

Programación multimedia y dispositivos móviles

UD01. Análisis de tecnologías para aplicaciones en dispositivos móviles

Desarrollo de Aplicaciones Multiplataforma



Anna Sanchis Perales

ÍNDICE

1. OBJETIVOS	3
2. DISPOSITIVOS MÓVILES	4
2.1. Historia y evolución	4
2.2. Tipos	13
2.3. Limitaciones	21
2.4. Lenguajes de programación	23
3. TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES MÓVILES	24
3.1 Web Apps o aplicaciones web responsivas	24
3.2 Aplicaciones híbridas	24
3.3 Aplicaciones Web progresivas (PWA)	25
3.4 Aplicaciones compiladas	25
4. ANDROID	26
4.1. ¿Qué es Android?	26
4.2. Características principales	27
4.3. Arquitectura	29
4.4. Máquina virtual	30
4.5. Versiones	35
4.6. Fragmentación del mercado	37
4.7. Distribución de las aplicaciones	38
5. ENTORNOS INTEGRADOS DE TRABAJO	39
6. ANDROID STUDIO	40
6.1. Instalación	40
6.2. Configuración del entorno	40
6.3. Estructura de un proyecto	43
6.4. Gestión del AVD (Android Virtual Device)	51
6.5. Ejecución del proyecto	54
6.6. ¿Qué es Gradle?	56

1. OBJETIVOS

Los objetivos a conocer y comprender son:

- Historia y evolución de los dispositivos móviles
- Tipos de dispositivos existentes.
- Limitaciones que se plantean en la ejecución de aplicaciones en los dispositivos móviles
- Lenguajes de programación existentes.
- Android: qué es, características, arquitectura, máquina virtual, versiones, fragmentación del mercado y distribución de las aplicaciones.
- Android Studio: instalación, configuración del entorno, estructura de un proyecto, AVD y ejecución de un proyecto.
- Gradle: qué función tiene y para qué sirve.

2. DISPOSITIVOS MÓVILES

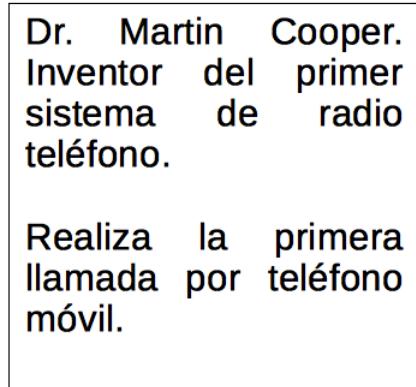
2.1. Historia y evolución

Los dispositivos móviles han cambiado la sociedad actual de una forma tan profunda como lo ha hecho Internet. Los nuevos terminales comienzan a ofrecer capacidades similares a un PC, lo que permite realizar multitud de tareas con ellos: navegar por internet, abrir un documento PDF, jugar, etc.

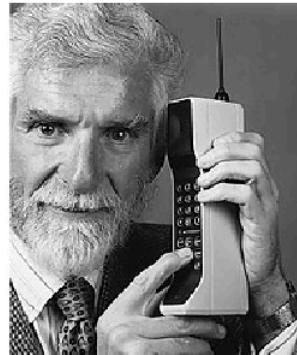
Esto nos lleva a hablar de Computación UbiCua, la cual se entiende como la integración de la informática en el entorno de la persona. Esta se basa en 4 aspectos fundamentales:

- Descentralización: ya no se basa en centralizar todo el proceso de cómputo en una sola máquina, sino que hablamos de arquitectura cliente-servidor. En este tipo de arquitectura la sincronización de la información jugará un papel esencial.
- Adaptabilidad: es necesario ofrecer el mismo servicio a cualquier dispositivo.
- Conectividad: es necesario que los distintos dispositivos se hablen entre ellos.
- Diseño centrado en el usuario: los interfaces de usuario han de ser lo más sencillos e intuitivos posibles.

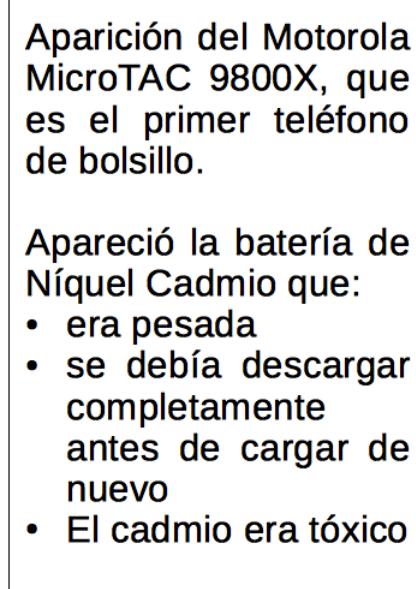
A continuación se muestra en la siguiente figura la historia y evolución de los dispositivos móviles entre los años 1973 y 2014:



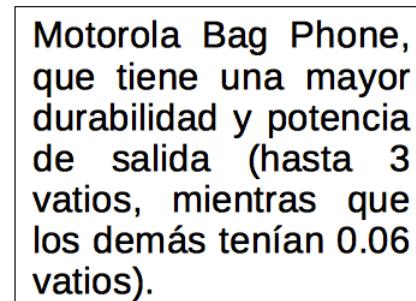
1973



1983



1989



1992



Nokia 1011. Primer teléfono GSM.

1992



IBM Simon. Primer SmartPhone.

1993



Nokia N103 Hyper.
Primer teléfono comercial con *ring tones* personalizables.

1996



Siemens S10. Primer teléfono con pantalla a color.

1997



Nokia 8810. Primer teléfono móvil sin antena exterior.

Apareció la batería de Níquel Metal Híbrido:

- No contiene materiales tóxicos
- Es más liviana
- Permite un mejor control de la temperatura

1998



Nokia 3210. Uno de los más populares. Mas de 160 millones vendidos.

1999



Samsung SPH-M100. Primer teléfono con soporte para música MP3.

1999



Benefon Esc!. Primer teléfono con GPS integrado.

1999



Nokia 7110. Primer teléfono con soporte para aplicaciones WAP.

1999



Móvil japonés, que ofrece una resolución de 0.1 megapixel.

2000



Ericson T39. Primer teléfono con soporte para BlueTooth.

2001



Motorola Razr. Primer teléfono con teclado plano. 50 millones vendidos.

Aparece la batería de Litio:
• Mayor duración
• Menor tiempo de recarga
• Variedad de formas y tamaños

2004



Motorola A845. Primer teléfono modo dual UMTS-WCDMA. Incluye video llamada.

2004



Treo 700W. Primer Palm SmartPhone que opera con Windows Mobiles.

2005



Apple iPhone.
Poderoso ordenador de bolsillo con capacidad multimedia, alta velocidad de acceso web, pantalla multitáctil, etc.

2007



HTC Dream (también conocido como T-Mobile G1). Primer móvil con Android. Dispone de pantalla táctil y teclado QWERTY completo.

Aparición del App Store de Apple y del iPhone 3G.

2008



Aumento de la resolución de pantalla y con ello el tamaño del móvil. Como ejemplo está el Motorola Milestone.

2009



LG Optimus 2X. Primer móvil del mundo que dispone de un procesador dual-core.

2010



Primer móvil Apple en tener procesador dual-core y dos antenas.

2011



HTC EVO 4G. Primer móvil con capacidad 4G. Funciona sobre Android 2.2.

2011



i'mWatch. Primer reloj de muñeca Android. Compatible con Android, IOS, Blackberry y Windows Phone.

2011



Meizu MZ. El primer móvil del mercado con 4 núcleos. Funciona sobre Android 2.3.5.

2012



Samsung Galaxy S III. Cámara de 8 megapíxeles, procesador de 4 núcleos, pantalla HD Super AMOLED de 4.8 pulgadas. El móvil del año.

2012



Nexus 5. Móvil propio de Google. Con GPU y 2 GB de RAM. Android 4.4 Kit Kat. Pantalla Full HD de 5".

2013



Sony Xperia Z1. Pantalla FULL HD de 5". Resistente al polvo y agua. 4 núcleos y 2 GB de RAM. Cámara de 20 megapíxeles.

2013



iPhone 6. Nueva revisión del modelo. Mejoras en la pantalla, procesador, cámara, peso, etc.

2014



Siguiendo con la evolución, llegamos al año 2024 donde la apuesta está en la irrupción de la inteligencia artificial en los dispositivos móviles. La integración de la IA (Inteligencia Artificial) en el mundo de la comunicación permitirá una gran capacidad de generación de contenido automatizado. El dispositivo iPhone 16 dispone de tecnología preparada para la 'Apple Intelligence' mediante el chip A18 que está diseñado específicamente para soportar capacidades de IA generativa. En la siguiente imagen vemos la presentación del iPhone 16 con funcionalidades IA que todavía no están disponibles y que aparecerán en futuras actualizaciones del dispositivo. Dicho dispositivo dispondrá de un asistente Siri impulsado por un 'Large Language Model' (LLM) e integrada con ChatGPT de OpenAI.



2.2. Tipos

Los tipos de dispositivos móviles existentes en el mercado son los siguientes:

PDA (Asistente Personal Digital)

El PDA es un ordenador de mano que inicialmente fue diseñado como una agenda electrónica que tenía las funcionalidades de agenda, lista de contactos, bloc de notas, recordatorios y calculadora, entre otras. Estas funcionalidades le permitían a los usuarios digitalizar la información personal y mantenerla de una forma organizada. El sistema de reconocimiento de escritura que tenían en sus pantallas era una de sus características fundamentales.

En la década de los años 90 aparecieron estas en el año de 1992 la compañía Apple presentó el primer Asistente Personal Digital. Sin embargo, en 1998 dejaron de sacar este dispositivo al

mercado por no tener un producto estable y correctamente desarrollado.

La compañía Palm Inc., nació en el año de 1995 y ayudó con su dispositivo Palm Pilot al desarrollo de las PDA. Tal fue el impulso y el impacto generado en el mercado que se convirtió en una marca registrada y en un nombre genérico.

En la siguiente década del 2000, apareció la empresa Microsoft Corp., con PDAs que trabajan con los sistemas operativos desarrollados por ellos, con mejoras en área de multimedia y conectividad de equipos. El aporte más importante de esta empresa fue el aplicar los mismos conceptos que venían trabajando con los sistemas operativos de los equipos personales y portátiles, por lo que aumentó el número de usuarios de estos equipos.

Una PDA es un dispositivo que combina el tamaño, las funcionalidades de un computador, el teléfono, las conexiones de red, el servicios de posicionamiento GPS y la conexión a internet; también se les llama palmtops, hand held computers y pocket computers .

Las características de las PDA:

- **La Pantalla Táctil:** La pantalla táctil es un característica de entrada de las PDA que sirve de interfaz entre el usuario y el equipo, y le permite ingresar al usuario la información. Generalmente se utiliza un lápiz que permite trabajar con la PDA, aunque también dispone de un teclado virtual para ingresar los caracteres letras o números al dispositivo. Es posible conectar un teclado externo por medio de una conexión inalámbrica, y también se puede utilizar un sistema de reconocimiento de palabras.
- **Las Tarjetas de Memoria:** Las tarjetas de memoria son utilizadas para aumentar el almacenamiento de la información de la PDA. Se utilizan memorias SD, MiniSD, MicroSD, entre otras.
- **La Conectividad Alámbrica:** Las PDA cuentan con puertos de conexión alámbricos que permiten conectarse a otros equipos como los ordenadores de sobremesa o los portátiles. Las conexiones que tienen las PDA son por medio de cable serial y cable USB, este último en especial ya que permite cargar de energía la PDA.
- **La Conectividad Inalámbrica:** Las PDA cuentan con diferentes tecnologías de conexión inalámbrica, entre las que se encuentran el Infrarrojo, Bluetooth, WAP, Wi-Fi, entre otros, las cuales le permiten la conexión de accesorios o dispositivos externos como teclados,

auriculares, equipos de computación y muchos otros más.

- **Reconocimiento de Voz:** Las PDA cuentan con un sistema de reconocimiento de voz, que le permite a los usuarios almacenar notas o recordatorios de sonido y también realizar la escritura en texto de lo que se está hablando.
- **Sincronización:** Unas de las características importantes de las PDA, ya que dispone de la capacidad de sincronizar su software con el de otros equipos, lo cual es una ventaja útil para los usuarios en cuanto a la persistencia y exactitud de información que manejan en estos equipos.

Las PDA se pueden clasificar, de acuerdo al sistema operativo con que trabaja, siguiendo esta clasificación:

Sistema Operativo	Compañía	Dirección Electrónica
Palm OS	Palm	http://www.palm.com/es/es/
Pocket PC	HP- ACER-Dell	http://www.microsoft.com/windowsmobile/en-us/default.mspx
iPhone OS	Apple	http://www.apple.com/iphone/
Android	Android de Google	http://www.android.com/
Windows 10	Windows Corp.	http://www.windowsphone.com/es-es
Linux	Linux	http://www.linux.org/
Symbian OS	Symbian	http://www.symbian.org/
BlackBerry	BlackBerry	http://www.blackberry.com/

Teléfono móvil

El teléfono móvil es un dispositivo inalámbrico electrónico que tiene acceso a la red de telefonía móvil. Su nombre se define por el uso de la red de estaciones base o antenas repetidoras, en la cual cada estación base está compuesta por celdas o células que proveen cobertura en un ángulo y rango determinado.

La principal característica de los teléfonos móviles es la portabilidad y la facilidad de realizar una comunicación desde cualquier lugar en donde se tenga cobertura de la red celular, la comunicación entre los móviles y las redes celulares se realiza a través del espectro electromagnético utilizando las frecuencias o bandas del mismo.

Cuando se realiza una llamada desde un móvil denominado A, este se comunica con la celda de la estación base que le está dando cobertura al móvil A en ese momento, la celda se comunica con

otras celdas y estaciones repetidoras hasta llegar a la celda que está dando cobertura al móvil B (que es el que se está llamando) y esta a su vez envía la comunicación al móvil B de destino y se realiza la comunicación.

La función de los móviles es la comunicación de voz, pero los avances tecnológicos en las diferentes áreas de las comunicaciones para la transmisión de datos, las conexiones a internet y la evolución de los equipos móviles, han generado nuevas características y servicios para los usuarios.

En la actualidad los teléfonos móviles tienen otras características como la cámara fotográfica, la agenda, la reproducción de audio y video, acceso a internet, posicionamiento global GPS, entre muchas otras características que se pueden encontrar en los nuevos celulares.

Según la historia del teléfono móvil, el primer modelo fue desarrollado por la compañía Motorola y tenía el nombre de DynaTAC 8000X, fue diseñado por el ingeniero Rudy Krolopp en el año de 1983.

Smartphone o Teléfono Inteligente

Un Smartphone o teléfono inteligente es un dispositivo electrónico que tiene el funcionamiento de un teléfono móvil con características de un ordenador personal.

Los teléfonos inteligentes tienen diferentes características especiales en tanto al hardware y al software, debido a que sus componentes son desarrollados para realizar tareas que exigen mayor capacidad de procesamiento y memoria.

Las características de hardware y software de los teléfonos inteligentes se encuentran definidas en el uso de un sistema operativo que administra los recursos del equipo, provee seguridad y optimiza las funcionalidades, la conectividad de los equipos a Internet y a diferentes redes utilizando las diferentes tecnologías y estándares de comunicación inalámbricas como Infrarrojo, Bluetooth, WAP, GPRS, Wi-Fi, posicionamiento global GPS, entre otros, administración del correo electrónico, los programas tradicionales de agenda, la administración de contactos, la sincronización con otros equipos, la instalación de aplicaciones, la instalación de juegos, edición de documentos ofimáticos, documentos portables como los PDF, etc. La mayoría de los teléfonos inteligentes tienen características multimedia como las cámaras que permiten grabar videos y tomar fotos, reproducción de archivos de sonido en diferentes formatos, utilizan pantallas táctiles, o teclados

QWERTY, etc, por lo que, en general son herramientas con bastante poder computacional.

Módem Inalámbrico

Un Módem Inalámbrico es un dispositivo electrónico que permite realizar una conexión a internet utilizando la red celular o de telefonía móvil. La función principal del modem es solicitar el permiso de acceso a la red y enviar y recibir datos por medio de las redes.

Los avances tecnológicos en la telefonía móvil digital han permitido la conexión a Internet. Inicialmente se utilizó la tecnología WAP que permitía el acceso a páginas diseñadas especialmente para los equipos móviles. El proceso de conexión se realizaba como una llamada telefónica y de esta forma se transmitían los datos. Luego apareció la tecnología GPRS que permitía acceder a Internet utilizando el protocolo TCP/IP.

Actualmente se puede acceder a los diferentes servicios de internet utilizando los protocolos FTP, de correo electrónico, entre otros, como si se estuviera trabajando desde un ordenador de escritorio o portátil.

Otras tecnologías y avances tecnológicos en la telefonía móvil digital que han permitido el acceso a internet son EDGE, EvDO, HSPA y LTE las cuales permiten el mejoramiento de la navegación en internet y la visualización del contenido multimedia de forma óptima.

La tecnología UMTS de las redes de telefonía móvil digital han permitido desarrollar los modem inalámbricos para los equipos de computación como los computadores de escritorio, los portátiles y los NetBooks, los cuales están implementado en dispositivos electrónicos que se conectan a los equipos por medio de los puertos USB, por tal motivo se conocen como modem inalámbricos o USB.

Tablet

Es un equipo de computación que se encuentra ubicado en el medio de un computador portátil y Smartphone, los Tablet tienen la pantalla táctil la cual es utilizada como una interfaz de ingreso de información, en la cual se puede escribir texto e ingresarlos en el equipo y el usuario puede trabajar con el equipo sin necesidad de utilizar un teclado y un mouse, también existen Tablet PC que se

pueden convertir y utilizar con un teclado y mouse.

Los Tablet utilizan hardware que consumen pocos recursos de energía, es decir los procesadores, las memorias, los discos duros, las pantallas entre otros, tienen la característica especial de diseño para la movilidad y para economizar recursos de energía en el funcionamiento normal del dispositivo, es decir estos dispositivos no están diseñados para el alto rendimiento o para un alto nivel de procesamiento.

El software de estos dispositivos está básicamente ligado al sistema operativo del fabricante del dispositivo, debido a esto las características especiales de estos dispositivos como la escritura en las pantallas, el dibujo, la conexión a internet y otros tipos de redes se encuentran limitadas por las características y permisos que puede proveer el fabricante.

Entre los modelos de tablets más conocidas se puede nombrar los siguientes:

Nombre Dispositivo	Fabricante	Dirección Electrónica
iPad	Apple	http://www.apple.com/ipad/
Galaxy Tab	Samsung	http://www.samsung.com/co/consumer/mobile-phones/mobile-phones/tablet/
iFree Tablet PC	Proyecto Grupo de Investigación EATCO	http://www.ifreetablet.com/
Xiaomi Pad 6	Xiaomi	https://www.mi.com/es/product/xiaomi-pad-6/

E-Book o Libro Electrónico

Los libros electrónicos son un tipo de dispositivo portátil que permite almacenar y leer libros digitalizados, o cualquier otro tipo de documento escrito que contenga imágenes. El objetivo de los E-Book es proveer una alternativa a los periódicos y libros tradicionales que están impresos en papel.

Actualmente estos dispositivos cuentan con características especiales en las pantallas para disminuir el consumo de energía en el dispositivo, las cuales manejan una escala de grises que se parece a la tinta tradicional pero que en realidad es un tinta electrónica que permite visualizar los contenidos de una forma tradicional.

Los E-Book cuentan con conexiones inalámbricas que le permiten conectarse a las diferentes redes que existen actualmente. Algunos de estos dispositivos también permiten navegar en internet o reproducir videos.

A continuación se presentan algunos dispositivos de E-Book:

Dispositivo	Empresa	Dirección Electrónica
Kindle	Amazon	http://www.amazon.com/Kindle-Wireless-Reading-Display-Generation/dp/B0015T963C
Sony Reader	Sony	http://www.sonystyle.com/webapp/wcs/stores/servlet/CategoryDisplay?catalogId=10551&storeId=10151&langId=-1&categoryId=8198552921644523779&N=4294954529
Papyre	Papyre	http://grammata.es/papyre
Nook	Barnes & Noble	http://www.barnesandnoble.com/nook/index.asp

NetBooks

Un Netbook es un ordenador portátil, pequeño y liviano, de bajo costo, el cual está optimizado para navegar por Internet y prestar servicios relacionados con la suite ofimática, los correos electrónicos...

Estos equipos tienen menor capacidad de procesamiento, almacenamiento y no cuentan con algunos componentes de hardware, como por ejemplo la unidad de DVD. Tiene pocos puertos USB y no cuenta con ranuras expandibles.

Las características que debe cumplir un Netbook como tal para ser considerado dentro de este tipo de equipos es que debe tener un tamaño y peso reducido, deben ser eficientes en el manejo de la energía y su costo ha de ser bajo.

Phablet

Un Phablet es un dispositivo móvil que combina las características de un teléfono inteligente y una tableta, es decir se encuentra en el medio de estos dispositivos. La pantalla tiene un tamaño mayor al de un teléfono inteligente y un tamaño menor al de las tabletas.

Otra característica que pueden tener estos dispositivos es el uso de un lápiz óptico para trabajar con el dispositivo, entre los más destacados se encuentran el Galaxy Note I y II.

Reloj Inteligente

Un reloj inteligente o smartwatch es un reloj que cuenta con un sistema operativo móvil con el cual, aparte de poder indicar la hora, dispone de una pantalla multimedia a color o monocromática táctil. Es capaz de conectarse a teléfonos inteligentes y otros dispositivos a través de bluetooth y a internet a través de WiFi. Se pueden instalar aplicaciones y personalizar el reloj según los gustos de los usuarios. Puede conectarse a redes sociales, compartir archivos multimedia, pueden realizar y recibir llamadas, enviar y recibir mensajes de texto, correos electrónicos, reproducir música, monitorear la frecuencia cardiaca, entrenamiento personal, etc.

Las características más interesantes son la inclusión de acelerómetros, termómetros, altímetros, barómetro, brújula, cronógrafo, equipo de buceo, localización GPS, micrófono, modem, manos libres, etc.

Los sistemas operativos del reloj inteligente son versiones personalizadas de los sistemas operativos móviles como Android y iOS (en los casos de los smartwatch más reconocidos actualmente). En otros casos utilizan versiones de los sistemas operativos más livianas que les ayudan a trabajar con estos tipos de reloj.

Lo más interesante es que estos dispositivos pueden interactuar con otros y administrarlos remotamente.

2.3. Limitaciones

El uso de dispositivos móviles ha crecido tanto en la sociedad actual debido, principalmente, a la aparición de soluciones de tratamiento de datos, la movilidad que proporcionan, la interfaz mucho más simple respecto al pc, la facilidad que tienen para conectarse a redes... pero también tiene sus limitaciones, como pueden ser:

- Tamaño de las pantallas: Android al ser un sistema operativo abierto permite que existan muchos modelos con pantallas de tamaños diferentes, divididas en cuatro categorías:
 - Pequeña: 426dp x 320dp
 - Normal: 470dp x 320dp
 - Grande: 640dp x 480dp
 - Extra grande: 960dp x 720dp

En iOS la cosa se facilita: Apple es el único fabricante, por lo que el abanico de tamaños de estos dispositivos es menor.

- Densidades de los dispositivos muy variables (veremos en el siguiente punto a que nos referimos con esta afirmación).
- Gestión de la rotación de la pantalla.
- Arquitecturas diversas:
 - ARM
 - x86
 - MIPS.
- Extras del móvil, que son muy heterogéneos, y que no siempre están disponibles: Acelerómetro, giroscopio, brújula, GPS, Barómetro, Cámaras, etc.
- Conexión a las redes:
 - Wifi
 - Bluetooth
 - NFC
 - GSM/UMTS/HSPA+
 - GSM/EDGE/GPRS (850, 900, 1800, 1900 MHz)
 - 3G/4G/5G

- HSPA+ 42
- Telefonía, SMS y MMS
- Soporte a distintos tipos de teclado:
 - Teclado real / Teclado virtual.
- Pantalla táctil: existen múltiples opciones de las mismas: táctil, multitáctil, con puntero, etc... El problema surge por la limitación de las medidas mínimas que tenemos para controlar la interfaz de usuario, ya que se controlan con los dedos.
- La CPU y la memoria son más limitadas respecto a las máquinas de escritorio.
- La batería, ya que es uno de los puntos débiles de Android. El uso intensivo de CPU reduce el tiempo de la misma considerablemente.

Por ello, a pesar de que dichos dispositivos han ido mejorando con el paso del tiempo las aplicaciones deben ser diseñadas teniendo en cuenta de que existen dichas limitaciones.

Android ha sido criticado muchas veces por la fragmentación que sufren sus terminales (con versiones distintas de Android), ya que las actualizaciones no se despliegan automáticamente en estos terminales una vez que Google lanza una nueva versión. Cada fabricante debe crear su propia versión. Sin embargo, esa situación cambiará ya que los fabricantes se han comprometido a aplicar actualizaciones al menos durante los 18 meses siguientes desde que empiezan a vender un terminal en el mercado.

Además, actualmente Google tiene la intención de unificar la funcionalidad entre las versiones del sistema operativo para tabletas y móviles en la versión 4.0.

Disponer del código fuente del sistema operativo no significa que se pueda tener siempre la última versión de Android en un determinado móvil, ya que el código para soportar el hardware de cada fabricante normalmente no es público; así que faltaría un trozo básico del firmware (controladores) para poder hacerlo funcionar en dicho terminal.

Hay que tener en cuenta que las nuevas versiones de Android suelen requerir más recursos, por lo que, en los modelos más antiguos, no se puede instalar la última versión por insuficiente memoria RAM, velocidad de procesador, etcétera.

2.4. Lenguajes de programación

Los lenguajes de programación para los dispositivos móviles dependen en gran parte del dispositivo en el que se quiera trabajar, sin embargo tienen en común que se pueden crear sistemas visuales robustos con mayor facilidad, independientemente del lenguaje de programación que se esté aplicando. Por supuesto dicho lenguaje debe soportar la metodología de programación con la que trabaja el dispositivo en particular.

Podemos encontrar:

- Java: Para dispositivos móviles la versión de Java en la que se programa es J2ME (Java Micro Edition).
- C++: Una de las herramientas que se utiliza para la programación de móviles basadas en el lenguaje C++ es Carbide. C++ la cual trabaja en la plataforma de Symbian.
- C#: Para desarrollar aplicaciones en dispositivos móviles a través de este lenguaje de programación se puede utilizar la herramienta de Visual C# que trabaja con un IDE para diseñar las ventanas de la aplicación. Está orientado a móviles basados en sistemas operativos Windows. Utiliza el Framework .NET (estructura de soporte para organizar y desarrollar software) de Microsoft, que permite un rápido desarrollo de aplicaciones.
- Objective C: Este lenguaje de programación está basado en C, y a diferencia de C++ es un súper conjunto del lenguaje C, ya que agrega a la sintaxis de C la manera de enviar mensajes en Smalltalk, y de definir e implementar objetos.
- Swift: es un lenguaje de programación multiparadigma creado por Apple que permite desarrollar aplicaciones para iOS, MacOS, watchOS y tvOS. Swift puede usar cualquier biblioteca programada con Objective C y también puede llamar a funciones programadas en en lenguaje C.
- Python: Este lenguaje de programación orientado a objetos es utilizado para dispositivos móviles tales como Symbian, Palm, teléfonos inteligentes de Nokia, etc. También se puede emplear en otras plataformas tales como Windows, Linux/Unix, Mac OS X, etc.

3. TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES MÓVILES

El desarrollo de una aplicación que funcione de forma nativa en un determinado sistema operativo pasa por el desarrollo mediante sus propias tecnologías. Sin embargo, existen varias tecnologías que con sus ventajas e inconvenientes pretenden abarcar el desarrollo multiplataforma en su sentido más amplio, desde aplicaciones web de tipo responsive hasta aplicaciones compiladas, pasando por las aplicaciones web híbridas o progresivas (PWA).

Las aplicaciones nativas son las que se desarrollan específicamente para el sistema en que se van a ejecutar y aprovechan todos sus recursos. Además, permiten acceder a todas las funcionalidades de las plataformas y disponen de cualquier novedad en el sistema de forma inmediata.

Para el desarrollo nativo en Android se usa Java o Kotlin, mientras que para iOS utilizaremos Objective-C y Swift, generando así código nativo de cada plataforma.

Todo ello da como resultado aplicaciones fluidas y con la mejor experiencia de usuario. Como desventaja, suponen un incremento del coste de producción y de mantenimiento cuando deseamos que estén disponibles en múltiples plataformas, ya que debemos llevar un desarrollo paralelo.

3.1 Web Apps o aplicaciones web responsivas

Se trata de aplicaciones basadas en tecnología web: HTML, CSS y JavaScript, y que para ejecutarse necesitan únicamente un navegador web. El hecho de ser responsivas implica que su interfaz se adapte a cualquier dispositivo. Para este tipo de aplicaciones no es necesario desarrollar nada en código nativo, y son totalmente multiplataforma, puesto que para ejecutarse lo hacen sobre el propio navegador web del sistema operativo.

Disponemos, pues, de un único código para ejecutarse en todas las plataformas, con la principal desventaja de que no ofrecen una experiencia de usuario tan buena como las apps nativas ni permiten el acceso a todos los componentes del sistema.

3.2 Aplicaciones híbridas

Utilizan las tecnologías HTML, CSS y JavaScript para construir un sitio web que se carga dentro de un componente WebView, que no es más que un navegador sin la barra de navegación ni otras

opciones, por lo que presentan la apariencia de una aplicación nativa. Este tipo de aplicaciones ya pueden acceder, a través de este componente, a algunas características del dispositivo, como la ubicación o el acelerómetro. Actualmente, el framework para el desarrollo de aplicaciones híbridas más popular es Ionic, que permite el desarrollo en otros frameworks web como **React, Angular o Vue**. Otro framework muy popular en su momento fue Adobe Phonegap, pero Adobe abandonó su desarrollo en 2020, a favor de las aplicaciones web progresivas.

3.3 Aplicaciones Web progresivas (PWA)

Un poco más cerca de las aplicaciones nativas tenemos las aplicaciones web progresivas, que están revolucionando el panorama actual. Estas aplicaciones incrementan y avanzan en las funcionalidades según el dispositivo móvil en el que vaya a actualizarse, para sacar mayor potencial, y pueden acceder al hardware, trabajar sin conexión o con mala conexión u ofrecer notificaciones del sistema. Existen multitud de frameworks para el desarrollo PWA, entre los que se encuentran React PWA Library, Angular PWA Library, Vue PWA Framework, Ionic PWA Framework, Svelte, PWA Builder o Polymer.

3.4 Aplicaciones compiladas

Se trata de tecnologías que pretenden utilizar solamente un lenguaje de programación para generar aplicaciones móviles en el código nativo de cada plataforma. Algunas de las tecnologías más utilizadas en este tipo de aplicaciones son las siguientes:

- **React Native y Native Script.** Utilizan como base el lenguaje de programación JavaScript, pero en lugar de construir interfaces mediante HTML utilizan componentes propios del framework que son compilados a código nativo, haciendo innecesario utilizar un WebView como intermediario.
- **Flutter.** Desarrollado y mantenido por Google, permite el desarrollo de aplicaciones multiplataforma mediante el lenguaje Dart, compilado a código nativo que se ejecuta en el dispositivo.

4. ANDROID

4.1. ¿Qué es Android?

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Symbian y Blackberry OS. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma.

Inicialmente fue desarrollado por Open Handset Alliance (OHA), una agrupación de 78 compañías para desarrollar estándares abiertos para dispositivos móviles y que está liderada por Google, aunque inicialmente Android fue desarrollado por la compañía Android Inc., que fue comprada en el año 2005 por Google, el cual liberó la mayoría de código Android bajo licencia Apache (licencia libre y de código abierto).

Desde su creación ha ido pasando por diferentes versiones, desde la primera (0.5) hasta la actual (13.0).

<https://andro4all.com/guias/android/versiones-android-historia>

Android se ha convertido de forma rápida en uno de los sistemas operativos de móviles con mayor presencia. Actualmente, hay más de 3.000 millones de dispositivos móviles Android activados, siendo el S.O. que más abunda en los terminales hoy en día.

Las ventajas que han hecho posible su gran éxito son:

- Ser de código abierto, con licencia Apache.
- Dar libertad al usuario del dispositivo para instalar el software que crea oportuno, sin imponer que sea software propietario.
- Los desarrolladores tienen libertad para desarrollar cualquier software y ofrecerlo a los usuarios.
- No está limitado a determinados proveedores, operadoras o fabricantes, etc.

4.2. Características principales

Las características principales de los sistemas Android son las siguientes:

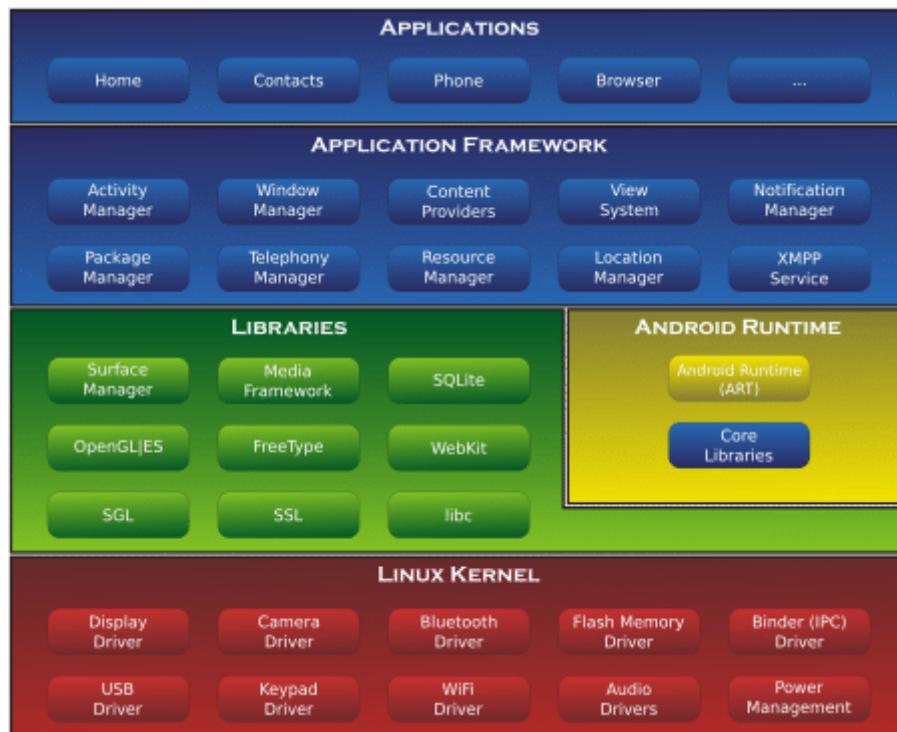
- Diseño de dispositivo: La plataforma es adaptable a pantallas de mayor resolución, VGA, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la OpenGL ES 2.0 y diseño de teléfonos tradicionales.
- Almacenamiento: SQLite, una base de datos liviana, que es usada para propósitos de almacenamiento de datos.
- Conectividad: Android soporta las siguientes tecnologías de conectividad: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, HSDPA, HSPA+, NFC y WiMAX. GPRS, UMTS y HSDPA+.
- Mensajería: SMS y MMS son formas de mensajería, incluyendo mensajería de texto y ahora la Android Cloud to Device Messaging Framework (C2DM) es parte del servicio de Push Messaging de Android.
- Navegador web: El navegador web incluido en Android está basado en el motor de renderizado de código abierto WebKit, emparejado con el motor JavaScript V8 de Google Chrome.
- Soporte de Java: Aunque la mayoría de las aplicaciones están escritas en Java, no hay una máquina virtual Java en la plataforma. El bytecode Java no es ejecutado, sino que primero se compila en un ejecutable Dalvik y corre en la Máquina Virtual Dalvik. Dalvik es una máquina virtual especializada, diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados. El soporte para J2ME puede ser agregado mediante aplicaciones de terceros como el J2ME MIDP Runner.
- Soporte multimedia: Android soporta los siguientes formatos multimedia: WebM, H.263, H.264 (en 3GP o MP4), MPEG-4 SP, AMR, AMR-WB (en un contenedor 3GP), AAC, HE-AAC (en contenedores MP4 o 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF y BMP.
- Soporte para streaming: Streaming RTP/RTSP (3GPP PSS, ISMA), descarga progresiva de HTML (HTML5 <video> tag).
- Soporte para hardware adicional: Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de

presión, sensores de luz, gamepad, termómetro, aceleración por GPU 2D y 3D.

- Entorno de desarrollo: Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software. Inicialmente el entorno de desarrollo integrado (IDE) utilizado era Eclipse con el plugin de Herramientas de Desarrollo de Android (ADT). Ahora se considera como entorno oficial Android Studio, descargable desde la página oficial de desarrolladores de Android.
- Google Play: Google Play es un catálogo de aplicaciones gratuitas o de pago en el que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC.
- Multi-táctil: Android tiene soporte nativo para pantallas capacitivas con soporte multitáctil que inicialmente hicieron su aparición en dispositivos como el HTC Hero. La funcionalidad fue originalmente desactivada a nivel de kernel (posiblemente para evitar infringir patentes de otras compañías). Más tarde, Google publicó una actualización para el Nexus One y el Motorola Droid que activa el soporte multi-táctil de forma nativa.
- Bluetooth: El soporte para A2DF y AVRCP fue agregado en la versión 1.5; el envío de archivos (OPP) y la exploración del directorio telefónico fueron agregados en la versión 2.0; y el marcado por voz junto con el envío de contactos entre teléfonos en la versión 2.2.
- Videollamada: Android soporta videollamada a través de Hangouts (ex-Google Talk) desde su versión HoneyComb.
- Multitarea: Multitarea real de aplicaciones está disponible, es decir, las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj.
- Características basadas en voz: La búsqueda en Google a través de voz está disponible como "Entrada de Búsqueda" desde la versión inicial del sistema.
- Tethering: Android soporta tethering, que permite al teléfono ser usado como un punto de acceso alámbrico o inalámbrico (todos los teléfonos desde la versión 2.2). Para permitir a un PC usar la conexión de datos del móvil android se podría requerir la instalación de software adicional.

4.3. Arquitectura

La arquitectura de Android sigue la siguiente figura:



Donde en ella destacamos:

- Aplicaciones: las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java. Es un sistema operativo multitarea.
- Marco de trabajo de aplicaciones: los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.
- Bibliotecas: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android; algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre

otras.

- Runtime de Android: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato.dex por la herramienta incluida "dx". Hay que tener en cuenta que a partir de la versión 5.0 ya existe la máquina virtual ART, por lo que al lado de la imagen de la máquina virtual Dalvik debería aparecer otra con la ART.
- Núcleo Linux: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

4.4. Máquina virtual

Máquina Virtual Dalvik

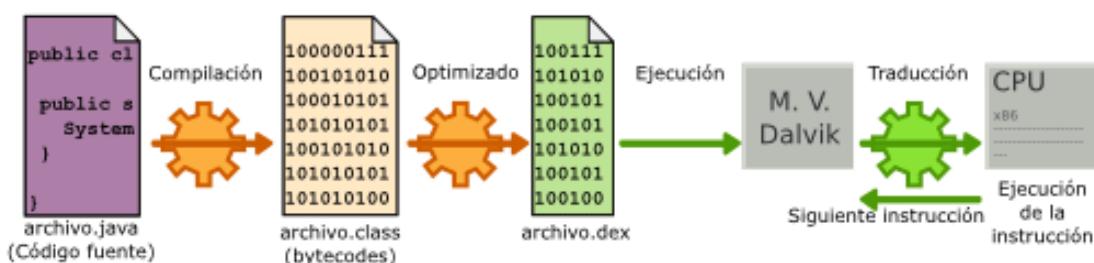
Dalvik es la máquina virtual usada en los dispositivos Android, la cual tiene una serie de características que la diferencian de las máquinas virtuales JAVA. Las aplicaciones Android están hechas en el lenguaje de programación java, pero usan un bytecode diferente al resto de aplicaciones creadas con las máquinas virtuales anteriores.

Características de Dalvik:

- El bytecode generado, primero es transformado al bytecode de Java y posteriormente se transforma en el usado por Android. Es decir, primero son generados los .class típicos de JAVA, y posteriormente se transforman al tipo de archivo .dex (dalvik executable). Estos archivos dex son comprimidos en los conocidos APK (Android Package), que son los famosos .apk instalables.

- Está diseñada para ejecutar varias instancias de la propia máquina simultáneamente.
- Optimizada para necesitar poca memoria.
- Se basa en registros en vez de en pilas aprovechando así el mejor rendimiento de los móviles con estos.
- Está distribuida como software libre, usando la licencia Apache.

En la siguiente imagen podemos comprobar cómo se produce la generación de los .dex hasta su ejecución.



La máquina virtual de Java, que podemos encontrar en casi todas las PC's actuales, se basa en el uso de las pilas. De modo contrario, Dalvik utiliza los registros, ya que los teléfonos móviles están optimizados para la ejecución basada en los mismos.

Aunque utilizamos el lenguaje Java para programar las aplicaciones Android, el bytecode de Java no es ejecutable en un sistema Android. De igual forma, las librerías Java que utilizan Android son ligeramente distintas a las utilizadas en Java Standard Edition (Java SE) o en Java Mobile Edition (Java ME), guardando también características en común.

El uso de Dalvik permite reducir bastante el tamaño del programa, buscando información duplicada en las diversas clases y reutilizándola.

Lo que llamamos en Java como “recolectar basura”, que libera el espacio en memoria de objetos que ya no utilizamos en nuestros programas, ha sido perfeccionada en Android con el fin de mantener siempre libre la máxima memoria posible.

De igual forma, el hecho de que Android haga un uso extenso del lenguaje XML para definir las interfaces gráficas y otros elementos, implica que estos archivos deben ser linkeados a la hora de compilar y para que su conversión a bytecode pueda mejorar el rendimiento de nuestras

aplicaciones.

Máquina Virtual ART

Desde el punto de vista más cercano a los dispositivos Android, este implementa un sistema operativo basado en Linux, y cuya máquina virtual (hasta ahora Dalvik) permite compilar los archivos de Java, convirtiéndolos en archivos .DEX, que permiten ahorrar espacio, además de estar optimizados para su ejecución en el dispositivo. Como parte de las características multitarea que proporciona un smartphone, es decir, la ejecución de varias máquinas virtuales, manteniendo varias de ellas en segundo plano, suele conllevar una falta de fluidez en comparación a su ejecución directa en un sistema operativo, y, por lo tanto, empeora la experiencia del usuario con el dispositivo.

Como solución a este problema, Google decidió desde la versión KitKat 4.4 (de manera experimental) ir introduciendo la nueva máquina virtual ART, cuyo lenguaje de programación es C++, y que es posible probar desde “Opciones para desarrolladores” en los ajustes del dispositivo (smartphones con Android KitKat o superiores).

Una vez expuestos algunos de los motivos que han llevado a Google a sustituir Dalvik en beneficio de ART, es necesario describir las principales características o ventajas de la nueva máquina virtual de Android, que podrían simplificarse en las siguientes:

- Compilación Ahead-of-time (AOT); Podríamos traducirla literalmente por ‘compilación por adelantado’ o precompilación. Es una de las principales diferencias con Dalvik, ya que este ejecuta una máquina virtual que interpreta el código a la vez que se ejecuta la aplicación (archivos .DEX mediante el sistema de compilación JIT o lo que es lo mismo “Justo a tiempo”). En cambio, ART realizará una precompilación durante el proceso de instalación de la aplicación (nuevo tipo de archivo compilado ELF), mediante el uso de la herramienta dex2oat (utilidad que permite compilar archivos de entrada DEX, generando un ejecutable de la aplicación para el dispositivo). Esta primera compilación permite realizar cierta carga de datos, para que las tareas de inicio o cierre de la aplicación se realicen de manera más rápida, y, por lo tanto, se disminuyan los tiempos de uso de CPU, aumentando el

rendimiento de la batería.

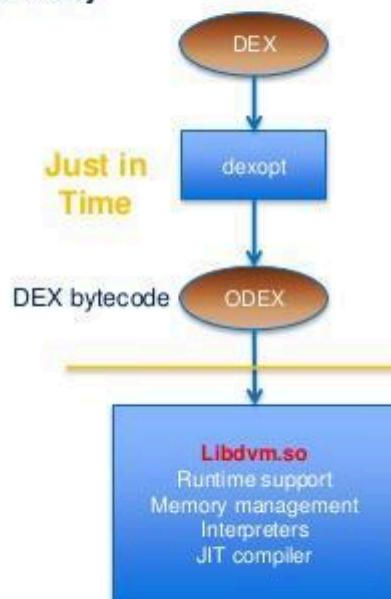
- Mejoras en el recolector de basura: Estas mejoras permiten evitar problemas de rendimiento que suelen acarrear, entre otras cosas, mala respuesta por parte de la interfaz de usuario, y que, en gran medida, se solucionan a partir de las siguientes acciones:
 - Menor tiempo de pausa en el recolector para objetos creados recientemente.
 - Procesamiento paralelo durante la pausa del recolector de basura.
 - Disminuir de dos a uno el número de pausas que realiza el recolector de basura, durante el tiempo que tarda en completar la tarea.
 - Mejora de la recolección de basura concurrente, que elimina procesos que ya no están en uso minimizando el consumo de recursos de la CPU. Además, no será necesario llamar de manera explícita al recolector de basura por medio de System.gc(), cuando el dispositivo no disponga de memoria y una aplicación requiere más recursos (tipo de concurrencia GC_FOR_ALLOC).
- Mejoras en el desarrollo y depuración de aplicaciones:
 - Soporte para nuevas características de depuración: ART implementa una serie de opciones, que permitirán desde saber el número de instancias de una clase, pasando por el filtrado de eventos de esta, o conocer el valor devuelto por un método mediante el evento “method-exit”.
 - Mejora del diagnóstico de los fallos y excepciones: ART proporciona mayor información de las excepciones que se produzcan en tiempo de ejecución, informando en mayor detalle de las siguientes: java.lang.ClassCastException, java.lang.ClassNotFoundException y java.lang.NullPointerException.
 - Soporte para perfiles en la vista de trazas, que a diferencia de su uso en Dalvik, ART no ha limitado su sobrecarga por problemas de rendimiento en tiempo de ejecución.

En las siguientes imágenes se podrán visualizar diferentes comparativas entre ART y Dalvik:

Dalvik VM vs. ART VM

Dalvik VM

32-bit only



Just in Time

DEX bytecode

ODEX

Runtime

Libdvm.so
Runtime support
Memory management
Interpreters
JIT compiler

ART VM

32-bit and 64-bit



Ahead Of Time

Native binary

Runtime

Libart.so
Runtime support
Memory management
Interpreter
Class Linking

11



[Video explicativo.](#)

4.5. Versiones

Las versiones de Android reciben, en inglés, el nombre de diferentes postres o dulces. En cada versión el postre o dulce elegido empieza por una letra distinta, conforme a un orden alfabético:

- A: Apple Pie (v1.0): Tarta de manzana
- B: Banana Bread (v1.1): Pan de plátano
- C: Cupcake (v1.5): Panque
- D: Donut (v1.6): Rosquilla
- E: Éclair (v2.0/v2.1): Pepito
- F: Froyo (v2.2): Yogurt helado
- G: Gingerbread (v2.3): Pan de jengibre
- H: Honeycomb (v3.0/v3.1/v3.2): Panal de miel
- I: Ice Cream Sandwich (v4.0): Sándwich de helado
- J: Jelly Bean (v4.1/v4.2/v4.3): Gominola
- K: KitKat (v4.4): Kit Kat
- L: Lollipop (v5.0/v5.1): Piruleta
- M: Marshmallow (v6.0) Malvavisco
- N: Nougat (v7.0/v7.1.2):
- O: Oreo (v8.0)(v8.1)
- P: Pie (v9.0)
- Android 10 (v10.0)
- Android 11 (v11.0)
- Android 12 (v12.0)
- Android 13 (v13.0)
- Android 14 (v14.0)
- Android 15 (v15.0)

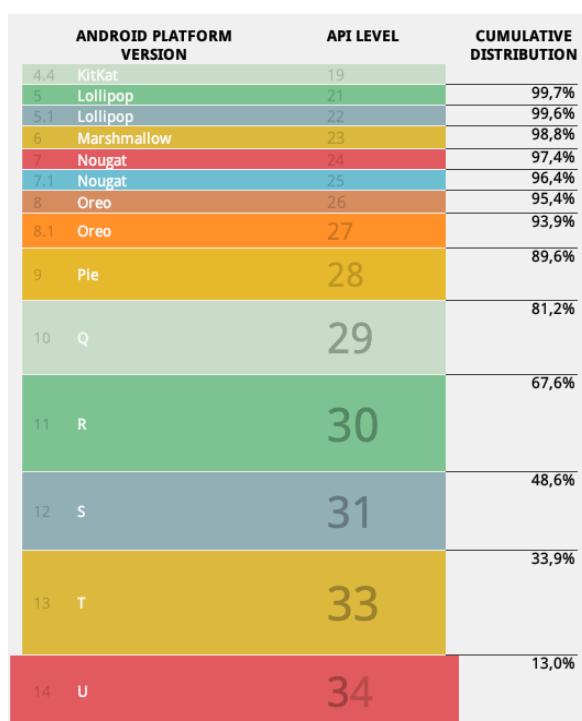
Podemos realizar una búsqueda por Internet para ver cómo han ido evolucionando las diferentes versiones de Android:

- <https://andro4all.com/guias/android/versiones-android-historia>

- <https://norfipc.com/celulares/todas-versiones-sistema-operativo-android.php>
- <https://www.nts-solutions.com/blog/versiones-android.html>

4.6. Fragmentación del mercado

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 mejor versión posible que nuestra aplicación pueda soportar. Para ayudarnos a tomar la decisión de qué plataforma utilizar puede ser interesante consultar los porcentajes de utilización:



Recomendamos consultar el link siguiente antes de tomar decisiones sobre las versiones a utilizar

- [Android developers -> Platform Versions](http://developer.android.com/resources/dashboard/platform-versions.html): Estadística de dispositivos Android según plataforma instalada, que han accedido a Android Market.

<http://developer.android.com/resources/dashboard/platform-versions.html>

- [Android developers -> About](http://developer.android.com/about/index.html): En el menú de la izquierda aparecen links 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>

4.7. Distribución de las aplicaciones

Google Play Store es una plataforma de distribución digital de aplicaciones móviles para los dispositivos con sistema operativo Android, así como una tienda en línea desarrollada y operada por Google. Esta plataforma permite a los usuarios navegar y descargar aplicaciones (desarrolladas mediante Android SDK), juegos, música, libros, revistas y películas. También se pueden adquirir dispositivos móviles como ordenadores Chromebook, teléfonos inteligentes Nexus, Google Chromecast, entre otros.

Las aplicaciones se encuentran disponibles de forma gratuita, así como también con costo. Pueden ser descargadas directamente desde un dispositivo con Android a través de la aplicación móvil Play Store. Es posible también instalar estas aplicaciones directamente y sin necesidad de una computadora, en dispositivos con sistema operativo BlackBerry 10.2

En marzo de 2012, con la fusión de Android Market con Google Music, el servicio fue renombrado a Google Play, como resultado de la nueva estrategia de distribución digital de Google. En julio de 2013, se anunció que Google Play había sobrepasado 1 millón de aplicaciones publicadas y se habían registrado más de 50 mil millones de descargas.

La gran novedad que aporta Google Play hace referencia a los desarrolladores: estos serán capaces de hacer su contenido disponible en un servicio abierto, el servicio de Google que ofrece una retroalimentación y sistema de calificación similar a YouTube. Los desarrolladores tendrán un entorno abierto y sin obstáculos para hacer su contenido disponible. El contenido puede subirse al mercado después de tres pasos: registrarse como comerciante, subir y describir su contenido y publicarlo. Para registrarse como desarrollador y poder subir aplicaciones hay que pagar una cuota de registro (US \$25,00) con tarjeta de crédito (mediante Google Checkout).

Los desarrolladores de las aplicaciones de pago reciben un 70% del precio total de la aplicación, mientras que el 30% restante es destinado a las empresas. El beneficio obtenido de Google Play es pagado a los desarrolladores a través de sus cuentas en el sistema Google Checkout.

5. ENTORNOS INTEGRADOS DE TRABAJO

Con casi 2.000 millones de dispositivos que montan Android como sistema operativo en la actualidad, es interesante pensar en una idea que pueda materializar y formar parte del mundo que rodea al smartphone.

Android es el sistema operativo móvil con más popularidad y que sigue creciendo a pasos agigantados. Es por ello que es muy importante conocer qué herramientas tenemos a nuestra disposición para poder desarrollar una aplicación de forma nativa para el sistema operativo Android. Para ello haremos uso de un IDE o entorno de desarrollo integrado.

Antes de seguir convendría aclarar el concepto de IDE, que no es más que un programa informático compuesto por un conjunto de herramientas de programación. Es un entorno de programación empaquetado como un programa o aplicación, que nos provee de un marco de trabajo agradable para la mayoría de los lenguajes de programación. Constan entre sus características básicas con:

- Editor de código
- Compilador
- Depurador (debugger)
- Constructor de interfaz gráfica

Como todos conocemos, el lenguaje que Google pensó que debería de usarse para programar aplicaciones para Android es Java. Existen multitud de entornos de desarrollo para poder realizar aplicaciones en Android, pero nosotros nos centraremos durante el curso en **Android Studio**.

6. ANDROID STUDIO

6.1. Instalación

La instalación de Android Studio es muy sencilla, por lo que vamos a dejar dicha instalación como actividad a realizar por el alumnado.

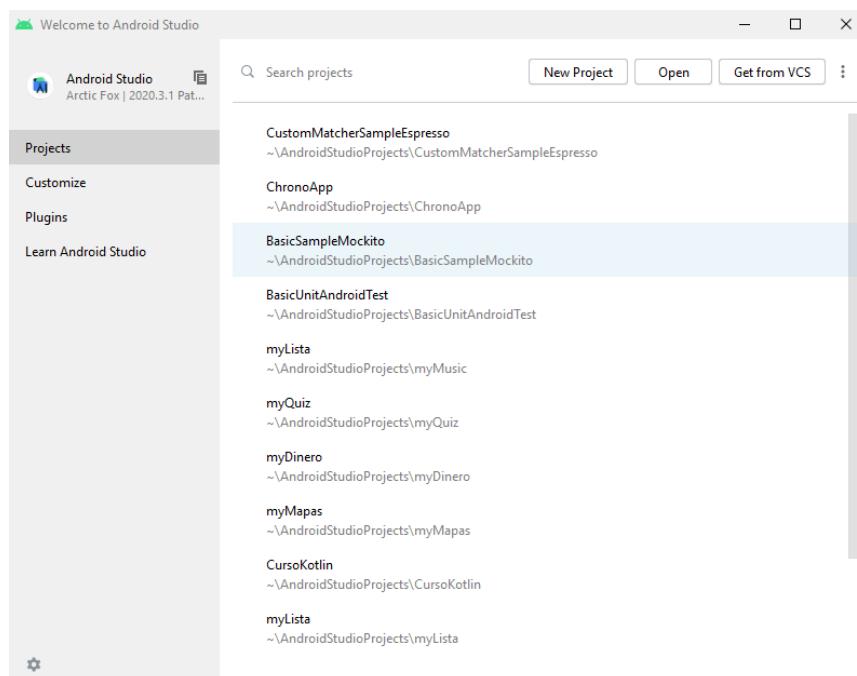
El enlace a Android Studio para su descarga es el siguiente:

<https://developer.android.com/sdk/index.html>

En dicho enlace podremos encontrar la aplicación para descargar, así como los requerimientos necesarios para su instalación.

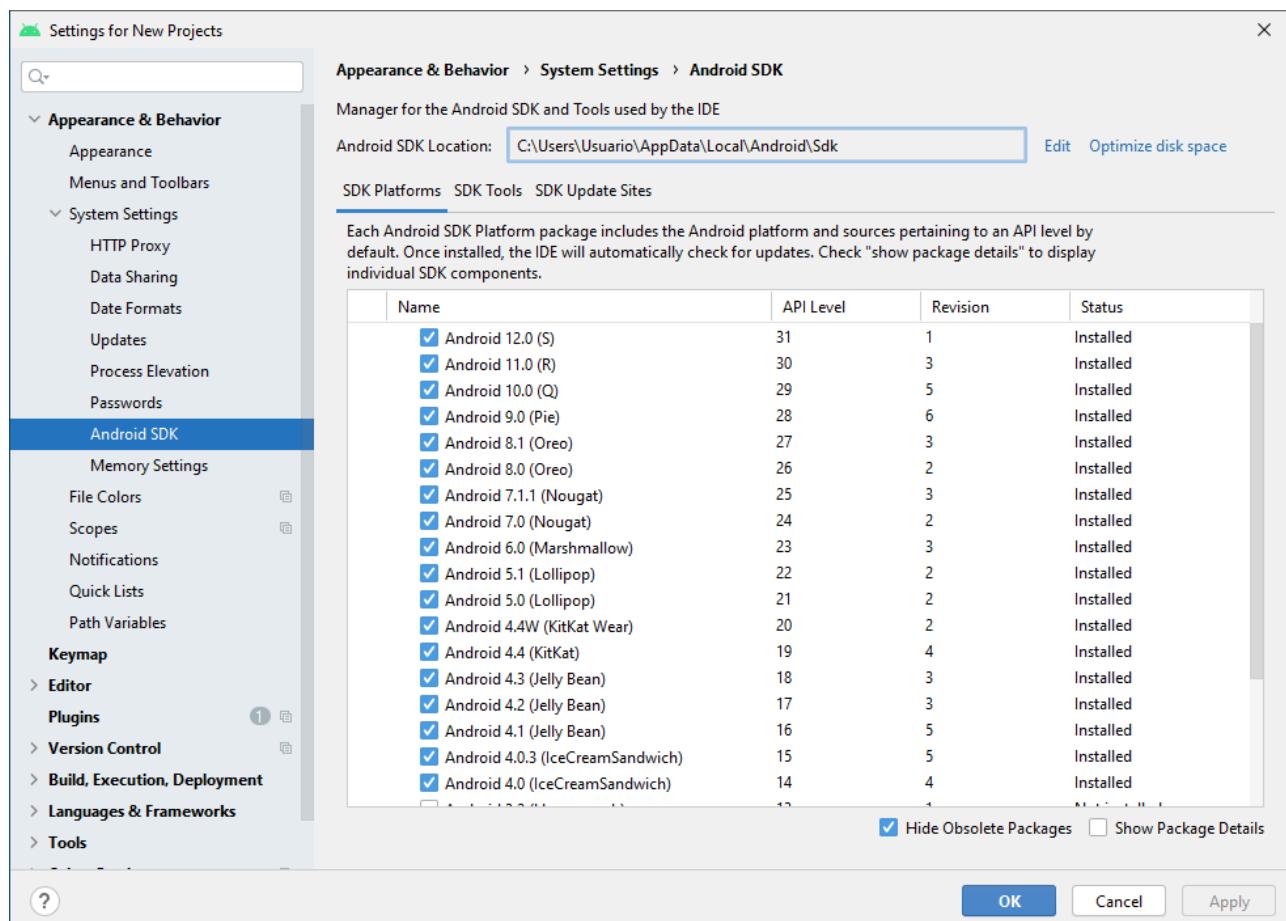
6.2. Configuración del entorno

Tras finalizar la instalación nos aparecerá la pantalla de bienvenida de Android Studio:

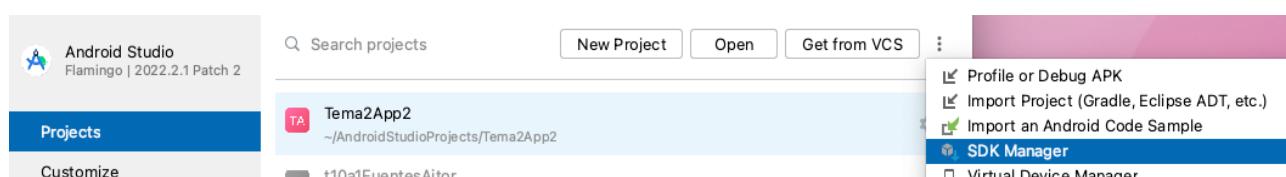


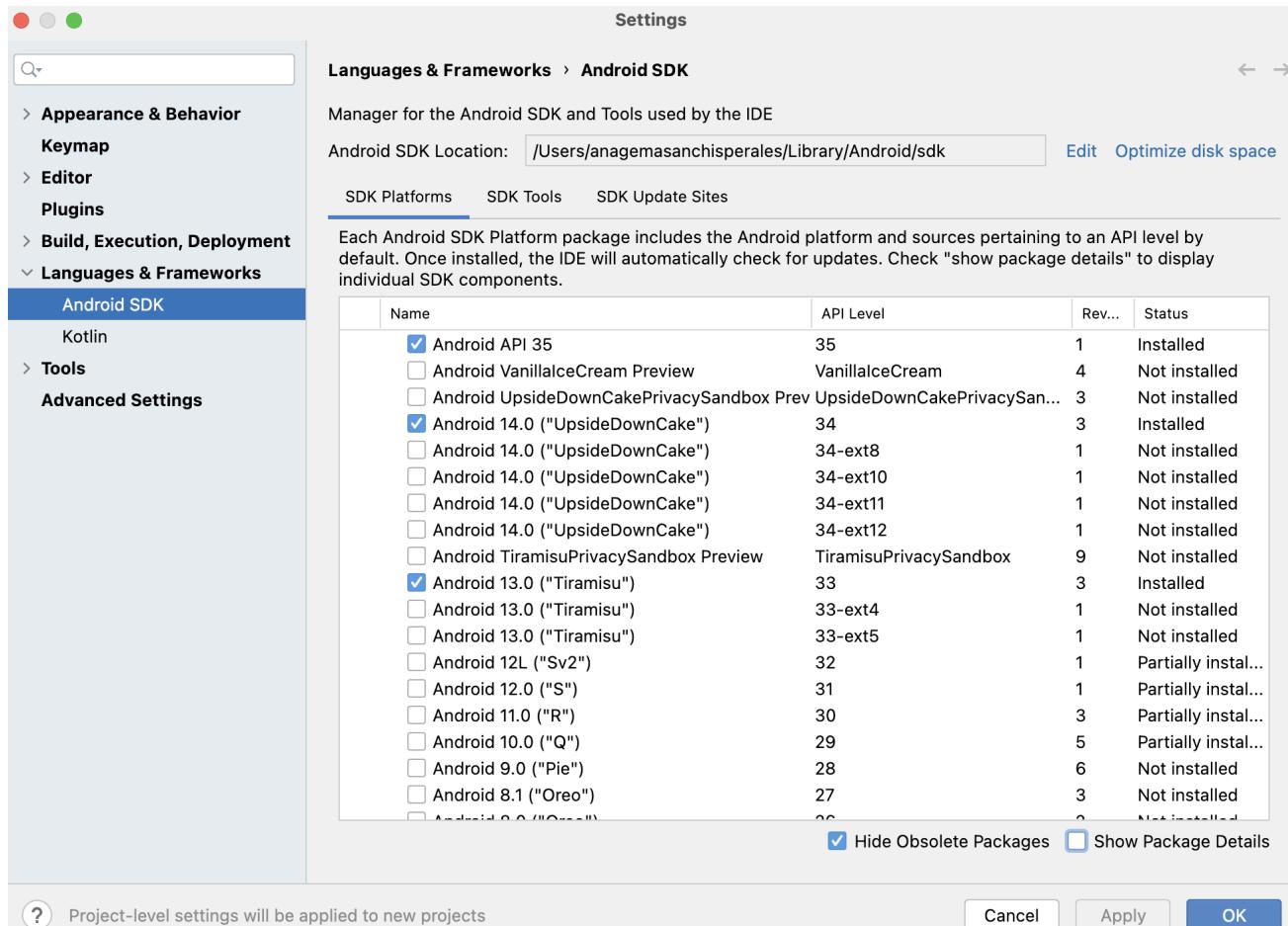
Ahora es necesario comprobar que, tanto la ruta del SDK como la del JAVA están configuradas correctamente.

Para ello pulsaremos la opción “...” de la pantalla de bienvenida, tras ésta accederemos a la opción “**SDK Manager**”. Buscamos “**Android SDK**”. Y revisaremos el apartado “**SDK Location**” asegurándonos de que tenemos correctamente configuradas las rutas al SDK de Android.



Además, con esta herramienta podremos instalar, desinstalar o actualizar todos los componentes disponibles como parte del SDK de Android.





The screenshot shows the 'SDK Platforms' tab of the Android SDK Manager. It lists various API levels with their names, API levels, revision numbers, and installation status. The 'Android API 35' row is selected, indicated by a checked checkbox. Other rows include 'Android Vanillacream Preview', 'Android UpsideDownCakePrivacySandbox Prev', 'Android 14.0 ("UpsideDownCake")', 'Android TiramisuPrivacySandbox Preview', 'Android 13.0 ("Tiramisu")', 'Android 13.0 ("Tiramisu")', 'Android 13.0 ("Tiramisu")', 'Android 12L ("Sv2")', 'Android 12.0 ("S")', 'Android 11.0 ("R")', 'Android 10.0 ("Q")', 'Android 9.0 ("Pie")', 'Android 8.1 ("Oreo")', and 'Android 8.0 ("Oreo")'. A note at the bottom says 'Project-level settings will be applied to new projects'.

Name	API Level	Rev...	Status
Android API 35	35	1	Installed
Android Vanillacream Preview	Vanillacream	4	Not installed
Android UpsideDownCakePrivacySandbox Prev	UpsideDownCakePrivacySan...	3	Not installed
Android 14.0 ("UpsideDownCake")	34	3	Installed
Android 14.0 ("UpsideDownCake")	34-ext8	1	Not installed
Android 14.0 ("UpsideDownCake")	34-ext10	1	Not installed
Android 14.0 ("UpsideDownCake")	34-ext11	1	Not installed
Android 14.0 ("UpsideDownCake")	34-ext12	1	Not installed
Android TiramisuPrivacySandbox Preview	TiramisuPrivacySandbox	9	Not installed
Android 13.0 ("Tiramisu")	33	3	Installed
Android 13.0 ("Tiramisu")	33-ext4	1	Not installed
Android 13.0 ("Tiramisu")	33-ext5	1	Not installed
Android 12L ("Sv2")	32	1	Partially instal...
Android 12.0 ("S")	31	1	Partially instal...
Android 11.0 ("R")	30	3	Partially instal...
Android 10.0 ("Q")	29	5	Partially instal...
Android 9.0 ("Pie")	28	6	Not installed
Android 8.1 ("Oreo")	27	3	Not installed
Android 8.0 ("Oreo")	26	2	Not installed

Hide Obsolete Packages Show Package Details

Project-level settings will be applied to new projects Cancel Apply OK

Tenemos que instalar la **API 35**, que es **Android 15.0**.

Seleccionaremos los componentes que queremos instalar o actualizar, pulsaremos el botón "**Apply**", aceptaremos las licencias correspondientes, y esperaremos a que finalice la descarga e instalación. Una vez finalizado el proceso es recomendable cerrar el SDK Manager y reiniciar Android Studio.

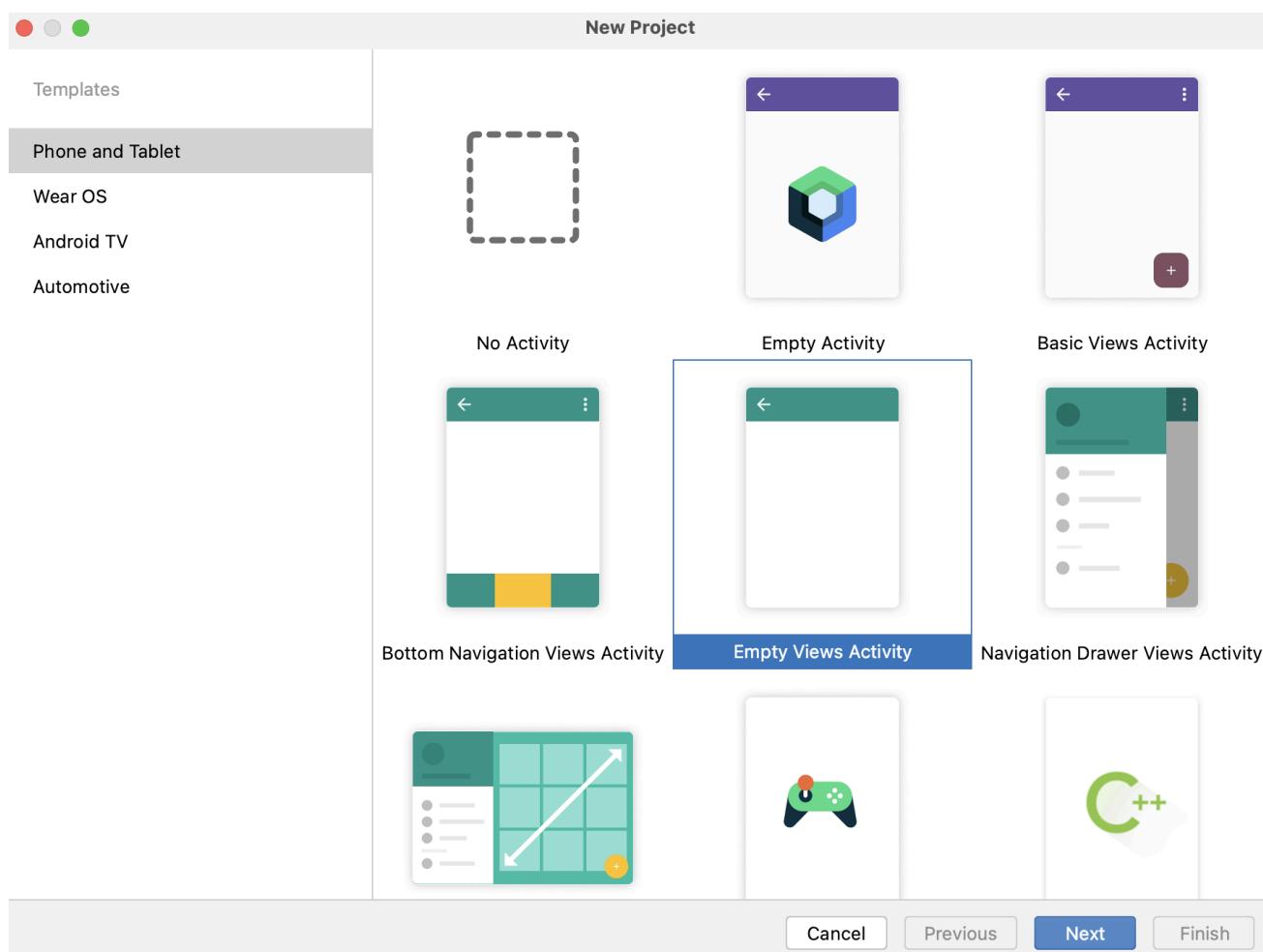
6.3. Estructura de un proyecto

Para empezar a comprender cómo se construye una aplicación Android vamos a crear un nuevo proyecto en Android Studio y echaremos un vistazo a la estructura general del proyecto creado por defecto.

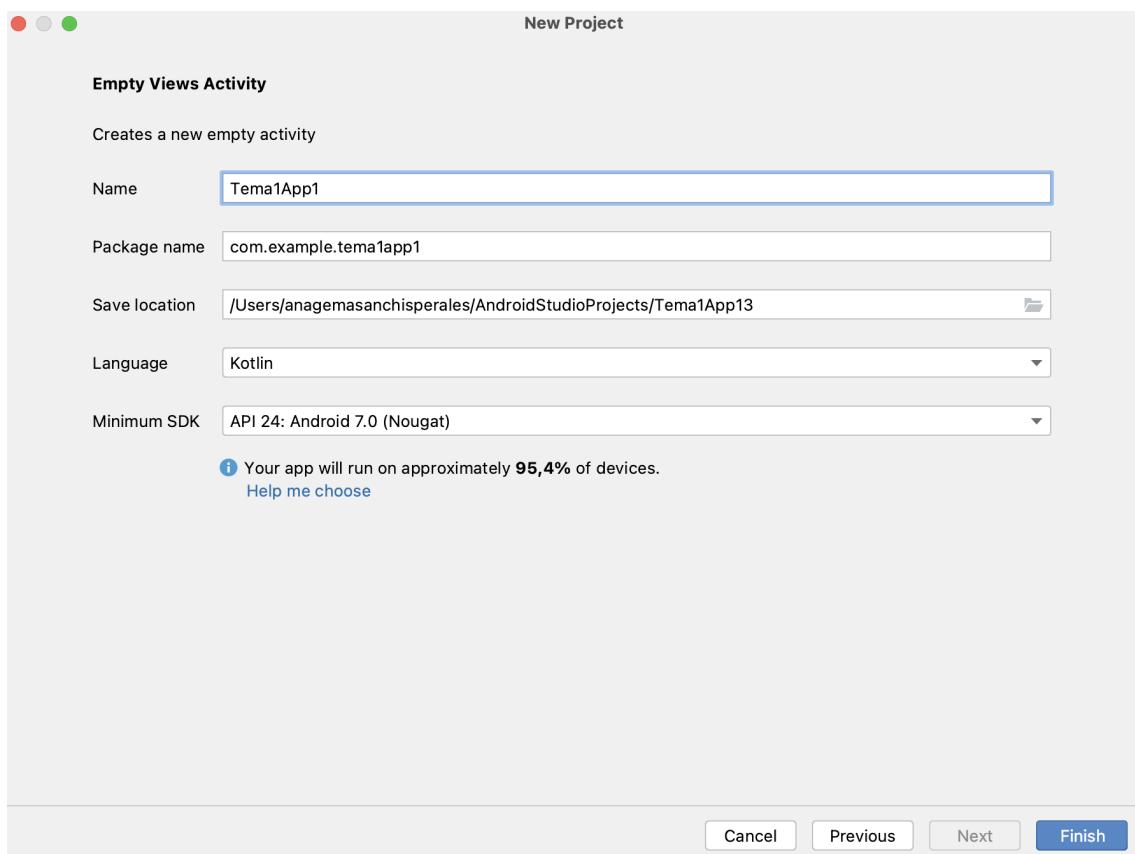
Para crear un nuevo proyecto ejecutaremos Android Studio y desde la pantalla de bienvenida pulsaremos la opción “**New Project**” para iniciar el asistente de creación de un nuevo proyecto.

El asistente de creación del proyecto nos guiará por las distintas opciones de creación y configuración de un nuevo proyecto Android.

En la primera pantalla indicaremos el tipo de actividad principal de la aplicación. Entenderemos por ahora que una actividad es una “ventana” o “pantalla” de la aplicación. Para empezar seleccionaremos **Empty Views Activity**, que es el tipo más sencillo.

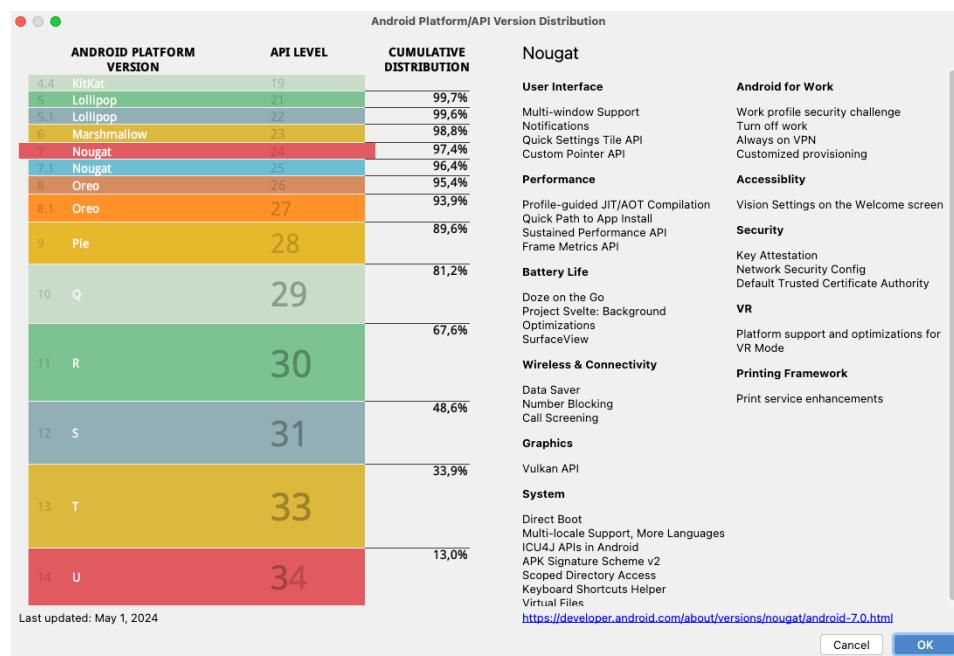


En la siguiente pantalla indicaremos por este orden, **el nombre de la aplicación, el dominio de la compañía y la ruta donde crear el proyecto**. El segundo de los datos indicados tan sólo se utilizará como paquete de nuestras clases java. El paquete java principal utilizado será **com.xxxxxxx**. El **xxxxxx** serán tus iniciales de dos en dos, esto se hace así ya que es el nombre del paquete y Android lo tomará como su ApplicationID, que debe ser único universalmente, en caso contrario, tendríamos problemas con Google Play si publicamos una App y ya existe con ese id, podemos observar que usa una notación DNS inversa, se supone que es el dominio de la empresa invertido. El nombre de la aplicación será **Tema1App1**. Así como el lenguaje de programación que será **Kotlin**. También tendremos que seleccionar la **API mínima** (es decir, la versión mínima de Android) que soportará la aplicación. A lo largo del curso nos centraremos en Android 7.0 como versión mínima (**API 24**).



La **versión mínima** que seleccionemos en esta pantalla implica que nuestra aplicación se pueda ejecutar en más o menos dispositivos. De esta forma, cuanto menor sea esta, a más dispositivos podrá llegar nuestra aplicación, pero más complicado será conseguir que se ejecute correctamente en todas las versiones de Android. Podríamos escoger, como versiones para comenzar a desarrollar, recomendamos lo más cercano al 95%, entre la API 19 y la 24, pero nos quedaremos con la API 24

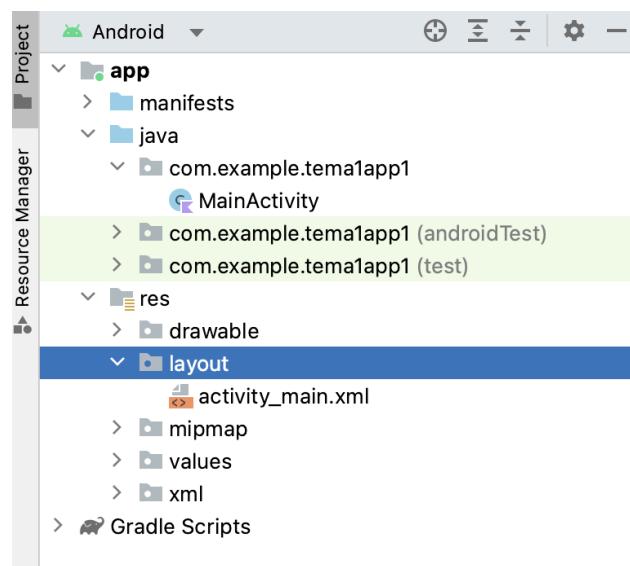
por ser más nueva.



Por último, una vez configurado todo pulsamos el botón **Finish** y Android Studio creará por nosotros toda la estructura del proyecto y los elementos indispensables que debe contener. Si todo va bien, aparecerá la pantalla principal de Android Studio con el nuevo proyecto creado.

Pasaremos ahora a ver la estructura del proyecto:

En la parte izquierda, podemos observar todos los elementos creados inicialmente para el nuevo proyecto Android.



En los siguientes apartados describiremos los elementos principales de esta estructura.

Lo primero que debemos distinguir son los conceptos de proyecto y módulo. La entidad proyecto es única, y engloba a todos los demás elementos. Dentro de un proyecto podemos incluir varios módulos, que pueden representar aplicaciones distintas, versiones diferentes de una misma aplicación, o distintos componentes de un sistema (aplicación móvil, aplicación servidor, librerías...). En la mayoría de los casos, trabajaremos con un proyecto que contendrá un sólo módulo correspondiente a nuestra aplicación principal. Por ejemplo en este caso que estamos creando tenemos el proyecto “Tema1App1” que contiene al módulo “app” que contendrá todo el software de la aplicación de ejemplo.

A continuación describiremos los contenidos principales de nuestro módulo principal.

Carpeta /app/java/.../MainActivity

Esta carpeta contiene todo el **código fuente de la aplicación**, clases auxiliares, etc. Inicialmente, Android Studio creará por nosotros el código básico de la pantalla (actividad o activity) principal de la aplicación, que recordemos que en nuestro caso era MainActivity, y siempre bajo la estructura del paquete kotlin definido durante la creación del proyecto.

El código es el siguiente:

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v,
        insets ->
            val systemBars =
            insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
            systemBars.right, systemBars.bottom)
            insets
        }
    }
}
```

}

La clase MainActivity que hereda de la clase AppCompatActivity que sobreescribe un método que se llama onCreate, es un método que se ejecuta cuando se crea nuestra actividad, que llama al constructor que nos va a servir posteriormente para resetear los estados y finalmente setContentView para cargar las vistas, que vienen definidas en xml.

Carpeta /app/res/

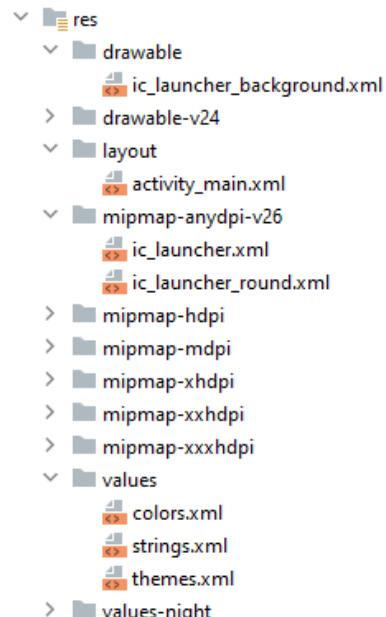
Contiene todos los **ficheros de recursos** necesarios para el proyecto: imágenes, layouts, cadenas de texto, etc. Los diferentes tipos de recursos se pueden distribuir entre las siguientes subcarpetas:

Carpeta	Descripción
/res/drawable/	Contiene imágenes y otros elementos gráficos usados por la aplicación. Para poder definir diferentes recursos dependiendo de la resolución y densidad de la pantalla del dispositivo se suele dividir en varias subcarpetas: <ul style="list-style-type: none"> • /drawable (recursos independientes de la densidad) • /drawable-ldpi (densidad baja) • /drawable-mdpi (densidad media) • /drawable-hdpi (densidad alta) • /drawable-xhdpi (densidad muy alta) • /drawable-xxhdpi (densidad muy muy alta)
/res/mipmap/	Contiene los iconos de lanzamiento de la aplicación (el ícono que aparecerá en el menú de aplicaciones del dispositivo) para las distintas densidades de pantalla existentes.
/res/layout/	Contiene los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica. Para definir distintos <i>layouts</i> dependiendo de la orientación del dispositivo se puede dividir también en subcarpetas: <ul style="list-style-type: none"> • /layout (vertical) • /layout-land (horizontal)
/res/anim/ /res/animator/	Contienen la definición de las animaciones utilizadas por la aplicación.
/res/menu/	Contiene la definición XML de los menús de la aplicación.

/res/values/	Contiene otros ficheros XML de recursos de la aplicación, como por ejemplo cadenas de texto (<i>strings.xml</i>), estilos (<i>styles.xml</i>), colores (<i>colors.xml</i>), arrays de valores (<i>arrays.xml</i>), tamaños (<i>dimens.xml</i>), etc.
--------------	--

No todas estas carpetas tienen por qué aparecer en cada proyecto Android, tan sólo las que se necesiten. Iremos viendo qué tipo de elementos se pueden incluir en cada una de ellas y cómo se utilizan.

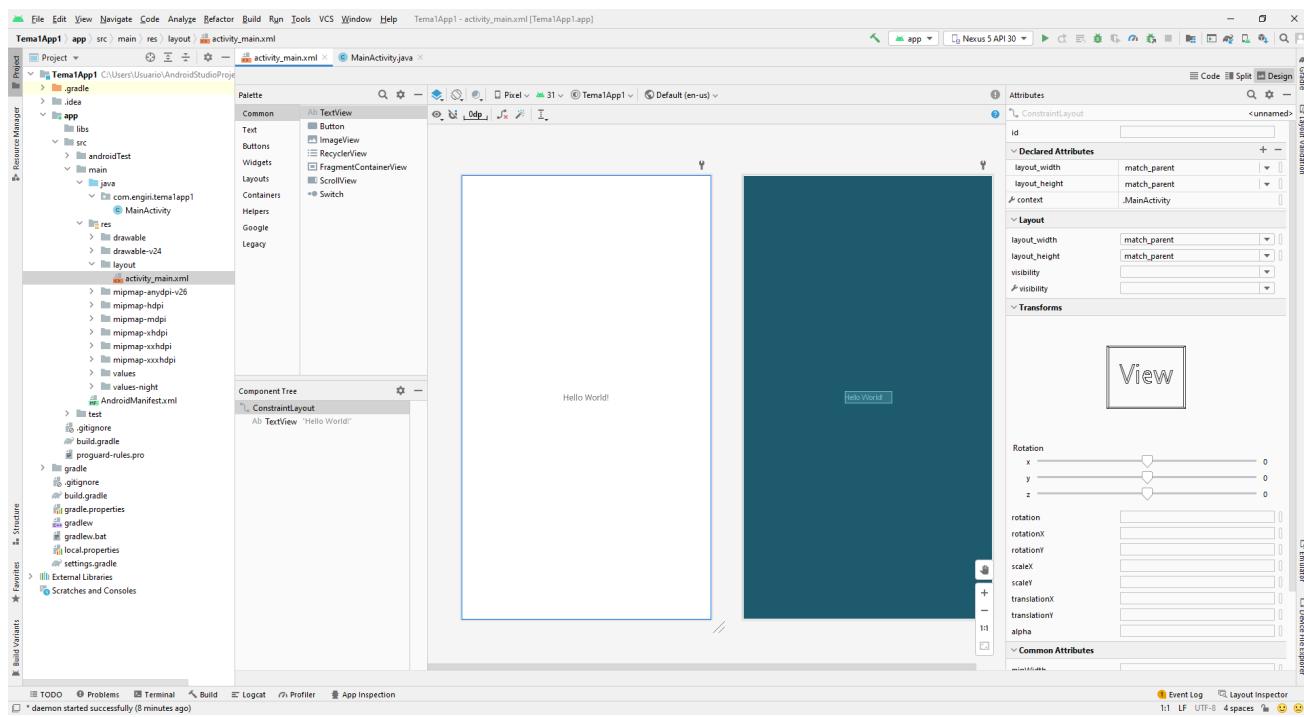
Como ejemplo, para un proyecto nuevo Android como el que hemos creado, tendremos por defecto los siguientes recursos para la aplicación:



Como se puede observar, existen algunas carpetas en cuyo nombre se incluye un sufijo adicional. Si nos situamos en dicha carpeta observamos seis carpetas dentro llamadas *mipmap-**, donde * es la densidad de la pantalla para la que está preparada la imagen usada para el ícono, y que son *anydpi-v26*, *hdpi*, *mdpi*, *xhdpi*, *xxhdpi* y *xxxhdpi*. Por otro lado *ic_launcher* e *ic_launcher_round* son los nombres de los archivos de las propias imágenes, es el ícono cuadrado y el ícono circular.

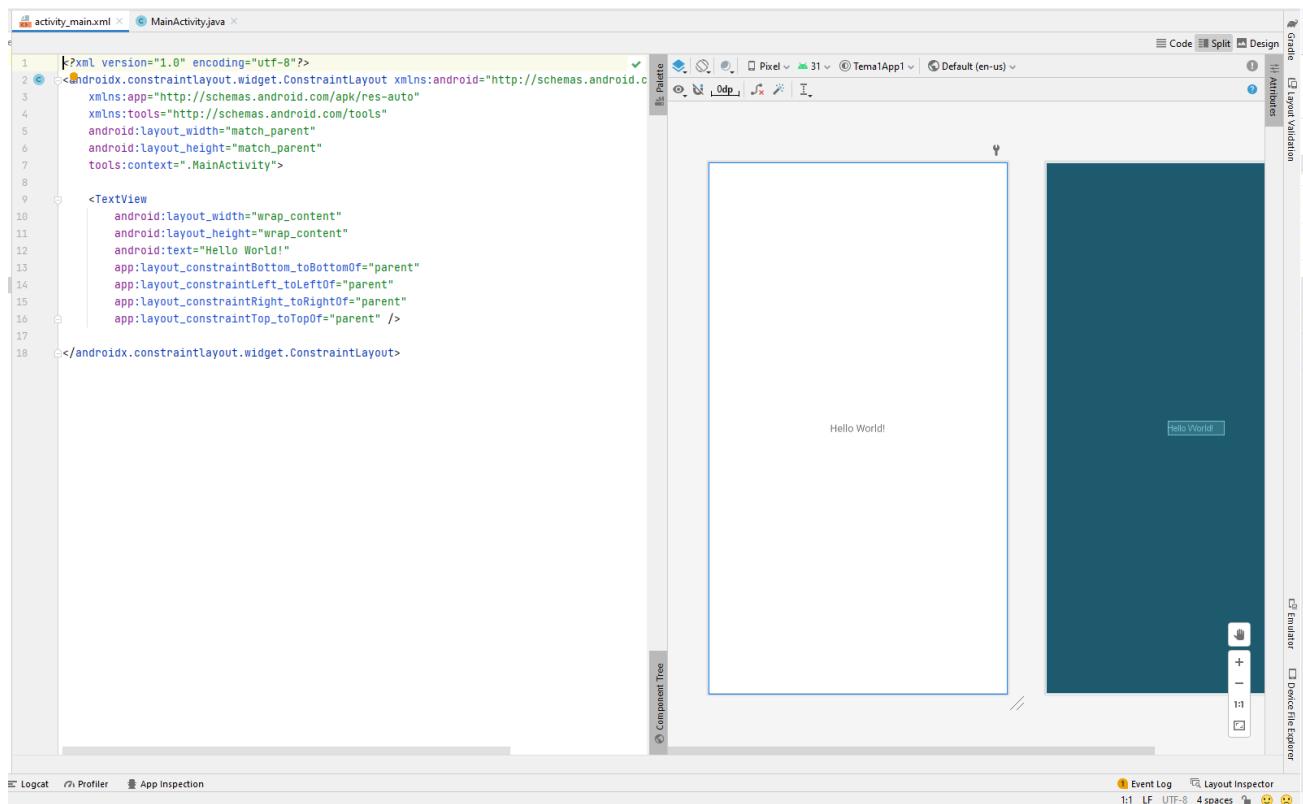
La moderna carpeta *mipmap-anydpi-v26* es distinta a las demás, realmente es un ícono adaptativo usando a partir de la API v26 de Android, son archivos XML. Se puede editar un ícono mediante el Image Asset Studio incluido en el propio IDE.

Entre los recursos creados por defecto cabe destacar los **layouts**, en nuestro caso sólo tendremos por ahora el llamado “activity_main.xml”, que contienen la definición de la interfaz gráfica de la pantalla principal de la aplicación. Si hacemos doble clic sobre este fichero Android Studio nos mostrará esta interfaz en su editor gráfico, y como podremos comprobar, en principio contiene tan sólo una etiqueta de texto con el mensaje “Hello World!”.



Pulsando sobre los botones superiores “**Code**”, “**Split**” y “**Design**” podremos alternar entre diferentes vistas.

- Design: editor gráfico (tipo arrastrar-y-soltar), mostrado en la imagen anterior.
- Code: editor XML .
- Split: combinación de vista gráfica y editor XML, que se muestra en la imagen siguiente:



Durante el curso utilizaremos tanto el editor gráfico como directamente su fichero XML asociado.

Fichero /app/manifests/AndroidManifest.xml

Contiene la definición en XML de muchos de los aspectos principales de la aplicación, como por ejemplo su nombre, icono..., sus componentes (pantallas, servicios...), o los permisos necesarios para su ejecución. Veremos más adelante más detalles de este fichero.

Fichero Gradle Scripts/build.gradle (Module:app)

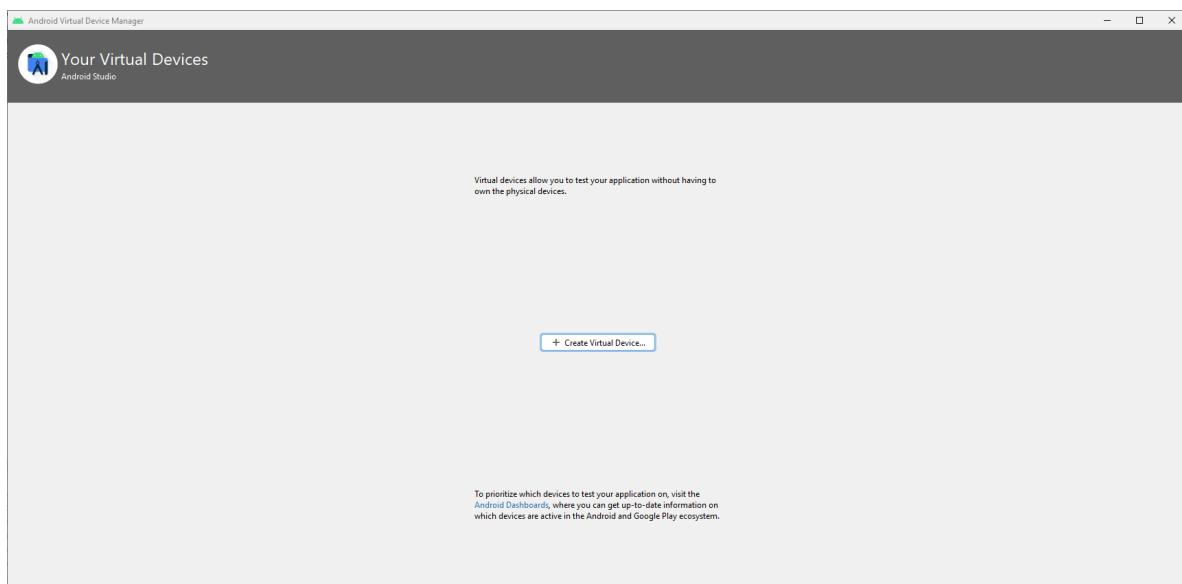
Contiene información necesaria para la compilación del proyecto, por ejemplo la versión del SDK de Android utilizada para compilar, la mínima versión de Android que soportará la aplicación, referencias a las librerías externas utilizadas, etc. Más adelante veremos también más detalles de este fichero.

En un proyecto pueden existir varios ficheros build.gradle, para definir determinados parámetros a distintos niveles. Por ejemplo, en nuestro proyecto podemos ver que existe un fichero build.gradle a nivel de proyecto. Que definirá parámetros globales a todos los módulos del proyecto.

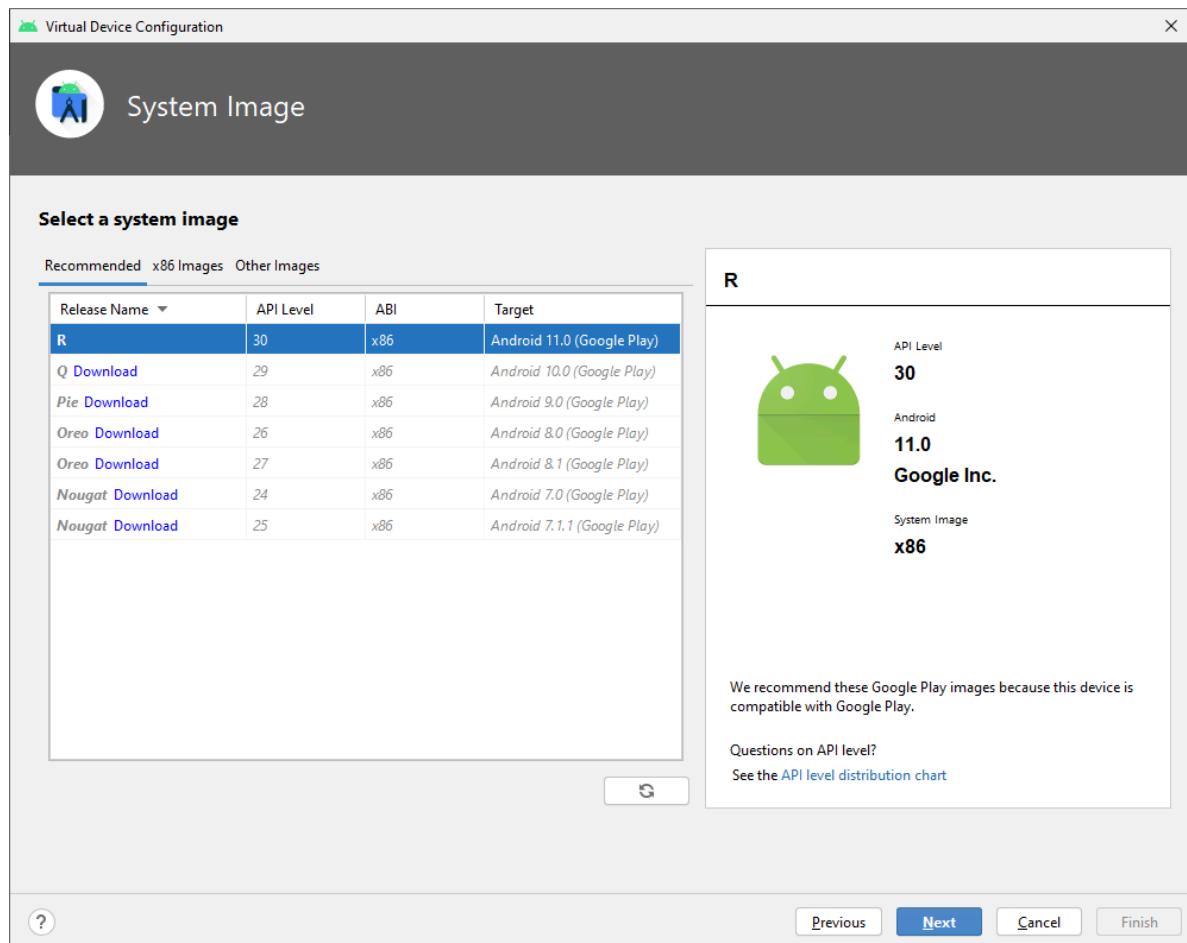
6.4. Gestión del AVD (Android Virtual Device)

Las AVD son unas herramientas imprescindibles para los desarrolladores y testers, ya que nos permiten emular en una computadora un entorno móvil a los que apuntará nuestra aplicación Android. Cuando recién hemos instalado el SDK (que incluye el AVD Manager) no contaremos con ningún dispositivo virtual. Así que te explicaré los pasos necesarios para crear una AVD para que puedas trabajar:

- Abrimos el AVD Manager desde la opción **Tools > AVD Manager**. También se puede abrir pulsando el botón  que tendremos en la barra de herramientas. Se abrirá la siguiente pantalla:



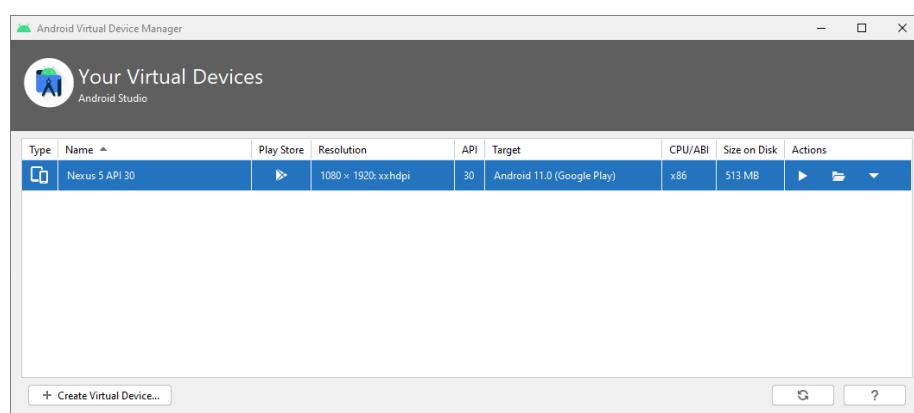
- Pulsamos el botón **Create Virtual Device...** para crear una nueva máquina virtual de Android.
- Se nos abrirá un asistente para crear la nueva máquina. En primer lugar hemos de seleccionar el hardware con el que vamos a trabajar. En nuestro caso trabajaremos con un móvil **Nexus 5X**. Pulsaremos el botón Next.
- Tras ello pasaremos a seleccionar la API con la que queremos trabajar. En nuestro caso trabajaremos con la **API 30 con procesador x86**. Si no tenemos instalado dicha API nos aparecerá un botón de Download. Si la tenemos instalada simplemente la seleccionamos y pulsamos el botón Next.



- A continuación introduciremos el **nombre de la AVD** y dejaremos los demás parámetros por defecto.
- Si pulsamos el botón de **Show Advanced Settings** podremos ver todas las características que se pueden configurar de la máquina virtual:
 - **Startup Orientation:** Indica cómo va a ser la orientación del dispositivo (horizontal o vertical).
 - **Front/Back Camera:** Para activar la emulación de la cámara delantera y trasera.
 - **Network:** Podemos configurar cómo va a ser el uso de la red en el dispositivo, así como su latencia.
 - **Snapshot:** Si lo seleccionas podrás congelar la ejecución del dispositivo en un determinado instante. Más tarde, podrás retomar la ejecución en este instante, sin tener que esperar a que se inicie el dispositivo. Conviene marcarlo para conseguir una carga más rápida.
 - **Use Host GPU:** Se habilita la emulación hardware para gráficos OpenGL ES. Su

navegación entre ventanas será más fluida. NOTA: No podemos seleccionar simultáneamente Snapshot y Use Host GPU.

- Memory and Storage: Memoria y almacenamiento que se dedicará al emulador.
 - RAM: Memoria total en MB. Dejarla por defecto.
 - VM Heap: Memoria dinámica asignada a la máquina virtual en MB. Dejarla por defecto.
 - Internal Storage: Memoria interna del dispositivo. Determinará el número de aplicaciones y datos que podrás instalar. Cuidado, esta memoria se reservará en tu disco duro, por lo que no es conveniente indicar un valor demasiado grande. Dejarla por defecto.
 - SD Card: Memoria externa del dispositivo. Tiene dos posibilidades:
 - Studio-managed: tamaño de la memoria. Esta creará un nuevo fichero.
 - File: se utilizará un fichero previamente creado.
- Device Frame: No lo utilizaremos.
- Skin: Si se selecciona se mostrarán a la derecha del dispositivo una serie de botones, entre los que se incluyen: volumen, on/off, teclas de navegación, retorno, menú, etc.
- Enable keyboard input: Si se selecciona se supondrá que el dispositivo tiene teclado físico, que será emulado por el teclado del ordenador. En caso contrario se utilizará el teclado en pantalla.
- Una vez finalizado pulsaremos el botón **Finish** para crear nuestra máquina virtual.



A continuación pulsar el botón del play que hay en la columna de Actions y arrancará el emulador.

6.5. Ejecución del proyecto

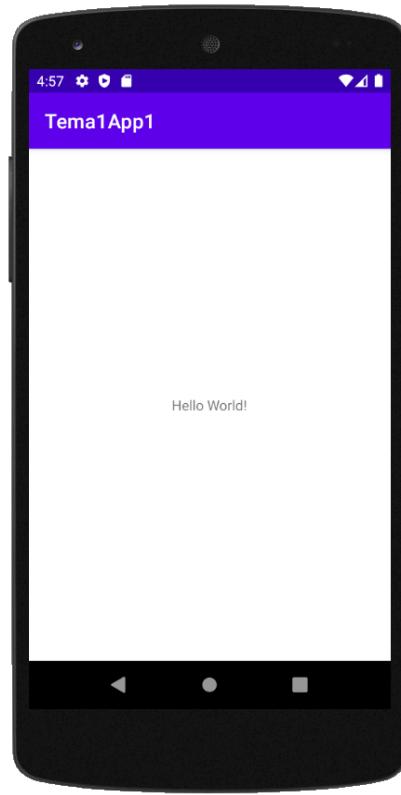
Para ejecutar la aplicación presionamos el **triángulo verde** o seleccionamos del menú de opciones **Run -> Run app** y en este diálogo procedemos a dejar seleccionado el emulador que hemos creado. Si no está arrancado seleccionamos **Launch emulator** y la máquina que hemos creado. A continuación presionamos el botón "**OK**".

Luego de un rato aparecerá el emulador de Android en pantalla (el arranque del emulador puede llevar más de un minuto, según el ordenador del que dispongamos), es **IMPORTANTE** tener en cuenta que una vez que el emulador se ha arrancado no lo debemos cerrar cada vez que hacemos cambios en nuestra aplicación o codificamos otras aplicaciones, sino que volvemos a ejecutar la aplicación con los cambios y al estar el emulador corriendo el tiempo que tarda hasta que aparece nuestro programa en el emulador es muy reducido.

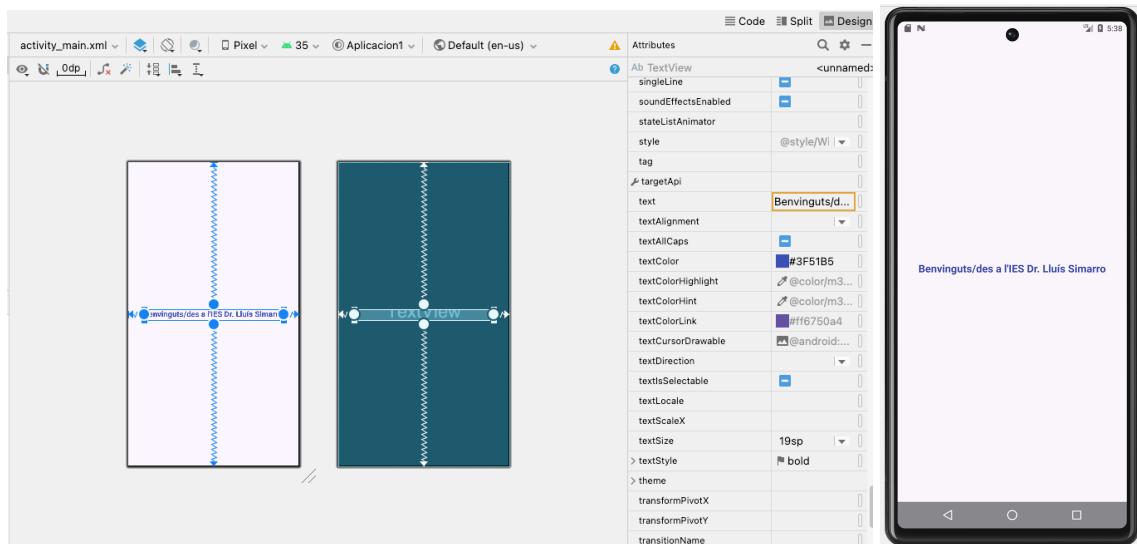
Cuando terminó de cargarse el emulador debe aparecer la interfaz del mismo:



Procedemos a desbloquear la pantalla de inicio y podremos observar la ejecución de nuestra primer aplicación en Android:

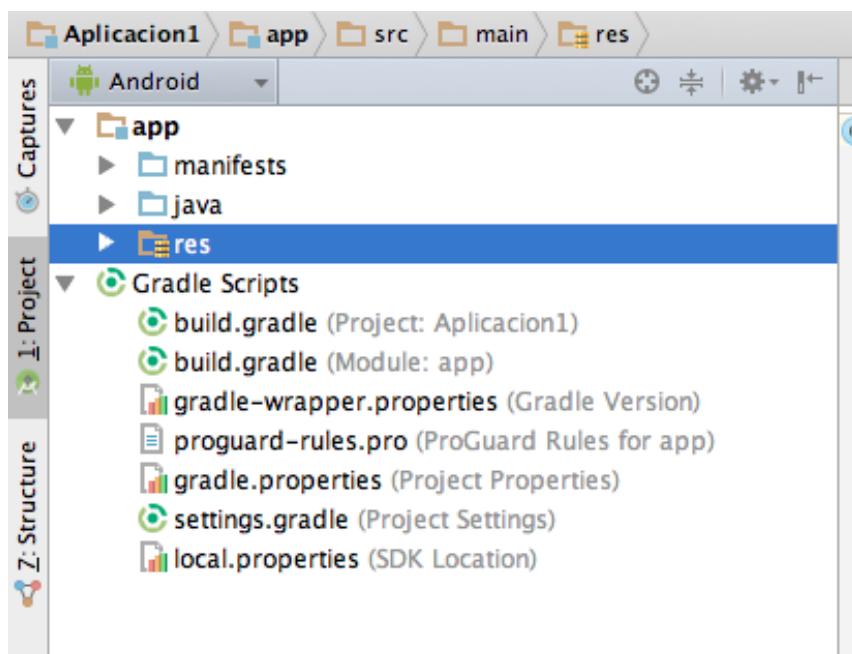


Posteriormente, podemos modificar el texto y sus propiedades con el objetivo de familiarizarse con el diseñador haciendo uso del editor gráfico (o usando el editor XML). En la imagen de la derecha vemos la actualización en el emulador.



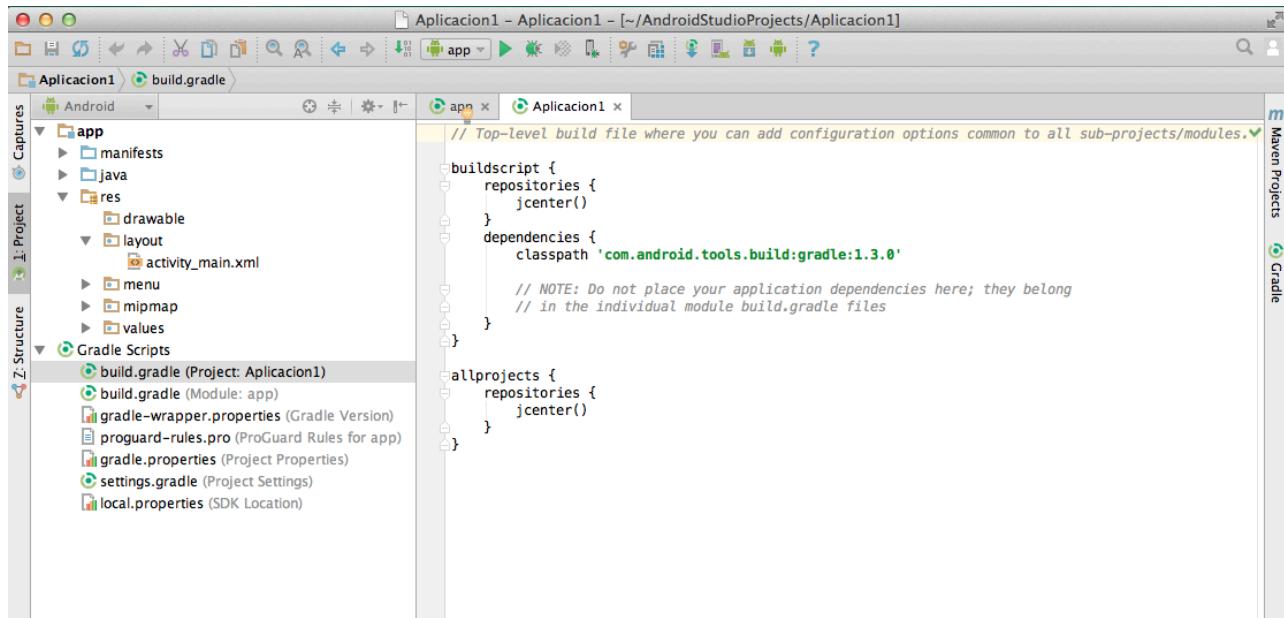
6.6. ¿Qué es Gradle?

Después de contarnos cómo crear nuestra primera aplicación en Android Studio, queremos profundizar un poco en lo que ocurre en la compilación de nuestra aplicación. El proceso que ocurre entre que le damos al botón verde “Play” y la consola muestra por pantalla la salida de nuestra app. Para ello tenemos que hacernos varias preguntas. ¿Qué es gradle en Android Studio?

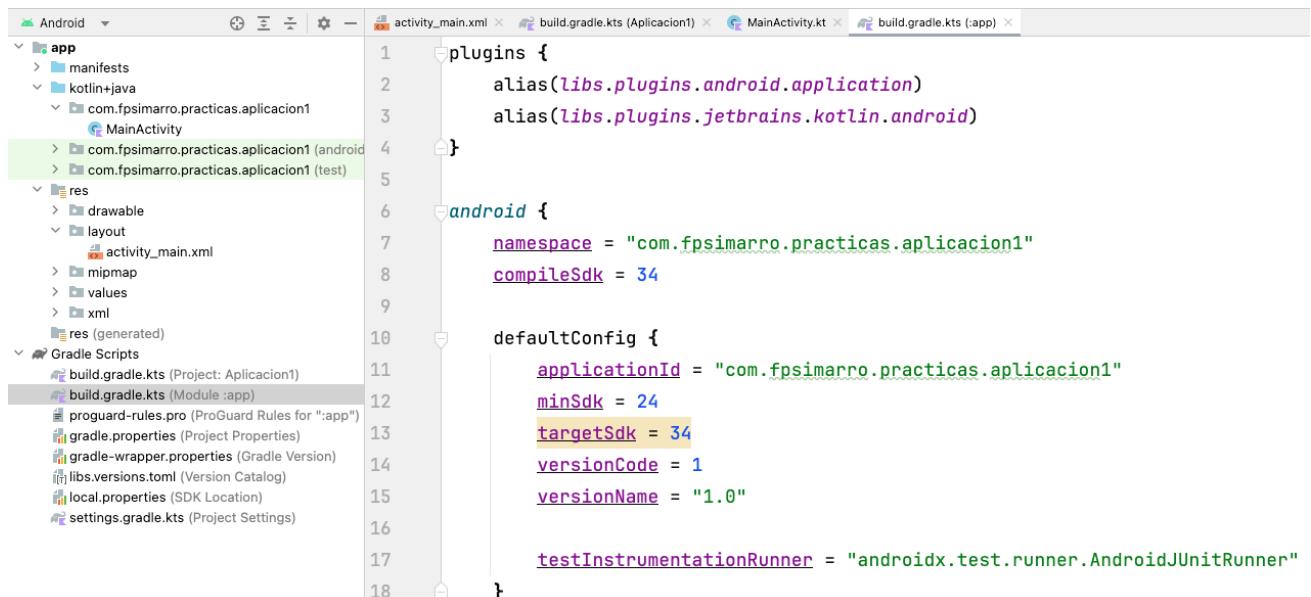


Gradle es un sistema de compilación que reúne en uno solo las mejores prestaciones de otros sistemas de compilación. Está basado en JVM (Java Virtual Machine), lo que significa que puedes escribir tu propio script en java, y que Android Studio lo entenderá y lo usará.

Lo mejor de gradle es que es un plugin, lo que facilita su actualización y su exportación de un proyecto a otro. Esto significa que puedes tener tu propio lenguaje de programación y automatizar el proceso de compilación en un solo paquete (de la misma manera que un jar en caso de java) y poder distribuirlo al resto del mundo.



Google ha creado uno de los sistemas de compilación más avanzados del mercado para permitir a todos los usuarios escribir sus propios scripts sin necesidad de aprender ningún nuevo lenguaje, disminuyendo así la curva de aprendizaje y permitiendo llegar a un mayor público la programación en Android.



¿Qué ventajas tiene Gradle?

- Permite reutilizar fácilmente código.
- Hace sencilla la tarea de configurar y personalizar la compilación.
- Permite la distribución sencilla de código al resto del mundo, y fomenta el trabajo en equipo.
- Gestiona las dependencias de forma potente y cómoda (está basado en Maven).
- Permite la compilación desde consola, lo que nos puede hacer más sencilla la tarea de compilación en sistemas sin el entorno de desarrollo montado.
- Lo más importante es que hace increíblemente fácil la creación de diferentes versiones de la aplicación, por ejemplo para hacer múltiples versiones para móviles o tablets, versiones de pago o gratuitas, etc.