un SDK que puede ser integrado en entornos de desarrollo (como *Eclipse IDE*), y que proporciona las herramientas necesarias para crear y desarrollar aplicaciones para dispositivos con este SO. Además, también se pueden escribir aplicaciones en otros lenguajes de programación, pero este método no está soportado oficialmente por *Google*. Un ejemplo de esto es desarrollar las aplicaciones en lenguaje C y posteriormente se compilan en código nativo ARM para su ejecución.

El kit de desarrollo de software (*software development kit* o *SDK*) incluye un conjunto de herramientas de desarrollo, tales como un *debugger*, librerías, documentación, códigos de ejemplo, tutoriales y un emulador (basado en *QEMU*). Está soportado en SO *Windows*, *Linux* y *Mac*. El entorno de desarrollo (*Integrated Development Environment* o *IDE*) oficialmente soportado es *Eclipse* conjuntamente con el *plugin ADT* proporcionado por *Google*.

En la actualidad, los dispositivos móviles o Smartphone que implementan el sistema operativo *android* están en lo más alto del mercado, pues disponen de buenas prestaciones de por sí solos, además de la experiencia que supone para el usuario utilizar este sistema operativo, pues exprime las capacidades del terminal ofreciendo las mayores prestaciones, y manteniendo un consumo con respecto a otros Smartphone muy bueno.

En la realización de este proyecto hemos hecho las pruebas con un móvil HTC Desire con versión de SO Android 2.3.7.

Figura 8: Móvil HTC Desire.

3.6 Base de datos MySQL

Se puede definir *Base de Datos* como un conjunto de datos que pertenecen a un mismo contexto, y que se almacenan de forma sistemática para su uso posterior. Normalmente este término siempre se asocia a bases de datos en formato digital, pues con el desarrollo de la informática y la electrónica, se ha incrementado mucho el uso de éstas en el formato mencionado debido a la gran variedad de soluciones que ofrece a la hora de almacenar datos. En prácticamente todos los escenarios donde se requiere el almacenamiento de información se utilizan bases de datos.

Como se ha mencionado, el uso de bases de datos en formato digital está muy extendido debido a la eficacia y rapidez con la que poder almacenar, recuperar y tratar datos almacenados gracias a programas desarrollados para tal fin, denominados *Sistemas Gestores de Bases de Datos* (*SGBD*), tales como *Microsoft Access*, *SQLite*, *MySQL*, *Open Access*, *Oracle*…

Para el presente proyecto se ha empleado la base de datos *MySQL* porque es una de las más populares, pues es de código abierto ofreciendo soporte para más de 20 plataformas como *Linux*, *Windows*, *NetWare*, *OS/X*, … y se pueden usar infinidad de lenguajes de programación para acceder a *MySQL*. Existen interfaces para *C*, *C++*, *PHP*, *C#*, Java, …, y para los lenguajes que no poseen interfaz es posible acceder a través de *ODBC*. Es multihilo, por lo que es capaz de trabajar con más de un procesador simultáneamente. Ofrece un alto rendimiento, fiabilidad y facilidad de uso, proporcionando además una amplia gama de herramientas de bases de datos. Debido a estas características, es la base de datos de elección para muchas empresas.

El diseño de la base de datos que hemos escogido es una sola tabla con 4 campos:

- Campo 1: Identificador numérico único. Se le asigna un número a cada etiqueta introducida en la base de datos.
- Campo 2: Descripción. Se trata de la información que le daremos a las consultas que se hagan sobre la etiqueta en cuestión.
- Campo 3: Objeto. Código de la etiqueta RFID.
- Campo 4. Visible. Si este campo está activo significa que la etiqueta está siendo leída por una antena RFID, de lo contrario estará desactivado.

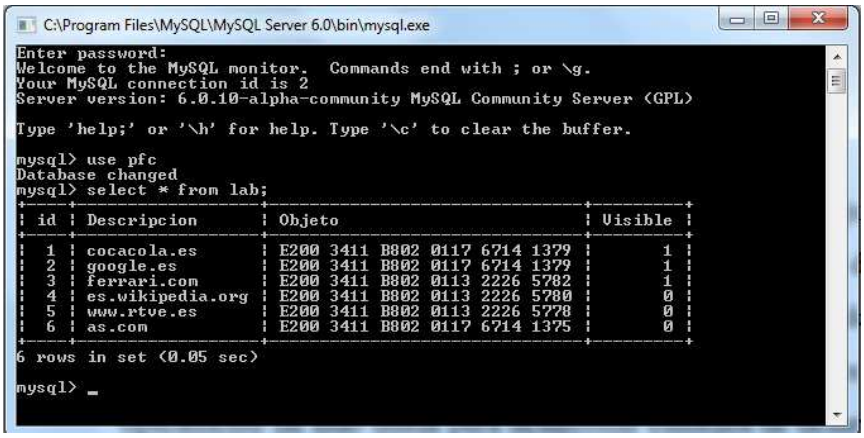

Figura 9: Tabla de etiquetas.

3.7 Entorno de desarrollo (Eclipse).

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

El entorno de desarrollo empleado para desarrollar la aplicación *android* ha sido el IDE Eclipse, un entorno de desarrollo integrado de código abierto multiplataforma.
A la hora de crear una aplicación, hay que tener en cuenta que el aspecto gráfico se hace por un lado, en lenguaje *xml* dada la pobreza del editor gráfico, y el código que ejecuta la aplicación por otro. De esta forma, todo lo referente a componentes gráficos se desarrolla y guarda en la carpeta *res*, mientras que el código fuente se guarda en *src*.

Aunque pueda parecer confuso esta separación, una vez trabajas un poco con la herramienta no supone ningún problema.

4. Arquitectura y desarrollo

4.1 Arquitectura

El sistema con el que se da solución al proyecto es un sistema formado por un servidor central, ejecutado en un PC, el cual se encontrará siempre en funcionamiento a la espera de clientes y para mantener siempre actualizada la base de datos y será el encargado de varias tareas:

- Conectarse a la base de datos para importar información.
- Mantener actualizada en la base de datos que etiquetas están activas.
- Conectarse con el reader y comprobar que etiquetas están activas.
- Conectarse con la aplicación móvil parar transmitirle la información.

Y la aplicación cliente, que será la que se ejecutará en los dispositivos móviles y que se conectará con el servidor y este le enviará la información de los tags activos que será mostrada por pantalla por medio de una ListView.

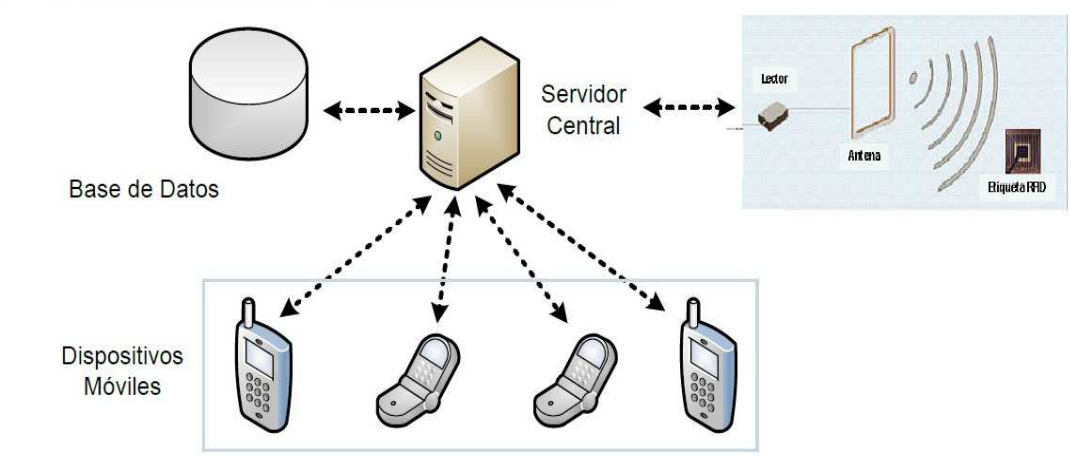

Figura 10: Arquitectura proyecto.

Los datos que se intercambiarán entre servidor y dispositivo móvil mediante el protocolo UDP.

Figura 11: Intercambio de datos.

Particularizando un poco más en el sistema diremos que se trata de una aplicación que cuando las antenas RFID detectan un tag acceden a una base de datos MySQL, donde cada tag tienen asociado una url determinada, y activa dicho tag. Por otro lado, tendremos un servidor en lenguaje Java el cual estará a la espera de peticiones, las cuales le llegarán cuando el usuario arranque la aplicación Android desde el terminal móvil. Una vez conectado al servidor, este accederá a la base de datos donde obtendrá las url's de los tag activos en ese momento, la cual será enviada al móvil, que mostrará en su pantalla una lista con las url's para que el usuario pueda acceder a la página web que desee en ese momento.

4.2 Desarrollo del servidor

La labor del servidor será, como se ha mencionado en capítulos previos, servir a los clientes (dispositivos móviles) que lo soliciten, enviándole la información relevante al tag o tags activos conectándose a la base de datos. Esta es una descripción muy simplificada de la labor del servidor, pues también dispone de multitud de opciones para tratar información, actualizar el sistema...

El servidor ha sido completamente programado en lenguaje Java. A continuación se explicará cada una de las clases desarrolladas con detalle,

explicando cada una de las variables y métodos implementados para llevar a cabo su labor.

4.2.1 Clase ServidorUDP

Esta clase contiene únicamente al método main. Está siempre en funcionamiento a la espera de peticiones por parte de los clientes, ya que si en algún momento se cerrara toda la aplicación dejaría de ser útil, ya que es el centro de todo el sistema. Para ello hemos creado un bucle infinito.

```
while (true) {
```

El servidor estará a la espera de recibir un paquete por parte de la aplicación móvil.

```
socket.receive(paquete);
```

Una vez recibido se conecta a la clase “LeerSQL” para obtener un string con las url’s de los tag activos.

```
MensajeSalida = LeerSQL.BD().toString();
```

Ya obtenida la información solo queda enviársela al cliente.

```
socket.send(paquete2);
```

4.2.2 Clase LeerSQL

A esta clase se conectará “ServidorUDP” cuando le llegue una petición del cliente.

Es la encargada de conectarse a la base de datos

```
static String login = "root";
static String password = "";
static String url = "jdbc:mysql://localhost:3306/pfc";

Class.forName("com.mysql.jdbc.Driver").newInstance();
conn = DriverManager.getConnection(url,login,password);
```

y obtener la información de los tags activos.

```
ResultSet rs = st.executeQuery ("select * from lab");
while (rs.next()) {
      if(rs.getInt(4) == 1) {mensaje2 = rs.getString(2);
      mensaje3 = mensaje3 + "-" + mensaje2;
      }
}//while
```

Este proceso lo realiza haciendo una consulta la cual es guardada en rs y accede al campo 4 (que si está a 1 significa que el tag está activo) y guarda en mensaje los tags activos.

Se puede dar el caso de que en un momento determinado no haya ningún tag activo en cuyo caso se le enviará al móvil una página por defecto.

```
if (mensajeFinal == ""){mensajeFinal = "www.upct.es";}
```

Para finalizar devolverá el string con la información.

```
return mensajeFinal;
```

4.2.3 Clase LeerTag

Esta clase es la encargada de conectarse con el lector de RFID.

```
AlienClass1Reader reader = new AlienClass1Reader();

reader.setConnection("127.0.0.1",20000);
//reader.setConnection("192.168.1.100",23);
reader.setUsername("alien");
reader.setPassword("password");
reader.open();
test(reader);
```

Una vez establecida la conexión accederá a la lista de tag detectados por las antenas RFID.

```
String readerName = reader.doReaderCommand("get TagList");
```

Esta consulta nos devuelve no solo el código de la etiqueta sino también mucha otra información, como fecha, tipo de etiqueta…, la cual no nos es necesaria para nuestra aplicación por lo que procederemos a eliminarla y quedarnos solamente con el código de la etiqueta.

```
while(( x = etiquetas.indexOf(patron)) > -1){

etiquetas = etiquetas.substring(etiquetas.indexOf(patron)+patron.length()+1,
etiquetas.length());
tag    = tag  + "-" + etiquetas.substring(0, 29);}
```

Una vez conseguida la información que necesitamos accederemos a la clase "Actualizar" pasándole los tags activos para que en la base de datos tenga siempre actualizadas las etiquetas.

```
Actualizar.BBDD(tag);
```

Esta clase también tendrá que estar siempre en funcionamiento y lo que es más interesante, que tendrá que estar autoejecutandose cada pocos segundos para poder mantener actualizada toda la información. Esto lo conseguimos mediante la implementación de un timer que llama a la clase cada 10 segundos, por lo que se vuelve a ejecutar todo el proceso una y otra vez ininterrumpidamente.

```
Timer timer;
int counter;
public void programar()
{
    timer = new Timer();
    timer.schedule(this, 2 * 1000, 10 * 1000);
}
```

4.2.4 Clase Actualizar

Esta clase es ejecutada solo cuando así lo reclama la clase "LeerTag", esta clase le pasa un string que contiene las etiquetas activas, entonces la clase "Actualizar" accede a la base de datos y compara el string que le ha enviado la clase "LeerTag" con los códigos de las etiquetas que tiene en la base de datos.

```
if (tag.indexOf(mensaje) != -1)  existe = true;
```

En el caso de que coincida el código de la base de datos con el contenido en el string haremos una actualización del campo 4 y lo pondremos a 1, que significa que la etiqueta está activa. En el caso de que no coincidan querra decir que esa etiqueta de la base de datos no esta activa por lo que pondrá el campo a 0.

```
if (existe == true)
            prueba.executeUpdate("UPDATE tag set Visible=1 where Id="+identidad);
            else
            prueba.executeUpdate("UPDATE tag set Visible=0 where Id="+identidad);
```

4.3 Desarrollo del cliente

Por aplicación cliente se refiere a aplicación que se ejecuta en el lado del dispositivo móvil. La misión de esta aplicación es obtener del servidor las url's de los tags activos y crear una lista para que el usuario pueda consultar la página web que le interese. Al igual que en la parte del servidor debemos de actualizar cada pocos segundos para que la información no quede desfasada, aquí deberemos de hacer lo mismo y veremos como la lista se va modificando automáticamente cada vez que se activa o desactiva un tag en la base de datos.

4.3.1 Clase MainActivity

Es la clase principal, la que se ejecuta al iniciar la aplicación. Proporciona al usuario el botón de inicio con el arrancará toda la aplicación, una vez pulsado esta se conectará con él servidor. Esta clase se hereda de la clase *Activity* (su característica principal es que permite interactuar con el usuario), e implementa la interfaz *OnClickListener* (detectar pulsaciones en un *view*). Una vez pulsado envía un intent para enlazar con la clase "SegundaActivity".

```
Intent i=new Intent(MainActivity.this,SegundaActivity.class);
```

4.3.2 Clase SegundaActivity

Esta clase es la más importante de todas dentro del cliente, puesto que es el nexo de unión de todas. Es la encargada de conectar con el servidor de procesar la información y de crear una lista llamando a otras clases para crear un ListView de ítems personalizados. Además es la clase que se autoactualiza cada pocos segundos y mantiene la aplicación móvil siempre actualizada.

Primeramente se envía un string al servidor para comunicarle que queremos que nos envíe la información. Para ello deberemos de proporcionar la dirección IP del servidor y crear el socket UDP para enviar el string.

```
String IP = "192.168.60.8";
```

```
socket.send(p);
```

Después creamos un buffer donde guardamos la información recibida.

```
final byte[] buf = new byte[1024];
final DatagramPacket packet = new DatagramPacket(buf, buf.length);
socket.receive(packet);
```

Una vez que ya tenemos las url's procederemos a crear la lista y a mostrar estas. Para crear una lista personalizada hemos tenido que crear las clases "ItemListAdapter" y "enlaces".

```
 mainListView = (ListView) findViewById( R.id.mainListView );
ArrayList<enlaces> etiquetas = obtenerItems(mensa);
ItemListAdapter adapter = new ItemListAdapter(this, etiquetas);

mainListView.setAdapter(adapter);
```

Para que la clase se autoactualice hemos creado en método empezar() para que cada cierto tiempo llame al método de inicio de la clase.

```
Runnable tarea = new Runnable() {
      @Override
      public void run() {

          conexion();

              handler.postDelayed(this, 300);
      }
  };

  handler.postDelayed(tarea, 300);
```

4.3.3 Clase ItemListAdapter

A esta clase se accede únicamente para crear la ListView. Esta herede de la clase BaseAdapter que a su vez hereda de Adapter, por ello implementamos algunos métodos de la clase Adapter como getCount(), getItem(), getItemId() y getView(). Este BaseAdapter nos permitirá crear un adapter a medida, es decir, totalmente personalizado a nuestras necesidades.

La idea de crear esta lista personalizada fue principalmente porque las normales te muestran string solamente y nosotros necesitábamos crear un textview para

poder aplicarle la modificación a hipervínculo. Esta modificación se hace aplicándole código html al textview.

```
View vi=convertView;

 TextView nombre = (TextView) vi.findViewById(R.id.elink);
 String nombrePagina = item.getNombre();
if(nombrePagina.equals("cocacola.es")){
 nombre.setText(Html.fromHtml(" <br /> <a href=http://"
+ item.getNombre()+ ">CocaCola</a> "));
nombre.setMovementMethod(LinkMovementMethod.getInstance());}
```

4.3.4 AndroidManifest

El manifiesto de *Android* (*AndroidManifiest.xml*) es un archivo que presenta información esencial para la aplicación al sistema *Android*, información que el sistema debe tener antes de que se ejecute cualquier código de aplicación. Para que puedan ejecutarse las clases explicadas, y además se tenga acceso a *Internet o WIFI*, es necesario indicarlas en este archivo *xml*, por lo que habrá que añadir las líneas que se detallan a continuación.

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

Únicamente hemos tenido que añadir estas dos líneas al archivo que venía por defecto para otorgarle permisos de conexión a internet o wifi, necesarios para poder establecer la conexión con el servidor.

5. Pruebas

5.1 Experimentos realizados

Como ya se ha mencionado con anterioridad, la aplicación servidor se ha desarrollado en Java, puesto que es el lenguaje que nos permite más fácilmente la conexión con el móvil ya que ambos usan el mismo lenguaje. La parte del servidor no necesita ser instalada para ejecutarse, tan solo será necesario correr la aplicación desde el IDE. No ocurre lo mismo con el cliente que deberá ser instalada la aplicación en un móvil con SO Android. Esta instalación resulta muy sencilla debido al pluging de android que posee eclipse. Tan solo es necesario conectar el terminal móvil al pc y darle a correr la aplicación, después de esto nos pedirá si queremos usar el móvil o el emulador creado. Si elegimos la opción móvil automáticamente se nos instalará el apk de nuestra aplicación en el móvil y estará listo para usar cuando deseemos.

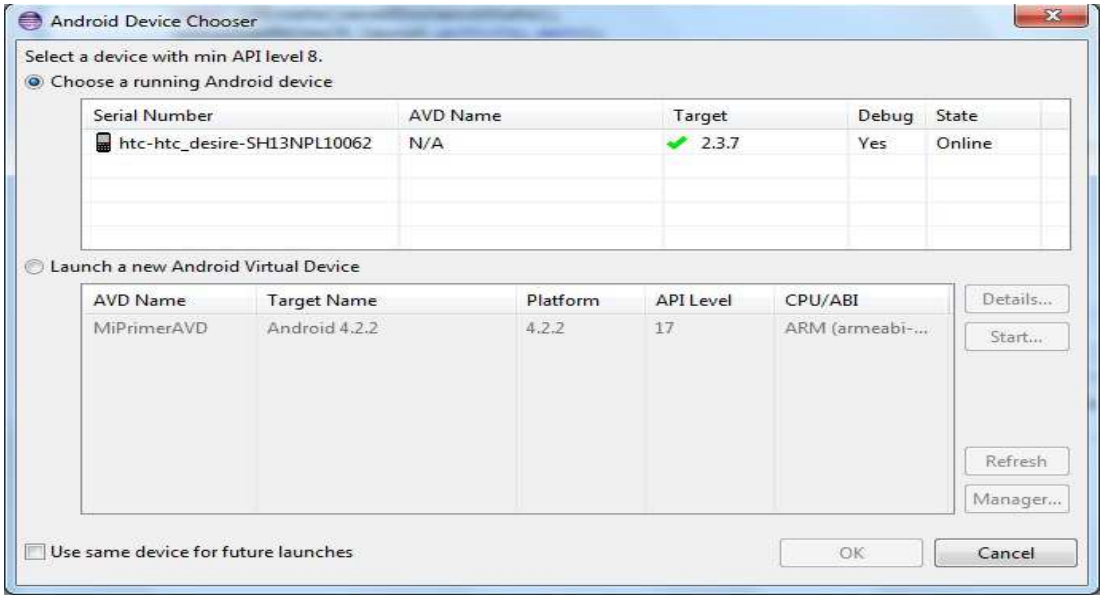

Figura 12: Captura eclipse.

Para realizar esta prueba vamos a usar el programa RIFIDI Emulator, ya que se pueden apreciar de forma más clara la activación y desactivación de etiquetas.

Para ello, procederemos a configurar el programa añadiendo primeramente nuestro reader con las características oportunas.

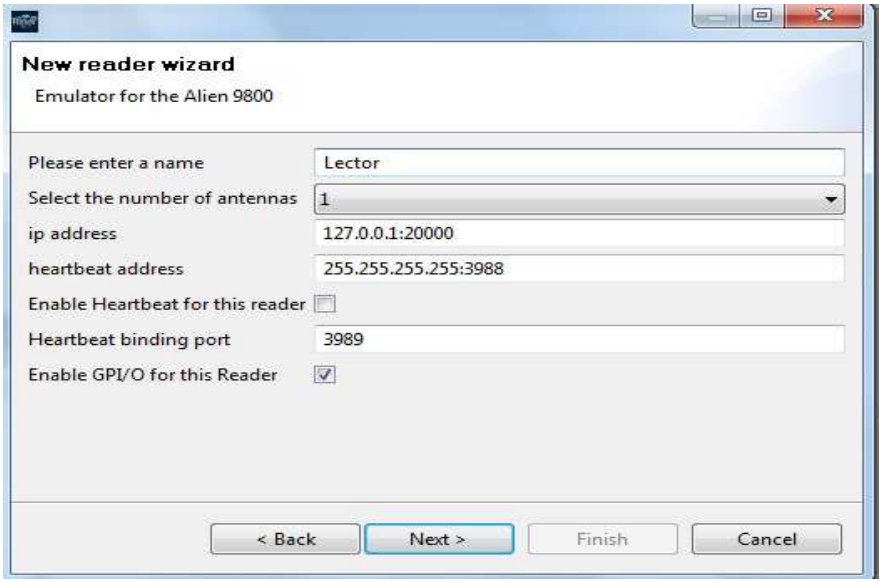

Figura 13: Configuración lector.

Una vez creado el lector procederemos a crear las etiquetas pertinentes, en este caso vamos a crear 4 tags como se puede apreciar en la captura.

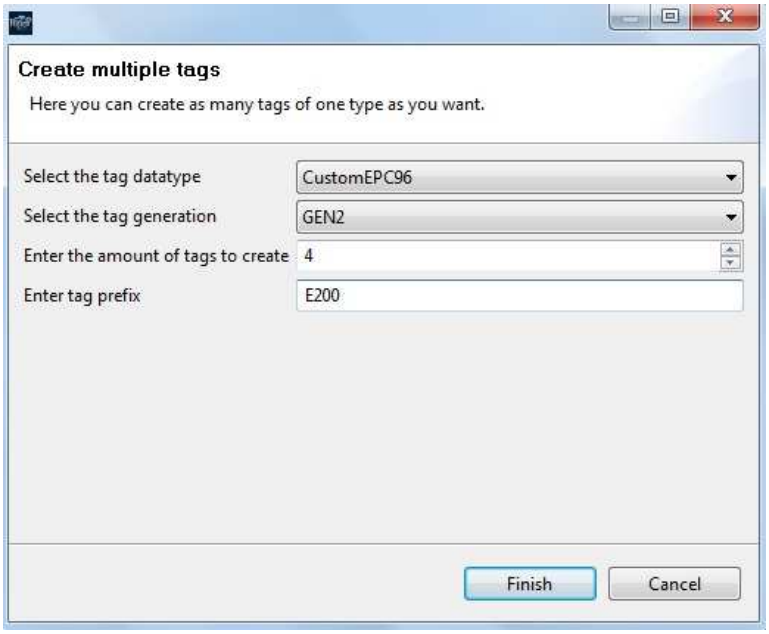

Figura 14: Configuración etiquetas

Una vez creadas todas las etiquetas procederemos a crear la base de datos MySQL con ellas, asociándole a cada etiqueta una url determinada.

Primero crearemos las especificaciones de la base de datos.

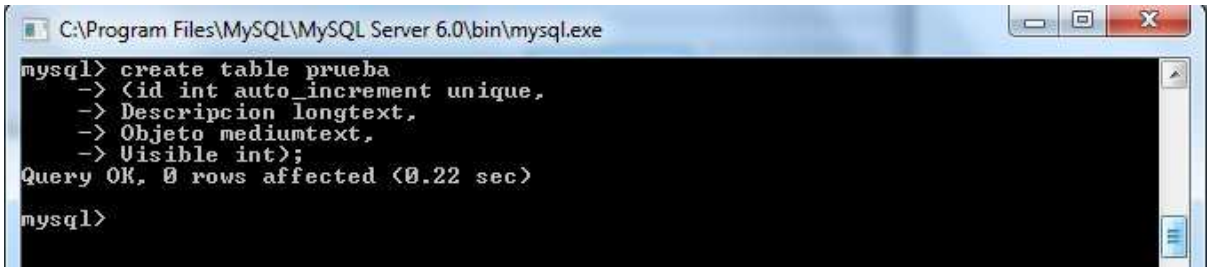

Figura 15: Creación BBDD.

Y después rellenaremos los campos.

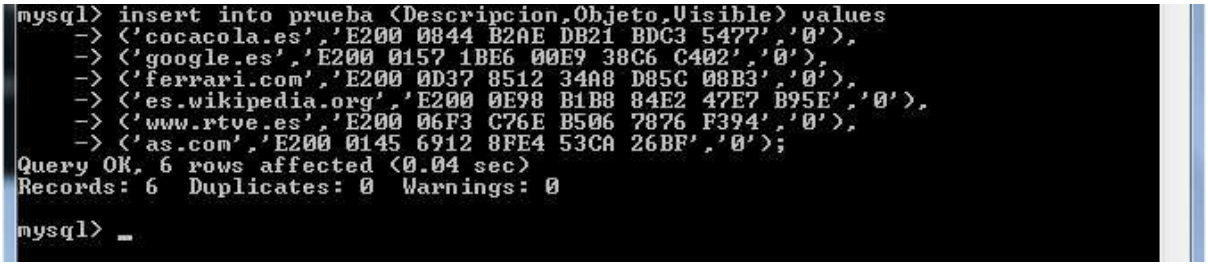

Figura 16: Rellenar campos.

Después de realizar estos pasos el resultado final quedará así.

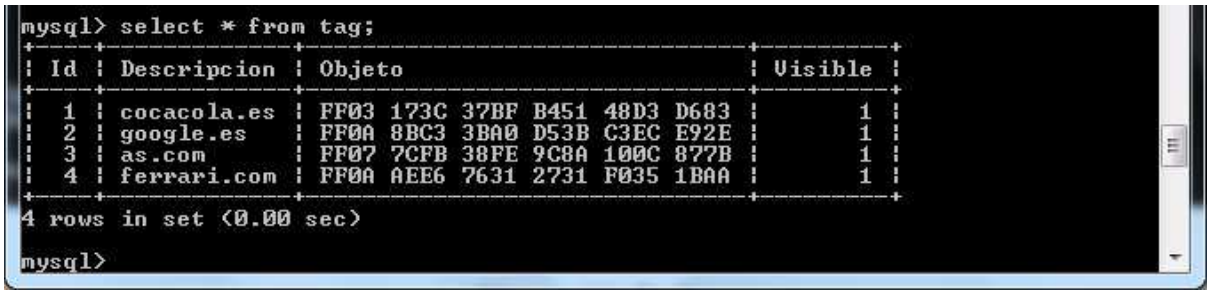

Figura 17: Tabla de etiquetas.

Ahora procederemos a cargar todas las etiquetas en el lector para que estén todas activas para la primera simulación.

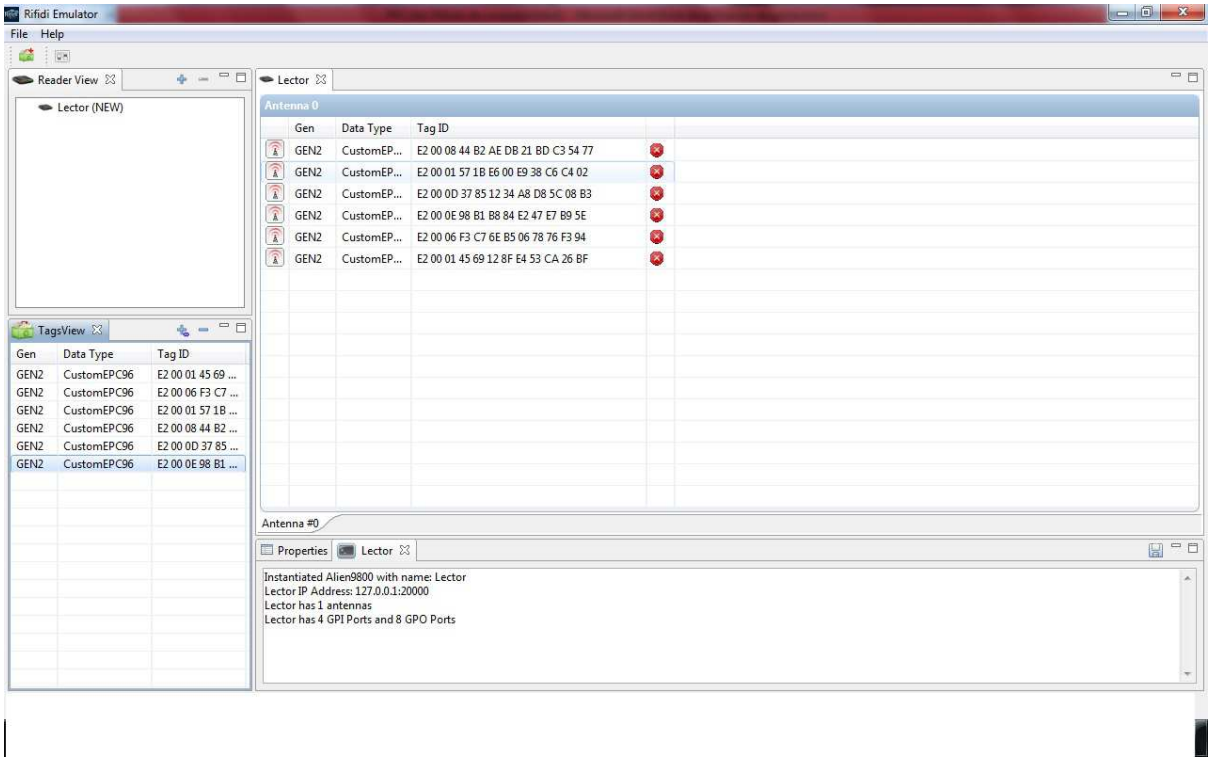

Figura 18: Etiquetas activas.

En la parte izquierda de la imagen podemos ver la pestaña de “TagsView” que muestra las etiquetas disponibles y en la derecha donde pone “Lector” son las etiquetas que el reader está leyendo en ese momento.

Despues de hacer esto nos iremos a la parte del servidor y lanzaremos las dos clases que poseen “main”. Una será la clase “LeerTag” que es la encargada de estar leyendo del reader y hacer las actualizaciones en la base de datos y la otra será “ServidorUDP” que estará siempre a la espera de peticiones por parte del móvil.

A continuación llega el momento de ejecutar la aplicación en el móvil. La primera pantalla que nos aparecerá será un botón el cual cuando sea pulsado se arrancará todo el proceso de conexión con el servidor.

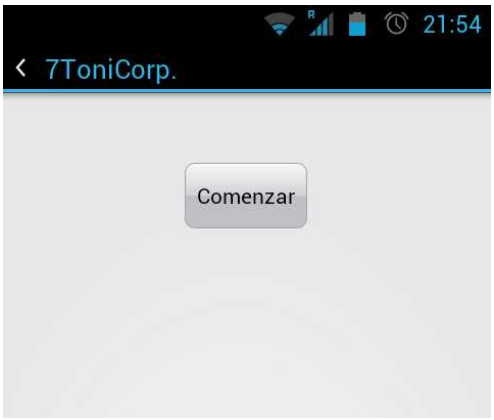

Figura 19: Primera pantalla móvil.

En la siguiente pantalla ya nos mostrará la lista de url's disponibles para poder acceder a cualquiera de ellas.

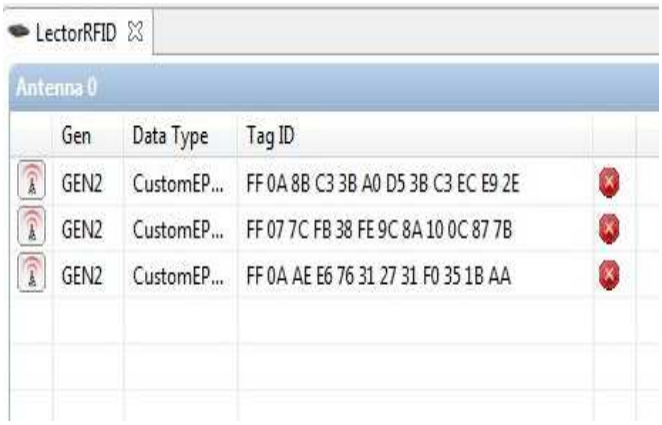

Figura 20: Etiquetas activas.

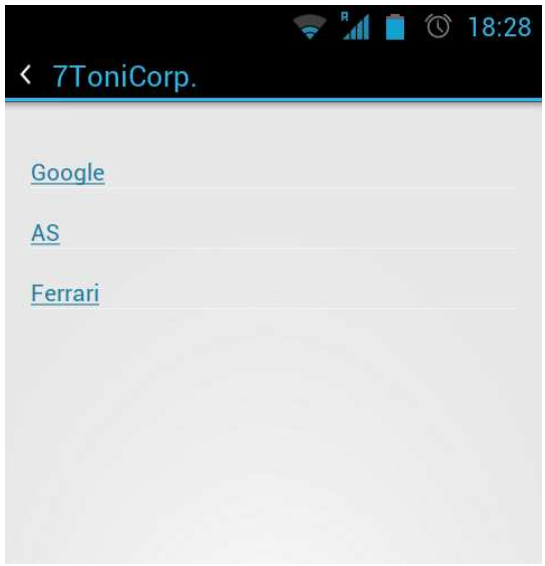

Figura 21: Lista de url's.

Como se puede comprobar en la Figura 20, en ese momento el reader está leyendo tres etiquetas activas cuya url asociada será enviada al móvil y mostrando los hipervínculos con las tres url's como se aprecia en la Figura 21.

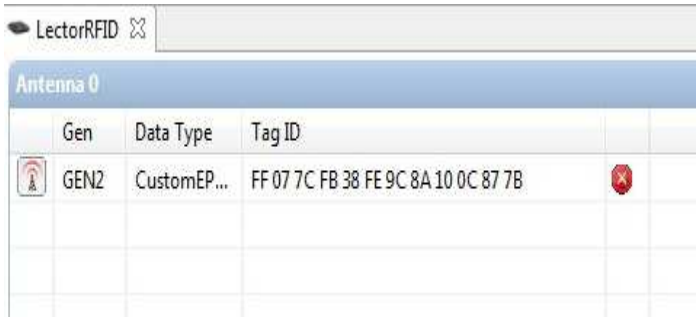

Figura 22: Etiqueta activa.

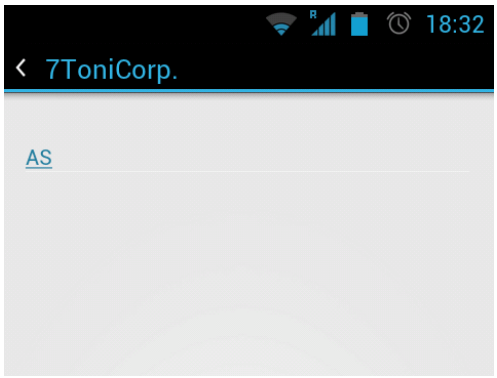

Figura 23: Lista url.

Como se puede apreciar en las siguientes imágenes han dejado de estar activas dos etiquetas y tan solo hay una en este momento, por lo que el móvil se autoactualiza y la aplicación por si sola muestra la url asociada a la etiqueta.

En el caso de que no haya ninguna etiqueta activa la solución realizada ha sido mostrar una página por defecto, en este caso la página de la UPCT para no mostrar una lista vacia.

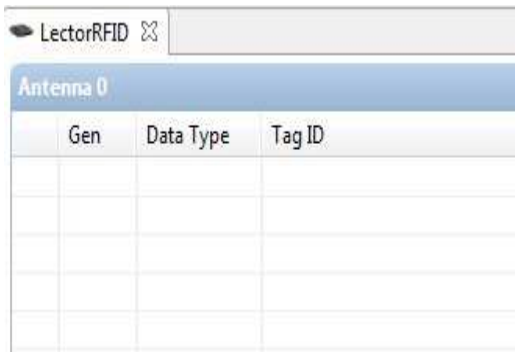

Figura 24: Etiquetas vacias.

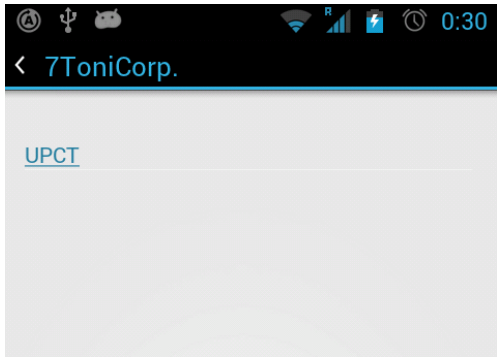

Figura 25: Url por defecto.

5.2 Resultados

En este apartado se van a explicar los resultados obtenidos por las simulaciones, principalmente los lapsos de tiempo necesarios. Como se ha podido ver en las imágenes anteriores el sistema funciona correctamente pero el sistema tiene un retardo desde que el tag es leído, actualizado por la base de datos, solicitado por el móvil y enviado finalmente ha este. En todo ese proceso transcurre un tiempo que es variable puesto que la actualización tanto de la base de datos como de la

aplicación móvil es cada cierto tiempo prefijado y no es lo mismo una detectar una etiqueta justo cuando la base de datos acaba de ser actualizada que cuando está en instantes antes de ser actualizada. Según varias simulaciones realizadas hemos obtenido una media de unos 18 segundos, desde que se añade o elimina alguna etiqueta hasta que se muestra finalmente por la pantalla del móvil. También es interesante decir que el tiempo mínimo de las simulaciones ha sido de 14 segundos y el máximo que hemos tenido que esperar fue de 23 segundos. Por supuesto, estos retardos se pueden reducir poniendo unas actualizaciones cada menos tiempo, pero hemos considerado que este tiempo de espera es adecuado, para no saturar al sistema con actualizaciones cada pocos segundos.

6. Conclusiones

6.1 Análisis crítico del proyecto

Una vez concluidas las tareas que forman este proyecto, es momento de realizar un balance y crítica de los resultados obtenidos. Comenzaré hablando sobre la parte de Android, sobre la cual a lo largo del desarrollo del proyecto se ha conseguido obtener un amplio conocimiento sobre este sistema operativo. Gracias a la extensa documentación de Google, se ha podido ir conociendo la arquitectura, funcionamiento, posibilidades ofrecidas, etc. En las fases iniciales es cuando se hace más necesaria esta información, y está totalmente disponible. Mencionar también que en los foros también se dispone información interesante sobre este sistema operativo y en ciertos momentos me ha sido de gran utilidad puesto que muchas veces mis dudas eran comunes a las dudas expuestas por otros programadores en los foros. Los comienzos no han sido fáciles, pero el conocer algo del camino, ya que tenía conocimientos de Java, se comienza a programar con mayor facilidad. Además, las partes de creación de interfaces de usuario, aunque desconocía el lenguaje xml, se podía hacer con elementos de diseño ya incorporados, lo que facilitaba bastante la tarea. A pesar de ello aprender un lenguaje nuevo siempre es complicado y los problemas surgidos han sido muchos, resueltos convenientemente conforme aparecían.

Para probar las aplicaciones se podía usar un emulador o directamente desde el móvil. Comencé usando el emulador pero resultaba tardaba mucho en cargarse y el proceso resultaba muy lento, por lo que al poco tiempo decidí instalar la aplicación en el móvil. De esta manera el proceso es rápido y mucho más ameno. Lo que también me ha resultado de gran utilidad ha sido el depurador de eclipse. Ya que en numerosas ocasiones la aplicación no funcionaba como era de esperar y era necesario buscar el fallo, y el depurador te deja ver el valor de todas la variables, ejecuciones paso a paso, edición de código y continuar ejecutando sin ser necesario parar la ejecución.

Sobre el resto de tecnologías usadas la dificultad no ha sido muy elevada puesto aprender a trabajar con bases de datos en la creación, inserción y modificación de tablas es bastante sencillo ya que no se requería una estructura amplia.

Desde mi punto de vista el concepto del proyecto me ha resultado bastante interesante. Si usamos tecnología RFID (que es inalámbrica) con un pc (es un dispositivo fijo) estamos muy limitados en cuanto a movilidad. Los datos llegan al pc pero tenemos que estar en esa ubicación siempre. Sin embargo, al usar un dispositivo móvil, podemos tener esos datos sin necesidad de estar siempre en el mismo lugar, lo cual, puede resultar muy cómodo en ciertos proyectos del mundo real. Y no solo eso, desde un pc estamos limitados a que un solo usuario pueda disponer de los datos, sin embargo, desde el móvil con la simple instalación del apk podremos tener varios usuarios a la vez disponiendo de estos datos.

Finalmente, considero que la aplicación puede resultar un tanto genérica, pero se tienen las bases para poder ampliarla y crear algo mas concreto sin necesidad del desarrollo de la conexión de todo el sistema. Tan solo sería necesario modificar la parte de android para adaptarla a una aplicación más concreta.

6.2 Posibles ampliaciones

Como posibles líneas de futuro, creo que este tipo de aplicaciones se les sacaría partido principalmente colocando los tags en objetos en movimiento o que se movieran o cambiaran de ubicación con cierta habitualidad. Como posible ejemplo podría decir una carrera de karts. Cada vez que un coche pasara por la meta se contarían el número de vueltas que lleva, tiempo por vuelta, distancia con otros pilotos…

También podría ser interesante en una cadena de montaje, por ejemplo, la fabricación de un coche. En todo momento podríamos saber en que sector se

encuentra el coche dependiendo de que antena sea la que reciba la información del tag, o para saber en una gran empresa en que parte de esta se encuentra cierto material o producto.

BIBLIOGRAFÍA

[1]http://www.android.com/
Página oficial de *Android*. Se encuentra toda la documentación de *Android*.

[2]http://www.mysql.com/

Página oficial de *MySQL*. Documentación de la base de datos *MySQL*.

[3]https://github.com/Androideity

Página para la búsqueda de código fuente que puede servir de ejemplo.

[4]http://www.tagingenieros.com/

Página para la búsqueda de casos de uso de aplicaciones RFID

[5]El gran libro de Android

Jesús Tomás Gironés

Marcombo, 2011

[6]Documentación y API’s proporcionadas por el fabricante del lector RFID.