Actualizado a versión

9.0 Pie y

Android Studio

El gran libro de Android

Jesús Tomás

7.ª edición

Marcombo

Alfaomega

El gran libro de Android

Jesús Tomás Gironés

El gran libro de Android

Jesús Tomás Gironés

Marcombo

Alfaomega

ALFAOMEGA

Empresas del Grupo

Colombia: Alfaomega Colombiana S.A.
Calle 62 No.20-46 esquina, Bogotá
Teléfono (57-1) 746 0102 Fax: (57-1) 210 0122
cliente@alfaomegacolombiana.com

México: Alfaomega Grupo Editor S.A. de C.V.
Calle Doctor Olvera No. 74, Colonia Doctores,
Delegación Cuaúhtemoc, Ciudad de México
C.P. 06720 • teléfono (52-55) 5089 7740
Fax (52-55) 5575 2420
Sin costo 01-800-020-4396
libreriapitagoras@alfaomega.com.mx

Argentina: Alfaomega Grupo Editor Argentino S.A.
Av. Córdoba 1215, Piso 10
Capital Federal, Buenos Aires
Teléfono/Fax: (54-11) 4811 7183 / 8352 / 0887
ventas@alfaomegaeditor.com.ar

Chile: Alfaomega Grupo Editor S.A.
Av. Providencia 1443. Oficina 24, Santiago
Teléfonos (56-2) 2235 4248 / 2947 9351 / 2235 5786
agechile@alfaomega.cl

www.alfaomega.com.co

El gran libro de Android
Bogotá, 2019

Diseño de la cubierta: Enedenú diseño gráfico
Corrección: Raquel Sayas Lloris

ISBN: 978-958-778-544-9 (Edición Colombia)
ISBN: 978-84-267-2662-9 (Edición España)

Hecho en Colombia
Printed and made in Colombia

Para Bea, con amor y gratitud.

Mis agradecimientos a los alumnos y lectores que con sus sugerencias y correcciones han ayudado a mejorar este libro.

Índice general

Lista de siglas y acrónimos

AIDL	*Android Interface Definition Language*
API	*Application Programming Interface*
AVD	*Android Virtual Device*
ART	*Android RunTime*
CSS	*Cascading Style Sheets*
CORBA	*Common Object Request Broker Architecture*
CPU	*Central Processing Unit*
DOM	*Document Object Model*
DTD	*Document Type Definition*
FTP	*File Transfer Protocol*
GPU	*Graphic Processing Unit*
GPS	*Global Positioning System*
GSM	*Global System for Mobile communications*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IDE	*Integrated Development Environment*
IMEI	*International Mobile Equipment Identity*
IMSI	*International Mobile Subscriber Identity*
IU	*Interfaz de Usuario*
JAR	*Java ARchive*
JDK	*Java Development Kit*
JRE	*Java Runtime Environment*
JSON	*JavaScript Object Notation*
JVM	*Java Virtual Machine*
MCC	*Mobile Country Code*
MNC	*Mobile Network Code*
MIME	*Multipurpose Internet Mail Extensions*
MTP	*Media Transfer Protocol*
NFC	*Near Field Communication*
NDK	*Native Development Kit*
OpenGL	*Open Graphic Library*

PCM	*Pulse-Code Modulation*
PDA	*Personal Digital Assistant*
PNG	*Portable Network Graphics*
PHP	*Hypertext Pre-processor*
PTP	*Picture Transfer Protocol*
RAM	*Random Access Memory*
REST	*Representational State Transfer*
RMI	*Remote Method Invocation*
RPC	*Remote Procedure Calls*
SAX	*Simple API for XML*
SD	*Secure Digital*
SDK	*Software Developers Kit*
SMS	*Short Message Service*
SIM	*Subscriber Identity Module*
SO	*Sistema Operativo*
SOA	*Service-Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
SVG	*Scalable Vector Graphics*
TCP	*Transmission Control Protocol*
UI	*User Interface*
URL	*Universal Resource Locator*
URI	*Uniform Resource Identifier*
USB	*Universal Serial Bus*
UTC	*Universal Time Coordinate*
UICC	*Universal Integrated Circuit Card*
W3C	*World Wide Web Consortium*
WSDL	*Web Services Description Language*
WWW	*World Wide Web*
XML	*Extensible Markup Language*

¿Cómo leer este libro?

Este libro quiere ser una guía para aquellos lectores que pretendan introducirse en la programación de Android. Se ha estructurado en 10 capítulos que abordan aspectos específicos del desarrollo de aplicaciones. Resulta conveniente realizar una lectura secuencial de estos capítulos, dado que muchos de los conceptos que se abordan se comprenderán mejor si se han leído los capítulos anteriores. Además, a lo largo del libro se desarrollan dos proyectos de ejemplo: el mítico juego Asteroides y la aplicación Mis Lugares. Para que muchos de los ejercicios funcionen correctamente, resulta imprescindible realizar los anteriores. Si el lector está interesado en un texto que aborde aspectos específicos de la programación en Android, le recomendamos otros libros de esta misma colección: *El gran libro de Android Avanzado, Firebase: Trabajando en la nube*, *Plataformas Android: Wear, TV, Auto y Google Play Games* y *Android Things y Visión Artificial*. Todos publicados en esta misma editorial.

El libro que tienes entre las manos no ha sido concebido solo para ser leído. Es más bien una guía estructurada que te irá proponiendo una serie de ejercicios, actividades, vídeos explicativos, test de autoevaluación, etc. Todo este material y muchos más recursos adicionales están disponibles en la web www.androidcurso.com. En ella se publicarán las novedades, erratas e información complementaria relativas a este libro. Por lo tanto, resulta imprescindible para sacarle partido a este libro un ordenador con el SDK de Android instalado para hacer los ejercicios y acceso a Internet para el material en línea.

A lo largo del libro se utilizan los siguientes iconos para indicar los tipos de actividades:

Objetivos: Antes de empezar cada capítulo, lee con detenimiento la introducción y los objetivos.

Vídeo[tutorial]: Más de 80 vídeos grabados por el autor del libro donde se exponen de forma didáctica los aspectos clave del sistema Android. Se utiliza una moderna herramienta desarrollada en la Universidad Politécnica de Valencia que te permitirá ver simultáneamente las presentaciones y al profesor mientras se desarrollan los conceptos de cada capítulo.

Ejercicio: La mejor forma de aprender es haciendo. No tendrás más que ir siguiendo los pasos uno tras otro para descubrir cómo se resuelve el ejercicio propuesto. Para que no se te haga pesado teclear todo el código, te proponemos que lo copies y pegues desde la página web del curso.

Práctica: Este será el momento de que tomes la iniciativa y trates de resolver el problema que se propone. Recuerda que para aprender hay que practicar.

Asteroides: Ejercicios y prácticas que te permitirán desarrollar el videojuego Asteroides.

Mis Lugares: Ejercicios y prácticas que te permitirán desarrollar la aplicación Mis Lugares.

Solución: Te será de ayuda si tienes problemas al resolver una práctica, o si simplemente quieres comparar tu solución con otra diferente.

Nota sobre Java: Si no dominas el lenguaje de programación Java, podrás seguir este libro. Cada vez que aparezca algún concepto complejo sobre Java trataremos de aclararlo. ***NOTA:*** *Sí que resulta imprescindible disponer de conocimientos sobre programación.*

Recursos adicionales: Te proporcionamos la información clave que te ayudará en el desarrollo de tus aplicaciones.

Enlaces de interés: Internet te será de gran ayuda para completar la información necesaria para programar en Android. Te proponemos las páginas más interesantes de cada apartado.

Preguntas de repaso: ¿Has comprendido correctamente los aspectos clave? Sal de dudas haciendo los test de autoevaluación.

Referencias rápidas: Utiliza los anexos para localizar rápidamente esa palabra clave o esa clase que no recuerdas.

Trivial programación Android: Instálate esta app y mide tus conocimientos jugando en red contra otros oponentes.

De forma adicional, en la web www.androidcurso.com encontrarás:

- **Tutoriales sobre Java:** ¿Sabes lo que es la herencia, el polimorfismo o la sobrecarga en Java? Si no dominas el lenguaje de programación Java, te recomendamos que realices alguno de los tutoriales propuestos.
- **Código abierto de proyectos Android:** Muchos alumnos que han realizado un curso basado en este libro han tenido la generosidad de compartir sus proyectos con todos nosotros. Te recomendamos que consultes la lista de proyectos disponibles de código abierto: puedes aprender mucho estudiando su código. Cuando termines de leer este libro, también tú podrás hacer un proyecto como los que se muestran.
- **Material adicional sobre Android:** Encontrarás, además, nuevos tutoriales, vídeos, referencias, etc., no incluidos en el libro.
- **Cursos *online*:** Si te interesa ampliar tu formación, puedes matricularte en cursos sobre Android impartidos por la Universidad Politécnica de Valencia en la plataforma EdX. Incluso puedes obtener un título de Especialización o de Máster de forma 100 % *online*.

CAPÍTULO 1.

Visión general y entorno de desarrollo

La telefonía móvil está cambiando la sociedad actual de una forma tan significativa como lo ha hecho Internet. Esta revolución no ha hecho más que empezar; los nuevos terminales ofrecen unas capacidades similares a un ordenador personal, lo que permite que puedan ser utilizados para leer el correo o navegar por Internet. Pero, a diferencia de un ordenador, un teléfono móvil siempre está en el bolsillo del usuario. Esto permite un nuevo abanico de aplicaciones mucho más cercanas al usuario. De hecho, muchos autores coinciden en afirmar que el nuevo ordenador personal del siglo XXI será un terminal móvil.

El lanzamiento de Android como nueva plataforma para el desarrollo de aplicaciones móviles ha causado una gran expectación y ha tenido una importante aceptación tanto por parte de los usuarios como por parte de la industria. En la actualidad se ha convertido en la alternativa dominante frente a otras plataformas como iPhone o Windows Phone.

A lo largo de este capítulo veremos las características de Android que lo hacen diferente de sus competidores. Se explicará también cómo instalar y trabajar con el entorno de desarrollo (Android Studio).

Objetivos:

- Conocer las características de Android, destacando los aspectos que lo hacen diferente de sus competidores.
- Estudiar la arquitectura interna de Android.

- Aprender a instalar y trabajar con el entorno de desarrollo (Android SDK).
- Enumerar las principales versiones de Android y aprender a elegir la más idónea para desarrollar nuestras aplicaciones.
- Crear una primera aplicación y estudiar su estructura de un proyecto en Android.
- Conocer dónde podemos conseguir documentación sobre Android.
- Aprender a utilizar herramientas para detectar errores en el código.

1.1. ¿Qué hace que Android sea especial?

Como hemos comentado, existen muchas plataformas para móviles (Apple iOS, Windows Phone, BlackBerry, Palm, Java Micro Edition, Linux Mobile (LiMo), Firefox OS, etc.); sin embargo, Android presenta una serie de características que lo hacen diferente. Es el primero que combina en una misma solución las siguientes cualidades:

- **Plataforma realmente abierta**. Es una plataforma de desarrollo libre basada en Linux y de código abierto. Una de sus grandes ventajas es que se puede usar y customizar el sistema sin pagar *royalties*.
- **Adaptable a cualquier tipo de *hardware***. Android no ha sido diseñado exclusivamente para su uso en teléfonos y tabletas. Hoy en día podemos encontrar relojes, gafas, cámaras, TV, sistema para automóviles, electrodomésticos y una gran variedad de sistemas empotrados que se basan en este sistema operativo, lo cual tiene sus evidentes ventajas, pero también va a suponer un esfuerzo adicional para el programador. La aplicación ha de funcionar correctamente en dispositivos con una gran variedad de tipos de entrada, pantalla, memoria, etc. Esta característica contrasta con la estrategia de Apple: en iOS tenemos que desarrollar una aplicación para iPhone y otra diferente para iPad.
- **Portabilidad asegurada**. Las aplicaciones finales son desarrolladas en Java, lo que nos asegura que podrán ser ejecutadas en cualquier tipo de CPU, tanto presente como futuro. Esto se consigue gracias al concepto de máquina virtual.
- **Arquitectura basada en componentes inspirados en Internet**. Por ejemplo, el diseño de la interfaz de usuario se hace en XML, lo que permite que una misma aplicación se ejecute en un reloj de pantalla reducida o en un televisor.
- **Filosofía de dispositivo siempre conectado a Internet**. Muchas aplicaciones solo funcionan si disponemos de una conexión permanente a Internet. Por ejemplo, comunicaciones interpersonales o navegación con mapas.
- **Gran cantidad de servicios incorporados**. Por ejemplo, localización basada tanto en GPS como en redes, bases de datos con SQL, reconocimiento y síntesis de voz, navegador, multimedia, etc.

- **Aceptable nivel de seguridad**. Los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja, que hereda de Linux. Además, cada aplicación dispone de una serie de permisos que limitan su rango de actuación (servicios de localización, acceso a Internet, etc.). Desde la versión 6.0 el usuario puede conceder o retirar permisos a las aplicaciones en cualquier momento.
- **Optimizado para baja potencia y poca memoria**. En el diseño de Android se ha tenido en cuenta el *hardware* específico de los dispositivos móviles. Por ejemplo, Android utiliza la máquina virtual ART (o Dalvik en versiones antiguas). Se trata de una implementación de Google de la máquina virtual Java optimizada para dispositivos móviles.
- **Alta calidad de gráficos y sonido**. Gráficos vectoriales suavizados, animaciones, gráficos en 3D basados en OpenGL. Incorpora los codecs estándares más comunes de audio y vídeo, incluyendo H.264 (AVC), MP3, AAC, etc.

Como hemos visto, Android combina características muy interesantes. No obstante, la pregunta del millón es: ¿se convertirá Android en el sistema operativo (SO) estándar para dispositivos móviles? Para contestar a esta pregunta habrá que ver la evolución del iPhone de Apple y cuál es la respuesta de Windows con el lanzamiento de su SO para móviles. No obstante, Android ha alcanzado un 85 % de cuota de mercado (90 % en España), cosa que lo deja en una posición predominante que es difícil que pierda a corto plazo.

En conclusión, Android nos ofrece una forma sencilla y novedosa de implementar potentes aplicaciones para diferentes tipos de dispositivos. A lo largo de este texto trataremos de mostrar de la forma más sencilla posible cómo conseguirlo.

1.2. Los orígenes

Google adquiere Android Inc. en el año 2005. Se trataba de una pequeña compañía, recién creada, orientada a la producción de aplicaciones para terminales móviles. Ese mismo año empiezan a trabajar en la creación de una máquina virtual Java optimizada para móviles (Dalvik VM).

En el año 2007 se crea el consorcio Open Handset Alliance[1] con el objetivo de desarrollar estándares abiertos para móviles. Está formado por Google, Intel, Texas Instruments, Motorola, T-Mobile, Samsung, Ericsson, Toshiba, Vodafone, NTT DoCoMo, Sprint Nextel y otros. Uno de los objetivos fundamentales de esta alianza es promover el diseño y la difusión de la plataforma Android. Sus miembros se han comprometido a publicar una parte importante de su propiedad intelectual como código abierto bajo licencia Apache v2.0.

[1] http://www.openhandsetalliance.com

En noviembre de 2007 se lanza una primera versión del Android SDK. Al año siguiente aparece el primer móvil con Android (T-Mobile G1). En octubre, Google libera el código fuente de Android, principalmente bajo licencia de código abierto Apache (licencia GPL v2 para el núcleo). Ese mismo mes se abre Android Market, para la descarga de aplicaciones. En abril de 2009, Google lanza la versión 1.5 del SDK, que incorpora nuevas características como el teclado en pantalla. A finales de 2009 se lanza la versión 2.0 y a lo largo de 2010, las versiones 2.1, 2.2 y 2.3.

Durante el año 2010, Android se consolida como uno de los sistemas operativos para móviles más utilizados, con resultados cercanos a iOS e incluso superando al sistema de Apple en EE.UU.

En el año 2011 se lanza la versión 3.x (Honeycomb), específica para tabletas, y la 4.0 (Ice Cream Sandwich), tanto para móviles como para tabletas. Durante ese año, Android se consolida como la plataforma para móviles más importante y alcanza una cuota de mercado superior al 50 %.

En 2012, Google cambia su estrategia en su tienda de descargas *online*, reemplazando Android Market por Google Play Store, donde en un solo portal unifica tanto la descarga de aplicaciones como la de contenidos. Ese año aparecen las versiones 4.1 y 4.2 (Jelly Bean). Android mantiene su espectacular crecimiento y alcanza, a finales de año, una cuota de mercado del 70 %.

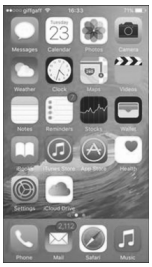
En 2013 se lanzan las versiones 4.3 y 4.4 (KitKat). En 2014 se lanza la versión 5.0 (Lollipop). A finales de ese año, la cuota de mercado de Android llega al 85 %. En octubre de 2015 ha aparecido la versión 6.0, con el nombre de Marshmallow. En 2016 se lanzó la versión 7.0, Android Nougat. A finales de 2017 aparece la versión 8.0, con nombre Android Oreo. En agosto de 2018 se lanza la versión 9.0, Android Pie.

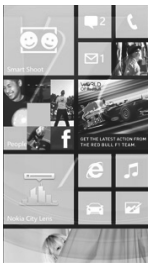
Vídeo[tutorial]: *Introducción a la plataforma Android*

Preguntas de repaso: *Características y orígenes de Android*

1.3. Comparativa con otras plataformas

En este apartado vamos a describir las características de las principales plataformas móviles disponibles en la actualidad. Dado la gran cantidad de datos que se indican, hemos utilizado una tabla para representar la información. De esta forma resulta más sencillo comparar las plataformas.






	Apple iOS 9	**Android 7.0**	**Windows Phone 8**	**BlackBerry 10**
Compañía	Apple	Open Handset Alliance	Microsoft	BlackBerry
Núcleo del SO	Mac OS X	Linux	Windows NT	QNX
Licencia de *software*	Propietaria	Libre y abierto	Propietaria	Propietaria
Año de lanzamiento	2007	2008	2010	1999
Fabricante único	Sí	No	No	Sí
Variedad de dispositivos	Modelo único	Muy alta	Media	Baja
Soporte memoria externa	No	Sí	Sí	Sí
Motor del navegador web	WebKit	WebKit/ Chromium (Blink)	Trident	WebKit
Tienda de aplicaciones	App Store	Google Play	Windows Marketplace	BlackBerry World
Número de aplicaciones*	2.400.000 (sept. 2016)	2.000.000 (jun. 2016)	700.000 (oct. 2016)	270.000 (2016)
Coste publicar	$99 / año	$25 una vez	$99 / año	Sin coste
Otras tiendas sin supervisión	No	Si	No	Si
Familia CPU soportada	ARM	ARM, MIPS, x86	ARM	ARM
Soporte 64 bits	Si	Si	No	No
Máquina virtual	No	Dalvik / ART	.net	No
Lenguaje de programación	Objective-C, Swift	Java, C++, Kotlin	C#, Visual Basic, C++	C, C++, Java
Plataforma de desarrollo	Mac	Windows, Mac, Linux	Windows	Windows, Mac
Multiusuario	No	Si	No	No
Modo invitado	Si	Si	No	No

Tabla 1: *Comparativa de las principales plataformas móviles (*Fuente www.statista.com).*

Otro aspecto fundamental a la hora de comparar las plataformas móviles es su cuota de mercado. En la siguiente gráfica podemos ver un estudio realizado por la empresa Gartner Group, donde se muestra la evolución del mercado de los sistemas operativos para móviles según el número de terminales vendidos. Podemos destacar la desaparición de la plataforma Symbian de Nokia, el declive continuo de BlackBerry, el estancamiento de la plataforma de Windows, que parece que no despega, y el afianzamiento de la cuota de mercado de Apple en torno al 15 %. En la gráfica se puede apreciar como Apple consigue anualmente un aumento significativo de ventas coincidiendo con el lanzamiento de un nuevo terminal. Finalmente, cabe señalar el espectacular ascenso de la plataforma Android, que en seis años ha alcanzado una cuota de mercado superior al 80 %.

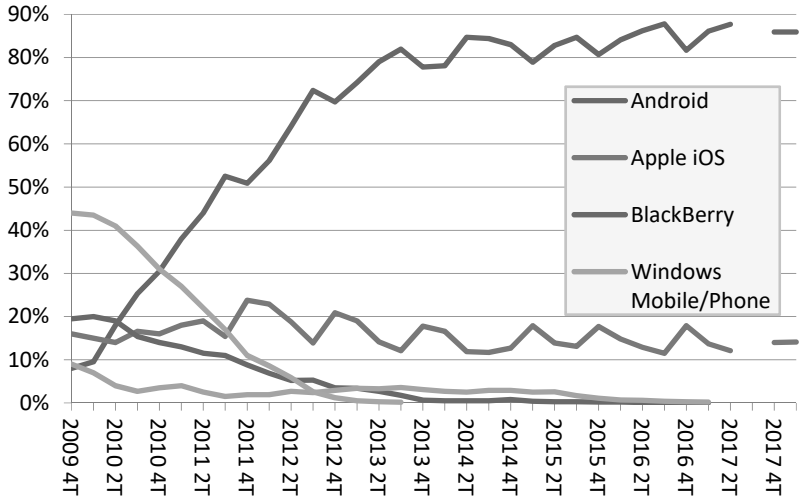


Figura 1: *Porcentaje de teléfonos inteligentes vendidos en todo el mundo, hasta el primer trimestre de 2018, según su sistema operativo (fuente: Gartner Group).*

Vídeo[tutorial]: *Comparativa de las plataformas para móviles*

Preguntas de repaso*: Plataformas para móviles*

1.4. Arquitectura de Android

El siguiente gráfico muestra la arquitectura de Android. Como se puede ver, está formada por cuatro capas. Una de las características más importantes es que todas las capas están basadas en *software* libre.

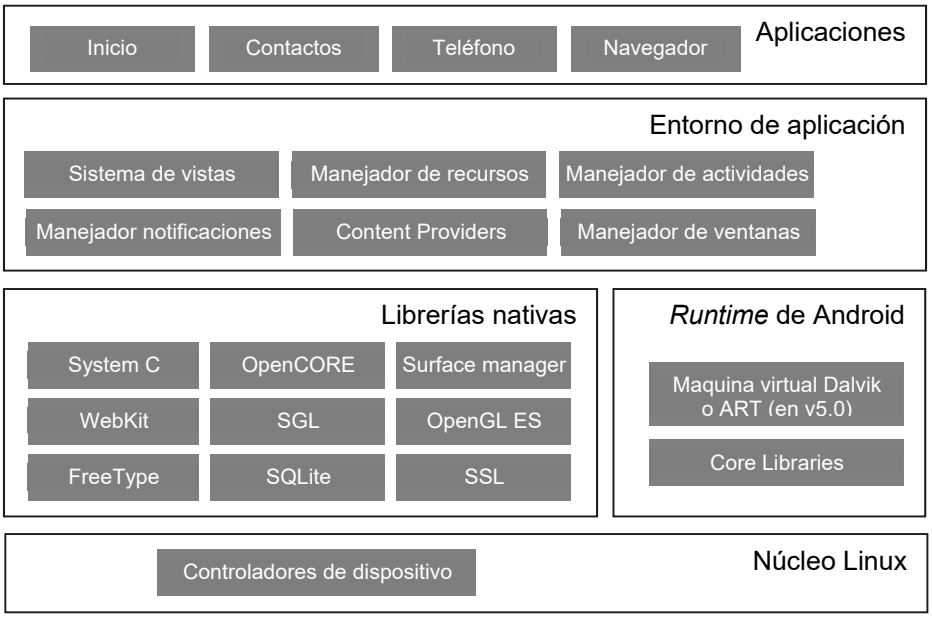


Figura 2: *Arquitectura de Android.*

1.4.1. El núcleo Linux

El núcleo de Android está formado por el sistema operativo Linux, versión 2.6. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de *drivers* para dispositivos.

Esta capa del modelo actúa como capa de abstracción entre el *hardware* y el resto de la pila. Por lo tanto, es la única dependiente del *hardware*.

1.4.2. *Runtime* de Android

Está basado en el concepto de máquina virtual utilizado en Java. Dadas las limitaciones de los dispositivos donde ha de correr Android (poca memoria y procesador limitado), no fue posible utilizar una máquina virtual Java estándar. Google tomó la decisión de crear una nueva, la máquina virtual Dalvik, que respondiera mejor a estas limitaciones.

Entre las características de la máquina virtual Dalvik que facilitan esta optimización de recursos se encuentra la ejecución de ficheros Dalvik ejecutables (*.dex*) –formato optimizado para ahorrar memoria–. Además, está basada en registros. Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual Dalvik. Delega al kernel de Linux algunas funciones como *threading* y el manejo de la memoria a bajo nivel.

A partir de Android 5.0 se reemplaza Dalvik por ART. Esta nueva máquina virtual consigue reducir el tiempo de ejecución del código Java hasta en un 33 %.

También se incluye en el *runtime* de Android el módulo Core Libraries, con la mayoría de las librerías disponibles en el lenguaje Java.

1.4.3. Librerías nativas

Incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android. Están compiladas en código nativo del procesador. Muchas de las librerías utilizan proyectos de código abierto. Algunas de estas librerías son:

- **System C library**: una derivación de la librería BSD de C estándar (libc), adaptada para dispositivos embebidos basados en Linux.
- **Media Framework**: librería basada en OpenCORE de PacketVideo. Soporta códecs de reproducción y grabación de multitud de formatos de audio y vídeo e imágenes MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.
- **Surface Manager**: maneja el acceso al subsistema de representación gráfica en 2D y 3D.
- **WebKit/Chromium**: soporta el navegador web utilizado en Android y en la vista *WebView*. En la versión 4.4, WebKit ha sido reemplazada por Chromium/Blink, que es la base del navegador Chrome de Google.
- **SGL**: motor de gráficos 2D.
- **Librerías 3D**: implementación basada en OpenGL ES 1.0 API. Las librerías utilizan el acelerador *hardware* 3D si está disponible, o el *software* altamente optimizado de proyección 3D.
- **FreeType**: fuentes en *bitmap* y renderizado vectorial.
- **SQLite**: potente y ligero motor de bases de datos relacionales disponible para todas las aplicaciones.
- **SSL**: proporciona servicios de encriptación *Secure Socket Layer* (capa de conexión segura).

1.4.4. Entorno de aplicación

Proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones (sensores, localización, servicios, barra de notificaciones, etc.).

Esta capa ha sido diseñada para simplificar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas (sujetas a las restricciones de seguridad). Este mismo mecanismo permite a los usuarios reemplazar componentes.

Los servicios más importantes que incluye son:

- **Views**: extenso conjunto de vistas (parte visual de los componentes).
- **Resource Manager**: proporciona acceso a recursos que no son en código.
- **Activity Manager**: maneja el ciclo de vida de las aplicaciones y proporciona un sistema de navegación entre ellas.

- **Notification Manager**: permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.
- **Content Providers**: mecanismo sencillo para acceder a datos de otras aplicaciones (como los contactos).

Una de las mayores fortalezas del entorno de aplicación de Android es que se aprovecha el lenguaje de programación Java. El SDK de Android no acaba de ofrecer para su estándar todo lo disponible del entorno de ejecución Java (JRE), pero es compatible con una fracción muy significativa de este.

1.4.5. Aplicaciones

Este nivel está formado por el conjunto de aplicaciones instaladas en una máquina Android. Todas las aplicaciones han de correr en la máquina virtual ART para garantizar la seguridad del sistema.

Normalmente las aplicaciones Android están escritas en Java. Para desarrollar aplicaciones en Java podemos utilizar el Android SDK. Existe otra opción consistente en desarrollar las aplicaciones utilizando C/C++. Para esta opción podemos utilizar el Android NDK (*Native Development Kit*)[2].

Vídeo[tutorial]: *La arquitectura de Android*

Preguntas de repaso*: La arquitectura de Android*

1.5. Instalación del entorno de desarrollo

Google ha preparado el paquete de *software* **Android SDK**, que incorpora todas las herramientas necesarias para el desarrollo de aplicaciones en Android. En él se incluye: conversor de código, depurador, librerías, emuladores, documentación, ejemplos de código, etc. Todas estas herramientas son accesibles desde la línea de comandos.

No obstante, la mayoría de los desarrolladores prefieren utilizar un IDE (entorno de desarrollo integrado). Un IDE agrupa, en un entorno visual, un editor de código con todas las herramientas de desarrollo. Google recomienda utilizar Android Studio (basado en el IDE IntelliJ IDEA).

[2] Para más información consultar *el Gran Libro de Android Avanzado*

1.5.1. Instalación de la máquina virtual Java

Las aplicaciones Android están escritas en Java, por lo que necesitas instalar un *software* para ejecutar código Java en tu equipo. Este *software* se conoce como máquina virtual Java, entorno de ejecución Java, Java Runtime Environment (JRE) o Java Virtual Machine (JVM).

Es muy posible que ya tengas instalada la máquina virtual Java en tu equipo. Si es así, puedes pasar directamente a uno de los apartados siguientes. En caso de dudas, puedes pasar también al punto siguiente. Al concluirlo te indicará si la versión de la máquina virtual Java es incorrecta. En caso necesario, regresa a este punto para instalar una que sea adecuada.

Para instalar la máquina virtual Java accede a http://www.java.com/es/download/, descarga e instala el fichero correspondiente a tu sistema operativo.

1.5.2. Instalación de Android Studio

En la edición de Google I/O 2014 se lanzó la primera versión estable de Android Studio. Se trata de un entorno de desarrollo para Android basado en el IDE IntelliJ IDEA. Entre las novedades introducidas destacamos:

- Construcción de proyectos usando la herramienta Gradle.
- Previsualización simultánea de un layout en varios tipos de dispositivos.
- Facilidades para el testeo de código basado en JUnit.
- Integración con herramientas de gestión de versiones (como GitHub).
- Desarrollo en un mismo proyecto de diferentes versiones (como Android Wear, Android TV y Android Auto).

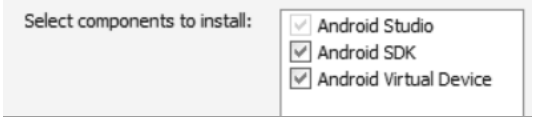
Ejercicio: *Instalación de Android Studio*

NOTA: *Puedes encontrar una descripción más detallada de la instalación en https://developer.android.com/studio/install.html*

1. Descarga el paquete correspondiente a tu versión de la siguiente dirección:

   http://developer.android.com/sdk/

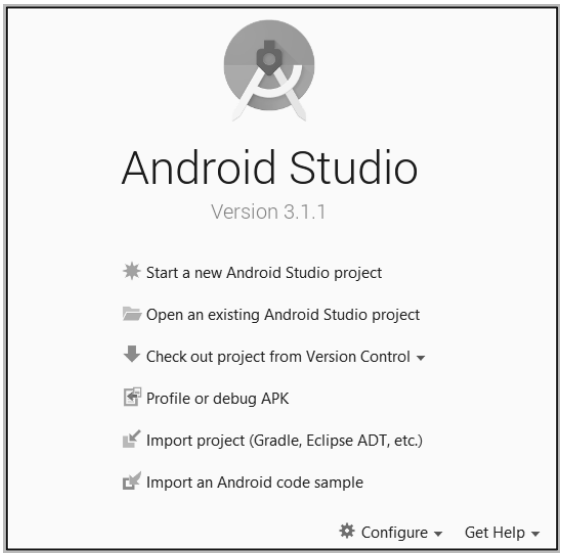
2. Ejecuta el fichero obtenido en el paso anterior:
3. Selecciona todos los componentes a instalar y pulsa *Next*.



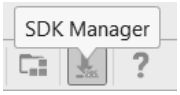
4. Acepta el contrato de licencia y selecciona las carpetas donde quieres instalar el IDE Android Studio y el SDK. En el resto de ventanas puedes utilizar las opciones por defecto. En la última ventana indica que quieres arrancar Android Studio.

5. Primero te preguntará si quieres importar la configuración desde una instalación anterior. Luego verificará si hay actualizaciones del SDK.

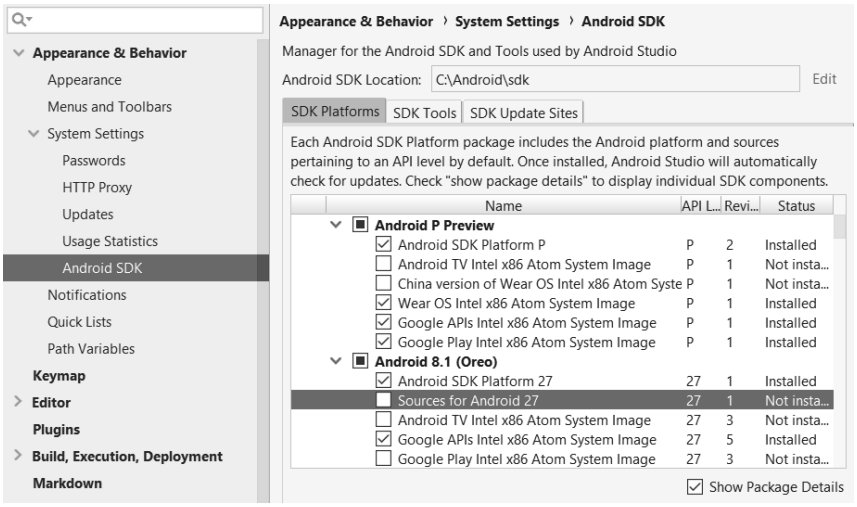
6. Tras pulsar en *Finish* pasamos a la ventana de bienvenida:



7. Comienza pulsando en *Configure*. Aparecerán varias opciones, selecciona *SDK Manager*. Esta herramienta es de gran utilidad para verificar si existen actualizaciones del SDK o nuevas versiones de la plataforma. Podrás acceder a ella desde la ventana principal de Android Studio pulsando en el botón *SDK Manager*:



8. Al entrar en el SDK Manager te muestra los paquetes instalados y los que puedes instalar o actualizar:



En la lengüeta *SDK Plataforms* se muestran los paquetes de plataforma. Pulsa en *Show Package Details* para ver los diferentes paquetes. Siempre es conveniente que tengas instalados los siguientes paquetes de la última plataforma disponible:

- *Android SDK Platform X (donde X es la última versión disponible)*
- *Sources for Android X (no es imprescindible)*
- *Google APIs … System Image (para crear emuladores con Google APIs)*
- *Google Play … System Image (para crear emuladores con Google APIs + Google Play)*

En la lengüeta *SDK Tools* se muestran paquetes con herramientas de la plataforma. Siempre es conveniente que tengas actualizados los siguientes paquetes:

- *Android SDK Build-Tools*
- *Android SDK Platform-tools*
- *Android SDK Tools*
- *Google Play services*
- *Support Repository*

Recursos adicionales: *Teclas de acceso rápido en Android Studio*

Alt-Intro: Solución rápida (Ej. añade *imports* de las clases no resueltas).
Shift-F10 (**Ctrl-R** en Mac): Ejecuta el proyecto.
Shift-F9 (**Ctrl-D** en Mac): Depura el proyecto.
Shift-F6: Cambia el nombre de un identificador.
Ctrl-Alt-L (**Option-Cmd-L** en Mac): Formatea automáticamente el código.
Ctrl-Q (**F1** en Mac): Muestra documentación del código.
Ctrl-P: Muestra parámetros del método seleccionado.
F4 (**Cmd-↓** en Mac): Salta a declaración.
Ctrl-Y (**Cmd-Espacio** en Mac): Borra línea.
Alt-Insert (**Cmd-N** en Mac): Inserta método.

Enlaces de interés: *Conoce Android Studio*

https://developer.android.com/studio/intro/index.html?hl=es-419

Preguntas de repaso: *Instalación y entorno de desarrollo*

1.5.3. Creación de un dispositivo virtual Android (AVD)

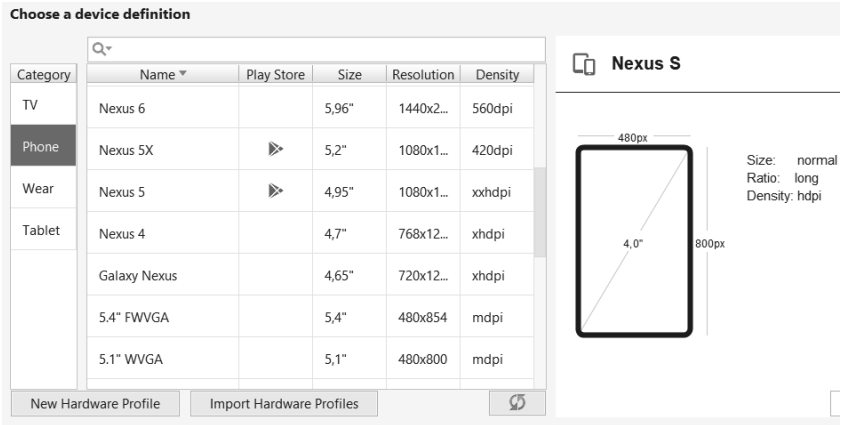
Un dispositivo virtual Android (AVD) te va a permitir emular en tu ordenador diferentes tipos de dispositivos basados en Android. De esta forma podrás probar tus aplicaciones en una gran variedad de teléfonos, tabletas, relojes o TV con cualquier versión de Android, tamaño de pantalla o tipo de entrada.

Ejercicio: *Creación de un dispositivo virtual Android (AVD)*

1. Pulsa el botón *AVD Manager*:

   Aparecerá la lista con los AVD creados. La primera vez estará vacía.

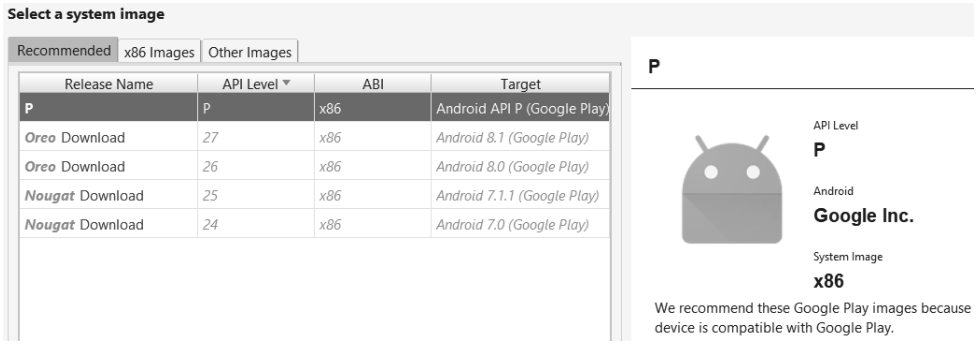
2. Pulsa a continuación el botón *Create Virtual Device...* para crear un nuevo AVD. Aparecerá la siguiente ventana:



3. En la primera columna podremos seleccionar el tipo de dispositivo a emular (móvil, tableta, dispositivo *wearable* o Google TV). A la derecha, se muestran distintos dispositivos que emulan dispositivos reales de la familia Nexus y también otros genéricos. Junto al nombre de cada dispositivo, se indica si tiene la posibilidad de incorporar Google Play, el tamaño de la pantalla en pulgadas, la resolución y el tipo de densidad gráfica. ***NOTA:*** *Los tipos de pantalla se clasifican en Android según su densidad gráfica: ldpi, mdpi, hdpi, xhdpi, ... Véase sección 2.6 Recursos alternativos.*

   Si quisieras añadir a esta lista crear un nuevo tipo de dispositivo, puedes seleccionar *New Hardware Profile.* Podrás indicar las principales características del dispositivo y ponerle un nombre. Usando *Clone Device* podrás crear un nuevo tipo de AVD a partir del actual. Pulsando con el botón derecho sobre un tipo de dispositivo podrás eliminarlos o exportarlos a un fichero.

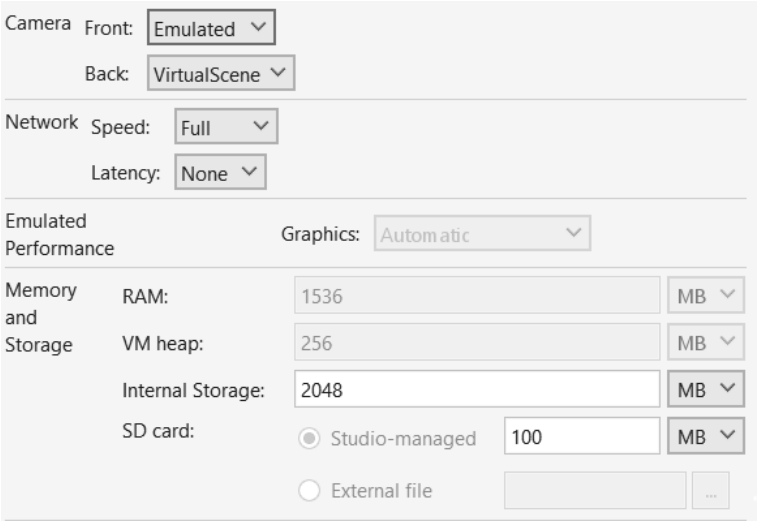
4. Pulsa *Next* para pasar a la siguiente ventana, donde podrás seleccionar la imagen del sistema que tendrá el dispositivo y el tipo de procesador:



Observa cómo las distintas versiones de Android se pueden seleccionar, solo con el código abierto de Android, añadiendo las API de Google (para utilizar servicios como Google Maps) o incluso incorporando Google Play (para poder instalar apps desde la tienda de Google).

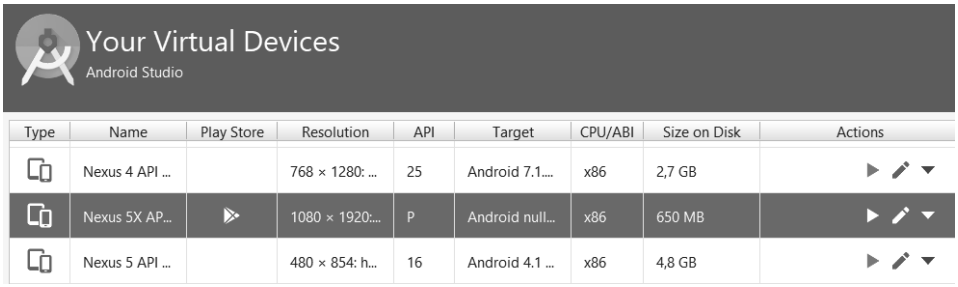
5. Pulsa *Next* para pasar a la última ventana. Se nos mostrará un resumen con las opciones seleccionadas; además, podremos seleccionar la orientación inicial del AVD, si queremos usar el coprocesador gráfico (GPU) de nuestro ordenador o si queremos que dibuje un marco alrededor del emulador simulando un dispositivo real.

6. Pulsa en el botón *Show Avanced Settings* para que se muestren algunas configuraciones adicionales:

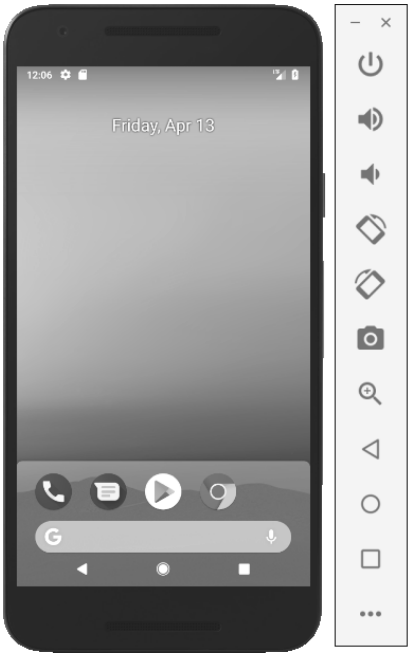


Podemos hacer que el emulador utilice la cámara o teclado de nuestro ordenador. También podemos limitar la velocidad y latencia en el acceso a la red. Finalmente, podremos ajustar la memoria utilizada: RAM total del dispositivo, memoria dinámica usada por Java y memoria para almacenamiento, tanto interna como externa.

7. Una vez introducida la configuración deseada, pulsa el botón *Finish*. Aparecerá el dispositivo creado en la lista:



8. Para arrancarlo, pulsa el botón con forma de triángulo verde que encontrarás en la columna de la derecha. Es posible que te pregunte por la entrada de vídeo para emular la cámara del AVD.



NOTA: *Algunas características de* hardware *no están disponibles en el emulador; por ejemplo, el* multi-touch *o los sensores.*

Vídeo[tutorial]: *Creación de dispositivos virtuales (AVD)*

Recursos adicionales: *Teclas de acceso rápido en un emulador*

Inicio: Tecla *Home*.
F2: Tecla *Menú*.
Esc: Tecla de volver.
F7: Tecla *On/Off*
Ctrl-F5/Ctrl-F6 o **KeyPad +/-**: Control de volumen de audio.
Ctrl-F11 o **KeyPad 7**: Cambia la orientación entre horizontal y vertical.

1.6. Las versiones de Android y niveles de API

Antes de empezar a programar en Android hay que elegir la versión del sistema para la que deseamos realizar la aplicación. Es muy importante observar que hay clases y métodos que están disponibles a partir de una versión; si las vamos a usar, hemos de conocer la versión mínima necesaria.

Cuando se ha lanzado una nueva plataforma, siempre ha sido compatible con las versiones anteriores. Es decir, solo se añaden nuevas funcionalidades, y en el caso de modificar alguna funcionalidad, no se elimina, sino que se etiqueta como obsoleta, pero normalmente se puede continuar utilizando.

A continuación se describen las plataformas lanzadas hasta la fecha, con una breve descripción de las novedades introducidas. Las plataformas se identifican de tres formas alternativas: versión, nivel de API y nombre comercial. El nivel de API corresponde a números enteros, comenzando desde 1. Para los nombres comerciales se han elegido postres en orden alfabético: Cupcake (v1.5), Donut (v1.6), Éclair (v2.0), Froyo (v2.2), Gingerbread (v2.3), etc. Las dos primeras versiones, que hubieran correspondido a las letras A y B, no recibieron nombre.

Vídeo[tutorial]: *Descripción de las versiones de Android*

1.6.1. Las primeras versiones

Android 1.0 Nivel de API 1 (septiembre 2008)

Primera versión de Android. Nunca se utilizó comercialmente, por lo que no tiene mucho sentido desarrollarla para esta plataforma.

Android 1.1 Nivel de API 2 (febrero 2009)

No se añadieron apenas funcionalidades: simplemente se arreglaron algunos errores de la versión anterior. Es la opción a escoger si queremos desarrollar una aplicación compatible con todos los dispositivos Android. No obstante, apenas existen usuarios con esta versión.

1.6.2. Cupcake

Android 1.5 Nivel de API 3 (abril 2009)

Es la primera versión con algún usuario, aunque en la actualidad apenas quedan. Como novedades, se incorpora la posibilidad de teclado en pantalla con predicción de texto (ya no es necesario que los terminales tengan un teclado físico), así como la capacidad de grabación avanzada de audio y vídeo. También aparecen los *widgets* de escritorio y *live folders*. Incorpora soporte para Bluetooth estéreo, por lo que permite conectarse automáticamente a auriculares Bluetooth. Las transiciones entre ventanas se realizan mediante animaciones.

1.6.3. Donut

Android 1.6 Nivel de API 4 (septiembre 2009)

Permite capacidades de búsqueda avanzada en todo el dispositivo. También se incorpora *gestures* y la síntesis de texto a voz. Asimismo, se facilita que una aplicación pueda trabajar con diferentes densidades de pantalla. Soporte para resolución de pantallas WVGA. Aparece un nuevo atributo XML, `onClick`, que puede especificarse en una vista. Soporte para CDMA/EVDO, 802.1x y VPNs.

1.6.4. Éclair

Android 2.0 Nivel de API 5 (octubre 2009)

Esta versión de API apenas cuenta con usuarios, dado que la mayoría de los fabricantes pasaron directamente de la versión 1.6 a la 2.1. Como novedades cabría destacar que incorpora una API para manejar el Bluetooth 2.1. Ofrece un servicio centralizado de manejo de cuentas. Se aumenta el número de tamaños de ventana y resoluciones soportadas. Nueva interfaz del navegador y soporte para HTML5. Mejoras en el calendario y soporte para Microsoft Exchange. La clase `MotionEvent` ahora soporta eventos en pantallas multitáctil.

Android 2.1 Nivel de API 7 (enero 2010)

Se considera una actualización menor, por lo que la siguieron llamando Éclair. Destacamos el reconocimiento de voz, que permite introducir un campo de texto dictando sin necesidad de utilizar el teclado. También permite desarrollar fondos de pantalla animados. Se puede obtener información sobre la señal de la red actual que posea el dispositivo. En el paquete WebKit se incluyen nuevos métodos para manipular bases de datos almacenadas en Internet.

1.6.5. Froyo

Android 2.2 Nivel de API 8 (mayo 2010)

Como característica más destacada se puede indicar la mejora de velocidad de ejecución de las aplicaciones (ejecución del código de la CPU de 2 a 5 veces más rápido que en la versión 2.1, de acuerdo con varios *benchmarks*). Esto se consigue con la introducción de un nuevo compilador JIT de la máquina Dalvik.

Se añaden varias mejoras relacionadas con el navegador web, como el soporte de Adobe Flash 10.1 y la incorporación del motor Javascript V8 utilizado en Chrome.

El desarrollo de aplicaciones permite las siguientes novedades: se puede preguntar al usuario si desea instalar una aplicación en un medio de almacenamiento externo (como una tarjeta SD), como alternativa a la instalación en la memoria interna del dispositivo; las aplicaciones se actualizan de forma automática cuando aparece una nueva versión; proporciona un servicio para la copia de seguridad de datos que se puede realizar desde la propia aplicación para garantizar al usuario el mantenimiento de sus datos; y por último, se facilita que las aplicaciones interaccionen con el reconocimiento de voz y que terceras partes proporcionen nuevos motores de reconocimiento.

Se mejora la conectividad: ahora podemos utilizar nuestro teléfono para dar acceso a Internet a otros dispositivos (*tethering*), tanto por USB como por Wi-Fi. También se añade el soporte a Wi-Fi IEEE 802.11n y notificaciones *push*.

Se añaden varias mejoras en diferentes componentes: en la API gráfica OpenGL ES; por ejemplo, se pasa a soportar la versión 2.0. Para finalizar, permite definir modos de interfaz de usuario («automóvil» y «noche») para que las aplicaciones se configuren según el modo seleccionado por el usuario.

1.6.6. Gingerbread

Android 2.3 Nivel de API 9 (diciembre 2010)

Debido al éxito de Android en las nuevas tabletas, ahora soporta mayores tamaños de pantalla y resoluciones (WXGA y superiores).

Incorpora una nueva interfaz de usuario con un diseño actualizado. Dentro de las mejoras de la interfaz de usuario destacamos la mejora de la funcionalidad de cortar, copiar y pegar y un teclado en pantalla con capacidad multitáctil. Se incluye soporte nativo para varias cámaras, pensado en la segunda cámara usada en videoconferencia. La incorporación de esta segunda cámara ha propiciado la inclusión de reconocimiento facial para identificar al usuario del terminal.

La máquina virtual Dalvik introduce un nuevo recolector de basura que minimiza las pausas de la aplicación, ayudando a garantizar una mejor animación y el aumento de la capacidad de respuesta en juegos y aplicaciones similares. Se

trata de corregir, así, una de las lacras de este sistema operativo móvil, que en versiones previas no ha sido capaz de cerrar bien las aplicaciones en desuso. Se dispone de un mayor apoyo para el desarrollo de código nativo (NDK). También se mejora la gestión de energía y el control de aplicaciones, y se cambia el sistema de ficheros, que pasa de YAFFS a ext4.

Entre otras novedades destacamos: el soporte nativo para telefonía sobre Internet VoIP/SIP; el soporte para reproducción de vídeo WebM/VP8 y codificación de audio AAC; el soporte para la tecnología NFC; las facilidades en el audio, los gráficos y las entradas para los desarrolladores de juegos; el soporte nativo para más sensores (como giroscopios y barómetros), y un gestor de descargas para las descargas largas.

1.6.7. Honeycomb

Android 3.0 Nivel de API 11 (febrero 2011)

Para mejorar la experiencia de Android en las nuevas tabletas se lanza la versión 3.0 optimizada para dispositivos con pantallas grandes. La nueva interfaz de usuario ha sido completamente rediseñada con paradigmas nuevos para la interacción y navegación. Entre las novedades introducidas destacan: los *fragments*, con los que podemos diseñar diferentes elementos de la interfaz de usuario; la barra de acciones, donde las aplicaciones pueden mostrar un menú siempre visible; las teclas físicas son reemplazadas por teclas en pantalla; se mejoran las notificaciones, arrastrar y soltar y las operaciones de cortar y pegar.

La nueva interfaz se pone a disposición de todas las aplicaciones, incluso las construidas para versiones anteriores de la plataforma. Esto se consigue gracias a la introducción de librerías de compatibilidad[3] que pueden ser utilizadas en versiones anteriores a la 3.0.

Se mejoran los gráficos 2D/3D gracias al renderizador OpenGL acelerado por *hardware*. Aparecerá el nuevo motor de gráficos Renderscript, que saca mayor rendimiento al *hardware* e incorpora su propia API. Se incorpora un nuevo motor de animaciones mucho más flexible, conocido como animación de propiedades.

Primera versión de la plataforma que soporta procesadores multinúcleo. La máquina virtual Dalvik ha sido optimizada para permitir multiprocesado, lo que permite una ejecución más rápida de las aplicaciones, incluso aquellas que son de hilo único.

Se incorporan varias mejoras multimedia, como listas de reproducción M3U a través de HTTP Live Streaming, soporte a la protección de derechos musicales (DRM) y soporte para la transferencia de archivos multimedia a través de USB con los protocolos MTP y PTP.

En esta versión se añaden nuevas alternativas de conectividad, como las nuevas API de Bluetooth A2DP para *streaming* de audio y HSP para conexiones

[3] http://developer.android.com/tools/support-library

seguras con dispositivos. También, se permite conectar teclados completos por USB o Bluetooth.

Se mejora el uso de los dispositivos en un entorno empresarial. Entre las novedades introducidas destacamos las nuevas políticas administrativas con encriptación del almacenamiento, caducidad de contraseña y mejoras para administrar los dispositivos de empresa de forma eficaz.

A pesar de la nueva interfaz gráfica optimizada para tabletas, Android 3.0 es compatible con las aplicaciones creadas para versiones anteriores.

Android 3.1 Nivel de API 12 (mayo 2011)

Se permite manejar dispositivos conectados por USB (tanto *host* como dispositivo). Protocolo de transferencia de fotos y vídeo (PTP/MTP) y de tiempo real (RTP).

Android 3.2 Nivel de API 13 (julio 2011)

Optimizaciones para distintos tipos de tableta. Zum compatible para aplicaciones de tamaño fijo. Sincronización multimedia desde SD.

1.6.8. Ice Cream Sandwich

Android 4.0 Nivel de API 14 (octubre 2011)

La característica más importante es que se unifican las dos versiones anteriores (2.x para teléfonos y 3.x para tabletas) en una sola compatible con cualquier tipo de dispositivo. A continuación destacamos algunas de las características más interesantes.

Se introduce una nueva interfaz de usuario totalmente renovada; por ejemplo, se reemplazan los botones físicos por botones en pantalla (como ocurría en las versiones 3.x). Nueva API de reconocimiento facial que, entre otras muchas aplicaciones, permite al propietario desbloquear el teléfono. También se mejora en el reconocimiento de voz; por ejemplo, se puede empezar a hablar sin esperar la conexión con el servidor.

Aparece un nuevo gestor de tráfico de datos por Internet, donde podremos ver el consumo de forma gráfica y donde podemos definir los límites de ese consumo para evitar cargos inesperados con la operadora. Incorpora herramientas para la edición de imágenes en tiempo real, para distorsionar, manipular e interactuar con la imagen en el momento de ser capturada. Se mejora la API para comunicaciones por NFC y la integración con redes sociales.

En diciembre de 2011 aparece una actualización de mantenimiento (versión 4.0.2) que no aumenta el nivel de API.

Android 4.0.3 Nivel de API 15 (diciembre 2011)

Se introducen ligeras mejoras en algunas API, incluyendo las de redes sociales, calendario, revisor ortográfico, texto a voz y bases de datos, entre otras. En marzo de 2012 aparece la actualización 4.0.4.

1.6.9. Jelly Bean

Android 4.1 Nivel de API 16 (julio 2012)

En esta versión se hace hincapié en mejorar un punto débil de Android: la fluidez de la interfaz de usuario. Con este propósito se incorporan varias técnicas: sincronismo vertical, triple búfer y aumento de la velocidad del procesador al tocar la pantalla.

Se mejoran las notificaciones con un sistema de información expandible personalizada. Los *widgets* de escritorio pueden ajustar su tamaño y hacerse sitio de forma automática al situarlos en el escritorio. El dictado por voz puede realizarse sin conexión a Internet (de momento, solo en inglés).

Se introducen varias mejoras en Google Search. Se potencia la búsqueda por voz con resultados en forma de ficha. La función Google Now permite utilizar información de posición, agenda y hora en las búsquedas.

Se incorporan nuevos soportes para usuarios internacionales, como texto bidireccional y teclados instalables. Para mejorar la seguridad, las aplicaciones son cifradas. También se permiten actualizaciones parciales de aplicaciones.

Android 4.2 Nivel de API 17 (noviembre 2012)

Una de las novedades más importantes es que podemos crear varias cuentas de usuario en el mismo dispositivo. Aunque esta característica solo está disponible en tabletas. Cada cuenta tendrá sus propias aplicaciones y su propia configuración.

Los *widgets* de escritorio pueden aparecer en la pantalla de bloqueo. Se incorpora un nuevo teclado predictivo deslizante al estilo Swype. Posibilidad de conectar dispositivo y TVHD mediante Wi-Fi (Miracast). Mejoras menores en las notificaciones. Nueva aplicación de cámara que incorpora la funcionalidad Photo Sphere para hacer fotos panorámicas inmersivas (en 360º).

Android 4.3 Nivel de API 18 (julio 2013)

Esta versión introduce mejoras en múltiples áreas. Entre ellas los *perfiles restringidos* (disponible solo en tabletas), que permiten controlar los derechos de los usuarios para ejecutar aplicaciones específicas y para tener acceso a datos específicos. Igualmente, los programadores pueden definir restricciones en las *apps*, que los propietarios pueden activar si quieren. Se da soporte para Bluetooth Low Energy (BLE), que permite a los dispositivos Android comunicarse con los periféricos con bajo consumo de energía. Se agregan nuevas características para la codificación, transmisión y multiplexación de datos multimedia. Se da soporte para OpenGL ES 3.0. Se mejora la seguridad para gestionar y ocultar las claves privadas y credenciales.

1.6.10. KitKat

Android 4.4 Nivel de API 19 (octubre 2013)

Aunque se esperaba la versión 5.0 y con el nombre de Key Lime Pie, Google sorprendió con el cambio de nombre, que se debió a un acuerdo con Nestlé para asociar ambas marcas.

El principal objetivo de la versión 4.4 es hacer que Android esté disponible en una gama aún más amplia de dispositivos, incluyendo aquellos con tamaños de memoria RAM de solo 512 MB. Para ello, todos los componentes principales de Android han sido recortados para reducir sus requerimientos de memoria, y se ha creado una nueva API que permite adaptar el comportamiento de la aplicación en dispositivos con poca memoria.

Más visibles son algunas nuevas características de la interfaz de usuario. El modo de inmersión en pantalla completa oculta todas las interfaces del sistema (barras de navegación y de estado), de tal manera que una aplicación puede aprovechar el tamaño de la pantalla completa. *WebViews* (componente de la interfaz de usuario para mostrar las páginas web) se basa ahora en el *software* de Chrome de Google y, por lo tanto, puede mostrar contenido basado en HTM5.

Se mejora la conectividad con soporte de NFC para emular tarjetas de pago tipo HCE, varios protocolos sobre Bluetooth y soporte para mandos infrarrojos. También se mejoran los sensores para disminuir su consumo y se incorpora un sensor contador de pasos.

Se facilita el acceso de las aplicaciones a la nube con un nuevo marco de almacenamiento. Este marco incorpora un tipo específico de *content provide*r conocido como *document provider*, nuevas intenciones para abrir y crear documentos y una ventana de diálogo que permite al usuario seleccionar ficheros. Se incorpora un administrador de impresión para enviar documentos a través de Wi-Fi a una impresora. También se añade un *content provider* para gestionar los SMS.

Desde una perspectiva técnica, hay que destacar la introducción de la nueva máquina virtual ART, que consigue tiempos de ejecución muy superiores a la máquina Dalvik. Sin embargo, todavía está en una etapa experimental. Por defecto se utiliza la máquina virtual Dalvik, y se permite a los programadores activar opcionalmente ART para verificar que sus aplicaciones funcionan correctamente.

Vídeo[tutorial]: *Android 4.4 KitKat*

1.6.11. Lollipop

Android 5.0 Nivel de API 21 (noviembre 2014)

La novedad más importante de Lollipop es la extensión de Android a nuevas plataformas, incluyendo Android Wear, Android TV y Android Auto. Hay un cambio significativo en la arquitectura, al utilizar la máquina virtual ART en lugar de Dalvik. Esta novedad ya había sido incorporada en la versión anterior a modo de prueba. ART mejora de forma considerable el tiempo de ejecución del código escrito en Java. Además se soporta dispositivos de 64 bits en procesadores ARM, x86, y MIPS. Muchas aplicaciones del sistema (Chrome, Gmail, …) se han incorporado en código nativo para una ejecución más rápida.

Desde el punto de vista del consumo de batería, hay que resaltar que en Lollipop el modo de ahorro de batería se activa por defecto. Este modo desconecta algunos componentes en caso de que la batería esté baja. Se incorpora una nueva API (*android.app.job.JobScheduler*) que nos permite que ciertos trabajos se realicen solo cuando se cumplan determinadas condiciones (por ejemplo con el dispositivo cargando). También se incluyen completas estadísticas para analizar el consumo que nuestras aplicaciones hacen de la batería.

En el campo Gráfico Android Lollipop incorpora soporte nativo para OpenGL ES 3.1. Además esta versión permite añadir a nuestras aplicaciones un paquete de extensión con funcionalidades gráficas avanzadas (fragment shader, tessellation, geometry shaders, ASTC, …).

Otro aspecto innovador de la nueva versión lo encontramos en el diseño de la interfaz de usuario. Se han cambiado los iconos, incluyendo los de la parte inferior (Retroceder, Inicio y Aplicaciones), que ahora son un triángulo, un círculo y un cuadrado. El nuevo enfoque se centra en Material Design (http://www.google.com/design/material-design.pdf). Consiste en una guía completa para el diseño visual, el movimiento y las interacciones a través de plataformas y dispositivos. Google pretende aplicar esta iniciativa a todas las plataformas, incluyendo wearables y Google TV. La nueva versión también incluye varias mejoras para controlar las notificaciones. Ahora son más parecidas a las tarjetas de Google Now y pueden verse en la pantalla de bloqueo.

Se incorporan nuevos sensores como el de pulso cardiaco, el de inclinación (para reconocer el tipo de actividad del usuario), y sensores de interacción compuestos para detectar ciertos gestos.

Como curiosidad la nueva versión introduce un modo de bloqueo que impide al usuario salir de una aplicación y bloquea las notificaciones. Esto podría utilizarse, por ejemplo, para que mientras un usuario realiza un examen, no pueda ver las notificaciones, acceder a otras aplicaciones, o volver a la pantalla de inicio.

Vídeo[tutorial]: *Android 5.0 Lollipop*

Android 5.1 Nivel de API 22 (marzo 2015)

Se añaden algunas mejoras a nivel de usuario en los ajustes rápidos. A nivel de API se añade soporte para varias tarjetas SIM en un mismo teléfono; la clase AndroidHttpClient se marca como obsoleta; y se añade un API para que las empresas proveedoras de servicios de telecomunicación puedan distribuir *software* de forma segura a través de Google Play. La característica más interesante es que para poder acceder a esta API la aplicación ha de estar firmada con un certificado que coincida con el que el usuario tiene en su tarjeta UICC.

1.6.12. Marshmallow

Android 6.0 Nivel de API 23 (octubre 2015)

Una de las novedades más interesantes es el administrador de permisos. Los usuarios podrán conceder o retirar ciertos permisos a cada aplicación. Con esto el sistema da mucha más protección a la privacidad de los usuarios.

Ahora, el sistema realiza una copia de seguridad automática de todos los datos de las aplicaciones. Esto resulta muy útil al cambiar de dispositivo o tras restaurar valores de fábrica. Para disponer de esta funcionalidad simplemente usa el *targer* Android 6.0. No es necesario agregar código adicional.

Android 6.0 integra el asistente por voz Now on Tap. Es una evolución de Google Now más integrada con las aplicaciones. Se activa con pulsación larga de home. Aparecerán tarjetas sobre la aplicación actual y lo que muestra. La aplicación actual podrá aportar información al asistente. En esta misma línea, se añade un API que permite interacciones basadas en voz. Es decir, si nuestra aplicación ha sido lanzada por voz, podremos solicitar una confirmación de voz del usuario, seleccionar de una lista de opciones o cualquier información que necesite.

Se introducen los enlaces de aplicación con los que podremos asociar la aplicación que abre una URL en función de su dominio web. Aunque muchos dispositivos ya lo permitían, en esta actualización se añade autentificación por huella digital a la API. Tu aplicación puede autentificar al usuario usando las credenciales para desbloquear su dispositivo (pin, patrón o contraseña). Esto libera al usuario de tener que recordar contraseñas específicas de la aplicación. Y te evita tener que implementar tu propia interfaz de autenticación.

Compartir con otros usuarios ahora es más fácil con Direct Share. Permite no solo escoger la aplicación con la que compartes, sino también el usuario. Si tu aplicación es un posible destino para compartir vas a poder indicar al sistema la lista de usuarios que pueden recibir información.

En Android 6.0 podemos utilizar parte de un dispositivo de almacenamiento externo, para que sea usado como almacenamiento interno. Podemos fragmentar, formatear y encriptar una tarjeta SD para ser usada como memoria interna. También podemos montar y extraer lápices de memoria USB de forma nativa.

Se incorpora la plataforma de pagos abierta *Android Pay* que combina NFC y Host Card Emulation. El nuevo gestor de batería, Doze, realiza un uso más eficiente de los recursos cuando el dispositivo está en reposo, con lo que podemos obtener dos horas extras de autonomía. Se da soporte de forma nativa a pantallas 4 K, lápices Bluetooth, múltiples tarjetas SIM y linterna. Mejoras de posicionamiento utilizando redes WiFi y dispositivos Bluetooth.

Vídeo[tutorial]: *Android 6.0 Marshmallow*

1.6.13. Android Nougat

Android 7.0 Nivel de API 24 (julio 2016)

Ahora los usuarios pueden abrir varias aplicaciones al mismo tiempo en la pantalla. Puedes configurar tu aplicación para que se visualice con unas dimensiones mínimas o inhabilitar la visualización de ventanas múltiples.

Las notificaciones han sido rediseñadas para un uso más ágil. Hay más opciones para personalizar el estilo de los mensajes (`MessageStyle`). Puedes agrupar notificaciones por temas o programar una respuesta directa.

En la versión anterior se utilizaba una estrategia de compilación *Ahead of Time* (AOT): cuando se descargaba una aplicación, su código era traducido de *bytecodes* a código nativo, lo que mejoraba los tiempos de ejecución. En la nueva versión se incorpora también la compilación *Just in Time* (JIT), donde no se compila hasta que el código va a ser ejecutado. Android 7.0 propone un planteamiento mixto según el perfil del código. Los métodos directos se compilan previamente (AOT), mientras que otras partes no se compilan hasta que se usan (JIT). Aunque AOT puede introducir retardos en ejecución, ahorra tiempo en la precompilación y en memoria. El mayor impacto de esta técnica se nota en la instalación de las aplicaciones y actualizaciones del sistema. Mientras que en Android 6.0 una actualización podría usar varios minutos, ahora se instala en cuestión de segundos.

Android Nougat incorpora la plataforma de realidad virtual *Daydream*. Se trata de una propuesta de Google que complementa la iniciativa *Cardboard*. Incluye especificaciones *software* y *hardware* que nos permitirán diferenciar a los dispositivos compatibles. Los principales fabricantes de móviles se han unido a esta iniciativa.

En la versión anterior, el gestor de batería Doze solo se activaba cuando el dispositivo estaba en reposo. Ahora, se activa poco tiempo después de apagarse la pantalla. Esto permite ahorrar batería cuando llevamos el dispositivo en el bolsillo.

También se ha añadido la nueva API para gráficos 3D, Vulcan, como alternativa a OpenGL. Minimiza la sobrecarga de CPU en el controlador, lo que permite aumentar la velocidad de los juegos.

El usuario va a poder activar el modo de ahorro de datos cuando se encuentre en itinerancia o cuando esté a punto de agotar un paquete de datos. En este caso, tanto el sistema como las aplicaciones han de tratar de minimizar al máximo las transferencias de datos.

Android 7.1 Nivel de API 25 (diciembre 2016)

La principal novedad son los accesos directos a aplicaciones. Desde el icono de la aplicación, con una pulsación prolongada, aparecen varias opciones que podremos seleccionar. Por ejemplo, podremos iniciar una navegación privada con Chrome de forma directa. Los accesos directos que quieras incorporar a tu

aplicación, los podrás configurar por medio de *intens*, que deben especificarse en un fichero de configuración[4].

Se incorporan otras novedades como la posibilidad de insertar imágenes desde el teclado, de la misma forma que ahora insertamos emoticonos.

Vídeo[tutorial]: *Android 7.0*

1.6.14. Android Oreo

Android 8.0 Nivel de API 26 (agosto 2017)

Destacan las siguientes mejoras en seguridad: se introduce Google Play Protect, que escanea regularmente las aplicaciones en busca de *malware*. La opción "Orígenes desconocidos" desaparece. Ahora podemos indicar qué aplicaciones pueden instalar apks y cuáles no. Desde la opción “Acceso especial de aplicaciones” podemos configurar qué aplicaciones pueden realizar ciertas acciones.

El sistema limita más los procesos en segundo plano para conseguir ahorro en la batería. Se mejora el tiempo de arranque del sistema.

Pensando en los países emergentes, se lanza Android Go: Una distribución adaptada para dispositivos de gama baja (1 GB de RAM o menos). Se preinstalan apps ligeras y en Google Play Store destacan aplicaciones ligeras adecuadas para estos dispositivos. Estas aplicaciones han de cubrir 3 requisitos: trabajar sin red, pesar menos de 10 MB y proporcionar un buen rendimiento de batería.

Con el fin de reducir la fragmentación de Android, aparece el proyecto Treble, que facilitará las actualizaciones a los fabricantes. Se reestructura la arquitectura de Android para definir una interfaz clara entre la capa del Núcleo Linux (con sus *drivers*) y las capas del Framework. Esto permite actualizar Android sin tener que tocar la capa del Núcleo Linux.

Las notificaciones presentan varias mejoras: Podemos añadir color de fondo. Se ordenan por importancia. Las aplicaciones pueden crear canales de notificaciones y el usuario decidir cuáles quiere recibir. Podemos posponer una notificación o verlas pulsando sobre el icono de la aplicación.

Los iconos tendrán que estar diseñados en dos capas: El icono y el fondo del icono. Esto permite adaptarse al dispositivo. Además, el usuario podrá escoger entre iconos circulares, cuadrados o de esquinas redondeadas.

Ahora podemos reproducir un vídeo en una ventana flotante mientras utilizamos otras aplicaciones. Al seleccionar un texto se nos sugieren acciones cuando se trata de un número de teléfono o una dirección. El Autocompletar de Google, que

[4] https://developer.android.com/guide/topics/ui/shortcuts.html

antes estaba disponible en Chrome para guardar contraseñas, ahora se puede usar en cualquier aplicación Android.

1.6.15. Android Pie

Android 9.0 Nivel de API 28 (agosto 2018)

Una de las novedades más interesantes es el nuevo API WiFi RTT introducido en IEEE 802.11mc. Permite estimar la distancia entre nuestro dispositivo y los puntos de acceso cercanos, lo que permite sistemas de posicionamiento en interiores con una precisión de 1 a 2 metros. Otro importante cambio es la navegación por gestos. Se reemplazan los tres botones en pantalla (triangulo, círculo y cuadrado) por solo 2 (retroceder e inicio). El botón de inicio admite diferentes gestos que nos permite ir al asistente de Google, cambiar entre apps recientes o abrir el menú de apps.

Una interesante innovación es el uso de Inteligencia Artificial, para mejorar diferentes aspectos. La idea consiste en aprender nuestros hábitos a la hora de usar las aplicaciones. Con esta información se puede quitar preferencia sobre el uso de la CPU a las apps menos utilizadas, consiguiendo una reducción de hasta un 30 %. Este menor uso de la CPU prolongará la vida de la batería. Usando técnicas similares se pretende aprender cuando el usuario va a arrancar una aplicación o una acción de esta. De esta forma el sistema puede cargar en memoria la aplicación antes incluso que el usuario decida utilizarla.

Se introducen algunas mejoras que fomentan un uso responsable y saludable del móvil. Por ejemplo, desde el Dashboard podemos consultar el uso que hacemos cada día, en cada aplicación. Podemos establecer alarmas de uso excesivo muy interesantes para el control parental. En esta línea, se introducen nuevos modos de relajación y no molestar para favorecer la desconexión digital.

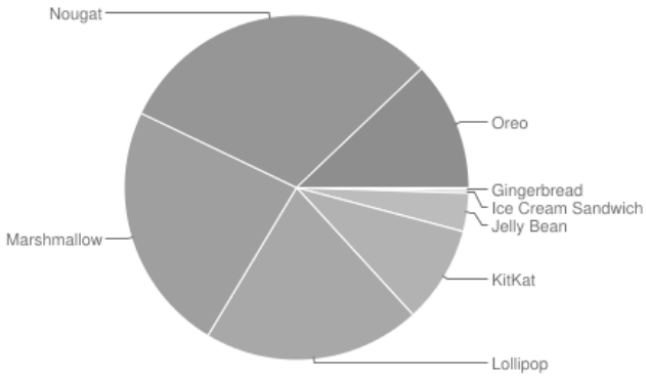
Preguntas de repaso: *Las versiones de Android*

1.6.16. Elección de la plataforma de desarrollo

Vídeo[tutorial]: *Elegir la versión en una aplicación Android*

A la hora de seleccionar la plataforma de desarrollo hay que consultar si necesitamos alguna característica especial que solo esté disponible a partir de una versión. Todos los usuarios con versiones inferiores a la seleccionada no podrán instalar la aplicación. Por lo tanto, es recomendable seleccionar la menor versión

posible que nuestra aplicación pueda soportar. Por ejemplo, si en nuestra aplicación queremos utilizar el motor de animaciones de propiedades, tendremos que utilizar la versión 3.0, al ser la primera que lo soporta. El problema es que la aplicación no podrá ser instalada en dispositivos que tengan una versión anterior a la 3.0. Para ayudarnos a tomar la decisión de qué plataforma utilizar, puede ser interesante consultar los porcentajes de utilización:



Plataforma	Nivel API	Porcent.
2.3 Gingerbread	10	0,2 %
4.0 Ice Cream S. 4.1 Jelly Bean 4.2 4.3 4.4 KitKat	15 16 17 18 19	0,3 % 1,2 % 1,9 % 0,5 % 9,1 %
5.0 Lollipop 5.1	21 22	4,2 % 16,2 %
6.0 Marshmallow	23	23,5 %
7.0 Nougat 7.1	24 25	21,2 % 9,6 %
8.0 Oreo 8.1	26 27	10.1 % 2.0 %

Figura 3: *Dispositivos Android, según la plataforma instalada, que han accedido a Google Play Store el 27 de julio de 2018 y los 7 días anteriores. Las versiones con porcentajes inferiores al 0,1 % no se muestran.*

Tras estudiar la gráfica podemos destacar el reducido número de usuarios que utilizan la versión 2.3 (0,2 %). Por lo tanto, puede ser buena idea utilizar como versión mínima la 4.1, 4.2 o 4.4 para desarrollar nuestro proyecto, dado que daríamos cobertura al 99%, 98% o 96% de los terminales. Las versiones 3.x han tenido muy poca difusión, por lo que no aparecen en la tabla. Las versiones 6.0 y 7.x son mayoritarias. La versión 8.x todavía no dispone de un número importante de usuarios. No obstante, estas cifras cambian mes a mes, por lo que recomendamos consultar los siguientes enlaces antes de tomar decisiones sobre las versiones a utilizar.

Enlaces de interés:

- ***Android Developers: Plataform Versions:*** Estadística de dispositivos Android, según la plataforma instalada, que han accedido a Android Market.

  http://developer.android.com/about/dashboards/index.html

- ***Android Developers:*** En el menú de la izquierda aparecen enlaces a las principales versiones de la plataforma. Si pulsas sobre ellos, encontrarás una descripción exhaustiva de cada plataforma.

  http://developer.android.com/about/index.html

Preguntas de repaso: *Elegir una versión de Android*

1.6.17. Las librerías de compatibilidad (*support library*)

Tal y como se ha descrito, la filosofía tradicional de Android ha sido que las novedades que aparecen en una API solo puedan usarse en dispositivos que soporten esa API. Como acabamos de ver, la fragmentación de las versiones de Android es muy grande, es decir, actualmente podemos encontrar dispositivos con una gran variedad de versiones. Con el fin de que la aplicación pueda ser usada por el mayor número posible de usuarios hemos de ser muy conservadores a la hora de escoger la versión mínima de API de nuestra aplicación. La consecuencia es que las novedades que aparecen en las últimas versiones de Android no pueden ser usadas.

En la versión 3.0 aparecieron importantes novedades que Google quería que se incorporaran en las aplicaciones lo antes posible (*fragments*, nuevas notificaciones, etc.). Con este fin creó las librerías de compatibilidad para poder incorporar ciertas funcionalidades en cualquier versión de Android[5]. Veamos algunas de ellas:

Vídeo[tutorial]: *Las librerías de compatibilidad (suport library)*

v4 Support Library

Esta librería permitía utilizar muchas clases introducidas en la versión 3.0 cuando trabajábamos con un API mínimo anterior. En la actualidad ya no es necesaria utilizarla, dado que ya es recomendable utilizar como API mínimo la versión 4.0 o, incluso, superior. Puede usarse en una aplicación con nivel de API 4 (v1.6) o superior. Incorpora las clases: `Fragment`, `NotificationCompat`, `LocalBroadcastManager`, `ViewPager`, `PagerTitleStrip`, `PagerTabStrip`, `DrawerLayout`, `SlidingPaneLayout`, `ExploreByTouchHelper`, `Loader` y `FileProvider`. Para más información, consúltese la referencia de `android.support.v4`.

v7 Libraries

Se incluyen las siguientes librerías que pueden usarse a partir del API 7 (v2.1):

- **v7 appcompat library**: Permite utilizar un IU basado en la Barra de Acciones siguiendo especificaciones de *material design*. Se añade por defecto cuando creamos un nuevo proyecto. Incorpora las clases: `ActionBar`, `AppCompatActivity`, `AppCompatDialog` y `ShareActionProvider`.
- **v7 recyclerview library**: Incorpora la vista `RecyclerView`, una versión mejorada que reemplaza a `ListView` y `GridView`.

[5] https://developer.android.com/tools/support-library/setup.html

- **v7 cardview library**: Incorpora la vista `CardView`, una forma estándar de mostrar información especialmente útil en Android Wear y TV.
- **v7 gridlayout library**: Incorpora el *layout* `GrigLayout`.
- **v7 preference support library**: Incorpora las clases `CheckBoxPreference` y `ListPreference` usadas en preferencias.
- **v7 palette library**: Incorpora la clase `Palette`, que permite extraer los colores principales de una imagen.
- **v7 mediarouter library**: Da soporte a Google Cast.

v8 Support Library

Añade soporte para utilizar RenderScript. Esta API permite paralelizar tareas en dispositivos con varias CPU o entre la CPU y la GPU. Esto resulta especialmente útil en el procesado de imágenes.

v13 Support Library

Un *helper* da soporte a la clase `FragmentCompat` para acceder a varias características de un *fragment*.

v14 Preference Support Library

Permite incorporar las últimas novedades incluidas en las preferencias. Define las clases `MultiSelectListPreference` y `PreferenceFragment`.

v17 Preference Support Library for TV

Incorpora preferencias para TV.

v17 Leanback Library

Incorpora importantes widgets usados en aplicaciones para TV: `BrowseFragment`, `DetailsFragment`, `PlaybackOverlayFragment` y `SearchFragment`.

Design Support Library

Librería que incorpora varios componentes de *material design*.

Percent Support Library

Podemos utilizar dimensiones basadas en porcentajes en nuestros diseños.

Annotations Support Library

Permite añadir metadatos al código fuente disponibles en tiempo de ejecución.

Custom Tabs Support Library

Permite el diseño personalizado de interfaces de usuario basados en pestañas.

App Recommendation Support Library for TV

Recomendaciones de contenido en aplicaciones para TV.

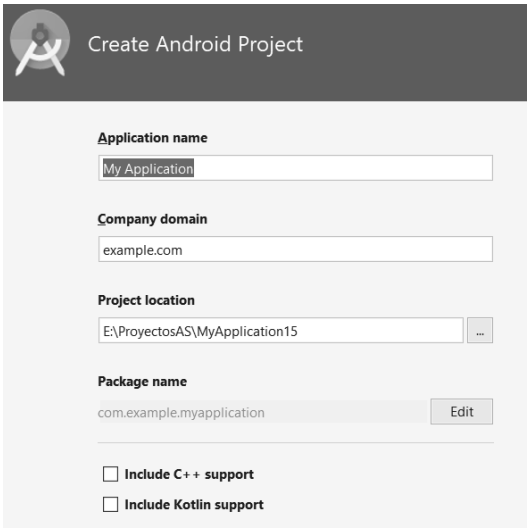
1.7. Creación de un primer programa

Utilizar un entorno de desarrollo nos facilita mucho la creación de programas. Esto es especialmente importante en Android dado que tendremos que utilizar una gran variedad de ficheros. Gracias a Android Studio, la creación y gestión de proyectos se realizará de forma muy rápida, acelerando los ciclos de desarrollo.

Ejercicio: *Crear un primer proyecto*

Para crear un primer proyecto Android, con Android Studio sigue los siguientes pasos:

1. Selecciona *File* > *New* > *New Project...*
2. Rellena los detalles del proyecto. Puedes dejar los valores por defecto:



A continuación vemos una descripción para cada campo:

Application name: Es el nombre de la aplicación que aparecerá en el dispositivo Android. Tanto en la barra superior, cuando esté en ejecución, como en el icono que se instalará en el menú de programas.

Company Domain: Indicamos el dominio web utilizado por nosotros o nuestra empresa. Tal y como se muestra en la siguiente línea, a partir de este dominio creará un nombre de paquete invirtiendo el orden de los campos y añadiendo el nombre de la aplicación. Las clases Java que creemos pertenecerán a este paquete. Indicar el nombre de paquete de esta forma, resulta un proceso algo extraño para un programador Java. Si lo prefieres puedes dejar el nombre de dominio en blanco y pulsar en el link *Edit* que aparece en la línea *Package name*. Como veremos a lo largo del curso, el nombre del paquete también es utilizado por Android para múltiples funciones. Por ejemplo, para determinar en qué directorio se instala la aplicación.

Nota sobre Java: *El nombre del paquete debe ser único en todos los paquetes instalados en un sistema. Por ello, cuando quieras distribuir una aplicación, es muy importante utilizar un dominio que no puedan estar utilizando otras empresas (por ejemplo:* es.upv.elgranlibroandroid.proyecto1*). El espacio de nombres "com.example" está reservado para la documentación de ejemplos (como en este libro) y nunca puede ser utilizado para distribuir aplicaciones. De hecho, Google Play no permite publicar una aplicación si su paquete comienza por "com.example".*

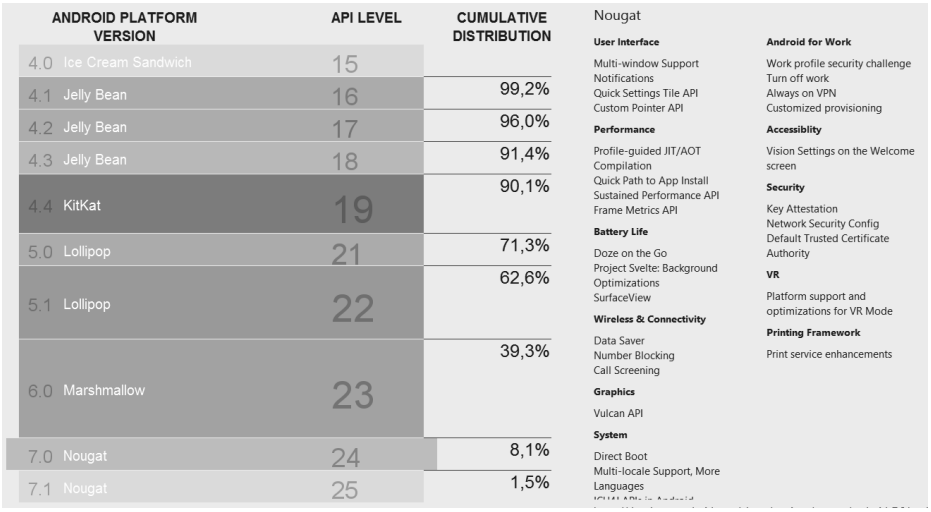
Project location: Permite configurar la carpeta donde se almacenarán todos los ficheros del proyecto.

3. Pulsa *Next* para pasar a la siguiente pantalla.

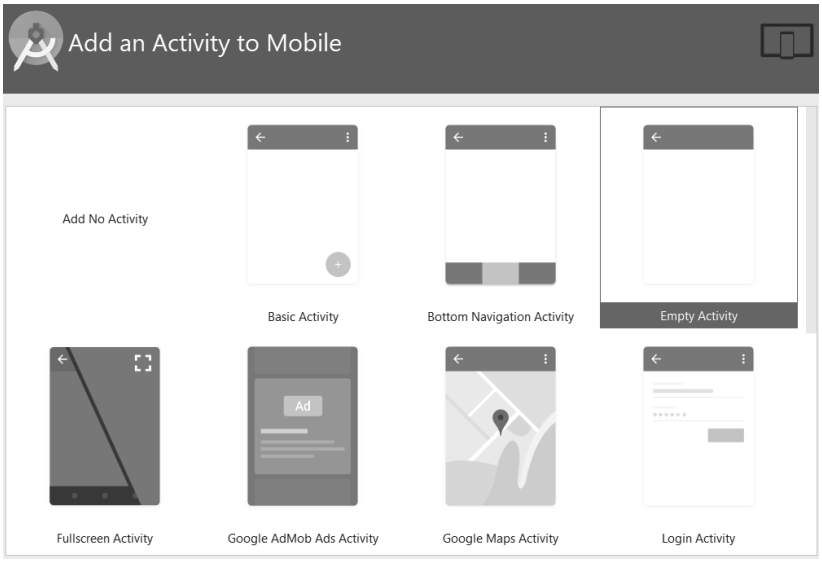
En esta ventana puedes elegir para que dispositivos quieres desarrollar la aplicación. En este libro nos centraremos en el desarrollo de aplicaciones para teléfonos y tabletas, por lo que has de seleccionar el primer chek-boox. Pero has de saber que la plataforma Android también permite desarrollar aplicaciones para dispositivos wearables, Google TV o Android Auto.

Minimum SDK: Este valor especifica el mínimo nivel de la API que requiere tu aplicación. Por lo tanto, la aplicación no podrá ser instalada en dispositivos con una versión inferior. Procura escoger valores pequeños para que tu aplicación pueda instalarse en la mayoría de los dispositivos. Un valor adecuado puede ser el nivel de API 14 (v4.0), dado que cubriría prácticamente el 100% de los dispositivos. O el nivel de API 16 (v4.1), que cubriría más del 99% de los dispositivos. Escoger valores pequeños para este parámetro tiene un inconveniente: no podremos utilizar ninguna de las mejoras que aparezcan en los siguientes niveles de API. Por ejemplo, si queremos utilizar el motor de animaciones de propiedades en nuestra aplicación, tendremos que indicar en este campo la versión 3.0, dado que esta API no aparece hasta esta versión. Pero, en este caso, nuestra aplicación no se podrá instalar en la versión 2.3.

Como se acaba de indicar escoger la versión mínima del SDK es un aspecto clave a la hora de crear un proyecto. Para ayudarnos a tomar esta decisión se indica en negrita el porcentaje de dispositivo donde se podrá instalar nuestra aplicación. En el apartado anterior se explica cómo se obtiene esta información. Pulsa en *Help Me chosee* para visualizar una gráfica donde se muestra los diferentes niveles de API y el porcentaje de usuarios que podrán instalarla la aplicación. Además si pulsas sobre un nivel te mostrará un resumen con las nuevas características introducidas en este nivel.

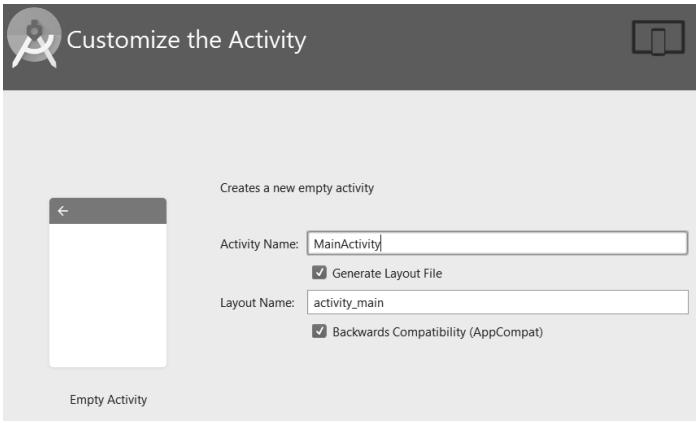


4. Pulsa *Next* para pasar a la siguiente pantalla. Aquí podrás indicar que tipo de actividad inicial quieres en tu aplicación:



El concepto de actividad será explicado más adelante. Selecciona *Empty Activity* para añadir una actividad inicial. Observa cómo puedes elegir otros tipos de actividades que incorporen ciertos elementos de uso habitual, como menús, botones, anuncios, etc.

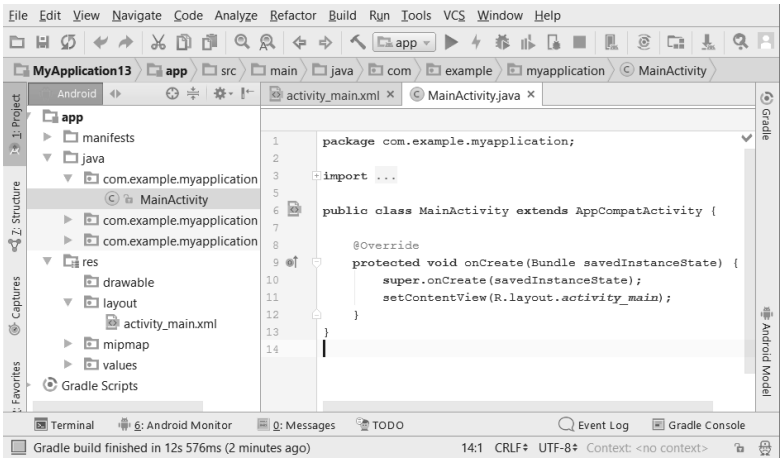
5. Pulsa *Next* para pasar a la siguiente pantalla. Aquí podrás indicar los nombres de diferentes elementos de la actividad inicial:



Activity Name: Nombre de la clase Java que se creará para la actividad.

Layout Name: Nombre del layout donde se diseña su aspecto visual.

6. Deja los valores por defecto y pulsa *Finish* para crear el proyecto. Deberías tener visible el explorador del proyecto (*Project*) a la izquierda. Abre el fichero *MainActivity* (situado en *app > java > com.example.myapplication*). Debe tener este aspecto:



Observa que la clase `MainActivity` extiende `AppCompactActivity` que a su vez es un descendiente de `Activity`. Una ***actividad*** es una entidad de aplicación que se utiliza para representar cada una de las pantallas de nuestra aplicación. Es decir, el usuario interactúa con solo una de estas actividades y va navegando entre ellas. El sistema llamará al método `onCreate()` cuando

comience su ejecución. Es donde se debe realizar la inicialización y la configuración de la interfaz del usuario. Las actividades van a ser las encargadas de interactuar con el usuario.

Nota sobre Java: *Antes de este método se ha utilizado la anotación* `@Override` *(sobrescribir). Esto indica al compilador que el método ya existe en la clase padre y queremos reemplazarlo. Es opcional, aunque conviene incluirlo para evitar errores.*

Lo primero que hay que hacer al sobrescribir un método suele ser llamar al método de la clase de la que hemos heredado. Para referirnos a nuestra clase padre usaremos la palabra reservada `super`. El método termina indicando que la actividad va a visualizarse en una determinada vista. Esta vista está definida en los recursos. Lo veremos un poco más adelante. Más adelante se describe la finalidad de cada fichero y carpeta de este proyecto.

Vídeo[tutorial]: *Un primer proyecto Android*

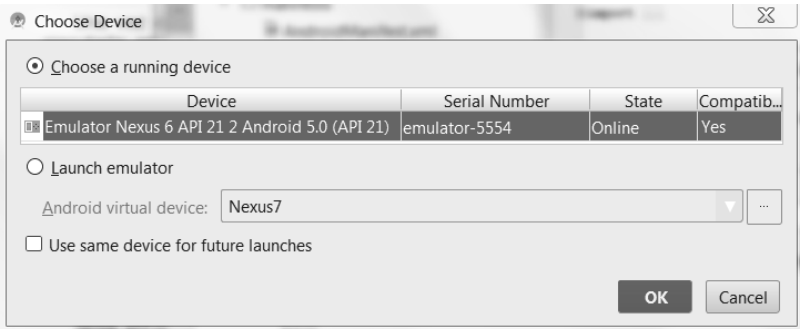
1.8. Ejecución del programa

Una vez creada esta primera aplicación, vamos a ver dos alternativas para ejecutarla: en un emulador y en un dispositivo real.

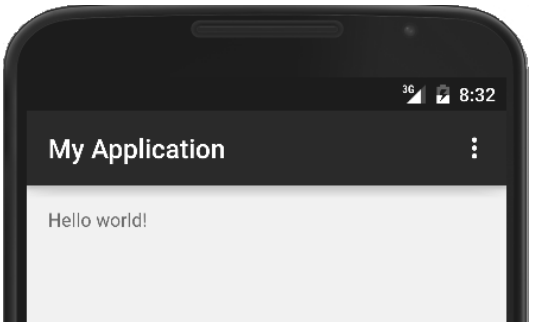
1.8.1. Ejecución en el emulador

Ejercicio: *Ejecución en el emulador*

1. Selecciona *Run* > *Run 'app'* (*Mayús-F10*) o pulsa el icono  de la barra de herramientas.
2. Te preguntará sobre que dispositivo quieres ejecutar la aplicación:



3. Una vez que el emulador esté cargado, debes ver algo así:



NOTA: *La inicialización del emulador puede ser algo lenta, por esta razón es mejor no cerrar el emulador.*

1.8.2. Ejecución en un terminal real

También es posible ejecutar y depurar tus programas en un terminal real. Incluso es una opción más rápida y fiable que utilizar un emulador. No tienes más que usar un cable USB para conectar el terminal al PC. Resulta imprescindible haber instalado un *driver* especial en el PC. Puedes encontrar un *driver* genérico que se encuentra en la carpeta de instalación del SDK *\sdk\extras\google\usb_driver*. Aunque lo más probable es que tengas que utilizar el *driver* del fabricante.

Ejercicio: *Ejecución en un terminal real*

1. Abre *Android SDK Manager* y asegúrate de que está instalado el paquete USB Driver. En caso contrario, instálalo.