ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

InfoLab. Aplicación de reconocimiento de imágenes en Android para usuarios de laboratorios

AUTOR: Pedro Juan Sánchez Belchí
DIRECTOR: Esteban Egea López
Septiembre / 2014

Autor	Pedro Juan Sánchez Belchí
E-mail del Autor	pedrojuan20@hotmail.com
Director	Esteban Egea López
E-mail del Director	esteban.egea@uptc.es
Coordinador(es)	
Título del PFC	InfoLab. Aplicación de reconocimiento de imágenes en Android para usuarios de laboratorios
Descriptores	Android, Reconocimiento de imagen

Resumen

El objetivo de este Proyecto Fin de Carrera es el desarrollo de una aplicación para dispositivos Android de reconocimiento de imagen que sea capaz de identificar los equipos de los laboratorios y ofrecer información relevante de ellos.

Toda la información estará almacena en un servidor web Apache y una base de datos MySQL. Se tendrá acceso a ella en tiempo real, en el momento en el que el usuario lo requiera, mediante la lectura por parte de la aplicación de un código QR que identifica el laboratorio en el que se encuentra. Tras ello, basta con enfocar el objeto del que se necesita información con la cámara del smartphone para que se muestre en pantalla todos los datos que puedan ser de utilidad.

Gracias a la elección de la plataforma Android podrá ser utilizada en una gran variedad de dispositivos y por la mayoría de alumnos.

Titulación	Ingeniero Técnico de Telecomunicación, especialidad en Telemática
Intensificación	
Departamento	Tecnologías de la Información y Comunicaciones
Fecha de presentación	Septiembre 2014

A mis padres, por su paciencia
y apoyo todos estos años.

Tabla de contenido

1. Introducción

1.1. Motivación

El desarrollo de la aplicación tratada en este trabajo, viene motivado por la necesidad de hacer conocer a los usuarios de laboratorios de la universidad los equipos por los que estos están compuestos. Los laboratorios pueden estar formados por todo tipo de dispositivos de cualquier índole que pueden no ser familiares para los usuarios, sobre todo para los primerizos. Se quiere dotar a los alumnos de una nueva herramienta educativa que les permita acceder a esta información. Información que no debe quedar solamente en cuál es el nombre de un dispositivo, el fabricante o el modelo, sino también acceder a todo un conjunto de información ampliada y detallada de gran utilidad para el alumno, como instrucciones de seguridad, manual de usuario, ejemplos, prácticas implicadas, uso adecuado, web del fabricante, etc. A todo esto, el alumno debería poder acceder en cualquier momento y al instante, de un modo fácil, rápido y sin necesidad de buscar entre una gran pila de datos. Esto otorgaría a los alumnos de un recurso extra para su aprendizaje y correcto desarrollo de las prácticas, permitiéndoles obtener respuestas sin necesidad de recurrir al profesorado con preguntas que aunque puedan parecer triviales son necesarias para la realización de cualquier práctica y beneficiar al profesorado con más tiempo para centrarse en otros temas más relevantes.

Por otra parte, hoy día la tecnología nos ofrece todo tipo de avances que nos pueden hacer la vida más cómoda. Tecnologías como la visión artificial y la realidad aumentada (RA), que se han convertido en los últimos años en un área de investigación activa. La visión artificial es un subcampo de la inteligencia artificial que abarca diversas disciplinas como procesado de imágenes y reconocimiento de patrones y cuyo propósito es programar un computador para que “entienda” una escena o las características de una imagen. Con la realidad aumentada lo que se pretende es crear una realidad mixta en tiempo real, donde se combina un entorno físico del mundo real con elementos virtuales.

Conociendo también que en estos últimos años Google, con su sistema operativo Android basado en el *kernel* de Linux para dispositivos móviles, se ha hecho con una enorme cuota de mercado (según Google cada día más de 1 millón de nuevos dispositivos Android se activan en todo el mundo). El utilizar esta tecnología nos permite llegar a un gran número de usuarios. Android, junto con sus socios está empujando constantemente los límites del hardware y software, proporcionado nuevas capacidades tanto a usuarios como a desarrolladores. Para los desarrolladores, la innovación Android les permite crear potentes aplicaciones que utilizan las últimas tecnologías móviles. Desarrollar para Android posibilita acceder a una plataforma de desarrollo de libre distribución y código abierto, a librerías de reconocimiento de imagen para el uso en desarrollos, documentación y la utilización de toda una gama de herramientas gratuitas.

De la unión de estos tres conceptos:

- La necesidad por parte de los usuarios de los laboratorios de conocer los equipos que van a utilizar en sus prácticas.
- Los avances existentes en tecnología de reconocimiento de imagen.
- La posibilidad de crear aplicaciones con herramientas gratuitas que podrán ser utilizadas por cualquiera de los numerosos poseedores de un dispositivo Android.

Se ve la necesidad de crear una aplicación de reconocimiento de imagen para Android, de carácter docente, dirigida a los usuarios de laboratorios de la universidad, que les permita en cualquier momento reconocer un dispositivo y descargar toda la información asociada al mismo, de un modo fácil, rápido e incluso divertido, con el simple hecho de enfocar con la cámara de un dispositivo móvil Android.

1.2. Objetivos

Con este trabajo se pretende crear una herramienta docente que proporcione una nueva forma de acceder a la información, de familiarizarse con los elementos de un laboratorio. Todo ello mediante una aplicación para dispositivos Android, aprovechando el uso de tecnología de reconocimiento de imagen y herramientas de libre distribución.

Según se ha discutido en el apartado anterior los objetivos son:

1. Desarrollo de una aplicación para móviles Android de reconocimiento de objetos de un laboratorio mediante librerías de reconocimiento de imagen. En particular, que sea capaz de:
    a. Reconocer códigos QR que identifican los laboratorios.
    b. Descargar de un servidor web la información requerida.
    c. Reconocer los equipos que contiene un laboratorio.
    d. Mostrar en pantalla la información del equipo identificado.
2. Desarrollo de una aplicación web para la gestión de los dispositivos de laboratorio reconocibles y la información asociada.
3. Creación de una BBDD que contendrá la información relativa a los objetos de los laboratorios.
4. Integración de la aplicación de reconocimiento y el servidor web.

1.3. Fases del PFC

Para la realización de este proyecto se ha seguido un proceso de actuación que se describe a continuación:

1. Planteamiento de las necesidades que debe cumplir la aplicación
2. Elección de la solución más viable y adecuada para las necesidades planteadas.
3. Desarrollo de la aplicación cliente
    a. Diseño de una interfaz de usuario sencilla, amigable e intuitiva.
    b. Búsqueda de una librería para el reconocimiento de imagen.
    c. Programación de la aplicación en sí.
4. Desarrollo de la una aplicación web para la gestión de la información.
    a. Creación de la BBDD que contendrá la información relativa a los equipos de los laboratorios.
    b. Desarrollo de la aplicación web.
5. Documentación y redacción.

1.4. Contenido de la memoria

A continuación se describe brevemente el contenido de los diferentes apartados de los que se compone esta memoria para hacer más fácil su lectura.

En el **segundo capítulo**, se habla del estado actual de las tecnologías utilizadas en el desarrollo del proyecto.

En el **tercer capítulo**, se describe la arquitectura del sistema diseñado e implementado, explicando el esquema general del sistema.

En el **cuarto capítulo**, se detallan las soluciones que se han desarrollado y las herramientas con las que se ha realizado.

En el **quinto capítulo**, trata sobre el manual de usuario orientado a los clientes.

En el **sexto capítulo**, se recogen las conclusiones sacadas tras la realización de este proyecto y las posibles líneas futuras de mejora.

En el **séptimo capítulo**, encontramos la bibliografía y referencias utilizadas para el desarrollo de este proyecto.

2. Tecnologías empleadas

2.1. Dispositivos móviles

Los dispositivos móviles son aparatos de pequeño tamaño, con capacidad de procesamiento, con conexión permanente o intermitente a una red, que tienen memoria limitada, diseñados específicamente para una función, pero que pueden llevar a cabo otras funciones más generales. Hoy en día podemos encontrar una multitud de dispositivos móviles, donde los teléfonos móviles son los tipos de dispositivos más utilizados.

Los **smartphones** (1) o teléfonos inteligentes es un término comercial para denominar una nueva modalidad de teléfonos móviles, construidos sobre una plataforma informática móvil, incorporan muchas más capacidades de procesado y de movilidad que los dispositivos tradicionales. El desarrollo de la tecnología microelectrónica y de las redes de telecomunicaciones es lo que ha hecho posible su aparición y su popularización, de forma que son uno de los dispositivos tecnológicos multifunción más demandados en la actualidad.

Pequeños, elegantes y versátiles, los modernos dispositivos móviles inteligentes se han convertido en poderosas herramientas que incorporan pantallas táctiles, cámaras, reproductores multimedia, sistema de posicionamiento global (GPS), WIFI, capacidad de navegación web, completo soporte al correo electrónico y un importante tamaño de memoria entre otras muchas cualidades como la instalación de programas adicionales, que pueden ser desarrollados por el fabricante del dispositivo, por el operador o por un tercero.

Los avances tecnológicos han convertido a los teléfonos móviles en mucho más que un simple aparato para realizar llamadas, pueden ser una útil herramienta de trabajo o para el ocio.

2.1.1. Sistemas operativos de dispositivos móviles

Un sistema operativo es el conjunto de programas que actúan como intermediario entre el usuario y el hardware de un computador. Su propósito es proporcionar un entorno en el cual el usuario pueda ejecutar programas. Es el software básico que controla una computadora, gestiona sus recursos y responde a las peticiones del usuario. El sistema operativo tiene tres grandes objetivos: provee de un ambiente conveniente de trabajo, hacer uso eficiente del hardware y proveer de una adecuada distribución de los recursos.

En la Ilustración 1 se puede comprobar el uso de los principales sistemas operativos móviles que existen actualmente.

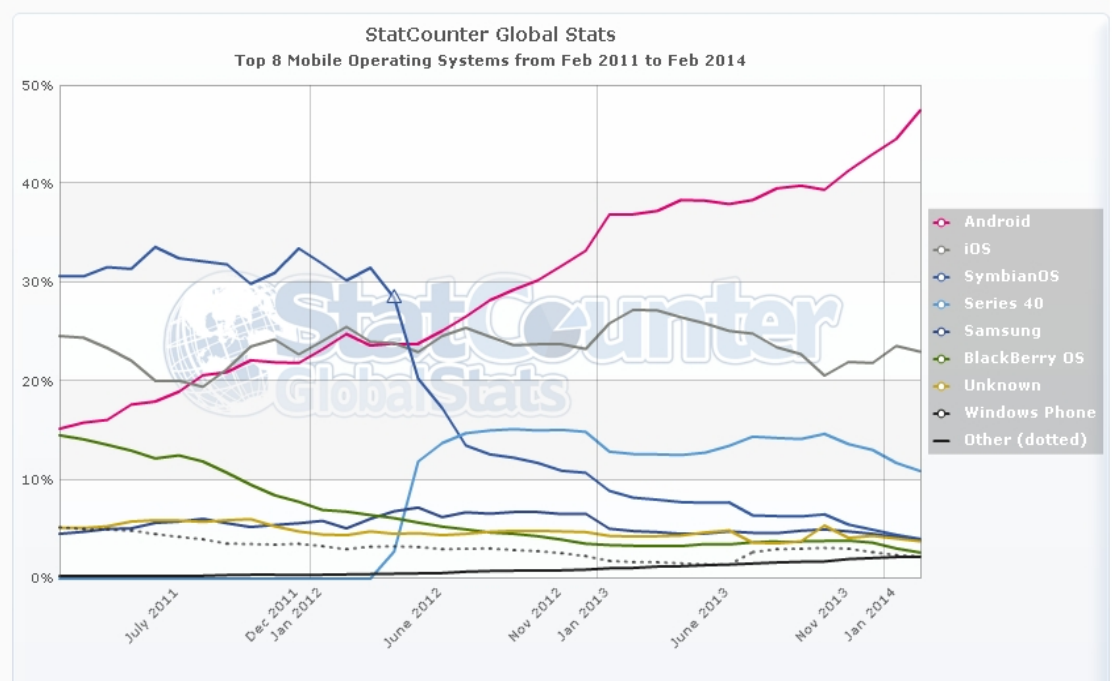


Ilustración 1. Cuotas de mercado mundial de Smartphones en función de su S.O.

Como podemos ver, a día de hoy el sistema operativo más utilizado en dispositivos móviles inteligentes en el mundo es Android, con casi el 50% de la cuota de mercado. Cuota que si nos centramos en Europa es aún mayor como podemos ver en el siguiente gráfico.

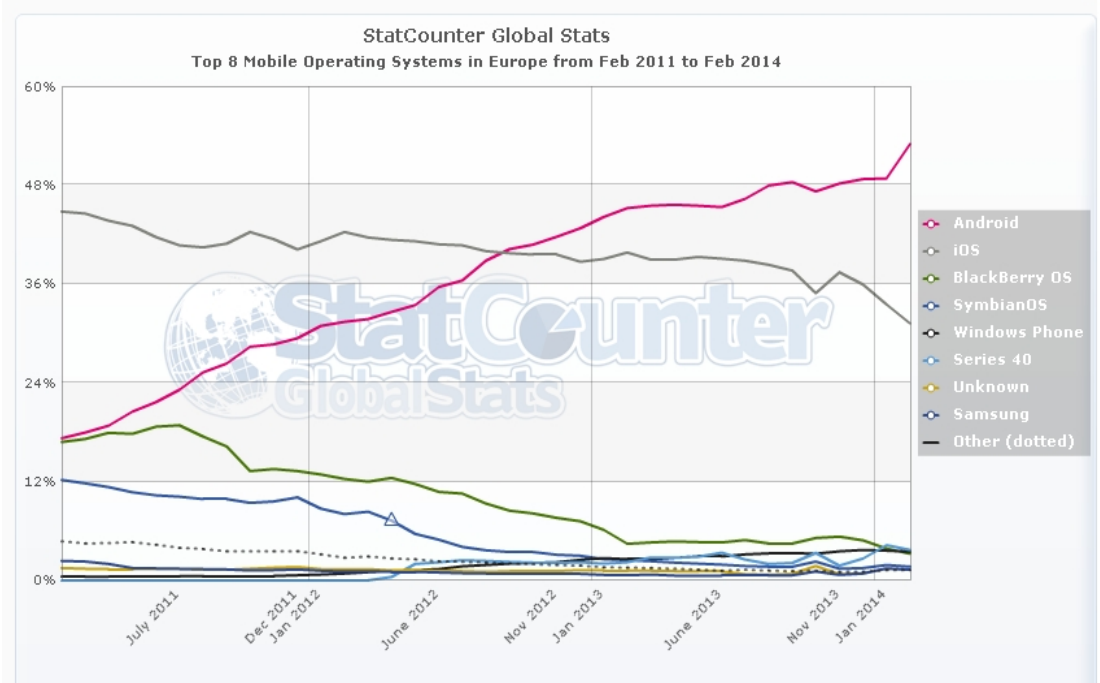


Ilustración 2. Cuotas de mercado en Europa de Smartphones en función de su S.O.

Si nos centramos ahora en las cuotas de mercado de España (Ilustración 3), se puede apreciar en la gráfica que Android es el sistema operativo más utilizado con diferencia, con una espectacular cuota de casi el 80%, seguido muy de lejos por iOS de Apple.

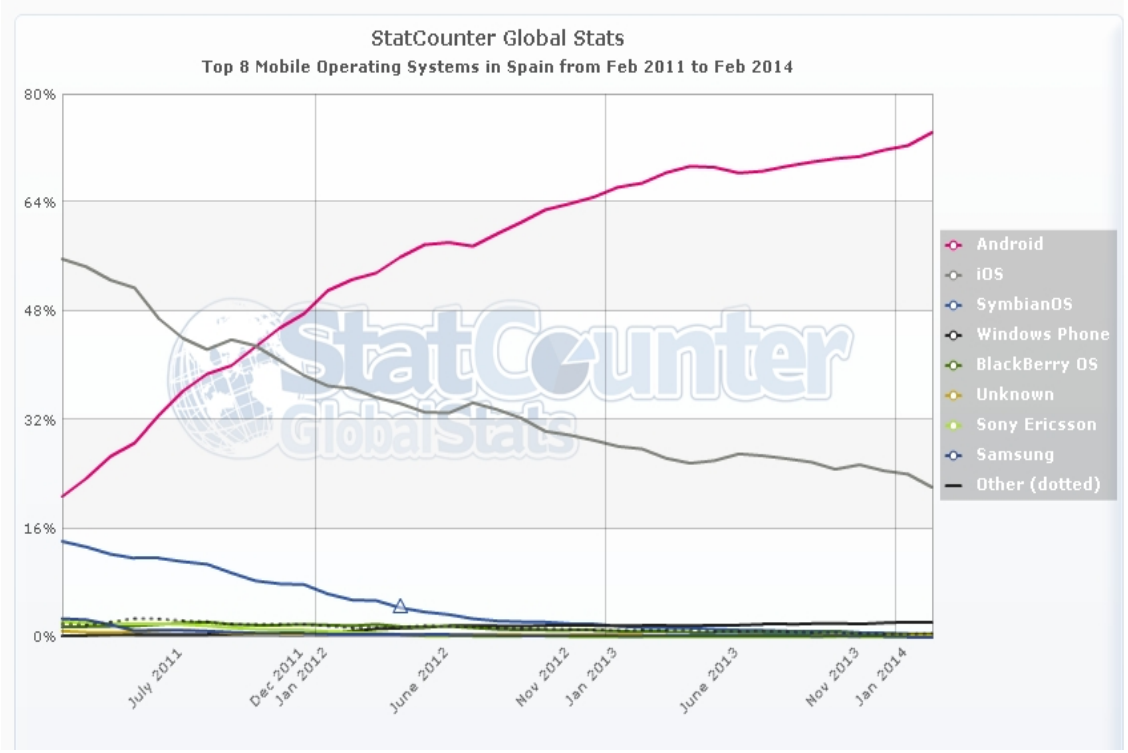


Ilustración 3 – Cuotas de mercado en España de Smartphones en función de su S.O.

Los sistemas operativos para Smartphone más populares son:

- **Android**, propiedad de Google. Basado en una versión modificada de Linux. Destaca por ser de licencia libre y código abierto. Es acto para una gran variedad de hardware de distintos fabricantes.
- **iOS**, propiedad de Apple. Versión reducida del sistema Mac OSX para PC. Diseñado específicamente para el dispositivo iPhone, un Smartphone de precio elevado pero de gran calidad.
- **Symbian OS**, propiedad de Nokia. En su día revolucionó y dominó el mercado. Hoy día sigue siendo una opción en dispositivos modestos cercano a las típicas funciones de un teléfono móvil.
- **Windows Phone**, propiedad de Microsoft. Su última versión (Windows Phone 8) sustituye a la arquitectura previamente basada en Windows CE por una basada en el kernel de Windows NT. Ofrece una interfaz muy intuitiva y espectacular. Su mayor inconveniente, el escaso catálogo de aplicaciones del que dispone actualmente.

- **BlackBerry OS**, originariamente dirigido al mundo empresarial por su gestión de correo y agenda, aunque tuvo éxito también entre los jóvenes gracias a su servicio de mensajería instantánea. Se caracteriza por tener teclado físico y su propio cliente de mensajería.

Ilustración 4 – Logos de los S.O. móviles más populares

2.2. Android

Android (2) es un sistema operativo móvil que se basa en una versión modificada de Linux. Originalmente fue desarrollado por una *startup*[1] del mismo nombre, Android Inc. En 2005, como parte de su estrategia para entrar en el mercado móvil, Google adquirió Android y se hizo cargo de su trabajo de desarrollo. Fue presentado en 2007 por la *Open Handset Alliance*, un consorcio compuesto por 84 compañías, liderado por Google y que se dedica al desarrollo de estándares abiertos para dispositivos móviles.

Google quería Android para ser abierto y libre, por lo que la mayor parte del código de Android fue lanzado bajo licencia Apache, una licencia libre y de código abierto, lo que significa que cualquier persona que quiera utilizar Android puede hacerlo y descargar el código fuente. Por otra parte, los proveedores (normalmente fabricantes de hardware) pueden agregar sus propias extensiones propietarias a Android y personalizar el sistema operativo para diferenciar su producto de los de la competencia. Este hecho además de una ventaja que ha favorecido su propagación tiene un inconveniente con las actualizaciones, ya que aunque Google lance una nueva versión, tiene que pasar antes por los fabricantes, que la adaptan a sus terminales (si quieren) antes de enviarla al usuario. El resultado es que a veces pasan varios meses hasta que se actualiza, en el caso de que llegue.

Aunque inicialmente enfocado para su uso en móviles y tablets con el tiempo se ha extendido a netbooks, televisores, relojes e incluso en gafas.

[1] Startup – Término utilizado para referirse a compañías emergentes apoyadas en la tecnología que pretenden emprender o montar un nuevo negocio con ideas que innovan el mercado.

2.2.1. Características

Debido a que Android es de código abierto y de libre acceso, no hay configuraciones de hardware o de software fijo, lo que permite a los fabricantes la personalización de sus dispositivos. Algunas de las características más destacadas de Android (3) son:

- Almacenamiento: Utiliza SQLite, una base de datos relacional ligera para el almacenamiento de datos.
- Conectividad: Soporta GSM, IDEN[2], CDMA[3], UMTS, 3G, Bluetooth, WIFI, LTE, WiMAX, etc.
- Mensajería: soporta SMS y MMS.
- Navegador Web: basado en el *WebKit* de código abierto.
- Multimedia: Incluye soporte para medios como H.264, MPEG-4, AMR, ACC, MP3, Ogg, WAV, JPEG, etc.
- Hardware: acelerómetro, cámara, brújula digital, sensor de proximidad, GSP, etc.
- Multi-touch: compatible con pantallas multi-táctiles.
- Multitarea: soporta aplicaciones multitarea.
- Utiliza la máquina virtual *Dalvik*, optimizada para dispositivos móviles.
- Etc.

2.2.2. Desarrollo de aplicaciones

Los dispositivos Android suelen venir con un paquete de aplicaciones nativas que forman parte del proyecto Android Open Source (AOSP) como un cliente de correo electrónico, una aplicación de gestión de SMS, un gestor de contactos y calendario, un navegador web basado en *WebKit*, una galería de imágenes y reproductor de música, una aplicación para la captura de imágenes y videos, un reloj con alarma y una calculadora entre otras. También es normal que incorporen aplicaciones de propiedad de Google como el "Google Play Store" para la descarga de aplicaciones de terceros, una aplicación de Google Maps o el cliente de correo Gmail entre otras. Además se dispone de un enorme catálogo de aplicaciones desarrolladas por terceros tanto gratuitas como de pago.

[2] IDEN. Red Mejorada Digital Integrada. Tecnología inalámbrica desarrolla por Motorola en 1994, proporciona a los usuarios múltiples servicios en un único e integrado sistema de comunicaciones móviles.

[3] CDMA. Término genérico para varios métodos de multiplexación o control de acceso al medio basados en la tecnología de espectro expandido.

La mayoría de aplicaciones Android están escritas usando el lenguaje Java, pero no se ejecutan dentro de una máquina virtual Java ME (Mobile Edition), sino que se compilan en un ejecutable *Dalvik* y corren en la máquina virtual *Dalvik*, diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados.

El éxito en el desarrollo para Android se basa en tres componentes:

- Un sistema operativo libre, de código abierto para dispositivos embebidos.
- Una plataforma de desarrollo de código abierto para la creación de aplicaciones.
- Amplia gama de dispositivos, en especial los teléfonos móviles, que ejecutan el sistema operativo Android y las aplicaciones creadas para ellos.

El verdadero atractivo de Android como un entorno de desarrollo se encuentra en sus APIs y la reutilización de código, además de disponer de un emulador, herramientas de depuración y una extensión para el programa eclipse.

2.2.3. Arquitectura

El sistema operativo Android (2) se divide a grandes rasgos en cinco secciones y cuatro capas principales.

- **Kernel Linux** – Es el núcleo sobre el que se basa Android. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.
- **Bibliotecas** – Éstas contienen todo el código que proporciona las principales características de un sistema operativo Android. Por ejemplo la biblioteca *WebKit* proporciona funcionalidades para la navegación web.
- **Runtime de Android** – En la misma capa que las bibliotecas, proporciona un conjunto de las bibliotecas base que permiten a los desarrolladores escribir aplicaciones Android usando el lenguaje de programación Java. También incluye la máquina virtual Dalvik, que permite a todas las aplicaciones de Android ejecutarse en su propio proceso, con su propia instancia de la máquina virtual Dalvik.
- **Marco de trabajo de aplicaciones** – Expone las distintas capacidades del sistema operativo Android a las aplicaciones de los desarrolladores para que puedan hacer uso de ellas reutilizando componentes. Las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas.
- **Aplicaciones** – Es la capa superior, en ella se encuentran las aplicaciones que se incluyen con el dispositivo Android como teléfono, contactos, navegador, etc. También se encuentran en ella las aplicaciones que se descargan e instalan los usuarios.

En la Ilustración 5 podemos ver la arquitectura aquí descrita.

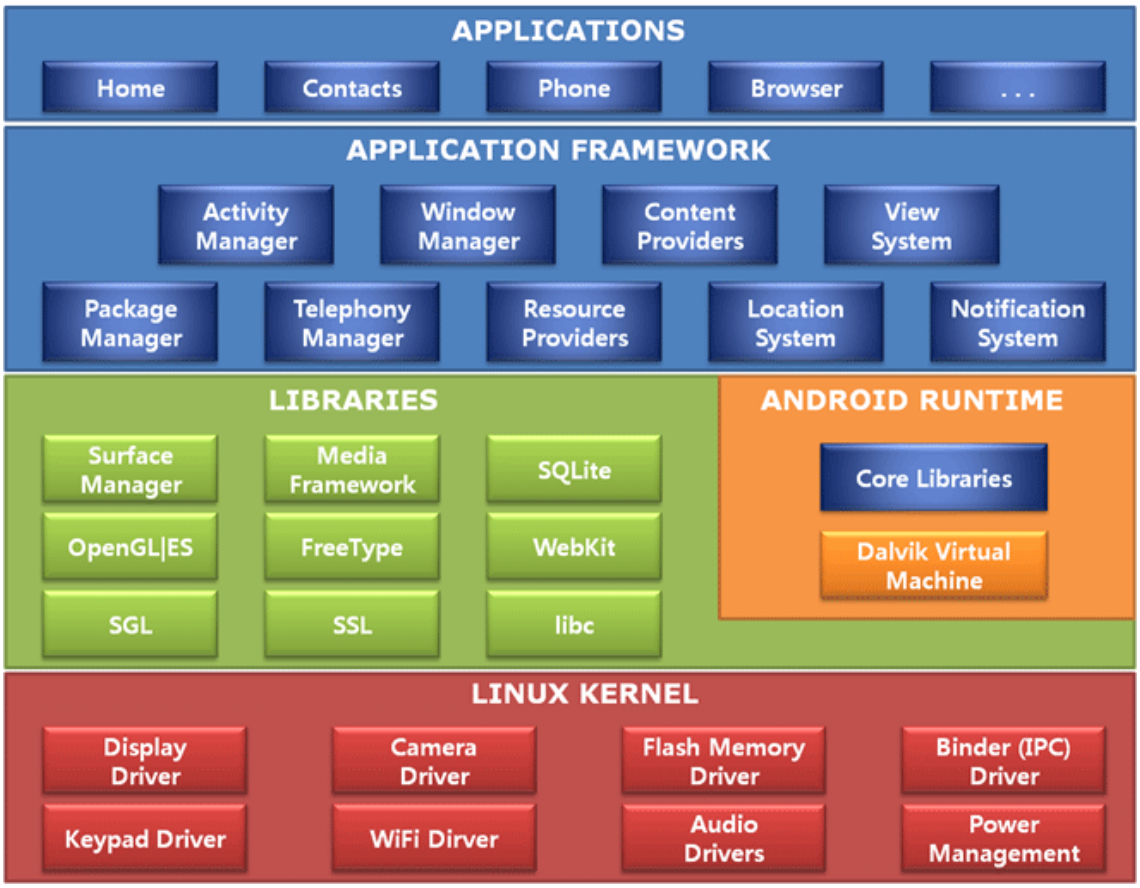


Ilustración 5 – Arquitectura de Android

2.2.4. Android SDK

El kit de desarrollo de software (Software Devolopment Kit o SDK) es el conjunto de herramientas básicas que permiten compilar y depurar aplicaciones creadas para el sistema operativo Android. También incluye librerías, documentación, códigos de ejemplo, etc. Está soportado tanto para Windows, Linux y Mac.

2.3. Códigos QR

Un código QR (Quick Response, respuesta rápida en español) (4) es un código de barras bidimensional creado en 1.994 por la compañía japonesa Denso Wave. Fue creado inicialmente para la industria automovilística, pero posteriormente se popularizó fuera de esta industria por su rápida lectura y gran capacidad de almacenamiento de datos en comparación con los códigos de barras tradicionales (aproximadamente 100 veces más información). Se caracterizan por los tres cuadrados que se encuentran en sus esquinas y

que permiten detectar la posición del código al lector. Un dato importante es que a diferencia de otros códigos de barras bidimensionales, su código es abierto y sus derechos de patente no son ejercidos.

Estos códigos se diseñaron para ser leídos como una imagen en dos dimensiones, que posteriormente es analizada digitalmente por un software. Los códigos QR pueden leerse desde un PC, Smartphone o tableta mediante escáner o la cámara de fotos y un software para interpretar los datos.

Los códigos QR han evolucionado con los años incrementando la información y el tipo de ésta que pueden almacenar. Van desde la versión 1 a la 40, cada versión tiene una configuración de módulo diferente (se refiere a los puntos blancos y negros que conforman el código). Cada número de versión superior consta de 4 módulos adicionales por lado.

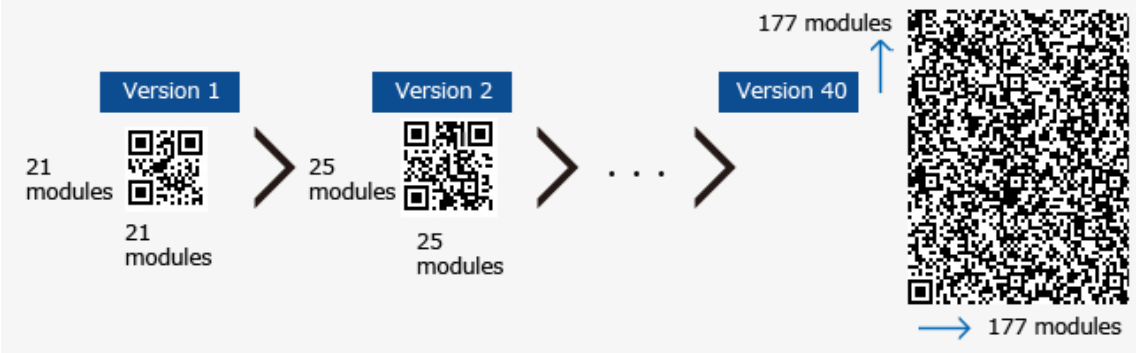


Ilustración 6 – Número de módulos en las versiones de códigos QR

Destacar también que estos códigos tienen la capacidad de corrección de errores para restaurar los datos si el código está dañado o sucio. Existen cuatro niveles de corrección de errores. El aumento de nivel mejora la capacidad de corrección pero también aumentan el tamaño del código.

2.4. Realidad aumentada

La realidad aumentada (AR) (5) agrupa aquellas tecnologías que permiten la superposición, en tiempo real de imágenes, marcadores o información, generados virtualmente, sobre imágenes del mundo físico. Se crea de esta manera un entorno en el que la información y los objetos virtuales se fusionan con los objetos reales.

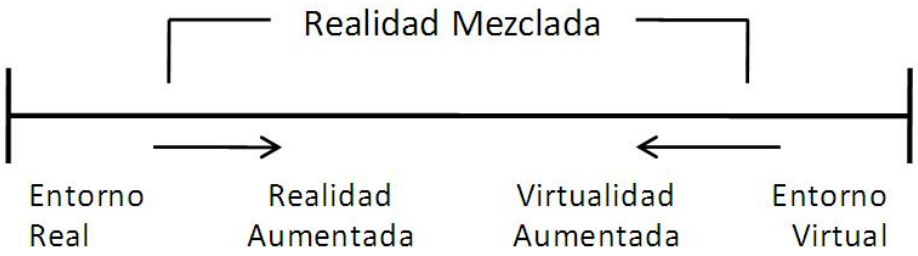


Ilustración 7 – Modelo de mezcla de entornos reales y virtuales

Para que la realidad aumentada sea posible tienen que intervenir varios elementos:

- **Un marcador o Target** que podría denominarse "activador de realidad aumentada", que es el que reconoce la aplicación de realidad aumentada y que a partir de ella el sistema debería reaccionar. Este activador puede ser la imagen que están visualizando los usuarios, pero debido a la complejidad que este proceso requiere, en muchos casos es sustituido por otros métodos como elementos de localización GPS, brújulas y acelerómetros que permiten identificar la posición y orientación. También se suelen apoyar en etiquetas, marcadores del tipo RFID (identificación por radiofrecuencia), códigos bidimensionales o cualquier otro elemento que sea capaz de suministrar una información equivalente a la que proporcionaría lo que el usuario ve.
- **Una cámara** que captura las imágenes de la realidad y para que el marcador pueda ser leído.
- **Una pantalla** donde visualizar el resultado.
- **Una aplicación** que realice el procesamiento e interprete la información del mundo real.

La aplicación que realiza la función de RA, ha de hacer uso de tecnologías como la visión por computador y el reconocimiento de objetos para, incorporar datos e información en un entorno real.

Los adelantos en dispositivos móviles, fundamentalmente en los Smartphone, han dado lugar a que hoy en día podamos disfrutar de estas aplicaciones y que la realidad aumentada esté ya posicionada para entrar en el sector de consumo de forma generalizada. Pero el elemento esencial para esta situación es la disponibilidad de conectividad permanente, ya que precisamente la potencialidad de gran parte de estos servicios es poder acceder a la información digital complementaria a la del mundo físico, actualizada en tiempo real y esto sólo es posible gracias a las infraestructuras y redes de telecomunicaciones.

2.5. Procesado digital de imagen

El procesado digital de imagen (PDI) (6) se refiere a procesar las imágenes del mundo real de manera digital por medio de un computador. Es un tema muy amplio, en el que se incluyen estudios de física, matemáticas, ingeniería eléctrica o computación. Estudia los fundamentos conceptuales de la adquisición y despliegue de imágenes y con detalle los fundamentos teóricos y algorítmicos del procesamiento como tal. Tiene además, como objetivo mejorar el aspecto de las imágenes y hacer más evidente en ellas ciertos detalles que se desean hacer notar.

La historia del procesado digital de imagen se remonta a la década de los 60 y está directamente ligada con el desarrollo y evolución de las computadoras. Su propósito ha ido de la mano con el desarrollo de las tecnologías de hardware, ya que requiere de un alto poder y recursos computacionales para almacenar y procesar las imágenes. De igual manera el desarrollo de los lenguajes de programación y los sistemas operativos han hecho posible el crecimiento continuo de aplicaciones relacionadas al procesamiento de imágenes, tales como imágenes médicas, de satélite, astronómicas, geográficas, arqueológicas y aplicaciones industriales entre otras.

2.5.1. Visión artificial

La visión artificial o visión por computador es un subcampo de la inteligencia artificial. Puede ser definida como los procesos de obtención, caracterización e interpretación de información de imágenes tomadas de un mundo tridimensional. Estos procesos pueden ser subdivididos en seis áreas principales y están agrupados de acuerdo a la complicación y delicadeza que lleva su implementación:

Procesos del PDI	Nivel de Visión
1. Captura/adquisición	Bajo
2. Preprocesamiento	
3. Segmentación	Medio
4. Descripción	
5. Reconocimiento	
6. Interpretación	Alto

Ilustración 8 – Niveles de visión y procesos del PDI

- **La captura o adquisición** es el proceso a través del cual se obtiene una imagen digital utilizando un dispositivo de captura como una cámara digital, video cámara, escáner, etc.
- **El preprocesamiento** incluye técnicas tales como la reducción del ruido, realce del contraste, realce de ciertos detalles, o características de la imagen.
- **La segmentación** es el proceso que divide una imagen en objetos que sean de nuestro interés de estudio.
- **La descripción** es el proceso que obtiene características convenientes para diferenciar un tipo de objeto de otro, como la forma, tamaño, área, etc.
- **El reconocimiento** es el proceso que identifica los objetos.
- **La interpretación** es el proceso que asocia un significado a un conjunto de objetos reconocidos y trata de emular la cognición.

No todas las aplicaciones de PDI requieren de todos los procesos descritos anteriormente. Por lo general, mientras la complejidad del problema a resolver crece, el número de procesos requeridos también crece.

2.5.2. Motor de reconocimiento de imagen "ImageMatching"

En la realización de este proyecto se ha hecho uso de la librería "*ImageMatching*" de reconocimiento de imagen desarrollada por la compañía ARLab.

ARLab (7) es una compañía líder en el campo de la Realidad Aumentada y visión por computador. Después de años de investigación y desarrollo, se ha convertido en un referente mundial en la industria y apoya a sus clientes desde Madrid y Haifa (Israel). ARLab ofrece una amplia carta de soluciones tecnológicas para la Realidad Aumentada. Su valor diferenciado es apoyar la creación no sólo de aplicaciones útiles, fáciles y rápidas, sino también las tecnologías que permiten un desarrollo eficiente.

La librería "*ImageMatching*" proporciona un potente motor de reconocimiento de imágenes en tiempo real. Soporta miles de imágenes añadidas en pilas de hasta 80 elementos y es capaz de obtener resultados en milisegundos gracias al aprovechamiento de la potencia hardware de los Smartphones para realizar el procesado.

ImagenMatching es compatible para la versión Android 2.3.3 o superior y con una arquitectura de procesador ARMv7 o superior. Es distribuida gratuitamente por el fabricante para desarrollos siempre que sea sin ánimo de lucro y no se suba a *Google Play Store*.

2.6. Base de datos

2.6.1. Introducción

Una base de datos es una serie de datos organizados y relacionados entre sí que se almacenan de forma sistemática para su uso posterior. Para acceder y modificar esos datos se emplea un software denominado **sistema de administración de bases de datos (DBMS).**

El objetivo principal de un DBMS es proporcionar una forma de almacenar y recuperar la información de una base de datos, de manera que sea tanto práctica como eficiente. Algunos de los servicios básicos requeridos por un DBMS son:

- Mover los datos de archivos de datos físicos, según sea necesario.
- Administrar la opción de que varios usuarios consulten datos de manera concurrente, e incluir medidas que eviten que las actualizaciones simultáneas tengan conflictos entre sí.
- Controlar las transacciones para que los cambios en la base de datos de cada transacción sean una unidad de trabajo tipo todo o nada.
- Permitir un lenguaje de consulta, que es un sistema de comandos empleado por el usuario de la base de datos para recuperar sus datos.
- Proporcionar medidas para respaldar la base de datos y recuperarla después de un fallo.
- Aportar mecanismos de seguridad para evitar la consulta y modificación no autorizadas de datos.

2.6.2. MySQL

MySQL (8) es un sistema de administración de bases de datos (DBMS) para bases de datos relacionales. Técnicamente es una aplicación que permite administrar archivos llamados bases de datos, aunque es normal escuchar el término “base de datos” para referirse exactamente a lo mismo.

Una base de datos relacional utiliza múltiples tablas para almacenar información, clasificándola en sus componentes más elementales. Aunque esto implica un mayor esfuerzo de diseño y programación, ofrece una fiabilidad e integridad de datos muy superior, que compensan con creces el esfuerzo adicional que requiere su manejo.

El software de MySQL consta de varios elementos, entre los que figuran el servidor MySQL que se encarga de ejecutar y administrar las bases de datos, el cliente MySQL que proporciona una interfaz para el servidor y numerosas utilidades de mantenimiento de

otra índole. Se puede interactuar con MySQL usando los lenguajes de programación más populares, como PHP, Perl y Java.

MySQL fue escrito en C y C++ y funciona exactamente igual en distintos sistemas operativos. Es capaz de manejar bases de datos de hasta 60.000 tablas, más de cinco mil millones de filas y puede trabajar con tablas de hasta ocho millones de terabytes.

Fue creado y sigue siendo desarrollado en la actualidad por MySQL AB, compañía Sueca, aunque ahora subsidiaria de Sun Microsystems, y esta a su vez de Oracle Corporation. Hasta cierto punto, MySQL creció a partir de una base de datos *open source* existente, mSQL, la cual sigue en fase de desarrollo, aunque su popularidad ha decaído. Afortunadamente, MySQL está muy lejos de sus modestos orígenes y ha pasado a convertirse en una aplicación de bases de datos robusta, fiable y fácil de manejar. MySQL se las ha ingeniado para conservar sus raíces *open source* y sigue estando disponible para ser usada o modificada sin tener que pagar nada por ello. Las características que ofrece MySQL en la actualidad explican por qué organizaciones de enorme peso mundial, como la NASA, Facebook, Twitter, YouTube o Yahoo!, la utilizan en sus operaciones diarias.

2.7. Servidor Web

2.7.1. Introducción

Un servidor Web es una colección de protocolos y estándares que sirven para intercambiar datos, a través de una red o de Internet, entre distintas aplicaciones. Estas aplicaciones pueden estar desarrolladas en lenguajes de programación diferentes y pueden ser ejecutadas sobre cualquier plataforma. Gracias a la adopción de estándares abiertos se consigue resolver los problemas de interoperabilidad. El servidor Web se ejecuta en un ordenador manteniéndose a la espera de peticiones por parte de un cliente.

Además de la transferencia de código HTML, los servidores Web pueden entregar aplicaciones Web que son porciones de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Se pueden distinguir entre:

- **Aplicaciones en el lado del cliente**. El servidor proporciona el código de las aplicaciones al cliente y éste, mediante el navegador, las ejecuta en su máquina. Son aplicaciones tipo Java “applets” o Javascript.

- **Aplicaciones en el lado del servidor**. El servidor Web ejecuta la aplicación, la cual al ejecutarse genera código que es enviado al cliente.

Las aplicaciones de servidor muchas veces suelen ser la mejor opción, debido a que al ejecutarse éstas en el servidor y no en la máquina del cliente, no necesita ninguna capacidad añadida. Así pues el cliente tan solo necesita disponer de un navegador Web básico.

2.7.2. Servidor Web Apache

Apache es el servidor Web más utilizado, líder con el mayor número de instalaciones a nivel mundial, muy por delante de otras soluciones como el IIS (Internet Information Server) de Microsoft. Apache es un proyecto de código abierto y uso gratuito, multiplataforma, muy robusto y que destaca por su seguridad y rendimiento.

La arquitectura del servidor Apache es modular. El servidor consta de una sección *core* o núcleo que realiza todas las funcionalidades importantes del sistema y diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor Web. El servidor de base puede ser extendido con la inclusión de módulos externos, lo que permite de una manera muy sencilla la posibilidad de ampliar sus funcionalidades de un modo muy flexible que le permiten adecuarse a las necesidades requeridas en cada escenario. Los módulos de Apache se pueden clasificar en tres categorías:

- **Módulo base**: módulo con las funciones básicas de Apache.
- **Módulos multiproceso**: son los responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender a las peticiones.
- **Módulos adicionales**: cualquier otro módulo que le añada una funcionalidad al servidor.

Por otro lado Apache tiene la capacidad de servir varios sitios web diferentes al mismo tiempo. Esto se llama Hospedaje Virtual (en inglés Virtual Hosting).

2.8. Lenguajes de programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones, y es utilizado para controlar el comportamiento físico y lógico de una máquina.

Los lenguajes de programación se pueden clasificar según:

- Según su **nivel de abstracción** en bajo nivel, medio nivel y alto nivel.
- Según su **forma de ejecución** en lenguajes compilados e interpretados.
- Según el **paradigma de programación** que posee cada uno de ellos. Los principales son: imperativos, declarativos y orientados a objetos.

Lenguajes de programación hay en gran cantidad, algunos han evolucionado a lo largo del tiempo y siguen vigentes en el transcurso de muchos años, mientras que otros han estado operativos durante un periodo de tiempo más o menos largo y actualmente no

se usan. Dada la gran variedad de lenguajes no se va a hablar de todos ellos, sólo de los utilizados para la realización de este proyecto.

2.8.1. PHP

El lenguaje PHP (cuyo nombre es acrónimo recursivo de PHP: Hipertext Preprocessor) es un lenguaje interpretado de código abierto muy popular de sintaxis similar a la de C++ o Java. Fue uno de los primeros lenguajes de programación del lado del servidor y que se puede incrustar en HTML. Aunque PHP se puede usar para realizar cualquier tipo de programa, es en la generación dinámica de páginas web donde ha alcanzado su máxima popularidad (9).

PHP fue creado en 1.994 por Rasmus Lerdorf y las primeras versiones no fueron distribuidas al público, fueron usadas en su página web para mantener el control sobre quien consultaba su currículum. La primera versión disponible para el público a principios de 1995 fue conocida como "Herramienta para páginas web personales" (Personal Home Page Tools). Consistía en un analizador sintáctico muy simple y una serie de utilidades comunes en las páginas web de la época. Desde entonces ha evolucionado mucho y actualmente sigue siendo desarrollado con nuevas funciones por *The PHP Group*, publicándolo bajo la PHP License, una licencia de software libre.

PHP puede usarse en todos los principales sistemas operativos como Windows, Mac OS, Linux y muchas variantes de Unix entre otros y admite la mayoría de servidores Web de hoy en día. De modo que con PHP se tiene la libertad de elegir el sistema operativo y el servidor Web. Es completamente expandible, está compuesto de un sistema principal, un conjunto de módulos y una variedad de extensiones de código para todo tipo de funcionalidades como para el manejo de sockets, generar documentos PDF, para generar dinámicamente páginas en Flash, etc. Sin olvidar una de sus características más importantes y destacables como es su soporte para un amplio abanico de bases de datos.

2.8.2. Java

Java es un lenguaje de programación desarrollado por Sun Microsystems, quien lo define como *"simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico"* (10). Java fue presentado en 1.995 y desde entonces se ha convertido en un lenguaje de programación muy popular. Java es un lenguaje muy valorado porque los programas realizados en Java se pueden ejecutar en diversas plataformas en distintos sistemas operativos como Windows, Mac OS, Linux o Solaris, por lo que es un lenguaje independiente de la plataforma. Para conseguir la portabilidad de los programas Java, se utiliza un entorno de ejecución para los programas compilados. Este entorno se denomina Java Runtime Environment (JRE). Es gratuito y está disponible para los principales sistemas operativos.

Los programas Java se compilan en un lenguaje intermedio, denominado Bytecode. Este código es interpretado por la máquina virtual de Java del entorno de ejecución (JRE) y así se consigue la portabilidad a distintas plataformas. El JRE es una pieza intermedia entre el código Bytecode y los distintos sistemas operativos existentes. Un programa Java compilado en Bytecode se puede ejecutar en sistemas operativos como Windows, Linux, Mac OS, Solaris, iOS o Android utilizando el entorno de ejecución de Java (JRE) apropiado.

La evolución del lenguaje de programación Java ha sido muy rápida. La plataforma de desarrollo de Java, denominada Java Development Kit (JDK) se ha ido ampliando y cada vez es mayor el número de programadores que trabajan con ella. Java no sólo es un lenguaje de programación, es también una plataforma de desarrollo, un entorno de ejecución y un conjunto de librerías para el desarrollo de programas. Las librerías para el desarrollo se denominan Java Application Programmming Interface (Java API).

Al programar en Java no se parte de cero. Cualquier aplicación que se desarrolle se apoya, por decirlo de algún modo, en un gran número de "*clases*" preexistentes que forman parte del propio lenguaje y otras que pueden ser comerciales o creadas por el propio usuario.

3. Arquitectura del sistema

En este capítulo se describe el sistema con el que se da solución al problema planteado en este proyecto. También será descrito en detalle cada uno de los componentes de la arquitectura.

La solución propuesta consiste en un modelo de aplicación distribuida cliente-servidor, donde las tareas se reparten entre estos. La información se encuentra centralizada en un servidor, garantizando que esté actualizada, pueda ser accesible por varios clientes a la vez y facilitando la administración. Es más eficiente que los clientes soliciten la información a un servidor y que éste consulte los datos y construya una respuesta adecuada, además de seguro porque los usuarios no dispondrán de información necesaria para pasos intermedios. Esta arquitectura también proporciona la posibilidad de crear una aplicación cliente más liviana, hecho importante a la hora de utilizar dispositivos móviles por sus recursos limitados. Y todo esto sin mantener una conexión persistente entre cliente y servidor, es decir, el cliente sólo cuando necesita información se conecta al servidor y en cuanto la descarga libera la conexión.

La arquitectura básica del sistema, como podemos ver en la Ilustración 9, está compuesta por cuatro elementos o bloques principales:

- **Aplicación cliente**: aplicación para dispositivos Android, programada en Java, con la que el cliente interactúa desde su Smartphone con ayuda de su cámara incorporada, para identificar los equipos de un laboratorio mediante tecnología de reconocimiento de imagen.
- **Servidor web y aplicación servidor**: compuesto por un servidor web Apache que recibe las peticiones de los clientes, provee a la aplicación cliente de las fotografías de los equipos, datos, manuales de usuario, etcétera y da acceso a la aplicación servidor que proporciona, procesa y envía la información requerida almacenada en una base de datos a los clientes.
- **Base de datos**: Base de datos MySQL que contiene toda la información relativa a los equipos de los laboratorios que pudieran necesitar los usuarios.
- **Aplicación de mantenimiento**: Costa de una pequeña aplicación que permite al administrador actualizar la información de la BBDD de forma cómoda y rápida (solución no implementada. Su labor es realizada mediante phpMyAdmin).

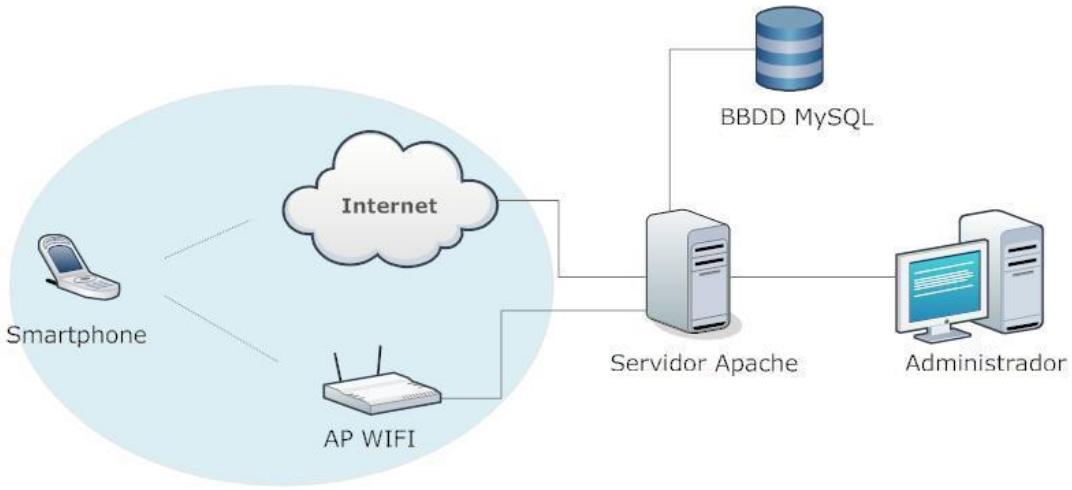


Ilustración 9 – Arquitectura general del sistema diseñado

3.1. Arquitectura de la aplicación cliente

La aplicación cliente elaborada y destinada a los usuarios de laboratorios de la UPCT, ha sido creada para su utilización en smartphones con sistema operativo Android y está programada en Java. Interactúa con el usuario mediante una interfaz gráfica, se encarga de realizar las peticiones al servidor y mostrar la información solicitada de un modo claro e intuitivo. Debido a los recursos limitados de que disponen los dispositivos móviles donde va a ser ejecutada se ha procurado que para su instalación se requiera el menor gasto de memoria posible, por lo que esta aplicación no almacena ningún tipo de información o imágenes para el reconocimiento, toda esta información y datos serán descargados del servidor en el momento en el que el usuario lo necesite.

Para que la descarga de la información que va a necesitar el usuario en un momento determinado sea sólo la referente al laboratorio en el que se encuentra, se hace uso de códigos QR que identifican los laboratorios. Estos códigos QR contienen la URL a la aplicación servidor, alojada en un servidor web Apache que está a la escucha, junto con el identificador del laboratorio, enviando este parámetro mediante el método *"GET"* de *HTTP*. Por tanto, cuando un usuario requiere de información de algún objeto del laboratorio donde se encuentra, ha de enfocar una vez abierta la aplicación *InfoLab*, con la cámara de su smartphone el código QR del laboratorio, el cual contiene toda la información necesaria para iniciar la descarga de los datos del servidor. Una vez descargadas y procesadas las imágenes de los objetos del laboratorio y de la información relativa a estos, desde el servidor y base de datos, el usuario ya puede pasar a identificar los equipos que le rodean.

Según el diseño elegido la Ilustración 10 muestra resumidamente los pasos que sigue la aplicación durante su funcionamiento.

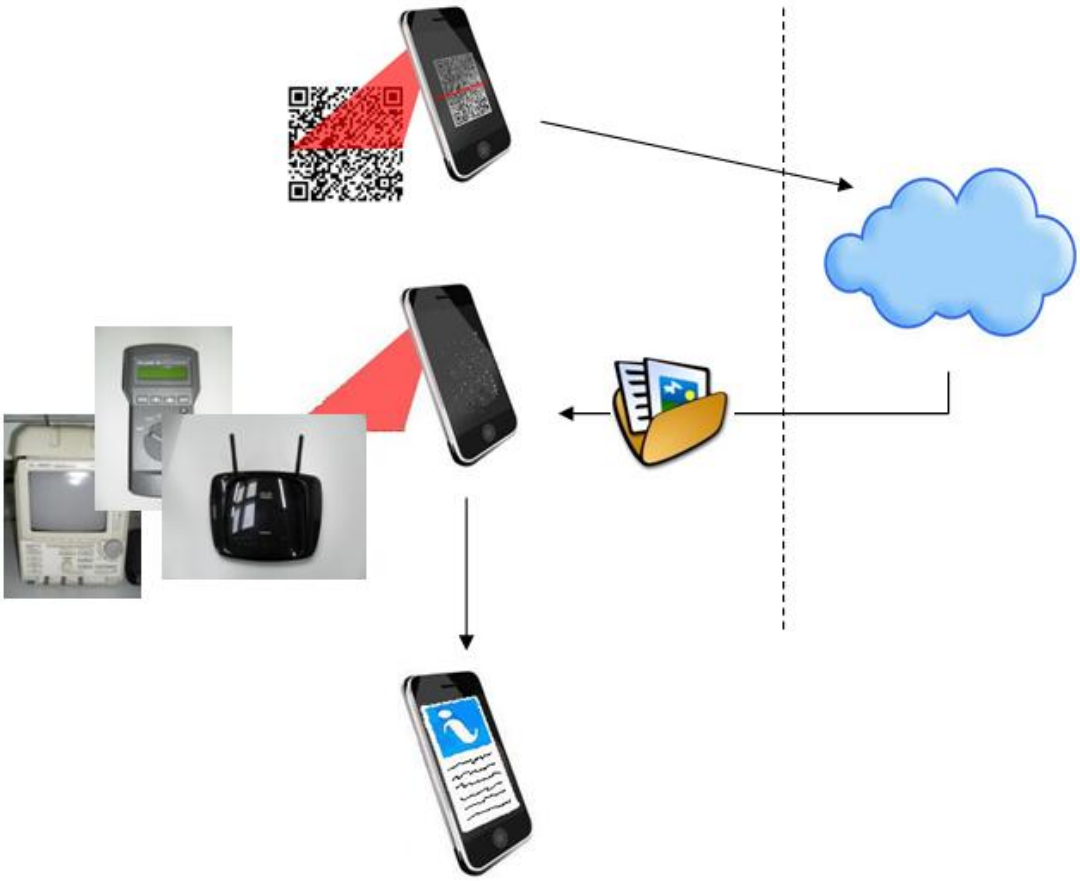


Ilustración 10 - Ciclo de funcionamiento de la aplicación cliente

3.2. Arquitectura del lado servidor

La solución dada en el problema descrito pasa por la utilización de una arquitectura cliente-servidor, que es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servidores y los clientes. De este modo se consigue que el servidor sea quien contenga la información que utilizarán los clientes y que el administrador pueda realizar cambios en los datos de un modo más sencillo y sin necesidad de que cada uno de los clientes tenga que actualizar su software. Consiguiendo así grandes ventajas de tipo organizativo con la centralización de la gestión

de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema. También se logra una importante cualidad y necesidad requerida para este proyecto y es que los usuarios no almacenen en sus smartphones un exceso de información que puede que nunca vayan a utilizar, ya que muchos de estos dispositivos tienen recursos muy limitados.

El sistema servidor está compuesto por tres elementos principales, el servidor Web como tal, la aplicación servidor (*infolab.php*) que se haya alojada en él y una base de datos MySQL que almacena toda la información referente a los equipos de los laboratorios.

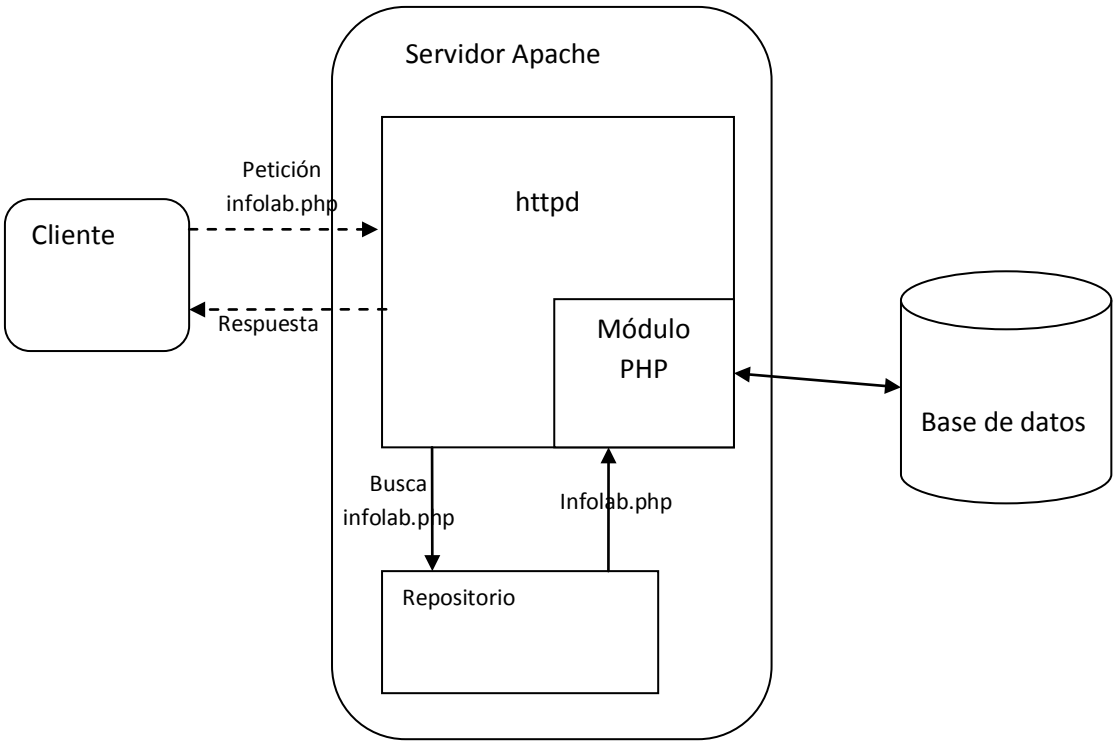


Ilustración 11 – Arquitectura del sistema servidor

El servidor Web es un servidor Web Apache extendido con la inclusión de su módulo para PHP. Se puede encontrar instalado en una máquina sin ninguna característica especial y su función es la de mantenerse a la escucha para atender las peticiones *HTTP* que los clientes realizan desde sus smartphones. Estas peticiones que realizan los clientes pueden consistir en el acceso a las imágenes de los equipos de laboratorio, a documentos en *pdf* como manuales de usuario o a la aplicación servidor *infolab.php*, haciendo uso del módulo PHP de Apache.

Otro elemento principal es una aplicación servidor escrita en *PHP* alojada en el repositorio del servidor Web, llamada *infolab.php*. Esta aplicación se ejecuta en el servidor y su misión es la de procesar las peticiones realizadas por los clientes. Estas peticiones se realizan automáticamente en cuanto el usuario captura un código QR con la aplicación cliente *InfoLab*. Cuando el servidor recibe una de estas peticiones ejecuta la aplicación servidor, la cual obtiene de la *URL* el identificador del laboratorio del que el cliente quiere obtener información, para lo cual se hace uso del método *"GET"* de *HTTP*.

Una vez extraído el identificador, se construye y realiza con él la consulta adecuada para obtener los datos solicitados de la base de datos *MySQL*. Tras obtenerlos, se les da formato mediante *JSON*[4] y son enviados de vuelta al cliente.

La ubicación de la información está dividida entre la base de datos y el servidor Web. En el repositorio del servidor Web se localizan las imágenes previamente tomadas de los equipos de los laboratorios y los documentos *pdf* como pueden ser los manuales de usuario. En base de datos está localizada toda la información descriptiva referente a los equipos de los laboratorios como nombre, modelo, fabricante, descripción, normas de seguridad, etcétera. Y además la localización en el servidor Web de la imagen y documentos *pdf* referentes a dicho equipo.

Con la elección de este diseño se loga la generación de una respuesta dinámica, devolviendo respuestas diferentes a los usuarios y específicas en función del laboratorio en el que se encuentren.

3.2.1. Requisitos del sistema servidor

En este apartado se describe las características de hardware y software que ha de cumplir la máquina o máquinas que compongan el sistema servidor, así como las versiones utilizadas en este desarrollo.

Las funciones del sistema servidor pueden ser realizadas por una única máquina que contenga tanto el servidor Web como la base de datos o estar repartidos el servidor Web en una máquina y la base de datos en otra.

Para la puesta en marcha del sistema diseñado en este proyecto, la máquina o máquinas que realice el papel de servidor deberá contar con el siguiente software:

- Servidor Web Apache y su módulo *mod_php* para páginas dinámicas en PHP.
- Contener el servidor Web en su repositorio la aplicación *infolab.php*.
- Una base de datos MySQL.
- El gestor de bases de datos phpMyAdmin.

[4] JSON, acrónimo de *JavaScript Object Notation*, es un formato ligero para el intercambio de datos. Forma parte del núcleo de PHP.

Los requisitos de hardware mínimos para el correcto funcionamiento de este sistema servidor no son nada exigentes y cualquier máquina de hoy día es capaz de soportarlo, siempre que disponga de suficiente memoria en disco para almacenar las imágenes y documentos a los que queramos que la aplicación cliente tenga acceso.

En el desarrollo del este proyecto se ha utilizado el software XAMPP para Windows en su versión 1.8.2 que incluye el servidor Web Apache 2.4, PHP versión 5.4.22, MySQL versión 5.5.34 y la versión 4.0.9 de phpMyAdmin.

4. Desarrollo del proyecto

A lo largo de esta capítulo describiremos el sistema realizado en este proyecto, tanto a nivel funcional general como entrando en detalles técnicos, cuando se crea necesario o de interés y las herramientas con las que se ha realizado.

4.1. Entorno de desarrollo y herramientas utilizadas

La elección de un tipo de herramienta u otra puede suponer una gran diferencia en el producto finalmente obtenido y el tiempo necesario para su realización. En este apartado se van a describir las herramientas de desarrollo utilizadas para la realización de este proyecto.

4.1.1. Eclipse IDE

Eclipse (11) es el entorno de desarrollo más utilizado para programar en Java. Está compuesto por un conjunto de herramientas de programación de código abierto multiplataforma. Es un desarrollo originalmente de la compañía IBM cuyo código fuente fue puesto a disposición de los usuarios. Actualmente es desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. La comunidad de Eclipse tiene más de 200 proyectos que cubren distintos aspectos del desarrollo software.

En sí mismo Eclipse es un marco y un conjunto de servicios para construir un entorno de desarrollo integrado (IDE) a partir de componentes o módulos conectados (*plug-in*) lo que permite a Eclipse extenderse ajustándose a las necesidades del desarrollador. Hay *plug-ins* para el desarrollo de Java (JDT Java Development Tools) así como para el desarrollo en C/C++, Cobol, etc.

Para este caso en particular se ha hecho uso del módulo para el desarrollo de aplicaciones Android (ADT). El *plug-in* ADT (Android Development Tools) está diseñado para brindar al programador un entorno potente e integrado, ampliando las posibilidades de Eclipse. Entre otras cosas ADT proporciona una forma fácil y rápida de crear un proyecto para desarrollar una aplicación para Android, crear una interfaz gráfica de usuario, creación de permisos, simulaciones, etc. Además nos permite depurar las aplicaciones usando las herramientas del SDK Android.

4.1.2. XAMPP

XAMPP es una herramienta independiente de plataforma, de software libre, que consiste principalmente en la base de datos MySQL, el servidor Web Apache y los intérpretes para lenguajes PHP y Perl. También incluye módulos como OpenSSL y phpMyAdmin. El nombre proviene del acrónimo **X** (para cualquiera de los sistemas operativos), **A**pache, **M**ySQL, **P**HP, **P**erl.

Este software está liberado bajo licencia GNU y actúa como un servidor Web libre, fácil de instalar y configurar. Dispone también de una amplia gama de complementos y herramientas. Para el trabajo llevado a cabo en este proyecto ha sido necesaria la instalación y configuración del servidor Web Apache, su módulo PHP, la base de datos MySQL y el sistema de administración de bases de datos (DBMS) phpMyAdmin.

4.1.3. Notepad++

Notepad++ (12) es un editor de texto potente de código fuente libre, con soporte para varios lenguajes de programación. De apariencia similar al Bloc de notas, incluye opciones más avanzadas que pueden ser útiles para usuarios avanzados como desarrolladores y programadores. Se distribuye bajo licencia GNU.

Está basado en el componente de edición Scintilla y está escrito en C++ utilizando directamente la API de Windows, lo que asegura una velocidad de ejecución y un tamaño más reducido del programa final.

Este software es muy útil para escribir código, ya sea HTML, JavaScript, C++, XML o cualquiera que le agreguemos. Si cargamos un archivo con la extensión correspondiente, por ejemplo .html, detectará automáticamente el lenguaje y nos pintará de colores las palabras especiales y símbolos claves.

En nuestro caso se ha hecho uso de Notepad++ como editor para PHP por su sencillez y gratuidad.

4.2. Desarrollo de la aplicación cliente InfoLab

Como ya se ha descrito en capítulos anteriores la aplicación cliente es una aplicación para Smartphones con sistema operativo Android y escrita en Java. Es la que ha ocupado el grueso de este proyecto y cuyo objetivo es interactuar con los usuarios de los laboratorios, para dar información de los equipos que éstos contienen.

Para lograr dicho objetivo, se ha hecho uso de tecnologías de reconocimiento y procesado de imagen, mediante las cuales se consigue identificar los equipos de los laboratorios y dar información sobre ellos.

Consta gráfica y estructuralmente de dos ventanas o vistas, cada una en un *activity* diferente, las cuales se van llamando mutuamente según las acciones llevadas a cabo por el usuario. Una inicial (Ilustración 12) en la que se muestra continuamente lo capturado por la cámara y en la que se nos indica según el momento si debemos enfocar códigos QR u objetos del laboratorio. Y una segunda ventana (Ilustración 13) en la que se muestra toda la información relacionada con el objeto una vez ha sido reconocido por la aplicación.

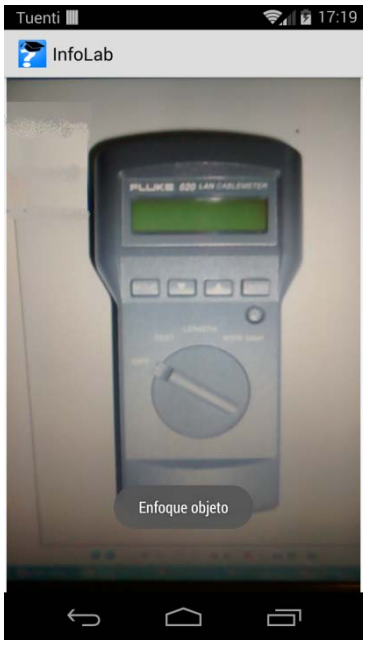


Ilustración 12 – Pantalla que muestra lo capturado por la cámara del dispositivo móvil

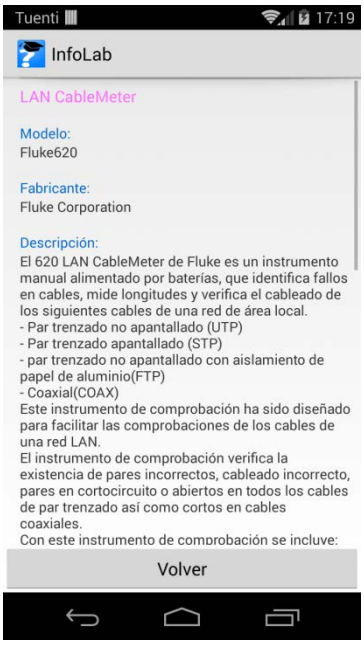


Ilustración 13 – Pantalla que muestra la información de un objeto reconocido

4.2.1. Comportamiento de la aplicación

En este punto entraremos en detalles del comportamiento de la aplicación cliente InfoLab y en cómo el usuario interactúa con ella. Para comenzar a describir su funcionalidad, en la Ilustración 14 se muestra gráficamente las fases por las que pasa y que son descritas a continuación.

Esta aplicación será el punto de entrada a todo el sistema y como podemos ver en el diagrama de flujo de la Ilustración 14, una vez iniciada la aplicación nos pide que introduzcamos el código QR del laboratorio en el que nos encontramos y enfocarlo con la cámara de nuestro teléfono. En cuanto se identifica un código, se inicia automáticamente la conexión con el servidor para la descarga de los datos y las imágenes de los equipos que van a poder ser reconocidos por la aplicación. Si todo se realiza sin problemas veremos un mensaje en pantalla del número de objetos de los que se ha descargado información y que desde eso momento podemos identificar enfocando con la cámara del teléfono móvil. Cuando la aplicación identifica un equipo lo comunica con un mensaje en pantalla junto a dos botones, uno de cancelar que hace que la aplicación vuelva a la fase de identificación de objetos y otro con el que podemos ver la información del objeto en detalle. La nueva ventana que se nos muestra contiene toda la información que el administrador del sistema ha introducido sobre el equipo, entre los que puede contener, aparte de su descripción, modelo y normas de seguridad, enlaces para visitar la web del fabricante o la descarga de manuales de usuario en formato *pdf*.

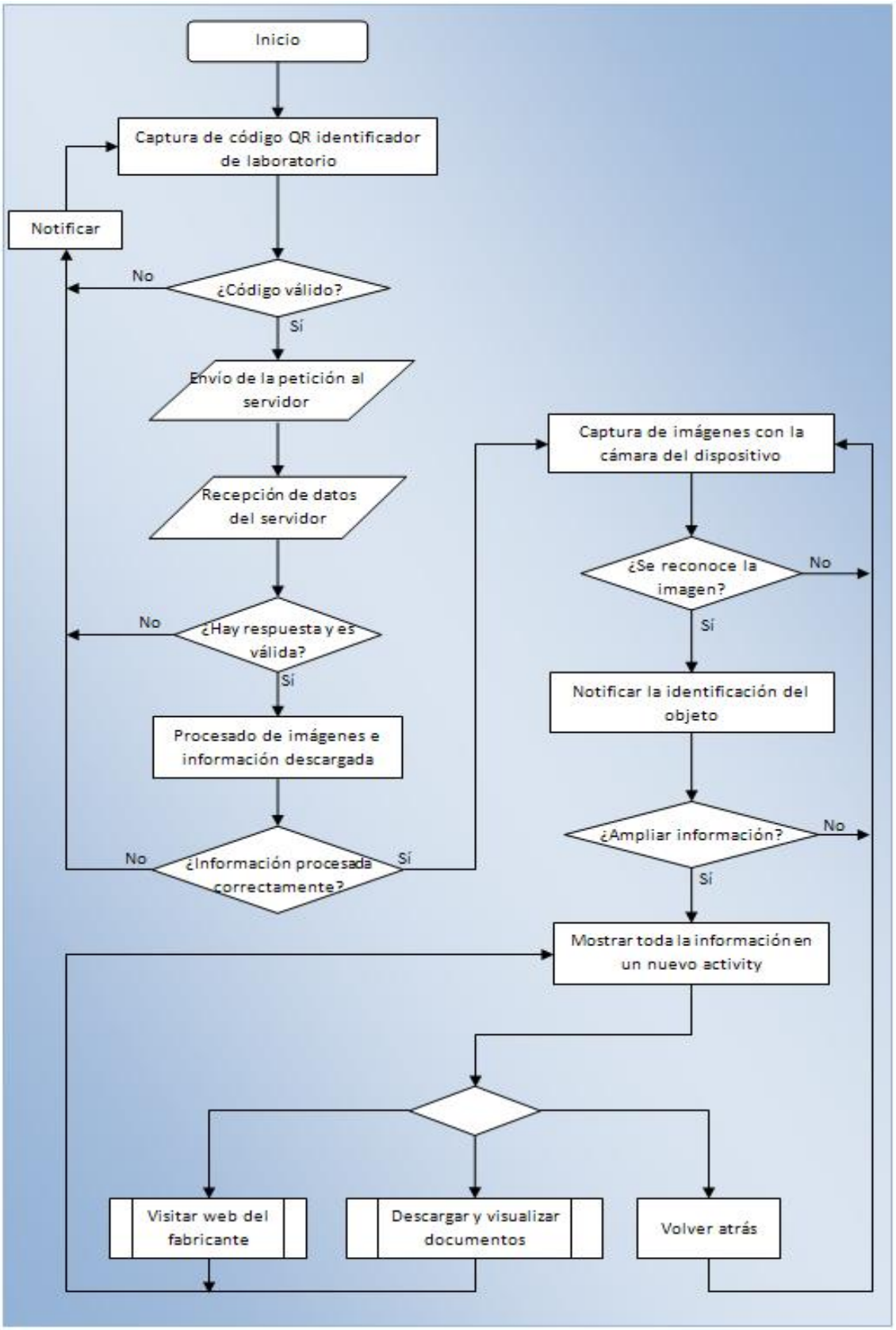


Ilustración 14 – Diagrama de flujo de la aplicación cliente InfoLab

4.2.2. Organización de clases y recursos

El diagrama de clases de la Ilustración 15 muestra la estructura y organización de las clases que se han implementado como solución a la aplicación cliente. En cuanto a la organización de paquetes en el proyecto Java, ésta ha estado condicionada a la utilización de la librería *ImageMatching* para el reconocimiento de imagen. El fabricante, para su utilización gratuita en desarrollos impone ciertas restricciones, como que su librería solo funcionará en una aplicación con un nombre de paquete establecido por ellos. Razón por la que se ha empleado el extraño nombre de paquete "*com.gmail.at.pedrojuan20.immandbeta120803*".

A continuación se hablará cada una de las clases, describiendo sus funciones más importantes.

InfoLabMainActivity.java

Se encarga de manejar la vista principal y es el punto de inicio de la aplicación, donde se ejecuta el hilo principal. La vista es muy simple y consiste en un *FrameLayout* donde se muestra en todo momento lo capturado por la cámara del dispositivo móvil. En esta clase tiene lugar la inicialización de variables y donde se crea una instancia de la clase *ARmatcher*. El objeto de la clase *ARmatcher*, perteneciente a la API *ImageMatching* es el encargado de la realización del procesado y reconocimiento de imagen. Una vez instanciado y configurados sus atributos y propiedades como tamaño de pantalla, orientación, instancia de la cámara, filtros etc, es capaz de reconocer códigos QR y más adelante imágenes.

Esta clase maneja un *HashMap* llamado "infoObjetosAReconocer" con la siguiente estructura `HashMap<Integer, ObjetoInfo>` el cual contiene la información de todos los equipos obtenidos del servidor tras la lectura de un código QR y su procesado en forma de objetos *ObjetoInfo* (clase creada para tal propósito) y como clave, el entero que devuelve el objeto de la clase *ARmatcher* cuando añade la imagen al *pool* de imágenes que va a ser capaz de reconocer.

InfoLabMainActivity implementa las interfaces *ARmatcherImageCallBack* y *ARmatcherQRCallBack* de la API *ImageMatching*, por lo que se ha de sobre escribir sus métodos, de los que ha sido necesario implementar *onSingleQRrecongnitionResult()* y *onImageRecognitionResult()*. Métodos a los que llama *ARmacher* cuando identifica, o bien un código QR o una imagen respectivamente.

Por último mencionar que los métodos *onStart(), onResume(), onPause(), onStop() y onDestroy(),* consecuencia de heredar de *Activity* son usados, entre otras cosas, para ir deteniendo o iniciando el proceso de chequeo de lo capturado por la cámara en busca de imágenes reconocibles o códigos QR.

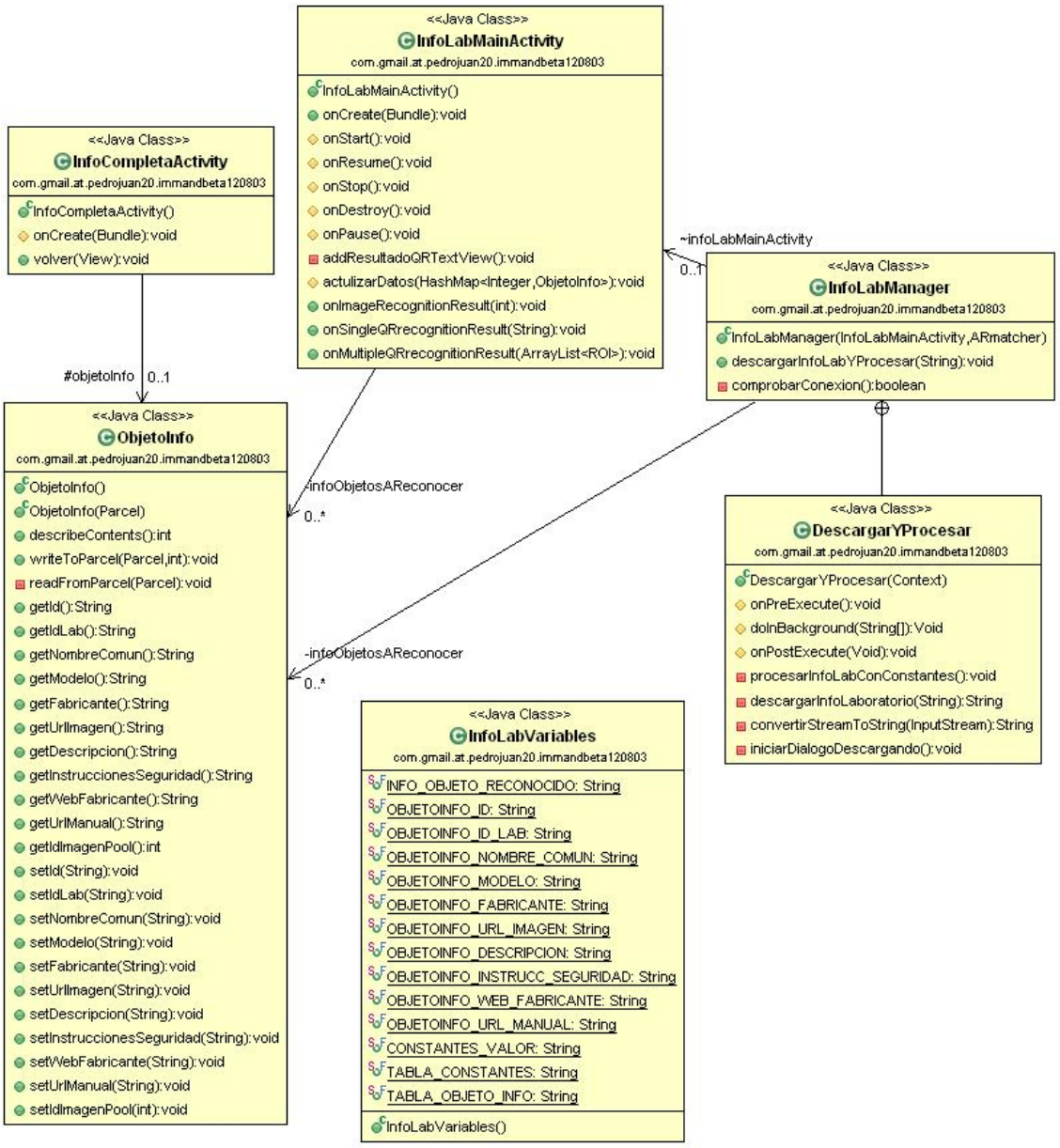


Ilustración 15 – Diagrama de clases de InfoLab

InfoLabManager.java

En la clase InfoLabManager es donde se encuentran el grueso de la lógica de negocio y gestión. Posee toda una variedad de métodos, de los que entre otras cosas se encargan de comprobar la conexión a la red, crear la conexión con el servidor, descargar tanto información como imágenes, procesarlas y añadirlas al *pool* de imágenes de las que la aplicación va a ser capaz de identificar.

Muchas de estas tareas interesa realizarlas en segundo plano, por lo que tienen lugar en la siguiente subclase interna.

DescargarYProcesar.java

DescargarYProcesar es una subclase o clase interna privada dentro de InfoLabManager y que hereda de *AsyncTask.* La razón de ser de esta clase es múltiple. Por un lado es establecer la conexión web con el servidor y por otro descargar y procesar los datos e imágenes.

AsyncTask (tarea asíncrona) permite el uso apropiado y fácil de *threads* (hilos), permitiendo realizar operaciones en segundo plano y publicar los resultados en el hilo de la interfaz de usuario (UI) sin necesidad de manipular hilos. Con ella se crean tareas asíncronas y resuelve el problema de que la UI de Android no permite llamadas desde otros hilos que no sea el suyo. Para utilizar *AsyncTask* debemos:

- Crear una subclase que herede de *AsyncTask*, comúnmente como una clase interna privada dentro de la actividad en la que estamos trabajando.
- Sobre escribir uno o más métodos de *AsyncTask* para poder realizar el trabajo en segundo plano, además de cualquier otra tarea asociada para poder mostrar alguna actualización en el hilo de la UI.
- Cuando sea necesario, crear una instancia de *AsyncTask* y llamar al método *execute()* para que empiece a realizar su trabajo.

La clase AsyncTask dispone de tres tipos genéricos de parámetros distintos, que tenemos que especificar cuando declaramos nuestra clase, como vemos en el ejemplo de la Ilustración 16. Los tres parámetros son los siguientes:

- Params: El tipo de parámetros que pasaremos al comenzar la tarea.
- Progress: El tipo parámetros que necesitaremos para actualizar la UI.
- Result: El tipo de dato que devolveremos una vez terminada la tarea.

```
1 public class MiTarea extends AsyncTask<Params, Progress, Result> {
2 }
```

Ilustración 16 – Ejemplo de declaración de una clase que hereda de AsyncTask

Cada vez que una tarea asíncrona se ejecuta, se pasa por 4 etapas:

- ***onPreExecute().*** Se invoca en el hilo de la UI inmediatamente después de que la tarea es ejecutada. Este paso se utiliza normalmente para configurar la tarea. En la solución dada se ha usado para inicializar y mostrar un *ProgressDialog* en la interfaz de usuario, como se puede ver en la Ilustración 17.

- ***doInBackground().*** Se invoca en el subproceso en segundo plano inmediatamente después de que *onPreExecute()* termina de ejecutarse. Los parámetros de la tarea asíncrona se pasan en esta etapa. Dentro de esta etapa se puede hacer uso del método *publishProgress(Progress…)* para publicar el progreso en el hilo de la UI con la ayuda de la siguiente etapa que es *onProgressUpdate(Progress…).* En nuestro caso es donde se realiza la descarga de datos del servidor y el procesado de las imágenes e información.
- ***onProgressUpdate(Progress…)***. Se invoca en el hilo de la UI después de una llamada a *publishProgress(Progress…).* El momento de la ejecución es indefinido. Este método es utilizado para mostrar cualquier tipo de progreso en la UI mientras la tarea en segundo plano sigue ejecutándose.
- ***onPostExecute(Result…)***. Se invoca en el *UI thread* después de que el proceso en segundo plano ha sido terminado. El resultado devuelto por todo el proceso se pasa a este método como parámetro. En este punto se le pasa al hilo principal los datos procesados y se elimina el *ProgressDialog* creado en la etapa *onPreExecute().*

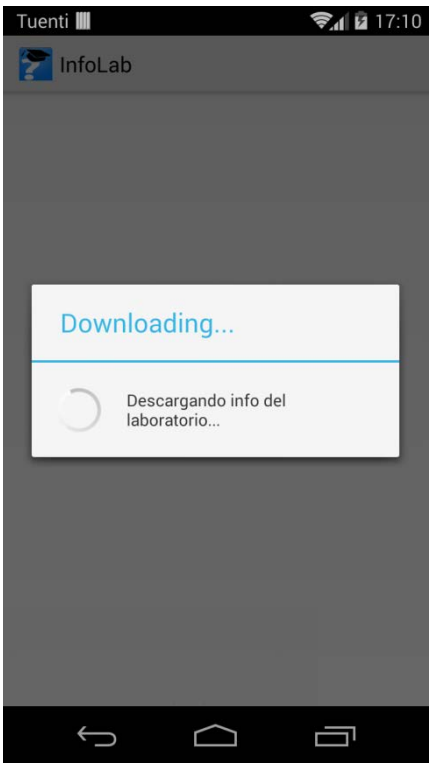


Ilustración 17 – Aspecto del *ProgressDialog* en la aplicación

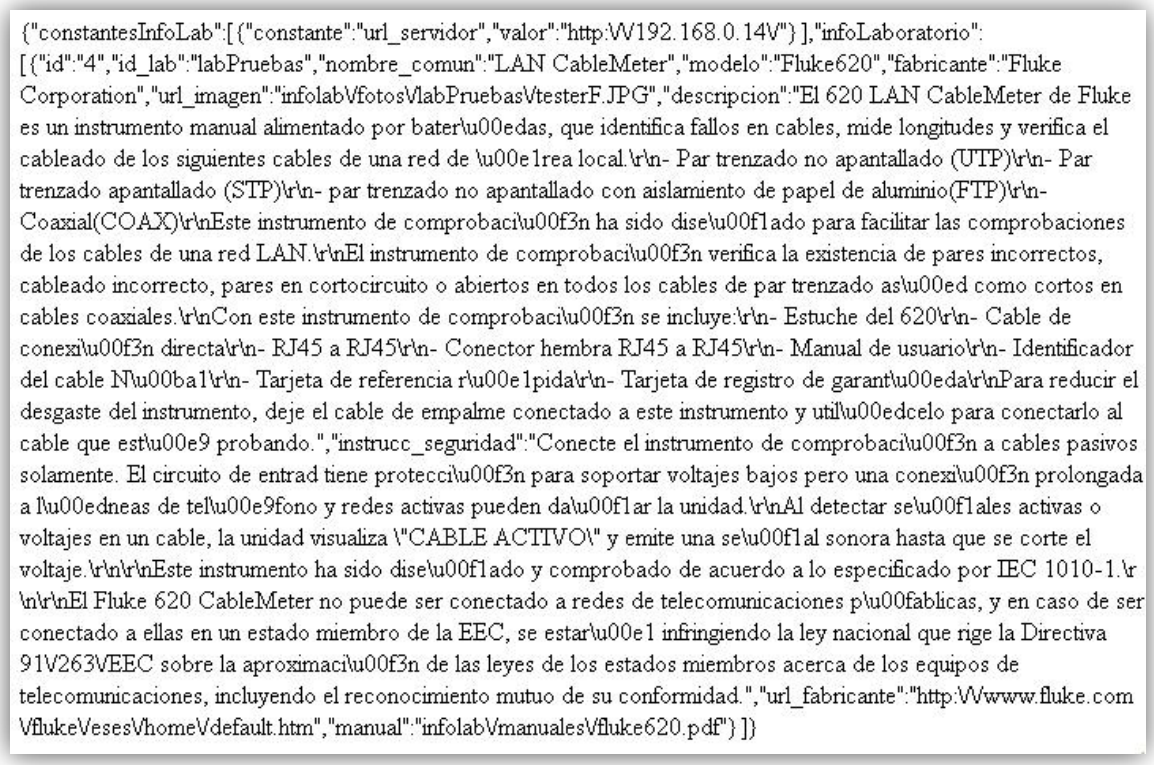
Como se ha dicho el uso de la subclase DescargarYProcesar ha sido necesario por dos razones principales. Por un lado ha sido por la obligación existente desde la versión 3.0 de Android (API 11) de que no se permite abrir una conexión de red desde el hilo principal (el *main* de la aplicación). Y la otra ha sido por la razón de que la tarea de descargar y procesar las imágenes puede tardar un tiempo indeterminado, dependiendo

de la velocidad de la conexión y la cantidad de imágenes devueltas por el servidor. Por lo que se ha tenido la obligación de realizar estas operaciones en segundo plano.

Por otro lado, la API de Android proporciona varias formas de implementar una comunicación cliente-servidor. Inicialmente esto se programó haciendo uso de *HttpClient* y *DefaultHttpClient,* pero pronto se vio que aunque en las versiones más antiguas de Android funcionaba bien en las más recientes se producían problemas, por lo que al final se ha decidido hacer uso de la clase ***HttpURLConnection*** como recomiendan desde *www.developer.android.com* (13). Ambos clientes HTTP tienen soporte para HTTPS, transferencia de archivos *streaming*, tiempos de espera configurables, IPv6, etc. Pero *HttpURLConnection* es más ligero, reduce el uso de la red, mejora la velocidad, ahorra batería y su uso está recomendado para las versiones más recientes de Android. Por tanto se ha creído adecuado su uso para la realización de esta aplicación.

En cuanto a lo referente al procesado de los datos que la aplicación recibe del servidor, se ha hecho uso de **JSON**. JSON es un acrónimo de *JavaScript Object Notation,* es un formato para el intercambio de datos y tiene un formato más sencillo que XML. Por tanto en la aplicación servidor, los datos obtenidos de la base de datos son formateados con JSON antes de ser enviados al cliente. La aplicación cliente utilizando las clases *JSONObject* y *JSONArray* simplifica el proceso de recuperación de datos contenidos en el mismo, proporcionando la colección de datos y los datos uno a uno respectivamente. Un *JSONArray* es una secuencia ordenada de valores. Su forma de texto externo es una cadena envuelta en corchetes por comas que separan los valores. La forma interna es un objeto que tiene métodos para acceder a los valores de índice y métodos para agregar o reemplazar valores. Los valores que contiene pueden ser cualquier mezcla de Strings, Booleans, Integers, Longs, Doubles, JSONObjects, otros JSONArrays o nulls. Al crear una instancia de *JSONArray* a partir del JSON recibido del servidor, obtendremos un array de *JSONObjects* con todos los elementos anidados en el JSON. Una vez obtenidos estos, deberemos ir recorriendo el array obteniendo cada *JSONObject* y pudiendo acceder a sus atributos, procesándolos a nuestro antojo. Algunos de estos atributos podrían ser a su vez otros *JSONArray* y deberíamos repetir el proceso para analizar todos los datos, aunque este caso no se da en la solución implementada.

En la Ilustración 18 podemos ver un ejemplo de los datos de un equipo, que la aplicación cliente recibe del servidor con formato JSON, donde son estructurados mediante llaves, corchetes, comilla y comas.



{"constantesInfoLab":[{"constante":"url_servidor","valor":"http:\/\/192.168.0.14\/"}],"infoLaboratorio":[{"id":"4","id_lab":"labPruebas","nombre_comun":"LAN CableMeter","modelo":"Fluke620","fabricante":"Fluke Corporation","url_imagen":"infolab\/fotos\/labPruebas\/testerF.JPG","descripcion":"El 620 LAN CableMeter de Fluke es un instrumento manual alimentado por bater\u00edas, que identifica fallos en cables, mide longitudes y verifica el cableado de los siguientes cables de una red de \u00e1rea local.\r\n- Par trenzado no apantallado (UTP)\r\n- Par trenzado apantallado (STP)\r\n- par trenzado no apantallado con aislamiento de papel de aluminio(FTP)\r\n- Coaxial(COAX)\r\nEste instrumento de comprobaci\u00f3n ha sido dise\u00f1ado para facilitar las comprobaciones de los cables de una red LAN.\r\nEl instrumento de comprobaci\u00f3n verifica la existencia de pares incorrectos, cableado incorrecto, pares en cortocircuito o abiertos en todos los cables de par trenzado as\u00ed como cortos en cables coaxiales.\r\nCon este instrumento de comprobaci\u00f3n se incluye:\r\n- Estuche del 620\r\n- Cable de conexi\u00f3n directa\r\n- RJ45 a RJ45\r\n- Conector hembra RJ45 a RJ45\r\n- Manual de usuario\r\n- Identificador del cable N\u00famero\r\n- Tarjeta de referencia r\u00e1pida\r\n- Tarjeta de registro de garant\u00eda\r\nPara reducir el desgaste del instrumento, deje el cable de empalme conectado a este instrumento y util\u00edcelo para conectarlo al cable que est\u00e9 probando.","instrucc_seguridad":"Conecte el instrumento de comprobaci\u00f3n a cables pasivos solamente. El circuito de entrad tiene protecci\u00f3n para soportar voltajes bajos pero una conexi\u00f3n prolongada a l\u00edneas de tel\u00e9fono y redes activas pueden da\u00f1ar la unidad.\r\nAl detectar se\u00f1ales activas o voltajes en un cable, la unidad visualiza \"CABLE ACTIVO\" y emite una se\u00f1al sonora hasta que se corte el voltaje.\r\n\r\nEste instrumento ha sido dise\u00f1ado y comprobado de acuerdo a lo especificado por IEC 1010-1.\r\n\r\nEl Fluke 620 CableMeter no puede ser conectado a redes de telecomunicaciones p\u00fablicas, y en caso de ser conectado a ellas en un estado miembro de la EEC, se estar\u00e1 infringiendo la ley nacional que rige la Directiva 91\/263\/EEC sobre la aproximaci\u00f3n de las leyes de los estados miembros acerca de los equipos de telecomunicaciones, incluyendo el reconocimiento mutuo de su conformidad.","url_fabricante":"http:\/\/www.fluke.com\/fluke\/eses\/home\/default.htm","manual":"infolab\/manuales\/fluke620.pdf"}]}

Ilustración 18 – Fragmento de JSON recibido por la aplicación cliente

ObjetoInfo.java

Esta clase modela un objeto o equipo de los laboratorios, con todos los atributos del mismo que se han considerado de utilidad para los usuarios. Está compuesta casi exclusivamente por métodos "*gets*" y "*sets*" que serán usados por otras clases de la aplicación tanto para su consulta como para su creación o modificación de objetos.

Uno de los problemas encontrados en este punto es que la aplicación desarrollada intercambia objetos de esta clase entre distintas *activitys* y pronto se vio que en Android no se pueden pasar simplemente objetos entre *activitys*, sólo tipos primitivos y *arrays* de los mismos. Para ello la API de Android proporciona dos soluciones, o bien hacer nuestro objeto *Serializable* o *Parcelable*. Hacer que nuestro objeto implemente la interfaz *Serializable* no es mala solución para objetos pequeños pero en la práctica es muy lenta y si vamos a serializar un objeto complejo y grande no es adecuado. Hacer que implemente la interfaz *Parcelable* es altamente recomendado en Android por su rendimiento superior. En la Ilustración 19 vemos un gráfico de un test de velocidad consistente en un bucle que simula el paso de un objeto *Serializable* entre *activitys* frente otro *Parcelable* en distintos dispositivos móviles.

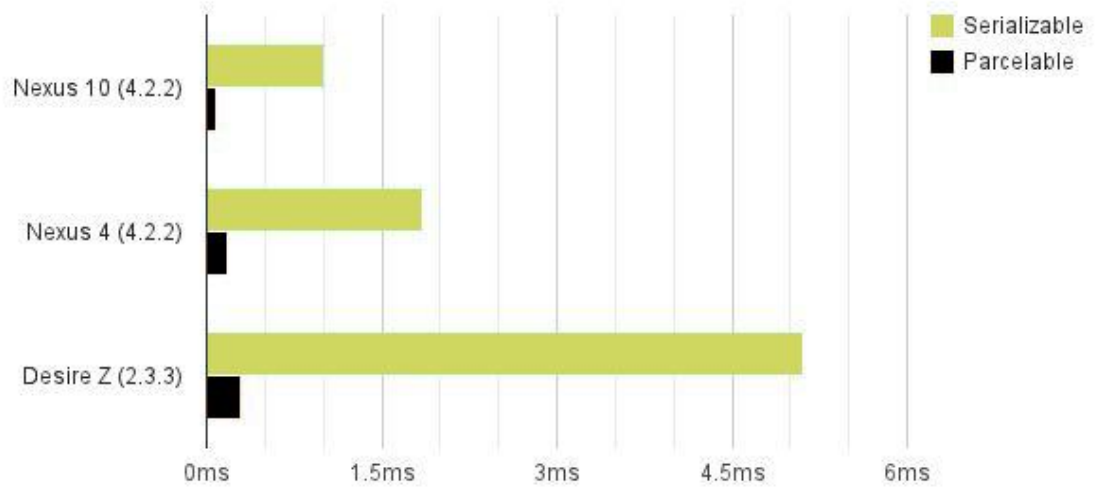


Ilustración 19 – Test de velocidad de un objeto *Parcelable* vs *Serializable*

Para hacer una clase *Parcelable*, hay que:

- Implementar el método **writeToParcel** para descomponerla en un **Parcel**.
- Declarar el atributo *public static Creator* que implementa **Parcelable.Creator**, una especie de factoría que permitirá "reconstruir" el objeto original. Tendremos también como consecuencia que hacer un constructor que reciba los atributos de nuestra clase contenidos en un objeto de tipo Parcel.
- Implementar **describeContents** y que devuelva cero. Este método se usa para añadir información adicional al Parcelable en casos muy concretos.

A esto se ha añadido el método **readFromParcel** para recuperar nuestro objeto de un Parcel y donde debido a que cuando leemos un Parcel no hay pares clave-valor, se ha de leer en el mismo orden en que se escribió en writeToParcel.

InfoCompletaActivity.java

Se encarga de manejar la vista que muestra la información completa de un equipo cuando ha sido identificado por la aplicación. Toda esta información se muestra en una nueva ventana por lo que esta clase hereda de *Activity*. Su función es la de mostrar de un modo claro toda la información del equipo reconocido, separada en distintos campos y con la posibilidad de enlazar con la página web del fabricante o con el manual de usuario en

formato *pdf*. Toda esta información la obtiene de un objeto de la clase ObjetoInfo que recibe desde la *activity* principal.

InfoLabVariables.java

Esta clase se ha creado para albergar las constantes utilizadas en la aplicación con el objetivo de centralizarlas, para si se da el caso de que se tenga que hacer alguna modificación su localización sea rápida. Entre estas constantes mayormente se encuentran por ejemplo, los nombres de ciertas tablas y columnas de la base de datos, que son necesarios para desde la aplicación cliente identificar y extraer cada uno de los datos que recibe del servidor en forma de JSON. De este modo, si en algún momento se realizan modificaciones en la estructura de la base de datos no habrá que recorrer todo el código en busca de donde es utilizado, sino que bastará con modificar las constantes de esta clase.

4.2.3. Configuración y permisos (AndroidManifest)

AndroidManifest es un documento escrito en XML donde se declaran todas las especificaciones de la aplicación creada, como las Activities que la componen, los Intents, bibliotecas, nombre de la aplicación, el hardware que se necesita, los permisos de la aplicación etc. Para que Android pueda ejecutar un componente cualquiera de una aplicación, esta ha de informar al sistema de cuáles son en este fichero.

Entre sus aspectos más destacables cabe resaltar algunos como el nombre de paquete que identifica la aplicación.

```
package="com.gmail.at.pedrojuan20.immandbeta120803"
```

Ilustración 20 – Nombre del paquete que identifica la aplicación

Para que la aplicación cliente creada tenga acceso a las diferentes funcionalidades del sistema operativo requeridas, se le ha de dar permisos. En nuestro caso se requieren permisos sobre el acceso y control a la cámara del dispositivo, establecer conexiones a través de internet y el estado de la red, a herramientas del sistema para impedir que el dispositivo entre en modo de suspensión o apague la pantalla, acceso al estado del teléfono para en caso de recibir una llamada poner en pausa la aplicación. Son algunos de los utilizados como se muestra a continuación.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

Ilustración 21 – Declaración de permisos en AndroidManifest.xml

En este fichero también se establece el nivel mínimo de versión de Android para poder ejecutar la aplicación. En nuestro caso se ha establecido en la versión 3.0 (API 11) debido a que las versiones anteriores daban problemas con las conexiones de red y como son cada vez menos comunes entre los usuarios se ha decidido excluirlas.

```
<uses-sdk
    android:minSdkVersion="11"
    android:targetSdkVersion="19" />
```

Ilustración 22 – Establecimiento de nivel mínimo de versión Android

Teniendo en cuenta la limitación de recursos de muchos de los dispositivos móviles, sobre todo en lo referente a memoria, donde se va a ejecutar esta aplicación se ha establecido que en caso de que se disponga de una tarjeta de expansión de memoria SD, la aplicación se instalará en ella en vez de en la memoria interna del dispositivo mediante la siguiente etiqueta.

```
android:installLocation="preferExternal" >
```

Ilustración 23 – Establece la preferencia de instalación en la memoria externa (si existe)

4.3. Desarrollo de la aplicación servidor

La aplicación servidor, llamada infolab.php, como ya se ha comentado anteriormente se aloja y ejecuta en el servidor Web. Está escrita en PHP y su labor es la de procesar las peticiones de los clientes, obtener la información requerida de base de datos, procesarla y enviarla de vuelta a los clientes.

PHP ofrece varios controladores y complementos de MySQL para acceder y manejar MySQL. Existen tres APIs de PHP para acceder a las bases de datos de MySQL: *mysql*, *mysqli* y *PDO*. Desde un principio se descartó el uso de *mysql* por estar declarada

obsoleta a partir de PHP 5.5.0 y será eliminada en el futuro. *PDO* se ha descartado por tener un rendimiento inferior a *mysqli* y porque sus ventajas frente a *mysqli* como el soportar multitud de bases de datos vía drivers no se han considerado necesarias para este proyecto.

Cuando el servidor, al recibir una petición de un cliente, ejecuta infolab.php, lo primero que se comprueba es la conexión con la base de datos y si ésta es correcta pasa a la construcción de una "query" o consulta SQL para la obtención de la información relativa al laboratorio solicitado por el cliente en la base de datos. El identificador del laboratorio es enviado por el cliente como parámetro en la URL mediante el método GET de HTTP y extraído por la aplicación servidor mediante la *superglobal*[5]$_GET.

Tras la obtención de la información de la base de datos, es necesario darle formato antes de su envío al cliente que la solicitó. Para ello se ha optado por el uso de JSON. Como ya se ha comentado en apartados anteriores JSON (JavaScript Object Notation) es un formato para el intercambio de datos, con un formato más sencillo que XML.

La Ilustración 24 muestra la estructura que le da al objeto JSON la aplicación servidor antes de su envío al cliente. Está formado a su vez por dos objetos. El primer objeto llamado "constantesInfoLab" contiene un array con datos necesarios para el correcto funcionamiento de la aplicación cliente. Está constituido con datos obtenidos de la tabla "constantes". El segundo objeto es llamado "infoLaboratorio" y está compuesto por tantos arrays como equipos hay en base de datos del laboratorio requerido. Cada uno de estos arrays posee la información detallada de los equipos y que posteriormente será mostrada en la aplicación cliente. Los datos que lo forman están extraídos de la tabla "objeto_info".

También es importante resaltar en este punto, la necesidad de usar el formato de codificación de caracteres Unicode UTF-8 sobre los datos obtenidos de base de datos antes de su envío al cliente, para evitar errores y la aparición que caracteres extraños cuando hacemos uso de acentos o la "ñ" entre otros.

Por último, la labor de la aplicación servidor finaliza con el envío de los datos al cliente y el cierre de la conexión con la base de datos.

[5] Superglobal, son variables internas que están siempre disponibles en todos los ámbitos a lo largo del script.

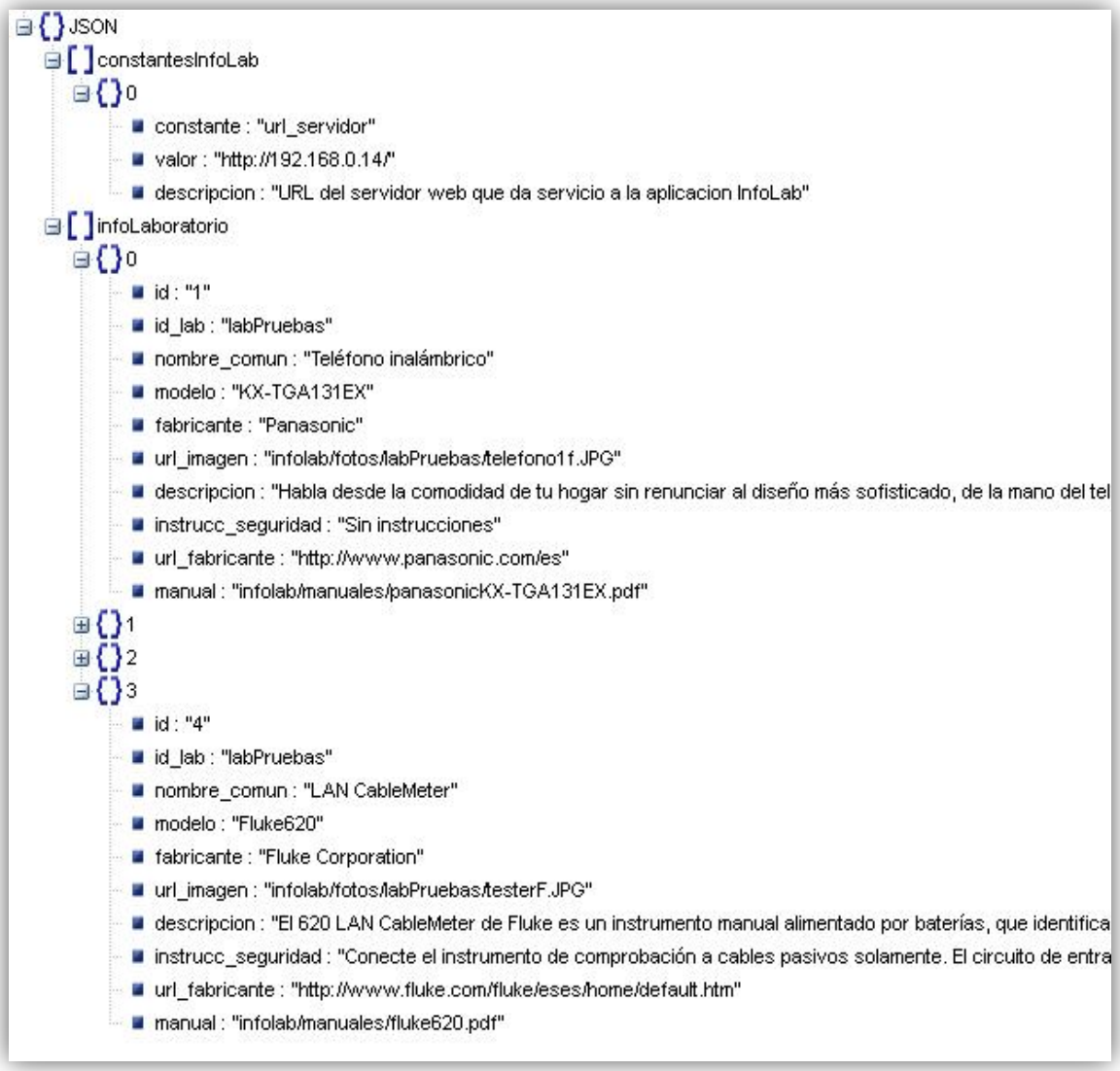


Ilustración 24 – Estructura de datos del JSON diseñado en este proyecto

4.4. Base de datos

La base de datos elegida para dar solución a este proyecto ha sido MySQL. La razón es que MySQL es gratuita, dispone de un sencillo administrador de bases de datos (DBMS) y se puede interactuar con MySQL usando los lenguajes de programación más populares, como PHP.

4.4.1. Estructura de la base de datos

Tras analizar las necesidades de la aplicación cliente se ha definido la siguiente estructura de tablas en base de datos y los siguientes tipos de datos a utilizar que se describen a continuación.

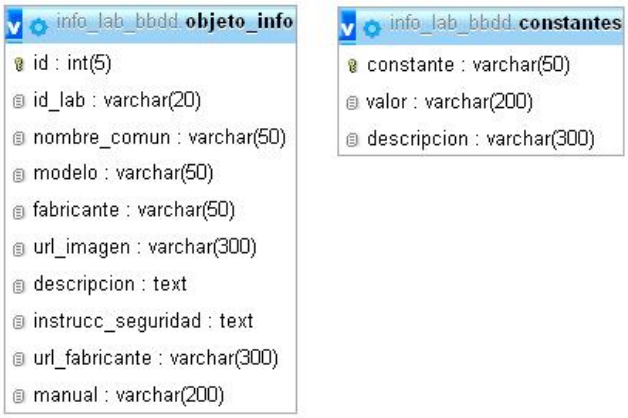


Ilustración 25 – Estructura y tipos de datos de las tablas en BBDD

Tabla objeto_info

Cada entrada de la tabla objeto_info representa un objeto o equipo de un laboratorio junto con toda la información relevante que los usuarios pueden necesitar y que les serán mostrados cuando sean reconocidos por la aplicación cliente.

A continuación, en la Ilustración 26 se muestra un ejemplo de lo que sería una entrada en esta tabla:

id	7
id_lab	labit501
nombre_comun	LAN CableMeter
modelo	Fluke620
fabricante	Fluke Corporation
url_imagen	infoLab/fotos/labit501/testerFluke.jpg
descripcion	El 620 LAN CableMeter de Fluke es un instrumento manual alimentado por batería, que identifica fallos en cables, mide longitudes y verifica el cableado de los siguientes cables de una red de área local. Este instrumento de comprobación…
instrucc_seguridad	Conecte el instrumento de comprobación a cables pasivos solamente. El circuito de entrada tiene protección para soportar voltajes bajos pero una conexión prolongada a líneas de teléfono y redes activas pueden dañar la unidad. Al detectar señales activas…

url_fabricante	http://www.fluke.com/fluke/eses/home/default.htm
manual	infolab/manuales/fluke620.pdf

Ilustración 26 – Muestra de entrada en la tabla objeto_info

Tabla constantes

Esta tabla contendrá constantes necesarias para la aplicación cliente. La razón de ser de esta tabla es que se puedan realizar ciertos cambios en el lado del servidor sin necesidad de que los clientes tengan que actualizar su software.

Deberá contener obligatoriamente al menos una entrada con la dirección del servidor de donde la aplicación cliente descargará las imágenes y los manuales en formato *pdf* en caso de que los requiera. Esta entrada deberá estar compuesta en su campo "constante" por "url_servidor" y en el campo "valor" la URL al servidor.

4.4.2. Codificación de los datos

Durante el desarrollo del proyecto se han producido problemas y errores con la codificación de las comunicaciones entre los distintos elementos del sistema. Como con los acentos o la "ñ" entre otros. Se ha conseguido solucionar utilizando el formato de codificación de caracteres Unicode UTF-8. Siendo necesaria su utilización, desde la creación de las tablas en base de datos, a el procesado de las respuestas en la aplicación servidor (archivos .php) en su preparación para la comunicación HTTP entre cliente y servidor.

4.5. Creación de códigos QR

Para el funcionamiento del sistema, cada laboratorio ha de tener un código QR que lo identifique. Para su creación hay a nuestra disposición varias herramientas gratuitas, desde programas que podemos descargar e instalar, hasta páginas web desde las que crearlos sin necesidad de ninguna instalación.

En el desarrollo de este proyecto, la creación de los códigos QR se ha realizado desde la web http://www.codigos-qr.com/generador-de-codigos-qr/ donde hay que establecer que se desea crear un código QR con una dirección URL y tamaño que queremos que tenga el código. Como ya se ha dicho esto se puede realizar con cualquier otro generador de códigos QR.

El contenido del código ha de ser la URL a la aplicación servidor, pasándole como parámetro el identificador del laboratorio. Este identificador del laboratorio se introduce

en la URL haciendo uso del método GET de HTTP debiéndose llamar este parámetro "idlab".

Veamos un ejemplo de lo que sería la URL que debe contener un código. Nuestra aplicación servidor se llama "infolab.php". Supongamos que utilizamos el servidor Web http://labit501.upct.es/ y que el identificador del laboratorio es "labitPruebas". Con estos datos, la URL con la que debemos construir el código QR sería:

http://labit501.upct.es/infolab.php?idlab=labitPruebas

Ilustración 27 – Ejemplo de código QR apto para la aplicación

5. Manual de usuario

En este capítulo se proporciona la documentación necesaria que detalla como los usuarios han de instalar la aplicación creada en sus smartphones, como interactuar con ella, sus capacidades y como se usan.

La interfaz de usuario para la aplicación cliente ha sido diseñada para su fácil uso de un modo intuitivo y claro. Aun así en este apartado se explica su funcionalidad junto con los requisitos de hardware y software que han de cumplir los dispositivos móviles para su ejecución.

5.1. Requisitos de los clientes

Para la utilización de la aplicación InfoLab los clientes deben cumplir los siguientes requisitos:

- Disponer de un dispositivo móvil con sistema operativo Android en su versión 3.0 o superior. Aunque teóricamente es compatible para versiones 3.0 y superiores, sólo se ha podido comprobar su funcionamiento en la versión 4.4.2.
- El uso de la librería *ImageMatching* para el reconocimiento de imagen, impone la condición de que se disponga de una arquitectura de procesador ARV7 o superior.
- Estar el dispositivo móvil equipado con cámara.
- Disponer de 15 MB de memoria disponibles para su instalación.
- Tener conexión a internet o a una red WIFI que de acceso al servidor.

5.2. Instalación aplicación cliente

La instalación normal de una aplicación Android es mediante Google Play Store[6], pero debido a la restricción puesta por el fabricante de la librería *ImageMatching* utilizada en este proyecto, prohíbe la distribución de la aplicación desarrollada mediante Google Play Store. Por tanto la instalación ha de hacerse desde un archivo **.apk**. Apk (Application PacKage File) es un paquete para el sistema operativo Android. Es una variante del formato JAR de Java y se usa para distribuir e instalar componentes empaquetados para la plataforma Android. Por tanto para la instalación de la aplicación se debe descargar el archivo **InfoLab.apk** en el dispositivo cliente y mediante un explorador de archivos ejecutarlo para su instalación como se puede ver en la Ilustración 28 e Ilustración 29.

[6] Google Play Store. Plataforma de distribución digital de aplicaciones móviles para dispositivos con sistema operativo Android.

Ilustración 28 – Archivo InfoLab.apk desde explorador de archivos

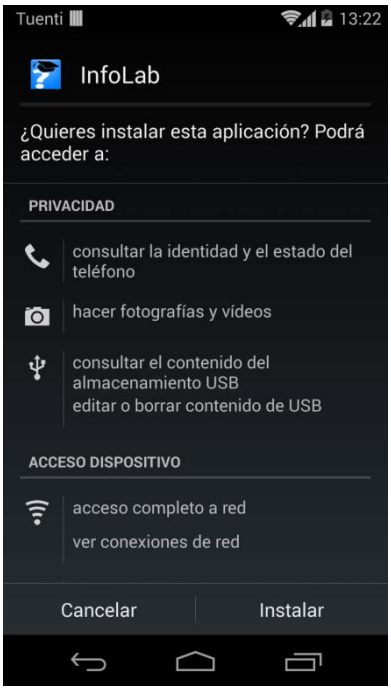


Ilustración 29 – Proceso de instalación

Para su instalación será necesario activar previamente en el smartphone la opción "Orígenes desconocidos", que nos permitirá instalar aplicaciones que no han sido descargadas del Play Store. Esta opción se encuentra en Ajustes>Seguridad>Orígenes desconocidos.

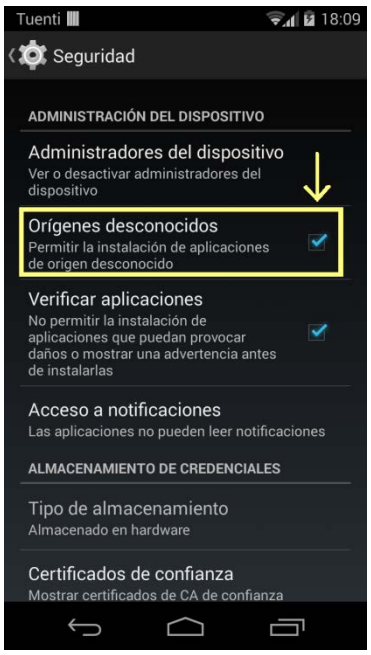


Ilustración 30 – Orígenes desconocidos en opciones de seguridad

5.3. Manejo de la aplicación cliente

Para la utilización de la aplicación cliente InfoLab, hemos de ejecutarla haciendo uso del icono creado en su instalación.

Ilustración 31 – Icono de la aplicación

Una vez abierta veremos la primera pantalla de la aplicación, en la que se muestra en todo momento lo capturado por la cámara y un mensaje que nos indica que enfoquemos al código QR de identificación del laboratorio (Ilustración 32). Una vez leído el código QR, se procede a la descarga y procesado de las imágenes (Ilustración 33), si todo va bien, una vez finalice volveremos a ver en pantalla lo capturado por la cámara junto con un mensaje indicándonos que enfoquemos el equipo del que queremos obtener información (Ilustración 34). Cuando la aplicación identifica un objeto, es comunicado al usuario mediante una ventana emergente con el nombre del objeto o equipo y preguntando si deseamos obtener más información (Ilustración 35). Al elegir ampliar la información se nos muestra una nueva pantalla, con toda la información referente a ese objeto de un modo ordenado y estructurado (Ilustración 36). Desde esta pantalla también se puede tener acceso a la web del fabricante (Ilustración 37), al manual de usuario (Ilustración 38) o volver atrás para identificar otro equipo.

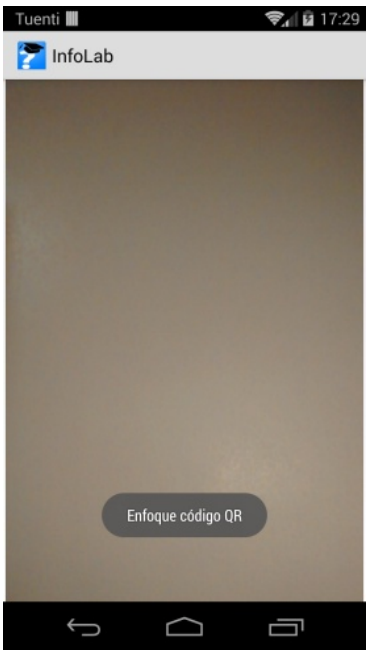


Ilustración 32

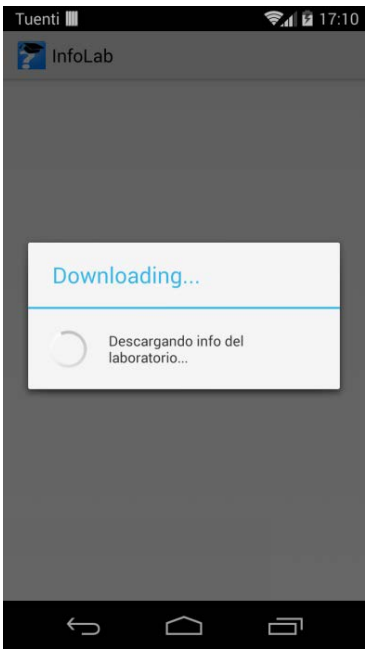


Ilustración 33

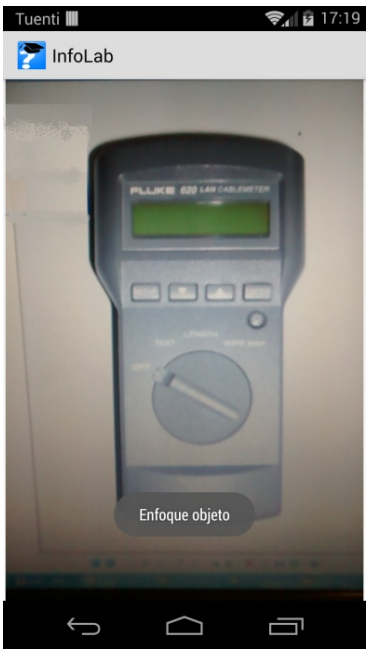


Ilustración 34

Ilustración 35

Ilustración 36

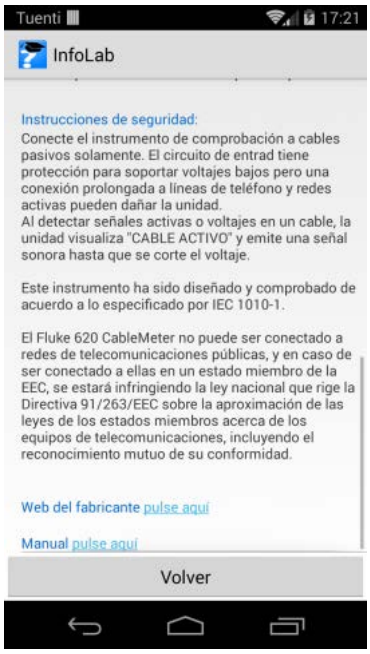


Ilustración 36a

Ilustración 37

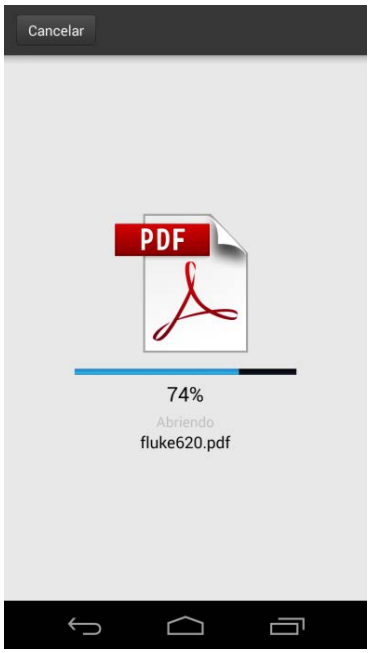


Ilustración 38

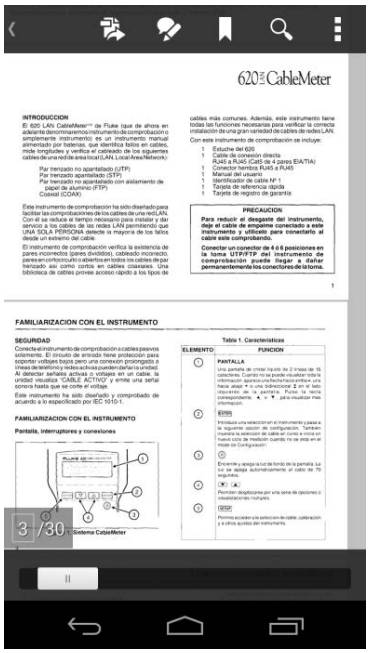


Ilustración 38a

Para salir de la aplicación simplemente hay que ir pulsando el botón “Volver” hasta que se salga de ella.

5.4. Mensajes de error

En la programación de la aplicación cliente se ha procurado cubrir todos los posibles errores, que en tiempo de ejecución pudieran ocasionar un cierre brusco de la aplicación o un mal funcionamiento, lo que transmite una sensación de mala calidad a la hora de interactuar con ella. Por este motivo se ha intentado en todo momento capturar los errores y tratarlos para su notificación al usuario y poder seguir con la ejecución.

Las notificaciones de error más comunes que pueden producirse son las derivadas de problemas con la conexión con el servidor ya sea por no haber conexión de red o que el servidor no responda.

El mensaje de error de la Ilustración 39 se produce cuando la aplicación, al intentar conectar con el servidor comprueba que el smartphone no dispone ni de conexión de datos móviles, ya sea por estar deshabilitados o por no haber cobertura, ni está conectado a ninguna red WIFI en ese momento. Para solucionar el problema hay que conectarse a una red WIFI o habilitar el uso de datos móviles.

Ilustración 39 – Mensaje sin conexión

Ilustración 40 – Mensaje de error de conexión

El mensaje de error de la Ilustración 40 se produce cuando no se obtiene una respuesta adecuada del servidor. Esta respuesta inadecuada puede ser causada por motivos varios, como que el código QR leído no es válido o que el servidor no esté en funcionamiento.

6. Conclusiones y futuras mejoras

6.1. Conclusiones

Con la realización de este proyecto se ha logrado crear una herramienta educativa de gran atractivo por su fácil manejo, utilidad y modo de funcionamiento. Con ella los alumnos de la UPCT tendrán un complemento a la hora de desempeñar su formación en los laboratorios, al facilitarles la realización de sus prácticas y reduciendo la carga del profesorado a la hora de resolver dudas. Todo ello a partir de un concepto muy simple, el identificar el equipo de los laboratorios con sólo enfocarlos con la cámara de un smartphone, obteniendo de un modo rápido toda la información necesaria para su uso.

Se ha puesto de manifiesto las posibilidades que otorga la plataforma Android para la creación de aplicación de realidad aumentada y reconocimiento de imagen en dispositivos móviles, mediante herramientas accesibles que permiten a desarrolladores iniciarse en la creación de aplicaciones de un modo relativamente sencillo y accediendo a un mercado que está en plena expansión. Desde un primer instante se puede tener acceso a todo tipo de herramientas de desarrollo, documentación y ejemplos facilitados tanto por desarrolladores independientes, empresas y aficionados, que entre todos han creado una gran comunidad en internet para compartir técnicas y conocimientos que facilitan el aprendizaje. Es de resaltar la encomiable labor que en este sentido ha realizado Google y la gran utilidad que ha tenido para la creación de este proyecto su web *developer.android.com* (13).

Ha resultado muy grato en este trabajo ver como finalmente se ha logrado la interacción entre distintas tecnologías, que han hecho a nuestro teléfono móvil capaz de reconocer objetos, crear conexiones web, comunicarse con un servidor el cual a su vez lo hace con una base de datos, haciendo uso de todo un abanico de tecnologías muy utilizadas en el mundo real y de libre distribución como son PHP, MySQL, HTTP, Apache y JAVA entre otras, llevando muchos conceptos conocidos de lo teórico a lo real.

Con el auge y la cada vez mayor importancia de Android, su futuro cada vez pinta mejor y no se reduce sólo al campo de la telefonía. Este proyecto me ha permitido iniciarme en el mundo de la programación de las aplicaciones Android y estoy seguro de que mi experiencia con él, no terminará aquí.

6.2. Futuras mejoras

Con respecto a las futuras posibilidades de trabajo para mejorar el sistema creado, tal vez la más destacable sería la de crear una sencilla aplicación web para el mantenimiento de los datos almacenados de la base de datos. Aplicación con la que introducir nuevos equipos en el sistema, modificar los existentes o eliminarlos. Esta aplicación sustituiría lo que hasta ahora se ha realizado mediante phpMyAdmin y se ganaría en comodidad a la hora de que el administrador realice el mantenimiento de los

datos. Además se lograría que al no tener acceso directo a la base de datos, no se pudieran modificar elementos sensibles como la estructura de las tablas.

Un campo que también podría mejorarse sería la interfaz gráfica de la aplicación cliente cuando ésta muestra los datos del equipo identificado, que aunque está completa y la muestra de un modo claro, en lo referente a la estética siempre es posible dar una vuelta más de rosca. En cuanto al tema visual, también sería interesante centrarse en el aspecto que tendría la aplicación Android al ejecutarse en *tablets*, ya que aunque en teoría no habría problemas no se ha podido probar en dispositivos con pantallas de más de 5 pulgadas.

Otro aspecto en el que se podría trabajar de un modo continuado sería en ir aumentando la información y descripción de los equipos de los laboratorios. Aparte de añadir nuevos equipos también se podría aumentar los campos de información, como ejemplos de uso, consejos de cuidado y mantenimiento, prácticas en los que son utilizados, opiniones o cualquier otro aspecto que se crea valioso para los usuarios.

7. Bibliografía y referencias

1. **Roca, José Miguel.** *Que es un smartphone.* [En línea] 2014. http://www.informeticplus.com/que-es-un-smartphone.

2. **Lee, Wei-Meng.** *Beginning Android 4 application development.* s.l. : Wiley, 2012.

3. **Meier, Reto.** *Profesional Android 4 application development.* s.l. : Wiley, 2012.

4. *QRcode.* [En línea] 2014. http://www.qrcode.com/en/about/.

5. **Tinoco, Ruth Gamero.** *En que consiste la realidad aumentada.* [En línea] 2014. http://blogthinkbig.com/en-que-consiste-la-realidad-aumentada/.

6. **García Santillán, Iván Danilo.** *Visión Artificial y Procesamiento Digital de Imágenes.* s.l. : Ibarra, 2008.

7. *ARLab.* [En línea] 2014. http://www.arlab.com/company.

8. **Ullman, Larry.** *MySQL Guía de aprendizaje.* [ed.] Pearson Educación. 2004.

9. *php.* [En línea] 2014. http://www.php.net/.

10. *Java.* [En línea] 2014. https://www.java.com/es.

11. *Eclipse.* [En línea] 2014. http://www.eclipse.org/org/.

12. *notepad++.* [En línea] 2014. http://notepad-plus-plus.org/.

13. *Developer android.* [En línea] 2014. http://www.developer.android.com/.

14. **Sood, Raghav.** *Pro Android Augmented Reality.* s.l. : Apress Berkely, 2012.

15. **Brandski, Gary y Kaehler, Adrian.** *OpenCV.* s.l. : O'Reilly Media, 2008.

16. **H.F.Korth.** *Fundamentos de bases de datos.* s.l. : McGraw-Hill, 1995.

17. Android ya. [En línea] 2014. http://www.javaya.com.ar/androidya/.

18. MySQL. [En línea] 2014. http://www.mysql.com/.

19. *Eclipse tutorial.* [En línea] 2014. http://www.vogella.com/tutorials/Eclipse/article.html.

20. StatCounter Global Stats(estadísticas). [En línea] 2014. http://gs.statcounter.com/.