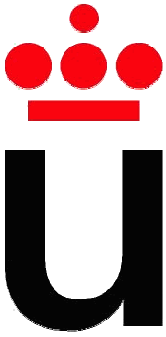

Universidad Rey Juan Carlos

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Curso Académico 2010/2011

Proyecto de Fin de Carrera

Guía Virtual Sensible al Contexto mediante Códigos de barras bidimensionales

Autor: Carlos Guzmán Ullán

Tutores: Estefanía Martín Barroso y Pablo A. Haya

Resumen

Hoy día, muchas personas no salen a la calle sin un teléfono móvil. Un teléfono móvil que ya no solo sirve para enviar y recibir llamadas, pues cada día se ha ido convirtiendo en el sustituto de otros aparatos tecnológicos como el reproductor MP3, cámara digital, GPS, etc.

Centrándonos en nuestro país, hay más líneas de teléfonos móviles que habitantes, y eso es un gran volumen de mercado. Por esto las compañías de teléfonos móviles cuyo objetivo comercial es el de captar usuarios, no paran de innovar y poner un *software* con mayores capacidades y un *hardware* más potente en nuestra palma de la mano, lo que permite explotar ese mercado tan gigante.

Con estas capacidades de *software* y *hardware*, y la movilidad total que nos otorga, es posible con un poco de ingenio y esfuerzo, realizar multitud de tareas o *hobbies*, a las que antes estábamos atrapados a hacerlas o desempeñarlas sobre un ordenador de escritorio. Incluso en algunos aspectos estos dispositivos han abierto nuevas barreras disponibles hasta entonces a unos pocos, como por ejemplo el uso de "realidad aumentada".

El objetivo de este proyecto es la realización de una guía digital sensible al contexto de la Universidad, cuya información se adapta en tiempo real y según el perfil del usuario y su ubicación física desde la que se ofrece la información.

El usuario podrá acceder a distintos contenidos gracias a la distribución por el campus de códigos de barras 2D o códigos QR. Los códigos QR al igual que los códigos de barras, almacenan información en forma de imagen monocromática. El código QR codifica la información en una matriz de puntos. Vamos a usar los códigos QR para guardar una URL, que posteriormente extraeremos de dicho código ayudándonos de una cámara. Esta URL es el enlace a un servidor de aplicaciones web que va a contener información generada previamente en un fichero XML, en el que podemos guardar un grueso mayor de información del que podría almacenar un código QR.

INDICE

ÍNDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

En esta sección plasmamos el objetivo final del proyecto de fin de carrera, la metodología de trabajo y la estructura que sigue esta memoria.

1.1 Objetivo

El objetivo del proyecto es la realización de una guía de digital de la Universidad, cuya información será accedida mediante la captura de un código QR. Dicha información es dinámica dependiendo del lugar desde el que se captura el código QR o del perfil del usuario que realiza dicha captura del código QR.

El proyecto de fin de carrera se basa en que un teléfono con sistema operativo Android, con cámara de fotos integrada y conexión a internet, capture una foto de un código QR, que contiene un enlace a un servidor. En el servidor se almacenan ficheros XML asociados los códigos QR. Los ficheros XML, contienen información diversa. Por ejemplo, para el registro de la Universidad podría contener los horarios de apertura, para la cafetería el menú del día o para los profesores los horarios de las tutorías. Además no es necesario cambiar el código QR cada vez que cambia la información, simplemente cambiando el contenido del fichero XML al que está asociado ese código QR hacemos visible la nueva información.

Para simplificar el proceso, a lo largo de esta memoria, nos centraremos en la información de los profesores. Sin embargo, la aplicación desarrollada serviría para cualquier servicio de la Universidad. En el caso concreto de los profesores, un profesor puede poner información de contacto, tutorías disponibles, conferencias, etc, y todo ello puede cambiar a lo largo del año o del mismo día. Lo que queremos hacer es acceder a la agenda del profesor, y para ello, mediante la API que Google proporciona para interactuar con muchos de sus programas, accedemos a la agenda/calendario del profesor que ha subido a su cuenta de Google Calendar. En esta agenda, se guardan los lugares y las horas de las citas o reuniones. Así de esta forma, si un profesor tenía una tutoría programada en su agenda/calendario a las 19:00 en el despacho 120 y tiene que cambiar el lugar de la tutoría o la hora, únicamente ha de actualizar dicha cita en su agenda/calendario de Gmail. Los alumnos que se acercaran al despacho 120 con su terminal Android, capturaran el código QR y veran en la pantalla de su terminal que dicha tutoría ha sido cambiada a otro día, a otro despacho, o cualquier tipo de información que se desee.

1.2 Metodología

El método de trabajo que se ha seguido para el desarrollo de este proyecto ha sido el siguiente:

a) Estudio de las posibilidades que ofrecen los códigos QR.
b) Estudio sobre las características de los dispositivos móviles.
c) Elección de plataforma de desarrollo.
d) Capacidades y limitaciones que ofrece la programación para Android.
e) Estructura e información que guardaremos en un fichero XML.
f) Recopilación de información sobre Android y sus APIs, que nos van a permitir implementar la aplicación.
g) Modo de interacción entre el dispositivo Android con el código QR.
h) Realización de las pruebas para comprobar el correcto funcionamiento y determinar si obtenemos la funcionalidad deseada.
i) Análisis de las pruebas.
j) Incrementos en la funcionalidad de la aplicación.

1.3 Estructura de la memoria

A continuación, se detalla la estructura de esta memoria:

En el apartado 1, **Introducción**, presentamos los objetivos y el método de trabajo seguido para su desarrollo.

En el apartado 2, **Elementos tecnológicos**, se exponen brevemente los componentes técnicos que son usados para el desarrollo del proyecto. Principalmente el proyecto está orientado a dos elementos, que son los códigos QR y la plataforma Android. En este apartado se detalla ampliamente ambos elementos, indicando una breve descripción, su funcionamiento y las posibles alternativas.

En el apartado 3, **Descripción informática**, se describe la implementación de las dos aplicaciones informáticas desarrolladas: una de ellas sirve para generar información y la otra para obtener dicha información.

En el apartado 4, **Conclusiones**, describimos cuáles han sido los logros alcanzados, las dificultades que hemos tenido durante el desarrollo del proyecto y los posibles trabajos futuros, como posible expansión del proyecto.

Por último, en el apartado 5, **Bibliográfica**, se presenta la bibliografía utilizada.

2 Elementos tecnológicos

El objetivo de esta sección es realizar una recopilación y un análisis previo de todos los elementos tecnológicos que intervienen en nuestra aplicación, como la descripción detallada y usos de los códigos QR, la forma de programar para Android, y nuestro tercer implicado en el proyecto, que es el lenguaje XML.

2.1 Tecnología usada

2.1.1 Hardware

Necesitamos un terminal o dispositivo que tenga acceso a Internet mediante *Wifi* o 3G y que disponga de una cámara digital.

Hoy día es fácil encontrar un terminal móvil con estas características, incluso tenemos capacidades en nuestro terminal móvil que sobrepasan estas necesidades, como por ejemplo el microprocesador, que corren actualmente a 1 GHz.

2.1.2 Software

Si por el *hardware* no estamos atados a un dispositivo especifico, por *software* sí que lo estamos, pues el proyecto ha sido desarrollado para Android.

Android es un sistema operativo de código abierto, basado en una versión modificada del Kernel de Linux, y los programas para dicho sistema operativo se escriben en Java.

2.2 ¿Qué son los códigos QR?

Los códigos QR, son un sistema para almacenar información en una matriz de puntos, similar a como funcionan los códigos de barras que estamos acostumbrados a ver en tantos productos, pero en lugar de en una sola dimensión, aplicado a dos dimensiones. En la figura 1, se puede observar un ejemplo de código QR.

Figura 1: Ejemplo de código QR

Los códigos QR, fueron creados en Japón en el año 1994 por la empresa Denso Wave. Esta empresa japonesa distribuye las especificaciones del mismo de manera libre y aunque posee una patente sobre el código QR, no ejerce los derechos sobre la misma.

La expansión que han tenido los códigos QR, se debe a que es un estándar abierto, por lo que cualquier persona es capaz de desarrollar una idea sobre ellos. Es importante resaltar que para su codificación no es necesario un *hardware* especifico, solamente una cámara de fotos. De hecho la combinación de teléfono móvil y cámara es perfecta para construir un decodificador de códigos QR.

Como los códigos QR, nacieron en Japón, su uso está muy extendido en este país y es raro que los móviles no tengan instalados algún tipo de *software* para decodificarlos.

Inicialmente se aplicaron al seguimiento de piezas en la fabricación de coches, pero poco a poco se han ido extendiendo a otros contextos.

2.2.1 ¿Cómo funcionan?

Su funcionamiento se basa en una matriz de datos, *datamatrix* o codificación de datos 2D, consistente en la codificación bidimensional que permite la generación de un gran volumen de información en un formato muy reducido, con una alta fiabilidad de lectura gracias a sus sistemas de información redundante y corrección de errores (legible hasta con un 20%-30% de *codeword* dañados). Un *codeword* es la unidad para construir un área de datos, en el caso de los códigos QR, un *codeword* son 8 *bits*. Además no es necesario un alto contraste para reconocer el código. El código está formado por celdas o módulos de color blanco y negro (perforadas en el caso de la micro percusión o no perforadas) que forman una figura cuadrada o rectangular. Cada una de esas celdas representa un *bit* de información.

2.2.2 Ventajas

Como hemos mencionado anteriormente, un código QR consiste en una codificación bidimensional, es decir, que es capaz de contener información en ambas direcciones (verticalmente y horizontalmente) a diferencia de los tradicionales códigos de barra (de una dimensión) como se muestra en la figura 2. Esto permite que la capacidad de almacenamiento sea mayor en el caso del código QR (así es posible almacenar 7089 caracteres numéricos o 2953 bytes).

Figura 2: Código QR vs Código de Barras [Ref. 1]

Además aporta otras características muy interesantes:

- Los códigos QR tienen la capacidad de corregir errores. El sistema de corrección de errores se basa en Reed Solomon.

Reed Solomon trocea la información en **n** bloques, y a partir de la misma, genera **x** bloques de información redundante. El receptor puede reconstruir los **n** bloques de información originales a partir de los **n+x** bloques de datos, siempre que no hayan desaparecido o se hayan corrompido más de **x** bloques. Un código muy popular de Reed Solomon es RS(255,233), donde cada palabra de código contiene 255 *bytes*, de los cuales 223 *bytes* son datos y 32 *bytes* son paridad (redundancia de información).

- Los códigos QR pueden ser leído a alta velocidad (Quick Response) desde todas las orientaciones (en 360º), gracias a los patrones de localización situados en las tres esquinas del símbolo.

2.2.3 Almacenamiento

Los códigos QR tienen 40 versiones diferentes. Cada versión tiene un módulo diferente de configuración o número de módulos. Un módulo son los puntos blancos y negros que componen un código QR.

La configuración del módulo, hace referencia al número de módulos contenidos en un código QR, que abarca desde la versión 1 (21x21 módulos) hasta la versión 40 (177x177 módulos), como se ve en el ejemplo de la figura 3. Cada número de versión mayor se compone de 4 módulos adicionales por cada lado.

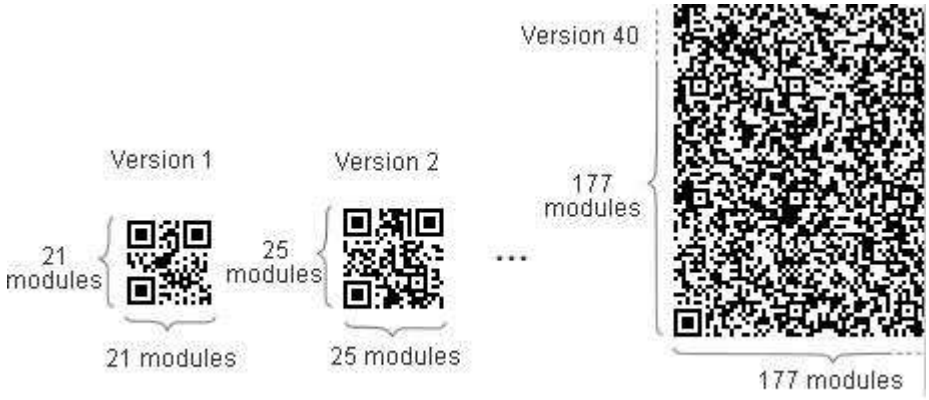

Figura 3: Versiones de códigos QR [Ref. 2]

Los códigos QR pueden guardar los siguientes tipos de datos:

1) Datos numéricos (0-9).
2) Datos alfanuméricos (0-9, A-Z y otros 9 caracteres: espacio, $, %, *, +, -, ., /,.
3) Bytes (por defecto ISO/IEC 8859-1).
4) Caracteres Kanji, compactados en 13 bits (caracteres de la escritura japonesa).

El sistema corrección de errores se basa en Reed Solomon y tiene 4 niveles:

1) L (low) bajo, puede corregir hasta el 7% de los *codewords* del símbolo.
2) M (médium) medio, puede corregir hasta el 15% de los *codewords* del símbolo.
3) Q (quality) calidad, puede corregir hasta el 25% de los *codewords* del símbolo.
4) H (high) alto, puede corregir hasta el 30% de los *codewords* del símbolo.

En la figura 4, podemos ver uno de los módulos individuales de los que se compone un código QR.

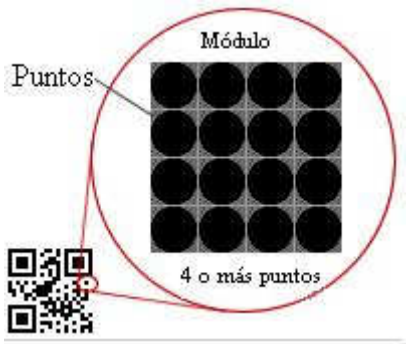

Figura 4: Módulo individual [Ref. 3]

2.2.4 Estructura del QR Code

Un código QR está formados por módulos (la unidad es el cuadrado blanco o negro, y representan el 0 y el 1 binario respectivamente) colocados en una estructura cuadrada.

La estructura cuadrada sigue un patrón en el que se diferencian las siguientes zonas:

- Patrón localizador: existe por triplicado en el código QR. Situado en las esquinas superiores y la esquina inferior izquierda. Sirve para calcular la orientación del código QR.
- Separador: separa los patrones localizadores del resto de zonas en el código QR.
- Patrón temporizador: secuencia de módulos blancos y negros que ayudan a calcular las coordenadas del resto de módulos.
- Patrón de alineamiento: permite re-sincronizar las coordenadas de mapeo de la imagen del código QR ante posibles distorsiones o ruidos en ésta.

En la figura 5, se muestra un código QR con las zonas mencionadas.

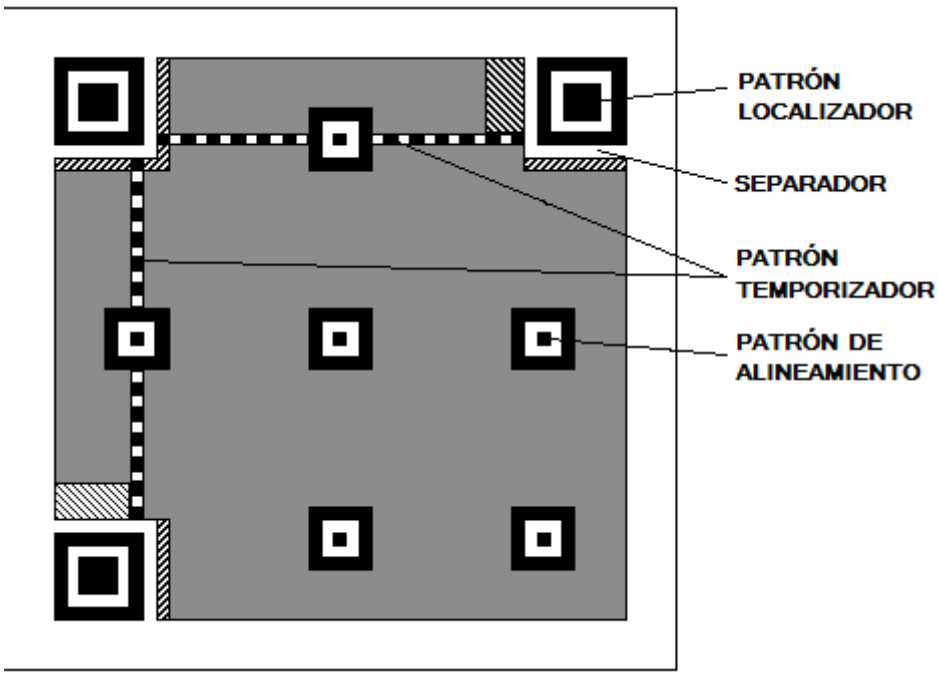

Figura 5: Estructura de un código QR [Ref. 4]

2.2.5 Alternativas a los códigos QR

A continuación presentamos algunas de las alternativas a los códigos QR, resumiendo sus principales características.

DataMatrix [Ref. 5]

Otro código de información bidimensional, es el **DataMatrix,** también estandarizado. La principal diferencia de un código QR con respecto de **DataMatrix**, es la velocidad con la que somos capaces de decodificar, siendo más lenta en **DataMatrix** que sobre un código QR. Sin embargo, sobre **DataMatrix**, somos capaces de almacenar mucha más información. Por estas razones, **DataMatrix** es más utilizado a nivel industrial y los códigos QR a nivel de usuario doméstico.

Microsoft Tag [Ref. 6]

El sistema de codificación de **Microsoft Tags** se basa en triángulos de colores y la información que contiene un **Microsoft Tag** es escasa, almacenando únicamente un identificador con el que buscar más datos en los servidores de Microsoft. Esto permite una mayor cantidad de información, a costa de depender de la conectividad del dispositivo, bien sea una PDA, Smartphones o Tablet PCs.

La aplicación **Microsoft Tag Reader**, mostrada en la figura 6, está disponible para Windows Mobile, pero también para el iPhone, BlackBerry, Symbian S60 y móviles con soporte para Java ME.

Figura 6: Microsoft Tag [Ref. 6]

Códigos Bidi [Ref. 7]

Al igual que Microsoft Tag, los **códigos Bidi** son un tipo de código de barras bidimensional desarrollado por **Movistar**, con el propósito de obtener mercado. Las posibilidades del **código Bidi** son prácticamente las mismas que puede ofrecer un QR Code**,** pero la licencia de uso es propiedad de Movistar. Dicho de otra manera mientras

que los códigos QR son gratuitos, estándar y existen infinidad de soluciones para su uso, generación y lectura **con coste cero**, los **códigos Bidi** están orientados a obtener "beneficios", ya sea desde la propia descarga del lector, en su uso, en la navegación web del móvil o en las descargas de contenido.

En la figura 7, se pueden ver las diferencias entre los códigos QR, los códigos *datamatrix* y los códigos Bidi.

Figura 7: Códigos monocromáticos

2.3 Android

Android es una plataforma móvil completa, abierta y libre. Fue un sistema operativo desarrollado inicialmente por Android Inc, una firma comprada por Google en 2005. Está basado en una versión modifica del *kernel* de Linux cuyo núcleo esta escrito en C y C++, pero las aplicaciones incorporadas o las diseñadas por el SDK de libre disposición, se escriben en Java.

El anuncio del sistema Android se realizó el 5 de noviembre de 2007 junto con la creación de la Open Handset Alliance, un consorcio de 78 compañías de *hardware*, *software* y telecomunicaciones dedicadas al desarrollo de estándares abierto para dispositivos móviles.

En tampoco tiempo ha visto incrementado su influencia en el mercado de los Smartphone y otros dispositivos como Tablet PCs, adquiriendo una cuota de mercado de más del 30% a escala mundial. Existen más de 300.000 aplicaciones para este sistema operativo y una de las causas de este importante crecimiento es que los desarrolladores tienen un *software* de desarrollo muy comprensivo, potente y con capacidades de crear aplicaciones complejas, sin coste alguno.

2.3.1 Elección de la plataforma Android

El desarrollo para algunas plataformas móviles es caro debido a los costes de los compiladores y del *software* de desarrollo, si lo comparamos con las aplicaciones que uno puede crear en Java o en C, con los compiladores y SDK de libre distribución que existen. En este terreno, Android rompe con esta política, ya que a diferencia de otras plataformas móviles, no hay costes para desarrollar aplicaciones para Android.

El SDK de Android, está disponible gratuitamente en el sitio web de desarrollo de Android. El entorno de desarrollo Eclipse, disponible también gratuitamente, se ha convertido en uno de los IDE (entorno de desarrollo integrado) más populares para el desarrollo de Android.

Como las aplicaciones de Android están escritas en Java, los desarrolladores están familiarizados con muchos de los paquetes proporcionados como parte del SDK de Android. Así la curva de nivel de aprendizaje para Android es bastante razonable.

Además existe una gran comunidad de desarrolladores compartiendo sus desarrollos y proponiendo ideas, de forma totalmente altruista.

2.3.2 Alternativas

En el mercado existen variedad de fabricantes *hardware* para móviles, y esta variedad de fabricantes también proporciona una amplia variedad de diversas plataformas de desarrollo con sus propios sistemas operativos para sus plataformas. A continuación se presentan distintas alternativas a Android, resaltando sus principales características.

MeeGo [Ref. 8]

Es la unión de los sistemas operativos Maemo y Moblin, con el que Intel y Nokia pretenden competir con el sistema Android de Google. El proyecto del nuevo sistema a diferencia de Android está auspiciado por la Linux Foundation.

MeeGo se presenta como un sistema preparado para funcionar en *netbooks*, dispositivos portátiles, sistemas en vehículos, televisiones y teléfonos multimedia. Básicamente se trata de una distribución Linux con soporte para ARM e Intel/Atom que usa Qt para su interfaz.

Symbian [Ref. 9]

Es un sistema operativo que fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Nokia, Sony Ericsson, Psion, Samsung, Siemens, LG, etc. Sus orígenes provienen de su antepasado EPOC32, utilizado en PDAs y Handhedls de PSION.

El objetivo de Symbian fue crear un sistema operativo para terminales móviles que pudiera competir con el de Palm o el Windows Mobile de Microsoft.

iOS [Ref. 10]

Anteriormente denominado iPhone OS, es un sistema operativo móvil de Apple desarrollado originalmente para el iPhone, siendo después usado en el iPod Touch e iPad. Es un derivado de Mac OS X, que a su vez está basado en Darwin BSD.

Las aplicaciones deben ser escritas y compiladas específicamente para la arquitectura ARM. Estas aplicaciones están escritas en Objective-C y es necesario pagar una cuota para obtener y poder desarrollar en el iPhone SDK.

webOS [Ref. 11]

Es un sistema operativo multitarea para sistemas embebidos basado en Linux, desarrollado por Palm Inc, ahora propiedad de Hewlett-Packard.

La interfaz gráfica de usuario de webOS fue diseñada para dispositivos con pantalla táctil. Usa tecnologías web como HTML5, JavaScript y CSS.

Windows Phone 7 [Ref. 12]

Es un sistema operativo móvil desarrollado por Microsoft, como sucesor de la plataforma Windows Mobile. Está pensado para el mercado de consumo generalista en lugar del mercado empresarial.

El desarrollo de aplicaciones para Windows Phone 7 puede hacerse empleando dos tipos de implementaciones, con Microsoft Silverlight o bien con Microsoft XNA Framework basado en una implementación nativa de .NET Compact Framework.

BlackBerry 6 [Ref. 13]

Es un sistema operativo desarrollado por Research In Motion, para sus dispositivos móviles. Esta claramente orientado a su uso profesional como gestor de correo electrónico y agenda, aunque la última versión incorpora muchas funcionalidades

que lo equilibran con otros sistemas operativos móviles enfocados a tareas multimedia o de ocio electrónico.

Se pueden desarrollar aplicaciones usando diversos SDK, como HTML y AJAX, Adobe Flash o Java; pero para tener acceso a ciertas funcionalidades del sistema operativo, se necesita ser firmado digitalmente para poder ser asociados a una cuenta de desarrollador de RIM.

2.4 XML

XML proviene de las siglas en inglés **eXtensible Markup Language**, es un lenguaje de marcas, o etiquetas desarrollado por la W3C, que permite definir la gramática de lenguajes específicos de una manera sencilla y estricta.

Juega un papel fundamental en el intercambio de archivos a través de la red. Es muy similar a HTML, ya que ambos lenguajes son adaptaciones del SGML, pero la función principal de XML es describir datos y no mostrarlos como es el caso de HTML.

2.4.1 Ventajas

Como su propio nombre indica es un lenguaje extensible, es decir, que después de diseñar el lenguaje, es posible extender/ampliar el fichero XML con la adición de nuevos elementos.

Mejora la compatibilidad entre aplicaciones, ya que dos o más aplicaciones escritas en lenguajes distintos, pueden usar ficheros XML para intercambiar información, por este motivo es tan popular en la red.

2.4.2 ¿Cómo es un fichero XML?

XML se desarrolló para dar solución al problema de expresar información de una manera estructura, por lo que interiormente un documento XML lo veremos como una estructura jerárquica.

Un documento XML tiene dos estructuras, una física y otra lógica.

Físicamente, el documento está compuesto por unidades llamadas entidades. Una entidad puede hacer referencia a otra entidad, incluyendo a esta en el documento. Cada documento tiene una única entidad raíz. Lógicamente, el documento está compuesto de declaraciones, elementos, comentarios, referencia a caracteres e instrucciones de procesamiento. Todos se encuentran definidos por una *markup* o marca.

Todo documento XML, en su primera línea ha de indicar que es un documento XML, en el que como atributos podemos indicar, la *versión* del lenguaje XML usada y la *codificación* utilizada.

En la figura 8, mostramos un ejemplo de cómo construir un correo electrónico en un documento escrito en XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<Mensaje>
  <Remitente>
    <Nombre>Nombre del remitente</Nombre>
    <Mail>Correo del remitente</Mail>
  </Remitente>
  <Destinatario>
    <Nombre>Nombre del destinatario</Nombre>
    <Mail>Correo del destinatario</Mail>
  </Destinatario>
  <Texto>
    <Asunto>Asunto del documento</Asunto>
    <Parrafo>Parrafo del documento</Parrafo>
  </Texto>
</Mensaje>
```

Figura 8: Ejemplo de un documento XML

Este mismo ejemplo puede mostrarse de una manera gráfica, donde se observa la jerarquía de las entidades del documento.

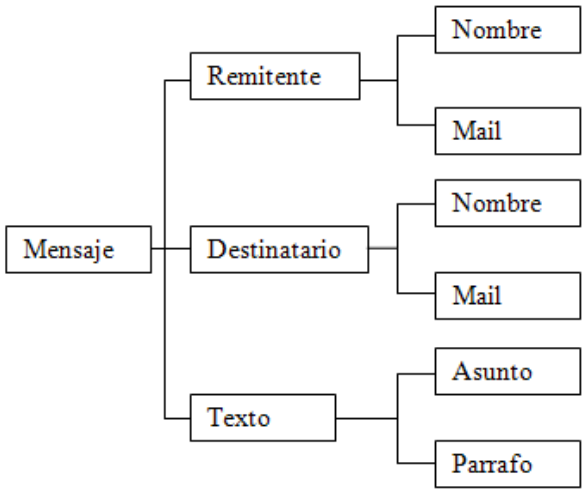

Figura 9: Documento XML representado gráficamente

2.4.3 XML y Android

Existe una fuerte relación entre XML y Android, ya que para programar en Android, es imprescindible el uso de documentos XML. Esto es debido a que por ejemplo, la manera de crear la interfaz de usuario, o los permisos de ejecución de nuestra aplicación, están contenidos en ficheros XML.

Una de las características más destacadas de la plataforma Android, es que se aprovecha el lenguaje de programación Java. El SDK de Android no termina de ofrecer toda la disponibilidad del estándar **Java Runtime Environment** (JRE), pero es compatible con una porción muy significativa del mismo.

Desde Java, somos capaces de trabajar de maneras diferentes con XML, y la mayoría de las API relacionas con XML de Java son totalmente compatibles con Android, como son los métodos más conocidos SAX o DOM.

Además de la conexión entre Android y XML, hemos decidido usar XML por la gran disponibilidad que ofrece a la hora de intercambiar archivos entre distintas plataformas, ya que no deja de ser un fichero de texto que se puede parsear conociendo las entidades existentes en el documento XML.

2.4.4 Analizando un fichero XML

En Java, podemos usar diferentes métodos para obtener la información contenida en el fichero XML, pero recordemos que estamos sobre la plataforma Android, que normalmente viene asociado a un *hardware* limitado en recursos de CPU y memoria, lo que nos va a condicionar la manera de analizar el documento XML. A continuación se presenta un resumen de algunos de los métodos de análisis de ficheros XML.

- SAX: Podemos usar el API de SAX si queremos que el programa de análisis sea rápido y queremos reducir el consumo de memoria de la aplicación. Esto hace que sea muy apropiado para la plataforma Android.

- DOM: Con DOM volcamos todo el documento a memoria y después hacemos búsquedas directamente de las entidades deseadas para recuperar los datos asociados. Es más sencillo que el método SAX. Sin embargo, DOM consume más memoria, ya que como hemos dicho, todo lo que se lee del documento XML se guarda en la memoria. Esto puede ser un problema en la plataforma Android, cuando el documento XML sea excesivamente grande.

- XML *Pull Parser*: Como hemos dicho, Android no tiene todo el API de Java, y por ejemplo no tiene las librerías para usar el método Stax, que es uno de los métodos más novedosos de obtener la información de un documento XML. Sin embargo, la plataforma Android tiene un analizador de documentos XML similar a Stax. Permite a la aplicación buscar eventos durante el análisis o lectura del documento XML, dicha lectura es secuencial, desde el principio del fichero hasta el final del mismo.

Resumidas las descripciones de algunos métodos de análisis de ficheros XML, tenemos que plantear cuál es el mejor método para nuestra aplicación. La primera elección seria usar SAX, ya que es muy rápido y fácil de implementar, pero el documento XML que nos va a devolver el API de Google Calendar es demasiado grande para conseguir un buen rendimiento con este método, por esta misma causa no podemos usar DOM en la aplicación de Android, ya que consumiríamos mucha memoria, y además no necesitamos toda la información del documento XML devuelto por Google Calendar. Así que nuestra mejor opción es utilizar XML *Pull Parser*, que realiza una lectura secuencial y nos permite quedarnos con aquella parte del fichero XML que nos interesa.

2.5 Google Calendar API

El API de Google Calendar permite a las aplicaciones ver, editar, crear o eliminar eventos existentes, buscar eventos por un criterio determinado, añadir eventos futuros de forma programada, o crear combinaciones con otras API de Google.

2.5.1 Tipos de Calendar feeds

Google Calendar distingue entre dos representaciones de calendarios, unos basados en la visibilidad del calendario y otro basado en la proyección. Cuando nos basamos en la visibilidad de un calendario, indicamos si este es público o no. Si nos basamos en la proyección de un calendario, indicamos la cantidad y tipo de información que vamos a representar en el calendario.

2.5.2 Event feeds

En el proyecto, sólo nos interesan los eventos que existan en un determinado calendario, el calendario del profesor o tutor, y dichos eventos pueden ser tutorías o clases, por ejemplo.

Así pues, para que no tengamos problemas desde el cliente (aplicación Android), a la hora de acceder a los eventos, su *visibilidad* ha de ser *pública,* y para obtener todos los detalles posibles que vengan, especificamos que su *proyección* es *completa*.

2.5.3 Obtener los eventos de un calendario

Para obtener los eventos de un calendario, necesitamos determinar la URL del evento que deseamos, teniendo presente lo descrito en el apartado anterior, y enviar una petición HTTP de tipo GET a la URL específica. Google Calendar devolverá un documento XML con la información del calendario sobre el que hemos hecho la petición, por ejemplo:

`GET https://www.google.com/calendar/feeds/userID/visibility/projection`

Donde *userID*, *visibility* y *projection*, toman valores según las características del calendario al que estamos accediendo y siguiendo las reglas proporcionadas por el API de Google Calendar.

A continuación mostramos dos tablas con los valores posibles y su descripción, tanto para el parámetro *visibility* (véase la Tabla 1), como para *projection* (véase la Tabla 2).

Visibility	Descripción	Actualizable	Seguridad
public	Muestra solo eventos públicos.	Sólo permite lecturas.	No requiere autorización para el acceso. Inaccesible si el usuario desactiva la opción de “calendario compartido”.
private	Muestra eventos públicos y privados.	Permite lectura y escritura.	Requiere autorización para el acceso. Actualizable solo para usuarios autorizados.
private-*magicCookie*	Muestra eventos públicos y privados.	Solo permite lecturas.	No requiere autorización para el acceso. La autorización está embebida en el *string* ***magicCookie*.**

Tabla 1: Valores para visibility- API Google Calendar [Ref. 14]

Proyección	Descripción	Actualizable
full	Contiene todos los eventos.	Permite lecturas y escrituras.
full-noattendees	Igual que full.	Permite lecturas y escrituras.
composite	Igual que full, pero el documento XML contiene más información.	Sólo permite lecturas.
attendees-only	Contiene mínima información del evento, pero podemos obtener quién convoca el evento.	Sólo permite lecturas.
free-busy	Contiene mínima información del evento, pero podemos obtener cuándo va a ocurrir.	Sólo permite lecturas.
basic	Información básica del evento sin ninguna otra extensión.	Sólo permite lecturas.

Tabla 2: Valores para projection – API Google Calendar [Ref. 14]

3 Descripción informática

El uso de dispositivos móviles se ha convertido en algo cotidiano de nuestra sociedad, una sociedad que cada vez quiere más información de una manera más simple, rápida y de manera concisa.

El objetivo de este proyecto es proporcionar información concisa a los profesores y sobre todo a los alumnos, de la disponibilidad de las tutorías, reuniones o eventos de los profesores del Campus.

Para conseguir este objetivo se han desarrollado dos aplicaciones, que ponen a disposición de los profesores y alumnos, la manera de crear esa información y la manera de obtenerla.

La creación de la información se realiza mediante una aplicación web, basada en un formulario. Este formulario debe ser cumplimentado por el profesor con el objetivo de crear un fichero XML en el que almacenaremos información como el nombre, despacho, dirección de la agenda del profesor, etc. Dicho fichero XML se alojará en un servidor web con un único nombre. A dicho nombre del fichero XML, se le asignará un código QR.

Inicialmente parece que la información generada es estática, es decir, una vez creada la información y almacenada en el fichero XML ésta no cambia. Sin embargo, en este proyecto hemos querido darle a esa información dinamismo. Por ello incluimos un apartado donde el profesor puede poner su dirección de agenda de Google Calendar, e ir modificando dicha agenda en tiempo real, lo que hará que a la hora de obtener la información no sea siempre la misma.

Hasta ahora no es nada nuevo, pues esto mismo se puede hacer mediante un navegador web y modificando una página HTML. Una pequeña pega del uso del navegador es, introducir la dirección web de la página HTML donde se vean reflejados los cambios. En este proyecto no usamos dirección web de la que el usuario se tiene que acordar o que alguien se la diga. Para sustituir esta necesidad, usamos los QR Codes, que contendrán la dirección embebida de nuestro fichero XML, como hemos mencionado anteriormente.

El poder distribuir a los demás ese QR Code, sigue siendo igual o más fácil que mostrar un enlace a una página web, pues se puede insertar como una imagen. Además podemos imprimir en una hoja ese código QR y pegarlo en la puerta del profesor o en un tablón de anuncios, por ejemplo.

La segunda aplicación se ocupa de obtener la información y mostrarla de una manera clara. Esta aplicación desarrollada para Android, es capaz de leer la información embebida en el código QR, conectándose al servidor web donde se encuentra alojado el fichero XML. Además de conectarse al servidor web, es capaz de extraer la información contenida en las etiquetas del fichero XML, por lo que es capaz de analizar ficheros XML. Esto nos abre una gran capacidad de crear y obtener información de manera estructura.

Como somos capaces de conectarnos a un servidor web y analizar un fichero XML, tenemos la capacidad de obtener la agenda de Google Calendar de una persona, pues Google pone a disposición un API para poder acceder a dicha agenda como si fuera un fichero XML.

Dentro de los ficheros XML que debemos parsear, puede existir una cantidad ingente de información, pero nosotros sólo queremos una información muy concisa para el propósito de nuestro proyecto, y poder mostrarla de una forma clara en nuestro dispositivo Android.

Resumiendo en pocas palabras el funcionamiento de la aplicación de Android, es la capacidad de leer un código QR, obtener un fichero XML alojado en un servidor web, analizar el fichero XML, extraer la información relevante del fichero XML y mostrar esta información en la palma de nuestra mano. Todo este proceso se realiza haciendo una simple foto al código QR.

3.1 Características

Al igual que la mayoría de las aplicaciones web que se ejecutan o acceden desde un navegador, este proyecto, se basa en una arquitectura de tres capas, ya que los distintos actores están distribuidos en tres bloques o capas, conectadas entre sí a través de una red (véase figura 10). Estas capas son:

Capa cliente: es aquella con la que interactúa el usuario de la aplicación. Sus funciones están limitadas a:

- Captura de datos de usuario y envío de estos a la capa intermedia.
- Presentación de resultados generados por la aplicación.

- Capa intermedia: contiene el núcleo de la aplicación. La aplicación se encuentra instalada en una máquina independiente conocida como servidor, a la que acceden los clientes a través de la red.

  Las funciones de esta capa son:

  - Captura de los datos enviados por el usuario desde la capa cliente.
  - Procesamiento de la información.
  - Generación de respuestas y envío de las mismas al cliente

  La comunicación entre las capas cliente e intermedia se realiza a través del protocolo HTTP.

- Capa de datos: tiene como misión el almacenamiento permanente de los datos utilizados por aplicación y la gestión de la seguridad de los mismos. En nuestro caso, debido a que no es un proyecto de grandes dimensiones, el almacenamiento de datos no se basa en un motor de base de datos relacionales, sino que guardamos la información en ficheros XML.

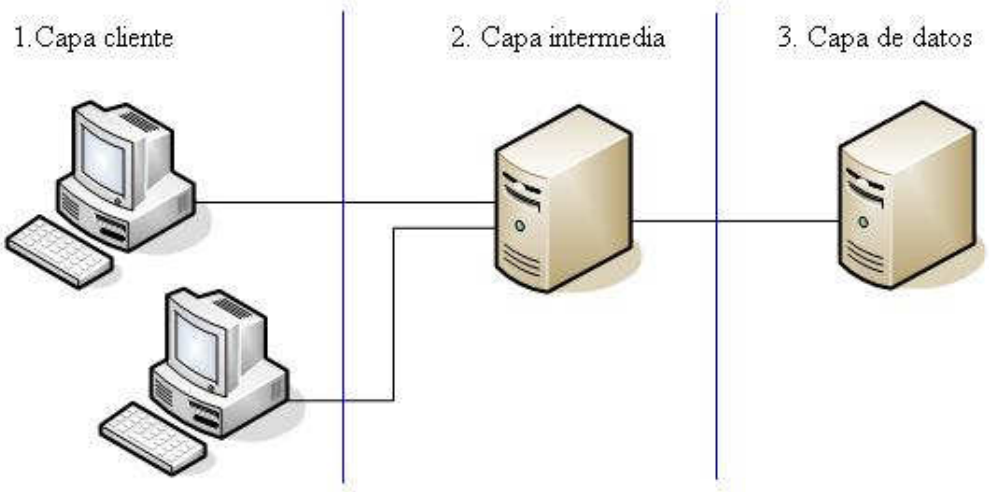

Figura 10: Modelo de tres capas

3.2 Aplicación web

Como se ha mencionado anteriormente, este proyecto se compone de dos aplicaciones. En este apartado vamos a detallar la implementación y uso de la aplicación web, que se encarga de **crear** los datos o información proporcionada por el profesor.

3.2.1 Implementación

Usamos la tecnología J2EE que nos proporciona un conjunto de especificaciones y librerías de clases idóneas para la creación de aplicaciones web de capa intermedia en Java.

Para realizar la captura de datos usaremos ***servlets***. Los ***servlets*** son componentes de la capa web que se utilizan para captura de datos de usuarios procedentes de la capa cliente y controlan el flujo de peticiones a la aplicación. Un servlets se implementa como una clase Java.

Como indica la especificación J2EE, las aplicaciones web, se han de organizar en una serie de directorios en los que se almacenarán cada uno de los componentes de la aplicación, tal y como se refleja en la siguiente figura:

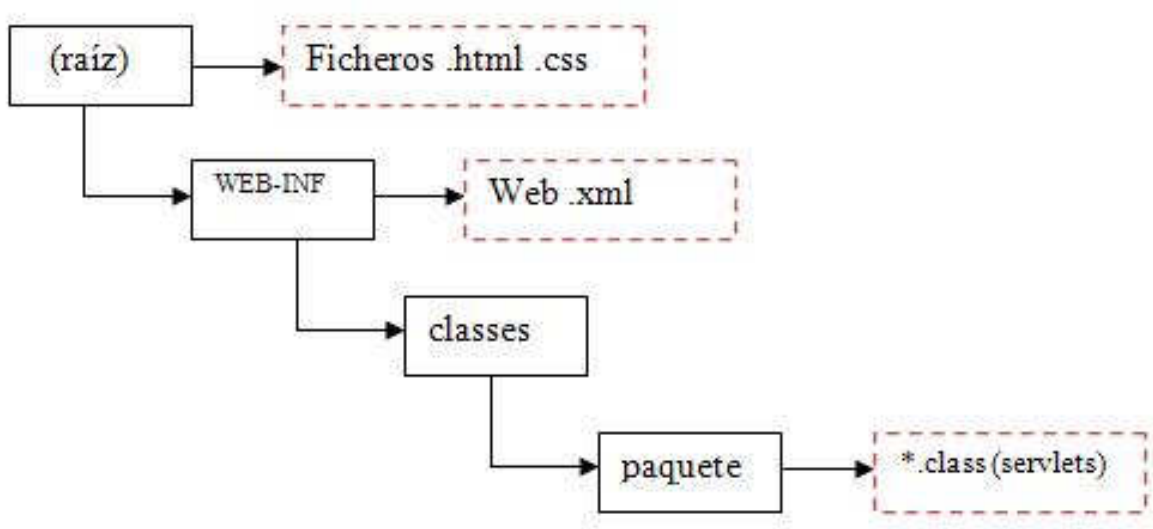

Figura 11: Estructura de la aplicación web

El contenido de estos directorios es el siguiente:

- Directorio raíz: Contiene los ficheros estáticos de la aplicación (páginas HTML, archivos de imágenes, páginas de estilo CSS, etc).

- Directorio WEB-INF: Contiene, además del subdirectorio de clases, al archivo de configuración de la aplicación. Este archivo es un documento XML, llamado web.xml, en el que se registra información sobre la aplicación.

- Directorio de clases: Contiene los ***servlets*** y demás clases Java utilizadas en la capa intermedia.

La estructura de directorios anterior con sus correspondientes archivos puede comprimirse en un fichero .**war**, para facilitar su distribución y despliegue en un servidor de aplicaciones.

Los ficheros principales de nuestra aplicación son los siguientes:

styles.css

Es nuestra hoja de estilo en cascada. CSS es un lenguaje usado para definir la presentación de un documento escrito en HTML o XML. Guardamos información de cómo representar una tabla y una cabecera.

error.html

Este documento web es mostrado en el navegador web en caso de que se produzca un error durante la creación/carga de datos/información. El propósito es advertir al usuario de que se ha producido un error.

exito.html

Este documento web, es mostrado en el navegador web en caso de éxito a la hora de crear el archivo XML, con la información que se ha rellenado en web_formulario.html.

web_formulario.html

Presenta al usuario un formulario para completar, y una vez finalizado, el usuario por medio de la pulsación de un botón llama al ***servlet*** (Get_Info.java) de la aplicación web que se encarga de la captura de datos.

Get_Info.java

Como se ha mencionado anteriormente, necesitamos un ***servlet*** para la captura de datos. Este *servlet* se encarga de capturar los datos que el usuario a introducido en web_formulario.html, procesarlos y enviarlos para su almacenamiento en el servidor de aplicaciones.

3.2.2 Diagramas

A continuación mostramos las dos secuencias posibles de la aplicación web: La primera de ellas se representan las acciones que el tutor ha de seguir para generar un fichero XML con la información deseada y su correcto guardado (ver figura 12). Estas

acciones son la conexión a la aplicación web, por parte del tutor desde un navegador web. Una vez conectado a la aplicación, aparece un formulario en el que el tutor ha de completar la información solicitada. La información completada es guardada en el servidor web en un fichero XML. Una vez finalizado el correcto guardado del fichero XML, al tutor se le presenta en el navegador un enlace para generar el código QR que contendrá la dirección del servidor donde se ha guardado dicho fichero XML.

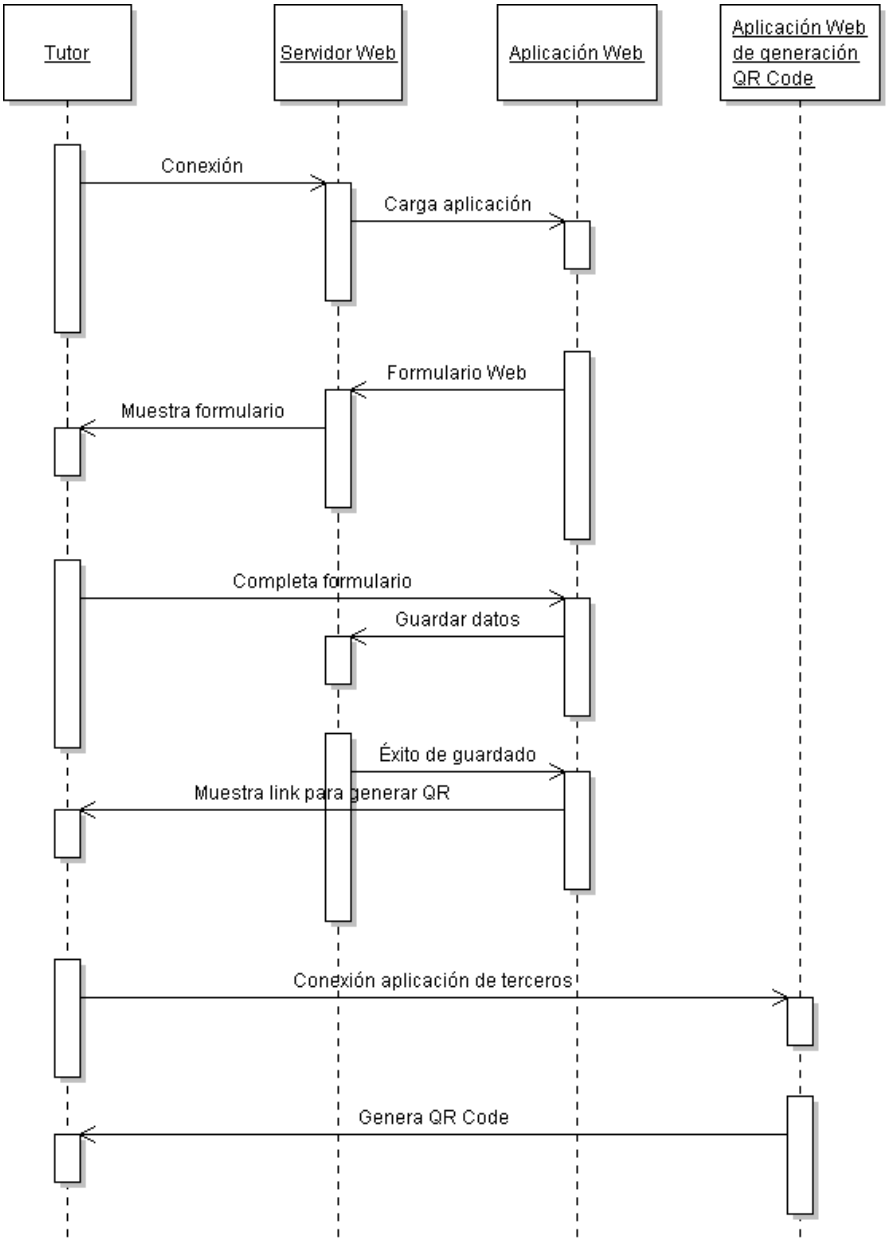

Figura 12: Diagrama de secuencia (éxito)

En la figura 13, se expone una posible secuencia de fallo provocada por duplicidad de información.

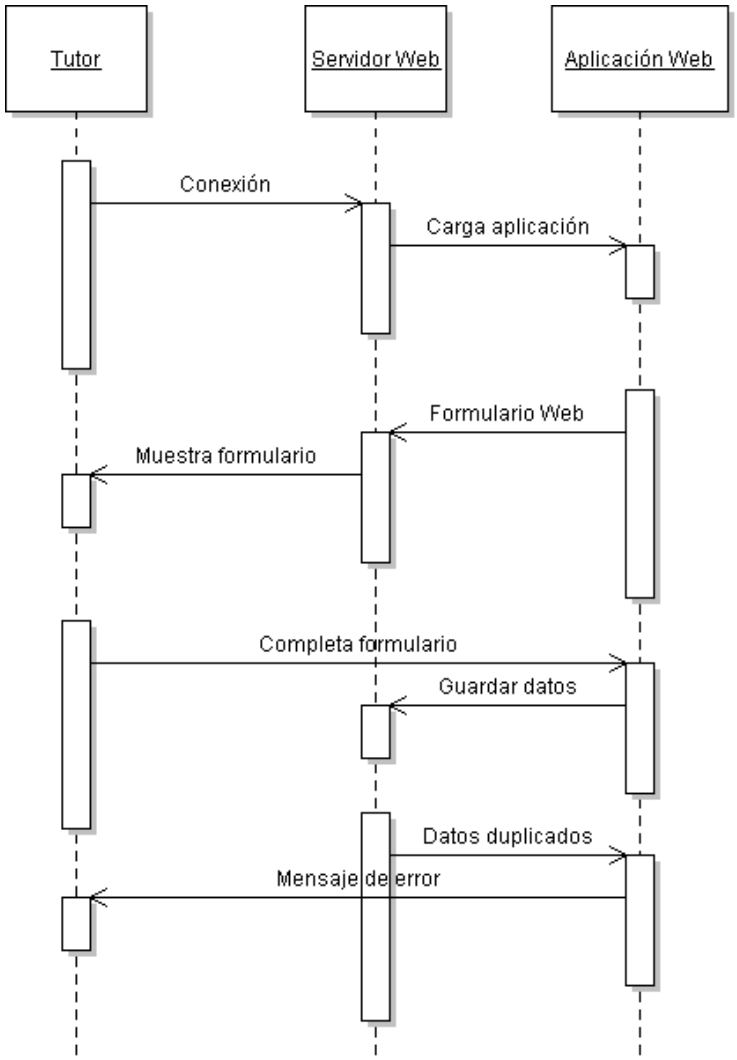

Figura 13: Diagrama de secuencia (fallo)

3.2.3 Ejemplo de uso

En los ejemplos de uso, mostramos con capturas de pantalla, el uso de la aplicación web, y al igual que en el punto anterior, en el que se mostraron las dos posibles secuencias, aquí también mostramos un ejemplo de éxito y un ejemplo de fracaso.

La figura 14, muestra el aplicativo web una vez ha sido cargado por el servidor. En esta página mostrada por la aplicación, el usuario (tutor) completará los datos para poder generara un código QR.

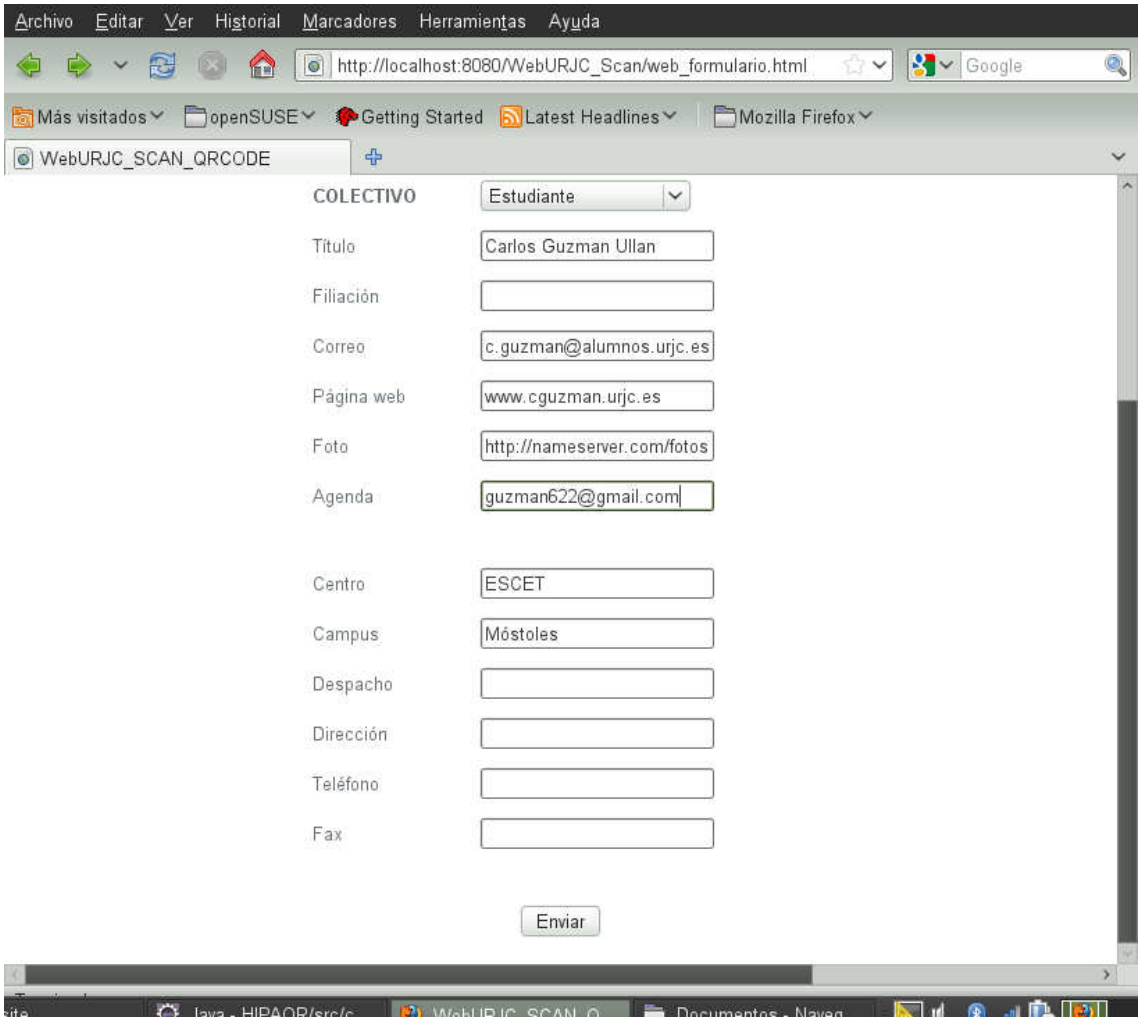

Figura 14: Formulario de la aplicación web

Una vez el usuario ha completado todos los datos del formulario, debe pulsar el botón “Enviar”, para que los datos sean guardados en el servidor web, y así posteriormente, mediante la aplicación Android, los podremos recuperar.

Tras este paso, el fichero se ha guardado en el servidor con el nombre CARLOS_GUZMAN_ULLAN.xml, que es la información completada en el campo de “Título”.

Una vez tenemos el fichero generado, el siguiente paso es generar el código QR, para ello nos ayudamos de una aplicación de terceros que genera un código QR (ver figura 15). El código QR que generemos debe contener o almacenar la dirección siguiente: http://www.nameserver.com/CARLOS_GUZMAN_ULLAN.xml, donde *nameserver.com*, será el nombre del servidor en el que esté nuestro fichero XML.

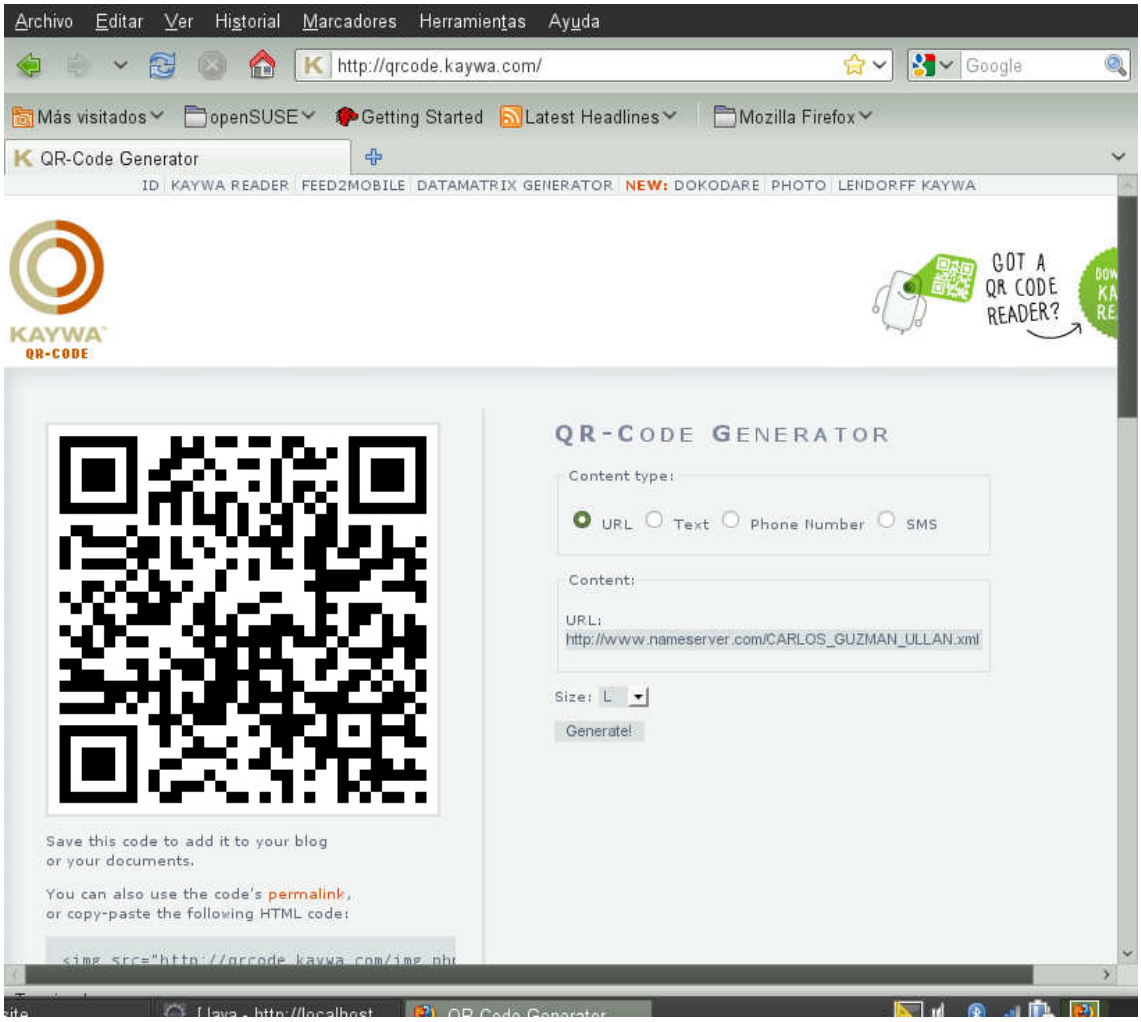

Figura 15: Creación de código QR

3.3 Aplicación Android

Es el momento de describir la funcionalidad de la otra aplicación que interviene en el proyecto, y que se encarga de **obtener** la información que ha sido generada anteriormente.

3.3.1 Implementación

Vamos a realizar una breve introducción de cómo y en qué se basa el desarrollo de una aplicación para el entorno Android. En esencia, Android podríamos decir que se basa en dos cosas: un núcleo de Linux y una máquina virtual de Java que es capaz de interpretar muchas de las librerías estándar de Java, así como las del API Android.

El **núcleo de Linux**, proporciona una capa de abstracción sobre el *hardware*, la gestión de procesos, la gestión de memoria y la gestión de energía.

Además del núcleo de Linux, disponemos de distintas **librerías**, que dan posibilidad de ser usadas por los programas para la gestión de tareas multitactiles, reproducción de contenido *multimedia* y un pequeño motor de base de datos SQLite.

Para ejecutar estas librerías, necesitamos algo sobre el que lanzarlas/ejecutarlas, es lo que se conoce como el ***run time* de Android**. Este ***run time***, engloba:

- ***Core libraries***: Estas librerías proporcionan la mayoría de las funcionalidades disponibles en el lenguaje de programación Java.
- ***Dalvik virtual machine***: Es una máquina virtual de Java, pero optimizada para dispositivos móviles.

Pero un programador no necesita llegar a tan bajo nivel del Android para programar, ya que el propio sistema operativo Android, nos ofrece capas a más alto nivel para que la programación sea más fácil. Estas capas son:

- ***Application framework*** que nos sirve las distintas clases para crear aplicaciones Android.
- ***Application layer***, que es la capa donde caen casi todas las aplicaciones que son creadas para Android.

3.3.2 Diagramas

Como se puede observar en la figura 16, el alumno tomara una fotografía a un código QR que vea pegado en los despachos de los profesores o en algún panel informativo. La aplicación Android decodificará el código QR para obtener una URL en la cual existe alojado un fichero XML con mucha más información. Al servidor web donde se encuentra alojado este fichero XML, se le hará una petición de que nos envíe el fichero XML. Así, si por parte de este servidor no existe ningún problema, la aplicación Android, recibirá el fichero XML y se dispondrá a analizar el contenido del mismo. Imaginemos que este fichero XML tiene un *tag* o elemento, que hace referencia a una agenda de Google Calendar, entonces nuestra aplicación Android, también realizará una petición al servidor de Google, para que nos envíe el contenido de esta agenda. Como Google Calendar, nos enviará el contenido en forma de fichero XML, nuevamente desde la aplicación Android, analizaremos el fichero que hemos recibido y juntando con la información del primer fichero XML, mostraremos toda la información por la pantalla del dispositivo Android de la forma más clara posible.

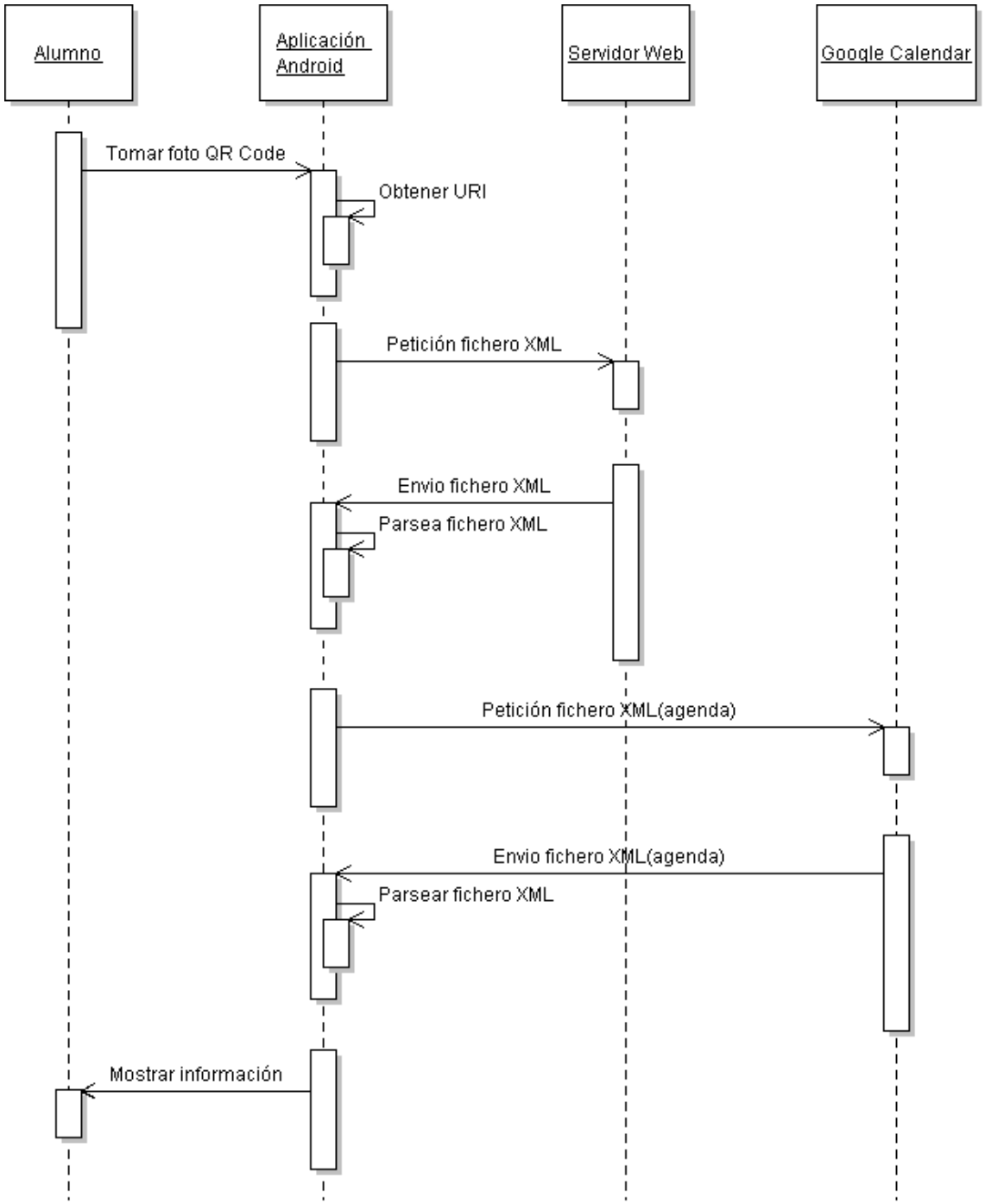

Figura 16: Diagrama de secuencia en la aplicación desarrollada para Android

3.3.3 Ejemplo de uso

Ahora, mediante capturas de pantallas mostramos el ejemplo de uso de la aplicación para Android, y también enseñamos un pequeño aporte de funcionalidad que nos da la aplicación al permitirnos hacer la consulta de la agenda del tutor/profesor, del "Día actual" o de la "Semana actual".

En esta captura de pantalla mostrada en la figura 17, se presenta la vista principal de la aplicación, donde vemos que tenemos la posibilidad de pulsar sobre 4 botones: escanear un código QR, acceder al menú de preferencias, información sobre la aplicación y salir de la aplicación.

Si el usuario pulsa sobre el botón “Scan QR”, se inicia el reconocimiento de un código QR.

Figura 17: Inicio de la aplicación de Android

La figura 18 es una instantánea de lo que está capturando la cámara del móvil o dispositivo donde tenemos instalada la aplicación.

Figura 18: Reconocimiento de código QR

Una vez reconocido el código QR, se nos presenta la información a semejanza de la figura 19.

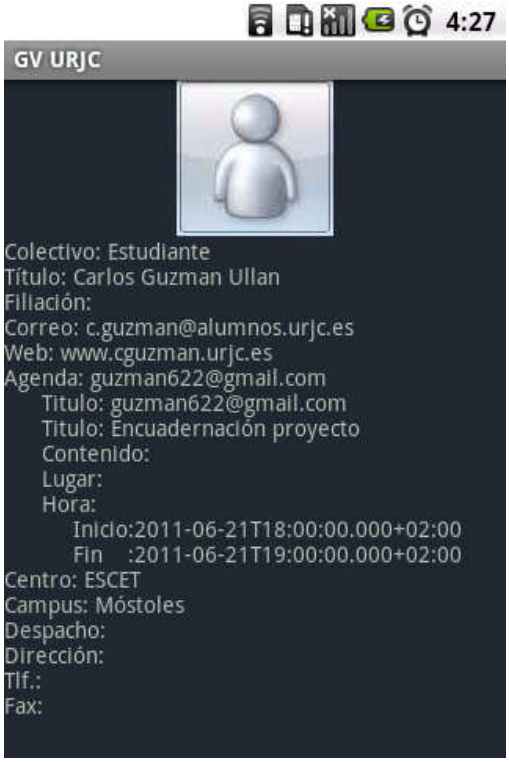

Figura 19: Información del día actual extraída de un código QR

Como se observa, sólo aparece la agenda de un día. Esto es configurable desde el menú principal de la aplicación (véase la figura 20). Tenemos que seleccionar el botón de “Preferencias”, y elegir el ***check box*** de “Semana actual” (ver las figuras 20 y 21).

Figura 20: Menú preferencias

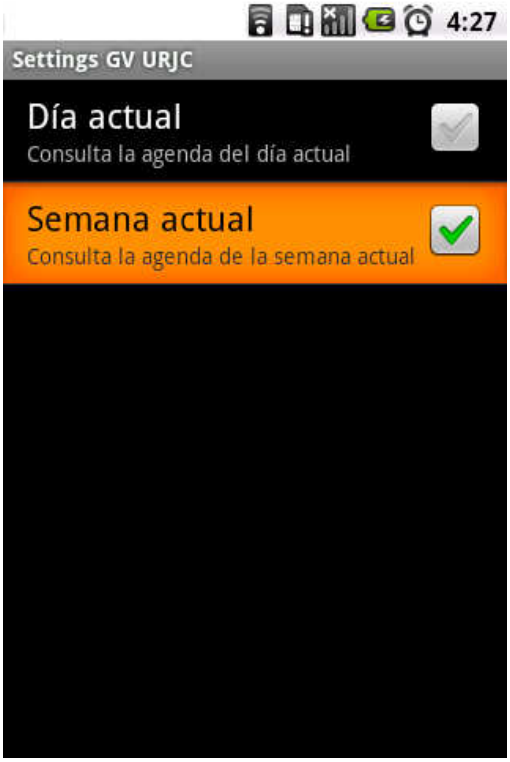

Figura 21: Preferencias - Semana actual

Si volvemos a procesar el reconocimiento del mismo código QR y tenemos eventos, futuros o pasados correspondientes a la semana actual, también serán mostrados (ver la figura 22).

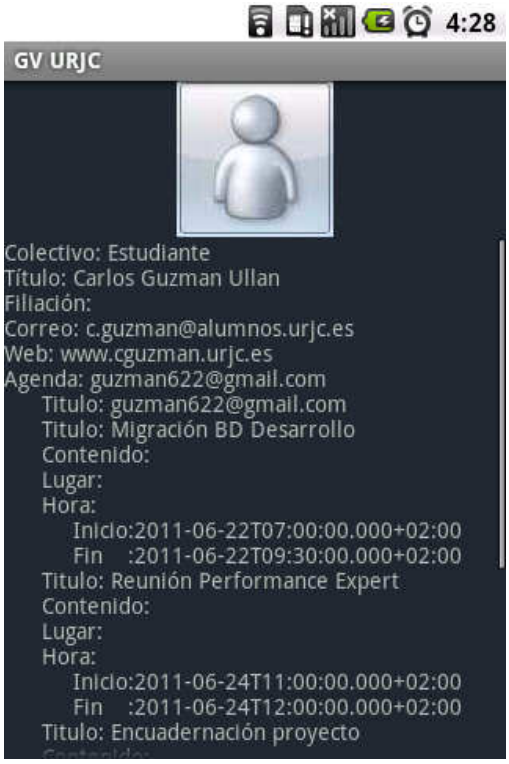

Figura 22: Información de la semana actual extraída de un código QR

4 Conclusiones

4.1 Logros alcanzados

Se ha conseguido realizar una guía sensible al contexto, cumpliendo así el objetivo del proyecto. Hemos focalizado el problema propuesto sobre un área muy determinada, quedándonos solamente con información de los tutores o profesores, pero no supondría un gran esfuerzo ampliar la funcionalidad del proyecto, incluyendo otro tipo de información, pues la idea de capturar una fotografía de un código QR y analizar el contenido que éste nos proporciona es ampliable a muchas áreas como se comentó en la introducción del proyecto.

Además de conseguir el objetivo deseado, el proyecto nos ha ayudado a ver el ciclo de vida de una aplicación, desde la concepción de la idea, pasando por la implementación, mejora y finalmente por la documentación.

Hemos descubierto la plataforma Android, la comunidad de *software* libre que rodea a esta plataforma, y las cualidades y bondades que nos aporta esta plataforma abierta.

También hemos descubierto los códigos QR y las capacidades que nos aportan.

4.2 Dificultades

La mayor dificultad encontrada es el primer contacto con la plataforma Android. Aunque se programa en Java, no llega a ser un Java SE, y difiere un poco su forma de programar.

Superado el problema de la plataforma Android, hay que pensar siempre en que estamos desarrollando dos aplicaciones que han de poder “comunicarse” o más bien entender una de ellas a la otra, para el éxito del proyecto. Así pues aunque seguramente se pueda ampliar mucho el aplicativo web, debemos ser consciente en que todo lo que pongamos de información, la aplicación de Android debe ser capaz de mostrarla en un tiempo razonable.

4.3 Trabajos futuros

Dos posibles mejoras para la implementación del proyecto serían desarrollar un enlace directo desde la aplicación de Android para que el usuario de la aplicación pudiera escribir un correo al tutor.

Otra posible mejora, sería la implementación de un sistema de notificaciones automáticas, es decir, una vez realizada la captura del código QR, nos podríamos subscribir a las notificaciones del tutor al que corresponde el código QR, así si el tutor cambia el lugar, el día o la hora de una tutoría o reunión, el alumno sería avisado del cambio sin necesidad de volver a capturar el código QR.

5 Bibliografía

5.1 Libros

- Antonio Martín Sierra: Programador Certificado JAVA 2
- Ed Burnette: Hello, Android – Introducting Google´s Mobile Development Plataform
- Reto Meier: Professional Android 2 Application Development

5.2 Webs utilizadas para el desarrollo

- http://code.google.com/p/zxing/
- http://developer.android.com/index.html
- https://groups.google.com/group/desarrolladores-android?hl=es
- http://code.google.com/intl/es-ES/apis/calendar/

5.3 Referencias

- [Ref.1] http://www.denso-wave.com/qrcode/aboutqr-e.html
- [Ref.2] http://www.denso-wave.com/qrcode/qrgene2-e.html
- [Ref.3] http://www.denso-wave.com/qrcode/qrgene3-e.html
- [Ref.4]http://es.wikipedia.org/wiki/Archivo:C%C3%B3digo_QR_Ejemplo_de_Estructura.svg
- [Ref.5] http://es.wikipedia.org/wiki/Matriz_de_datos
- [Ref.6] http://tag.microsoft.com/consumer/index.aspx
- [Ref.7] http://www.codigos-qr.com/tag/datamatrix/
- [Ref.8] https://meego.com/
- [Ref.9] http://licensing.symbian.org/
- [Ref.10] http://developer.apple.com/technologies/ios/

- [Ref.11] https://developer.palm.com/
- [Ref.12] http://msdn.microsoft.com/es-es/ff380145
- [Ref.13] http://us.blackberry.com/developers/
- [Ref.14] http://code.google.com/intl/es-ES/apis/calendar/data/1.0/reference.html