UNIVERSIDAD
REY JUAN CARLOS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

INGENIERÍA DE TELECOMUNICACIÓN

Proyecto Fin de Carrera

BURJC FUENLABRADA
APLICACIÓN ANDROID PARA LA BIBLIOTECA DE LA UNIVERSIDAD REY JUAN CARLOS DE FUENLABRADA

Autor : Alfonso Noriega Vizuete

Tutor : Gregorio Robles

Curso Académico 2013/2014

Índice general

Índice de figuras

Capítulo 1

Resumen

Hoy en día es impensable para la mayoría de nosotros la vida sin teléfonos inteligentes. Ocupan nuestra rutina, desde la lectura diaria del periódico o contestar correos en cualquier lugar, hasta el uso de herramientas profesionales que nos son fundamentales para el trabajo.

Esto se debe, en gran medida, a la aceptación de estos dispositivos entre el público en general. Se trata de una tecnología sencilla e intuitiva que está apoyada por un concepto atractivo: una pantalla táctil. Además, hay que tener en cuenta la aparición de potentes plataformas de desarrollo que permiten que tanto fuera como dentro de las empresas, desarrolladores de todo el mundo, puedan crear aplicaciones que cubran las necesidades más diversas.

El objetivo de este proyecto es el diseño e implementación de una aplicación para *smartphones*. Además, se le ha querido añadir una funcionalidad extra: la realidad aumentada.

Los servicios ofrecidos por estos dispositivos han ido evolucionando en complejidad. En muchos casos esto se ha conseguido, apoyándose en tecnologías como la cámara de fotos. Es precisamente la cámara de fotos y el análisis de video en tiempo real lo que ha dado lugar a la aparición de la realidad aumentada. Una tecnología que aporta información adicional a la obtenida en el mundo físico, superponiendo a la información real la información virtual.

Es por eso que en este proyecto hemos querido proporcionar una herramienta móvil para la biblioteca de la Universidad Rey Juan Carlos de Fuenlabrada que suponga un acceso ágil e intuitivo a la información que esta aloja en su página web y además, permita mediante una funcionalidad basada en realidad aumentada, localizar un libro dentro de su sala de lectura de la planta baja.

Capítulo 2

Introducción

Este proyecto consiste en el desarrollo de **BURJC Fuenlabrada**. Este es el nombre que se le ha dado a una aplicación para *smartphones*, basada en el sistema operativo **Android**. Se ha elegido como versión mínima para el desarrollo la versión Froyo (Android 2.2). Se puede acceder a la aplicación en el siguiente enlace:

https://play.google.com/store/apps/details?id=shockes.burj

A lo largo de todo el proyecto vamos a utilizar como dispositivo principal para las pruebas de la aplicación el smartphone Sony Xperia Arc S (LT18i). También se han realizado pruebas sobre el terminal Samsung Galaxy S III (GT-I9300) durante gran parte del desarrollo, aunque no ha estado siempre presente. La motivación de utilizar dos terminales para pruebas, viene justificada por las diferencias en tamaño de display y configuración técnica, siendo más potente y teniendo un tamaño de pantalla mayor el terminal de Samsung. Esto me permitía comprobar como se comportaba la aplicación respecto a tiempos de carga/descarga de datos y los tiempos de procesado en dos terminales de potencias distintas.

En cuanto a implementación, el proyecto va a contar con dos partes bien diferenciadas. En primer lugar, vamos a tener todas las funcionalidades desarrolladas sobre Android o sobre librerías externas proporcionadas Android, Google Play Services. En segundo lugar, vamos a tener las funcionalidades desarrolladas sobre librerías externas no proporcionadas por Android, como Jsoup o la SDK de Metaio, que nos permite desarrollar las funcionalidades para la **realidad aumentada**.

El desarrollo de toda la aplicación se hará utilizando como lenguajes principales Java y XML.

2.1. Motivación del proyecto

Los dispositivos móviles en general y los *smartphones* en particular, se están convirtiendo en pilar fundamental de nuestras rutinas, ya sea en lo personal o en lo profesional. Esto supone una motivación para trabajar con ellos y realizar desarrollos de aplicaciones nuevas, que proporcionen a los usuarios servicios interesantes.

Por otro lado, la realidad aumentada se está postulando como el futuro más reciente en cuanto a aplicaciones móviles se refiere. Es por eso que queríamos darle este enfoque al proyecto. En primera instancia, la intención era la utilización de una librería basada en código abierto para el desarrollo del motor de la RA, pero una vez que elegimos el tipo de solución que se consideró más apropiada, se vio que el desarrollo necesario era excesivo y por eso se seleccionó la SDK de Metaio que es de distribución gratuita y se adecua perfectamente a nuestras necesidades.

A todo esto, hay que sumarle la intención de la biblioteca de la Universidad Rey Juan Carlos de Fuenlabrada de apostar por las nuevas tecnologías, así como el entusiasmo y la predisposición que mostraron a colaborar en el desarrollo del proyecto. La participación de la biblioteca, de forma activa durante todo el proceso, le daba una cierta perspectiva realista al proyecto.

Desde mi punto de vista, juntando todos estos elementos tenemos como resultado un proyecto interesante que nos permite explorar tecnologías punteras que además, se van a poder exportar a una plataforma que tiene una importante presencia dentro del mundo de los smartphones como es Android. Y todo ello, acompañado por un cliente real e ilusionado que va a colaborar activamente en el diseño de la aplicación y que va ayudar a tomar el pulso al proyecto con una fase final de pruebas, que culminará con una encuesta de satisfacción.

2.2. Android

Como se ha comentado, Android es el sistema operativo móvil elegido para el desarrollo de este proyecto. Android es el sistema operativo ideado principalmente para sistemas móviles con pantalla táctil y basado en kernel Linux. Estos sistemas comprenden básicamente a los

smartphones y las tabletas, pero poco a poco estamos empezando a ver sistemas Android en otros dispositivos.

Android fue creado originalmente por la compañía Android Inc. Desde el principio, Google demostró un gran interés en el proyecto que demostraba realizando un apoyo activo en el desarrollo. Finalmente en 2005, Android Inc., es adquirida por la empresa del buscador pasando a formar parte su oferta de servicios.

El sistema operativo del androide es presentado por primera vez en 2007 junto a la Open Handset Alliance. Un consorcio de empresas de hardware, software y telecomunicaciones que se dedican al desarrollo de estándares abiertos. Desde entonces, su grado de penetración en el mercado no ha dejado de crecer. Desde el punto de vista del usuario se ha acuñado la reputación de sistema sencillo y altamente personalizable. Mientras que desde el punto de vista de los fabricantes, se hacía atractiva la posibilidad de ofrecer sus propias interfaces de usuario, *launchers*, que se ejecutaban directamente sobre Android; además de la sencilla portabilidad de Android sobre cualquier hardware. En las siguientes gráficas podemos observar la evolución entre 2012 y 2013 siendo ya un periodo en el que estaba ampliamente asentado.

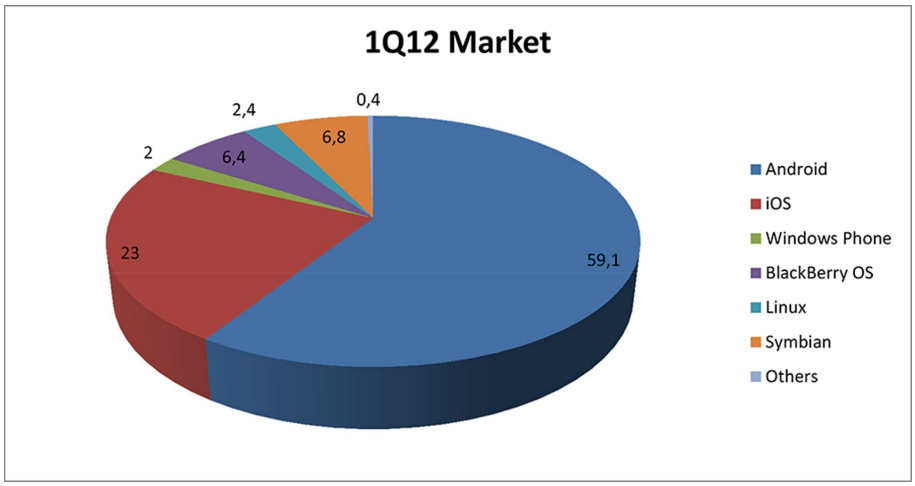


Figura 2.1: Cuota de mercado en el primer cuatrimestre de 2012

En la fig. 2,1 podemos ver como Android ya se impone claramente sobre el resto de sistemas operativos teniendo una cuota de mercado superior al 59 % . Se puede observar en la figura 2,2 como en el primer cuatrimestre del siguiente año, la cuota de mercado a la que Android tiene acceso ha crecido notablemente en detrimento de la mayor parte de sus competidores, alcanzando el 75 % del mercado global de *smartphones*.

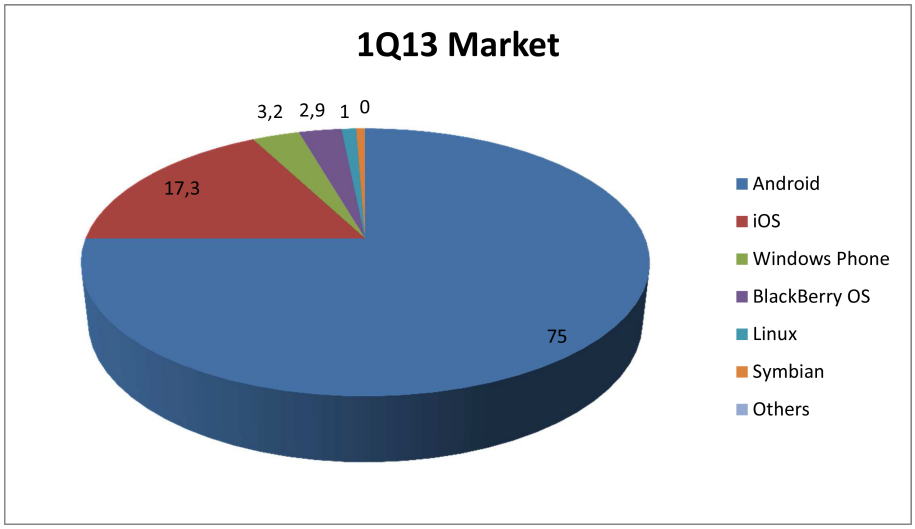


Figura 2.2: Cuota de mercado en el primer cuatrimestre de 2013

2.2.1. Características

Según la filosofía de Google, Android te da la libertad de desarrollar cualquier tipo de dispositivo y sus drivers. A partir de ese momento es el HAL (*Hardware Abstraction Lawyer*) el que te proporciona una manera estándar de desarrollar el software entre la plataforma Android y el hardware. Parte del éxito de Android, es que al ser un sistema de código abierto, es más fácil el desarrollo de drivers y su adaptación.

Para tener una perspectiva general de cómo se comporta el sistema Android [4], en la siguiente figura 2,3 podemos ver como está diseñado por capas desde su interacción con el hardware hasta la capa de aplicación o *aplication framework*.

Haciendo un recorrido por capas:

- **Aplication framework**: Este es el nivel que más concierne a los desarrolladores. Para trabajar en esta capa Google proporciona diferentes API en función de las versiones de Android, que en la mayoría de las veces mapea uno a uno las interfaces del HAL lo que resulta una información muy útil para implementar los drivers.

- **Binder IPC**: El mecanismo *Binder Inter-Process Communication* permite al *aplication framework* hacer la gestión de procesos, así como las llamadas a los servicios de Android. Esto, básicamente, permite que APIs de alto nivel interactúen con los servicios del sistema Android. Desde el punto de vista del *aplication framework* esto es transparente.

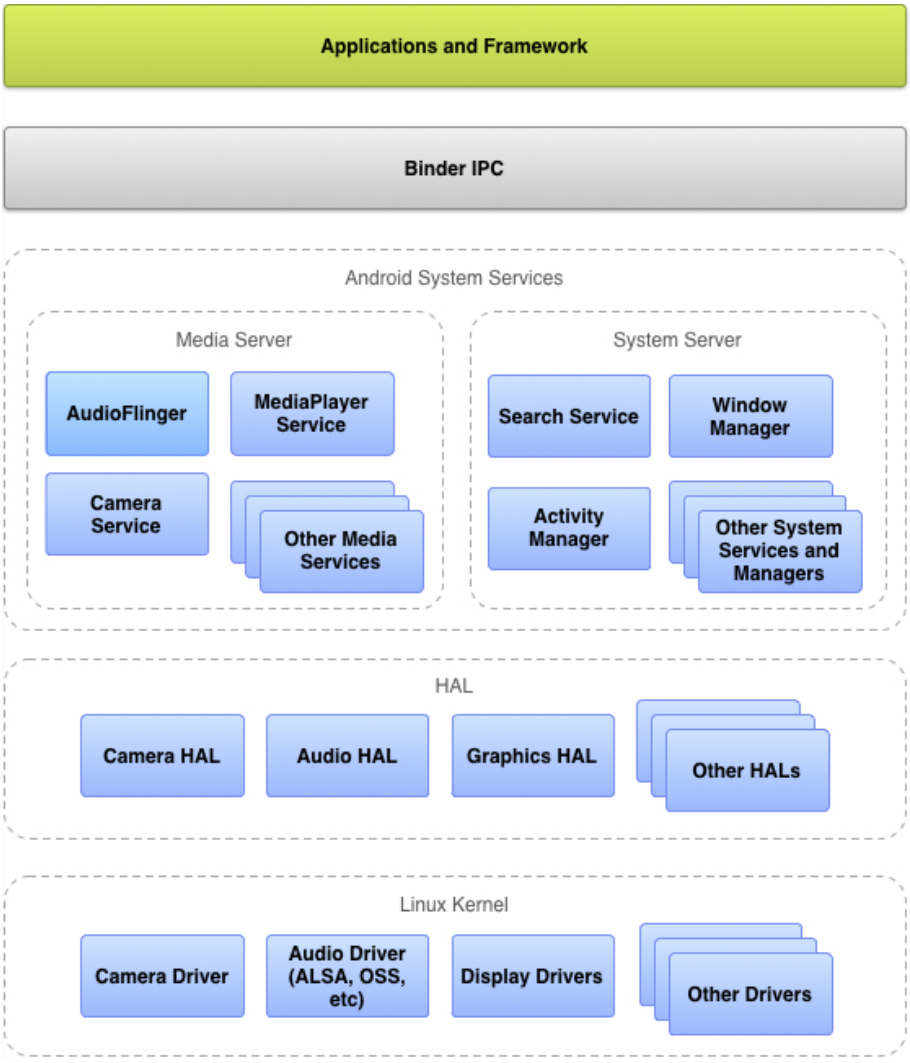


Figura 2.3: Arquitectura del sistema Android [5]

- **Hardware abstraction layer (HAL)**: El HAL funciona como una interfaz estándar que permite al sistema Android llamar a la capa de los controladores. Desde el punto de vista de una nueva plataforma hardware, se deberían de desarrollar tanto la capa de drivers, como el HAL. Esto es así, porque Android no define una forma estándar de interactuar entre los drivers y el HAL, así que el fabricante tiene la libertad de adaptar estas dos capas de la forma que le sea más conveniente.

- **Linux Kernel**: En la mayor parte de los casos, desarrollar drivers para un dispositivo que pretenda usar Android debería ser lo mismo que desarrollar un driver para uno con un sistema Linux típico. Android utiliza una versión específica del kernel Linux con algunas características especiales, como un sistema de gestión de memoria más agresivo que

ayuda a conservar más cantidad de memoria libre; o el Binder IPC y otras características importantes para las plataformas en las que se instala un sistema Android embebido. Sin embargo, estás características especiales están más relacionadas con cómo se comporta el sistema, que con el desarrollo de los drivers.

2.3. Realidad aumentada

La realidad aumentada (AR, *Augmented Reality* en inglés) es la fusión de la realidad física con la realidad virtual, donde gráficos creados por ordenador se fusionan con la realidad en tiempo real. AR crea la ilusión, de que objetos virtuales existen en el mundo real. Estos objetos virtuales tienen la misión de aportar una información adicional a la recibida por el entorno real. Esta información extra tiene la capacidad de ser modificada, borrada y recuperada [1].

Actualmente, existen tres técnicas distintas de visualización:

- **Head Mounted Display (HMD)**: Suele consistir en un dispositivo a modo de casco que dispone de un display a la altura de los ojos. El usuario verá el mundo a través de este display, tanto las imágenes del mundo físico, como las digitales de la realidad aumentada.

- **Handset Display**: Es todo aquel dispositivo que nos quepa en la palma de la mano y nos permita capturar las imágenes reales y mostrarlas a través de un display, donde veremos superpuestas las imágenes de AR. Es decir, estamos hablando de que, al menos, ha de existir una cámara y un display. En la actualidad los dispositivos con mayor potencial para esta técnica son el *smartphone* o la *tablet*, por que reúnen las características de movilidad y además, presentan una capacidad cada vez más sobresaliente de procesado. A pesar de esto, se han desarrollado aplicaciones de realidad aumentada en dispositivos PDA, reproductores mp3, etc.

- **Spatial Augmented Reality (SAR)**: La realidad aumentada espacial tiene una diferencia fundamental frente a las dos anteriores técnicas y es que, en este caso, las imágenes virtuales son proyectadas directamente sobre los elementos físicos. Esta técnica permite a un mismo grupo de usuarios ver la misma información sin que cada uno deba tener una pantalla asociada a nivel individual.

En la actualidad, la realidad aumentada está en auge y están saliendo aplicaciones con un sinfín de variedades. Atendiendo a algunas de las distintas disciplinas existentes:

- **Educación**: Dentro del panorama de la educación se podrían hacer libros de texto más interactivos donde los alumnos pudieran explorar de una forma diferente lo aprendido en clase. Por ejemplo, en lecciones de anatomía. Se podrían realizar lecciones prácticas con coste cero de desplazamientos y materiales utilizando aplicaciones de realidad aumentada.

- **Turismo**: Por ejemplo, para acompañar las visitas guiadas en un museo aportando información adicional a las obras solamente con enfocar al cuadro con la cámara de nuestro *smartphone*.

- **Cirugía**: A partir del análisis de video el cirujano va recibiendo información en tiempo real de lo que está viendo o por ejemplo, de donde debe cortar.

- **Mecánica**: Permitiendo al mecánico a través de un análisis de video en tiempo real recibir información adicional sobre las piezas que está viendo.

- **Localización**: Se puede usar realidad aumentada para guiar a un usuario desde un punto a otro sobrescribiendo información virtual en la pantalla. Es más, se puede añadir información adicional como los restaurantes de la zona, el tiempo, etc. En este sentido existen varias aplicaciones que son muy populares como pueden ser Juanio, Layar o Wikitude.

2.3.1. Localización *indoor*

A día de hoy son ampliamente conocidos y utilizados los sistemas de posicionamiento basados en el GPS (Global Positioning System, en inglés). Son sistemas que, ayudados por la información de los satélites GPS, nos dan información de donde estamos y en el caso de los navegadores por ejemplo son capaces de llevarnos a otro punto geográfico distinto de forma guiada.

Aunque en la localización exterior se ha avanzado mucho y existen muchos medios de navegación, es en la localización en interiores o *indoor location* donde estamos encontrando más problemas para alcanzar los niveles de funcionalidad conseguidos en exteriores. Siguiendo el

ejemplo de los sistemas GPS, es habitual que dentro de edificios la señal sea muy mala o inexistente y por tanto poco fiable .

Centrándonos en la realidad aumentada, para guiar a un usuario dentro de un espacio cerrado podemos utilizar dos estrategias:

- Utilizar un sistema de marcadores que reconozcan los patrones y dibujen flechas en la dirección a seguir: se utiliza en visitas guiadas a edificios oficiales o museos, ver figura 2,4. Es decir, el sistema reacciona en tiempo real al reconocer un marcador.

Figura 2.4: Arquitectura del sistema Android

- Utilizar un sistema basado en el posicionamiento del móvil, es decir emular el funcionamiento de los sistemas en exteriores. Nosotros hemos optado por este último.

Existen diversas técnicas que pretenden predecir la posición en la que nos encontramos dentro de un edificio. Tal es el caso de los *indoor positioning system* (IPS), que pretenden, a través de una red de dispositivos inalámbricos cuya posición geográfica exacta conocemos, calcular la posición del móvil mediante una triangulación. La naturaleza de estos sistemas es diversa: Wifi, Bluetooth, elementos ópticos o sonoros. En primera instancia pensamos en poder utilizar esta técnica, pero el error relativo era próximo a 2 metros lo que excedía generosamente el margen

de nuestra tolerancia. La distancia entre estanterías dentro de la biblioteca es alrededor de dos metros, por lo que un error de esas dimensiones confundiría totalmente al usuario aportando información engañosa.

En nuestro caso vamos a utilizar una solución implementada por Metaio basada en marcadores LLA (Latitud Longitud Altitud). Como se puede observar en la fig. 2,5, tiene una apariencia parecida a los códigos QR. En ellos esta codificada información sobre la latitud, longitud y altitud del punto donde hemos fijado el marcador. De esta forma tan sencilla, con solo leer el código LLA, el móvil va a ser capaz de determinar su posición en un entorno *indoor*.

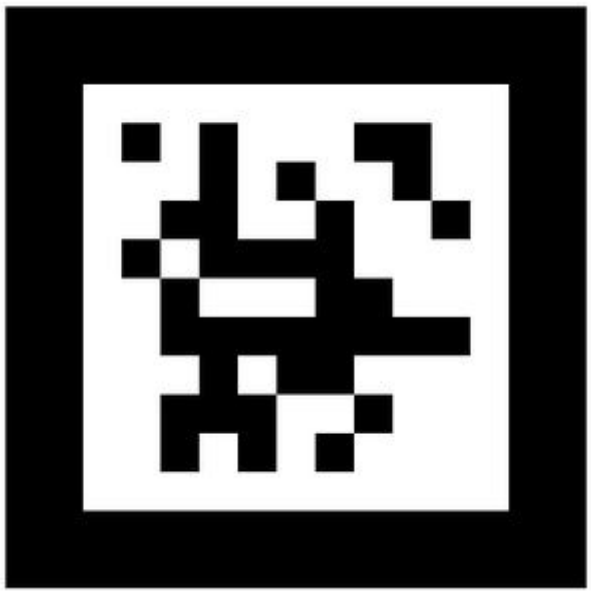


Figura 2.5: Arquitectura del sistema Android

En el momento en el que el dispositivo móvil realiza la lectura del código LLA ya conoce su posición, solo resta conocer la posición destino para poder orientarnos. En nuestro caso, las posiciones destino serán las de las distintas estanterías de la biblioteca, que al ser elementos inmóviles tienen una posición geográfica conocida y fija. Con esto datos ya tenemos todo lo que necesitamos para orientar al usuario de su posición a la posición final.

Según se vaya desarrollando el proyecto, se irá facilitando información más detallada del funcionamiento del sistema

2.4. Herramientas utilizadas

En este apartado vamos hablar de las herramientas utilizadas: tanto librerías externas, como el SDK de Android o de Metaio

2.4.1. Jsoup

Jsoup es una librería de Java que trabaja con HTML. Provee de un amplio API para la extracción y manipulación de datos, usando lo mejor de DOM, CSS y métodos parecidos a Jquery.

Una de las características que le hace atractivo, es que Jsoup es un proyecto de código abierto bajo la licencia del MIT.

Las principales funcionalidades de Jsoup son:

- limpia y analiza código HTML desde URL, ficheros o cadenas
- encuentra y extrae datos usando DOM transversal o selectores CSS
- manipula elementos, atributos y texto HTML
- genera código ordenado de HTML

2.4.2. Metaio SDK

En principio lo que hizo que nos decidiéramos por utilizar esta librería fue su implementación de localización *indoor* basada en marcadores LLA, pero no es la única de sus características.

EL SDK no es lo único que Metaio proporciona, también ofrece herramientas de desarrollo y modelado 3D, servicios de almacenamiento en la nube, sistemas de seguimiento, etc. Ofrecen un ecosistema de desarrollo de realidad aumentada completo.

Una de las características más interesantes a mi modo de ver, desde el punto de vista del desarrollo, es que ofrecen un entorno que es independiente de la plataforma para la que se desarrolle. Se puede utilizar lo que ellos llama AREL AR scripting que es portable a Android, iOS, Windows y Unity.

2.4.3. Android SDK

Como ya se habló en la introducción de Android y su entorno de desarrollo, en esta sección me gustaría nombrar dos elementos del API de Android que han sido parte importante de la aplicación y no han sido nombrados a lo largo del proyecto:

- **Toast**: Un *Toast* provee de un sencillo feedback sobre una operación dentro una pequeña ventana. Solo va a ocupar la cantidad de espacio que ocupe el mensaje y solo va a ser visible durante unos instantes. Este tipo de notificación se ha utilizado a la largo de la aplicación para mostrar mensajes informativos, como por ejemplo el de la figura.

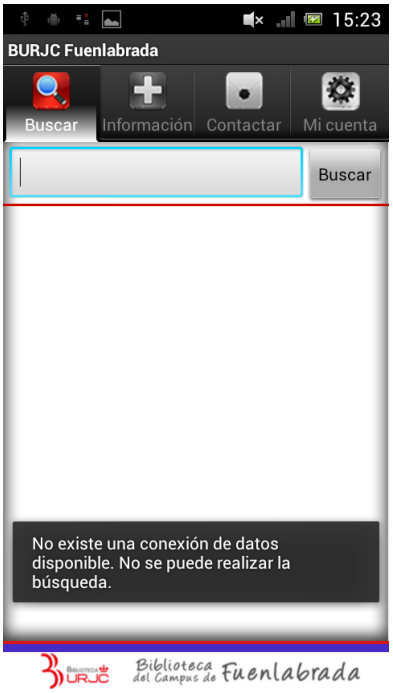


Figura 2.6: Mensaje de que falta conexión de datos

- **ConnectivityManager**: Es una clase que proporciona información sobre el estado de la red o la conectividad. También notifica a la aplicación cuando la conectividad de la red cambia. Estamos hablando de una aplicación desarrollada para un móvil que utiliza datos descargados de la red. No podría definirse así, si no tuviera un control de la conectividad en todo momento.

2.4.4. Android Development Toolkit

Para ayudar en el desarrollo de aplicaciones, Android dispone de todo un ecosistema de herramientas gratuitas a disposición de los desarrolladores:

- **Eclipse**: Es un conjunto de herramientas de programación de código abierto multiplataforma. Típicamente ha sido usado para utilizar entornos de desarrollo integrados (IDE). Eclipse, aparte, se compone de una comunidad de usuarios, que ayudan a la expansión de las áreas de aplicación de la herramienta.
- **Android Developer Tools (ADT)**: Es un plugin para Eclipse que provee un entorno de desarrollo de nivel profesional para crear aplicaciones Android. Es un completo IDE de Java que permite, entre otras cosas, navegación integral entre recursos Java y XML; provee de editores XML y Android XML; posee herramientas de análisis estático de errores y usabilidad; diseño de la interfaz de usuario con un sencillo sistema **drag and drop**; etc. Esta herramienta permite tener acceso a las diferentes APIs de las diferentes versiones de Android, pudiendo discriminar los desarrollos por versiones, por ejemplo.
- **Opciones de desarrollador en el terminal**: Todos los sistemas Android poseen una opción en el menú de configuración que activa las opciones de desarrollador que permiten, entre otras cosas, depurar sobre un terminal la aplicación, delimitar los bordes de los componentes de un layout, da información del uso del CPU del móvil en tiempo real; etc. Para poder depurar la aplicación sobre un terminal físico es necesario tenerlo conectado por cable al equipo donde se está desarrollando la aplicación.
- **Desarrollo en terminales virtuales**: Otra herramienta interesante es el Android Virtual Device Manager (AVDM) que permite emular en el equipo en el que se está desarrollando un terminal Android con la versión y características que se le indiquen. Sobre este terminal virtual es posible cargar y depurar como si se tratara de un terminal real.

2.5. Estructura de la memoria

A lo largo de la memoria se van a ir describiendo los distintos pasos dados para la realización de este proyecto. La memoria se ha planteado de tal forma que siga lo mejor posible la cronología del desarrollo de la aplicación:

- **Objetivos**

  En este capítulo se van a describir los primeros pasos necesarios para desarrollar una aplicación, como son: la descripción de la necesidad que nos lleva a realizar esta tarea; los requisitos que se han de cumplir para considerarla completa; así como el modelo de desarrollo elegido para completarla.

- **Diseño e implementación**

  Una vez definido el libro de ruta a seguir para el desarrollo de la aplicación, nos toca reflexionar sobre cómo vamos a llevar a cabo esa tarea y finalmente llevarlo a buen puerto. En este capítulo se trata de explicar y diferenciar el proceso de diseño y el de la implementación. Para ello se va a cubrir todo el proceso de diseño de la interfaz de usuario y las funcionalidades implementadas para la aplicación, que van a quedar descritas en la arquitectura general.

- **Pruebas y resultados**

  Ninguna aplicación está terminada si un periodo de pruebas donde una serie de personas se presten voluntarias a probar las distintas funcionalidades del proyecto. Estos reportes de errores se recogerán para depurar el correcto funcionamiento de la aplicación. Se medirá el grado de satisfacción del cliente realizando una encuesta a las personas que realizaron las pruebas y que pertenecen a la biblioteca.

- **Conclusiones**

  Una vez todas las fases del proyecto están terminadas, toca recoger toda la información obtenida a lo largo de los procesos de desarrollo y pruebas y comparar de donde partíamos al principio, los objetivos que nos habíamos fijamos y cuántos de ellos se han conseguido.

Capítulo 3

Objetivos

3.1. Descripción de la necesidad

La biblioteca del Campus de Fuenlabrada quería una aplicación que pudiera recoger, de forma ordenada y sencilla, la mayor cantidad de información posible de la que está alojada en la página web. Esto incluía de forma específica la información relativa al Campus de Fuenlabrada y parte de la normativa general de la biblioteca.

Conociendo esto, se elaboró la siguiente propuesta:

- Una aplicación organizada en pestañas. Se realizó un estudio de la información que debía ser mostrada y se identificaron grupos muy diferenciados entre sí que podían ser fácilmente recogidos por categorías, y así cada una de estas categorías daría lugar a una pestaña. En la fig. 3,1 podemos ver dos pantallazos del diseño inicial propuesto.

- Las pestañas propuestas fueron:

  - "Buscar" que se encargaría de la funcionalidad del buscador.

  - "Información" donde se muestre la normativa de la biblioteca, los horarios, las preguntas frecuentes y como llegar.

  - La tercera pestaña sería "Contactar" e incluiría las distintas formas de contactar con la universidad presentes en la web.

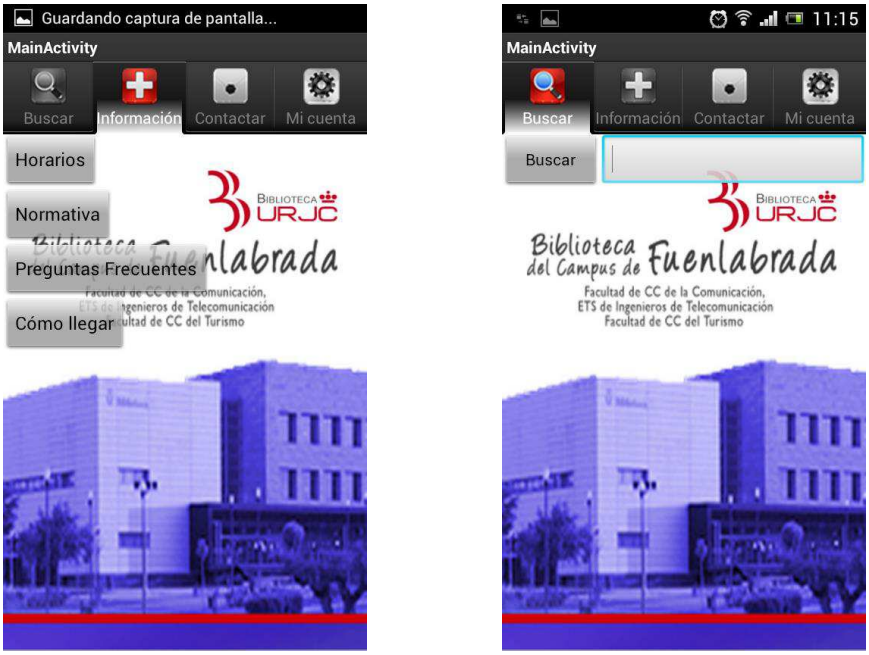


Figura 3.1: Boceto de las pestañas presentado a la universidad

- Y la cuarta y última pestaña se llamaría “Mi cuenta” y daría acceso a la información relativa a préstamos y devoluciones de cada usuario, una vez ingresadas las credenciales necesarias.

- Para hacer que la aplicación fuera compatible con el mayor número de dispositivos posible, se propondría como versión Android para el desarrollo, la versión Froyo (Android 2.2). La motivación viene porque es una versión que lleva tiempo en el mercado y se desarrollaría pensando en que fuera compatible con futuras versiones.

3.2. Requisitos

Al ser un proyecto en el que son protagonistas tanto el interés de la biblioteca de la universidad, como el interés nuestro en desarrollar un proyecto con unos requisitos determinados. Vamos a dividir los requisitos de cada una de las partes en dos categorías:

- **Requisitos de la biblioteca de Fuenlabrada**

  Están basados en las necesidades descritas anteriormente y son:

- Una aplicación para dispositivos móviles.
- Mostrar de forma ordenada toda la información relativa a la biblioteca del campus de Fuenlabrada y la normativa general de la biblioteca.
- Interfaz de usuario sencilla.

- Nuestros requisitos
  - Utilizar el sistema operativo Android.
  - Implementar una funcionalidad de realidad aumentada.

Capítulo 4

Diseño e implementación

A la hora de enfrentarnos a la forma en la que se iba a desarrollar la aplicación, en primer lugar e realizó un análisis tanto de la información que tendríamos que mostrar, cómo sería la mejor forma de mostrarla y cómo se iba a ser la implementación. Este ejercicio de abstracción se ha de realizar a dos niveles distintos, uno de ellos corresponde a la interfaz de usuario y el otro a la arquitectura general de la aplicación. Aunque estemos hablando de planteamientos distintos, están intrínsecamente unidos e influyen en el diseño de la aplicación. En la fig. 4,1 podemos ver cómo se relacionan las distintas funcionalidades respecto a la interfaz de usuario, existiendo funcionalidades que podemos considerar propias de ciertas pestañas y otras que son transversales a todas ellas.

En lo referente a la interfaz de usuario, podemos observar cómo se han realizado cuatro divisiones correspondientes a cada una de las pestañas. En cada una de estas pestañas va a estar incluida tanto la información, como las funcionalidades específicas de cada uno de ellos. Por otro lado, también vemos representadas las principales funcionalidades en las que se apoya la aplicación y en qué pestañas van a ser utilizadas.

De esta forma tan gráfica, se puede apreciar como funcionalidades como el HTML parser, implementado a partir de una librería externa, o el método HTTP POST, implementado a partir del API de Android, se van a utilizar en todas las pestañas. Esto es así debido a la naturaleza de la aplicación, que se va a nutrir de la información alojada en la web.

Sin embargo, funcionalidades como la realidad aumentada están relacionadas únicamente con la localización de libros indoor, así que solamente se encontrará definida en la pestaña *“Buscar”*.

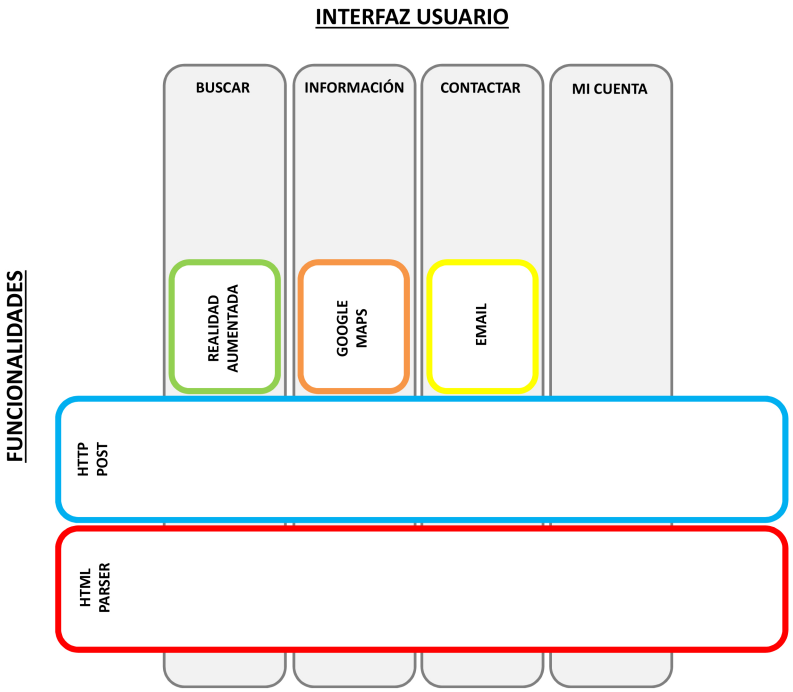


Figura 4.1: Esquema de diseño de la aplicación

4.1. Modelo de desarrollo

En lo concerniente al modelo de desarrollo, he optado por un modelo de desarrollo de software en cascada. Tal y como se refleja en la fig. 4,2, se han seguido una serie de etapas que tienen como consecuencia final la aplicación que se está describiendo a lo largo de esta memoria.

La primera fase, la definición de los **requisitos del proyecto**, comienza con la reunión con el tutor, para concretar los aspectos que deseamos cubrir en este proyecto. Tras esta reunión, conociendo a priori las necesidades de la universidad, se realiza una propuesta de aplicación para presentar a la biblioteca. Esta primera fase queda definitivamente cerrada cuando nos reunimos con el personal de la biblioteca, les presentamos nuestra idea del proyecto, lo aceptan y se nos proporciona de forma definitiva su listado de requisitos

En lo relativo al **diseño**, se han tenido en cuenta todos los requisitos acordados en la fase previa. Se siguió con la idea propuesta a la universidad de una aplicación organizada en cuatro pestañas, donde cada una de esas pestañas presenta información de distinta naturaleza. Sin embargo, en cuanto a arquitectura de la aplicación hay que hacer otras distinciones: parser HTML, realidad aumentada, correo electrónico y HTTP. Algunas de estas funcionalidades, como el parser HTML o el cliente HTTP, son transversales en la aplicación utilizándose en varias pestañas,

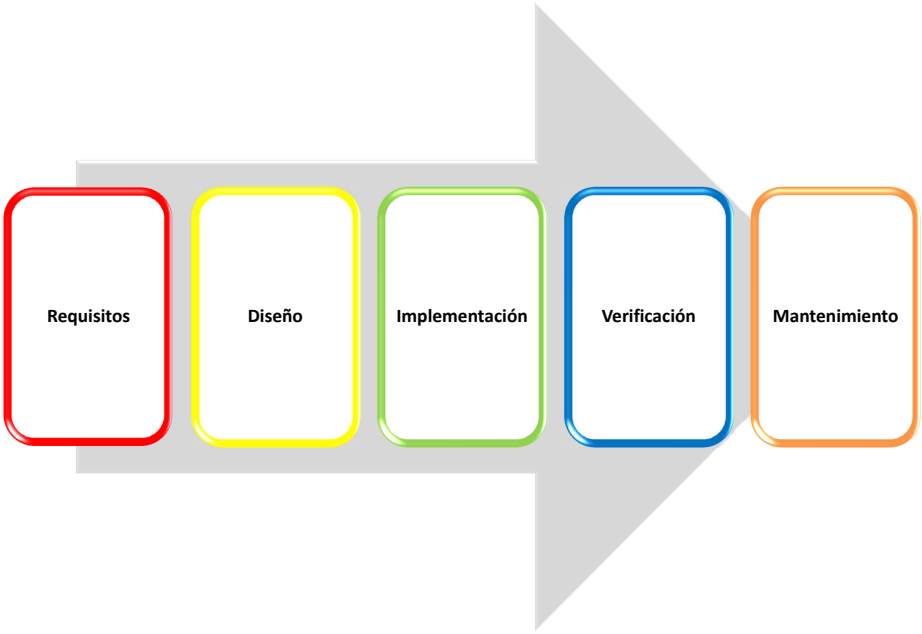


Figura 4.2: Modelo de desarrollo software en cascada

mientras que otras, como la realidad aumentada o el correo electrónico, solamente se van a utilizar en una de las pestañas.

La **implementación** se ha realizado siempre intentando mantener como filosofía la claridad en el código y la posible reutilización del mismo pensando en futuras funcionalidades o proyectos. Para el desarrollo de la aplicación se han utilizado diferentes tecnologías: Jsoup, la SDK de Metaio y en lo relativo a Android entre otras muchas características se van a destacar Toast, BaseAdeapter, WebChromeClient y otros módulos.

Tanto la fase de diseño, como la implementación, van a ser desarrolladas con más profundidad en la siguiente sección de la memoria.

La fase de **verificación** se ha realizado con ayuda de la biblioteca. A pesar, de que durante el proceso de implementación la aplicación se ha ido probando, se dedicó un periodo de prueba de dos semanas con la ayuda de los funcionarios de la biblioteca. Estas pruebas incluían el reporte de incidencias de funcionamiento de la aplicación, así como una encuesta de satisfacción donde se les daba la oportunidad de opinar y evaluar el funcionamiento y diseño de la aplicación.

La fase del desarrollo relativa al **mantenimiento** es la única que no se puede realizar durante

mucho tiempo, puesto que el proyecto termina con la verificación del mismo por parte del cliente. Aun así, como parte de la verificación incluía el reporte de incidencias, estas se han ido corrigiendo en función del tiempo disponible y se ha actualizado la aplicación para poder solventar esos errores.

4.2. Interfaz de usuario

En esta sección se va a tratar todo lo relativo a la UI (*User Interface*) de la aplicación, justificando las decisiones de **forma**, **aspecto** y **funcionalidad**.

A la hora de determinar la "**forma**" definitiva de la aplicación, se tuvo muy en cuenta uno de los requisitos de la universidad: Una **interfaz de usuario sencilla**. Que algo sea sencillo o no para el usuario puede resultar subjetivo. Ese es el motivo por el que acudí al mercado para ver que aplicaciones eran las más populares y qué tipo de interfaz presentaban.

Algunas de las aplicaciones más populares de Android y de uso más extendido son las denominadas sociales como Facebook, Instagram o de mensajería como WhatsApp. Todas ellas comparten algo en común, sus interfaces gráficas están basadas en un sistema de pestañas. Aunque hayan evolucionado e introducido mejoras, como es lógico, siguen basándose en este sistema. Y es por eso que, viendo que se ajustaba perfectamente al orden de la información que previamente se había realizado, se asumió el sistema de pestañas como el más apropiado para esta aplicación, ya que iba a resultar un sistema que el usuario encontraría reconocible y amigable.

Cuando hago referencia al "**aspecto**" me refiero a la configuración de colores, a la iconografía es decir, al apartado estético de la aplicación. Desde el principio he pretendido que la aplicación esté directamente relacionada con la página web biblioteca de Fuenlabrada. Con esa idea se creó la siguiente imagen de la aplicación, como se puede ver en la fig. 4,3

Figura 4.3: Imagen al pie de la aplicación

La figura anterior va estar presente en todas las pantallas de la aplicación. Es una imagen que se va a mostrar en el margen inferior de la pantalla y su misión es que identifique a la aplicación

en un golpe de vista y aporte coherencia con respecto al diseño de la web. Como se puede apreciar en la siguiente figura, la fig. 4,4, los colores predominantes son el azul que identifica el campus de Fuenlabrada y el rojo que es el color del logotipo. Por otro lado, era importante destacar que esta aplicación es perteneciente a la **biblioteca del campus de Fuenlabrada**. Es por eso que se aprovechó el título superior, para incluirlo en la imagen anterior.

Figura 4.4: Web de la biblioteca URJC de Fuenlabrada

Respecto a la funcionalidad, se ha intentado a lo largo de toda la aplicación que el uso de la misma sea lo más sencillo posible y reducir el número de gestos necesario para acceder a la información al mínimo posible y que estos gestos sean siempre parecidos. Por ejemplo, dentro de las pestañas que contienen distintos tipos de información, “Información” y “Contactar”, se han desarrollado botones cuyo nombre aportan una descripción suficiente del contenido que muestran. Al pulsar dichos botones, se despliega la información relativa a ellos, de tal forma que la información siempre se presenta ordenada y categorizada. Si la información mostrada, a su vez, alberga más información de distintos tipos, esta será también recogida en botones que mostrarán su contenido de forma similar a la anterior. Con este tipo de comportamiento repetitivo se pretende que la curva de aprendizaje para usar la aplicación sea lo más baja posible.

A partir de este punto se explicará de forma más técnica y detallada cómo se ha dotado de funcionalidad a cada apartado de la aplicación.

En lo referente a las interfaces de usuario, Android utiliza ficheros XML llamados ***layouts***. Los layouts son elementos no visibles que determinan como se van a distribuir en el espacio los

controles que incluyamos en su interior. Una vez creado un layout podremos realizar modificaciones sobre el dinámicamente utilizando código en Java. En el siguiente fragmento podemos ver la definición del elemento *TabHost* definido para poder implementar la aplicación basada en el sistema de pestañas:

```
<TabHost
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true" >
```

Y como ya habíamos adelantado, la implementación del layout solamente distribuye contenido en la pantalla para poder utilizarlo y modificarlo. Es decir, hemos de inicializarlo y asociarle los layout que van a definir como se distribuye el contenido en cada una de las pestañas. Esto se consigue con el siguiente código:

```
setContentView(R.layout.activity_main);

TabHost tabHost = getTabHost();
TabHost.TabSpec spec;
Intent intent;
Resources res = getResources();

intent = new Intent().setClass(this, Pestana1.class);
spec = tabHost.newTabSpec("Pestana1")
              .setIndicator("Buscar",res.getDrawable(R.drawable.pestana1)).setContent(intent);
tabHost.addTab(spec);
intent = new Intent().setClass(this, Pestana2.class);
spec = tabHost.newTabSpec("Pestana2")
              .setIndicator("Información",res.getDrawable(R.drawable.pestana2)).setContent(intent);
tabHost.addTab(spec);
intent = new Intent().setClass(this, Pestana3.class);
spec = tabHost.newTabSpec("Pestana3")
              .setIndicator("Contactar",res.getDrawable(R.drawable.pestana3)).setContent(intent);
tabHost.addTab(spec);
intent = new Intent().setClass(this, Pestana4.class);
spec = tabHost.newTabSpec("Pestana4")
              .setIndicator("Mi cuenta",res.getDrawable(R.drawable.pestana4)).setContent(intent);
tabHost.addTab(spec);
```

En este caso **activity_main** es el layout definido para la aplicación donde se encuentra el TabHost, así como el pie de imagen de la fig. 4,3. Como se puede observar se fija el contexto sobre el que vamos a trabajar (*setContentView(R.layout.activity_main);*) y es entonces cuando se inicializa el TabHost y se le asignan tanto las clases donde vamos a implementar su funcionalidades (*intent = new Intent().setClass(this, Pestana1.class);*), como los layout que van a tener asociados (*spec = tabHost.newTabSpec("Pestana3").setIndicator("Contactar",res .getDrawable(R.drawable.pestana3)).setContent(intent);*)

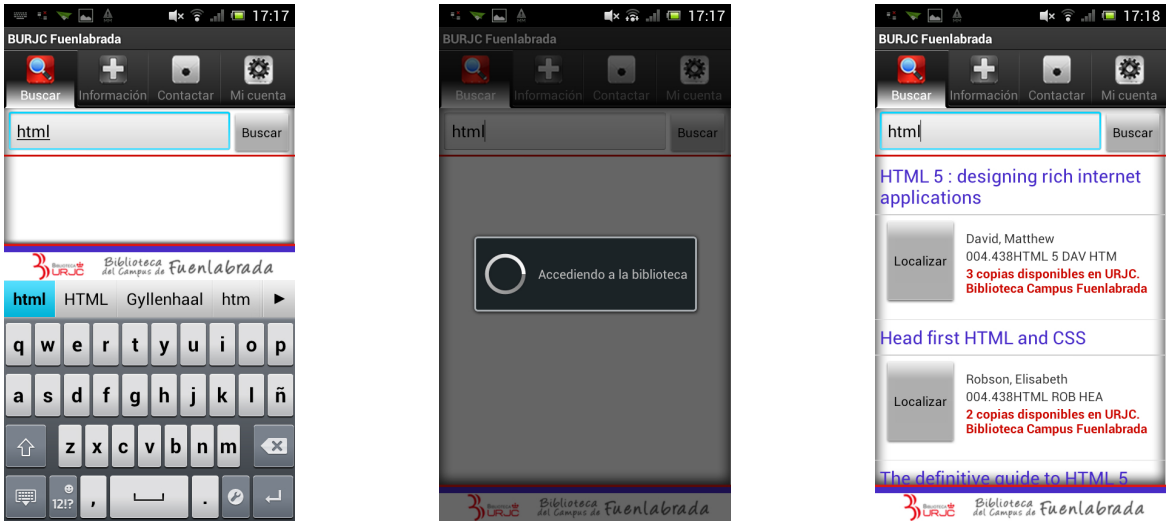
4.2.1. Splash

Se conoce como Splash a la imagen que se muestra al principio de una aplicación. Esta imagen se muestra durante un breve periodo de tiempo. En el caso de mi aplicación es un elemento meramente decorativo que nos presenta un icono como el de la fig. 4,5 y un mensaje que nos indica que la aplicación está cargando, pero en realidad no tiene ninguna funcionalidad añadida. La imagen se muestra durante 3 segundos.

Figura 4.5: Splash de la aplicación

4.2.2. Buscar

Una vez que la pantalla de Splash se retira, la primera imagen que ve el usuario es la de la pestaña *Buscar*, tal y como se muestra en la fig. 4,6a. Como se puede observar, el teclado está habilitado desde el principio asumiendo que el usuario va a utilizarlo para realizar una búsqueda. Una vez introducidos los parámetros de búsqueda, se ha de pulsar en el botón "Buscar". En secciones futuras explicaremos cómo la aplicación se conecta al servidor de búsquedas WebCat de la biblioteca y obtiene la información. En este apartado nos vamos a centrar en lo referente a la interfaz de usuario. Mientras se está realizando la descarga de información la aplicación mostrará el mensaje "Accediendo a la biblioteca". Una vez se cierra la conexión y se analiza la información se muestra en pantalla "Mostrando el contenido" que apenas dura un segundo. Una vez la información ha sido correctamente parseada, esta se muestra tal y como se observa en la fig. 4,6c.



(a) Portada de búsqueda (b) Buscando (c) Resultado de búsqueda

Figura 4.6: Búsqueda en el catálogo

Fijándonos en el resultado de la búsqueda, se aprecia que hay más resultados de los que simplemente son mostrados en la pantalla, a los que tendremos acceso haciendo *scrolling* con el dedo.

Esta pestaña tiene una peculiaridad y es que el contenido del layout es modificado dinámicamente en función del número de resultados que nos devuelva el catálogo de la biblioteca. Para ello, se ha tenido que crear una clase nueva llamada myAdapter. Esta clase es una instan-

cia de una clase ofrecida por el API de Android que se llama **BaseAdapter** y que actúa como base para crear un **Adapter** que pueda usar un *listView*, es decir, un listado de *Views*. Un View representa la unidad básica para construir componentes de una interfaz de usuario y el Adapter es el elemento que nos da es acceso a los datos que van a manejar cada uno de los Views que compongan el listView. En el caso que nosotros manejamos, un View corresponderá a cada uno de los elementos de la lista que se está mostrando. Es decir, la información relativa a cada libro.

Como la información mostrada tiene siempre la misma estructura, se ha definido un layout común para la información de los libros, **entrada.xml**. En él, se van a incluir el título del libro, el autor, el CDU (identificador del libro en la biblioteca), la disponibilidad del mismo, así como el formato para cada uno de esos datos y el botón *Localizar*.

Es la clase myAdapter la que se va a encargar de generar el listado rellenando los datos. Aquí vemos el método encargado de esta tarea:

```
public View getView(int position, View convertView, ViewGroup parent) {

  View view = null;
  if (convertView == null)
  {
    LayoutInflater inflater = (LayoutInflater)myContext.
                  getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    view = inflater.inflate(R.layout.entrada, null);
  }else
  {
    view = convertView;
  }

  TextView title = (TextView) view.findViewById(R.id.entradaTitle);
  title.setText(myBooks.get(position).title);

  TextView author = (TextView) view.findViewById(R.id.entradaAuthor);
  author.setText(myBooks.get(position).author);

  TextView ISBAN = (TextView) view.findViewById(R.id.entradaISBAN);
  ISBAN.setText(String.valueOf(myBooks.get(position).ISBAN));

  TextView availb = (TextView) view.findViewById(R.id.entradaDisp);
  availb.setText(myBooks.get(position).availability);

  Button launcher = (Button)view.findViewById(R.id.butSearch);
  launcher.setTag(position);
  launcher.setOnClickListener(new View.OnClickListener() {
```

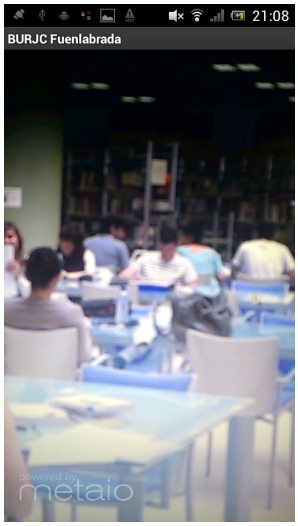
```
    @Override
    public void onClick(View v) {
    Intent intent = new Intent(myContext, LLACore.class);
    intent.putExtra("ISBAN",
            String.valueOf((myBooks.get((Integer)v.getTag()).ISBAN)));
    myContext.startActivity(intent);
    }
  });

  return view;
}
```

Este método se va a llamar una vez por cada elemento que tenga el objeto **myBooks**. Ese objeto es creado como resultado del parser de los datos enviados desde la web de la biblioteca y contiene una lista con todos los datos de todos libros devueltos en la búsqueda.

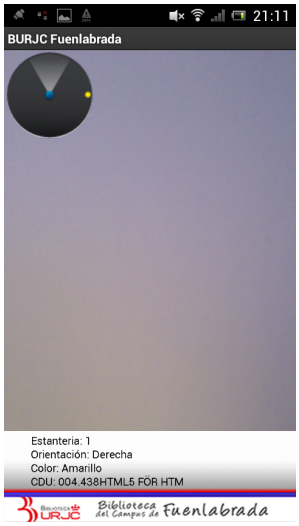
Aunque se hablará de ello en profundidad más adelante, cabe llamar la atención sobre el método *public void onClick(View v)*, que sirve para dar funcionalidad a los botones una vez que son pulsados. En este caso, se está definiendo la funcionalidad del botón "Localizar", indicándole que está asociado a la clase **LLACore.class** encargada de la gestión de la aplicación de realidad aumentada. Además, se le pasa el identificador del libro seleccionado para poder localizarlo dentro de la biblioteca: *intent.putExtra("ISBAN", String.valueOf((myBooks.get((Integer)v.getTag()).ISBAN)));*

4.2.3. Realidad Aumentada

A pesar de que no existe ninguna pestaña titulada como realidad aumentada, creo que se merece un apartado aparte dentro de la sección reservada a la interfaz de usuario. Siguiendo el hilo del apartado anterior, una vez que el usuario ha pulsado el botón "Localizar", se lanza esta aplicación. En primer lugar, aparece una imagen de la empresa desarrolladora que apenas está un segundo en pantalla. Cuando esta desaparece, se activa la cámara a pantalla completa y solo se observa la marca de agua de Metaio tal y como se muestra en la fig. $4{,}7a$. En la fig. $4{,}7b$ podemos ver la apariencia de la aplicación una vez es detectado un código LLA. En la figura más a la derecha, la fig. $4{,}7c$, podemos observar la que será la imagen más representativa de esta sección, la que aparece cuando se ha encontrado un libro.



(a) RA buscando código LLA



(b) RA orientando hacia un libro

(c) RA con el libro localizado

Figura 4.7: Interfaz de realidad aumentada

Como se puede apreciar aparecen los siguientes elementos:

- Pie de aplicación que le aporta coherencia con el diseño general de la misma.
- Un radar en la esquina superior izquierda. Este radar nos da información sobre la posición relativa que tenemos con respecto al libro. Según nos movemos sobre el sitio esta posición se actualiza hasta mostrar una imagen que determina la posición de nuestro libro.
- Indicador con el literal “Tu libro”, que nos informa de que finalmente hemos encontrado la estantería donde se encuentra el libro
- Panel informativo que nos aporta información adicional sobre la localización del libro para acercar aún más al usuario a su objetivo.

De hecho, la información reflejada en este panel informativo no formaba parte del diseño original del proyecto. Es el producto de un esfuerzo conjunto con la biblioteca para mejorar el funcionamiento de la aplicación.

Esta modificación del planteamiento original surge durante el proceso de diseño de la aplicación. En la biblioteca se utiliza la Clasificación Decimal Universal o CDU como sistema de

clasificación del conocimiento que sive para ordenar y clasificar las obras. Estando en la biblioteca planteando como realizar la organización de los CDU correspondiente a cada libro para que la aplicación apuntara a la estantería correcta, desde el personal de la biblioteca se me plantea la posibilidad de guiar al usuario hasta el libro de forma más efectiva. Esto es, con la funcionalidad planteada en origen, la aplicación sería capaz de apuntar a la estantería donde se encuentra el libro almacenado. Lo que se plantea es "acercar" al usuario más al libro, no solo a la estantería.

Finalmente, se decide iniciar una colaboración con la biblioteca que se lleva a cabo de la siguiente manera: Cada una de las estanterías están divididas en módulos o columnas más pequeños. A cada uno de esos módulos o columnas, se les asigna un color que se va a distinguir con una pegatina en la parte alta de las estanterías. Estos colores serán simétricos desde la cabecera de la estantería a derecha e izquierda. De esta manera, conseguíamos proporcionar la usuario mucha más información. La aplicación le indica en que estantería está almacenado el libro, si se encuentra orientado a la derecha o la izquierda en esa la misma y en qué color se encuentra dentro de ella. En la fig. 4,8 se puede observar un esquema de cómo resultaría una de las estanterías.

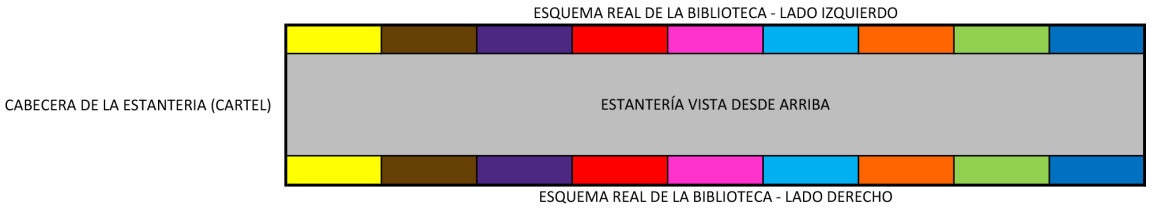


Figura 4.8: Esquema de las estanterías

Para actualizar esta información en la aplicación, desde la biblioteca tuvieron que recoger y mandarme los datos que ligaban los CDU que identifican los libros, con los colores asignados para cada estantería. Esta es una fracción de código de la base de datos generada para localizar los libros.

```
est++;// Estanteria 3
    ori = 0; //Orientación: izqda

    collectionCDU.add(new BookData(est, 5, ori, "METODOS DE ENCUESTA" ,"303.6"));
    collectionCDU.add(new BookData(est, 5, ori, "ESTUDIO DE GENERO" ,"305"));
    collectionCDU.add(new BookData(est, 5, ori, "SOCIOGRAFIA. ESTUDIOS DE SOCIEDAD" ,"308"));
    collectionCDU.add(new BookData(est, 5, ori, "DEMOGRAFIA" ,"3014"));
```

```
collectionCDU.add(new BookData(est, 4, ori, "SOCIOLOGIA" ,"316"));
collectionCDU.add(new BookData(est, 3, ori, "CORRIENTES SOCIOLÓGICAS" ,"316.2"));
collectionCDU.add(new BookData(est, 3, ori, "ESRTUCTURA SOCIAL" ,"316.3"));
collectionCDU.add(new BookData(est, 1, ori, "DINÁMICA SOCIAL" ,"316.4"));
ori = 1; col=1; //Orientación: dcha
collectionCDU.add(new BookData(est, 1, ori, "PSICOLOGÍA SOCIAL" ,"316.6"));
collectionCDU.add(new BookData(est, 2, ori, "OPINIÓN PUBLICA" ,"316.653"));
collectionCDU.add(new BookData(est, 3, ori, "SOCIOLOGÍA DE LA CULTURA" ,"316.7"));
```

La variable *est* identifica la estantería, el *color* viene definido por el carácter numérico, *ori* se refiere a la orientación siendo 0 izquierda y 1 derecha, el siguiente campo es la *descripción* de la sección y el último los valores relevantes del *CDU* que sirven para identificarla posición de los libros.

A pesar de los esfuerzos, el sistema no es perfecto debido a que los libros de las estanterías se mueven en función del periodo lectivo y las necesidades del momento. El sistema falla en los extremos, es decir, aquellos libros que se encuentran al principio de un color determinado o al final, son sensibles de acabar en el color adyacente. Sin embargo, al haber estado consensuado con la biblioteca el error fue minimizado en la medida de lo posible ya que ellos replantearon la posición de los libros para que sufrieran el menor número de variaciones posible.

4.2.4. Información

Lo primero que se ve al pulsar la pestaña ***Información*** son cuatro botones. Cada uno de estos botones representa una subcategoría diferente: *Horarios*, *Normativa*, *FAQ* y *Cómo llegar*. Vamos a estudiar una a una cada subcategoría para ver sus diferencias en cuanto a implementación desde el punto de vista de la interfaz de usuario. Pero primero, vamos a ver cómo se han diseñado estos botones y su funcionalidad ya que van a estar presentes en gran parte de la aplicación.

Ya hemos hablado con anterioridad de los *layouts*, que nos permiten dar forma a la interfaz de usuario. Dentro de los layouts, podemos definir como queremos ordenar la información de forma vertical, horizontal, como una tabla de datos, etc. Uno de los atributos que tienen estos tipos de layout es la visibilidad. Es decir, podemos definir un layout como visible o invisible para el usuario. En este caso los botones que vemos nada más seleccionar la pestaña lo que hacen es

habilitar la visibilidad de otros layouts que realmente contienen la información asociada a ellos, pero van a ser invisibles *a priori*.

Vamos a verlo con más detalle para los Horarios.

1. **Horarios**

   Es un botón que nos da información de los horarios de apertura y servicio de la biblioteca. Esta información se actualiza directamente con la información mostrada en la página web de la universidad en el siguiente enlace:

   http://www.urjc.es/biblioteca/fuenlabrada/Horario/horario.html

   Como se decía anteriormente, este botón tiene como funcionalidad dar visibilidad y quitársela al layout que realmente contiene la información sobre los horarios. Pero para entender esto más claramente vamos a ver la siguiente figura, la fig. 4,9.

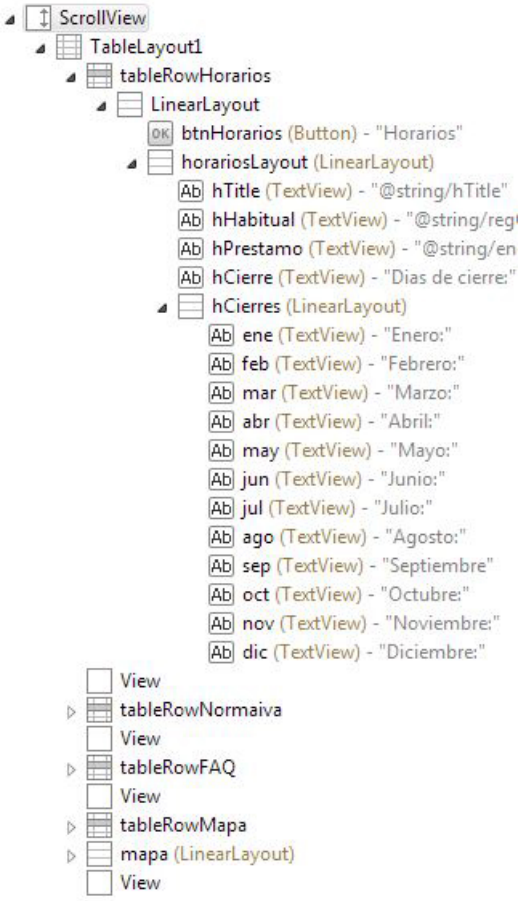


Figura 4.9: Esquema del layout de la pestaña de Información

Es una captura de pantalla de la estructura que tiene el layout completo de la pestaña de Información. Como se puede observar, se compone de una serie de elementos que se van combinando entre sí, para darle una coherencia al conjunto. En nuestro caso, vemos que el elemento principal del layout donde cargamos todos los demás es el *Scroll View*. Este View nos va a permitir, en el caso de que la información mostrada sea mayor que el tamaño de la pantalla, poder hacer *scrolling*. El siguiente layout que encontramos es un *TableLayout* que nos va a permitir organizar el contenido como si de una tabla se tratara. Es decir, cada una de las subcategorías va a estar definida dentro de un *TableRow* o fila de la tabla. El *TableLayout* también permite definir celdas en cada una de las *TableRow*, pero en nuestro caso no lo hemos utilizado. Se puede observar que al mismo nivel de la estructura de los *TableRow* vemos unos elementos llamados simplemente View. Estos se han utilizado para pintar las líneas que separan los botones. Ahora vamos a ver, más en detalle, como se ha definido el contenido de las *TableRow*.

El primer layout que encontramos, es el que nos va a ayudar a situar los siguientes elementos que fijemos en él. Se trata del ***LinearLayout*** con orientación vertical. Cada elemento añadido a este layout se va ordenar de forma vertical, es decir, de arriba a abajo. Como se puede observar en la fig.4,9, el primer elemento añadido es **btnHorarios** que representa el primer botón que vemos en la fig. $4{,}10a$. Android nos da ciertas facilidades a la hora de definir botones y, como se puede apreciar en el siguiente código, nos permite insertar imágenes a ambos lados de la etiqueta, donde vamos a poner los iconos. Ya se había comentado antes la visibilidad en los layouts y en este caso podemos ver cómo el atributo *android:visibility* se encuentra fijado a visible.

```
<Button
  android:id="@+id/btnHorarios"
  style="buttonStyle"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:background="@android:drawable/screen_background_light_transparent"
  android:clickable="true"
  android:drawableLeft="@drawable/calendar"
  android:drawableRight="@drawable/arrow"
  android:gravity="left|center"
  android:text="@string/btm1"
  android:textAppearance="?android:attr/textAppearanceMedium"
  android:textStyle="bold"
```

```
    android:visibility="visible" />
```

El siguiente elemento añadido al *LinearLayout* es otro LinearLayout, que hemos distinguido como el identificador **horariosLayout**. En él se van a definir varios elementos *TextView* que nos permiten añadir texto a la interfaz y otros LinearLayout. Lo interesante de horariosLayout es que está definido de la siguiente manera.

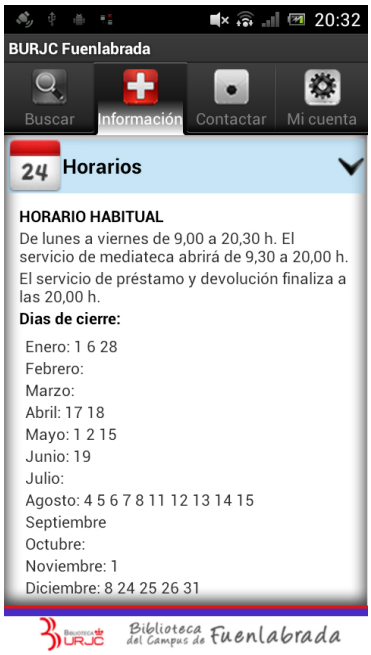
```
<LinearLayout
  android:id="@+id/horariosLayout"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:layout_margin="10dp"
  android:orientation="vertical"
  android:visibility="gone" >
  [...]
  </LinearLayout>
```

Si nos fijamos de nuevo en el atributo *android:visibility* que en este caso indica que está *gone*. Es decir, no lo vamos a ver de inicio en el layout porque está oculto. Es por eso, que se implementa como acción al pulsar btnHorarios que se cambie a visible dicho atributo. Así como cambiar de color el btnHorarios y modificar la imagen mostrada a la derecha con una flecha apuntando hacia abajo indicando que se ha desplegado información. Si se le vuelve a pulsar volvemos a la situación original, como se puede ver en la fig. $4{,}10b$.

```
public void showHorarios(View view){
  LinearLayout horariosLayout = (LinearLayout) findViewById(R.id.horariosLayout);
  Button horarios = (Button) findViewById(R.id.btnHorarios);

  if (horariosLayout.getVisibility()==View.VISIBLE){
    horariosLayout.setVisibility(View.GONE);
    horarios.setBackgroundColor(Color.TRANSPARENT);
    horarios.setCompoundDrawablesWithIntrinsicBounds(getResources()
      .getDrawable(R.drawable.calendar), null, getResources()
      .getDrawable(R.drawable.arrow), null);

  }else if (horariosLayout.getVisibility()==View.GONE){
    horariosLayout.setVisibility(View.VISIBLE);
    horarios.setBackgroundColor(0x7789c9ee);
    horarios.setCompoundDrawablesWithIntrinsicBounds(getResources()
      .getDrawable(R.drawable.calendar), null, getResources()
```

```
        .getDrawable(R.drawable.arrow_down), null);
    }
}
```

(a) Pestaña información



(b) Pestaña información tras pulsar Horarios

Figura 4.10: Comportamiento del botón Horarios

2. **Normativa**

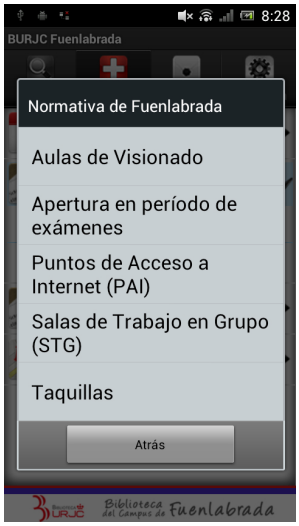
En esta subcategoría se pretende reunir toda la normativa relativa a la biblioteca de forma general, y a la de Fuenlabrada de forma particular. Para ello vamos a acceder a los contenidos las siguientes dos direcciones:

http://www.urjc.es/biblioteca/LaBiblioteca/normativas.html - Normativa general

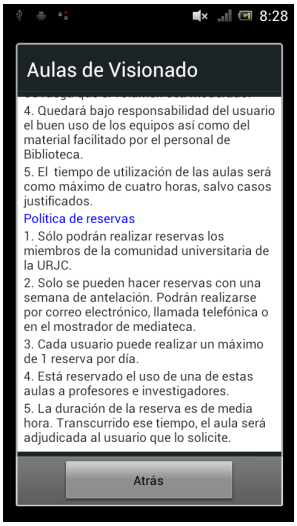
http://www.urjc.es/biblioteca/fuenlabrada/LaBiblioteca/normativas.html - Normativa de Fuenlabrada

En este caso el botón Normativa hace visible un layout muy sencillo del estilo *LinearLayout* que contiene otros dos botones, haciendo cada uno de ellos referencia al contenido de los enlaces anteriores como se puede apreciar en la fig. $4{,}11a$.

(a) Nomartiva



(b) Tras seleccionar Biblioteca de Fuenlabrada



(c) Se selecciona Aulas de Visionado

Figura 4.11: Comportamiento del botón Normativa

En ambos casos, la información está dividida en secciones; esto implica que hay que hacer una segunda división. La aplicación va a acceder a cada uno de los enlaces listados en la página web y a almacenar los contenidos. Para no complicar más la interfaz de usuario con un tercer layout que debería ser desplegado al tocar cualquiera de los botones de normativa, se optó por una solución diferente, el ***AlertDialog***. A efectos prácticos es una ventana o Dialog, que nos va a permitir definir un título y hasta tres botones de control. En nuestro caso solo hemos definido el botón *Atrás* para volver a la aplicación. Una de las características del AlertDialog, que lo hacía tan práctico en este caso, es que simplemente pasándole un objeto CharSecuence[] te añade tantos botones como elementos haya almacenados en el objeto, ver la fig. 4,11*b*. El siguiente método hace referencia al botón **Biblioteca de Fuenlabrada**:

```
public void showNormFuenlabrada()
{
final CharSequence[] rules=myFuenlaRules.toArray(new String[myFuenlaRules.size()]);

    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    builder.setTitle("Normativa de Fuenlabrada")
           .setItems(rules, new DialogInterface.OnClickListener() {

                public void onClick(DialogInterface dialog, int which) {
```

```
                    View view = getLayoutInflater().inflate(R.layout.norm_f1, null, false);
                    selectRulesLayout(which,view);

                    AlertDialog.Builder builder = new AlertDialog.Builder(context);
                    builder.setView(view);
                    builder.setTitle(rules[which])// Add action buttons
                             .setNegativeButton("Atrás", new DialogInterface.OnClickListener() {
                               public void onClick(DialogInterface dialog,int id) {
                                 dialog.cancel();
                               }
                             });
                    AlertDialog d = builder.create();

                    // Makes the dialog box fits the 90% of the screen
                    WindowManager.LayoutParams lp = new WindowManager.LayoutParams();
                    lp.copyFrom(d.getWindow().getAttributes());
                    lp.width = WindowManager.LayoutParams.FILL_PARENT;
                    lp.height = WindowManager.LayoutParams.FILL_PARENT;

                    d.show();
                    d.getWindow().setAttributes(lp);
            }
    })
    .setNegativeButton("Atrás",new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog,int id) {
dialog.cancel();
}
});
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

Para cada uno de los botones del AlertDialog se define su comportamiento al ser pulsados. En este caso vamos a generar un segundo AlertDialog al que le hemos modificado las características ligeramente para que aparezca a pantalla completa sin dejar ver la aplicación de fondo, como se puede ver en la fig. $4{,}11c$. Con este efecto se quiere conseguir la sensación de haber accedido a un nivel distinto. A este segundo AlertDialog le hemos definido un elemento Builder que nos va a permitir cargar un *LinearLayout* dentro de un *ScrollView* donde poder volcar el contenido que hemos parseado de la página web con el método *selectRulesLayout(which,view);*.

El comportamiento del botón **BURJC** es muy similar. También genera un *AletDialog* con una serie de botones que coinciden con el contenido enlazado en la página web que se indicó arriba. Pero existe una diferencia, en este caso, no se enlaza a otra página web, se

enlaza a la descarga de un documento PDF. Es por eso que la implementación del *public void onClick(DialogInterface dialog, int which)* difiere de la anterior y es como sigue:

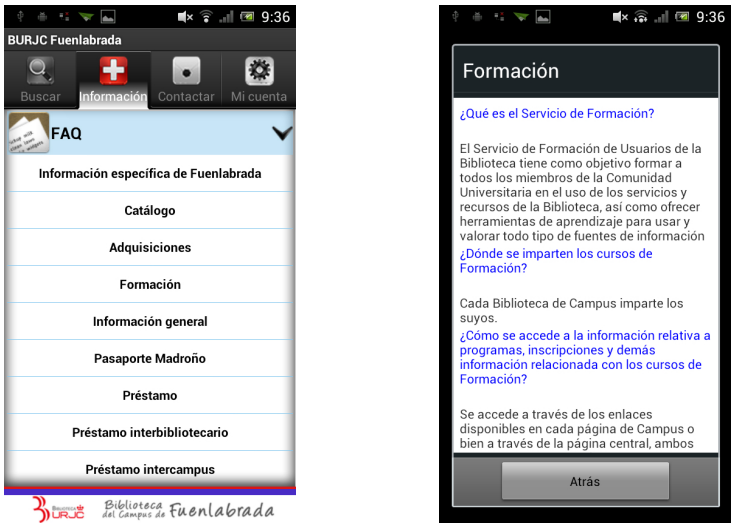
```
public void onClick(DialogInterface dialog, int which)
{
  DownloadManager dm = (DownloadManager) getSystemService(DOWNLOAD_SERVICE);
  Request request = new Request(
        Uri.parse(myBurjCategories.get(which).getLink().replace(" ", "%20")));
        dm.enqueue(request);
}
```

El ***DownloadManager*** es un servicio del sistema que se encarga de manejar las descargas HTTP de larga duración. Una vez que el cliente solicita la descarga de un recurso, el *DownloadManager* se va a encargar de gestionar en según plano la descarga, haciéndose cargo de todas las interacciones HTTP, reintentando la descarga en caso de fallos de red por motivos de cambios de conectividad o por reinicio del sistema.

3. **FAQ**

Las preguntas frecuentes (FAQ) están enlazadas en la siguiente dirección:

http://www.urjc.es/biblioteca/fuenlabrada/ - Pestaña Preguntas frecuentes



(a) FAQ (b) Tras pulsar Formación

Figura 4.12: Comportamiento del botón FAQ

El aspecto de la interfaz de usuario es análogo al descrito para el botón de Normativa como se puede observar en la fig. $4{,}12a$. Aunque existen ciertas diferencias con respecto

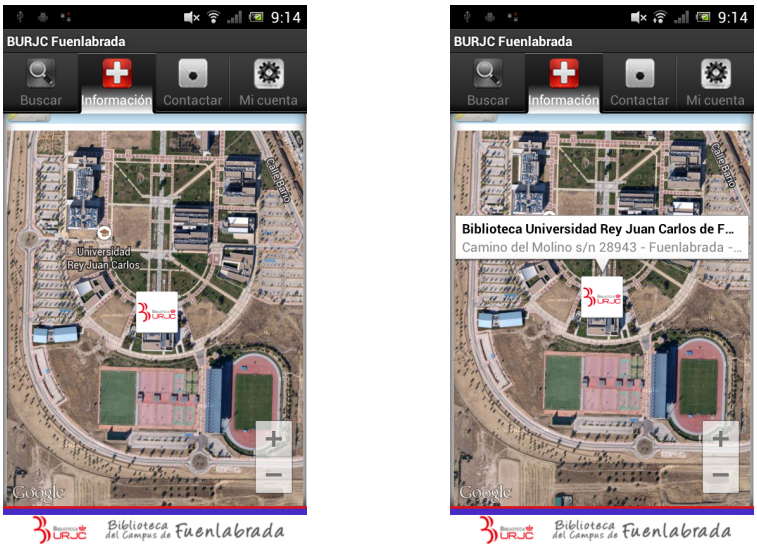
al comportamiento de Normativa. En este caso los botones que mostramos en la IU si nos enlazan directamente con la información que tenemos que mostrar y no con una nueva división. Es por eso que nos saltamos el primer AlertDialog y se implementa directamente un ALertDialog full screen con el elemento Builder que nos permite volcar la información parseada. Como se puede apreciar en la fig. $4{,}12b$ es análogo al obtenido en la fig. $4{,}11c$.

4. **Cómo llegar**

En cuanto a la funcionalidad del botón "Cómo llegar", lo que se ha decidido hacer, es incrustar un elemento de Google Maps. Cuando se pulsa el botón, se vuelve visible el elemento Fragment que contiene el mapa que ya se ha centrado previamente en las coordenadas GPS de la biblioteca (Lat: 40.284043650362996, Lng: -3.8391530513763428) que se indican en la siguiente dirección.

http://www.urjc.es/comollegar/fuenlabrada/cll_fuenlabrada.html - Cómo llegar

Para indicar claramente la posición de la biblioteca se ha añadido un marcador con el logotipo de la biblioteca de la universidad, ver la fig. $4{,}13a$, que al ser pulsado descubre la dirección de la misma, como se observa en la fig. $4{,}13b$.



(a) Cómo llegar (b) Tras pulsar el marcador

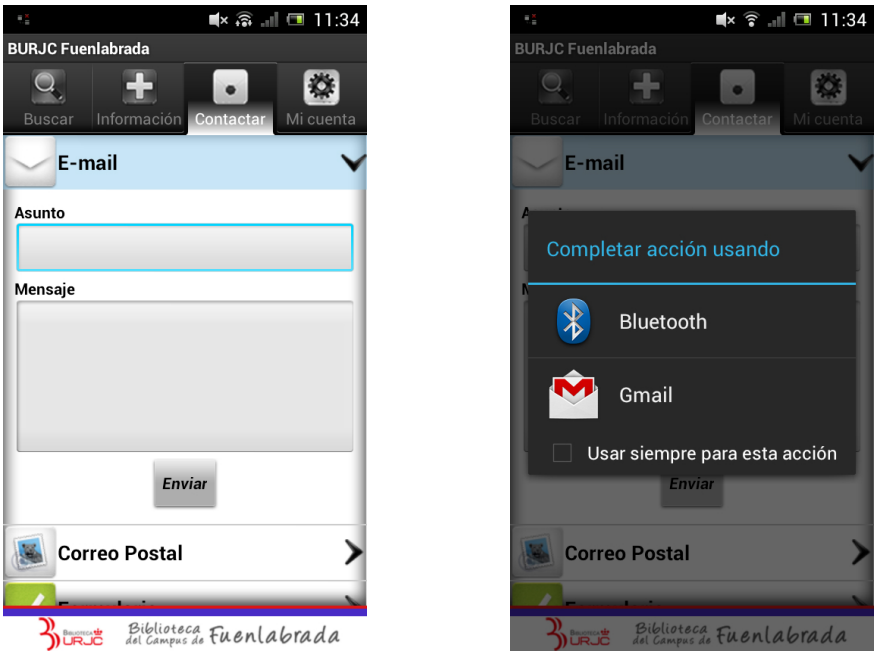
Figura 4.13: Comportamiento del botón Cómo llegar

4.2.5. Contactar

La pestaña ***Contactar*** tiene un aspecto muy similar al de la pestaña *Información*, pero su finalidad es totalmente distinta. En ella vamos a descubrir cuatro botones: *E-mail*, *Correo Postal*, *Formulario* y *Teléfono/Fax*. Cada uno representa e implementa formas distintas de ponerse en contacto con la biblioteca de la universidad. En ese apartado vamos a ver, una a una, las distintas posibilidades recogidas en la aplicación.

1. **E-mail**

   Como podemos observar en la fig. 4,14*a* la interfaz es sencilla. Solo hay que completar el campo *Asunto* y el contenido del *Mensaje*. Pulsando en el botón *Enviar*, el sistema nos preguntará sobre la aplicación que queremos utilizar para enviar el correo, ver la fig. 4,14*b*.



(a) E-mail (b) Tras pulsar botón Enviar

Figura 4.14: Envío de E-mail

   En esta pantalla vemos un elemento que no se habían visto hasta ahora, el ***EditText***, que nos va a permitir recoger datos introducidos por el usuario. Gracias al API (Aplication Program Interface, en inglés) de Android nos va a resultar muy sencillo dotarlos de ciertas funcionalidades que los van a hacer diferentes entre sí.

```
<EditText
  android:id="@+id/asuntoET"
  android:layout_width="fill_parent"
  android:layout_height="match_parent"
  android:ems="10"
  android:inputType="textAutoComplete" />
<EditText
  android:id="@+id/bodyET"
  android:layout_width="fill_parent"
  android:layout_height="140dp"
  android:ems="10"
  android:gravity="top"
  android:scrollbarAlwaysDrawVerticalTrack="true" />
```

En el código se puede ver cómo están definidos los dos EditView del layout del e-mail. El primero corresponde al más pequeño y se va a usar para el asunto, mientras que el segundo, se va a usar para recoger el cuerpo del mensaje. Los cuatro primeros atributos son comunes para ambos y hacen referencia al identificador de elemento, el ancho, el alto y el tamaño de la fuente si vamos de arriba hacia abajo. Es el último el que varía en ambos casos. En el *asuntoET* hemos definido un tipo de texto autocompletar. Sin embargo, en el segundo asumiendo que puede haber más texto del que cabe en le recuadro, le hemos dado la capacidad de hacer *scroll* vertical.

Esta funcionalidad hace referencia a la primera entrada de la pestaña Consulte al Bibliotecario en la siguiente dirección:

http://www.urjc.es/biblioteca/fuenlabrada/ - Pestaña Consulte al Bibliotecario/Correo electrónico

2. **Correo Postal**

   Se trata sencillamente de un TextView que contiene la información sita en la siguiente dirección:

   http://www.urjc.es/biblioteca/fuenlabrada/LaBiblioteca/directorio.html#postal - Pestaña Consulte al Bibliotecario/Correo postal

   El resultado es el que se muestra en la siguiente figura, la fig. 4,15

Figura 4.15: Comportamiento del botón de Correo Postal

3. **Formulario**

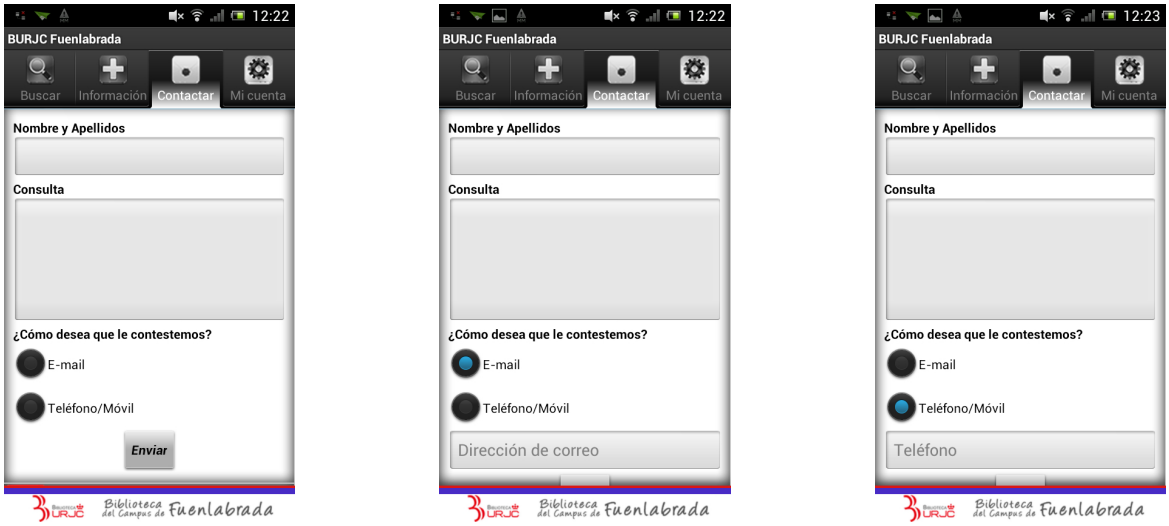
   El ***Formulario electrónico*** es un método de comunicación que la biblioteca pone a disposición de los alumnos en la siguiente dirección:

   http://www.urjc.es/biblioteca/Formularios/ConsulteBibliotecario/fuenlabrada.html - Pestaña Consulte al Bibliotecario/Formulario electrónico

   Si accedemos a la página que indicamos en el enlace anterior, se puede observar como en el formulario se ofrecen dos posibilidades a la hora de ponerse en contacto con el alumno que ha enviado el formulario. Una es a través de correo electrónico y la otra a través del teléfono, y ambas están visibles en la misma pantalla. Esto en la aplicación no era posible puesto que el layout se volvía más grande que la pantalla del móvil en la que estaba realizando la pruebas. Es por eso que aunque mantuve las dos posibilidades modifiqué sensiblemente la forma de presentar el formulario.

   Si nos fijamos en la fig. $4{,}16a$, la composición del layout es muy similar a la del *E-Mail*. La diferencia fundamental es el elemento ***RadioGroup*** donde están definidos dos ***RadioButton*** que nos va a permitir elegir E-Mail o Teléfono. El *RadioButton* se selecciona cuando lo pulsamos y la funcionalidad añadida es que cuando lo juntamos con otro dentro

de un *RadioGroup*. Este último deselecciona todos aquellos RadioButton que no son el seleccionado. Es decir, nos permite seleccionar un RadioButton y solo uno de toda la lista de RadioButton definida dentro del RadioGroup. Esto me va a permitir como se observa en las figuras $4{,}16b$ y $4{,}16c$, según se haya seleccionado e-mail o teléfono respectivamente, que se haga visible un tercer EditText. Para el caso del E-mail será especial para introducir direcciones de correo electrónico y en el segundo para introducir números.



(a) Formulario (b) Tras elegir E-mail (c) Tras elegir Teléfono

Figura 4.16: Funcionamiento del botón Formulario

4. **Teléfono/Fax**

Este último botón tiene como finalidad recoger todo el directorio de teléfonos y correos electrónicos que se listan en la página web de la biblioteca en la siguiente dirección:

http://www.urjc.es/biblioteca/fuenlabrada/LaBiblioteca/directorio.html - Pestaña Consulte al Bibliotecario/ Teléfono/Fax

Para ello se accede la dirección indicada anteriormente y se parsea la información. La idea original era respetar al máximo el formato de la página web. Sin embargo, como es de esperar al portar la información a una plataforma distinta esto supone cambios. En este caso es imposible disponer la información en columnas como está distribuida en la web. Es por eso que se va a distribuir como un listado en el que las diferencias de formato van

a dar una idea clara de los diferentes tipos de información, tal y como se aprecia en la fig. 4,17.

Figura 4.17: Comportamiento del botón de Teléfono/Fax

Además de formatear de una forma determinada la información, se le ha añadido una funcionalidad extra y es que tan solo con pulsar los número de teléfono, se abre la aplicación del teléfono de Android con el número seleccionado. Por otro lado, respecto a las direcciones de correo, al ser pulsadas, se nos va a ofrecer el medio a través del cual queremos escribir a esa dirección de correo. El comportamiento va a ser análogo al descrito en el apartado de “E-mail”.

4.2.6. Mi cuenta

En esta pestaña lo que se pretende es informar al usuario del estado en el que se encuentra su cuenta personal de la biblioteca. Esto es, cuántos libros tiene prestados y cuándo tiene que devolverlos, si ha vencido o no ha vencido el préstamo y en caso de haberlo hecho, cuál sería la sanción que impondría la biblioteca.

Para ello se le solicitan al usuario dos datos para identificarlo, ver la fig. 4,18a, el ID de usuario que se puede ver en la parte trasera del carnet de la universidad y el PIN personal que se encuentra en el Portal de Servicios de la universidad.

(a) Portada de Mi Cuenta

(b) Sin libros prestados

(c) Con libros prestados

Figura 4.18: Pestaña Mi Cuenta

En la figura 4,18*a*, podemos ver un elemento que no había aparecido hasta ahora junto a la etiqueta *"Recordar"*. Es el ***CheckBox*** que nos va a permitir tomar una decisión u otra, en función de que el elemento este seleccionado. Si se encuentra seleccionado, la aplicación va a recordar las credenciales del usuario, del tal forma que cuando vuelva a abrir la aplicación no tenga que volver a insertar los datos.

4.3. Arquitectura general

En este apartado vamos a cambiar el punto de vista desde el que estamos analizando la aplicación dejando de lado la interfaz de usuario y así, poder poner el foco en la implementación de las distintas funcionalidades que dan lugar a los comportamientos que se han descrito en el apartado anterior. De esta forma, vamos a ver cómo se ha gestionado desde la aplicación el acceso a la web de la universidad y cómo se ha procesado y ordenado esta información para luego ser mostrada de forma sencilla en la aplicación; la implementación de la aplicación de realidad aumentada, etc.

4.3.1. HTML Parser

El análisis de contenido HTML para la extracción de información relevante es una de las partes fundamentales de este proyecto. Esto es así debido a la naturaleza del mismo. Se trata de una aplicación que va a tomar la información que está almacenada en la web de la biblioteca de Fuenlabrada de la Universidad Rey Juan Carlos para mostrarla o para utilizarla en otras funcionalidades.

Sin embargo, ese análisis se va a realizar de dos formas diferentes. En el primer caso se va a utilizar un parser propio llamado **myParser** y nos va a servir para analizar todo el contenido volcado desde el WebCat de la biblioteca que es el servidor que nos devuelve los resultados del Catálogo:

http://centauro.urjc.es/uhtbin/cgisirsi/?ps=u9Xyjpt1zP/CENTRAL/217970010/60/80/X - Catálogo de la bibloteca

El segundo caso es diferente y para analizar el texto HTML vamos a utilizar ***Jsoup*** del que se habla en la sección 2,4,1 y a partir del cual se ha generado la clase **htmlParser**.

El motivo de esta distinción es que en el primer caso utilizando un parser como *Jsoup* los resultados no se podían tratar con facilidad y prácticamente había que realizar una rectificación específica para cada uno de los elementos extraídos. Se valoró la situación y puesto que la respuesta del servidor de la biblioteca es generada automáticamente y no es proclive a ser cambiado en un corto periodo de tiempo, se decidió desarrollar un parser específico para el tipo de respuesta que devuelve. Mientras que en el resto de los casos se podía dar una buena solución con *Jsoup*.

Esta decisión implica que tanto la pestaña "Buscar", como la de "Mi Cuenta" que descargan la información del servidor del Catálogo utilizan *myParser*. Mientras que "Información" y "Contactar", van a utilizar *htmlParser*. A partir de este punto vamos a estudiar cómo funcionan cada uno de ellos.

- **myParser**

A lo largo de esta sección vamos a explicar cómo se ha implementado la función ***searchAndParse()*** que va a ser la encargada de la descarga y parseado. La información que estamos

intentando obtener, es la relativa a los libros que almacena la biblioteca o que tenemos prestados. LA información de cada uno de estos libros, los vamos a definir con la clase Node que va a tener la información sobre el título, el autor, el identificador del libro, la disponibilidad y en el caso de los préstamos, la fecha de entrega y la sanción si existiera.

```
public class Node
{
  public String title;
  public String author;
  public String ISBAN;
  public String availability;
  public String deadline;
  public String sanction;
}
```

Para poder recoger esos datos se implementa la clase myParser que va a tener como atributos *String* ***myText*** y *ArrayList<Node>* ***myNodes***, donde *myText* contendrá el código HTML que se tiene que parsear y *myNodes* será el resultado del análisis del HTML. A continuación, en la fig. 4,19 se puede observar el esquema que representa como se ha analizado la información almacenada en *myText*. El análisis se realiza secuencialmente recorriendo todo el contenido HTML y analizando los distintos elementos en el orden en el que son devueltos desde el servidor.

Figura 4.19: Esquema del análisis del HTML para la obtenición de la información de los libros

El primer paso para poder usar cualquier parser es descargar el contenido. Y en esta ocasión, este paso no se resolvía de una forma trivial. La primera intención fue utilizar un cliente HTTP del API de Android que gestionase una petición POST con los campos necesarios para la búsqueda. El problema surge porque se ha de indicar al servidor que soportamos Java Script, en caso contrario nos devuelve un mensaje sin resultados. El cliente HTTP utilizado en Android no soportaba esta funcionalidad por lo que se optó por utilizar un elemento WebView que se puede configurar para funcionar como un navegador que soporte JavaScript.

```
public void searchAndParse(View view){

  final WebView web = (WebView)findViewById(R.id.browser);
  final ProgressDialog progressDialog=ProgressDialog.show(Pestana1.this,
                            "", "Accediendo a la biblioteca",true,false);
  progressDialog.setProgress(0);
  web.getSettings().setJavaScriptEnabled(true);
  web.setWebChromeClient(new WebChromeClient()
  {
    @Override
    public void onProgressChanged(WebView view, int progress)
    {
      progressDialog.setProgress(progress);
    }
  });
```

Se observa que a la vez que se define el navegador *WebView*, se ha definido una variable *ProgressDialog*. Este nuevo elemento va a ser el encargado de abrir un Dialog que va ir mostrando mensajes según vaya avanzado el proceso de descarga y análisis.

En este punto, una vez definido el navegador, el único problema era como obtener el código HTML que se ha descargado ya que el WebView no tiene ningún método de acceso a esa información. Para ello vamos a declarar una interfaz que nos permita obtener el código HTML inyectando código JavaScript en la respuesta del servidor.

```
// Class used to create an interface instance that allows the application to insert
// java script code for HTML code extracting
public class IFace {

  public void print(String data) {
    data ``<html>''+data+``</html>'';
    Pestana1.htmlGlobal=data;
  }
}
```

Este método va a volcar el contenido HTML de la respuesta del servidor al atributo htmlGlobal de la pestaña de “Buscar”.

Por otro lado, cada vez que se realiza una petición al servidor de la biblioteca, se establece una sesión y estos datos de sesión se han de respetar a lo largo de todas las consultas que se vayan a realizar. Es por eso que la secuencia de acciones a realizar para obtener la información deseada es la siguiente:

1. Acceder a *http://centauro.urjc.es/uhtbin/webcat*
2. Obtener la sesión de la dirección web a la que te redirige
3. Realizar la petición usando HTTP POST con los datos de sesión antes conseguidos
4. Al recibir la respuesta ejecutar el código JavaScript que nos va a devolver el código HTML

Es decir, cada vez que hemos terminado de cargar una página, hemos de ejecutar alguna acción y es aquí donde el WebView se vuelve interesante. Como se puede observar en el código anterior hemos definido un cliente *WebChromeClient* que nos permite modificar el comportamiento del navegador. En ese caso en particular, vamosa realizar las acciones que se describen en la sigueinte figura, la fig. 4,20

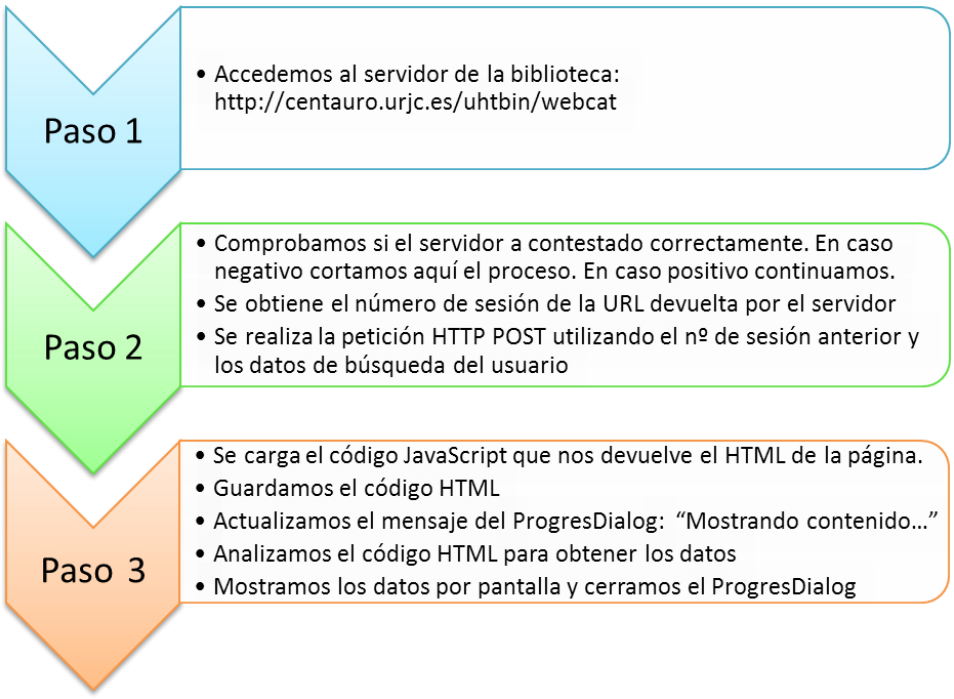


Figura 4.20: Esquema del proceso de descarga y análisis de la información

Estas acciones completan la funcionalidad del método *searchAndParse()*. Podemos conseguir este comportamiento gracias a la redifinición del método *onPageFinished()* del cliente *WebChromeClient*. Se va a utilizar una variable para controlar el flujo del programa, *loadedControl*, que en función de si tiene el valor puesto a 1, permite capturar la sesión y realiza la petición POST, o si tiene el valor a 2, permite cargar el código JavaScript en la respuesta y espera hasta tener el código HTML para llamar a la función parser() que tiene la siguiente implementación:

```
public void parse(View view){

  myParser htmlParser = new myParser(htmlGlobal,this);

  if(htmlParser.parseHtml())
  {
    myBooks = htmlParser.getParsed();
    entryAdap.notifyDataSetChanged();
  }
  else
    Toast.makeText(getApplicationContext(),
       "No existen resultados para esa entrada", Toast.LENGTH_SHORT).show();

}
```

Como se puede observar se inicializa un objeto de la clase **myParser** y se llama al método parseHtml() que ya explicamos anteriormente.

Una vez que myBooks está inicializado correctamente, se llama a notifyDataSetChanged() que actualiza el ListView de la pestaña Buscar que ya describimos en la sección dedicada a la interfaz de usuario.

- **htmlParser**

Como ya adelantamos antes, el caso del htmlParser es bastante diferente. De hecho, es una clase muy sencilla como se puede apreciar:

```
public class htmlParser extends AsyncTask<String, Integer, Elements>{

  @Override
  protected Elements doInBackground(String... params) {
    Elements elem = null;
    try {
```

```
        Document doc = Jsoup.connect(params[0]).get();
        elem = doc.select(params[1]);

      } catch (IOException e) {
        e.printStackTrace();
      }
      return elem;
    }
  }
```

Lo más destacable es que htmlParser va a heredar de **AsyncTask**, que permite un uso sencillo y apropiado de los hilos. Esta clase, permite realizar operaciones en segundo plano y publicar los resultados en un thread de la interfaz gráfica por ejemplo, sin tener que hacer manipulación concreta de threads.

En este caso, nos vamos a centrar en la pestaña de Información. Vamos a estudiar como se realiza el proceso de *parsing* para las normas relacionadas con la biblioteca de Fuenlabrada, contenido alojado en el siguiente enlace:

http://www.urjc.es/biblioteca/fuenlabrada/LaBiblioteca/normativas.html - Normativa de la biblioteca de Fuenlabrada

Accediendo a la dirección indicada, veremos que nos dirige a una web que contiene cinco enlaces, que a su vez, nos dirige cada uno a una dirección distinta con una normativa determinada. El análisis lo vamos a hacer en dos fases. En primer lugar, si recordamos lo que se explicó en el apartado 4.2.4 en la fig. $4{,}11b$, al pulsar el botón de la normativa de Fuenlabrada se abría un diálogo con cinco botones, que coinciden con los cinco enlaces de la dirección a la que acabamos de acceder. Esto significa, que lo primero que hemos de hacer es almacenar los datos de estos cinco enlaces. El segundo paso consistirá en acceder a la dirección a la que apuntan cada uno de ellos y analizar la información que contienen.

Para ello necesitamos una forma de poder guardar toda esa información ordenada, para que luego sea accesible y poder utilizarla en el layout y es por eso que definimos la siguiente clase, **InfoNode.class**.

```
public class InfoNode{

  private String nombre = null;
```

```
private String link = null;
private List<String> titles = null;
private List<String> entries = null;
```

En este caso *nombre* será el encargado de almacenar la etiqueta de cada uno de los cinco enlaces de la dirección anterior. *Link* almacenará la dirección a la que están enlazados. Y una vez que hayamos accedido a esa dirección, *titles* almacenará los títulos del sitio donde estamos accediendo y *entries* el contenido debajo de esos títulos.

En la siguiente figura 4,21 vamos ver de forma esquemática cómo se ha implementado esta funcionalidad. Siguiendo como hasta ahora, este esquema corresponde con la primera parte del proceso de almacenar los cinco enlaces y sus etiquetas.

Figura 4.21: Esquema del análisis de infromación para obtener el contenido de los cinco enlaces

Hay dos variables que cabe destacar: *myFuenlaRues*, que es del tipo List<String>y que va a almacenar las etiquetas y *myFuenlaCategories*, que está definida como List<InfoNode>y que

va a albergar el contenido de cada uno de los cinco enlaces. He comentado que este esquema resuelve el primer paso del proceso y es verdad a medias. Esto es así, porque que una vez que se han obtenido el título y la dirección de cada uno de los cinco enlaces, en los pasos 6 y 7 del esquema se está llamando a un segundo método, *loadNode()*, que va a ser el encargado de realizar el análisis de la segunda parte del proceso. El proceso queda descrito en el esquema de la figura 4,22.

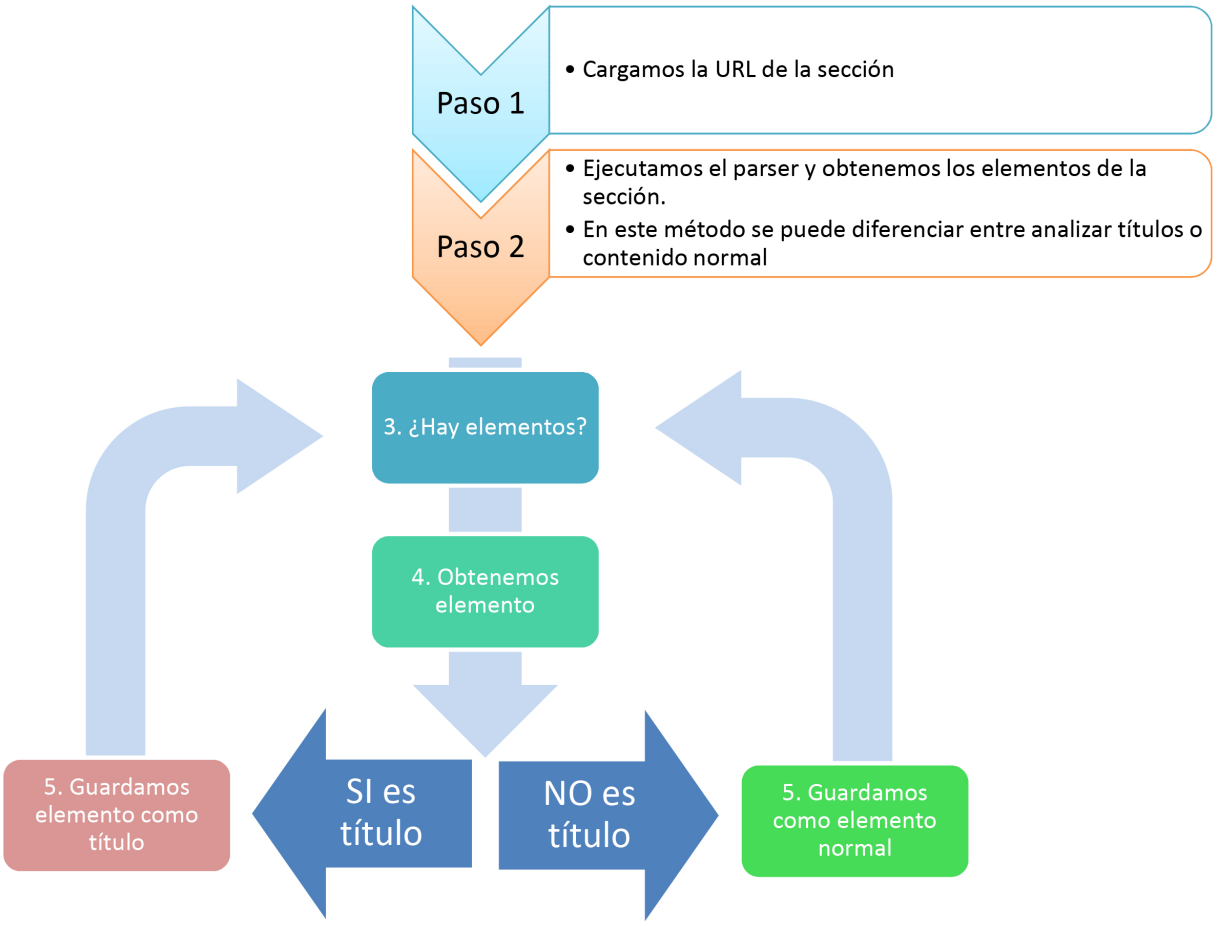


Figura 4.22: Esquema del análisis de la segunda parte del proceso, función *loadNode()*

Una vez terminado este proceso ya tenemos la información almacenada, ordenada y lista para usarla en la interfaz gráfica de una forma sencilla.

4.3.2. RA: Realidad Aumentada

Para poder explicar bien el funcionamiento de la Realidad Aumentada, hay que contextualizar la situación. Vamos a partir de que el usuario ha realizado una búsqueda y ha seleccionado

un libro. El primer paso entonces es hacerle saber al **Activity** de la RA qué libro ha elegido el usuario. Un Activity es algo sencillo, en lo que la aplicación está centrada en ese momento y además, con lo que el usuario puede hacer algo. Casi todas las actividades interactúan con el usuario.

En nuestro caso, debemos pasar información del Activity en el que se estaba desarrollando la pestaña "Buscar" a la de RA, y no es trivial. Por eso el API de Android nos facilita una solución, que en nuestro caso, se resuelve como vimos en el apartado $4{,}3{,}5$, cuando se explicó la función *getView()*. En este método se define el comportamiento al pulsar el botón Localizar:

```
launcher.setOnClickListener(new View.OnClickListener() {

  @Override
  public void onClick(View v) {
    Intent intent = new Intent(myContext, LLACore.class);
    intent.putExtra("ISBAN", String.valueOf((myBooks.get((Integer)v.getTag()).ISBAN)));
    myContext.startActivity(intent);
  }
```

Como se puede apreciar, se define un objeto *Intent* en el que se asocia el contexto actual con la clase a la que vamos a llamar y se le añade la información que queremos pasar de un Activity al otro en forma de pares. A la etiqueta ISBAN se le asocia el valor de ISBAN almacenado en myBook. Lo siguiente que se deberá hacer una vez que se inicie el Activity de RA será recuperar este valor para poder utilizarlo después.

En este punto podemos decir que nos independizamos de la aplicación y entramos dentro de la RA.

La RA queda totalmente definida en dos ficheros con finalidades distintas:

1. **MetaioSDKViewActivity.java**: Es la base sobre la que se va a apoyar todo lo demás. En él se inicializa y configura el funcionamiento de la SDK. Solo se le ha añadido la firma necesaria para que funcione la aplicación.

2. **LLACore.java**: Es donde se define el comportamiento que va a tener la aplicación. En nuestro caso, es donde se han realizado todas las modificaciones.

Ahora, poniendo el foco en la funcionalidad de la RA, vamos a centrarnos en LLACore. Hay que tener en cuenta, que como Metaio proporciona una SDK con una configuración genérica

que sirve para diferentes aplicaciones, lo primero será configurar la aplicación para que funcione en modo LLA. Lo que significa que desde el momento que arranca la RA, la aplicación está analizando imágenes en tiempo real buscando un marcador LLA.

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);

  //Get Book Data
  ISBAN=getIntent().getStringExtra("ISBAN");
  BookDataBase BD = new BookDataBase();
  myBookData = BD.checkBookDataBaseBestMatching(ISBAN);

  //SDK setting
  metaioSDK.setLLAObjectRenderingLimits(10, 10000);
  g = metaioSDK.createBillboardGroup(800, 800);

  //Set Tracking configuration
  boolean result = metaioSDK.setTrackingConfiguration("LLA");
  MetaioDebug.log("Tracking data loaded: " + result);

  mText = new TextView(this);
  mGUIView = mText;
}
```

El método onCreate se ejecuta siempre mientras se crea un Activity. Como podemos ver, están sucediendo tres cosas. No vamos a explicarlas de forma cronológica, de hecho en tercer lugar se está ajustando la configuración para LLA como dijimos anteriormente. En el segundo caso se están definiendo unas condiciones necesarias para poder ver correctamente en pantalla todos los elementos que se van a mostrar por pantalla. Uno de los motivos por los que se hace esto es porque si tenemos dos elementos a distintas distancias, la imagen asociada al más cercano se mostrará más grande que la que se encuentra más alejado y esto puede ser molesto. Es por eso que la primera sentencia crea unos límites de distancia mínima y máxima entre los cuales se dibujarán los elementos que se encuentren en ese rango, mientras que el resto no aparecerán. La segunda sentencia, lo que hace es crear un grupo al que se podrán inscribir todos los elementos que queremos. En este caso estamos diciendo que todos los elementos dentro del grupo g se muestren como si estuvieran a una distancia de 800.

He dejado para el final la primera acción que se está realizando. Como ya dijimos antes, una vez que pasamos el ISBAN del Activity de Buscar, hay que recuperarlo en el Activity de RA y

éste es el caso. Vemos que además, se crea un objeto de la clase **BookDataBase**, que va a tener como único atributo *ArrayList<BookData>collectionCDU*. El **BookData** es otra clase que se ha definido para poder implementar la aproximación del usuario al libro que se comentó en el apartado 4.2.3:

```
public class BookData
{
  private int myEstanteria;
  private int myColor;
  private int myOrientation;
  private String myTopic;
  private String myDataBaseCDU;
```

En *collectionCDU* se va a grabar la base de datos que relaciona las estanterías con los identificadores de libros y los colores. Y es en el método *checkBookDataBaseBestMatching(ISBAN);* el que va a permitir obtener la información del libro que se ha pasado desde la aplicación y asociándole su estantería, su orientación, su color y su tema general.

```
public BookData checkBookDataBaseBestMatching (String currCDU)
{
  ArrayList<BookData> myCollection = collectionCDU;
  ArrayList<BookData> auxCollection = new ArrayList<BookData>();
  BookData bestMatchedBookData = null;
  String numCurrCDU = removeLettersFromCDU(currCDU);
  int maxCharMatched=0;
  int currCharMatched=0;
  BookData currBook;
  int i;

  for(i=0; i < myCollection.size();  i++)
  {
    currBook=myCollection.get(i);
    if(currBook.getDataBaseCDU().charAt(0) == numCurrCDU.charAt(0))
    {
      auxCollection.add(myCollection.get(i));
    }
  }
  for (i=0; i < auxCollection.size(); i++)
  {
    int index=0;
    currBook=auxCollection.get(i);

    while(index < numCurrCDU.length() && index < currBook.getDataBaseCDU().length())
```

```
    {
      if(currBook.getDataBaseCDU().charAt(index) == numCurrCDU.charAt(index))
      {
        currCharMatched++;
      }
      index++;
    }
    if(currCharMatched > maxCharMatched )
    {
      maxCharMatched = currCharMatched;
      bestMatchedBookData = currBook;
    }
    currCharMatched = 0 ;
  }
  return bestMatchedBookData;
}
```

En este punto ya tenemos la aplicación de RA corriendo, configurada y cargada con los datos que se le han pasado. El siguiente paso sería la detección del código LLA y la sucesión de acciones que esto desencadena. La detección del código es casi instantánea y se registra en la MetaioSDKCallbackHandler. Eso actualiza en el sistema la posición del usuario en ese momento, puesto que las coordenadas codificadas en el marcador LLA son reales. Es decir, se actualizan manualmente los datos de longitud, latitud y altitud en el GPS. Por lo que ya podemos dibujar los diferentes elementos en pantalla llamando al método *loadContent()* que vamos a ir viendo paso a paso.

En primer lugar, vamos a cargar la imagen de lo que Metaio define como POI y es aquello que consideramos relevante en la aplicación y lo queremos remarcar:

```
String filepath = AssetsManager.getAssetPath("ex/POI\_bg.png");
if (filepath != null)
  mMyPOI = metaioSDK.loadImageBillboard(createBillboardTexture("Tu libro"));
```

La variable *mMyPOI* es del tipo IGeometry que no es más que una interfaz geométrica que se puede cargar en el sistema. En este caso la imagen cargada va a ser la utilizada para indicar la estantería objetivo, y se le pondrá el literal “Tu libro”, fig. $4{,}7c$. Una vez que tenemos la figura que vamos a mostrar cuando apuntemos en la dirección correcta, vamos a cargar el letrero con la información relativa al libro que hemos obtenido usando *checkBookDataBaseBestMatching()* y el pie de la aplicación.

```
String imagePath = AssetsManager.getAssetPath("ex/pie_localizacion.png");
if (filepath != null)
{
  bImagePlane=true;
  mImagePlane = metaioSDK.createGeometryFromImage(imagePath);
  mTransparentPlane = metaioSDK.createGeometryFromImage(createBillboardTexture(
              "Estanteria: " + myBookData.getEstanteria()
              + "\nOrientación: " + getOrientation(myBookData.getOrientation())
              + "\nColor: " + getColor(myBookData.getColor())
              + "\nCDU: " + ISBAN   ));

  if(mImagePlane != null && mTransparentPlane != null)
  {
    bImagePlane=true;
    mImagePlane.setRelativeToScreen(IGeometry.ANCHOR\_BOTTOM, IGeometry.FLAG\_MATCH_DISPLAY );
    mTransparentPlane.setRelativeToScreen(IGeometry.ANCHOR\_BOTTOM, IGeometry.FLAG\_MATCH_DISPLAY);
    mImagePlane.setVisible(true);
    mTransparentPlane.setVisible(true);
    g.addBillboard(mImagePlane);
    g.addBillboard(mTransparentPlane);
    bImagePlane=false;

  }
}
```

Es en este punto, donde le debemos asignar a la estantería que estamos buscando los datos de longitud, latitud y altitud correspondientes, pero para ello vamos a explicar primero cómo obtuvimos dichos datos. Todo el escenario al que nos referimos está situado en la sala de lectura de la planta baja de la biblioteca. Hacer mediciones GPS para posicionar las estanterías es complicado debido al amplio margen de error que supone estar dentro de un edificio y a eso hay que añadirle el margen propio del sistema GPS.

La solución surgió a raíz de encontrar en Internet los planos de planta de la planta baja de la biblioteca, tal y como muestra la figura 4,23.

Para calcular los datos relacionados con la latitud y longitud se ha utilizado Google Maps. Se buscó la biblioteca en la aplicación de mapas del buscador y se ajustó la imagen de Google Maps a la del plano. Una vez superpuestas, se fueron determinando uno a uno los puntos relativos a cada columna y cada estantería. Se tuvieron en cuenta las pequeñas diferencias entre la distribución de estanterías del plano y la actual distribución de la biblioteca. Sin ser una solución precisa, se antojaba más precisa que la medición de las coordenadas GPS *indoor*.

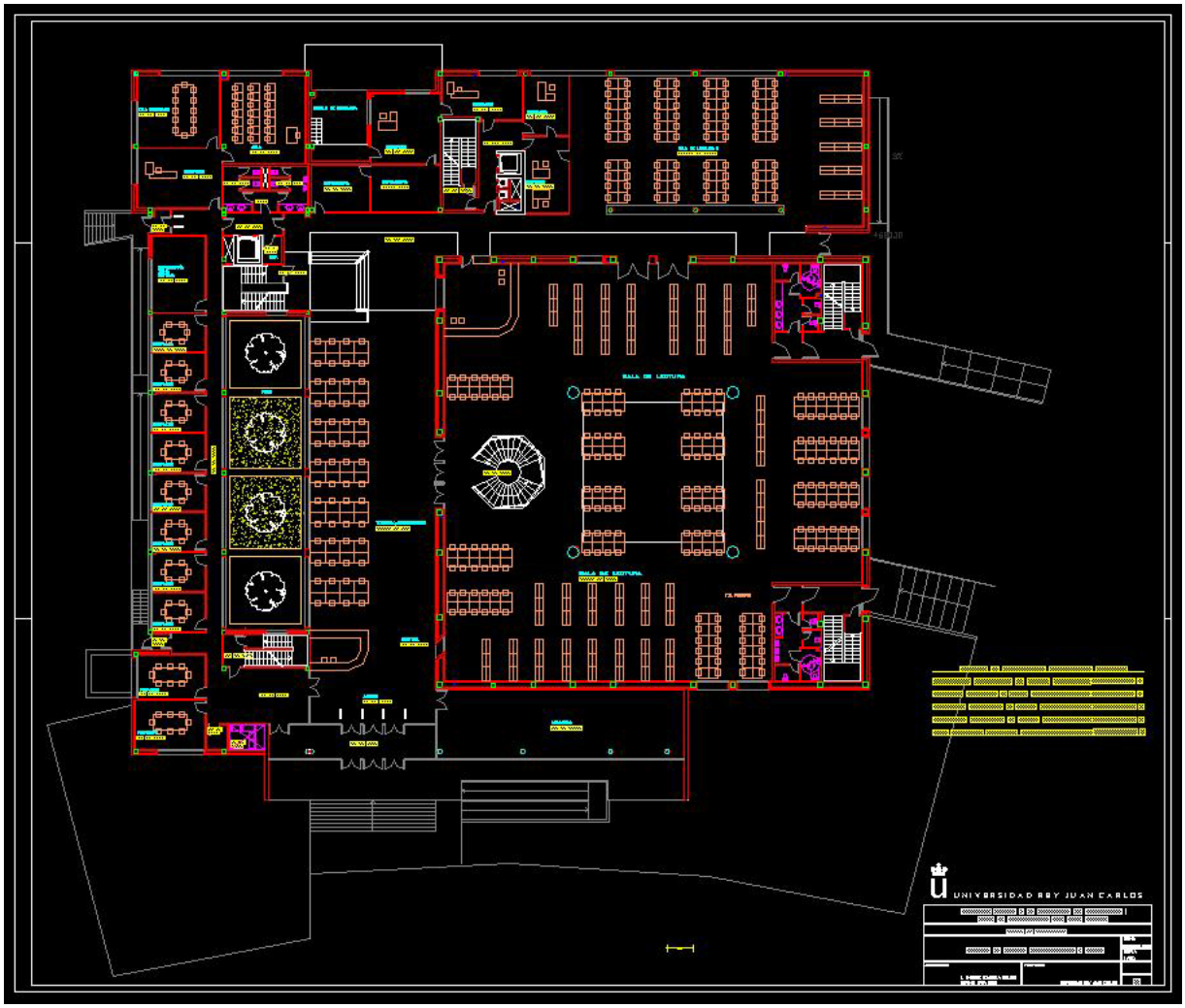


Figura 4.23: Planta baja de la biblioteca

Ahora que ya sabemos que coordenadas se le van a asignar a cada estantería, seguimos con el método loadContent() donde lo dejamos. Se estaba pretendiendo ubicar a mMyPOI en la aplicación. Esto lo vamos a conseguir de la siguiente manera:

```
private LLACoordinate myTarget= new LLACoordinate(0, 0,0, 0);
switch(myBookData.getEstanteria())
{
  case 1:
  myTarget.setLatitude(40.281048);
  myTarget.setLongitude(-3.81988);
  myTarget.setAltitude(664);
  break;

  [...]

  case 17:
```

```
    myTarget.setLatitude(40.281356);
    myTarget.setLongitude(-3.819772);
    myTarget.setAltitude(664);
    break;
  }

  myTarget.setAccuracy(10);
  mMyPOI.setTranslationLLA(myTarget);
  g.addBillboard(mMyPOI);
```

Una vez actualizada la posición del punto que vamos a buscar, solamente nos queda crear el radar que vamos a mostrar y actualizarlo con el punto que acabamos de crear para que nos indique su posición respecto a la nuestra. Para ellos hacemos lo siguiente:

```
mRadar = metaioSDK.createRadar();
mRadar.setBackgroundTexture(AssetsManager.getAssetPath("ex/radar.png"));
mRadar.setObjectsDefaultTexture(AssetsManager.getAssetPath("ex/yellow.png"));
mRadar.setRelativeToScreen(IGeometry.ANCHOR_TL);
mRadar.add(mMyPOI);
```

En este punto termina el método *loadContent()* y ya es misión de la SDK cargar todas las texturas sobre la pantalla y la aplicación de RA quedaría funcionando.

4.3.3. Google Maps

Tal y como se comentó en el apartado 4.2.4, se ha incrustado dentro de la aplicación una funcionalidad basada en Google Maps. En concreto se refiere a la versión ***Google Maps v2*** de Google Maps.

En primer lugar, como siempre que se mete un nuevo elemento a la interfaz gráfica, hay una definición dentro del layout en este caso de la pestaña de Información y es el siguiente:

```
<LinearLayout
    android:id="@+id/mapa"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:visibility="gone" >

    <fragment
      android:id="@+id/map_view"
```

```
    android:layout_width="fill_parent"
    android:layout_height="400dp"
    android:layout_marginBottom="4dp"
    android:layout_marginLeft="4dp"
    android:layout_marginRight="4dp"
    android:layout_marginTop="4dp"
    class="com.google.android.gms.maps.SupportMapFragment"
    android:clickable="true"
    android:enabled="true" />
</LinearLayout>
```

En él se definen las características que ha de tener el *Fragment* que va a albergar el mapa. Cabe destacar que la siguiente sentencia, *class="com.google.android.gms.maps.SupportMapFragment"*, se añade por que los mapas de la versión dos de Google no soportan la versión de Froyo que es para la que estamos realizando el desarrollo. Podíamos haber optado por usar los mapas de la versión uno pero se encuentran actualmente obsoletos y metiendo esa sentencia conseguimos que los terminales con esa versión de Android sean soportados.

Una vez que tenemos definido el aspecto y tamaño que tendrá el mapa en nuestra aplicación, nos queda inicializarlo, añadirle el marcador de la biblioteca, definir el tipo de mapa a utilizar y centrar el mapa en las coordenadas que nos interesa.

```
//=================== Google maps ============================
mapView = ((SupportMapFragment) getSupportFragmentManager()
  .findFragmentById(R.id.map_view)).getMap();

mapView.addMarker(new MarkerOptions()
  .position(BIBLIOTECA)
  .title("Biblioteca Universidad Rey Juan Carlos de Fuenlabrada")
  .snippet("Camino del Molino s/n 28943 - Fuenlabrada")
  .icon(BitmapDescriptorFactory
  .fromResource(R.drawable.ic_launcher)));

mapView.setMapType(GoogleMap.MAP_TYPE_HYBRID);

CameraPosition cameraPosition = new CameraPosition.Builder()
  .target(BIBLIOTECA)         // Sets the center of the Library
  .zoom(16)                   // Sets the zoom
  .build();                   // Creates a CameraPosition from the builder

mapView.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition));
```

En el código inmediatamente superior, podemos ver la implementación de las necesidades descritas anteriormente. Sin embargo, esto no es suficiente para hacer funcionar los mapas. Al igual que ocurría con la SDK de Metaio, hay que firmar la aplicación y obtener la *GOOGLE_API_KEY* que nos va a validar en el servidor de servicios de Google, que es desde donde se descargan los mapas. Para eso debemos ir al fichero donde definimos las características de la aplicación, en nuestro caso *Burjc Manifest*, y añadir lo siguiente:

```
<permission
  android:name="net.learn2develop.googlemaps.permission.MAPS_RECEIVE"
  android:protectionLevel="signature" />


<uses-permission android:name="net.learn2develop.googlemaps.permission.MAPS_RECEIVE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<meta-data
  android:name="com.google.android.maps.v2.API_KEY"
  android:value="GOOGLE_API_KEY" />
```

En *GOOGLE_API_KEY* se ha de sustituir la clave obtenida de Google.

4.3.4. E-mail

A lo largo de la memoria hemos hablado hasta en dos ocasiones de que la aplicación era capaz de enviar correos electrónicos. En este apartado vamos a ver cómo se consigue de dos formas distintas. La principal diferencia radica en el contexto donde se realiza la solicitud de envío.

En primer lugar vamos a hacer referencia a la pestaña “Contactar” del apartado 4.2.5, donde una de las opciones de contacto es el e-mail. En el siguiente código podemos observar cómo se le dio funcionalidad al botón “Enviar”:

```
public void sendEmail (View view){

  //Obtenemos los datos para el envío del correo
  TextView etEmail = (TextView) findViewById(R.id.toET);
  EditText etSubject = (EditText) findViewById(R.id.asuntoET);
```

```
    EditText etBody = (EditText) findViewById(R.id.bodyET);

    //Es necesario un Intent que levante la actividad deseada
    Intent itSend = new Intent(android.content.Intent.ACTION_SEND);

    itSend.setType("plain/text");

    //Colocamos los datos para el envío
    itSend.putExtra(android.content.Intent.EXTRA_EMAIL, new String[]{ etEmail.getText().toString()});
    itSend.putExtra(android.content.Intent.EXTRA_SUBJECT, etSubject.getText().toString());
    itSend.putExtra(android.content.Intent.EXTRA_TEXT, etBody.getText());

    startActivity(itSend);
}
```

El resultado es que el sistema operativo nos ofrece varios medios para mandar dicho mensaje, tal y como se observa en la fig. 4,14*b*. En este caso, al sistema le estamos pidiendo enviar un correo electrónico a *toET* que es un valor constante en la aplicación:

```
<string name="toET">biblioteca.websc@urjc.es</string>
```

Y de la misma forma le estamos indicando asunto y contenido.

El segundo método vamos a elegir la dirección de correo de una lista y ese va a ser el único dato que la aplicación pueda proveer. Por lo tanto, lo que se busca es facilitar la labor al usuario y en vez de tener que copiar la dirección de correo elegida en su aplicación de email, habiendo tenido que abrirla previamente y seleccionar el botón crear un nuevo correo. Vamos a hacer que con solo pulsar la dirección mostrada en pantalla, se muestren todas las opciones disponibles en el smartphone para el envío de mensajes. Y esto se consigue de la siguiente forma:

```
tv.setText(Html.fromHtml("<a href=\"mailto:"+p.text()+"\">"+p.text()+"</a>"));
tv.setMovementMethod(LinkMovementMethod.getInstance());
```

Esto lo que hace es permitir que cierto texto que estamos dibujando en la interfaz, se comporte como lo haría en un HTML consiguiendo un enlace a correo electrónico.

4.3.5. HTTP POST

Como ocurre con el e-mail, a lo largo de la memoria hemos mencionado repetidas veces que se realizan acciones POST sobre el servidor del catálogo o sobre el servidor de la biblioteca. Sin embargo, estos métodos POST se han implementado de dos formas diferentes en función de las necesidades. Es decir, en el caso de las pestañas *"Buscar"* y *"Mi Cuenta"*, para implementar

el parser se utilizaba un elemento WebView, mientras que en el resto de la aplicación se podía utilizar el cliente HTTP facilitado por el API de Android. Aunque la forma de implementarla sea similar vamos a analizar ambas en las siguientes líneas.

El primer caso va a ser el relacionado con el WebView. La primero será declarar un objeto de dicha clase y después definir los pares que vamos a enviar: identificador y valor. En el siguiente código podemos ver como un se ha declarado un objeto web de la clase WebView y se realiza un post con él.

```
byte[] post = EncodingUtils.getBytes(
              "searchdata1="+search.getText().toString()
              +"&srchfield1=GENERAL^SUBJECT^GENERAL^^Palabras o fras&library=FUENLABRAD",
              "BASE64");
web.postUrl("http://centauro.urjc.es/uhtbin/cgisirsi/?ps=CyojYuo5gO/CENTRAL/"
              +sessionNum+"/123",
              post);
```

En este caso es tan sencillo como definir un array del tipo byte, que posteriormente se va a usar junto a la dirección deseada en el método *postUrl()*.

El segundo caso se van a definir tres objetos: un cliente HTTP, un objeto *HttpPost* que le pasaremos como argumento al cliente HTTP y un objeto *HttpResponse* que recoja la respuesta del host

```
HttpClient httpclient = new DefaultHttpClient();
HttpPost httppost = new HttpPost("http://www.urjc.es/biblioteca/Formularios/PHPs");

try
{
  List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(4);
  nameValuePairs.add(new BasicNameValuePair("Nombre", nombreF.getText().toString()));
  nameValuePairs.add(new BasicNameValuePair("Consulta", consultaF.getText().toString()));

  if (radioEmail.isChecked())
     nameValuePairs.add(new BasicNameValuePair("Email", emailF.getText().toString()));
  else
    nameValuePairs.add(new BasicNameValuePair("Tel&eacute;fono", phoneF.getText().toString()));
  httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));

  // Execute HTTP Post Request
  HttpResponse response = httpclient.execute(httppost);
```

Capítulo 5

Pruebas y resultados

Se decidió realizar un periodo de pruebas con los funcionarios de la biblioteca. Para ello, se realizó una reunión en la biblioteca en la que se realizó un breve sesión de demostración de como funciona la aplicación. En ese momento la aplicación aún se encontraba en fase *beta*. Es decir, tenía sus funcionalidades completas pero aún faltaban aspectos por pulir.

El seguimiento de la fase de pruebas se ha realizado mediante informes regulares que se enviaban desde la biblioteca, informando de que modelos estaban siendo utilizados para probar y que problemas presentaban a la hora de realizar las pruebas. Por último se evaluaría el comportamiento de la aplicación y la satisfacción del usuario utilizando una encuesta que se encuentra en el Anexo A.

A continuación se muestra un listado de los dispositivos en los que se ha probado la aplicación y su versión de Android:

- Samsung Galaxy S - Android 2.2
- Samsung Galaxy S III - Android 4.3
- Samsung Galaxy Mini - Android 4.1.2
- Samsung Galaxy Y - Android 2.3.6
- Sony Xperia E - Android 4.1.1
- Sony Xperia Arc S - Android 4.2.1
- HTC Wildfire - Android 2.3.5

- LG Optimus L3 -Android 2.3.6

Los primeros problemas surgieron al poco de instalar la aplicación. La principal sorpresa fue que los modelos Samsung Galaxy Y y HTC Wildfire y LG Optimus L3 no encontraban la aplicación en Google Play, el *market* de aplicaciones de Google. Esto es así porque Google filtra las aplicaciones que muestra en el Google Play, al que accedemos desde nuestro teléfono, en función de las características del mismo y los requisitos mínimos de cada aplicación.

Mi primera investigación concluyó con que algunos móviles de gama media/baja no soportan la función auotfocus de la cámara del móvil. Esta característica aparecía como un requisito en el listado de la aplicación. El motivo por el que aparecía como requisito, es porque algunas funcionalidades de la SDK de Metaio la necesitan. Sin embargo, BURJC no las utiliza, así que se quitó y se subió al market de nuevo.

La **Google Development Console**, la aplicación web que proporciona Google a los desarrolladores para poder gestionar sus aplicaciones, te da información sobre el volumen de modelos de smartphone que soportan tu aplicación en función del listado de requisitos de la misma. Me sorprendió comprobar que solamente quitando el autofocus aumentaba casi en 1000 el número de modelos que soportan la aplicación. Esto significa que 4462 modelos, de los 6060 reconocidos por Google, soportan la instalación de BURJC. Esto supone un 73,6 % de la oferta total de móviles.

A pesar de esta modificación, los tres modelos citados seguían sin encontrar la aplicación en Google Play. Una segunda investigación, esta vez centrada en el tipo de arquitectura *hardware* del smartphone, revela que como requerimiento mínimo se necesita una **armeabi-v7a** o lo que lo mismo, la versión 7 de la arquitectura ARM. Sin embargo, los terminales antes citados tienen todos versiones 6 o inferiores por lo que no podrán soportar nunca la aplicación. En este caso no se podía hacer nada puesto que el requerimiento de nuevo viene impuesto por la SDK de Metaio.

Otro problema que me sorprendió, porque solo afectaba al modelo Samsung Galaxy S, consistía en que al realizar una búsqueda o cualquier tipo de consulta al servidor de la biblioteca, la aplicación se cerraba. Esta incidencia venía provocada por la utilización de la función *isEmpty()* de la clase *String*. Esta función solo estaba definida para versiones del API posteriores a la 2.2 de Android. Una vez fue sustituida por otra funcionalidad, este terminal ha podido realizar búsquedas sin problemas.

El siguiente problema más mencionado se reproducía de forma aparentemente aleatoria una vez que se pulsaba el botón *Atrás* mientras se estaba utilizando la funcionalidad de la realidad aumentada. Al volver a la aplicación, todos los caracteres mostrados por pantalla presentaban forma de cuadrado del color que tuviera el fuente previamente definido. Este problema solo se ha podido solucionar para terminales con versiones Android 3.0 o superior. El motivo es que hasta la versión 3.0, Android no permitía inhabilitar la aceleración hardware que era lo que estaba provocando el problema.

Sin embargo, el problema fundamental y más reportado tiene que ver con la realidad aumentada. Los problemas reportados a este respecto los podemos dividir en dos categorías: los de funcionalidad y de uso.

- Problemas de funcionalidad

En los primeros pasos funciona siempre, es decir, la aplicación de RA arranca y detecta los códigos LLA. El problema surge a partir de este punto y se dan tres situaciones. La primera es que el sistema funcione correctamente, en cuyo caso no habría incidencia. La segunda es que la situación de la estantería que estamos buscando no se encuentre donde nos indica la aplicación. Tercera, el terminal nos muestra la posición de la estantería que estamos buscando pero al mover el smartphone el sistema no actualiza la información del radar que se muestra en pantalla.

En cuanto a la segunda se realizó una profunda investigación al respecto puesto que durante la fase final de desarrollo de la aplicación ya se detectó que sucedía. Se cubrieron diversas hipótesis, entre ellas que la versión gratuita de Metaio no fuera todo lo precisa que era la versión de pago, que no se le estuvieran introduciendo bien los datos al GPS, o que la aplicación estuviera tomando lecturas GPS erróneas del satélite independientemente de las cargadas manualmente. Según la documentación aportada por Metaio, una vez que se actualizan los datos de forma manual al GPS, este deja de intentar recibir datos del satélite, es decir siempre espera que le actualizamos de forma manual. Por otro lado realice una batería de pruebas metiendo siempre los mismos datos, y buscando la misma estantería a distintas horas y un elevado número de veces. Las conclusiones eran que cuando la aplicación funcionaba correctamente el sistema apuntaba en la dirección adecuada. Pero si no era así, el sistema podía apuntar en cualquier dirección. Respecto a la primera posibilidad, quedó descartada la duda sobre esta incidencia cuando se encontró una respuesta oficial de los desarrolladores de la SDK Metaio, diciendo, que la precisión

de la aplicación dependía de la precisión del dispositivo GPS y de los sensores del smartphone que esté corriendo la aplicación. Es decir, la SDK de Metaio utiliza los sensores del teléfono para corregir los problemas de precisión del GPS pero si se producen errores de lectura en los sensores, se pueden producir errores en la localización. En cualquier caso admitieron que las imágenes publicadas en su página web no eran más que conceptuales, situaciones que se podrían llegar a conseguir en condiciones ideales.

- Problemas de uso

Otro de los problemas más reportados es que en los casos que la funcionalidad de realidad aumentada apuntaba correctamente a la estantería, cuando el usuario se acercaba andando al objetivo, el móvil dejaba de apuntar a la estantería correctamente. En este caso el problema no es de funcionamiento de la aplicación, si no de que no se le ha dado al usuario toda la información necesaria para utilizar la aplicación o ésta no es lo suficientemente clara. Una vez que la aplicación lee un marcador LLA te orienta hacia la estantería donde se encuentra el libro, pero no te guía hasta ella puesto que es imposible actualizar la posición GPS del terminal.

5.1. La encuesta y los resultados

En primer lugar vamos a explicar la estructura de la encuesta, sus distintas secciones y la intención de cada una de ellas.

Se trata de una encuesta extensa, compuesta de 90 preguntas divididas en 8 secciones. La mayor parte de las preguntas son de valoración, permitiendo elegir del 1 al 5 (siendo el 1 el valor de menor y el 5 el mayor puntuación). También se van a encontrar preguntas de “SI” o “No”, así como recuadros donde se le pedirá al encuestado su opinión si la tiene.

El motivo de haber hecho una encuesta tan extensa, a pesar de que esto lleve asociado correr ciertos riesgos, como que el encuestado pierda el interés, por ejemplo, es por el gran entusiasmo mostrado por la gente de la biblioteca durante la demostración de la aplicación, así como durante el proceso de desarrollo de la misma. Además, de esta forma, se puede tener una idea más precisa de cuanto nos hemos acercado a los requisitos impuestos por la biblioteca al principio del desarrollo y su grado de satisfacción.

Las secciones y su intención son los que siguen:

- **Preguntas generales**: En esta primera sección se pretende crear un perfil del encuestado y tomar las impresiones generales que tiene de la aplicación antes de entrar en detalle en cada una de sus secciones.

- **Google Play**: Se le pregunta al encuestado sobre la información que ha encontrado en el *market* de Google, así como su utilidad.

- **Diseño de la aplicación**: Se pregunta fundamentalmente por la interfaz de usuario, su parecido con la web de la universidad y su grado de sencillez.

- **Pestaña "Buscar", Realidad Aumentada, Pestaña "Información", Pestaña "Contactar", Pestaña "Mi cuenta"** : Se pregunta sobre el comportamiento, la utilidad y grado de satisfacción de cada una de estas funcionalidades.

Todos los resultados comentados a partir de este momento, están valorados siendo 1 el caso peor y 5 la mejor puntuación.

Finalmente, la encuesta fue cumplimentada por un grupo 10 personas compuesto de hombres y mujeres con edades comprendidas entre los 37 y los 51 años. Salvo uno, todos ellos se reconocen como personas con poca habilidad con la tecnología. Ocho de ellos dice haber probado la aplicación de 1 a 5 veces, mientras que los dos restantes la han probado de 5 a 10 veces o más de 10.

Un dato a destacar y con el que vamos a comparar el resto de datos a lo largo de todo el análisis de resultados de la encuesta, es que en media valoran el funcionamiento de la aplicación con un 2,2. Es decir, según ese primer dato la aplicación no habría gustado.

En la siguiente figura, se observa el resultado de calcular la valoración media en cada una de las secciones. Estas puntuaciones se obtienen sumando la puntuación en cada una de las preguntas en las que se evalúa del 1 al 5 y calculando la media de todas ellas. Se pueden consultar los datos totales y asociados a cada pregunta en el Anexo B.

Como se puede observar en la gráfica, si calculamos las medias de cada una de las secciones, la valoración individual es muy superior al 2,2 de la valoración de la aplicación en su conjunto, salvo en el caso de la realidad aumentada.

Según se explicó al principio del capítulo, en el apartado de *Preguntas generales*, se pretendía recoger el sentir general del usuario sobre la aplicación y sobre cada una de las secciones por

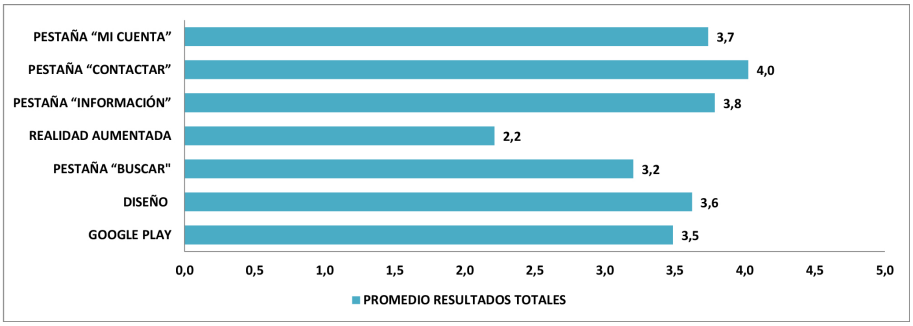


Figura 5.1: Resultado de la valoración media de cada sección

separado. En la siguiente figura vamos a comparar los resultados obtenidos en la figura anterior, fig. 5,1, con las valoraciones de las Preguntas generales.

Observando la figura fig. 5,2 se pude apreciar como la tendencia generalizada ha sido la de valorar en menor medida en la impresión general que cuando se les ha preguntado expresamente por cada una de las secciones.

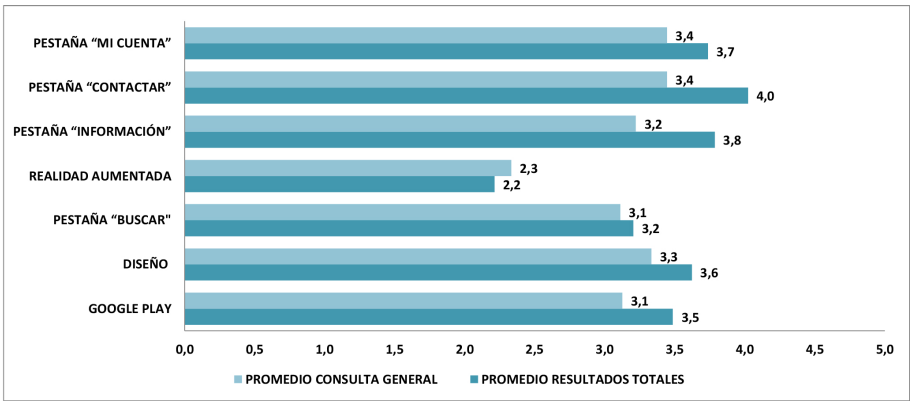


Figura 5.2: Resultado de la valoración media de cada sección comprada con la opinión de las Preguntas Generales

Un dato importante a tener en cuenta, es el relativo a la pregunta si se considera la aplicación una ayuda a la hora de interactuar con la biblioteca. Como se puede apreciar en la siguiente gráfica, fig. 5,3, mayoritariamente la opinión es que es que realmente si supone una ayuda a la hora de utilizar los recursos de la biblioteca.

De nuevo, se vuelve a plantear el contraste de la evaluación realizada sobre la aplicación y los datos proporcionados por los encuestados en relación al funcionamiento y la utilidad de la aplicación.

Una vez que hemos estudiado cual es la opinión de los encuestados de forma general y

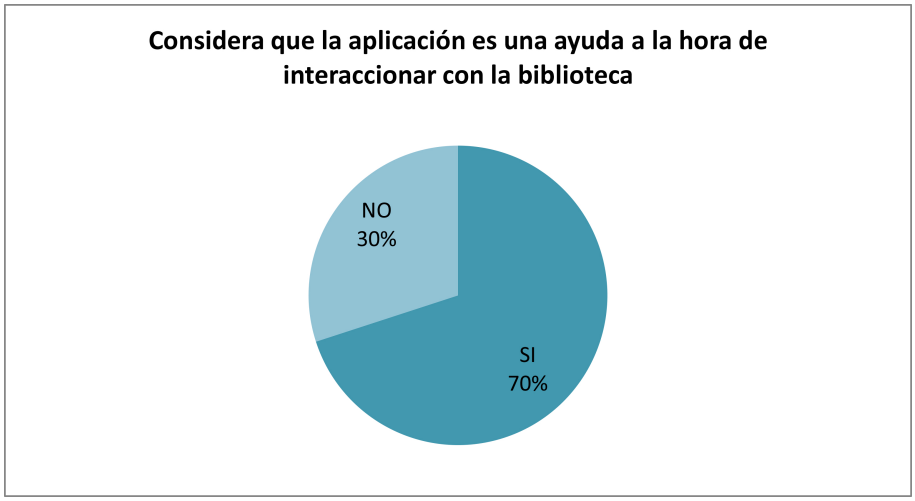


Figura 5.3: Ayuda a la interacción con la biblioteca

vemos que casi todas las secciones aprueban, vamos a centrarnos en la sección de la realidad aumentada y la que está íntimamente relacionada con ella la pestaña “Buscar”.

Como ya habíamos comentado anteriormente, una de las incidencias más reportadas tenía que ver con la realidad aumentada y este es el motivo de los siguientes resultados.

Figura 5.4: Índice de utilidad de la funcionalidad de realidad aumentada

Como se puede apreciar solo el 30 % de los encuestados cree que se trata de una funcionalidad que pueda serle útil. Entre las opiniones vertidas en las encuestas la más repetida hace referencia a que la aplicación no apunta en la dirección correcta. Sin embargo, hay un dato que llama la atención y es que a la pregunta *¿Qué información de la mostrada es la que considera más útil?*, de forma casi unánime han respondido que el letrero informativo que se muestra en

la parte inferior de la pantalla una vez que se localiza el libro.

También en relación a la realidad aumentada, tres de los encuestados ha coincidido en echar de menos unas instrucciones de cómo utilizar dicha funcionalidad. Este debe ser el principal motivo de que en el apartado de *Facilidad de uso* se haya valorado con un 2,6.

Si nos fijamos esta vez en la pestaña Buscar, la incidencia más reportada es que el buscador no soporta la ñ, ni caracteres acentuados, lo que da pista de por qué se ha recibido una evaluación de 2,8 en cuanto a su utilidad. Un dato que llama la atención puesto que una de las principales características de una aplicación de biblioteca es el buscador del catálogo de libros.

A pesar de los resultados en la realidad aumentada, cabe destacar los buenos resultados obtenidos en el resto de las pestañas. Lidera el ranking de valoración la pestaña "Contactar" evaluada con un 4.0, donde los encuestados destacan de entre todas, las opciones de comunicación el E-mail y el listado de contactos. La siguiente en aceptación es la pestaña "Información" valorada con un 3,8. En esta ocasión cada una de las secciones de la pestaña ha sido evaluada con un 4 y la claridad con la que se muestra la información un 3,6. Otra de las pestañas mejor valoradas es la referente a "Mi cuenta". Durante las reuniones con la biblioteca se comentó que problemas para acceder a esta información en la web, es que resulta difícil forma de acceso a ella, en el caso de la aplicación a la pregunta *Valore del 1 al 5 la forma de acceso* ha sido puntuado con un media de 3,9.

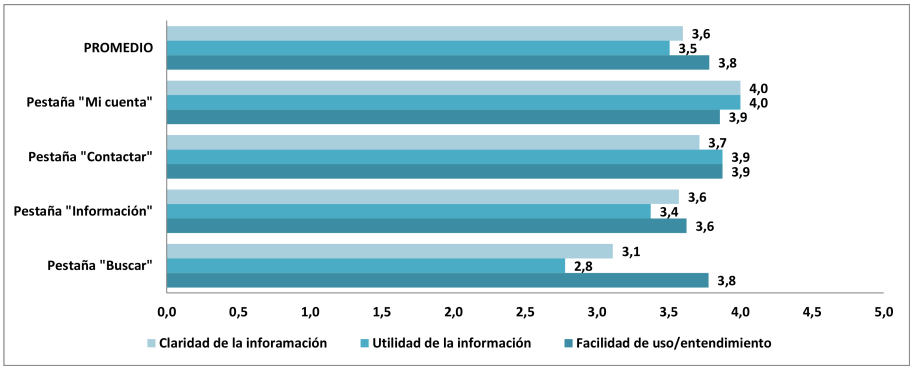


Figura 5.5: Análisis de los requisitos de la biblioteca

Por último, vamos a analizar los datos de la encuesta relacionados con los requisitos de una interfaz sencilla e información ordenada que propuso la biblioteca. Para ello vamos a estudiar los resultados a las preguntas relacionadas con la facilidad de uso/entendimiento y la utilidad y claridad de la información, para cada una de las pestañas.

En la figura, fig. 5,5 se puede observar que la puntuación otorgada por los encuestados a cada una de las cuestiones planteadas sobre el tratamiento de la información, en media, se encuentra por encima del 3,5. Parece que de forma generalizada, la aplicación aprueba holgadamente según el criterio de los encuestados en este apartado. Está la excepción de la pestaña "Buscar", que como ya hemos visto antes existían incidencias relacionados con las búsquedas con caracteres especiales.

Capítulo 6

Conclusiones

Para poder sacar conclusiones hemos de ser capaces de analizar los resultados. Para ello, vamos comparar de los requisitos definidos antes del comienzo del proyecto, con la opinión de los encuestados.

Centrándonos primero en nuestros requisitos, podemos concluir que se han cumplido. El proceso de este desarrollo a cubierto desde el diseño inicial, pasando por la recopilación de herramientas necesaria para su desarrollo, hasta la firma de la aplicación y la posterior subida a Google Play. Además, hemos podido hacer un desarrollo adicional para una funcionalidad basada en la realidad aumentada. Una tecnología puntera que desde el principio hemos considerado muy interesante.

Por otro lado, si nos fijamos en los requisitos impuestos por la biblioteca, ellos necesitaban una aplicación móvil que mostrara de forma ordenada toda la información relativa a la biblioteca del campus de Fuenlabrada y la normativa general de la biblioteca. Pero, que además, tuviera una interfaz sencilla que permitiera que cualquiera que no tenga conocimientos avanzados sobre aplicaciones móviles pudiera usar. Para hacer este análisis, vamos a recordar que prácticamente todas las personas encuestadas se reconocen con un perfil tecnológico bajo. Teniendo esto en cuenta, y basándonos en los resultados obtenidos por su evaluación de la aplicación móvil, según su opinión (Fig. 5,5), sí que hemos conseguido una aplicación que ellos consideran sencilla y que contiene la información que necesitan.

Sin embargo, a pesar de que les hemos proporcionado una aplicación que cumple con sus requisitos la evaluación de la misma ha sido de 2,2 sobre 5. En el momento de la propuesta del proyecto se les presentó el concepto de realidad aumentada y nuestra idea para usarla en la

biblioteca. LA aceptaron de buen grado ya que consideraron que podía ser de gran utilidad. En el capítulo referente a la encuesta $5{,}1$, podemos ver que cuando se consulta de forma específica por las características del resto de la aplicación, la evaluación es muy positiva. Esto me lleva a pensar que a pesar de que ellos definieron los requisitos que consideraban fundamentales, a la hora de evaluar inconscientemente le han dado mucha más importancia al comportamiento de la realidad aumentada, que al resto de la aplicación, condicionando así las demás opiniones.

Es importante mencionar el esfuerzo que se le ha puesto al desarrollo y las pruebas de la realidad aumentada, que se comieron una parte importante del tiempo destinado al desarrollo del proyecto. Es cierto que a pesar las ganas de hacerlo funcionar no se han obtenido los resultados deseados. La experiencia nos ha enseñado que, quizá por un exceso de ambición, hemos querido desarrollar una aplicación basada en una tecnología que aún no está madura en el ámbito de la localización en interiores.

Pero es precisamente el interés que provoca ese esfuerzo y la importancia inconsciente que le ha dado la gente de la biblioteca lo que me hace concluir que no deja de ser un proyecto muy interesante, que tiene potencial de futuro una vez que la tecnología esté más preparada para ello y que ha motivado por igual a personas con gran interés e la tecnología y a personas que no comparten ese interés.

6.1. Lecciones aprendidas

Para mí ha sido muy interesante seguir desde el principio el diseño y desarrollo de un proyecto de software. El haber podido experimentar que por muy concienzudo que pretendas hacer el diseño, siempre hay flecos sueltos y aspectos que, o no se han tenido en cuenta. o se desconocían y que plantean problemas adicionales que han de ser solucionados o que incluso obligan a cambiar parte del desarrollo por suponer una barrera que o no se puede o no interesa salvar.

Por otro lado .el haber tenido la figura del “cliente” presente a lo largo de todo el desarrollo, me ha dado la perspectiva del desarrollador, en la que unas veces recibes propuestas interesantes que se pueden asumir, pero que en otras ocasiones te toca ceñirte a los requisitos fijados desde el principio para no tener que cambiar el diseño en mitad del desarrollo.

Cuando se diseña un proyecto hay que prestar mucha atención a los detalles que pueden ser más sensibles para el cliente. En concreto me estoy refiriendo, al momento en el que se entregó

la versión beta de la aplicación. La funcionalidad de buscar libros tenía problemas que no se habían detectado durante la implementación y eso ha influido muy negativamente en la opinión del cliente sobre la aplicación. Esto se hubiese solucionado si se hubieran identificado con anterioridad las partes críticas del proyecto para realizar pruebas de una forma más intensivas sobre ellas.

6.2. Trabajos futuros

En este apartado se enumeran los posibles trabajos futuros para la aplicación:

- Pulir los pequeños problemas que presenta la aplicación a la hora de buscar libros con caracteres especiales como la ñ o los acentos.
- Darle la forma definitiva de una aplicación de producción
- Actualizar el desarrollo para versiones más modernas de Android
- En lo referente a la realidad aumentada caben varias posibilidades:
    - Realizar un desarrollo paralelo propio de un motor de realidad aumentada que funcione de forma específica para la biblioteca de la universidad y que sustituya al de Metaio.
    - Utilizar otra técnica de realidad aumentada basada en el *tracking* de marcadores, por ejemplo.

Anexos

Anexo A

Encuesta de satisfacción

ENCUESTA DE SATISFACCIÓN DE LA APLICACIÓN DE LA BIBLIOTECA DE LA UNIVERSIDAD REY JUAN CARLOS DE FUENLABRADA

PREGUNTAS GENERALES

Edad:

Sexo:

○ Hombre ○ Mujer

Valore del 1 al 5 sus conocimientos técnicos:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Cuántas veces ha usado la aplicación a lo largo del periodo de prueba:

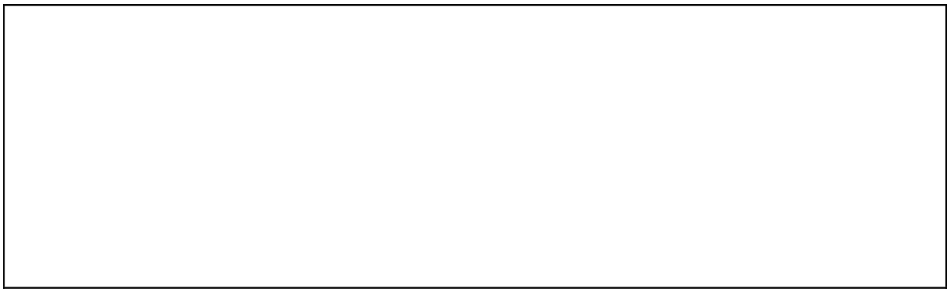
○ 1 a 5 ○ 5 a 10 ○ más de 10

Valore del 1 al 5 el buen funcionamiento de la aplicación:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

De haber tenido algún problema en el uso de la aplicación, por favor indique cual:

Cuál es su grado de satisfacción general con la aplicación, valore del 1 al 5:



○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Considera que la aplicación es una ayuda a la hora de interaccionar con la biblioteca:

○ Si ○ No

Valore del 1 al 5 el grado de satisfacción con los siguientes aspectos:

Información mostrada en Google Play

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Diseño general de la aplicación

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Pestaña buscar y sus funcionalidades

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Realidad aumentada

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

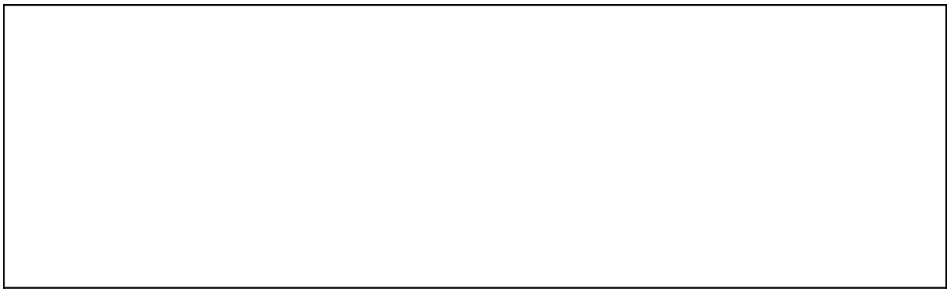
Pestaña información y sus funcionalidades

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Pestaña contactar y sus funcionalidades

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Pestaña mi cuenta y sus funcionalidades



○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Por favor, cumplimente este campo si desea añadir alguna otra información de utilidad:

GOOGLE PLAY

Valore del 1 al 5 la facilidad para encontrar la aplicación en Google Play:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 la descripción ofrecida sobre la aplicación:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 si la información ofrecida la invita a descargar la aplicación:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 cuan representativo le parece el icono de la aplicación:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore el 1 al 5 el diseño del icono:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 cuan representativas son las capturas de pantalla mostradas en la descripción:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Por favor, cumplimente este campo si desea añadir alguna otra información de utilidad:

DISEÑO DE LA APLICACIÓN

Valore el 1 al 5 el diseño general de la aplicación:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Con cuales de estos adjetivos describiría el diseño de la aplicación (puede escoger más de uno):

○ Profesional ○ Juvenil ○ Sencillo ○ Intuitivo ○ Confuso ○ Desordenado

○ Anticuado ○ Moderno ○ Original ○ Práctico

¿Le parece que el orden es el adecuado?

○ Si ○ No

En caso de haber contestado no a la anterior pregunta, por favor indique qué orden hubiese preferido y por qué:

Valore del 1 al 5 la claridad con la que se muestra la información:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore el 1 al 5 el tipo de letra con la que se muestra la información:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore el 1 al 5 la combinación de colores:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore el 1 al 5 los iconos elegidos:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 el diseño del pie de la aplicación:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Considera que hay un diseño general coherente en toda la aplicación:

○ Si ○ No

En caso de haber contestado no a la anterior pregunta, por favor indique qué orden hubiese preferido y por qué:

PESTAÑA "BUSCAR"

Valore del 1 al 5 su grado de satisfacción con la pestaña"Buscar":

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 la facilidad a la hora de realizar una búsqueda:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 el tiempo invertido en resolver una búsqueda:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

En cuanto a la información mostrada en una búsqueda, valore del 1 al 5 los siguientes aspectos:

Utilidad

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Organización de la información

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

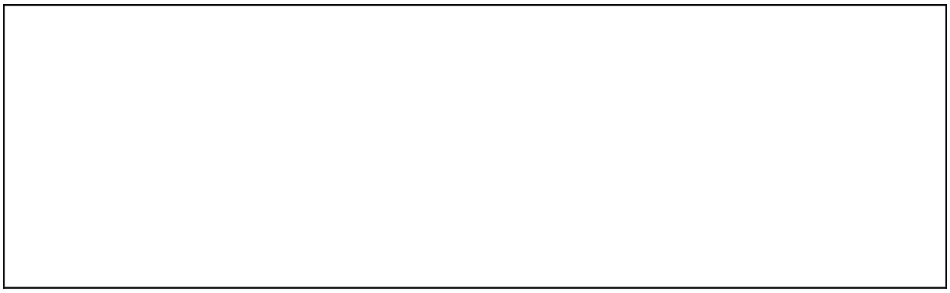
Facilidad para reconocer los distintos campos

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Indique brevemente qué es lo que más le ha gustado:

Indique brevemente qué es lo que menos le ha gustado: Por favor, cumplimente este campo

si desea añadir alguna otra información de utilidad:



REALIDAD AUMENTADA

Valore del 1 al 5 su grado de satisfacción con la funcionalidad de realidad aumentada:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

En cuanto a la información mostrada en una búsqueda, Valore del 1 al 5 los siguientes aspectos:

Utilidad

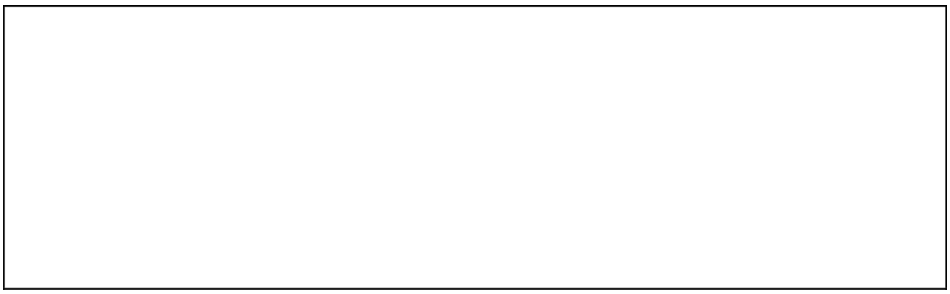
○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Facilidad de uso

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

¿Le ha resultado una ayuda a la hora de buscar el material deseado? Indique por qué.

○ Si ○ No



¿Le ha parecido suficiente la información mostrada?. Indique por qué.

○ Si ○ No

¿Le ha parecido relevante la información mostrada?. Indique por qué.

○ Si ○ No

¿Qué información de la mostrada es la que considera más útil?

¿Qué ha echado en falta?

Por favor, cumplimente este campo si desea añadir alguna otra información de utilidad:

PESTAÑA "INFORMACIÓN"

Valore del 1 al 5 su grado de satisfacción con la pestaña Información:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 la utilidad de la información:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 la organización de la información:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

¿Considera que la información mostrada es suficiente?

○ Si ○ No

En caso de haber contestado no a la anterior pregunta, por favor indique qué orden hubiese preferido y por qué:

Valore del 1 al 5 la claridad a la hora de mostrar la información:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 los siguientes apartados:

Horarios

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Normativa

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

FAQ (preguntas frecuentes

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Cómo llegar

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 el diseño de los iconos:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Por favor, cumplimente este campo si desea añadir alguna otra información de utilidad:

PESTAÑA "CONTACTAR"

Valore del 1 al 5 su grado de satisfacción con la pestaña "Contactar":

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 la utilidad de la información:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 la organización de la información:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

¿Considera que la información mostrada es suficiente?

○ Si ○ No

En caso de haber contestado no a la anterior pregunta, por favor indique qué orden hubiese preferido y por qué:

Valore del 1 al 5 la claridad a la hora de mostrar la información:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 la utilidad de los distintos medios de contacto ofrecidos:

E-mail

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Correo Postal

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Formulario

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Teléfono/Fax

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 el diseño de los iconos:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

¿Has utilizado alguna de los métodos de contacto ofrecidos en la aplicación para ponerte en contacto con la Biblioteca?

○ Si ○ No

En caso afirmativo, indica cual/cuales:

Por favor, cumplimente este campo si desea añadir alguna otra información de utilidad:

PESTAÑA "MI CUENTA"

Valore del 1 al 5 su grado de satisfacción con la pestaña "Mi cuenta":

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 la forma de acceso:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 la utilidad de la información mostrada:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Valore del 1 al 5 la organización de la información mostrada:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

¿Considera que la información mostrada es suficiente?

○ Si ○ No

En caso de haber contestado no a la anterior pregunta, por favor indique qué orden hubiese preferido y por qué:

Valore del 1 al 5 la claridad a la hora de mostrar la información:

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Por favor, cumplimente este campo si desea añadir alguna otra información de utilidad:

Anexo B

Resumen de resultados de la encuesta

En la siguiente figura se muestra el resumen de los resultados de la encuesta. Se ha decidido poner en la página siguiente para que entre a pantalla completa.

	PROMEDIO RESULTADOS NUMÉRICOS
PREGUNTAS GENERALES	**3,0**
Valore del 1 al 5 el buen funcionamiento de la aplicación	2,2
Cuál es su grado de satisfacción general con la aplicación, valore del 1 al 5	2,3
Información mostrada en Google Play	3,1
Diseño general de la aplicación	3,3
Pestaña buscar y sus funcionalidades	3,1
Realidad aumentada	2,3
Pestaña información y sus funcionalidades	3,2
Pestaña contactar y sus funcionalidades	3,4
Pestaña mi cuenta y sus funcionalidades	3,4
GOOGLE PLAY	**3,5**
Valore del 1 al 5 la facilidad para encontrar la aplicación en Google Play	3,1
Valore del 1 al 5 la descripción ofrecida sobre la aplicación	3,1
Valore del 1 al 5 si la información ofrecida la invita a descargar la aplicación	3,0
Valore del 1 al 5 cuan representativo le parece el icono de la aplicación	4,0
Valore el 1 al 5 el diseño del icono	4,0
Valore del 1 al 5 cuan representativas son las capturas de pantalla mostradas en la descrip	3,7
DISEÑO DE LA APLICACIÓN	**3,6**
Valore el 1 al 5 el diseño general de la aplicación:	3,7
Valore del 1 al 5 la claridad con la que se muestra la información	3,4
Valore el 1 al 5 el tipo de letra con la que se muestra la información	3,7
Valore el 1 al 5 la combinación de colores	3,7
Valore el 1 al 5 los iconos elegidos	3,6
Valore del 1 al 5 el diseño del pie de la aplicación	3,7
PESTAÑA "BUSCAR"	**3,2**
Valore del 1 al 5 su grado de satisfacción con la pestaña "Buscar"	3,2
Valore del 1 al 5 la facilidad a la hora de realizar una búsqueda	3,8
Valore del 1 al 5 el tiempo invertido en resolver una búsqueda	3,0
Utilidad de la información mostrada en la búsqueda	2,8
Organización de la información de la información mostrada en la búsqueda	3,1
Facilidad para reconocer los distintos campos en información mostrada en la búsqueda	3,3
REALIDAD AUMENTADA	**2,2**
Valore del 1 al 5 su grado de satisfacción con la funcionalidad de realidad aumentada	1,8
Utilidad de la información mostrada en la búsqueda	2,3
Facilidad de uso de la información mostrada en la búsqueda	2,6
PESTAÑA "INFORMACIÓN"	**3,8**
Valore del 1 al 5 su grado de satisfacción con la pestaña Información	3,4
Valore del 1 al 5 la utilidad de la información	3,4
Valore del 1 al 5 la organización de la información	3,6
Valore del 1 al 5 la claridad a la hora de mostrar la información	3,6
Horarios	4,0
Normativa	4,0
FAQ (preguntas frecuentes)	4,0
Cómo llegar	4,0
Valore del 1 al 5 el diseño de los iconos	4,1
PESTAÑA "CONTACTAR"	**4,0**
Valore del 1 al 5 su grado de satisfacción con la pestaña "Contactar"	3,9
Valore del 1 al 5 la utilidad de la información	3,9
Valore del 1 al 5 la organización de la información	3,9
Valore del 1 al 5 la claridad a la hora de mostrar la información	3,7
E-mail	4,3
Correo Postal	4,1
Formulario	4,1
Teléfono/Fax	4,3
Valore del 1 al 5 el diseño de los iconos	4,1
PESTAÑA "MI CUENTA"	**3,7**
Valore del 1 al 5 su grado de satisfacción con la pestaña "Mi cuenta"	3,6
Valore del 1 al 5 la forma de acceso	3,9
Valore del 1 al 5 la utilidad de la información mostrada	4,0
Valore del 1 al 5 la organización de la información mostrada	4,0
En caso de haber contestado no a la anterior pregunta, por favor indique qué orden hubies	3,0
Valore del 1 al 5 la claridad a la hora de mostrar la información	4,0
PROMEDIO GENERAL	**3,4**

Bibliografía

[1] Stephen Cawood, Mark Fiala . *Augmented Reality: A Practical Guide*. The Pragmatic Bookshelf, 2008.

[2] Rogers, Rick. *Android application development*. O'Reilly, 2009.

[3] Bussinnes Wire Web -Android market penetration rate.

`http://www.businesswire.com/news/home/20130516005342/en/Android-iOS-Combine-92.3-Smartphone-Operating-System#.U3iN7fl_vZF`

[4] Android Web Site - Android Arquitecture

`http://source.android.com/devices/index.html`

[5] Handheld Augmented Reality

`http://studierstube.icg.tugraz.at/handheld_ar/signpost2003.php`

[6] Metaio Developer Portal

`http://dev.metaio.com/sdk/api-reference/main-page/`

[7] Metaio Helpdesk

`http://helpdesk.metaio.com/`

[8] Android Developers

`http://developer.android.com/intl/es/reference/packages.html`

[9] Biblioteca Urjc Fuenlabrada

`http://www.urjc.es/biblioteca/fuenlabrada/`