

Caso práctico



El uso generalizado en los últimos tiempos de dispositivos móviles como smartPhone o tablets ha dado lugar a una gran demanda de software para este nuevo tipo de hardware. Es por esta razón por la que **Ada**, socia fundadora de la empresa **BK Programación** y con cierta experiencia en el desarrollo de aplicaciones para dispositivos móviles, ha decidido que su compañía entre también en este nuevo mercado. Se trata de una oportunidad interesante para abrir nuevos negocios tanto para sus clientes actuales como para la captación de otros nuevos.

Hasta este momento la única persona de la empresa que ha trabajado antes con este tipo de tecnologías es la propia **Ada**, de manera que va a ser ella quién se encargará de formar a **Maria** y a **Juan** en estas nuevas tecnologías. En concreto **Ada** ha desarrollado aplicaciones para **Android** utilizando lenguajes como **Java** y **XML**.

1.- Introducción



El gran avance tecnológico de los últimos años en la telefonía móvil unido a la evolución que ha tenido Internet, ha supuesto un gran cambio en el uso que hacemos de los nuevos terminales. Tanto los nuevos dispositivos móviles o smartphones como las tabletas o tablets, están revolucionando el mercado de las telecomunicaciones. Podríamos hablar de los nuevos ordenadores del futuro. Con la aparición de estos dispositivos, han aparecido diferentes plataformas para desarrollar aplicaciones.

Entre estas plataformas se encuentra Android, que apareció a finales de 2008 cuando Google lanzó su primer dispositivo móvil con este sistema operativo. Y desde ahí hasta la actualidad ha incrementado de forma espectacular su cuota de mercado como veremos a lo largo de la unidad.

También veremos qué necesitamos para poder desarrollar una aplicación en Android y desarrollaremos una primera aplicación muy sencilla que podremos probar tanto en un emulador como en nuestro propio dispositivo móvil.

Vamos a comenzar con el estudio de los dispositivos móviles, viendo una clasificación de los mismos y pasaremos a identificar las principales plataformas que existen actualmente para decantarnos finalmente por Android debido a su gran éxito como ya hemos mencionado.

1.1.- Definición de dispositivos móviles



La primera pregunta que podemos hacernos es: ¿qué entendemos por "móvil"?

Si se nos ocurre investigar sobre ese término, a través de algún buscador de Internet, podremos observar que no hay una respuesta única y que en algunas ocasiones las diferencias pueden ser sustanciales en función de qué es lo que consideremos "móvil". Obviamente esto da lugar a su vez a muchas otras preguntas como por ejemplo:

Es móvil... ¿alguna parte del dispositivo? ¿El dispositivo completo? ¿La aplicación que usamos en el dispositivo? ¿Una aplicación cliente? ¿Una aplicación servidor? ¿El usuario del dispositivo? ¿Es "móvil" sinónimo de "portátil"? ¿Es "móvil" sinónimo de "limitado"? ¿Hasta qué punto "móvil" es sinónimo de "autónomo"? ¿Existen diversos grados de "movilidad"? ¿Se pueden clasificar los dispositivos móviles en distintos tipos? ¿Bajo qué criterios?

Y así sucesivamente podríamos plantearnos más y más preguntas... Para evitar este tipo de controversias, en nuestro caso vamos a intentar dar una definición con la que trabajaremos a lo largo del desarrollo del módulo:

RaHu Rodriguez

¿Qué es un dispositivo móvil?

Se trata de un aparato de pequeño tamaño y de poco peso, con pantalla y teclado (puede ser táctil), con pequeñas capacidades de procesamiento, memoria limitada y conexión (permanente o no) a una red. Este tipo de dispositivos están diseñados para cumplir algún tipo de función específica, como realizar llamadas telefónicas, servir como agendas, jugar, navegación GPS, escuchar música, acceso al correo electrónico, navegar por Internet, etc., aunque normalmente pueden llevar a cabo también funciones más generales.

Estos aparatos son cada vez más populares especialmente para aquellos entornos en los que llevar consigo un ordenador convencional (incluso un portátil) no es práctico.

Autoevaluación

¿Cuáles de las siguientes opciones se corresponden con las de un dispositivo móvil?

Alta capacidad de procesamiento.

Conectividad a una red.

Pequeño tamaño.

Cantidad de memoria limitada.

Solución

1. Incorrecto
2. Correcto
3. Correcto
4. Correcto

Para saber más

Para obtener más información acerca de cuáles pueden ser las características de un dispositivo móvil puedes consultar el siguiente artículo en la Wikipedia sobre dispositivos móviles:

[Dispositivo Móvil](#)

1.2.- Clasificación de los dispositivos móviles

¿Qué tipos de dispositivos móviles existen?

Entre los dispositivos móviles más habituales se encuentran:

- ✓ **SmartPhones** o "teléfonos inteligentes".
- ✓ **PDA** (Personal Digital Assistant – asistentes digitales personales) o **PocketPC**(PC de bolsillo) ambos ya en desuso.
- ✓ **Handhelds** (o PC de mano).
- ✓ **Internet tablets**, que se encontrarían entre las **PDA** y los **PC**.
- ✓ **Ultramóviles**. Pueden incluir teclado físico.(pequeños tablet PC).



Según las fuentes que consultes puedes encontrar diversas clasificaciones donde se incluyan unos u otros tipos de dispositivos; en nuestro caso, los principales aparatos a los que nos referiremos al hablar de dispositivos móviles debido a su gran uso, serán los **SmartPhones** y las **tablets** (tabletas). Por otro lado, según la tecnología vaya avanzando te podrás ir encontrando con nuevos tipos de productos y servicios que irán ampliando las posibilidades de elección. Como siempre sucede en el mundo de la tecnología habrá que estar continuamente al día de los nuevos avances que van apareciendo para no quedarse atrás.

Actualmente debido a sus dimensiones hablamos de **Phablets**, que son dispositivos móviles que por su tamaño se acercan a las tablets. Además tenemos los dispositivos **Wearables** (relojes con funciones Mobile)

Autoevaluación

¿Cuáles de los siguientes aparatos podrían considerarse dispositivos móviles según la descripción que se ha realizado anteriormente?

Ordenador portátil.

Teléfono móvil de última generación.

Tabletas.

Reloj de muñeca convencional.

Solución

1. Incorrecto
2. Correcto
3. Correcto
4. Incorrecto

Caso práctico



Maria y Juan acaban de descubrir cómo se les puede abrir todo un mundo lleno de posibilidades en este nuevo entorno del desarrollo de aplicaciones para dispositivos móviles.

Lo primero que deben hacer es informarse adecuadamente de qué se puede y qué no se puede hacer con este tipo de dispositivos. Es decir, qué limitaciones se van a encontrar en estos aparatos para los cuales desean construir programas

Antes de comenzar a desarrollar software para alguno de estos dispositivos, es necesario ser conscientes de las limitaciones con las que nos podemos encontrar en estos aparatos. ¿Cuáles son las restricciones a las que nos vamos a tener que enfrentar?

Hasta hace poco nos encontrábamos con algunas de las siguientes restricciones.

- ✓ Suministro de energía limitado (normalmente dependiente de baterías).
- ✓ Procesadores con capacidad de cómputo reducida. Suelen tener una baja frecuencia de reloj por la necesidad de ahorrar energía. En algunos casos, por ejemplo, podrían no disponer de la capacidad de cálculos en punto flotante.
- ✓ Poca memoria principal (RAM).
- ✓ Almacenamiento de datos persistente reducido (pequeña memoria flash interna, tarjetas SD, etc.).
- ✓ Conexión a algún tipo de red intermitente y con ancho de banda limitado.
- ✓ Pantallas de reducidas dimensiones.
- ✓ Teclados con funcionalidad muy básica y muy pequeños.



Pixaline (Licencia Pixabay)

Este tipo de restricciones, y algunas otras que dependían de cada dispositivo en concreto, habrán de ser tenidas muy en cuenta a la hora del análisis y diseño de una aplicación "móvil", pues no podemos pretender que esa aplicación pueda contener la misma funcionalidad que la que podemos encontrar habitualmente en un programa que es ejecutado en un ordenador de sobremesa o un portátil. Debido a la evolución muchas de ellas han desaparecido, como la limitación en RAM o Procesador, y el tamaño de la pantalla de manera parcial. Sin embargo hablamos de dispositivos de gama alta, teniendo en cuenta que muchos usuarios solo disponen de dispositivos de gama media o baja.

Por otro lado, no todo van a ser restricciones. También habrá que tener en consideración que esta tecnología va a aportar una serie de **ventajas muy importantes**:

- ✓ Movilidad.
- ✓ Poco peso.
- ✓ Pequeño tamaño.
- ✓ Facilidad para el transporte.
- ✓ Conectividad a diversos tipos de redes de comunicaciones (3G, 4G, 4G+, mensajería SMS y MMS, voz, Internet, Bluetooth, infrarrojos, radiofrecuencia, etc.).

Éstas serán las ventajas que podrás explotar en tus aplicaciones.

Para saber más

A través del siguiente vídeo puedes aprender más sobre limitaciones de los dispositivos móviles, cada vez más cubiertas por funcionalidades mejoradas.

Limitaciones de los dispositivos móviles

<http://www.youtube.com/embed/Fm5GsDUScCA>

Natalia Arroyo. Limitaciones de los dispositivos móviles
[Resumen textual alternativo](#)

Citas Para Pensar

Quien consiga solucionar al menos uno de los tres problemas fundamentales se hará rico.

Estos son:

- ✓ Cobertura.
- ✓ Duración de la batería.
- ✓ Limitación de la conectividad a alta velocidad.

2.- Tecnologías disponibles

Caso práctico



Maria y Juan ya tienen claro que programar para un dispositivo móvil no va a ser exactamente lo mismo que programar para un ordenador convencional, debido a las limitaciones y características especiales que aquellos aparatos pueden presentar. Tienen ahora que empezar a descubrir y conocer con qué tipos de tecnologías se pueden encontrar en este nuevo mundo, tanto a nivel de hardware (dispositivos sobre los que se realizarán las aplicaciones) como a nivel de software (sistemas operativos que funcionan sobre esos dispositivos, plataformas de desarrollo disponibles, entornos API, lenguajes de programación, etc.)

Ada, la responsable de **Maria y Juan** en la compañía, va a realizar un pequeño estudio de algunas de las tecnologías que más se están utilizando hoy día. Este estudio será muy importante para el conocimiento que la empresa **BK Programación** quiere ir acumulando para proporcionar valor añadido a los productos y servicios que desea ofrecer a sus clientes. **Ada** aprovechará el desarrollo de este estudio para ir mostrando a **Juan** y a **Maria** las diversas tecnologías con las que se van a poder encontrar.

Es importante que este tipo de análisis y estudios se sigan realizando de manera continua en la empresa al menos una vez al año, pues se trata de un mundo que avanza muy deprisa y en el que es muy fácil quedar desfasado. Hay que estar siempre al día de los últimos productos que salen al mercado así como de los sistemas operativos, las aplicaciones, los entornos de desarrollo, nuevos lenguajes, etc.

Cuando vas a desarrollar una aplicación para un dispositivo móvil, algunas de las primeras preguntas que te puedes hacer son:

- ✓ ¿Sobre qué tipos de dispositivos móviles se pueden hacer programas? ¿Sobre qué tipo de hardware se puede programar?
- ✓ ¿Qué sistema operativo puede llevar ese hardware?
- ✓ ¿Qué plataformas de desarrollo existen para desarrollar sobre ese hardware y ese sistema operativo?
- ✓ ¿Con qué lenguajes puedo programar? ¿Qué herramientas (compiladores, bibliotecas, entornos, etc.) hay disponibles?

Las respuestas a este tipo de preguntas pueden ser múltiples y muy variadas:

- ✓ Respecto al hardware, te puedes encontrar con teléfonos móviles (smartphones), tabletas (tablets) y otros dispositivos. Entre los principales fabricantes de smartphones se encuentran **Samsung** (el mayor proveedor en 2019), **Huawei** que ha desbancado a la tercera posición a **Apple**, y el resto de empresas cuyo ranking de ventas oscila según el trimestre **Xiaomi**, **Oppo**, **Vivo**, **LG**, etc. Un dato interesante es que el mercado de los smartphones es cada vez más asiático. Si nos referimos a las tablets el número de ventas es más variable y depende más de los modelos que hay en el mercado. Como principales fabricantes de tablets están **Apple** (iPad) que domina el mercado, **Samsung** (Samsung Galaxy xxx, y Note), **Huawei**, **Amazon**, etc.
- ✓ En cuanto a los sistemas operativos, dependiendo del hardware habrá sistemas diseñados para unos u otros dispositivos. Los hay basados en **Microsoft Windows**, en **Linux**, y en **MAC OS X**, así como otros totalmente originales y desarrollados específicamente para estos nuevos tipos de dispositivos. Entre los más populares se encuentran **Android**, **iOS** y **Windows Phone**.
- ✓ Si lo que deseas es conocer algo acerca de las plataformas de desarrollo disponibles para cada entorno (hardware y/o sistema operativo), podemos hablar de **Android**, **Java ME**, **Windows Mobile SDK**, **Maemo SDK**, o bien de **IDE** como **Microsoft Visual Studio**, **Code Warrior**, **Eclipse**, **Netbeans** y el recién llegado **Android Studio**, al que **Google** presta soporte.
- ✓ Si te refieres a lenguajes de programación, normalmente te encontrarás con lenguajes que son ya viejos conocidos para otras plataformas, como pueden ser las aplicaciones de escritorio para los **PC** o las aplicaciones web (**Java**, **C#**, **C**, etc.).

En definitiva puedes observar que en este nuevo mundo del desarrollo para dispositivos móviles te encuentras con una problemática similar a la que te puedes enfrentar con los ordenadores convencionales: distintos tipos de hardware, distintas opciones de sistemas operativos dependiendo del hardware que los soporte, diferentes lenguajes de programación, plataformas, API y bibliotecas, entornos de desarrollo, etc.



Tabletas/Videos

Caso práctico

El primer acercamiento que **Ada** va a realizar con **Maria** y **Juan** al mundo de la tecnología de los dispositivos móviles va a ser en lo que respecta al hardware, es decir, lo físico. Éste es el momento en el que tanto **Maria** como **Juan** se hacen toda una serie de preguntas al respecto, tales como:

- ✓ ¿Qué tipo de aparatos hay disponibles?
- ✓ ¿Qué características tienen?
- ✓ ¿Cómo son por dentro?
- ✓ ¿Utilizan un microprocesador como los ordenadores de escritorio o los portátiles?
- ✓ ¿Quién fabrica esos micros?
- ✓ ¿Qué aspecto externo puede tener un dispositivo móvil?
- ✓ ¿Son todos parecidos?
- ✓ ¿Los hay con usos específicos?
- ✓ ¿Qué marcas y qué modelos existen?
- ✓ ¿Cuáles son los más vendidos?



En este apartado **Ada**, como experta en tecnología móvil, va a intentar contestar a algunas de estas cuestiones.

Ya todos hemos tenido la experiencia de tener que comprar un móvil y hemos tenido que tomar la decisión de qué móvil comprar. Esa decisión la tomamos en base a las **características** del teléfono y son las que hacen que un dispositivo sea considerado gama alta, media o baja.

En el año 2019, para que un teléfono fuera considerado gama media tenía que tener al menos estas cinco características:

- ✓ Pantalla FullHD;
- ✓ Un procesador potente (Kirin 710 o Snapdragon 710);
- ✓ Doble cámara (algunos modelos ya tienen tres como Xiaomi Mi A3);
- ✓ Gran batería y carga rápida (mínimo 4.000 mAh);
- ✓ Un gran diseño de forma que ya se abandone el policarbonato por un plástico pulido que aparente ser cristal, porque la estética sí importa.

Nosotros como desarrolladores podemos pensar que nuestra aplicación no tendrá límites, pero la **capacidad** de nuestro terminal sí, tal vez nos hemos encontrado ya con el mensaje "queda poco espacio de almacenamiento interno" sin pensar el espacio que ocupan las aplicaciones, fotografías, videos, audios... Si queremos crear una aplicación que tenga como método de desbloqueo la huella digital tenemos que tener presente que nuestra aplicación sólo podrá ejecutarse en aquellos móviles que tenga el sensor de huellas en su lista de características y el sistema operativo la capacidad de identificar las líneas de la yema y compararlas con la captura almacenada de la huella. Si un dispositivo móvil no tuviera esta característica o capacidad como desarrolladores nos vemos obligados a incorporar otro método de seguridad.



Bru-nQ (Pixabay Licence)

Autoevaluación

Aunque el hardware de dispositivos móviles puede ser muy variado, el sistema operativo que funciona sobre todos ellos es básicamente siempre el mismo (con ciertas variaciones dependiendo del modelo y fabricante del dispositivo) y desarrollado por la misma empresa.

Verdadero Falso

Falso

La afirmación es falsa. Existe una gran diversidad de sistemas operativos dependiendo del tipo de dispositivo. Entre los más populares se encuentran Android, iOS y Windows Phone.

2.2.- Sistemas Operativos

Caso práctico

Una vez que Ada ha mostrado algunas de las peculiaridades hardware de los dispositivos móviles a **Maria** y a **Juan**, es necesario pasar al siguiente nivel: el software, y en concreto el nivel de software más bajo, es decir, el sistema operativo. Nuevamente surgen los interrogantes entre **Maria** y **Juan**:

- ✓ ¿Estos aparatos también tienen sistema operativo como un ordenador?
- ✓ ¿Hay más de un sistema operativo disponible?
- ✓ ¿Hay sistemas operativos "libres" y "propietarios" como sucede con los PC?
- ✓ ¿Quién desarrolla esos sistemas?
- ✓ ¿Con qué herramientas se puede programar para ellos?
- ✓ ¿Funcionan todos los sistemas sobre todos los tipos de hardware?



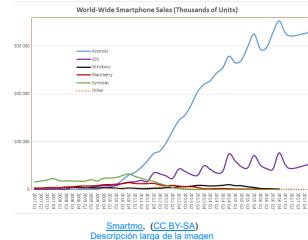
Ésas son algunas de las preguntas que Ada va a intentar responder en esta ocasión.

Los sistemas operativos más habituales que te puedes encontrar en un dispositivo móvil son:

- ✓ **Android**. Desarrollado por Android Inc. y comprada en 2005 por Google, está basado en el núcleo de Linux. El primer fabricante de móviles que lo incorporó fue HTC. Actualmente está en la mayoría de los dispositivos del mercado bajo versiones fragmentadas para cada dispositivo.
- ✓ **iOS**. Desarrollado por Apple para el iPhone y usado más tarde también para el iPod Touch y el iPad.
- ✓ **Windows Phone** (anteriormente llamado **Windows Mobile**). Desarrollado por Microsoft tanto para smartPhones como para otros dispositivos móviles (como por ejemplo, PDA). Algunos fabricantes de teléfonos móviles que incorporan este sistema operativo son Samsung, LG o HTC.
- ✓ **Blackberry OS**. Desarrollado por Research in Motion (RIM) para sus dispositivos Blackberry. (En desuso)
- ✓ **Symbian OS**. La mayoría de teléfonos móviles con Symbian son del fabricante Nokia. (En desuso)
- ✓ **HP webOS**. Desarrollado por Palm, adquirido por HP. Está basado en un kernel de Linux. (En desuso)
- ✓ **Palm OS**. Desarrollado por PalmSource para PDA, aunque las últimas versiones también funcionan para smartPhones. (En desuso)
- ✓ **Maemo OS**. Desarrollado por Nokia para smartPhones, PDA e Internet tablets. (En desuso)
- ✓ **Bada**. Desarrollado por Samsung.

Hay que tener en cuenta que a medida que los dispositivos móviles crecen en popularidad, los sistemas operativos con los que trabajan también adquieren mayor importancia. En el siguiente gráfico comparativo de las cuotas de mercado a nivel mundial de los sistemas operativos para móviles hasta el primer trimestre de 2018 se puede comprobar el creciente aumento de la plataforma **Android** así como el descenso significativo de dispositivos que utilizan **Symbian**. **Apple** se coloca muy de lejos como segunda opción y **BlackBerry** por detrás. Debido a la gran competitividad del mercado, estos datos pueden variar pero una cosa queda clara y es que Android es el sistema operativo para teléfonos inteligentes más popular.

Gráfico comparativo diferentes plataformas



Smartphone (CC BY-SA)

Descripción larga de la imagen

Autoevaluación

El sistema operativo que más cuota de mercado tiene en el primer trimestre de 2018 es **Android** seguido de **iPhone** y **BlackBerry**.

Verdadero Falso

Verdadero

Es cierto. Android ha experimentado un gran crecimiento en su cuota de mercado dejando atrás a iPhone y BlackBerry. Además Symbian ha sido la que más ha bajado.

2.3.- Plataformas de desarrollo y lenguajes de programación

¿Qué necesitamos para poder programar dispositivos móviles?

Para poder desarrollar aplicaciones para alguno de los anteriores sistemas operativos es necesario disponer de alguna plataforma de desarrollo que permita generar código ejecutable sobre esos sistemas, o bien sobre alguna máquina virtual; o algún intérprete que esté instalado en el dispositivo y que sea soportado por el sistema operativo.

Entre las plataformas de desarrollo para móviles más populares se encuentran normalmente las que los propios autores de los sistemas operativos ofrecen para trabajar sobre su plataforma. Algunas de las más conocidas son:

- ✓ **Android.** Google proporciona también de manera gratuita el **Android SDK** para programar aplicaciones en Java sobre su sistema operativo. El IDE más utilizado es **Eclipse** con el plugin **ADT (Android Development Tools)**. También es posible programar en otros lenguajes como C o C++ gracias al uso del **Android NDK (Native Development Kit)** que permite generar código nativo (native code), frente al código gestionado (managed code), que se ejecuta sobre una máquina virtual, como es el caso de **Java** o **C#**.
- ✓ **iOS.** El **SDK** también se puede descargar gratis, pero para poder comercializar el software a través de su tienda de aplicaciones hay que registrarse en el programa de desarrollo del **iPhone**, lo cual **no es gratuito**. El lenguaje de programación en este caso es **Objective-C**.
- ✓ **Windows Phone.** Para desarrollar aplicaciones sobre **Windows Phone** pueden utilizarse las tecnologías **Silverlight**, **XNA** y **.NET Compact Framework de Microsoft**. Las herramientas **Visual Studio 2010 Express** y **Expression Blend para Windows Phone** son ofrecidas gratuitamente por Microsoft. El lenguaje más habitual para el desarrollo ha sido **C#**, aunque la idea es que se pueda utilizar también otros lenguajes de la plataforma **.NET** como **Visual Basic .NET**.
- ✓ **Java ME.** Este caso sería una excepción respecto a los demás, pues no se trata de una serie de herramientas (bibliotecas, compiladores, IDE, etc.) asociadas a un sistema operativo, sino de un subconjunto de la plataforma **Java** orientada para el desarrollo sobre dispositivos móviles. Se programaría en lenguaje **Java** y sería necesario que hubiera instalada una máquina virtual en el sistema operativo del dispositivo sobre el que se deseara ejecutar la aplicación desarrollada como sucede por ejemplo en **Symbian** (que se puede programar en modo nativo o bien con **Java ME**).
- ✓ **Android Studio.** La última plataforma para desarrollo Android específico con soporte de Google.



Autoevaluación

¿Cuáles de las siguientes plataformas están basadas en Java?

Windows Phone.

Android.

iOS.

Java ME.

Solución

1. Incorrecto
2. Correcto
3. Incorrecto
4. Correcto

Caso práctico



Dado que **Maria** y **Juan** saben programar en Java, pues lo aprendieron en el módulo de Programación, y que **Ada** ha decidido elegir una opción que esté relacionada con este lenguaje de programación, las alternativas que les quedan son Java ME y Android. En cualquier caso han de ser siempre conscientes, al igual que ya lo fueron en el módulo de Programación, que no se trata más que de una de las posibles alternativas disponibles en el mercado y que más adelante (en futuros proyectos dentro de la empresa o bien en alguna otra empresa) es probable que quizás tengan que trabajar con otro lenguaje (por ejemplo C#, C++, Python, Objective-C) para otras plataformas.

En nuestro caso **Java** puede ser una buena elección dado que es el lenguaje con el que aprendiste a programar en el módulo de Programación y por tanto no tendrás que aprender un nuevo lenguaje y podrías dedicarte de lleno al aprendizaje de las características específicas para el desarrollo de aplicaciones para dispositivos móviles. Por otro lado, un lenguaje **multiplataforma** como **Java** es también muy adecuado para un ciclo con un nombre como "**Desarrollo de Aplicaciones Multiplataforma**". Pero en cualquier caso no debes olvidar que se trata de una de las muchas opciones con las que te puedes encontrar en el mercado, y que estas alternativas irán apareciendo y desapareciendo constantemente en función del éxito que vayan obteniendo.

Respecto a la plataforma, habría que elegir entre aquellas que estén relacionadas con **Java** (**Java ME** o **Android**). Finalmente se ha decidido **Android** debido a que la gran cuota de mercado que ha alcanzado esta reciente plataforma nos permitirá instalar las aplicaciones que creemos en multitud de dispositivos. Además para su desarrollo, Google proporciona a los desarrolladores todas las herramientas totalmente gratuitas como hemos visto en el apartado anterior. No hay que olvidar que la gran competitividad en este mercado hace que tengamos que estar atentos a los nuevos cambios y necesidades. Hay que intentar siempre formarse en aquellos entornos más demandados sin olvidar los que pueden venir detrás.



2.5.- Introducción a la plataforma Android

Android Inc., una pequeña compañía que se dedicaba a desarrollar aplicaciones para dispositivos móviles, es comprada por Google en 2005 y comienzan a desarrollar una máquina virtual Java para móviles llamada Dalvik VM.

Con la creación en 2007 del consorcio Handset Alliance (integrado por varias compañías como Google, Intel, Motorola, Samsung, Ericson, etc.) se persigue la difusión de la plataforma Android. Para ello, se lanza una primera versión de Android SDK y en 2008 aparece el primer móvil con Android. También se libera el código fuente de Android con licencia Software Apache, una licencia de software libre y de código abierto a cualquier desarrollador. Aparece Android Market, donde se pueden descargar todo tipo de aplicaciones. En 2009 se incorpora en una nueva versión de Android el teclado en la pantalla y ya en 2010 Android se convierte en una de las plataformas de móviles más utilizadas. En 2011 se desarrollan versiones específicas para tabletas y en 2012 Google reemplaza su tienda de descargas online Android Market por Google Play Store desde donde se pueden descargar además de aplicaciones, contenidos de forma online.

Como hemos visto, a lo largo de la historia de Android, han ido apareciendo diferentes versiones en las que se han ido incluyendo clases y métodos para añadir nuevas funcionalidades. Por tanto, será necesario a la hora de desarrollar una aplicación elegir la versión para la cual se implementará. Hay que tener en cuenta que las nuevas versiones no eliminan ninguna de las versiones anteriores, se etiquetarán como obsoletas pero se pueden seguir utilizando.

Para identificar las diferentes plataformas de Android se puede hacer de diferentes formas:

- ✓ La **versión**, números separados por el operador punto (.) utilizado en muchos productos.
- ✓ Nivel de **API**, son números enteros comenzando por el 1.
- ✓ **Nombre comercial**, son nombres de postres ordenados alfabéticamente (Cupcake, Donut, Éclair, etc.)

Algunas de las principales plataformas de Android son:

Principales plataformas de Android.

Versión	Nivel de API	Nombre comercial	Fecha
v1.0	1		Septiembre 2008
v1.1	2		Febrero 2009
v1.5	3	Cupcake	Abril 2009
v1.6	4	Donut	Septiembre 2009
v2.0	5	Éclair	Octubre 2009
v2.1	7		Enero 2010
v2.2	8	Froyo	Mayo 2010
v2.3	9	Gingerbread	Diciembre 2010
v3.0	11	Honeycomb	Febrero 2011
v3.1	12		Mayo 2011
v3.2	13		Julio 2011
v4.0	14	Ice Cream Sandwich	Octubre 2011
v4.0.3	15		Diciembre 2011
v4.1	16	Jelly Bean	Julio 2012
v4.2	17		Noviembre 2012
v4.3	18		Julio 2013
v4.4	19	Kitkat	Octubre 2013
v4.4.1	10		Diciembre 2013
v4.4.2	19		Diciembre 2013
v4.4.3	19		Junio 2014
v4.4.4	19		Junio 2014
v4.4w	20	Kitkat con extensiones	Agosto 2014
v.5	21	Lollipop	Noviembre 2014
v5.1	22		Marzo 2015
v6.0	23	Marshmallow	Octubre 2015
v7.0	24	Nougat	Mayo 2016
v7.1.2	25		Abri 2017
v8.0	26	Oreo	Agosto 2017
v8.1	27		Enero 2018
v9.0	28	Pie	Agosto 2018
v10	29	Android 10	Septiembre 2019



Como puedes comprobar en la tabla anterior, las actualizaciones de Android incorporando mejoras se publican con mucha frecuencia. Como consejo, hay que intentar programar nuestras aplicaciones para que funcionen en la mayor cantidad de dispositivos móviles, por tanto, es interesante conocer qué versión o versiones están más extendidas entre los dispositivos móviles que utilizan Android.

Para saber más

Si quieres conocer qué cambios se han ido produciendo en las distintas versiones, puedes consultar el siguiente enlace.

[Versiones de Android en su página oficial.](#)

Autoevaluación

La máquina virtual para dispositivos móviles basados en la plataforma Android recibe el nombre de...

- JRE.
- Dalvik.
- SDK.
- Android Machine.

No es correcto.

Correcto, Dalvik se basa en Java y requiere poca memoria.

Incorrecto.

No es la opción correcta.

Solución

1. Incorrecto
2. Opción correcta
3. Incorrecto
4. Incorrecto

3.- Arquitectura del sistema Android

Caso práctico

Una vez que Ada ha decidido que en la empresa **BK Programación** se comience a trabajar en el entorno Android con el lenguaje de programación Java, lo primero que **María y Juan** tienen que hacer es empezar a analizar cómo es este nuevo entorno. Dado que ambos ya tienen ciertos conocimientos de programación en lenguaje Java, las preguntas que se hacen ahora son del tipo:

- ✓ ¿Qué tipo de API nuevas (paquetes, clases, métodos, etc.) hay que aprender?
- ✓ ¿Existen nuevos conceptos relacionados con estas API?
- ✓ ¿Qué estructura tienen?
- ✓ ¿Qué máquina virtual se utiliza?
- ✓ ¿Qué características mínimas deben cumplir los dispositivos para poder ejecutar aplicaciones desarrolladas bajo este entorno?
- ✓ ¿Con qué herramientas de desarrollo vamos a trabajar?

Ada intentará resolver algunas de estas dudas.



Android está estructurado en varias capas. Aquí, vamos a centrarnos en las más importantes, que son las siguientes:

- ✓ El **núcleo de Android** (capa más interna) es Linux. En esta capa se gestionan los servicios de gestión de memoria y de procesos, de servicios, la pila de red y los controladores para los dispositivos. Esta capa depende del hardware y actúa como un nivel de abstracción entre el software y el hardware.
- ✓ Las **librerías nativas** incluyen una serie de bibliotecas escritas en C/C++ y que se muestran a los desarrolladores a la hora de implementar aplicaciones como por ejemplo las bibliotecas de gráficos (Surface Manager) y bibliotecas de medios (Media Framework), el motor de base de datos (SQL Lite), un navegador para Android (WebKit), etc.
- ✓ El **Runtime de Android** es la capa donde se encuentra la máquina. La primera máquina creada por Android se llamaba Dalvik y fue una adaptación de la máquina virtual Java estándar debido a que ésta no podía ser utilizada por las limitaciones que presentan los dispositivos móviles como son la escasa memoria y un procesador muy limitado. La máquina virtual Dalvik estaba preparada para ejecutar los distintos procesos de las aplicaciones Android optimizando la memoria disponible. Cuando Oracle adquirió todos los derechos de Sun, empezaron los problemas legales, ya que Dalvik usaba la máquina virtual de Java creada por Sun. Así que Google con Android 4.4 KitKat lanzó ART o Android RunTime como alternativa de Dalvik, y se convirtió en el sustituto definitivo a partir de Android Lollipop. En esta capa también se incluyen las librerías disponibles en lenguaje Java (Core Libraries).
- ✓ El **entorno de aplicación** está formado por un conjunto de herramientas útiles para el desarrollo de cualquier aplicación Android. Los desarrolladores de aplicaciones tendrán acceso a los mismos API del framework usados por las aplicaciones ya sean propias del dispositivo, creadas por Google o incluso las que haya creado el propio desarrollador. Algunas de las API que podremos encontrar aquí serían:
 - ◆ **Activity Manager:** gestiona el ciclo de vida de las aplicaciones Android.
 - ◆ **Window Manager:** gestiona las ventanas de las aplicaciones y utiliza la librería Surface Manager.
 - ◆ **Telephone Manager:** gestiona las funciones propias del teléfono como llamadas y mensajes.
 - ◆ **Views:** elementos para construir interfaces de usuario (parte visual de los componentes como Button, TextView, etc.).
 - ◆ **Content Provider:** mecanismo para compartir los datos entre las distintas aplicaciones Android, así por ejemplo podremos acceder a la agenda, los mensajes, etc., desde otras aplicaciones.
 - ◆ **Notification Manager:** comunican al usuario eventos que ocurren en la ejecución de las aplicaciones (ej: llamada entrante, conexión Wi-Fi disponible, etc.).
- ✓ En la **capa de aplicaciones** se encuentran instaladas todas las aplicaciones Android (tanto las que vienen por defecto con el terminal como las que descarga de otros desarrolladores o las suyas propias). Todas correrán en la máquina virtual Dalvik utilizando los servicios, API y librerías de las capas inferiores.



[Guía para desarrolladores Android](#) (Todos los derechos reservados)

Autoevaluación

¿Qué librerías nativas incluye Android?

- SQL Lite.
-
- Dalvik.
-
- WebKit.
-
- Activity Manager.
-

[Mostrar retroalimentación](#)

Solución

1. Correcto
2. Incorrecto
3. Correcto
4. Incorrecto

3.1.- Funcionalidades de Android

¿Qué nos permite hacer un dispositivo móvil que utiliza la plataforma Android?

Si eres usuario habitual de Android, conocerás muy bien su gran variedad de usos pudiendo realizar llamadas de voz, enviar [SMS](#) o [MMS](#) e incluso conectarnos a la red para navegar por Internet, descargarnos ficheros, recopilar datos mediante los sensores que tiene el dispositivo, etc.

En este bloque del módulo aprenderemos las principales funcionalidades de Android, dejando a un lado otras más avanzadas debido a que este módulo tiene una carga horaria limitada y no es posible profundizar en muchos aspectos.

Entre las funciones básicas que ofrece Android y que desarrollaremos en las siguientes unidades de este módulo están:

- ✓ [Creación de aplicaciones de usuario](#) utilizando interfaces de usuario interactivas.
- ✓ [Gestión de los datos](#) en una aplicación.
- ✓ [Procesamiento y reproducción de objetos multimedia](#) en una aplicación.



Toshi Levent-Levi

Otras funcionalidades de Android son:

- ✓ Envío y recepción de mensajes de texto y multimedia utilizando los permisos adecuados.
- ✓ Recepción de mensajes tipo Push.
- ✓ Utilización de servicios en segundo plano mediante hilos.
- ✓ Utilización de sensores de los dispositivos móviles.
- ✓ Conexión de red vía Wi-Fi y bluetooth.
- ✓ Manejo de conexiones [HTTP](#) y [HTTPS](#).

El propósito de este módulo es acercar al alumnado la tecnología Android y comenzar a desarrollar aplicaciones sencillas utilizando las herramientas apropiadas para ello. Así mismo se desarrollarán los contenidos necesarios para implementar una aplicación con distintas interfaces de usuario donde se puedan guardar y manejar datos y por supuesto utilizar objetos multimedia. No obstante, si después de completar el módulo deseas emplear más funcionalidades en tus aplicaciones puedes regresar aquí para consultar algunas de las páginas webs que te proponemos.

Para saber más

Cuando hayas creado una aplicación con las funciones básicas que aprenderemos en este bloque de cuatro unidades, podrás añadir si lo deseas más funcionalidades a tus aplicaciones. Para ello podrías consultar los siguientes enlaces:

- ✓ Servicios para descargar ficheros desde Internet o escuchar música sin bloquear una aplicación: [Servicios de Android](#)
- ✓ Intents implícitos para realizar llamadas, enviar mensajes, lanzar el navegador Web, etc. [Uso de intents implícitos](#).
- ✓ Sensores de los dispositivos móviles con los cuales una aplicación puede recibir datos del medio externo como la temperatura, la orientación, etc. [Manejo de sensores](#)
- ✓ Conexiones de red: consulta de conexiones disponibles, comprobación del estado de la conexión, etc. [La clase ConnectivityManager](#)

4.- Entorno de desarrollo para aplicaciones en Android

Caso práctico

Una vez realizado un repaso general sobre la arquitectura del sistema Android, **María** y **Juan** discuten acerca de lo que es necesario tener instalado en el ordenador para poder empezar a desarrollar algo.

- ✓ ¿Qué bibliotecas adicionales hay que instalar para poder desarrollar con Android?
- ✓ ¿Qué IDE es el más apropiado para desarrollar aplicaciones Android?



Ya en un apartado anterior ya mencionamos por qué escogimos **Java** como lenguaje para desarrollar Android. A continuación explicaremos los motivos que hicieron de este lenguaje la mejor opción y qué otras opciones tenemos actualmente.

Inicialmente Google pensó en Java como lenguaje para programar aplicaciones para Android ya que es un lenguaje fácil, muy extendido y es independiente de la plataforma con lo cual una aplicación puede ejecutarse en cualquier dispositivo sin tener en cuenta su hardware. Sólo es necesario una Máquina Virtual Java (JVM) para ejecutarse en cualquier sistema operativo que soporte esta máquina, actualmente casi todos. Java es un lenguaje que a medida que se lanzan nuevas versiones, hace que las antiguas también sigan siendo soportadas. Y aunque Android Studio acepta todas las funcionalidades de Java 7 sólo incluye ciertas funcionalidades de Java 8 con lo cual los desarrolladores se ven limitados a la hora de optimizar las aplicaciones.

Es en 2017 cuando Google anuncia que **Kotlin** pasa a ser un lenguaje oficial en Android y su IDE Android Studio tendría soporte para el nuevo lenguaje. Kotlin es un lenguaje nuevo y suple los problemas de Java al desarrollar aplicaciones para Android con lo cual crea código más sencillo y simple, de ahí que vaya ganando popularidad. Además es 100% compatible con Java: el código de Java y Kotlin pueden convivir en un mismo proyecto sin problemas, las librerías en código Java se pueden importar y ejecutar en un proyecto Kotlin y viceversa. En nuestro caso **aprovecharemos nuestros conocimientos de Java** y así **evitar la curva de aprendizaje de un lenguaje nuevo, lo que no tendría cabida en las horas de este módulo**.

No obstante, adquirir más conocimientos, experiencia y práctica en Kotlin, te aportará una ventaja competitiva con respecto a otros programadores, es algo que no deberías perder de vista.

Finalmente utilizaremos el entorno de **Android Studio** que es el IDE oficial para desarrollar aplicaciones Android y el que Google actualiza con las últimas herramientas para crear aplicaciones en cada tipo de dispositivo Android. Como podrás comprobar, ya existe mucha documentación y páginas web que hacen referencia a este nuevo entorno.

Normalmente tu sistema operativo tiene ya instalado Java pero **es necesario tener JDK 7 para Android Lollipop o superior**. Para comprobar si tienes instalado el JDK y la versión abre una ventana de comandos o terminal y escribe javac -version. Si no tienes Java o la versión es inferior puedes descargar el JDK desde su página oficial [Java SE Download](#).

Antes de instalar el IDE daremos unas pinceladas sobre el grupo de herramientas que permiten la programación de aplicaciones móviles conocido como **SDK** de Android. Actualmente la instalación del kit de desarrollo forma parte del proceso de instalación del IDE con lo cual cuando finalice la instalación de Android Studio tendremos todo el entorno listo para crear nuestra primera aplicación. Dentro de este paquete viene incluido:

- 1.- El conjunto de clases o API para el desarrollo de nuestras aplicaciones.
- 2.- La herramienta para depurar código.
- 3.- Un emulador que nos permitirá ejecutar nuestra aplicación en un dispositivo virtual con unas características específicas.

Vamos a ver cómo se instala y crearemos nuestro primer proyecto.

Debes conocer

El siguiente enlace a la documentación oficial de desarrolladores de Android te ofrece una guía de instalación por cada sistema operativo. Ten en cuenta que antes debes descargarte el fichero de instalación acorde a tu sistema operativo:

[Versión más reciente de Android Studio](#)

[Guía de instalación Android Studio](#)

En el siguiente enlace puedes seguir los pasos de cómo instalar **Android Studio para Windows 64 bits**.

[Instalar Android Studio en tu PC](#)

5.- Desarrollo de una aplicación sencilla en Android

Caso práctico

Ha llegado la hora de ponerse manos a la obra.

María va a crear una primera aplicación sencilla que le permita ver alguno de los componentes básicos que **Ada** les ha explicado en funcionamiento. Su primera aplicación consistirá en hacer un “**Hola Mundo**”, algo muy típico cuando comenzamos a desarrollar aplicaciones para una nueva plataforma o lenguaje utilizando además un IDE con el que tenemos que familiarizarnos.

Cuando tiene el entorno preparado para comenzar, le surgen ciertas dudas:

- ✓ ¿Por dónde comenzar?
- ✓ ¿Qué tipo de ficheros hay que crear?
- ✓ ¿Qué carpetas componen el proyecto?
- ✓ ¿Dónde se guardan los códigos fuente?
- ✓ ¿Qué ficheros genera automáticamente el IDE al crear un proyecto?

Ada les comenta las principales carpetas y ficheros que se van a encontrar en cualquier proyecto y les explica lo más importante de cada uno de ellos.



Ya tenemos instalado el IDE y vamos a desarrollar una primera aplicación muy simple que sacará por pantalla el famoso "Hola mundo".

Para nombrar el proyecto y el resto de componentes utilizaremos el inglés con lo cual el nombre de nuestro proyecto será **HelloWorld**. No te preocupes si no sabes alguna palabra, siempre se puede utilizar el [Traductor de Google](#).

Algunos aspectos que debemos tener en cuenta cuando creamos nuestra aplicación son los siguientes:

1. Si es la primera vez que se ejecuta Android Studio, en la pantalla de Bienvenida se debe seleccionar la opción **Start a new Android Studio Project**.
2. Seleccionar el tipo de proyecto, en nuestro caso **Phone and Tablet**, y partimos de una actividad vacía o **Empty Activity**.
3. A continuación nos sale la pantalla para configurar el proyecto. Según la versión de Android Studio puede variar pero siempre nos pedirá los siguientes datos:
 - ✓ El **nombre de la aplicación (Name)** es el nombre que le daremos al proyecto para identificarlo.
 - ✓ El **nombre de la compañía (Package Name)** es el dominio con el que crearemos nuestra aplicación y que adoptará el nombre del paquete. Cuando se publica una aplicación en [Google Play](#) se debe garantizar que es única, de ahí que se utilice el nombre del dominio junto con el nombre de aplicación. Posteriormente cuando un usuario instala nuestra aplicación, el nombre de paquete será la ruta de instalación de nuestra aplicación en el dispositivo y se garantiza que no hay ninguna otra aplicación con la misma ruta instalada y el mismo nombre.

Para el proyecto que estamos creando será `com.example.helloworld`. Con este paquete se indica que es un proyecto de ejemplo que nunca se publicará en [Google Play](#)

✓ La **localización del proyecto (Save Location)** es la ruta donde se guardará el proyecto con toda su estructura de carpetas y ficheros.

✓ La **versión mínima (Minimum SDK)**: Hay que indicar el valor mínimo de la **API** que se requiere para que se ejecute nuestra aplicación. Este valor es importante cuando se crea un dispositivo virtual ya que la versión de Android de ese dispositivo debe ser igual o superior a este valor, si no fuera así, nuestra aplicación no de podría instalar en dicho dispositivo. Aparece un enlace **Help me Choose** que nos muestra el porcentaje de dispositivos que tiene cada versión de Android.

Vamos a ver qué ficheros y carpetas se han generado y así podremos empezar a entender con algo más de detalle cómo se estructura una aplicación Android

Debes conocer

En el siguiente vídeo puedes ver cómo se crea una primera aplicación Android.

Cómo empezar un primer proyecto en Android Studio

<http://www.youtube.com/embed/9dgxuYW0YOs>

[Resumen textual del video](#)

Autoevaluación

Dos aplicaciones de Android podrían tener el mismo Package Name en Google Play.

Verdadero Falso

Falso

Es falso. El nombre del paquete debe ser un nombre único para identificar la aplicación en Google Play.

5.1.- Estructura de un proyecto Android

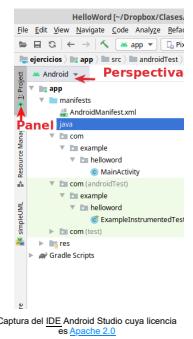
Android Studio es un entorno que facilita mucho la tarea de desarrollo de aplicaciones utilizando asistentes y herramientas que generan, por ejemplo, la estructura de carpetas necesarias del proyecto, los ficheros para la correcta configuración, así como ayudar en el diseño de las interfaces de usuario. Para ver la estructura de un proyecto Android hay que tener en cuenta lo siguiente:

- 1.- El listado de archivos se muestra en el **panel Project** de la barra de la ventana de herramientas.
- 2.- Este listado puede variar en base a la vista o **perspectiva** que esté seleccionada, que por defecto suele ser la **vista Android** como muestra la imagen de la derecha.

Esta estructura de carpetas que se muestra en la **vista Android** no es la estructura real de carpetas del proyecto que se obtendría seleccionando la **vista Project**. Una vez hayas seleccionado esta vista identificaremos las principales carpetas y ficheros que se han creado en nuestra primera aplicación. No tienen por qué crearse todas las carpetas que se especifican a continuación, pero es importante que conozcas dónde se guardan cada uno de los ficheros y su finalidad.

Además, un proyecto creado en Android Studio puede estar formado por varios módulos, pero nuestro proyecto se estructura en un único módulo principal llamado app donde están los ficheros de la aplicación:

- ✓ **/libs/** ficheros de tipo **JAR** donde se incluyen las librerías que se utilizarán en el proyecto.
- ✓ **/build/** esta carpeta contiene ficheros generados automáticamente cuando se compila el proyecto.
- ✓ **/src/main/java** almacena los ficheros fuente (**.java**) de la aplicación. Android Studio crea el código básico de la pantalla principal de la aplicación (**Activity**), en este caso llamado **MainActivity.java**.
- ✓ **/src/main/res** contiene todos los recursos que necesitará la aplicación como imágenes, vídeos, cadenas de texto, etc. Se verá en profundidad en las siguientes unidades. Algunas de las subcarpetas que podemos destacar serían:
 - ❖ **/res/drawable/** almacena las imágenes y elementos gráficos dependiendo de la resolución y densidad de las pantallas.
 - ❖ **/res/mipmap/** almacena los iconos para lanzar la aplicación y al igual que en el caso anterior, se divide en subcarpetas dependiendo de la resolución y densidad de las pantallas.
 - ❖ **/res/layout/** contiene ficheros **XML** que define las pantallas de la interfaz gráfica.
 - ❖ **/res/values/** contiene ficheros **XML** como por ejemplo las cadenas de textos, estilos, colores, etc.
 - ❖ **/res/menu/** contiene los menús de la aplicación.
- ✓ **/src/main/AndroidManifest.xml** este fichero contiene los aspectos más importantes de la aplicación como el nombre, versión, icono, pantallas, mensajes, etc. Lo estudiaremos más adelante.
- ✓ **build.gradle** este fichero almacena la información necesaria para compilar el proyecto.



Captura del IDE Android Studio cuya licencia es Apache 2.0

Debes conocer

Es necesario que te familiarices con la interfaz de Android Studio y conozcas el entorno junto con las herramientas que te ayudan en el desarrollo de una aplicación. El primer enlace es una introducción de la documentación oficial:

[Descripción general de Android Studio](#)

Y este segundo enlace es una página que describe con detalle toda interfaz de usuario de Android Studio:

[Tutorial De Android Studio: Navegación En La Interfaz](#)

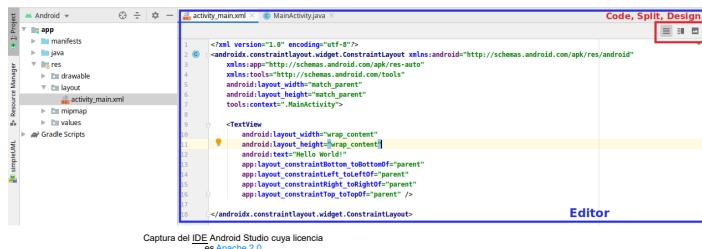
5.2.- Diseño de la pantalla principal



En cualquier aplicación Android, vamos a encontrar una **pantalla principal** cuando ejecutemos la aplicación generada. Esta pantalla contendrá diferentes elementos organizados de una forma determinada. Nos referimos a su diseño, el fichero con extensión **XML** lo hemos nombrado **activity_main** y se encuentra en la carpeta **/app/src/main/res/layout/**. Este fichero es el encargado de definir el diseño (**layout**) de los componentes de tipo **view** que contendrá la aplicación.

La pantalla podemos verla de forma gráfica o bien el código generado. Tan sólo hay que seleccionar cada una de las pestañas que Android Studio nos muestra en la parte inferior: **Design** (para ver de forma visual cómo quedarán la pantalla principal) o bien **Text** (para ver su código en **XML**) en versiones inferiores a Android Studio 3.6. En versiones posteriores, se cambia de vista mediante los iconos que se ubican en la parte superior derecha del **Editor** como muestra la siguiente imagen: **Code** (que muestra el código), **SPLIT** (que muestra el código y el diseño) y **Design** (que muestra el resultado final).

Espacio CAMON



Captura del IDE Android Studio cuya licencia es Apache 2.0

Fíjate cómo el **IDE** nos ayuda resaltando las diferentes palabras del código mediante colores, lo que nos facilita entender el código separando los conceptos por colores:

- ✓ En **morado** aparece el espacio de nombres que se declara en el nodo raíz mediante el atributo `xmlns:<name>=<"uri">`, donde `<name>` es el prefijo de espacio de nombres y `<"uri">` es el **URI** que identifica el espacio de nombres.
- ✓ En **azul** aparece el atributo que siempre tiene como prefijo su espacio de nombres.
- ✓ En **verde** el valor que toma el atributo.

Pasemos a estudiar el código:

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Por el momento nos interesará conocer de este fichero que está formado por el nodo **ConstraintLayout**, que a su vez contiene un componente de tipo **View**, concretamente un **TextView** con el texto que mostrará la aplicación "Hello World!"

Ejercicio Resuelto

Modifica la posición del **TextView** que contiene el texto "Hello World!" para que quede centrado en la pantalla. Reemplaza el texto por el siguiente "Bienvenido a Android".

Mostrar retroalimentación

Añadir en la etiqueta > las siguientes líneas de código:

```
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
```

Para cambiar el texto Hello World! por Bienvenido a Android, modificamos el fichero strings.xml de la carpeta /res/values/ en la siguiente línea:

```
<string name="hello_world">Bienvenido a Android</string>
```

Para saber más

En el siguiente video puedes ver cómo se puede alternar entre las diferentes vistas **Text** y **Design** en una versión de Android Studio inferior a la 3.6. Este video servirá de ayuda para conocer otros paneles del **Editor** como **Component Tree** que muestra la jerarquía de objetos View, el panel **Palette** donde se encuentran todos los **widget** o vistas que se pueden usar en un diseño y el panel **Attributes**.

Conocer el Editor Visual de Android Studio

<http://www.youtube.com/embed/pXp9WN-muRc>

Resumen textual del video

5.3.- Mostrar la pantalla principal

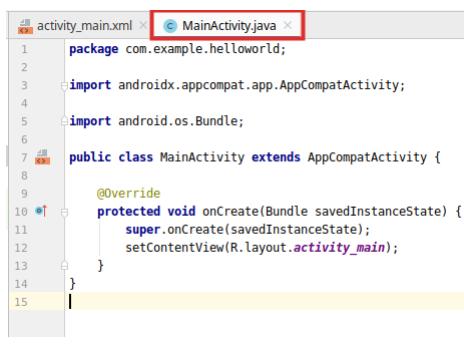
El siguiente fichero que veremos es la clase `MainActivity.java` que se almacena en la carpeta `/app/src/main/res/java/` y dentro del paquete llamado `com.example.helloworld`. Este fichero es el encargado de que se cargue la pantalla principal (**Layout**) estudiada en el código anterior en nuestra **Activity**.

```
package com.example.<span>helloworld</span>;<br /><br />
import androidx.appcompat.app.AppCompatActivity;<br />
import android.os.Bundle;<br /><br />
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Es mediante el método `setContentView(R.layout.main)` que indicamos que se visualice un recurso a través de la clase estática llamada `R.layout`. Esta clase referencia a todos los ficheros XML que se encuentran en el directorio `/res/layout`. En este caso le decimos a través de una constante entera llamada `activity_main` qué vista queremos que se visualice.

Por cada archivo de Layout que tengamos en `/res/layout` se generará una entrada en la clase `R.java`.

En la siguiente imagen puedes ver el aspecto del código en el IDE.



```
activity_main.xml x MainActivity.java x
1 package com.example.helloworld;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14
15 }
```

Captura del IDE Android Studio cuya licencia es Apache 2.0

Lo primero que nos interesa es ver que esta clase extiende la superclase `AppCompatActivity`, por lo que será una **actividad o pantalla**. De los métodos que se observan, nos fijamos en cómo se sobrescribe el método `onCreate()`, encargado de iniciar la aplicación. Concretamente en la segunda línea del método vemos cómo se carga el diseño (`layout`) creado en el fichero XML anterior (`activity_main`).

Autoevaluación

En el caso de tener un recurso Layout llamado `second_activity` indica cómo se accedería a él en código Java:

- R.layout.second_activity
- R.second_activity
- layout.second_activity
- XML.second_activity

¡¡Correcto, has acertado!!

No es la opción correcta.

Incorrecto.

No es correcto.

Solución

1. Opción correcta
2. Incorrecto
3. Incorrecto
4. Incorrecto

5.4.- El fichero de manifiesto

Sabemos cómo diseñar una pantalla utilizando XML y cómo podemos mostrarla después mediante código Java, pero ¿qué hacemos para que comience una aplicación Android? ¿Cómo sabe nuestra aplicación de qué componentes está formada y cómo interactúan entre ellos? La respuesta está en el **fichero de manifiesto**.

Podemos decir que el **fichero de manifiesto es uno de los más importantes de la aplicación**, pues registra muchas de las características para su correcto funcionamiento.

Si estudiamos su código vemos lo siguiente:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```



El nodo principal manifest, destaca entre otros atributos xmlns:Android donde referencia al espacio de nombres de Android y siempre tendrá ese valor y package con el nombre del paquete de la aplicación.

En el nodo application se especificarán todos los componentes de la aplicación así como las características de configuración. Podemos encontrarnos atributos como android:icon o android:label donde se especifican mediante referencias la imagen que se utilizará como ícono para la aplicación, así como el nombre de la misma.

Dentro del nodo application nos encontraremos el nodo activity o pantalla, componente principal de cualquier aplicación Android. Uno de sus atributos es android:name que precisamente referencia al nombre de la clase principal que carga la actividad.

La etiqueta intent-filter se verá más adelante, ahora simplemente podemos decir que nos serviría para indicar que **esta actividad es principal y ejecutable**. Una aplicación suele contener más de una pantalla o activity, por tanto cada una de ellas tendrá que registrarse en el fichero de manifiesto para que se pueda ejecutar.

Autoevaluación

El fichero de manifiesto se encarga de definir el diseño de la aplicación.

Verdadero Falso

Falso

Es falso. El fichero que define el diseño del layout está en la carpeta /res/layout/.

El método onCreate() lo encontramos en el fichero de manifiesto.

Verdadero Falso

Falso

Es falso. Ese método se encarga de iniciar la aplicación y está dentro del fichero que se encarga de mostrar la pantalla principal (en nuestro caso MainActivity.java dentro de la carpeta /src/)

6.- Componentes de una aplicación Android

Caso práctico

Llegados a este punto, Ada explica a Juan y María cuáles son los componentes principales con los que van a trabajar a partir de ahora. Deben comprender para qué se utiliza cada uno de ellos y cómo funcionan. Ada les explica con ejemplos claros que pueden ver en sus propios móviles, ya que utilizan dicha plataforma. Por ejemplo, el reloj que les aparece en la pantalla principal es un tipo de componente que viene por defecto, cuando pulsamos sobre un ícono para ejecutar una aplicación:

- ✓ ¿Qué mecanismo se lleva a cabo?
- ✓ ¿Qué nombre reciben cada una de las ventanas o pantallas que se abren?
- ✓ ¿Cómo se organizan los elementos que hay en ellas?
- ✓ ¿Cómo se puede compartir datos entre varias aplicaciones?



Es importante que conozcan e identifiquen cada uno de estos componentes, porque son la base de cualquier aplicación.

Si has programado en Java, estarás acostumbrado a trabajar con ventanas, controles, eventos, servicios, etc. a la hora de desarrollar una aplicación. En Android disponemos también de estos componentes aunque con algunas variaciones que pasaremos a comentar.



RafaB.

En una aplicación Android podemos encontrar los siguientes componentes:

- ✓ **Activity (Actividad):** es el componente principal de la interfaz gráfica de una aplicación Android. Es similar a una ventana o pantalla en cualquier lenguaje visual.
- ✓ **Service (Servicio):** es el componente que se ejecuta en segundo plano. Es similar a los servicios de cualquier sistema operativo. Estos servicios se encargan, por ejemplo, de enviar notificaciones, mostrar elementos visuales (activity), actualizar datos, etc.
- ✓ **Content Provider (Proveedor de contenidos):** es el componente que se encarga de compartir datos entre distintas aplicaciones Android.
- ✓ **Broadcast Receiver (Receptor de anuncios):** es el componente que se encarga de detectar y reaccionar ante algunos mensajes o eventos generados por el sistema (como por ejemplo batería baja, tarjeta SD insertada, SMS recibido, etc.) o por otras aplicaciones (cuando generan mensajes que no van dirigidos a una aplicación concreta).

Estos componentes se deben definir siempre en el fichero manifiesto dentro de la etiqueta <application>. Por cada tipo de componente de nuestra aplicación se debe declarar uno de los siguientes elementos XML en función del tipo:

- ✓ <activity> para cada subclase de **Activity**.
- ✓ <service> para cada subclase de **Service**.
- ✓ <receiver> para cada subclase de **BroadcastReceiver**.
- ✓ <provider> para cada subclase de **ContentProvider**.

También podemos encontrarnos otros elementos que tienen una funcionalidad específica dentro de nuestra aplicación:

- ✓ **Intent (Intención):** es el objeto encargado de la comunicación entre el resto de componentes. Esta comunicación se realiza mediante mensajes o peticiones. Como ejemplo un intent nos permite abrir una actividad desde otra inicial, enviar un mensaje broadcast, iniciar un servicio, iniciar otra aplicación, etc.
- ✓ **Widget:** es un elemento visual que suele mostrarse en la pantalla principal del dispositivo. Muestra información de la aplicación al usuario. Pero en el ámbito de la programación de Android esta palabra adquiere otro significado, ya que también puede hacer referencia a un objeto **View** en el diseño de una interfaz de usuario.
- ✓ **View (Vista):** es el componente básico con el que se construye la interfaz gráfica de la aplicación, será algo similar a los controles de Java. Entre alguno de los controles básicos o views que tiene Android estarán los cuadros de texto, botones, listas, etc.
- ✓ **Layout (Diseño):** se utiliza para agrupar y organizar los diferentes componentes de tipo view que componen una actividad o pantalla (activity). Existen diferentes tipos y además un layout también puede contener a su vez otros layouts.

Autoevaluación

El componente básico utilizado para construir la interfaz gráfica de una aplicación Android es...

- Service.
- Intent.
- View.
- Activity.

Incorrecta.

No es correcta, inténtalo de nuevo.

¡Has acertado! En efecto, ya que son los controles a partir de los cuales se construye cualquier aplicación.

Fallaste, ésta no es la opción correcta.

Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta
4. Incorrecto

Caso práctico



Juan ha visto cómo María ha creado la primera aplicación, el proceso no es muy complejo pero, ¿cómo se pueden ver los resultados? Es decir, ¿cómo pueden comprobar lo que realmente hace esa aplicación? ¿Dónde la pueden ejecutar? ¿Existen tipos de emuladores para ver el resultado de diferentes formas? ¿Se puede ejecutar la aplicación en sus propios móviles?

Normalmente las herramientas de desarrollo integradas suelen incorporar uno o varios emuladores que te servirán para poder probar las aplicaciones antes de instalarlas en el dispositivo para el cual se han programado. Así podrás observar si la aplicación funciona correctamente y si tiene más o menos el aspecto que tenías pensado.

Hay que tener en cuenta que los emuladores no siempre van coincidir en aspecto (y a veces incluso en funcionalidad) con el dispositivo real y podrás encontrar algunas diferencias. En Android Studio esta herramienta se llama **AVD Manager**, y es el gestor de dispositivos virtuales o emuladores. Esta herramienta viene integrada dentro de Android Studio con lo cual sólo se tendrá que crear el emulador.

Es importante **probar nuestra aplicación en diferentes emuladores con diferentes niveles de API de Android** y así garantizar que nuestra aplicación se ejecuta correctamente. Ten en cuenta que existen muchas marcas y modelos de dispositivos Android y no tenemos a nuestro alcance tantos dispositivos físicos. Será a través de un emulador que probemos casi todas las funciones de un dispositivo Android real.

La mayor parte del proceso de depuración (por no decir todo) se realizará normalmente con el emulador, pues podremos depurar el software como con un programa en un ordenador convencional (ejecutar las instrucciones paso a paso, establecer breakpoints, observar el contenido de variables, etc).

Finalmente podremos probarlas en un dispositivo real.



Captura del IDE Android Studio cuya licencia es Apache 2.0

7.1.- Desplegando una aplicación en el emulador

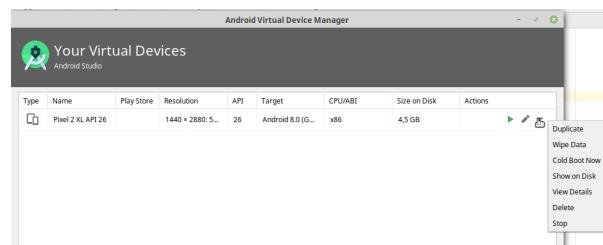
Lo primero que tenemos que hacer es crear un emulador mediante la herramienta AVD Manager seleccionando la opción **Tools -> AVD Manager** o a través del ícono de la barra de herramientas:



Captura del IDE Android Studio cuya licencia es Apache 2.0

Sólo tienes que seguir los pasos del asistente para crear un emulador, si tuvieras alguna duda puedes consultar en enlace de la documentación oficial [Cómo crear y administrar dispositivos virtuales](#).

Una vez se ha creado el emulador puedes ejecutarlo mediante el ícono ejecutar (▶), editar su configuración mediante el ícono de edición (⚡), y a través del menú que se despliega con el ícono desplegar menú (▾) duplicar la configuración del emulador mediante **Duplicate** y volver a la configuración inicial mediante la opción **Wipe Data**.



Captura del IDE Android Studio cuya licencia es Apache 2.0

Para ejecutar nuestra aplicación en el emulador tenemos en la barra de herramientas la configuración de depuración/ejecución de nuestro módulo **app** y a la derecha está la herramienta donde puedes seleccionar el dispositivo en el que deseas ejecutar la aplicación:

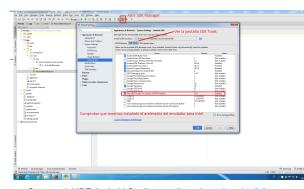


Captura del IDE Android Studio cuya licencia es Apache 2.0

A continuación, haz clic en **Run** (Ejecutar, ▶) y se instalará la aplicación en el dispositivo seleccionado.

Debes conocer

Es cierto que estos emuladores son muy lentos, aunque si se dispone de un procesador Intel, la instalación por defecto te habrá tenido que instalar un acelerador de hardware. Lo puedes comprobar consultando el SDK Manager:



Captura del IDE Android Studio cuya licencia es Apache 2.0

Recomendación

Una opción muy extendida entre los desarrolladores de aplicaciones Android para emular sus aplicaciones es utilizar cualquier emulador que ofrece **Genymotion**. Esta empresa proporciona emuladores más potentes y rápidos que los que podemos encontrar en Android Studio. Para utilizarlos, debemos instalar primero una máquina virtual con **VirtualBox**. En los siguientes enlaces puedes encontrar todo el proceso para trabajar con estos emuladores. Sería conveniente que lo instales y lo configures.

[Descargar VirtualBox](#)

[Página oficial de Genymotion](#)

Configurar Genymotion con Android Studio en Linux

<http://www.youtube.com/embed/s1LohXtaZAc>



[Resumen textual alternativo](#)

Configurar Genymotion en Windows

<http://www.youtube.com/embed/RnK7zLoOkMc>

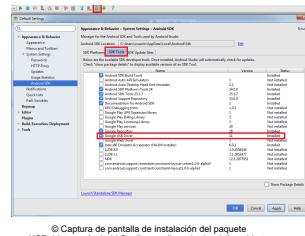


[Resumen textual alternativo](#)

7.2.- Desplegando una aplicación en un dispositivo real

Todas las pruebas que podamos hacer con los distintos emuladores nos facilitan mucho el trabajo en el desarrollo de nuestras aplicaciones, pero, ¿de qué sirven si finalmente no logramos que funcionen igual en un dispositivo real? ¿Puede variar algo la funcionalidad o aspecto de nuestra aplicación si la ejecutamos en un smartphone, tablet o cualquier otro dispositivo Android? Por supuesto que puede haber diferencias, por tanto se hace imprescindible probar también las aplicaciones en los dispositivos móviles reales para comprobar que todo funciona correctamente. Para eso, será necesario seguir los siguientes pasos:

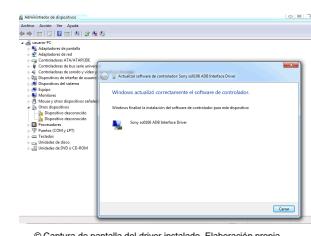
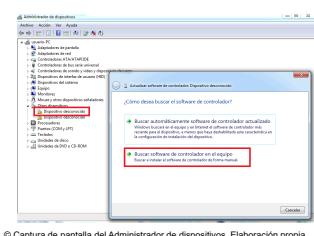
1. Debemos instalar el paquete **Google USB Driver** del **SDK Manager**. Hay que tener en cuenta que este controlador es genérico y puede ser que necesites descargar el controlador o driver específico del fabricante de tu dispositivo.



2. Activar el modo **Depuración de USB** en nuestro dispositivo desde **Ajustes – Opciones del desarrollador** y activamos la **Depuración USB**.

Si no encuentras las opciones del desarrollador en el móvil realiza los siguientes pasos del siguiente enlace [Cómo configurar las opciones para desarrolladores en el dispositivo](#)

3. Si al conectar el dispositivo móvil al ordenador solicita instalar el driver del fabricante, consulta la página web oficial de tu dispositivo y descarga los drivers **USB** específicos para tu terminal. Puedes encontrarlos en la página de **OEM** en la documentación oficial de Android. Pulsa [en este enlace](#) si necesitas descargar los drivers oficiales de tu dispositivo.
4. Ahora debemos instalar correctamente el driver en el dispositivo y para ello nos vamos a **Administrador de dispositivos** en el **Panel de control**.
5. Hacemos clic derecho del ratón sobre el dispositivo que tengamos en **Otros dispositivos** y actualizamos el driver que nos hemos descargado anteriormente.



6. Una vez instalado y reconocido el dispositivo real ya podremos lanzar nuestra aplicación Android en dicho dispositivo puesto que aparecerá en la ventana donde tenemos creados los emuladores anteriores.
7. Seleccionamos el dispositivo real entre todos los dispositivos que podemos lanzar y la aplicación se ejecutará en nuestro dispositivo conectado al ordenador.

Para saber más

Si deseas más información sobre cómo lanzar una aplicación en un dispositivo real, incluso si eres usuario de otro sistema operativo distinto de Windows, puedes consultar la página oficial de Android.

[Conectar un dispositivo real](#)

8.- Ciclo de vida de una actividad

Caso práctico



Juan y María han terminado la primera aplicación Android y la han desplegado tanto en distintos emuladores como en algunos dispositivos móviles reales. Antes de continuar viendo el diseño de la interfaz de usuario para realizar aplicaciones más complejas deciden preguntar a Ada cuál es el funcionamiento interno de la aplicación que acaban de realizar. Ada les explica que como desarrolladores tendrán que conocer los métodos que gestionan todo el proceso de una actividad o pantalla.

Ya está todo listo para poder empezar a crear aplicaciones algo más complejas y poderlas instalar y probar tanto en los emuladores como en dispositivos reales. Pero antes de comenzar con esa tarea, vamos a ver un aspecto muy importante a tener en cuenta en las aplicaciones **Android**, hablamos del ciclo de vida de las actividades o pantallas.

Nuestra aplicación constará de varias Activity. La primera que se ejecuta suele llamarse **actividad principal** y ésta invocará a otra actividad de nuestra aplicación mientras el usuario interacciona con nuestra aplicación. Este cambio de estado de las actividades es debido a que el sistema operativo llama de forma automática a los métodos del ciclo de vida de las actividades y estos devuelven el resultado al sistema, a este tipo de llamadas se les conoce como métodos **callback**.

Esto no quiere decir que las actividades que dejamos de ver se destruyan, sino que se guardan en una **pila de actividades** en el mismo orden en el que han sido abiertas. Es cuando el usuario pulsa el botón back o **Atrás**, cuando la actividad finaliza y se quita de la pila de actividades. Cuando finalmente la actividad principal de nuestra aplicación finalice, se eliminará toda la información de la aplicación.

Las actividades de la pila nunca se reordenan, sólo se insertan o se quitan

También una actividad se elimina cuando hay un **cambio de configuración** del dispositivo, el ejemplo más común es el cambio de idioma. En este supuesto todo lo que tenga interfaz de usuario deberá actualizarse para que coincida con esa configuración de ahí que la actividad se elimine llamando a sus métodos onPause(), onStop() y onDestroy() y se creará una nueva instancia de la actividad mediante las llamadas a onCreate(), onStart() y onResume().

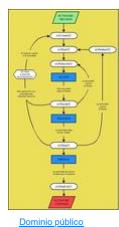
Debemos ser conscientes que la capacidad de memoria de los dispositivos es limitada y por tanto tener muchas aplicaciones en ejecución puede ser un problema. En estos casos, el sistema para liberar memoria, puede destruir actividades que estén detenidas.

Los distintos estados por los que puede pasar una actividad son:

- ✓ **Activa:** la actividad está visible y puede ser utilizada por el usuario.
- ✓ **Pausada:** la actividad está visible pero ha perdido el foco por otra actividad (por ejemplo: ventanas de diálogo o avisos). Se puede ver la actividad anterior porque la nueva actividad no suele ocupar la pantalla completa o es semitransparente.
- ✓ **Parada:** la actividad deja de estar visible, es conveniente guardar el estado por si el usuario vuelve a esta actividad.

Los métodos que gestionan el ciclo de vida son los siguientes:

- ✓ **onCreate():** la actividad es creada por primera vez. Aquí tiene lugar la inicialización de la interfaz gráfica así como los datos. Es obligatorio sobrescribir este método en una actividad. Este método recibe como parámetro una instancia de la clase **Bundle** que contiene la información necesaria para restaurar la actividad de su estado anterior. Si es la primera vez que se ejecuta la actividad, el valor de este parámetro será nulo.
- ✓ **onStart():** la actividad está preparada para ser visualizada. No implica que se visualice al usuario.
- ✓ **onResume():** la actividad se encuentra en la parte superior de la pila del sistema y está visible al usuario. Se puede empezar a interactuar con la actividad.
- ✓ **onPause():** la actividad va a pasar a segundo plano porque otra actividad toma el foco. Es fundamental guardar el estado de la actividad.
- ✓ **onStop():** la actividad va a dejar de estar visible por completo. Si el sistema no tiene memoria suficiente puede ser que la actividad sea destruida sin pasar por este método.
- ✓ **onRestart():** la actividad volverá a visualizarse después de haber sido parada y sin haber llegado a destruirse.
- ✓ **onDestroy():** la actividad se destruye completamente para liberar la memoria utilizada. Si el sistema no tiene memoria suficiente puede ser que la actividad sea destruida sin pasar por este método.



Los métodos que abarcan todo el ciclo de vida son **onCreate()** y **onDestroy()**, inicio y fin de la actividad. La actividad será visible al usuario desde el método **onStart()** hasta el método **onStop()**, aunque puede ser que no tenga el foco porque hay otras actividades con las que esté interactuando el usuario. La actividad está visible y además tiene el foco desde **onResume()** hasta **onPause()**.

Para saber más

Puedes consultar la documentación oficial para conocer mejor el ciclo de vida de una actividad y cómo funciona la pila de actividades:

[Cómo interpretar el ciclo de vida de una actividad](#)

[Funcionamiento de la pila de actividades](#)

Anexo.- Licencias de recursos

Licencias de recursos de la unidad

Miniatura del recurso	Credenciales
 Montaje de elaboración propia usando recursos externos.	Autoría: Elaboración propia. Licencia: Uso Educativo no comercial, citando fuentes. Procedencia: Montaje de elaboración propia a partir de las imágenes siguientes: <ul style="list-style-type: none">✓ Imagen 1: Android<ul style="list-style-type: none">❖ Autoría: astanush❖ Licencia: CC BY-NC-SA.❖ Procedencia: https://www.flickr.com/✓ Imagen 2: iOS<ul style="list-style-type: none">❖ Autoría: Mike Vasilev (cliquunited)❖ Licencia: CC BY-SA✓ Imagen 3: Windows Phone.<ul style="list-style-type: none">❖ Autoría: Michele Ficara Manganelli❖ Licencia: CC BY-NC-SA.✓ Imagen 4: Java ME<ul style="list-style-type: none">❖ Autoría: Oracle.❖ Licencia: Copyright

Nota: Este anexo incluye exclusivamente recursos cuya licencia obliga a la cita expresa, y que por la extensión de sus credenciales o por cualquier cuestión técnica no se pueden citar en línea. El resto de recursos son de elaboración propia y no requieren cita expresa, quedando su licencia expresada en la de los propios materiales, o se citan en línea, en el lugar donde aparecen en contenidos.

Condiciones y términos de uso de los materiales

Materiales desarrollados inicialmente por el Ministerio de Educación, Cultura y Deporte y actualizados por el profesorado de la Junta de Andalucía bajo licencia Creative Commons BY-NC-SA.



Antes de cualquier uso leer detenidamente el siguiente [Aviso legal](#)

Historial de actualizaciones

Versión: 03.00.00	Fecha de actualización: 01/06/20	Autoría: Lourdes Rodríguez Morón
<p>Ubicación: 1.5.1 Mejora (tipo 3): La clase surfaceview muy importante y de complejidad no se explica adecuadamente. Esta clase es la recomendada por Android para la programación multimedia. Ubicación: En la sección 1.1. (recuadro Para saber más) Mejora (tipo 1): Hay dos enlaces que no funcionan que son los indicados de la siguiente forma: - Puedes ver en funcionamiento de uno de los primeros videojuegos documentados de la historia siguiendo este enlace. - En el siguiente enlace podrás comprobar cómo era el primer videojuego multijugador: Tenis para dos. Ubicación: 2.8 Mejora (tipo 2): El componente TabHost está obsoleto (legacy) Ubicación: 7.1, 7.2 Mejora (tipo 2): Añadir diseño de pantalla con la vista BluePrint, no sólo con la vista xml. Uso de la interfaz gráfica para el diseño de pantallas. Ubicación: 4. Mejora (tipo 1): Actualizar a la nueva versión de Android Studio Ubicación: 2.4, 2.5 Mejora (tipo 1): Actualizar: No referenciar j2me. Actualizar: completar la tabla hasta android 9 Ubicación: 2.3 Mejora (tipo 1): El IDE más utilizado ya no es Eclipse, es Android Studio. Es al que Android (Google) da soporte. Ubicación: 2.5 Mejora (tipo 1): La tabla de versiones de Android hay que completarla hasta la última versión Android 9.0 Pie Ubicación: 2.5 Mejora (tipo 1): La actual máquina virtual de Android a cambiado a ART. Ubicación: Nodos del mapa Mejora (Mapa conceptual): Actualización en base a los nuevos contenidos. Ubicación: Tabla de contenidos Mejora (Orientaciones del alumnado): Actualización en base a los nuevos contenidos</p>		
<p>Versión: 02.02.00</p>		
<p>Fecha de actualización: 05/07/16</p>		
<p>Autoría: Víctor Gil Rodríguez</p>		
<p>Ubicación: No especificada. Mejora (tipo 2): Adaptar los contenidos a Android Studio en todas las unidades Ubicación: Punto 2. Parrafo 2º Mejora (tipo 1): Dice: Samsung (el mayor proveedor en 2014), Apple, Huawei, Sony, ZTE, HTC, Blackberry), LG, Motorola, etc. Debe decir: Samsung (el mayor proveedor en 2014), Apple, Huawei, Sony, ZTE, HTC, Blackberry, LG, Motorola, etc. Ubicación: 3.1.- Funcionalidades de Android. Mejora (tipo 1): El formato de fuente es distinto Ubicación: 3.1.- Funcionalidades de Android. Mejora (tipo 1): En el segundo parrafo dice: Recepción de mensajes tipo Push debe decir: Recepción de mensajes tipo Push Ubicación: No aparece mapa conceptual correcto. Mejora (Mapa conceptual): Aparece el mapa conceptual del tema 6, cuando debería de aparecer el del tema 4</p>		
<p>Versión: 02.01.00</p>		
<p>Fecha de actualización: 05/01/15</p>		
<p>Autoría: Antonio José López Jiménez</p>		
<p>Actualización de datos generales y características con las tendencias actuales. eliminación de detalles obsoletos (algunos marcados en desuso). Inclusión de pequeña introducción a Android Studio.</p>		
<p>Versión: 02.00.00</p>		
<p>Fecha de actualización: 25/04/14</p>		
<p>Autoría: Víctor Gil Rodríguez</p>		
<p>Nuevos contenidos actualizados con Android.</p>		
<p>Versión: 01.00.00</p>		
<p>Fecha de actualización: 25/04/14</p>		
<p>Versión inicial de los materiales.</p>		

