UNIVERSIDAD DE

VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

MENCIÓN EN TELEMÁTICA

Desarrollo de una app móvil de ayuda al estudio de la asignatura "Técnicas y Protocolos de Redes Telemáticas"

Autor:

Ana García Riol

Tutor:

Isabel de la Torre Díez

Valladolid, 14 de Julio de 2017

TÍTULO:	Desarrollo de una app móvil de ayuda al estudio de la asignatura “Técnicas y Protocolos de Redes Telemáticas”
AUTOR:	Ana García Riol
TUTOR:	Isabel de la Torre Díez
DEPARTAMENTO:	Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TRIBUNAL

PRESIDENTE:	Miguel López-Coronado
VOCAL:	Beatriz Sainz de Abajo
SECRETARIO:	Isabel de la Torre Díez
SUPLENTE:	Carlos Gómez Peña
SUPLENTE:	Salvador Dueñas Carazo

FECHA:	14 de Julio de 2017
CALIFICACIÓN:	

Resumen

En la actualidad, la tecnología cada vez tiene una mayor relevancia en el día a día de las personas. Sobre todo el empleo de *smartphones* es algo que está totalmente extendido. La utilización habitual de estos dispositivos, junto a su facilidad de uso, ha dado lugar al desarrollo de numerosas *apps*. Casi cualquier persona, independientemente de sus conocimientos sobre tecnología, es capaz de manejar un móvil, instalando y manejando diversas *apps* a diario.

Esta evolución del uso de los *smartphones* y las *apps* ha permitido que se empleen más allá del ámbito social de las personas, apareciendo por ejemplo, *apps* destinadas a la educación, ya sea solo vía online o como apoyo al método de docencia tradicional. Esto es lo que se conoce con el nombre de *m-learning*.

Estas aplicaciones permiten a los estudiantes acceder a la información en cualquier momento y lugar, empleando simplemente su *smartphone*. Por lo tanto, suponen un ahorro tanto económico como de tiempo, al no tener que desplazarse a un lugar concreto para recibir la información. También existen múltiples plataformas y webs con el mismo fin (*e-learning*), pero la ventaja del *m-learning* es que no se necesita disponer de equipos pesados como un ordenador.

El objetivo de este proyecto es el desarrollo de una aplicación para dispositivos móviles *Android* que permita a los estudiantes de la asignatura *Técnicas y Protocolos de Redes Telemáticas* acceder a los recursos de la asignatura simplemente a través de su dispositivo móvil. Esto permite a los alumnos acceder a la información de la asignatura en aquellos "ratos muertos" que no tienen nada que hacer, como cuando viajan en bus. También es muy útil cuando el alumno tiene que realizar una práctica de laboratorio, ya que dispone de toda la información en su *smarthphone* y no necesita una versión impresa. Y aunque prefiera tener la información en papel, puede disponer igualmente de esta información si algún día se le olvida llevar la versión impresa.

Palabras claves

Android, m-learning, app, Técnicas y Protocolos de Redes Telemáticas (TPRT), educación.

Abstract

Day by day technology is becoming more relevant in people life. The use of smartphones is something common in all areas, because they are easy to use. That makes posible the development of many applications. Everyone, regardless of their knowledge, are able to work with a smartphone, installing or handling several applications everyday.

The evolution in the use of smartphones and the improvement of the software have made possible to use them outside of the social environment. For example, releasing applications for education not only as an online way, but also as a traditional way of learning. This is known as m-learning.

These applications allow the students to access the information everywhere and everytime by using only a smartphone. This situation is more economic and saves time. Students don't need to go to class to recieve the information. Also there are multiples ways and webs with the same porpouse (e-learning) but the main advantage of m-learning is that you don't need heavy devices like a computer.

The porpouse of this Project is to release a mobile application for Android devices that allow students of the subject *Técnicas y Protocolos de Redes Telemáticas* to get the resources only with their smartphone. This allows students to access to the information of the subject in rest times like when they are travelling by bus. It is also very useful when students are in a laboratory practice, because they can have all the information in their smarthphones and don't need a printed versión. And even if students prefer to have the information on paper, they can also check this information if they forget to take the printed version.

Keywords

Android, m-learning, app, Técnicas y Protocolos de Redes Telemáticas (TPRT), education.

Agradecimientos

Agradecer en primer lugar a mis padres por el enorme apoyo que me han brindado desde el inicio de la carrera y por los valores inculcados durante toda mi vida. Gracias también a mi hermano, por sus ánimos y sus fabulosas ideas.

Agradecer también el apoyo a las personas que han hecho posible este TFG, en especial a mi tutora Isabel de la Torre.

Y por último, dar las gracias a mis amigos y compañeros de carrera, que me han acompañado todos estos años, estando presentes tanto en los buenos momentos, como en aquellos que no son tan buenos.

A todos vosotros, gracias.

Índice

Indice de Figuras

Capítulo 1. Introducción

1.1 Introducción

Las Tecnologías de la Información y la Comunicación (*TIC*) cada vez están más presentes en la vida cotidiana de las personas, formando ya parte de la mayoría de sectores, como la robótica, la salud o la educación.

La aplicación de las *TIC* al sistema educativo tiene tres variantes: *e-learning (electronic learning), b-learning (blended learning) y m-learning (mobile learning).*Estas variantes, de las que se hablará en el Capítulo Segundo de este documento, permiten la formación de los alumnos a cualquier hora, y en casi cualquier lugar, ya que se desarrollan gracias al empleo de múltiples plataformas virtuales, como plataformas vía web o aplicaciones para teléfonos móviles.

En este documento se presentará el desarrollo de una herramienta *m-learning*: una aplicación para dispositivos móviles *Android* que ayudará al estudio de la asignatura de tercer curso de *Grado en Ingeniería de Tecnologías Específicas de Telecomunicación con mención en Sistemas Electrónicos* denominada *Técnicas y Protocolos de Redes Telemáticas*.

1.2 Motivación

Vivimos en una época marcada por la revolución tecnológica. Hoy en día la mayoría de las personas tienen acceso a un ordenador y un teléfono móvil que casi siempre llevan consigo. Sobre todo entre la población joven el teléfono móvil está totalmente extendido.

Esto ha llevado a que en la actualidad, casi todas las empresas e instituciones que ya tenían una página web, hayan desarrollado también una *app* para estar disponibles tanto en la web como en los dispositivos móviles. Este cambio no solo es debido al uso masivo de los dispositivos móviles, sino también a la facilidad que estos proporcionan para distribuir las *apps* en las distintas plataformas de distribución, como *Play Store* o *App Store*.

Al igual que las empresas e instituciones, la educación también necesita renovarse y modificar sus métodos docentes, aplicando las *TIC* al sistema educativo. Esto es lo que se conoce como *e-learning*, o *m-learning* en el caso de las *apps* para teléfonos móviles.

La motivación principal de este proyecto es dar respuesta a la necesidad indicada en el párrafo anterior, creando una aplicación móvil que permita el acceso a los contenidos e información de la asignatura *Técnicas y Protocolos de Redes Telemáticas* de la Universidad de Valladolid.

Esta aplicación permitirá a los alumnos tener acceso a la información de la asignatura en cualquier momento y lugar, siempre y cuando dispongan de un *smartphone* o una *tablet*. Esto permite por ejemplo que puedan estudiar la asignatura mientras van en autobús o que si algún día al alumno se le olvidan las prácticas o la teoría, pueda acceder a estas desde su teléfono móvil.

Pero no solo ayudará a los alumnos de esta asignatura a estudiarla, si no que cualquier persona que quiera aprender conceptos relacionados con la materia puede encontrar en ella información muy interesante, así como algunas partes prácticas para poner a prueba sus conocimientos teóricos.

1.3 Objetivos

Este proyecto tiene como objetivo principal desarrollar una nueva aplicación móvil, que permita a los alumnos de la asignatura *Técnicas y Protocolos de Redes Telemáticas* de la Universidad de Valladolid visualizar los contenidos de la misma en cualquier lugar, usando simplemente un Smartphone. También está destinada a todas las personas interesadas en este tema, ya que cualquiera debería poder tener acceso a la aplicación y la información.

Con esta aplicación se pretende lograr una mayor eficacia en el estudio de la asignatura, poniendo a disposición de los alumnos toda la información sobre esta, accesible en cualquier momento y lugar, con el único requisito de disponer de un *smartphone* o una *tablet*.

1.4 Materialización del prototipo y recursos necesarios

Esta versión de la *app* ha sido desarrollada solamente en *Android*. Para su desarrollo ha sido necesario disponer de un ordenador en el que instalar el entorno de desarrollo integrado (*IDE*) *Android Studio*, que es un software de código abierto y gratuito. En este caso se ha utilizado un ordenador con el sistema operativo *Windows 10*. Para probar el correcto funcionamiento de la aplicación hace falta un *smartphone* o una *tablet* con el sistema

operativo *Android 4.2* o superior. En este caso se ha empleado el *smartphone Cubot x9*, con el sistema operativo *Android 4.4.4*.

A la hora de calcular el coste de los recursos utilizados, se podría distinguir entre los costes asociados al *hardware* y los asociados al *software*. Los recursos *hardware* (ordenador y *smartphone*) ya estaban disponibles para el alumno, así que no serían computables. Para calcular los costes asociados al *software* habría que tener en cuenta el IDE *Android Studio* y el sistema operativo del ordenador (*Windows 10*), pero cabe destacar que *Android Studio* es de código abierto y gratuito, y que de *Windows 10* se ha empleado la versión sin licencia y gratuita, por lo que estos costes tampoco son computables. Por lo tanto, el único coste real a tener en cuenta es el relacionado con el desarrollo software del proyecto.

1.5 Partes del documento

El presente documento está formado por seis capítulos, como se explica a continuación:

- El primer capítulo es la introducción al proyecto y al presente documento. Explica la motivación para realizar este proyecto, los objetivos del mismo y la y recursos necesarios para realizarlo.
- En el segundo capítulo se explicará como las Tecnologías de la Información y la Comunicación han influido en el ámbito educativo.
- En el tercer capítulo se analizan las diversas tecnologías existentes para llevar a cabo este proyecto, exponiendo en motivo de la elección de cada una.
- El cuarto capítulo trata del desarrollo del proyecto. En él se explica el desarrollo de la aplicación, los resultados finales y su evaluación en distintos dispositivos.
- En el quinto capítulo se exponen las conclusiones y líneas futuras de la aplicación.
- El sexto capítulo recoge la bibliografía utilizada para realizar este proyecto.

Capítulo 2. TIC en educación

2.1 Introducción

Las *Tecnologías de la Información y la Comunicación (TIC)* cada vez están más presentes en la vida cotidiana de las personas, formando ya parte de la mayoría de sectores, como la robótica, la salud o la educación. En el ámbito de la educación hay tres ramas fundamentales de las TIC: *e-learning (electronic learning), b-learning (blended learning) y m-learning (mobile learning)*.

Con estas ramas de *TIC* lo que se pretende es intentar contribuir a aumentar la eficiencia y la calidad de la educación. Muchas veces se asocian a la educación a distancia y esto es un error, ya que como mejor eficiencia tienen es como un apoyo a las técnicas de enseñanza tradicionales, y no como un sustituto de las mismas.

Lo que se pretende con estas tecnologías es aumentar la calidad de la educación, elaborando herramientas virtuales que apoyen los cursos presenciales del alumno. Estas herramientas también permiten desarrollar parte de las enseñanzas desde el lugar de estudio del alumno, de manera que se optimice el tiempo y se logre una mayor eficacia de las reuniones presenciales.

Actualmenten el *e-learning* sirve de apoyo a los métodos de docencia tradicionales. Pero según un estudio de *Online Business School* (*OBS*), publicado en el periódibo "*El mundo*" [12] en el 2019 el 50% de la enseñanza superior del mundo se impartirá a través de e-learning. Un ejemplo de que la enseñanza online es posible es la *Universidad Obertade Catalunya* (*UOC*), que presume de ser la primera institución de grado superior solo virtual a nivel mundial y cuyas titulaciones cumplen los estándares exigidos a nivel europeo, ya que están avaladas por la *Agencia Nacional de Evaluación de la Calidad y Acreditación* (*ANECA*) y la *Agencia para la Calidad del Sistema Universitario de Cataluña* (*AQU*).

2.2 E-Learning: Electronic Learning

E-*learning*, teleformación o aprendizaje virtual es un modelo de formación que utiliza Internet y las *TIC* como herramientas básicas para la comunicación e interacción entre todos los participantes en el proceso educativo, el acceso a los contenidos didácticos y para el

desarrollo integral de todas las actividades formativas. Algunas de las características de *e-learning* más importantes que menciona Pau Yanez en su artículo "*E-learning, M-learning y B-learning*" [15] son:

- Pone a disposición de los alumnos un amplio volumen de información que se puede actualizar fácilmente. Esta información puede ser auditiva, visual o audiovisual.
- Favorece la interactividad del alumno con la información, el docente y los otros alumnos.
- Ofrece diferentes herramientas de comunicación para estudiantes y para profesores, tanto síncronas como asíncronas.
- Tiende a reducir el tiempo de formación de las personas, ya que puedes acceder a la información en cualquier momento y desde cualquier lugar. Esta formación es conocida como *anytime, anywhere*.
- Permite que los servidores registren toda la actividad realizada por los estudiantes en el *e-learning*.
- Ahorra costes por desplazamiento tanto por parte de los docentes como por parte de los estudiantes.

2.3 M-Learning: Mobile Learning

Con el aumento del uso de los teléfonos inteligentes, el concepto de *e-learning* ha evolucionado, permitiendo la participación de los docentes y los estudiantes en acciones formativas a través de sus dispositivos móviles, lo que se conoce como *m-learning*. Las características adicionales de *m-learning* con respecto a *e-learning, según* meciona Pau Yanez en su artículo [15], son:

- Se utiliza con dispositivos más pequeños e inalámbricos, como *smartphones* o *tablets*.
- Interacción inmediata.
- Permite usarlo en cualquier lugar, incluso mientras el usuario está cambiando de ubicación.
- El empleo de dispositivos móviles está tan integrado en las actividades diarias que apenas se nota el cambio.

2.4 B-Learning: Blended Learning

Según Alejandra Collazos [1], el *b-learning* es una combinación de la enseñanza presencial y de la enseñanza virtual, es decir, un curso realizado en este formato incluye tanto clases presenciales como actividades de *e-learning* o *m-learning*. Es una muy buena opción para los profesores para traer elementos al campo presencial en vez de simplemente utilizar materiales digitales como elementos complementarios del curso.

Este método de aprendizaje no define exáctamente qué porcentaje de las actividades tiene que ser presencial y cuál no, sino que cada profesor puede acomodarlo a su gusto para cada material que imparta.

Este modelo de formación hace uso tanto de las ventajas de la formación 100% online como las de la formación presencial, combinándolas en un solo tipo de formación que agiliza la labor tanto del docente como del estudiante. Algunas de las principales ventajas que indica Santiago Moll en su artículo "*Blended learning, 15 Razones para adopter este modelo de enseñanza*" [16] son las siguientes:

- El aprendizaje combinado beneficia a los estudiantes, ya que ofrece un modelo de aprendizaje más personalizado. Los estudiantes tienen más probabilidades de mejorar al poder desarrollar en su casa parte del contenido de las asignaturas. Esto permite a cada estudiante ajustar su ritmo de aprendizaje.
- Aumenta la flexibilidad de acceso, ya que las herramientas *e-learning* y *m-learning* están siempre disponibles.
- No pretende sustituir las clases presenciales, es un complemento a estas.
- Permite la convivencia con distintos materials (formato en papel y formato digital), aunque aboga por un contenido eminentemente digital.
- Mejora el aprendizaje cooperativo. Es tipo de aprendizaje trata de mejorar la solidaridad y la ayuda mutual tanto en las actividades presenciales como en las virtuales.
- Mejora la retroalimentación o feedback, al poder llevarse acabo de forma presencial y de forma virtual, a través de herramientas como un chat o un foro de discusión.
- Aunque la asistencia a las clases presenciales sigue siendo muy recomendable, *blended learning* puede reducer su importancia en caso de que algún estudiante no pueda asistir. Mediante la enseñanza no presencial el estudiante puede ponerse al día de una

forma más rápida y efeciva, llevando un ritmo de trabajo desde su ubicación similar al que sus compañeros realizan en el aula.

Se distinguen seis modelos diferentes de blender learning, en function del porcentaje de material que se de de forma prsencial o de forma online. Estos modelos, ordenados del modelo más presencial al más virtual, son los siguientes: *cara a cara, rotación, flexible, laboratorio online, mitad y mitad y online*.

Capítulo 3. Justificación de las tecnologías empleadas

3.1 Introducción

En el capítulo presente se hará un análisis de las distintas tecnologías disponibles para desarrollar la aplicación móvil, analizando una a una y justificando por qué se ha seleccionado una u otra. Por un lado se expondrán los diferentes sistemas operativos móviles, y por otro las plataformas de desarrollo para el sistema operativo seleccionado.

3.2 Tipos de aplicaciones móviles

Hay tres tipos de aplicaciones móviles: aplicaciones nativas, aplicaciones web y aplicaciones híbridas. En este apartado se explicará cada uno de ellos, justificando el tipo elegido para el desarrollo de esta aplicación.

3.2.1 Aplicaciones nativas

Este tipo de aplicaciones son las que se desarrollan de forma específica para un determinado sistema operativo. Cada una de estas plataformas tiene un sistema operativo diferente, por lo que para que una *app* funcione de forma eficiente en varias plataformas, se debería de desarrollar una *app* diferente para cada una de las plataformas deseadas. Cuando hablamos de desarrollo de aplicaciones para móviles, generalmente nos referimos a las aplicaciones nativas.

Las ventajas que ofrecen estas aplicaciones nativas son las siguientes:

- Posibilidad de acceder a todas las características *hardware* del dispositivo.
- Mejor experiencia de usuario.
- Envío de notificaciones a los usuarios.
- Visibilidad en todas las tiendas virtuales de distribución de aplicaciones.

Los inconvenientes de las aplicaciones nativas son:

- Desarrollo de una *app* diferente para cada plataforma de destino.
- Generalmente son más costosas de desarrollar.

3.2.2 Aplicaciones web

Las aplicaciones web, a diferencia de las nativas, se pueden programar independientemente del sistema operativo en el que se usará la aplicación, por lo que se podrán ejecutar en diferentes dispositivos sin tener que crear varias aplicaciones, independientemente del sistema operativo de estos. Se desarrollan en lenguajes muy conocidos, por ejemplo *HTML, Javascript* y *CSS*.

Estas aplicaciones web se ejecutan directamente dentro del propio navegador web del dispositivo a través de una URL, lo que por una parte es una ventaja al no ser necesario instalarlas, pero por otra es un inconveniente ya que no pueden estar visibles en *App Store*, la tienda virtual de *Apple*.

Ventajas de las aplicaciones web:

- Proceso de desarrollo más sencillo y económico, ya que el código base se puede utilizar en múltiples plataformas.
- No necesita aprobación externa para publicarse.
- Al ser una *URL* a una web, el usuario siempre dispone de la última versión de la aplicación.
- Pueden reutilizase webs "*responsive*" ya diseñados.

Los inconvenientes de las aplicaciones web son:

- Acceso muy limitado a las características *hardware* del dispositivo.
- Requiere conexión a internet para su utilización además de un gasto de datos que no siempre es posible por parte del usuario.
- La experiencia de usuario es peor, ya que no siempre están perfectamente adaptadas a los diversos tamaños de dispositivos móviles.

3.2.3 Aplicaciones híbridas

Estas aplicaciones son una combinación de las dos anteriores, empleando las ventajas de cada una de ellas. Las principales ventajas de estas aplicaciones son:

- Visibilidad en todas las tiendas virtuales de distribución de aplicaciones.
- Requieren instalación al igual que las aplicaciones nativas, aunque están desarrolladas en *HTML*, *Javascrip*t y *CSS*.

- El mismo código base se puede utilizar en múltiples plataformas.
- Posibilidad de acceder a la mayor parte del *hardware* del dispositivo.

Este tipo de aplicaciones, aunque menos que las anteriores, también tiene algunos inconvenientes:

- Experiencia de usuario más propia de una aplicación *web* que de una *app* nativa.
- Diseño visual no siempre relacionado con el sistema operativo en el que se muestra.

3.2.4 Elección

El fin de este proyecto es la creación de una *app* que esté disponible para los estudiantes de la asignatura el mayor tiempo posible, por lo que aunque una aplicación híbrida serían una buena opción, es preferible realizar una aplicación nativa de *Android* para que esté disponible en cualquier momento y lugar, aunque no se disponga de conexión a internet. Además las aplicaciones nativas generalmente tienen una mejor experiencia de usuario al mostrar un diseño visual más acorde al sistema operativo en el que se encuentra la aplicación.

3.3 Sistemas operativos móviles

Un Sistema Operativo (*OS*) es un conjunto de programas especialmente hechos para la ejecución de varias tareas (*software*), en las que sirve de intermediario entre el resto de programas del sistema, el usuario y los dispositivos *hardware*.

Las funciones básicas de un sistema operativo son:

- Coordinar el *hardware*.
- Administrar los recursos de la máquina, para evitar que los programas entren en conflicto.
- Cargar en memoria y facilitar la ejecución de los programas que el usuario utiliza. Cuando un programa está en ejecución, el sistema operativo continúa trabajando.
- Organizar los archivos y directorios en dispositivos de almacenamiento.

Hay varios sistemas operativos posibles para los dispositivos móviles. El más extendido actualmente es *Android*, el sistema operativo de *Google*, que con un 37.93% de usuarios supera incluso al sistema operativo de *PC Windows*, con un 37.91%. Los principales

competidores de *Android* son *iOS* y *Windows 10 Mobile*, aunque lo siguen a larga distancia (ver Figura 1).

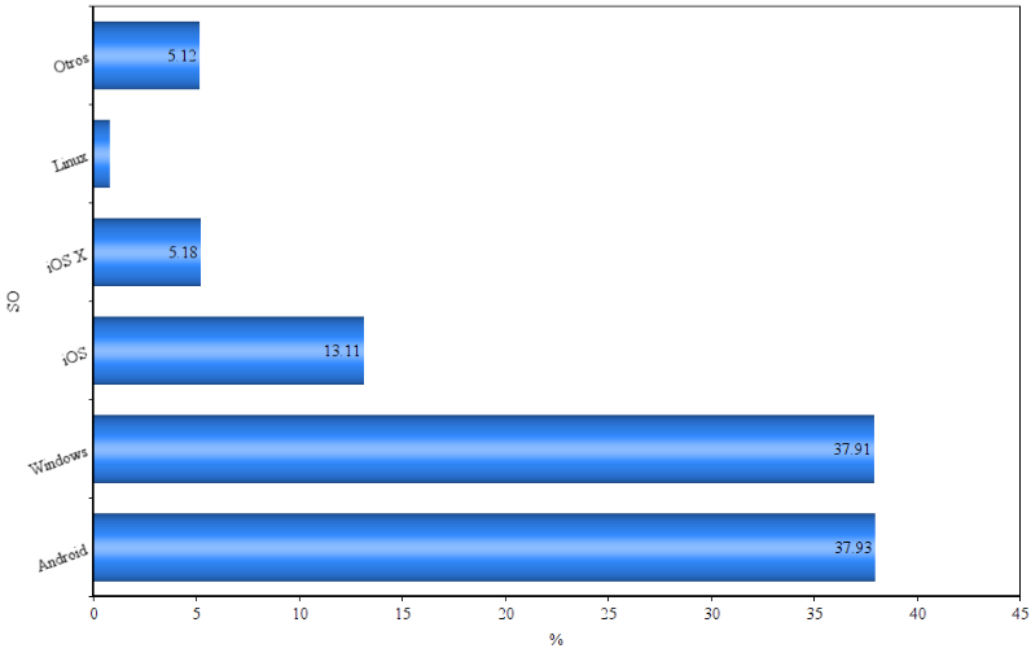

Figura 1. Sistemas Operativos más utilizados en marzo de 2017.

Este apartado se expondrá una breve explicación de los sistemas operativos móviles mencionados anteriormente y se justificará el sistema operativo elegido para el desarrollo de la aplicación.

3.3.1 Android

Android es un sistema operativo diseñado inicialmente por *Android Inc.*, empresa que *Google* respaldó económicamente y más tarde compró. Fue diseñado inicialmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes y *tablets*, al igual que *iOS* o *Windows Phone*. La principal diferencia entre *Android* y estos otros sistemas operativos es que está basado en *Linux*, un núcleo de sistema operativo libre, gratuito y multiplataforma, lo que le hace diferenciarse totalmente del resto.

3.3.1.1 Historia y evolución de Android

Como indica Jesús Tomás Gironés en el manual *"El gran libro de android"* [11], *Android Inc.* fue adquirida por *Google* en el año 2005. Ese mismo año empezaron a trabajar en la creación de una máquina virtual Java optimizada para móviles, a la que llamaron *Dalvik CM*.

En el año 2007 se crea el consorcio *Handset Alliance V*, del que *Google* decidió formar parte, con el objetivo de desarrollar estándares abiertos para móviles. Entre los miembros de esta alianza estaban, entre otros, grandes empresas como *Google, Intel, Motorola, Samsung, Ericsson, Texas Instrumente, Sprint Nextel, Toshiba o Vodafone*, que se comprometieron a publicar una parte importante de su propiedad intelectual como código abierto bajo la licencia *Apache v2.0.* Un punto clave de esta alianza era promover el diseño y difusión del sistema operativo *Android*.

En noviembre de ese mismo año se lanzó la primera versión de *Android SDK*. Un año después apareció el primer móvil con *Android* de la historia, el *T-Mobile G1*. En octubre se abrió *Android Market*, para la descarga de aplicaciones y *Google* liberó el código fuente de *Android* bajo licencia de código abierto *Apache* (*GPL v2* para el núcleo).

Aproximadamente año y medio después, en abril de 2009, *Google* lanzó la versión 1.5 del *SDK*, en la que incorporó nuevas características como el teclado en pantalla. A finales de ese año se lanzó la versión 2.0 y, durante 2010, las versiones 2.1, 2.2 y 2.3.

El uso *Android* fue aumentando entre los usuarios de *smartphones*, hasta que durante el año 2010 se consolidó como uno de los sistemas operativos para móviles más utilizados, con resultados cercanos a *iOS* e incluso superándolo en *EE.UU*.

El año 2011 se lanzó la primera versión de*Android* específica para *tablets*, la versión 3.x (*Honeycomb*), y otra versión que era compatible tanto con móviles como con *tablets*, la 4.0 (*Ice CreamSandwich*). Durante este año *Android* alcanzó una cuota de mercado superior al 50%, consolidándose así como la plataforma para móviles más importante.

En el año 2012 *Google* fusionó la descarga de las aplicaciones y de los contenidos en una única tienda de descargas online, *Google Play*, reemplazando esta a la antigua tienda virtual de descarga de aplicaciones *Android Market*. Ese año aparecieron las versiones 4.1 y 4.2 (*JellyBean*) de *Android*, que alcanza a finales de año una cuota de mercado del 70%.

Durante los siguientes años, la cuota de mercado de *Android* siguió aumentando, hasta llegar en marzo de 2017 a convertirse en el sistema operativo más utilizado, superando incluso a los sistemas operativos de ordenadores. Durante estos años, *Android* fue sacando una nueva versión de sus sistema operativo cada año: en el 2013 lanzó las versiones 4.3 y 4.4 (*KitKat*), en el 2014 la versión 5.0 (*Lollipop*), en el 2015 la versión 6.0 (*Marshamallow*) y en el 2017 la versión 7.0 (*AndroidNougat*).

3.3.1.2 Arquitectura de Android

Android está desarrollado en *C, C++, Java y XML*. Según se indica en el artículo "Arquitectura de la plataforma" de Android Developers [2] tiene una arquitectura basada en capas, como se puede observar en la Figura 2. A continuación, se explica brevemente cada una de ellas:

- Aplicaciones: Este nivel contiene las aplicaciones incluidas por defecto de *Android (teléfono, calendario, navegador, contactos, etc..)* y las que el usuario va añadiendo posteriormente. Todas estas aplicaciones utilizan los servicios, las *API* y librerías de los niveles inferiores.Normalmente, estas aplicaciones están desarrolladas en Java usando *Android SDK*, aunque es posible su desarrollo en *C/C++* usando *Android NDK*.
- *Framework* de Aplicaciones: Esel conjunto de herramientas de desarrollo de cualquier aplicación. Cualquier aplicación que se desarrolle para *Android* utiliza este conjunto de *APIs*, escritas en el lenguaje *Java*. Esta capa ha sido diseñada para simplificar la reutilización de componentes, y permite que cada aplicación pueda publicar sus capacidades para que otras aplicaciones puedan hacer uso de ellas. Algunas de estas *APIs* son:
    - *Activity Manager*: conjunto de *APIs* que gestiona el ciclo de vida de las aplicaciones en Android.
    - *Content Provider*: permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android.
    - *Location Manager*: gestión de información de localización.
    - *Notification Manager*: posibilita a las aplicaciones comunicar al usuario eventos que ocurran durante su ejecución.
    - *Package Manager*: permite obtener información sobre aplicaciones instaladas en el dispositivo.
    - *Resource Manager*: gestiona el acceso a recursos.
    - *Telephony Manager*: gestiona las funcionalidades del teléfono.
    - *View System*: proporciona un gran número de elementos para poder construir interfaces de usuario (*GUI*).
    - *Window Manager*: gestiona ventanas de las aplicaciones.
    - *XMPP Service*: permite utilizar éste protocolo de intercambio de mensajes basado en XML.

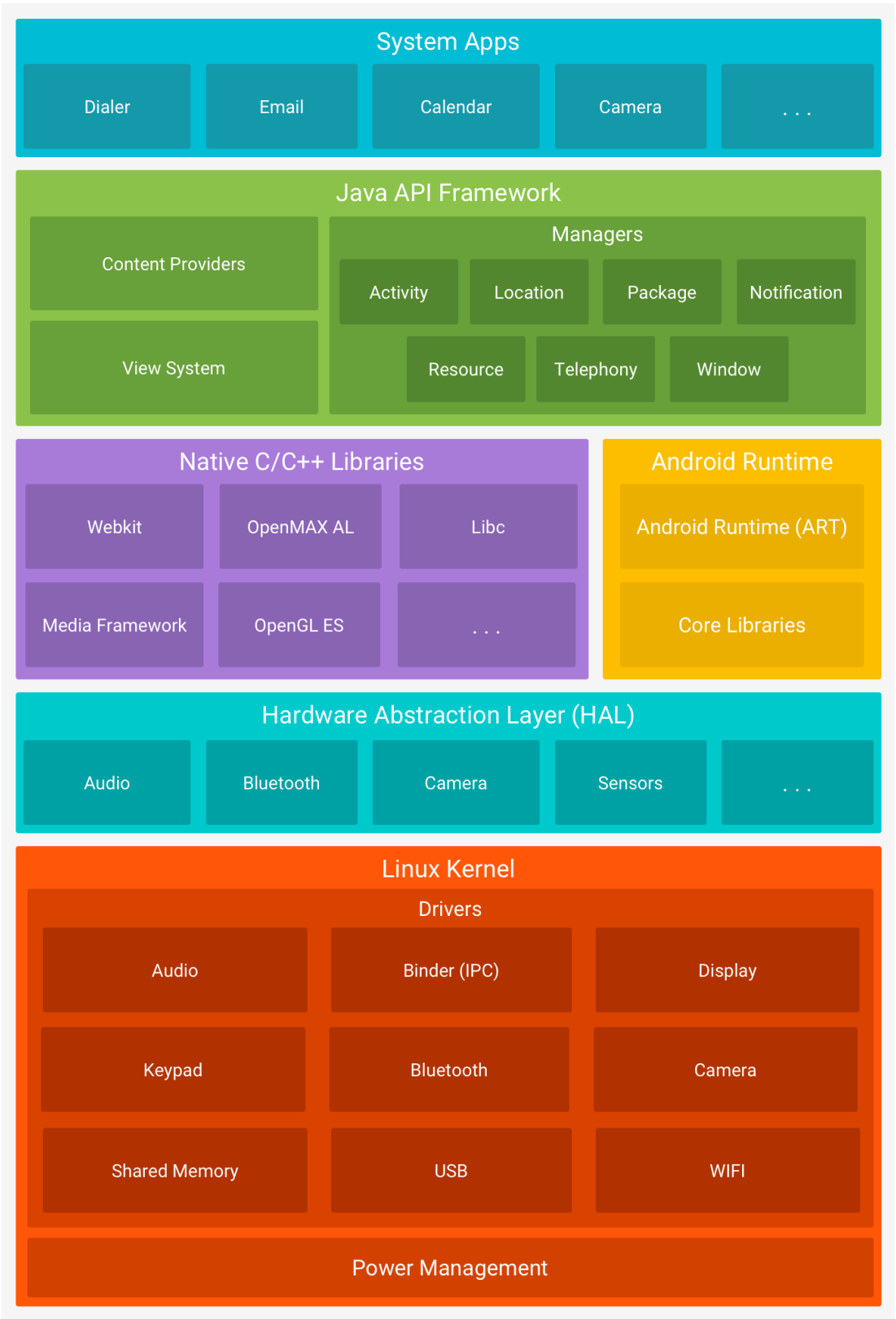

Figura 2. Arquitectura de Android (Android Developers, 2017)

- Librerías: Han sido escritas utilizando *C/C++* y proporcionan a *Android* la mayor parte de sus capacidades más características. Junto al núcleo basado en *Linux*, constituyen el corazón de *Android*. Algunas de estas librerías son:
    - *Libc*: cabeceras y funciones estándar de C.
    - *Surface Manager*: compone los diferentes sistemas de navegación y ventanas en la pantalla.
    - *OpenGL ES/SGL*: permite gáficos 3D y 2D.
    - *Media Framework*: posibilita la grabación y reproducción de diversos formatos de audio.
    - *SQLite*: permite la creación de bases de datos, y también su gestión.
    - *Free Type*: permite utilizar distintos tipos de fuentes.
    - *SSL*: posibilita utilizar de comunicaciones seguras.
    - *WebKit*: permite el empleo de aplicaciones web.
- AndroidRuntime: está formado por las *Core Libraries*, que son librerias con mulititud de clases *Java* y una máquina virtual. Como dispositivos en los que se ejecuta *Android* tienen poca memoria y un procesador limitado, no es posible poner una máquina virtual *Java* estándar. *Google* solucionó este problema creando una nueva máquina virtual, a la que llamó *Dalvik*, que responde mejor a las limitaciones de los dispositivos *Android*.
- Capa de abstracción de *Hardware* (*HAL*): proporciona interfaces estándares que permiten la interacción entre capacidades de *hardware* del dispositivo y *framework* de la *API* java de nivel más alto.
- Núcleo *Linux*: *Contiene* los *drivers* necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Está basado el *Linux 2.6* y permite la gestión de memoria, gestión de procesos, gestión de batería o instalar los drivers para poder utilizar el hardware, entre otras cosas.

3.3.2.3 Ventajas e inconvenientes de Android

Las principales ventajas de *Android* son:

- Software libre y multiplataforma, esto permite que cualquier persona pueda programar en este lenguaje.
- Gestión multitarea: es capaz de dejar en modo suspensión las aplicaciones que no se utilizan, y cerrarlas si llevan un tiempo determinado sin uso.
- Gran número de aplicaciones, en su mayoría de uso gratuito.

- Actualizaciones del sistema operativo. Poco a poco, van saliendo nuevas versiones de *Android* que permiten al usuario tener su sistema operativo actualizado constantemente, siempre y cuando el dispositivo soporte estas nuevas versiones.
- Interfaz intuitiva.

Sin embargo, *Android* también cuenta con una serie de desventajas:

- Vulnerabilidad: al ser un software libre es más vulnerable.
- La gestión multitarea comentada anteriormente también es una desventaja, ya que tener varias aplicaciones en segundo plano ralentiza el dispositivo y aumenta el consumo de batería.
- La gran variedad de versiones obliga a los desarrolladores a revisar que la aplicación funciona correctamente en varias versiones.

3.3.2 iOS

iOS es un sistema operativo móvil de la multinacional *Apple Inc.* desarrollado originalmente para el *iPhone* (teléfono de la marca *Apple*), aunque después se ha usado en dispositivos como el *iPod touch* (mp3 de la marca *Apple*) y el *iPad* (tableta de la marca *Apple*). Se encuentra únicamente en los dispositivos *Apple*, ya que son los únicos que lo pueden implementar y esto limita de manera notable su cuota de mercado.

Este sistema operativo, que supone el máximo competidor para *Android*, no permite trabajar utilizando *Adobe Flash* ni *Java*, lo que hace poco compatible el desarrollo paralelo de aplicaciones *iOS* y *Android* al no poderse reutilizar el código. En cambio sí que permite utilizar *HTML5*.

3.3.2.1 Historia y evolución de iOS

Según indica Cristian Rus en su artículo "*La evolución de iOS desde sus orígenes: una carrera para ser el mejor sistema operativo móvil de la historia*" [8], en 2007 se presentó el primer sistema operativo móvil de la marca *Apple*, que no soportaba aplicaciones creadas por terceros. Al año siguiente *Apple* lanzó la primera versión del *SKD*, y modificó el nombre de su sistema operativo a *iPhone OS*.

En la segunda versión de *iPhone OS* (*iPhone OS 2*), se incluyó la tienda virtual *App Store*, pudiendo así distribuir entre los usuarios las aplicaciones creadas por terceros.

En 2009 surgió la tercera versión de *iPhone OS*, que incluía nuevas funciones básicas como copiar, cortar y pegar. En esta versión también se añadió la posibilidad de ver la pantalla de forma horizontal.

En el 2010, Apple renombró sus sistema operativo con el nombre de *iOS* y lanzó la siguiente versión de su sistema operativo, *iOS 4*. Esta versión muestra una evolución importante de *Apple*, ya que se añadieron la posibilidad de cambiar el fondo de pantalla, las carpetas de apps y la multitarea, entre otras cosas.

La sexta versión de *iOS* salió al mercado en el año 2012. Sus dos principales mejoras fueron la introducción de *Siri*, un asistente virtual, y la posibilidad de realizar actualizaciones mediante *OTA*. Esto permitía poder actualizar el sistema directamente desde internet.

El sistema operativo ha seguido evolucionando, con sus pros y sus contras, hasta el momento. La última versión lanzada es *iOS 10*, presentada en septiembre de 2016. En esta versión se ha permitido a los desarrolladores tener acceso a Siri, su asistente virtual, y a las apps nativas del sistema operativo, como el teléfono. También han hecho posible borrar las apps preinstaladas de *Apple* que el usuario no vaya a utilizar, y por lo tanto no sea útil que estén en el dispositivo.

3.3.2.2 Arquitectura de iOS

Según indica *Apple* en el artículo "*About the iOs Technologies*" [4], este sistema operativo está basado en el *OS X de Apple*, que fue desarrollado a partir *Darwin BSD*. La arquitectura *iOS* está basada en capas, donde las capas más altas contienen los servicios y tecnologías más importantes para el desarrollo de aplicaciones, y las capas más bajas controlan los servicios básicos. Las capas de *iOS*, como se puede ver en la Figura 3, son:

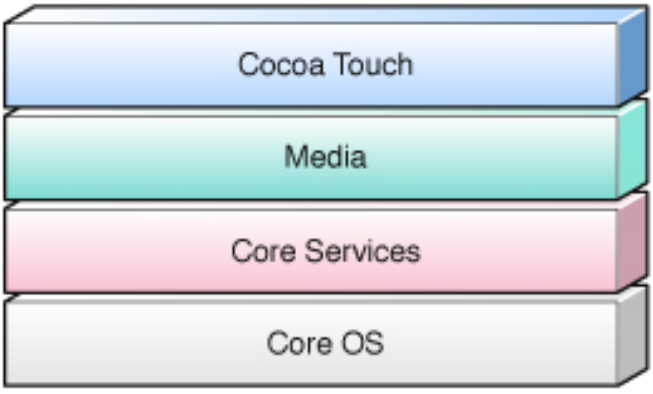

Figura 3. Arquitectura iOS (Carltics, 2015)

- *CocoaTouch*: Posee un conjunto de Frameworks para desarrollar aplicaciones. El lenguaje utilizado para programar estos framewoks, y por tanto la mayoría de las aplicaciones de *iOS*, es *Objetive-C*. Esta capa está formada por dos *Frameworks* fundamentales:
    - *UIKit*: define las clases necesarias para el desarrollo de interfaz de usuario.
    - *Foundation Framework*: define las clases necesarias para el acceso y manejo de objetos y servicios del sistema operativo.
- *Media*: una capa basada en la mezcla de lenguaje *C* y *Objetive-C* que provee los servicios gráficos y multimedia a la capa superior. Permite el uso de:
    - Tecnologías gráficas con funcionalidades avanzadas para interfaces gráficos.
    - Tecnologías de audio: La *API* de acceso a dispositivos de audio es *OpenAL*. Ofrece un soporte nativo para generación, grabación, mezcla y reproducción de audio (*Core Audio framework*), que también permite el acceso a la capacidad de vibración.
    - Tecnologías de video: permite la reproducción de vídeo a través de *Media Player framework*. Soporta *H.264* y *MPEG-4*.
- *CoreServices*: contiene los servicios fundamentales del sistema que usan todas las aplicaciones. Algunos de estos servicios son:
    - *Address Book*: posibilita el acceso a los contactos.
    - *Core Data*: permite gestionar los modelos de datos de una aplicación que sigan el patrón MVC.
    - *CoreFoundation*: posibilita la creación de interfaces en *C* para la gestión de datos (*array, strings, date, socket*, etc).
    - *CoreLocation*: permite obtener la localización del dispositivo utilizando el hardware disponible.
    - *Foundationframework*: posibilita el acceso a la funcionalidad del *CoreFoundation* mediante *wrappers* en *Objective-C*.
    - *Store kit framework*: permite realizar transacciones de pago a través de *i-tunes*, la tienda de contenidos multimedia de Apple.
    - *SQLite*: permite la creación de bases de datos *SQL*.
    - *XMLSupport*: permite la manipulación de ficheros *XML*.
- *Core OS*: esta capa es el núcleo del sistema e interactúa directamente con el hardware. Contiene características de bajo nivel como el sistema de ficheros, administración de memoria, seguridad, drivers del dispositivo, entre otras.

3.3.2.3 Ventajas e inconvenientes de iOS

Las principales ventajas de *iOS* son:

- Calidad de las aplicaciones, ya que antes de estar disponibles en *App Store*, pasan por números filtros manuales.
- Buena optimización de los recursos de los terminales que permite una mayor duración de la batería.
- Posibilidad de sincronización de todos los dispositivos *Apple*.
- Interfaz intuitiva.

Sin embargo, *iOS* también cuenta con una serie de desventajas:

- Precio elevado de sus terminales. Esta es una de las mayores desventajas de los terminales *iOS*. En comparación con otras marcas, tiene un precio muy elevado en comparación con las características de los dispositivos. Es necesario uno de estos terminales para comprobar el funcionamiento de la aplicación a desarrollar.
- El precio para poder desarrollar aplicaciones *iOS* también es elevado, ya que los desarrolladores tienen que abonar una cuota de 99$ anuales para poder acceder a las herramientas de desarrollo, y para poder subir las aplicaciones a la tienda.
- El único entorno de desarrollo disponible para este lenguaje de programación, *Xcode*, es propiedad de *Apple*. Para poder instalar este *IDE* es necesario tener un ordenador con el sistema operativo *Mac OS X* y una cuenta de *Apple*.
- El lenguaje de programación empleado es poco intuitivo y difícil de usar.
- No está permitido modificar la API de cualquier componente del *framework*, restando así libertad a los desarrolladores.

3.3.3 Windows 10 Mobile

Windows 10 Mobile es un sistema operativo móvil desarrollado por *Microsoft*, como sucesor de *Windows Phone*. El objetivo principal de este sistema operativo móvil es llevar la integración y unificación con su homólogo de *PC Windows 10*, y proporcionar una plataforma para los teléfonos inteligentes y pequeñas tabletas de 8 pulgadas de tamaño.

Aunque con menos aplicaciones disponibles que en *Android* e *iOS*, no faltan aquellas aplicaciones que se usan la mayor parte del tiempo para comunicarse, estar al día de la información, disfrutar de la buena música o entretenerse en las horas muertas. Además ofrece

una experiencia de usuario muy buena independientemente del tipo y gama de terminal en que se esté usando.

Uno de los mayores puntos a favor de este sistema operativo móvil es la funcionalidad *Continuum*, que permitirá convertir al teléfono en un PC portátil con un simple accesorio dedicado para conectarse a un monitor o *TV* y manejar periféricos tales como un ratón o un teclado.

3.3.3.1 Historia de Windows 10 Mobile

Microsoft lanzó en 1996 un sistema operativo desarrollado para dispositivos de capacidad limitada, como eran los teléfonos móviles disponibles en la época, denominado *Windows CE*. Esté sistema evolucionó durante los años siguientes hasta *Windows Phone 7* y su actualizaciones *Windows Phone 7.5* y *Windows Phone 7.8*, que fueron los últimos sistemas operativos creados a base de Windows CE.

En el año 2003 lanzaron el sistema operativo específico para móviles: *Windows Mobile 2003*, que fue el primer lanzamiento bajo el nombre Windows Mobile. Este sistema operativo fue seguido en el año 2005 por *Windows Mobile 5.0*, en el 2007 por *Windows Mobile 6.0* y en el 2010 por *Windows Phone 7.*

EL sistema operativo *Windows Phone*, a diferencia de los anteriores que estaban enfocados en el mercado empresarial, está destinado al mercado de consumo, compitiendo así con *Android* e *iOS*.

En el año 2015 *Microsoft* lanzó Windows 10 Mobile, unificando así todos sus sistemas operáticos en uno solo. Este sistema operativo está disponible para todo tipo de plataformas, ya sean *smartphone*, *tablets* u ordenadores.

3.3.3.2 Arquitectura de Windows Phone

Windows Phone, precursor de *Windows 10 Mobile*, tiene una arquitectura bastante diferente a los sistemas operativos anteriores. Su arquitectura, según indica Carlos Luis Murillo en su artículo "*Winsows phone 8*" [6], es la siguiente:

Figura 4. Arquitectura de Windows Phone (Carlos Luis Murillo, 2015)

- *Run Time – Interfaz de Usuario*: está desarrollado en código *C#* siguiendo un modelo *sandbox* que permite el desarrollo de aplicaciones seguras fácilmente. .Esta capa está basada en las plataformas *Silverlight* y *XNA*, ya existentes previamente en *Windows*, por lo que las adaptaciones para *Windows Phone* de aplicaciones que han sido previamente desarrolladas en estos entornos son mínimas.
- *Herramientas*: basado en las herramientas de *Microsoft Visual Studio* y *ExpressionBlend*. Las herramientas necesarias para desarrollar una aplicación de *Windows Phone* son:
    - *Visual Studio*: este entorno de desarrollo permite el desarrollo de aplicaciones *Silverlight* y *XNA*.
    - *ExpressionBlend*: diseño de interfaces gráficas basadas en *XAML*.
    - *Windows PhoneEmulator*: permite realizar la depuración y testeo de las aplicaciones desarrolladas.
    - *XNA Game Studio*: ofrece las funcionalidades específicas para juegos.
- *Servicios en la nube*: facilita la integración de las aplicaciones con los servicios web.
- *Servicios Portal Desarrollador*: Permite los servicios relacionados con la tienda de aplicaciones, como registro, validación del usuario, facturación o publicación y gestión de actualizaciones.

3.3.4 Elección

Ninguno de los sistemas operativos mencionados anteriormente tiene ventajas claras con respecto a sus competidores, todos tienen sus pros y sus contras.

Una de las motivaciones principales de esta aplicación es permitir el uso de la misma por el mayor número de alumnos y docentes, por lo que parece lógico que sea desarrollada para el sistema operativo que abarque la mayor cuota de mercado. Como se ha dicho anteriormente, el sistema operativo para móviles más usado en la actualidad es claramente *Android*, por lo que ese será el sistema operativo en el que se desarrollará la aplicación.

Otro motivo de peso para desarrollar está aplicación en *Android* y no en *iOS*, su principal competidor, es que para desarrollar en*iOS* hace falta disponer de un terminal *Mac* (*PCs* de la marca *Apple*) y de un *Iphone* (terminal móvil de la marca *Apple*), ambos de precio elevado. Sin embargo, para desarrollar una aplicación para *Android* únicamente hace falta cualquier *PC*, y un móvil *Android*, que se puede encontrar a un precio mucho más asequible en el mercado.

3.4 Entornos Integrados de Desarrollo para Android

Este apartado se expondrá una breve explicación de los principales IDE (Entornos Integrados de Desarrollo) disponibles para desarrollar aplicaciones en *Android*, y se justificará el entorno de desarrollo elegido para desarrollar esta aplicación.

3.4.1 Eclipse

Eclipse es un entorno de desarrollo de código abierto y gratuito, cuyo diseño sigue un patrón de actualización basado en *plugins*. No fue concebido para ser utilizado con un único lenguaje de programación, sino que es compatible con una gran variedad de lenguajes. Las principales características de *Eclipse* son:

- Gestión de proyectos: El desarrollo sobre *Eclipse* se basa en proyectos.
- Depurador de código: Proporciona de forma gráfica una opción de mejorar nuestros proyectos. Tiene una perspectiva dedicada a la depuración donde se puede realizar y supervisar dicha tarea.
- Acceso a Base de Datos: Permite conectarse a distintos gestores de bases de datos y consultar tablas y datos. Aunque para ello hace falta un *plugin* de terceros.

- Perspectivas, editores y vistas: En Eclipse el concepto de trabajo se basa en las perspectivas, que son una configuración de ventanas y editores que nos permiten trabajar en un determinado entorno de trabajo de forma óptima.
- Colección de *plugins*: Hay una gran cantidad de *plugins* disponibles, algunos desarrollados por Eclipse y otros por terceros.
- Construcción de paquetes .apk mediante el uso de *ANT*: no permite importar proyectos creados en *Android Studio*, que a diferencia de *Eclipse* y *Netbeans*, utiliza *Gradle*.
- Para poder desarrollar en *Android*, hace falta descargar *plugins* secundarios.

3.4.2 Netbeans

Netbeans es un entorno de desarrollo de código abierto y gratuito. Al igual que *Eclipse*, está pensado para poder utilizarlo con más de un lenguaje de programación. Sus principales características son:

- Asistentes y Gestor de Proyectos: El desarrollo sobre *Netbeans* se basa en proyectos. Tiene asistentes para configuración de distintos proyectos y selección de *frameworks*.
- Depurador de Código: el depurador permite, entre otras cosas, monitorizar en tiempo real los valores de las propiedades y variables.
- Acceso a Base de Datos y *Plugins*: Permite conectarse a distintos gestores de bases de datos y consultar tablas y datos.
- Construcción de paquetes *.apk* mediante el uso de *ANT*: no permite importar proyectos creados en *Android Studio*, que a diferencia de Eclipse y *Netbeans*, utiliza *Gradle*.
- Para poder desarrollar en *Android*, hace falta descargar *plugins* secundarios.

3.4.3 IntelliJ IDEA

IntellineJ IDEA tiene dos distribuciones disponibles, una gratuita y de código abierto denominada *IntelliJ IDEA Community Edition* y otra de pago denominada *IntelliJ IDEA Ultimate*. Las características principales de la versión gratuita son:

- Editor de Código: El editor resalta advertencias y errores inmediatamente.Así mismo, permite finalización inteligente de código y autocompletado sofisticado y sensible.

- Herramientas integradas de *Android*: Posee un diseñador de interfaz de usuario que permite arrastrar y soltar elementos. También posee filtros de búsqueda personalizados.
- Aumento de productividad: Tiene soporte para *Maven* y *Gradle*, y un control de versiones compatible con *Gib*, *GibHub y SVN*, entre otros.
- Lenguajes soportados: Soporta varios lenguajes basados en *JVM*, como *Java*.

La versión de pago, entre otras cosas, añade un mayor soporte de lenguajes de programación, como *PHP*, *SQL*, *MySQL*.

3.4.4 Android Studio

Android studio es, según Google, la IDE oficial para desarrollar aplicaciones para *Android*. A diferencia de *Eclipse* o *Netbeans*, está desarrollado única y exclusivamente para la programación de aplicaciones para dispositivos *Android*, por lo que no es necesario añadir ningún *plugin* para su uso con *Android*. Sus principales características son:

- Gestión de proyectos: El desarrollo sobre *Android Studio* se basa en proyectos.
- Acceso a Base de Datos: Permite conectarse a un gestor de bases de datos y consultar tablas y datos almacenados en la base de datos a través de dicho gestor.
- Editor de diseño: muestra una vista previa de los cambios realizados directamente en el archivo *xml*.
- Permite la creación de nuevos módulos dentro de un mismo proyecto, sin necesidad de estar cambiando de espacio de trabajo para el manejo de proyectos, algo habitual en *Eclipse*.
- Construcción de paquetes *.apk* mediante el uso de *gradle*: Permite compilar desde línea de comandos y da una mayor facilidad para la creación de diferentes versiones de una misma aplicación. Si permite importar proyectos creados en *Eclipse* o *Netbeans*, que a diferencia de *Android Studio*, utilizan *ANT*.
- Simplemente descargando *Android Studio* ya se dispone de todas las herramientas necesarias para el desarrollo de aplicaciones para la plataforma *Android*.
- Ofrece plantillas de código con las características más comunes.

3.4.5 AIDE

AIDE permite programar y compilar los proyectos directamente en un dispositivo *Android*. Sus principales características son:

- Editor de Código: Permite autocompletado inteligente de código.
- Posibilidad de programar aplicaciones directamente en un dispositivo móvil.
- Comprobación de errores en tiempo real.

3.4.6 Elección

Tras analizar los principales entornos de desarrollo de aplicaciones para la plataforma *Android*, se puede comprobar que los IDEs *Eclipse* y *Netbeans* son bastante similares en cuanto a características y facilidad o dificultad de instalación y de uso.

IntelliJ IDEA a priori parece una buena opción, pero la versión gratuita tiene algunas limitaciones que podrían interferir con el desarrollo de la aplicación planteada en este documento.

AIDE es una buena opción si se quiere realizar la aplicación directamente en un dispositivo móvil, o para realizar los últimos pasos del desarrollo de esta. Pero realizar una aplicación entera en un dispositivo *Android* probablemente sea una labor algo complicada.

Android Studio tiene alguna funcionalidad menos que sus competidores, pero como se ha expuesto anteriormente también tiene ventajas con respecto a los otros *IDEs* analizados en esta sección. Por lo que finalmente la elección para desarrollar esta aplicación será *Android Studio*.

3.5 Herramientas de control de versiones

El control de versiones es una herramienta que nos permite llevar un control sobre los diversos cambios que se llevan a cabo sobre un fichero o conjunto de ficheros a lo largo del tiempo. A la hora de desarrollar una aplicación, esta herramienta es muy útil y aconsejable, ya que nos permite llevar un control sobre los distintos cambios que se llevan a cabo en los ficheros que forman parte de esta. Estas herramientas, entre otras cosas, nos permiten:

- Recuperar un código anterior.
- Comparar diversas versiones del código.
- Tener una copia de seguridad de nuestros ficheros, ya sea en local o en un servidor.
- Trabajar en paralelo con otros desarrolladores sobre el mismo proyecto. Esto solo es posible con un control de versiones centralizado o distribuido.

Según indicant Scott Chavon y Ben Straub en su manual "*Pro Git*" [20], los sistemas de control de versiones se diferencian en tres grupos distintos en función del lugar en el que se almacenen los ficheros en los que se registran los cambios de la aplicación: local, centralizado y distribuido.

3.5.1 Sistemas de control de versiones locales

Mucha gente usa el "método casero de control de versiones" de copiar los ficheros en otro directorio, ya que es una forma simple de tener una copia de seguridad de tus ficheros. Este método, aparte de simple es muy propenso a errores, ya que por accidente puedes confundirte de directorio y no saber dónde has guardado la copia, o sobrescribir archivos que no querías.

Los sistemas de control de versiones locales se crearon para intentar mejorar este sistema tradicional de copia de ficheros, creando una base de datos en la que se lleva el registro de todos los cambios realizados sobre los ficheros.

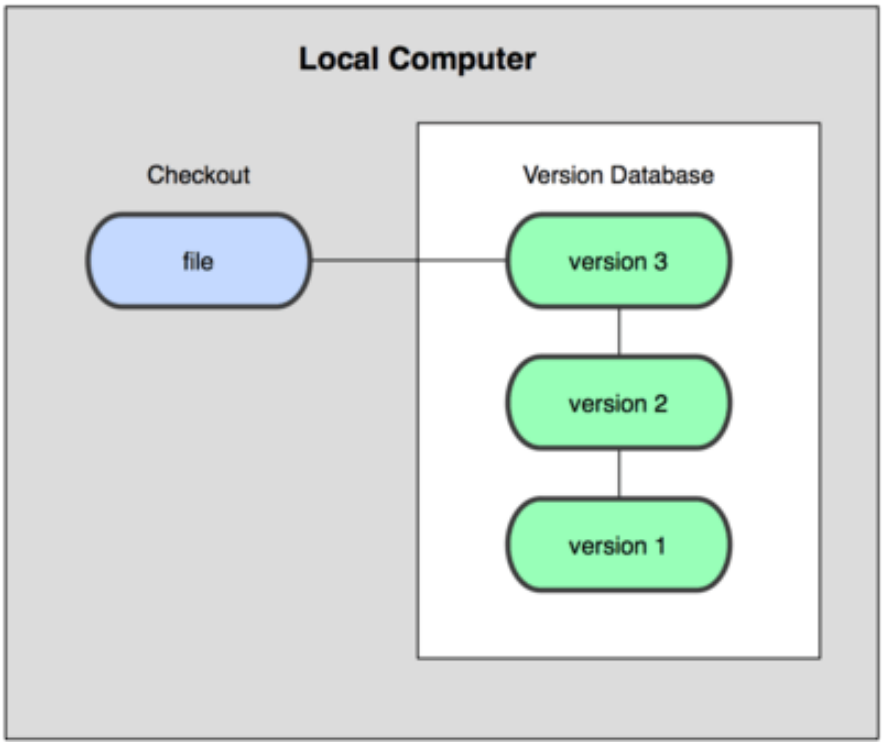

Figura 5. Sistema de control de versiones local (Scott Chacon y Ben Straub, 2014)

Un ejemplo es el sistema *RCS*. Esta herramienta guarda las diferencias entre una versión de un fichero y otra (parches) en un formato especial, de tal manera que puede recrear en

cualquier momento cualquier versión anterior de un determinado archivo sumando los distintos parches.

Con este sistema es versiones es conveniente realizar copias de seguridad en algún otro dispositivo por si el disco duro local se corrompe.

3.5.2 Sistemas de control de versiones centralizados

Los sistemas de control de versiones centralizados tienen un único repositorio, situado en un servidor. Este sistema permite la colaboración entre varios desarrolladores, ya que varias personas pueden descargarse los archivos desde el servidor. También permite a los administradores llevar el registro de las tareas asignadas a cada cliente más cómodamente.

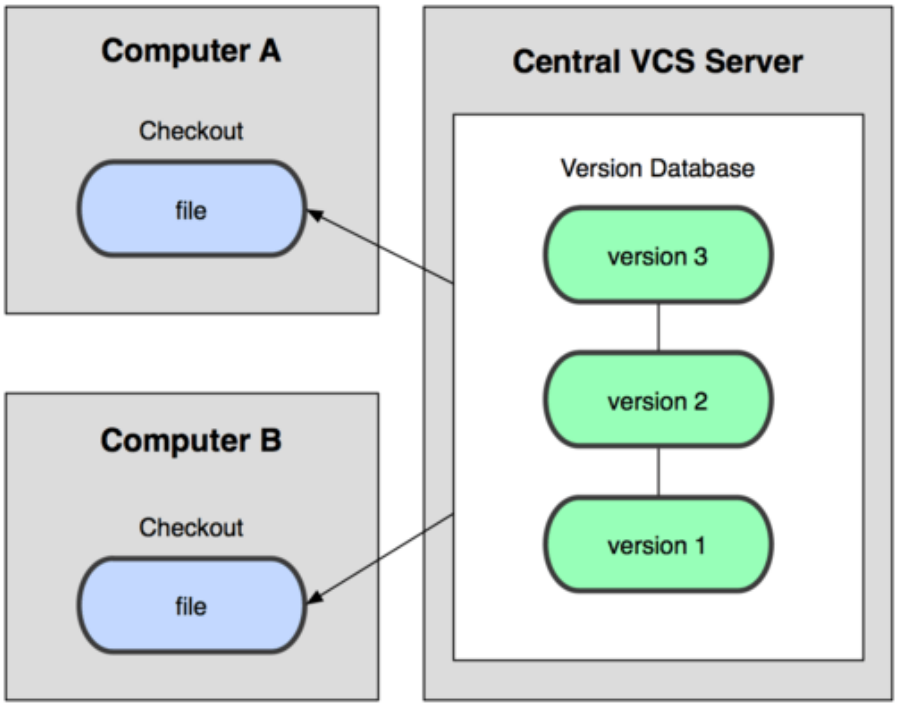

Figura 6. Sistema de control de versiones centralizado (Scott Chacon y Ben Straub, 2014)

Pero no todo en este sistema son beneficios, también tiene alguna desventaja. La principal desventaja es que al estar todo centralizado en un único servidor, el sistema tiene un único punto de fallo:

- Si el servidor se cae, imposibilita el acceso al repositorio para todos los desarrolladores, imposibilitando así tanto la descarga de los archivos del servidor, como guardar los cambios realizados en los archivos locales.

- Si el disco duro del servidor en el que se encuentra la base de datos se corrompe y no se tienen más copias de seguridad, se pierde toda la información.

Otra desventaja de este sistema es que sin acceso a internet no se puede acceder al servidor, y por lo tanto es imposible descargar o subir modificaciones al repositorio.

3.5.3 Sistemas de control de versiones distribuidos

Los sistemas de control de versiones distribuidos tienen varios repositorios: uno en el servidor y otro local en el ordenador de cada desarrollador. Cada vez que se descarga el repositorio a un ordenador local es como si se realizase una copia de seguridad del mismo, ya que se replica la información. Esto permite que si hay alguna pérdida de datos en el servidor, cualquiera de los repositorios locales de los desarrolladores pueda replicar su contenido para restaurar el del servidor.

Este sistema también permite a los desarrolladores subir al control de versiones local varias versiones del código antes de subirlo todo al repositorio del servidor.

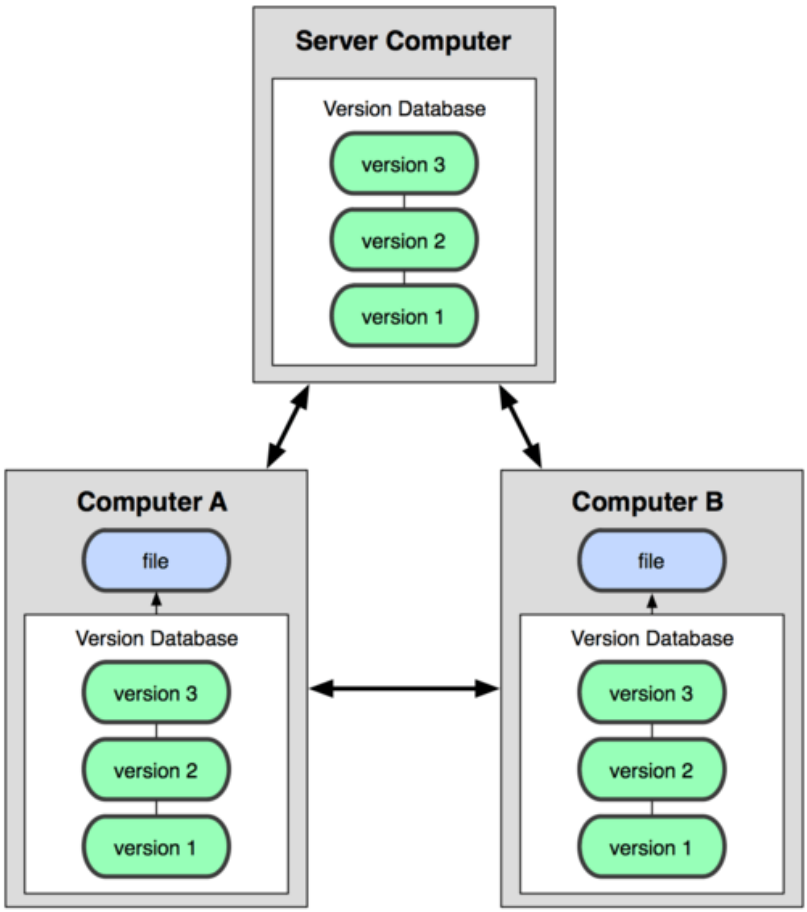

Figura 7. Sistema de control de versiones distribuido (Scott Chacon y Ben Straub, 2014)

3.5.4 Elección

Teniendo en cuenta la complejidad añadida que supone usar un control de versiones central o distribuido, ya que hay que disponer de un servidor y configurar el sistema elegido, y la dificultad de que se estropee un disco duro ssd nuevo como con el que se ha contado a la hora de realizar este proyecto, se ha elegido realizar el control de versiones de forma local.

Capítulo 4. Desarrollo del proyecto

4.1 Introducción a Android

A la hora de desarrollar una aplicación para este sistema operativo, hay que tener una serie de conocimientos básicos sobre el mismo. En este apartado se expondrán los componentes básicos de una aplicación *Android* y la estructura de directorios que ha de tener un proyecto desarrollado en este sistema operativo.

4.1.1 Componentes de una aplicación Android

Hay varios componentes disponibles para una aplicación *Android*. Los principales son los siguientes:

- *Actividad* (*Activity*): cada una de las pantallas de la aplicación es una actividad. Cada aplicación suele estar formada por varias actividades, que en conjunto crean la interfaz de usuario.
- *Fragmento (Fragment):* porción de interfaz de usuario que puede añadirse o quitarse de la interfaz, independientemente del resto de elementos de la actividad.
- *Intención* (*Intent*): representa la intención de realizar una acción. Se utiliza para lanzar una actividad o servicio. También permite comunicarse con un servicio o intercambiar información entre componentes lanzados.
- *Layout*: estos elementos, descendientes de la clase *View*, agrupan un conjunto de vistas de una determinada forma. Hay diferentes tipos de *Layout* en función de cómo se quieran organizar las vistas. Algunos de estos son:
    - *Lineal*: organiza sus hijos en una única fila, que puede ser vertical u horizontal.
    - *Relativo*: permite especificar la posición de los objetos hijo en relación al padre o a otros objetos hijo.
    - *Web*: permite mostrar fácilmente páginas web.
    - *Listview*: define una columna con desplazamiento, es decir, una lista.
    - *Gridview*: define una cuadrícula de filas y columnas con desplazamiento.
- *Proveedor de contenidos* (*Content Provider*): permite a las aplicaciones compartir datos con otras aplicaciones sin comprometer la seguridad de su sistema de ficheros.

- *Servicio* (*Service*): proceso que se ejecuta como actividad secundaria, sin necesidad de interacción con el usuario.
- *Vista* (*View*): estos elementos, descendientes de la clase View, componen la interfaz de usuario de una aplicación. Habitualmente se definen en un fichero XML, aunque también pueden ser definidos en un .java.

4.1.2 Estructura de directorios de Android

Android tiene una estricta estructura de directorios, que a su vez permite tener la documentación de cualquier proyecto ordenada siempre del mismo modo. Esta estructura es la siguiente:

- *AndroidManifest.xml*: describe los componentes de la aplicación, así como sus características principales.
- *Assets/*: directorio que contiene ficheros adicionales para la aplicación, como pueden ser los ficheros de datos.
- *Bin/*: directorio donde se compila el código y se genera el .apk (fichero comprimido de la aplicación para instalar).
- *Gen/*: directorio con código generado automáticamente por el entorno de desarrollo al construir la aplicación.
- *Libs/*: directorio con las bibliotecas necesarias para el funcionamiento del proyecto.
- *Raw/*: directorio que contiene ficheros adicionales para la aplicación que no están en formato XML.
- *Res/*: directorio que contiene los recursos de la aplicación. Está formado por varios directorios, cada uno de los cuales tiene una función.
    - *Anim/* y *animator/*: directorio para ficheros *XML* relacionados con animaciones.
    - *Drawable/*: directorio para imágenes (*jpg* y *png*) y descripciones de imágenes (*XML*). Los nombres de los archivos almacenados en esta carpeta solo pueden contener números y letras de la "*a*" a la "*z*" en minúscula, excluyendo la "*ñ*".
    - *Layout/:* directorio para los ficheros que describen la interfaz de usuario de la aplicación.
    - *Menu/*: directorio para los ficheros XML con los menús de cada actividad.

- *Values/*: directorio para los ficheros que contienen colecciones de recursos. Algunos de estos ficheros son:
  - *Colors.xml*: fichero que define los distintos colores usados en la aplicación.
  - *Strings.xml*: fichero que define las distintas cadenas (textos) usados en la aplicación. Se puede definir uno distinto para cada idioma.
  - *Styles.xml*: fichero que define los distintos estilos usados en la aplicación.
- *XML/*: directorio para otros ficheros *XML* requeridos por la aplicación.

- *Src/:* directorio para los ficheros principales de la aplicación.

4.2 Análisis de la aplicación

Teniendo en cuenta los requisitos funcionales con los que debe contar la aplicación a desarrollar, se decidió diseñar una aplicación con dos pestañas en la parte superior:

- **Asignaturas:** está es la pantalla principal que se muestra al abrir la aplicación. En ella aparecerá el acceso a cada una de las asignaturas disponibles en la aplicación. Al pinchar sobre la asignatura deseada, se abrirá una nueva ventana con dos pestañas en la parte superior.
  - **Temas:** en esta pantalla se mostrarán con los temas de la asignatura y la opción de ver una lista con los acrónimos que se emplean en esta. Al seleccionar el tema indicado se abrirá una nueva pantalla con dos o tres pestañas en la parte superior:
    - **Teoría:** muestra toda la teoría relacionada con el tema indicado.
    - **Problemas:** muestra los problemas relacionados con el tema indicado. En algunos de estos problemas se mostrará la solución al mismo.
    - **Prácticas:** mostrará en el enunciado de la práctica. Esta pestaña solo estará disponible en aquellos temas que tengan alguna práctica asociada a los mismos.

    Si alguna de estas secciones se hace demasiado extensa, y por tanto sea incómodo navegar por la página y encontrar la información requerida, se desplegará en una página nueva, con varias pestañas separando cada parte de la información.

Si en vez de seleccionar un tema, se selecciona la lista de acónimmos, se abrirá una nueva ventana con una única pestaña:

- **Acrónimos:** Muestra una lista con los acrónimos utilizados en esa asignatura. Separa los acrónimos según los temas en los que se han utilizado.

- **Guía:** en está pestaña se mostrará la guía docente de la asignatura.

- **Acerca de:** en esta pantalla se muestra información sobre la aplicación móvil.

Con el diseño aquí expuesto se pretende que la aplicación sea lo más intuitiva posible. Se espera que esta característica, junto con su funcionalidad y la información que se va a presentar en ella, formen una aplicación que realmente sea útil para los alumnos de las asignaturas incluidas en la *app*, así como para cualquiera que quiera adquirir o consultar conocimientos sobre las mismas.

4.3 Desarrollo de la aplicación

Cada pantalla principal de la aplicación es una *activity*. Estas tienen en su parte superior un *toolbar* con varias "*pestañas*" que permiten decidir que fragment (*fragment*) se prefiere ver en la parte inferior de la pantalla.

4.3.1 Desarrollo de cada activity

Todo lo *activity* está formado por un *toolbar* con varias "pestañas" (*tabs*). De forma que el código principal de toda *activity*es el mismo, y se diferencian en los *fragments* que muestra cada una de estas pestañas. Este código es el siguiente:

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

 //Coge la vista del toolbar
 setContentView(R.layout.activity_toolbar);

 //Añade a la variable 'toolbar' el toolbar definido en la vista
 Toolbar toolbar = (Toolbar) findViewById( R.id.toolbar);
 setSupportActionBar(toolbar);

 //Se añade un ViewPager a cada pestaña(tab)
 ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);
 setupViewPager(viewPager);

 //Las pestañas (tabs) se añaden al toolbar
 TabLayout tabs = (TabLayout) findViewById(R.id.tabs);
 tabs.setupWithViewPager(viewPager);
}
```

El código que se acaba de mostrar es el esqueleto principal del *toolbar*, pero hay que rellenarlo con las distintas pestañas que dan acceso a la información que se quiere mostrar en cada una, para ello:

```
// Se añaden los fragmentos a las pestañas
private void setupViewPager(ViewPager viewPager) {

 //Se define el adaptador
 Adapter adapter = new
 Adapter(getSupportFragmentManager());

 //Se añaden nuevos fragmentos
 adapter.addFragment(new fragmentoAMostrar(), "nombrePestaña");
 adapter.addFragment(new fragmentoAMostrar2(), "nombrePestaña2");

 //Se incluye el adaptador con los fragmentos a la vista de la página
 viewPager.setAdapter(adapter);
}
```

Como se puede observar, para añadir los fragmentos se hace uso de un *adapter*, el *adapter* usado en este caso es el siguiente:

```
static class Adapter extends FragmentPagerAdapter {

 //Se definen las variables.
 private final List<Fragment>mFragmentList = new ArrayList<>();
 private final List<String>mFragmentTitleList = new ArrayList<>();

 public Adapter(FragmentManager manager) {
  super(manager);
 }

 @Override
 public Fragment getItem(int position) {
  return mFragmentList.get(position);
 }

 @Override
 //Indica cuantas pestañas (tabs) hay en el toolbar
 public int getCount() {
  return mFragmentList.size();
 }

 //Se añade el nuevo fragmento
 public void addFragment(Fragment fragment, String title) {
  mFragmentList.add(fragment);
  mFragmentTitleList.add(title);
 }

 @Override
 //Indica que pestaña (tab) está seleccionada
 public CharSequence getPageTitle(int position) {
  return mFragmentTitleList.get(position);
```

```
  }
}
```

El diseño básico del *toolbar* está definido en un XML, ya que es el mismo para cada una de las diferentes *activity*. Este XML es el siguiente:

```
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/activity_subject_index"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!--Barra principal, muestra el nombre de la activity -->
    <android.support.design.widget.AppBarLayout
        android:id="@+id/appbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimaryDark"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

        <!--Barra secundaria, permite deslizarse entre varios
        fragments -->
        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimaryDark"
            app:layout_scrollFlags="scroll|enterAlways"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Dark" />

        <!--Pestañas (tabs) que se añaden a la barra secundaría -->
        <android.support.design.widget.TabLayout
            android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

    </android.support.design.widget.AppBarLayout>

    <!--ViewPager -->
    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        />

</android.support.design.widget.CoordinatorLayout>
```

4.3.2 Desarrollo de los fragmentos

Se puede diferenciar entre dos tipos de fragmentos principales: los destinados a la elección de la información deseada, y aquellos que muestran esta información.

4.3.2.1 Fragmentos destinados a elección de opciones

En la aplicación hay dos fragmentos destinados a la elección de la información deseada. El primero de ellos muestra un *TextView* con el texto "*Pulsa para acceder a los contenidos*" seguido de un *Button* por cada asignatura, que permite accede a la información relacionada con dicha asignatura. Estos botones están descritos en XML, y en el fichero .java se especifica la acción asociada a cada uno. El texto XML en el que se definen los botones es el siguiente:

```
<LinearLayout
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorIndigo600"
    android:gravity="center_horizontal">

    <TextView
        android:paddingTop="@dimen/activity_horizontal_marginx4"
        android:paddingBottom="@dimen/activity_horizontal_margin"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:textAlignment="center"
        android:background="@color/colorIndigo600"
        android:text="Pulse para acceder a los contenidos"
        android:textColor="@color/colorIndigo50"/>

    <Button
        android:id="@+id/botonAsignatura1"
        android:paddingTop="@dimen/activity_horizontal_margin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/colorIndigo600"
        android:textColor="@color/colorIndigo50"
        android:text="NombreAsignatura1"
        android:textSize="30pt"/>
    <View style="@style/Separator" />
    <Button
        android:id="@+id/botonASignatura2"
        android:paddingTop="@dimen/activity_horizontal_margin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/colorIndigo600"
        android:textColor="@color/colorIndigo50"
        android:text="NombreAsignatura2"
        android:textSize="30pt"/>
    <View style="@style/Separator" />
</LinearLayout>
```

El otro fragmento destinado a la elección de la información deseada, muestra las distintas opciones en una lista. La definición de la lista, así como de cada una de sus partes se

ha realizado en dos ficheros *XML*. El fichero *XML* en el que se define la lista está formado por una vista relativa, dentro de la cual se define el *ListView*, su contenido es el siguiente:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.redes.protocolos.tfg.LevelsContentFragment"
    >

    <ListView
        android:id="@+id/mi_lista"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="@dimen/activity_vertical_margin"/>

</RelativeLayout>
```

El fichero XML en el que definen las distintas partes de cada fila de la lista está formada por un una vista lineal horizontal, en la que la primera parte es una imágent (ImageView) y la segunda parte es una nueva vista lineal, en este caso horizontal, en la que se define primero un TextView que contendrá en número del tema, y luego otro TextView que contendrá el nombre del título del tema.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <!--Imagen que se muestra en cada fila de la lista -->
    <ImageView
        android:id="@+id/icon"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:padding="5dp" />

    <LinearLayout android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <!--Texto principal de la lista. Muestra el número del tema-->
        <TextView
            android:id="@+id/texto_principal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Medium Text"

android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_marginLeft="10dp"
```

```
            android:layout_marginTop="5dp"
            android:padding="2dp"
            android:textColor="@color/colorIndigo700" />

        <!--Texto secundario de la lista.Muestra el título del tema-->
        <TextView
            android:id="@+id/texto_secundario"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="TextView"
            android:layout_marginLeft="10dp"
            android:textColor="@color/colorIndigo500"/>

    </LinearLayout>
</LinearLayout>
```

Una vez definido cómo va a ser cada componente de la lista, se puede proceder a generar esta lista en el fragmento en el que se quiere mostrar. El código necesario para ello es el siguiente:

```
//Se definen las variables.
private String descripcion[]=new String[]{
 "Descripción fila 1",
 "Descripción fila 2",
 "…"};
private String tema[]=new String[]{
 "Título fila 1",
 "Título fila  2",
 "…"};
private Integer[] imgid={
 R.drawable.imagenfila1,
 R.drawable.imagenfila2,
 R.drawable.siguientesimagenes};
private ListView lista;



   @Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {

 //Se infla la vista del fragment.
 View v =inflater.inflate(R.layout.fragment_levels, container, false);

 //Se crea el adaptador de la lista
 listAdapter adapter=new listAdapter(getActivity(),tema, descripcion,
 imgid);
 //Se añade la vista de la lista a la lista
 lista=(ListView)v.findViewById(R.id.mi_lista);
 //Se une el adaptador a la lista
 lista.setAdapter(adapter);

 //Método que se llama al pulsar alguna fila de la lista.
 lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {

  @Override
  public void onItemClick(AdapterView<?> parent, View view,
```

```
        int position, long id) {

    switch (position) {
     case 0:
      //Se ha pulsado la primera fila de la lista.
      Intent intent1 = new Intent(getActivity(),
        activityDestinofila1.class);
      startActivity(intent1);
     break;
     case 1:
      //Se ha pulsado la segunda fila de la lista.
      Intent intent2 = new Intent(getActivity(),
        activityDestinofila2.class);
      startActivity(intent2);
     break;
     default:
     break;
    }
   }
  });

  return v;
}
```

En este ejemplo se ha hecho una lista con dos filas, para añadir nuevas filas a la lista habría que poner el título (*String tema[]*), la descripción (*String descripcion[]*) y la imagen (*Integer[] imgid*) de cada una de las nuevas filas, y añadir un nuevo *case* al *switch* con la acción asociada a cada fila.

4.3.2.2 Fragmentos destinados a mostrar información

Los fragmentos destinados a mostrar la información constan de varios *TextView* e *ImageView* definidos en *XML*, del siguiente modo:

```
<!--Muestra un texto -->
<TextView
     style="@style/Body"
     android:id="@+id/idText1"
     android:text="@string/Texto1"/>

<!--Muestra una imagen -->
<ImageView
     style="@style/Imagen1"
     android:src="@drawable/Imagen1" />
```

Los *TextView* muestran el texto, y pueden tener un estilo u otro en función de que sean el título principal, títulos secundarios o el desarrollo de la idea.

Con *ImageView* se muestra una imagen. Todas tienen el mismo estilo, definido en el fichero específico para ello (*Styles.xml*). Las imágenes se han editado haciéndolas menos pesadas, consiguiendo con ello una aplicación más fluida y con un menor tiempo de carga.

4.4 Diagrama de flujo

Al iniciar la aplicación, el proceder va a ser siempre el mismo. Los pasos posibles a seguir se muestran en la Figura 8:

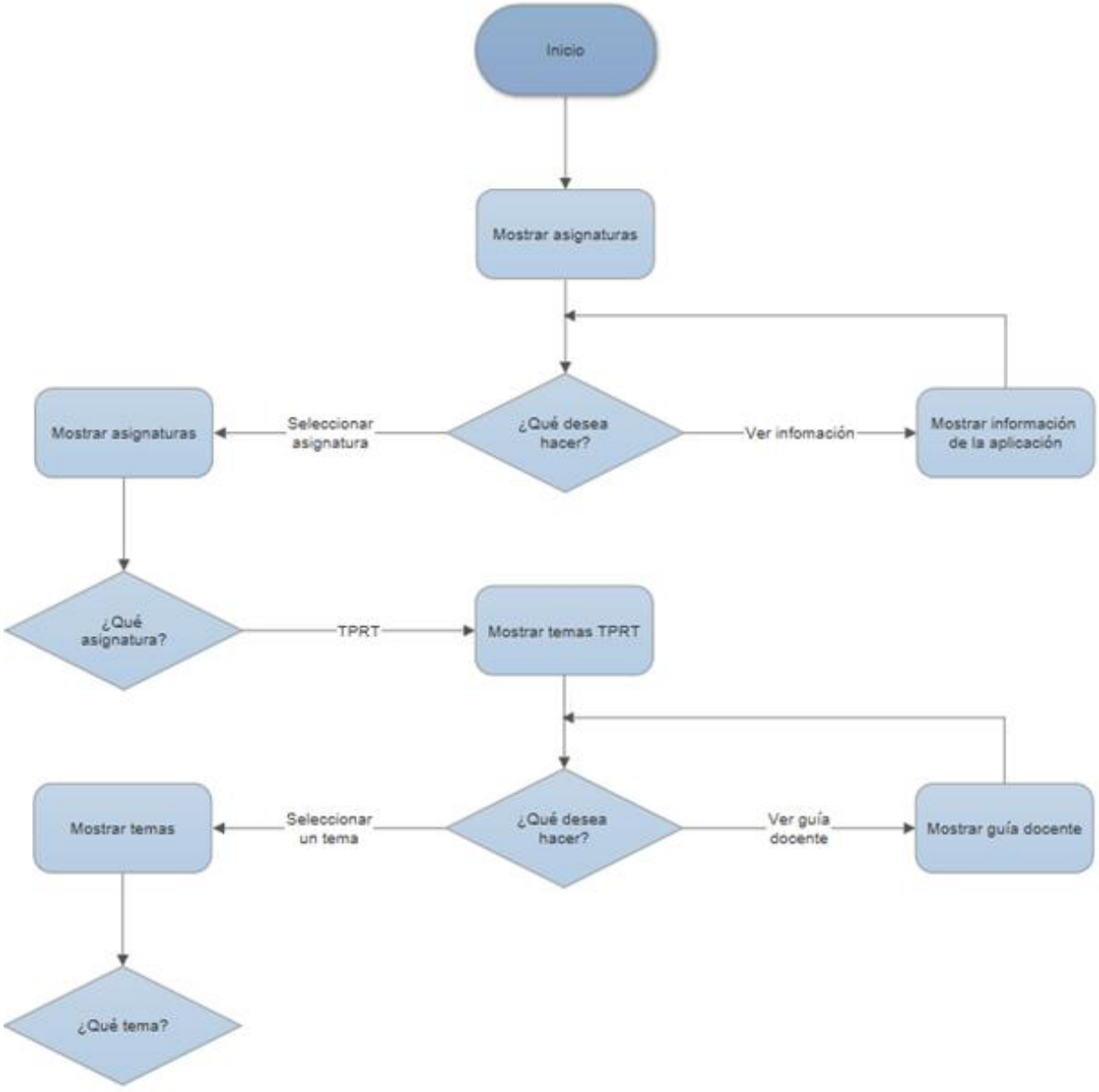

Figura 8. Diagrama de flujo comienzo de la aplicación.

Una vez llegados a este punto, el usuario puede elegir uno de los cuatro temas disponibles, o la lista de acrónimos que se usan en la asignatura. Los diagramas de flujo de cada uno de los temas son similares al del tema 2, que se muestra en la Figura 9, con la única diferencia de que los temas uno y tres no tienen prácticas de laboratorio.

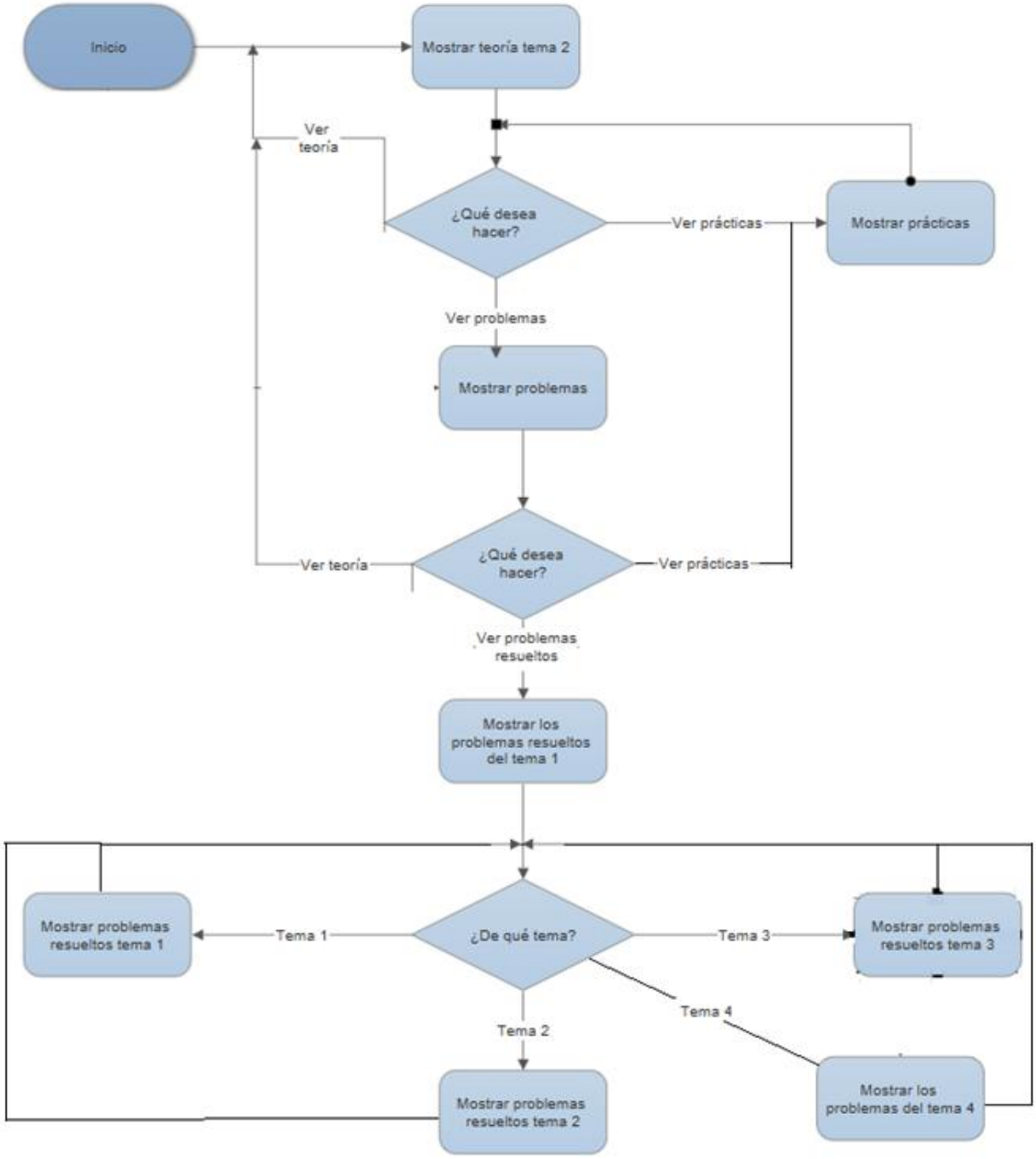

Figura 9. Diagrama de flujo del tema 2.

4.5 Resultados

En este apartado se explicarán las distintas pantallas que el usuario se va a encontrar al navegar por la aplicación. Se mostrará una imagen de cada una de ellas para ayudar a entender sus distintas partes.

Las primeras pruebas de funcionalidad y diseño se han realizado en un dispositivo *Smartphone Android*, concretamente un Cubot x9, que cuenta con el sistema operativo *Android 4.4.4* y una pantalla de 5" con 720 x 1280 píxeles de resolución.

4.5.1 Pantalla de inicio

Tras pulsar el icono de la app comienza la ejecución de la misma. En primer lugar aparece la pantalla de inicio, como se muestra en la *Figura 10*.

Figura 10. Pantalla de inicio

Pasados los tres segundos establecidos para la pantalla de inicio, esta desaparece dando lugar a la pantalla principal.

4.5.2 Pantalla principal

La pantalla principal muestra el acceso a la asignatura *TPRT*, única asignatura disponible actualmente en la aplicación, y a la asignatura *RST*, que aún no está implementada. En esta pantalla se observan en la parte superior dos pestañas que dan acceso al resto de pantallas de la aplicación.

En la pestaña "Asignaturas" se muestra un botón con una imagen de fondo, que permite el acceso a los distintos temas de la asignatura. Esta pantalla se muestra en la *Figura 11*.

En la pestaña “Acerca de”, que se muestra en la *Figura 11*, se presenta información sobre la aplicación TPRT. Es una pantalla informativa sobre los objetivos de la aplicación, la importancia de la misma y el equipo desarrollador de la misma.

Figura 11. Pantalla principal – Pestañas "Asignaturas" y “Acerca de”.

4.5.3 Pantalla principal de cada asignatura

Se accede a esta pantalla seleccionando el botón de la asignatura en la pestaña “Asignaturas” de la pantalla principal. En la pantalla principal de cada asignatura se muestra el acceso a cada uno de los temas que tiene la asignatura, así como a una lista de acrónimos usados en esta. En esta pantalla se pueden observar en la parte superior dos pestañas.

En la pestaña “temas” se muestra una lista, con una fila nueva por cada tema de la asignatura, que da acceso a la teoría, los problemas y, en caso de que hubiera, las prácticas de laboratorio de cada uno de los temas de la asignatura. La última fila de la lista da acceso a la información sobre los distintos acrónimos utilizados en la asignatura. Esta pantalla se muestra en la *Figura 12*.

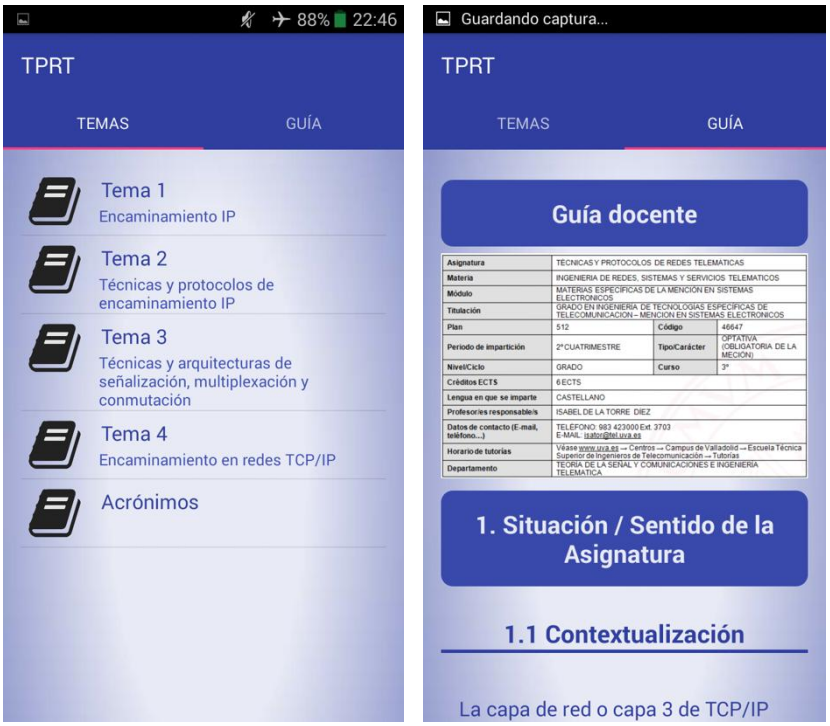

Figura 12.Pantalla principal de cada asignatura – Pestañas "Temas" y "Guía".

En la otra pestaño, tras el título de "guía" se muestra la guía docente de la asignatura. Esta guía tiene el mismo contenido que la guía que puede encontrarse en la página web de la ETSIT. Esta pantalla se muestra en la *Figura 12*.

4.5.4 Pantalla de cada tema

A esta sección se accede pulsando en el botón del tema deseado en la pestaña "temas" de la pantalla principal de la asignatura. En esta pantalla se muestra la distinta información sobre el tema en cuestión. En la parte superior de la pantalla se pueden observar dos o tres pestañas, en función si el tema tiene laboratorio o no.

En la pestaña "teoría", que se muestra en la *Figura 13*, se muestra la información teórica sobre el tema.

En la pestaña "problemas" se muestran varios problemas relacionados con la teoría indicada en la pestaña anterior. Esta pantalla se puede ver en la *Figura 13*.

La pestaña "prácticas" solo está disponible en aquellos temas que tienen práctica de laboratorio asociada, y muestra el enunciado de la misma. Si la práctica de laboratorio de algún tema es muy extensa, parte de esta se mostrará en otra sección. Esta pantalla se puede ver en la *Figura 13*.

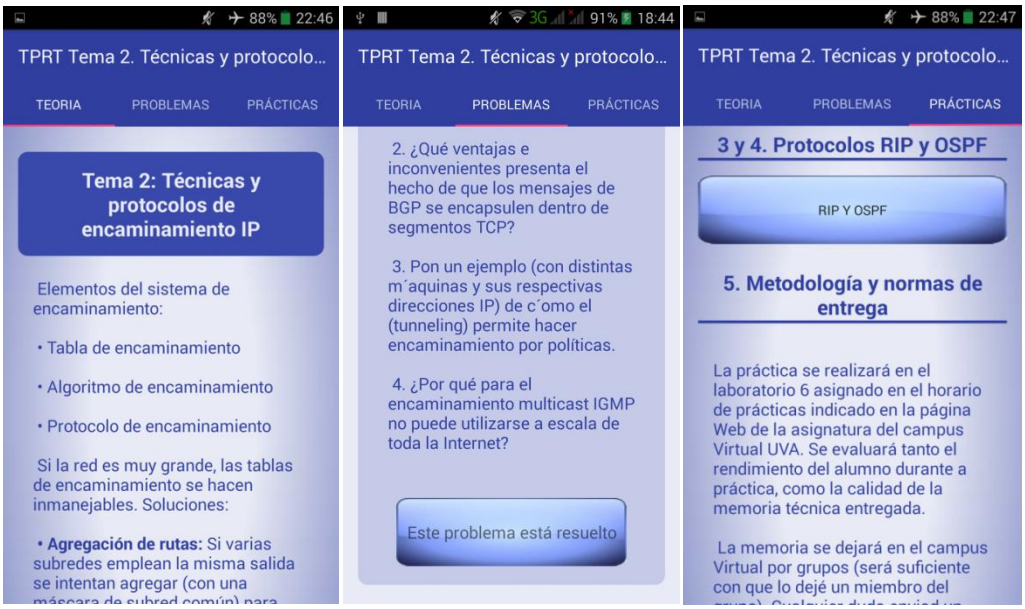

Figura 13. Pantalla de cada tema - Pestañas "Teoría", “Problemas” y “Prácticas”.

En el caso de la práctica del tema 2, se ha decidido dividirla en varias partes, mostrando en una pantalla nueva las partes de la práctica relacionadas con SIP y OSPS. Esta nueva pantalla se divide en dos pestañas, la pestaña “RIP” muestra la parte de la práctica relacionada con RIP y la pestaña “OSPF” muestra la parte de la práctica relacionada con OSPF. Esta pantalla se puede ver en la *Figura 14.*

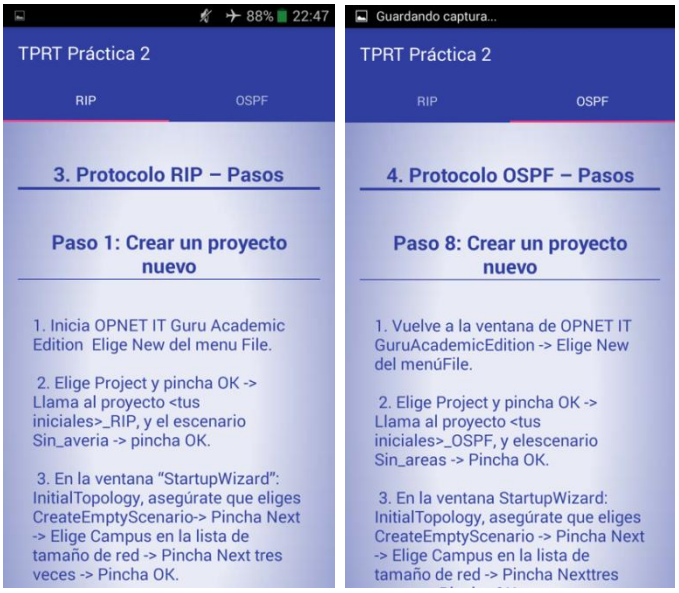

Figura 14. Pantalla extra práctica 2 - Pestañas “RIP” y “OSPF.

4.5.5 Pantalla acrónimos

Esta pantalla se muestra cuando se selecciona el botón “acrónimos” de la pantalla principal de la asignatura. Muestra una lista con los distintos acrónimos usados en la asignatura, separados por temas y ordenados alfabéticamente. Esta pantalla se muestra en la *Figura 15*.

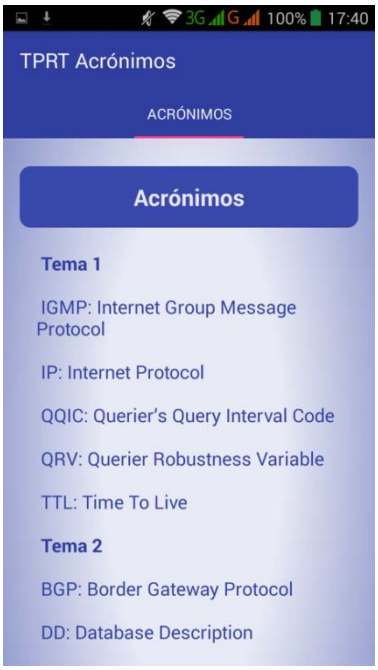

Figura 15. Pantalla “Acrónimos”.

4.5.6 Pantalla de problemas resueltos

Esta pantalla se muestra cuando se selecciona el botón “Este problema está resuelto” de pestaña “problemas” de la pantalla principal de cada tema. En esta pantalla se muestra los problemas resueltos disponibles para cada tema de la asignatura seleccionada. En la parte superior de la pantalla se pueden observar varias pestañas, que permiten mostrar los problemas resueltos en función del tema. Esta pantalla se muestra en la *Figura 16*.

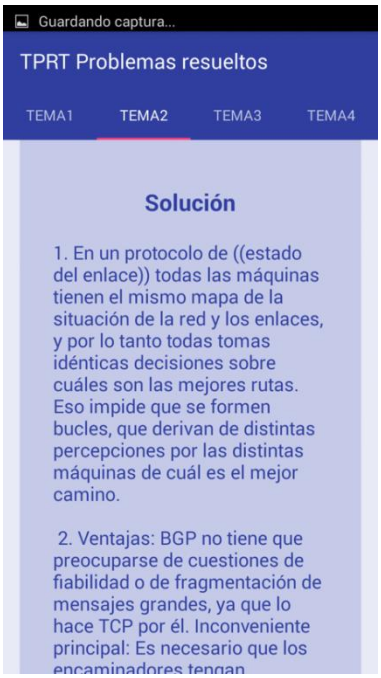

Figura 16. Pantalla de problemas resueltos.

4.6 Evaluación en dispositivos

Para comprobar la adaptabilidad de la aplicación a los diversos tamaños y pantallas posibles, se ha comprobado el correcto funcionamiento de la misma en dos *smartphones* y una *tablet* cuyas características principales son:

- *Tablet Samsung Galaxy Tab A*: pantalla de 9.7” con una resolución de 1024x768px. Sistema operativo *Android 5.0 Lollipop*.
- *Smarthpone Oukitel k6000 pro*: pantalla de 5.5” con una resolución de 1080x1920 px. Sistema operativo *Android 6.0 Marshmallow*.
- *Smarthpone Cubot x9*: pantalla de 5.0” con una resolución de 720x1280 px. Sistema operativo *Android 4.4.4 KitKat*.

A continuación se muestran las imágenes de la prueba, tanto en posición vertical (*portrait*) como horizontal (*land*).

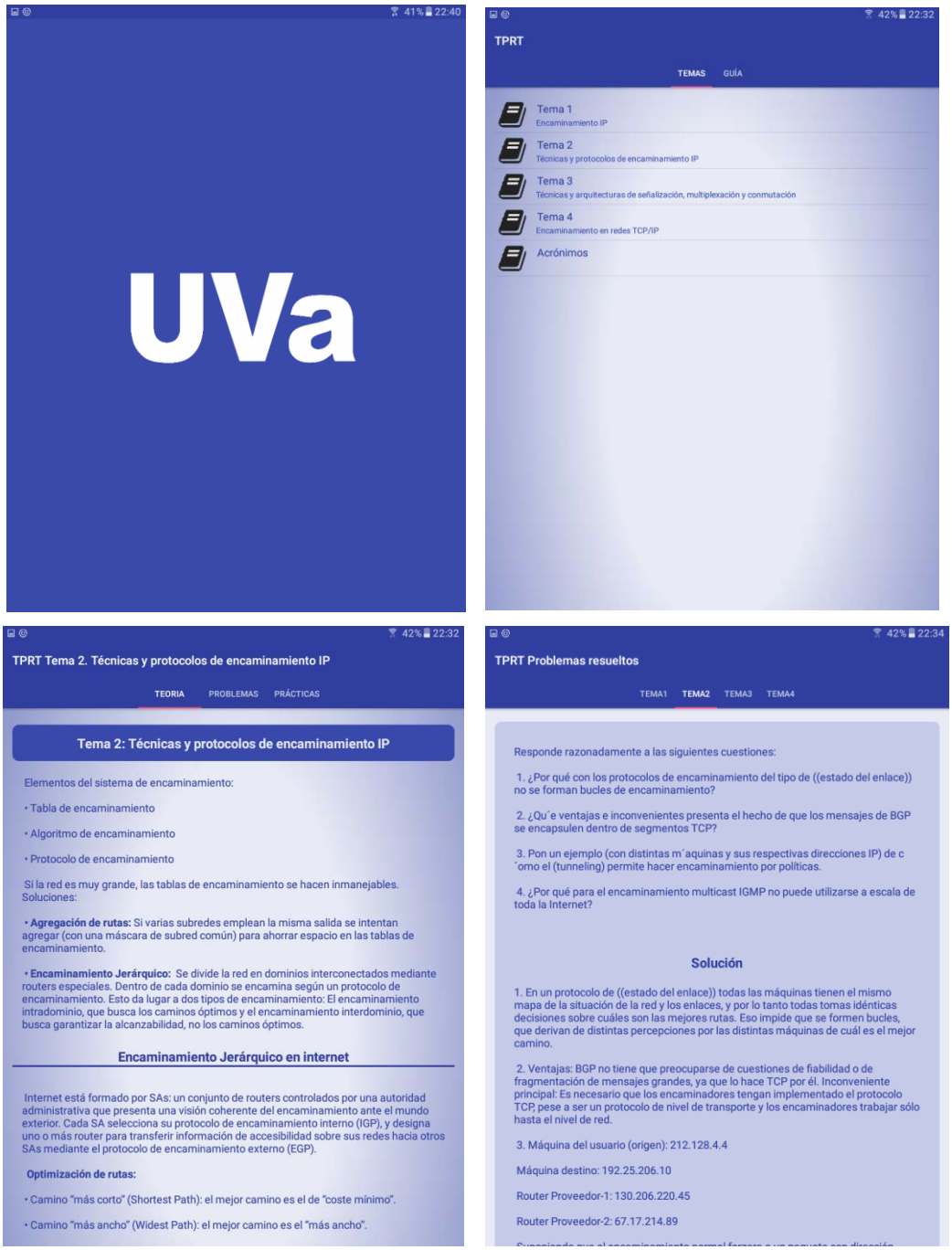

Figura 17. Ejecución en *portrait* en Samsung Galaxy Tab A.

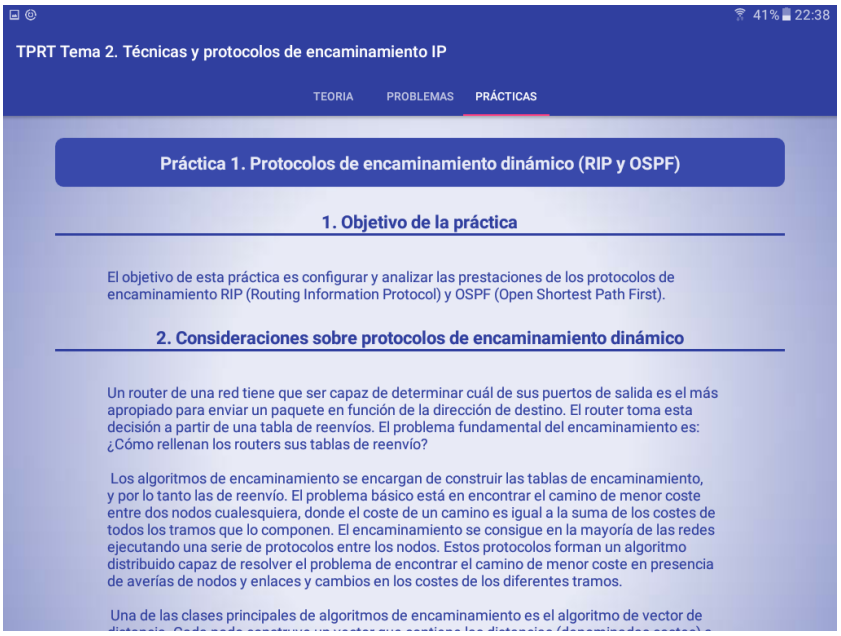

Figura 18. Ejecución en *land* Samsung Galaxy Tab A.

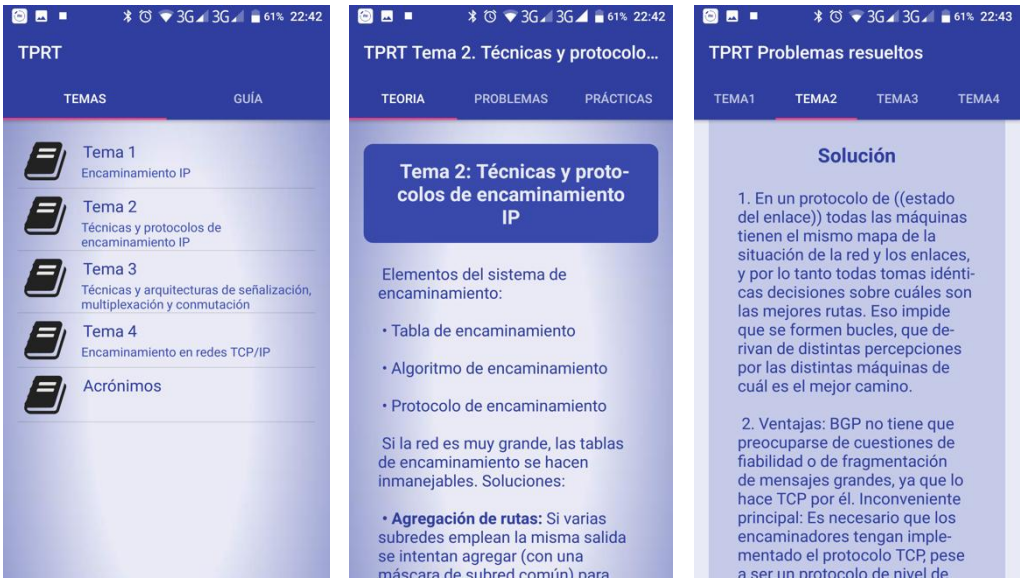

Figura 19. Ejecución en *portrait* en Oukitel k6000 pro.

Figura 20. Ejecución en *portrait* y *land* en Cubot X9.

Capítulo 5. Conclusiones y líneas futuras

5.1 Conclusiones

En este apartado se expondrán tanto las conclusions generales, como las conclusiones de la *app* y su proceso de desarrollo.

5.1.1 Conclusiones generales

Tras finalizar el desarrollo de este TFG, hay varios aspectos a comentar en este apartado de conclusiones. Sobre todo las conclusiones relacionadas con *e-learning*, y su posterior evolución a *m-learning*.

Hoy en día las tecnologías están muy integradas en la vida de las personas. Todo el mundo emplea en su vida diaria el *smartphone* para varias funciones, no solo en el ámbito del ocio, sino también en el ámbito profesional y básicamente en cualquier campo. Esto se debe a que hoy en día hay una gran cantidad de aplicaciones móviles que tratan diversos campos. Las aplicaciones más empleadas son las destinadas al ocio o a las comunicaciones personales, pero hay muchas otras aplicaciones destinadas a la ayuda en la medicina o en la educación, entre otras cosas.

Estas aplicaciones móviles relacionadas con la educación se van implantando poco a poco en los diversos sistemas educativos, haciendo los métodos de enseñanza más eficientes tanto para los estudiantes, que pueden acceder a la información sobre la materia en cualquier lugar y en cualquier hora, como para los docentes, que pueden por ejemplo, informar a los alumnos sobre modificaciones en el acto a través de la aplicación.

Pero no solo son útiles en el sentido de intentar mejorar el sistema educativo, sino que también permiten a cualquier persona interesada en la materia obtener información sobre esta. Pudiendo así adquirir los conocimientos deseados sin necesidad de tener que desplazarse para encontrar esta información.

Esta nueva necesidad de poder tener acceso a la información sin tener que desplazarse a otro lugar, junto con la facilidad de distribución de aplicaciones móviles, ha llevado a que cada

vez sea mayor el número de aplicaciones que se pueden encontrar en las diversas tiendas virtuales, como *Play Store* o *App Store*. Pero como suele pasar en internet, es fácil encontrar aplicaciones que traten temas genéricos como "aprende matemáticas" o "aprende informática", pero a la hora de querer profundizar en una determinada materia cada vez es más difícil encontrar información en estas *apps*.

La aplicación llevada a cabo en este proyecto trata un tema bastante específico. Esto, junto con los otros motivos que se acaban de comentar, da lugar a pensar que el proyecto que se ha llevado a cabo sí que puede tener cabida en el mercado en constante evolución que es el de las aplicaciones móviles.

5.1.2 Conclusiones de la app y su desarrollo

En este trabajo se ha realizado una *app* destinada al uso académico, más concretamente al estudio de la asignatura *Técnicas y Protocolos de Redes Telemáticas*. Tras haber probado la *app* en varios dispositivos se han podido sacar algunas conclusiones sobre esta.

En cuanto a la adaptabilidad de esta *app* a los diversos tamaños y resoluciones de pantallas en varios dispositivos, se puede decir que se ha conseguido que la aplicación tenga un tamaño y una disposición de los elementos acordes a la pantalla en la cual se ubica. Esto, junto con los tonos elegidos para la aplicación y el diseño, ha creado una interfaz agradable para el usuario, que puede reconocer perfectamente las acciones que realiza cada uno de los elementos que se muestran en la pantalla, facilitando así su uso.

Al analizar su funcionalidad, se puede observar que se han cumplido los objetivos propuestos, consiguiendo una *app* que muestra la información de forma ordenada y agradable a la vista, pudiendo ayudar así a los estudiantes de esta materia a poder aprender eficientemente sus contenidos.

5.2 Líneas futuras

Sin duda alguna, la mejor forma de probar esta aplicación y ver qué aspectos se podrían añadir o cambiar, sería recibir un *feedback* de su usuario final: los estudiantes de esa asignatura. Para ello los estudiantes tendrían que usar la aplicación durante un tiempo, y luego contestar a una sencilla encuesta que tendría que tener en cuenta varios aspectos de la aplicación, como pueden ser:

- Diseño.
- Utilidad de las funciones e información ofrecidas.
- Facilidad de uso.
- Aspectos a mejorar.

No obstante, tras haber finalizado el desarrollo de este proyecto, se me ocurren algunos aspectos que se podrían tener en cuenta en las sucesivas versiones de esta aplicación:

- Añadir más asignaturas a la aplicación.
- Permitir a los usuarios registrarse en la aplicación.
- Permitir a los usuarios registrados marcar que problemas saben hacer y en cuales han tenido dificultades, así como notas sobre estos. También podrían marcar los temas que se saben bien y los que aún tienen que estudiar o repasar.
- Separar los usuarios registrados en varios roles, dando a cada uno de estos distintas funcionalidades. Por ejemplo se puede crear un rol de docente, y permitir a este añadir nueva información, o modificar la información a existente.
- Permitir a los usuarios registrados indicar qué día quieren haber aprendido (al menos en teoría) cada tema. Mostrando una notificación si al llegar ese día, aún no han indicado que saben la teoría y que les salen todos los problemas.
- Desarrollar esta aplicación para otros sistemas operativos. Principalmente para iOS, ya que es el segundo sistema operativo para móviles más utilizado. De este modo los usuarios de este sistema operativo también tendrían acceso a la aplicación.
- Mostrar el contenido de la aplicación en varios idiomas, como inglés y francés, con el objetivo de aumentar el número de destinatarios.
- Al ser una primera versión de la aplicación, también se puede mejorar la interfaz gráfica.
- Incluir una forma de conexión telemática, como un foro o un chat, donde los usuarios de la aplicación puedan discutir sobre diversos aspectos de cada asignatura, compartiendo distintos puntos de vista y resolviendo en común las dudas que le pueda surgir a cada uno.

Capítulo 7. Bibliografía

[1] Alejandra Collazos. (2014). Blended Learning o Aprendizaje Semipresencial, el Nuevo método que ahora están adoptando los profesores. [Consulta: junio de 2017] Disponible en: https://revistaeducacionvirtual.com/archives/944

[2] Android Developers. (2017). *Arquitectura de la plataforma.* [Consulta: junio de 2017] Disponible en: https://developer.android.com/guide/platform/index.html?hl=es

[3] Android Developers (2017). Todo lo que necesitas para realizar compilaciones en Android. [Consulta: junio de 2017] Disponible en: https://developer.android.com/studio/features.html?hl=es-419

[4] Apple. (2017). *About the iOS Technologies*. [Consulta: junio de 2017] Disponible en: https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007898-CH1-SW1

[5] btilford. (2017). *Comparison of Git with CVS and Git Overview* . [Consulta: junio de 2017] Disponible en: https://gist.github.com/btilford/716213/f504ee44c5849bd0107193beb96eec2684f6e9a4

[6] Carlos Luis Murillo. (2015). Windows pone 8. [Consulta: junio de 2017] Disponible en: http://windowsphone8general.blogspot.com.es

[7] Carltic. (2017). *iOS y su arquitectura interna en 4 capas*. [Consulta: junio de 2017] Disponible en: http://blog.ticsandroll.es/ios-y-su-arquitectura-interna-en-4-capas/

[8] Cristian Rus. (2016). *La evolución de iOS desde sus orígenes: una carrera para ser el mejor sistema operativo móvil de la historia*. [Consulta: junio de 2017] Disponible en: https://www.applesfera.com/ios/la-evolucion-de-ios-desde-sus-origenes-una-carrera-para-ser-el-mejor-sistema-operativo-movil-de-la-historia

[9] David López Villegas. (2014). *IDE: Entornos Integrados de Desarrollo para Android.*[Consulta: junio de 2017] Disponible en:https://academiaandroid.com/ide-entornos-integrados-de-desarrollo-para-android/

[10] iPadizate. (2016). *La Evolución de iOS: Un Paseo desde iOS 1 hasta iOS 9*. [Consulta: junio de 2017] Disponible en: https://www.ipadizate.es/2016/05/03/evolucion-ios-1-ios-9/

[11] Jesús Tomás Gironés. (2013). *El gran libro de Android. 3º Edición.*[Consulta: junio de 2017] Disponible en: https://books.google.es/books?id=K9hnCJ_NGq4C&pg=PT23&lpg=PT23&dq=En+noviembre+del+2007+se+lanza+una+primera+versi%C3%B3n+del+Android+SDK.&source=bl&ots=KGGoSEx4gO&sig=-FOBiduHD3OLxzfFMXsbreIUIss&hl=es&sa=X&ved=0ahUKEwjcquPsz9PUAhVKahoKHXWYDdwQ6AEILTAA#v=onepage&q=En%20noviembre%20del%202007%20se%20lanza%20una%20primera%20versi%C3%B3n%20del%20Android%20SDK.&f=false

[12] El mundo (2016).*El 50% de la educación superior en el mundo se impartirá por E-Learning*. [Consulta: junio de 2015] Disponible en: http://www.elmundo.es/sociedad/2016/04/28/571f94b222601dab7c8b45c8.html

[13] Genbetadeb (2014). *Eclipse IDE.* [Consulta: junio de 2017] Disponible en: https://www.genbetadev.com/herramientas/eclipse-ide

[14] Lancetalent. (2014). *Los 3 tipos de aplicaciones móviles: ventajas e inconvenientes.* [Consulta: junio de 2015] Disponible en: https://www.lancetalent.com/blog/tipos-de-aplicaciones-moviles-ventajas-inconvenientes/

[15] Pau Yanez. (2015). *E-learning, M-learning y B-learning.*[Consulta: junio de 2015] Disponible en:http://www.icalia.com/e-learning-m-learning-y-b-learning/

[16] Santiago Moll. (2014). *Blended learning. 15 Razones para adopter este modelo de enseñanza*. [Consulta: junio de 2015] Disponible en: http://justificaturespuesta.com/blended-learning-15-razones-para-adoptar-este-modelo-de-ensenanza/

[17] Torre Díez, Isabel. (2016). *Técnicas y protocolos de redes telemáticas*. Valladolid: Universidad de Valladolid.

[18] Jetbrains (2017). *IntelliJ IDEA.* [Consulta: junio de 2017] Disponible en: https://www.jetbrains.com/idea/?fromMenu

[19] Rodríguez Cayetano, M. (2013). *Introducción al sistema operativo Android*. Valladolid: Universidad de Valladolid.

[20] Scott Chacony Ben Straub. (2014). *Pro Git*. [Consulta: junio de 2017] Disponible en: https://git-scm.com/book/en/v2

[21] Tecnología iOS. (2017). *Arquitectura iOS*. [Consulta: junio de 2017] Disponible en:ghttps://sites.google.com/site/tecnologiaiostm/desarrollo-de-aplicaciones/arquitectura-ios

[22] Universidad Carlos III. (2017) *2.2. Arquitectura Android.* [Consulta: junio de 2017] Disponible en: https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android

[23] Windows Phone. (2016). *Introdcción a Windows Phone*. [Consulta: junio de 2017] Disponible en: http://conceptodefinicion.de/windows-phone/