Análisis de tecnologías para aplicaciones en dispositivos móviles.

# 1.- Primeros conceptos.

La inmediatez, hiperconectividad y movilidad son aspectos claves para que ya, en la actualidad, la tecnología smartphone y de dispositivo móvil tenga más usuarios en el mundo que las propias tecnologías de PC.

En los próximos apartados, cubriremos de forma general los aspectos más importantes de estas nuevas tecnologías.

## 1.1.- Introducción. ¿Qué es un dispositivo móvil?

Es un aparato de pequeño tamaño (normalmente que quepa en un bolsillo) y de poco peso, con pantalla y teclado, con pequeñas capacidades de procesamiento, memoria limitada y conexión (permanente o no) a una red. Este tipo de dispositivos están diseñados para realizar llamadas telefónicas, servir como agenda, jugar, navegación GPS, escuchar música, acceso al correo electrónico, navegar por Internet, proporcionar servicio de chat con otros dispositivos móviles, etc. Aunque normalmente, pueden llevar a cabo también funciones más generales.

**Para saber más**: [características de un dispositivo móvil](https://es.wikipedia.org/wiki/Dispositivo_m%C3%B3vil)

Clasificación de los dispositivos móviles

* **Smartphones**
* PDA (Personal Digital Assistant – asistentes digitales personales) o PocketPC (PC de bolsillo).
* Handheld, o PCs de mano.
* Internet tables, que se encontrarían entre las PDA y los PC Ultramóviles (pequeños Tablet PC).

Según las fuentes que consultes, puedes encontrar diversas clasificaciones donde se incluyan unos u otros tipos de dispositivos, como:

* Pagers (o buscapersonas), hoy día ya en desuso.
* Navegadores GPS.
* E-Readers (lectores de libros digitales).
* Pequeñas videoconsolas de mano.
* Cámaras digitales.
* Calculadoras programables.

En nuestro caso, **los principales aparatos a los que nos referiremos al hablar de dispositivos móviles serán los smartPhones y las tablets**. 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.

## 1.2.- Limitaciones de las tecnologías móviles.

Algunas de las restricciones de estos dispositivos son:

* **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 RAM**.
* Almacenamiento de datos persistente reducido (pequeña memoria flash interna, tarjetas SD, etc.).
* Pantallas de reducidas dimensiones.
* Teclados con funcionalidad muy básica y muy pequeños.

Este tipo de restricciones, y algunas otras que dependerán de cada dispositivo en concreto, deben ser tenidas en cuenta a la hora de analizar y diseñar una aplicación **“móvil”**.

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 (mensajería SMS y MMS; voz; Internet; [Bluetooth](http://localhost/Proyectos/2DAM/Programaci%c3%b3n%20multimedia%20y%20dispositivos%20m%c3%b3viles/Evaluacion%201/Tema%201/Apuntes/PMDM01/WEB/HTML/12_LIMIT.HTM#ta1ac036b-35d8-c5a8-ba86-99cfb5a153c4); infrarrojos; radiofrecuencia, etc.). Ésas serán las ventajas que podrás explotar en tus aplicaciones.

**Para saber más**: sobre las características y limitaciones de los dispositivos móviles. [(1548) Limitaciones de los dispositivos móviles - YouTube](https://www.youtube.com/watch?v=Fm5GsDUScCA)

## 1.3.- Tecnologías disponibles.

Hay que conocer qué tipos de tecnologías se pueden encontrar en este ámbito, 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, APIs, lenguajes de programación, etc.).

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

* **Hardware**: móviles y tabletas de las principales marcas. **Samsung**, **Apple**, **Huawei**, **HTC**, **LG**, **Motorola**, **Sony Ericsson**, **Nokia**, **Alcatel-Lucent**, **Xiaomi**, Oppo y Vivo.
* **S.O**: basados en Windows, Linux, Mac OS, Android, iOS, Symbian OS, Blackberry OS y Windows Phone.
* **Plataformas de desarrollo disponibles para cada entorno**: Java ME, Windows Mobile SDK, Maemo SDK, Xamarin o bien de IDEs como Microsoft Visual Studio, CodeWarrior, Eclipse, Netbeans o **Android Studio**.
* **Lenguajes de programación**: Java, C#, C, Kotlin, 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.

**Para saber más**: [Un trocito de historia: evolución y "muerte" de las PDA (muycomputer.com)](https://www.muycomputer.com/2014/07/06/un-trocito-de-historia-evolucion-y-muerte-de-las-pda/)

[Xamarin - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/Xamarin)

### 1.3.1.- Hardware.

Dependiendo de los criterios que se utilicen para clasificar los dispositivos móviles, se puede hablar de más o menos tipos. Este curso se va a centrar sobre todo en smartphones y tablets.

Smartphone

Un smartphone es un terminal de telefonía móvil que proporciona unas prestaciones y una funcionalidad mayor que la que podría ofrecer un teléfono móvil normal.

Algunas otras **características** que suelen tener este tipo de dispositivos son:

* Funcionamiento en multitarea (ejecución concurrente de varios procesos en el sistema operativo).
* Acceso a Internet.
* Conectividad Wi-Fi, Bluetooth, etc.
* Posibilidad de ampliación de memoria mediante tarjeta SD.
* Pequeñas pantallas pero de alta resolución y/o con millones de colores.
* Pantallas multitáctiles (multitouch).
* GPS.
* Cámaras digitales integradas. Capacidades fotográficas. Grabación de audio y vídeo.
* Posibilidad de conexión con un ordenador para cargar y descargar información. Normalmente con conexión USB o bien una conexión inalámbrica.
* Sensores (de orientación, de temperatura, de presión, acelerómetros, magnetómetros, etc.).
* Receptor de radio FM.
* Emisor de radio FM.
* Posibilidad de instalar y ejecutar aplicaciones sofisticadas:
  + Aplicaciones de asistente personal (gestión de contactos, calendarios, citas, agendas, alarmas, etc.).
  + Gestión del correo electrónico.
  + Gestión del sistema archivos del dispositivo.
  + Microaplicaciones de ofimática (procesador de textos, hoja de cálculo, etc.).
  + Aplicaciones multimedia (reproducción de audio y vídeo en diversos formatos).
  + Aplicaciones de cartografía y navegación.
  + Diccionarios.
  + Pequeñas aplicaciones científicas (matemáticas, física, medicina, etc.).
  + Juegos.
  + Aplicaciones de mensajería instantánea. Chats.

La **principal ventaja de un smartphone** es que no se limita solo a hacer y recibir llamadas, sino que podrá: hacer fotos, navegar por Internet, reproducir archivos de audio, jugar, gestionar la agenda personal, consultar un mapa, usar un diccionario, escuchar la radio, ver una película, trazar una ruta para el navegador por satélite GPS, etc.).

# 2.- Sistemas operativos.

Los más habituales en un dispositivo móvil son

* **Android**: desarrollado inicialmente por Google y basado en el núcleo de Linux. El primer fabricante de móviles que lo incorporó fue HTC.
* **iOS**: desarrollado por Apple para el iPhone, y usado más tarde para el iPod Touch y el iPad.
* **Windows Phone**: anteriormente Windows Mobile, y posteriormente derivado en Windows 10 Mobile (descontinuado). Desarrollado por Microsoft tanto para smartphones como para otros dispositivos móviles (por ejemplo, PDA).
* **Blackberry OS (descontinuado)**: desarrollado por Research in Motion (RIM) para sus dispositivos Blackberry.

Según Global stat (statcounter.com), en diciembre de 2021, la cuota de mercado de S.O para móviles se encuentra distribuida de la siguiente forma a nivel mundial:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sistema Operativo** | Android | iOS | Samsung | KaiOS | Unknown | Nokia Unknown | Windows | Series 40 | Linux |
| **Cuota de Mercado** | **70.01%** | 29.24% | 0.43% | 0.13% | 0.12% | 0.002% | 0.01% | 0.01% | 0.01% |

Según esa misma fuente y para la misma fecha, la cuota de mercado de S.O para tabletas se encuentra distribuida de la siguiente forma a nivel mundial:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sistema Operativo** | iOS | Android | Windows | Linux |
| **Cuota de Mercado** | **55.66%** | 44.25% | 0.04% | 0.02% |

Si además trasladamos estas estadísticas a nivel nacional, se puede decir que en España, la cuota de mercado de Android en smartphones supera el 78% del mercado.

## 2.1.- Android.

**Android** fue inicialmente desarrollado por **Android Inc**., hoy día parte de la compañía **Google**. Está basado en una versión modificada del [kernel](http://localhost/Proyectos/2DAM/Programaci%c3%b3n%20multimedia%20y%20dispositivos%20m%c3%b3viles/Evaluacion%201/Tema%201/Apuntes/PMDM01/WEB/HTML/21_ANDRO.HTM" \l "td88628e3-0a8b-b7ce-63f9-a44b82fa42c7" \o "Kernel | ) de **Linux**.

El primer fabricante que incorporó **Android**en sus dispositivos fue **HTC**con su terminal **HTC Dream** (comercializado también como **T-Mobile G1** y popularmente conocido con los nombres de **Google Phone** o **GPhone**) en 2008.

Es uno de los S.O más jóvenes dentro del grupo de S.O para dispositivos móviles, y se ha hecho un notable hueco alcanzando más 2/3 de la cuota de mercado actual. Existen miles de aplicaciones que funcionan sobre **Android**, con un crecimiento cada vez mayor.

A diciembre de 2021, la última versión de Android es 12.0. Las versiones Android tenían un simpático criterio de asignación de nombres, donde cada versión se correspondía con un postre, y que avanzaban según avanzaba alfabéticamente la letra del postre.

Para desarrollar aplicaciones sobre **Android**, es necesario el **kit de desarrollo de software para Android (Android SDK)**, proporcionado gratuitamente por **Google**. Este paquete incluye: depurador, bibliotecas, emulador, documentación, etc.).

**El lenguaje de programación** es Java (y por tanto, es necesario el JDK de Oracle para poder compilar programas Java.

**El IDE oficial**: durante años, fue Eclipse (a partir de la versión 3.2), junto con el plugin **ADT** (**Android Development Tools**). Sin embargo, las desavenencias entre **Oracle** (propietaria de Java) y **Google** (propietaria de **Android**) en los últimos años, han hecho que **Android** deje de recomendar Eclipse, y haya desarrollado su propio IDE partiendo del IDE IntelliJ IDEA, convirtiéndose, a partir de finales del año 2013, en el nuevo IDE oficial para el desarrollo de aplicaciones para Android.

**Para saber más**: [Android - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/Android)

## 2.2.- iOS.

Sistema operativo desarrollado por **Apple** originalmente para su **iPhone**, aunque hoy día también es utilizado por otros dispositivos de la empresa.

Es un sistema operativo derivado del **Mac OS X**, también de **Apple**.

En enero de 2022, según **www.apple.com**, la tienda de aplicaciones de **Apple** (servicio de descarga de aplicaciones), llamada **App Store**, dispone ya de **1,8 millones de aplicaciones** para dispositivos con iOS. En la misma fecha, la cuota de mercado de Apple en los dispositivos móviles es del 29,21% según [gs.statcounter.com](https://gs.statcounter.com/vendor-market-share/mobile)

La última versión de **iOS** en octubre de 2021 es la versión de 15.0.2.

En 2015, **Apple** lanza su **Apple Watch**. Un reloj inteligente (**smartwatch**) que requiere un **iPhone** asociado para funcionar. Usa el sistema operativo **watchOS**basado en **iOS**.

Cosas a tener en cuenta para desarrollar aplicaciones sobre **iOS:**

* iPhone SDK. Para poder usarlo, es necesario contar con un ordenador Mac, ya que se requiere un S.O MAC OS X Leopard o superior.
* Lenguaje de programación principal: Objective-C.

A mediados de 2011, ni el entorno .NET ni **Adobe Flash** eran soportados por **iOS**, aunque **Adobe Flash** es utilizable a través de aplicaciones de terceros.

Lo mismo sucede con **Java**, que aunque se han producido algunos intentos de máquinas virtuales de Java (basadas en Java ME) para iOS, rozan la legalidad a la que están sometidas las restrictivas licencias de Apple. Pero siempre habrá comunidades de usuarios y desarrolladores que intentarán saltarse estas restricciones (para más información al respecto puedes consultar el término **iOS Jailbreaking** en la web).

La descarga de iOS SDK es gratuita, pero requiere registrarse en el [Apple Developer Program - Apple Developer](https://developer.apple.com/programs/) para poder publicar el software creado, lo cual sí requiere aprobación de Apple y un pago por ese servicio, que propoprcionará al programador las claves firmadas que le permitirán publicar en la App Store.

**Para saber más**: [iOS - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/IOS)

## 2.3.- Windows Phone.

Sustituye a su antecesor **Windows Mobile**. Las primeras versiones de sistemas operativos para dispositivos móviles desarrolladas por **Microsoft** eran conocidas como **Windows CE** (**Compact Embeded**), la cual, hoy en día, sigue siendo una versión para sistemas empotrados como PDA y otros pequeños dispositivos.

Los primeros **Windows CE** podían encontrarse en las PDA iPAQ de Compaq o las de Palm (Palm Pilot). Las iPAQ con Windows CE ofrecían un entorno más parecido al **Windows** de los ordenadores de escritorio frente al aspecto más sobrio de la interfaz de usuario de **Palm OS**.

Microsoft anunció en enero de 2015 que daría de baja a Windows Phone, para enfocarse en un único sistema más versátil llamado **Windows 10 Mobile**. Actualmente es un sistema operativo descontinuado.

La última versión de **Windows Phone** fue la 8.1 (lanzada en marzo de 2015), desarrollada para smartphones y PDA. Algunas de las versiones anteriores han sido: **Windows Phone 8, Windows Phone 7.x, Windows Mobile 6.x, Windows Mobile 5.x, Windows Mobile 2003 o Pocket PC 2002**. Todas ellas basadas en la plataforma **Windows CE**.

Entre los fabricantes que incluían Windows Phone en sus dispositivos se encuentran **HTC**(series S y T), **Samsung**(serie Omnia), **LG**(Optimus 7 y Pacific) y **Sony Ericsson** (Xperia X7) y Nokia.

Para desarrollar aplicaciones sobre **Windows Phone** pueden utilizarse las tecnologías **Silverlight**y **XNA**de **Microsoft**, así como el entorno de desarrollo **Visual Studio**.

**Para saber más**: [Windows Phone - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/Windows_Phone), [Windows Phone - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/Windows_Phone)

# 3.- Plataformas de desarrollo y lenguajes de programación.

Necesitaremos disponer de alguna **plataforma de desarrollo** que nos permita generar código ejecutable sobre esos sistemas operativos para móvil, o sobre alguna máquina virtual o intérprete que esté instalado en el dispositivo y que sea soportado por el S.O.

**Dependiendo de la versatilidad** de la plataforma de desarrollo, **la aplicación podrá ser, más o menos portable a otros sistemas** operativos y/o dispositivos. **Por ejemplo**: si realizamos programas en **Objective-C** utilizando el **SDK** de **Apple**, lo más probable es que nuestra aplicación solo pueda ser ejecutada en un dispositivo **Apple** sobre el que esté implantado **iOS**. En cambio, si desarrollamos un programa en **Java** usando el **Java ME** de **Oracle** y generamos una aplicación de tipo MIDlet (programa desarrollado en Java para dispositivos embebidos con máquina virtual Java Micro Edition, Java ME).

Hablaremos de las plataformas de desarrollo para móviles que los propios autores de los S.O ofrecen para trabajar sobre su plataforma:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.O target** | **Plataforma y lenguaje** | **IDE** | **Plugins** |
| **Android** | Android SDK (Java, C# o Kotlin) | Eclipse (antiguamente) | ADT (Android Development Tools) |
| Android Studio |  |
| IntelliJ IDEA |  |
| Android NDK (Native Development Kit): permite generar Código native, frente al Código gestionado, que se ejecuta sobre una máquina virtual como es el caso de Java o C# | | |
| **iOS** | iPhone SDK (gratis también), pero para poder comercializar el sofware a través de su tienda de aplicaciones, hay que registrarse en el programa de desarrollo del iPhone (no es gratis). Lenguaje: Objective-C | | |
| **Java ME** | Es un subconjunto de la plataforma **Java** orientada al desarrollo en móviles. Lenguaje: Java. Sería necesario instalar la máquina virtual en el S.O sobre el que se desea ejecutar la aplicación, como por ejemplo en Symbian (que se puede programar en modo nativo, o bien con **Java ME**). | | |
| **Windows Phone (antiguo)** | Silverlight, XNA y .NET Compact Framework de Microsoft.  Lenguaje: C#, Visual Basic .NET | Visual Studio 2010 Express y Expression Blend. |  |

**Para saber más**: sobre Kotlin: [Kotlin ya es un lenguaje oficial en Android: ¿qué implicaciones tiene y por qué es tan importante? (xatakandroid.com)](https://www.xatakandroid.com/programacion-android/kotlin-ya-es-un-lenguaje-oficial-en-android-que-implicaciones-tiene-y-por-que-es-tan-importante), [¿ Por qué Kotlin ? ¿Por qué Google elige Kotlin? | Apiumhub](https://apiumhub.com/es/tech-blog-barcelona/por-que-kotlin/), [Kotlin, ¿otra moda más? (Spoiler: no) - Paradigma (paradigmadigital.com)](https://www.paradigmadigital.com/dev/kotlin-otra-moda-mas-spoiler-no/)

## 3.1.- Elección de una alternativa.

Hemos de ser siempre conscientes de que en una empresa, es probable que quizá tengas que trabajar con otros lenguajes (por ejemplo **C#**, **C++**, **Python**, **Objective-C**, **Kotlin**,...) y para otras plataformas. Por ello, no hay que tomar una alternativa como la única válida.

Lo importante es adquirir unos conocimientos generales sobre este nuevo tipo de entornos, sobre algunas bibliotecas concretas para estos dispositivos (por ejemplo los paquetes **javax**en el caso de **Java ME**) y comenzar a desarrollar algunos ejemplos de aplicaciones para ir desarrollando una nueva filosofía de trabajo.

Entre las alternativas, tenemos:

* **Android**:
  + IDE: Java ME/Android Studio.
  + Lenguaje: Java y XML.
  + S.O: uno que permita una máquina virtual **Java** para la emulación, aunque normalmente usaremos u emulador proporcionado por el propio **IDE**.
  + **Hardware** (smartPhones, tablets, etc.) que soporte un sistema operativo del tipo anterior.

Como suele suceder en el mundo real, normalmente no podrás elegir una opción u otra en función de tus preferencias personales, sino más bien dependiendo cuáles sean las necesidades de tu cliente. Esa es una buena razón para intentar estar siempre al día.

## 3.2.- La plataforma Android.

Consta de los siguientes componentes:

* **Aplicaciones**: Android incluye aplicaciones base (escritas en Java) como:
  + **Cliente de correo electrónico**
  + **Programa de SMS**
  + **Calendario**
  + **Mapas**
  + **Navegador**
  + **Contactos**
  + **Otros**
* **Marco de trabajo de aplicaciones (framework)**: acceso completo a las mismas API del entorno de trabajo usados por las aplicaciones base. Se reutilizan sus 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 base que proporcionan la mayor parte de las funciones disponibles
* **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 ejecutaba hasta la versión 5.0 archivos en el formato de ejecutable Dalvik (.dex), el cual está optimizado para memoria mínima.
  + La Máquina Virtual está basada en **registros** y corre **clases compiladas en Java** que han sido **transformadas al formato .dex** por la herramienta incluida dx. Desde la versión 5.0 utiliza el ART, que compila totalmente al momento de instalación de la aplicación.
* **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.

## 3.3.- El entorno de ejecución.

**Debes conocer**: a partir de ahora, haremos muchas referencias a la documentación oficial de Google sobre el desarrollo de Android (developer.android.com). De hecho, se puede decir que es la principal guía para elaborar los contenidos relativos a Android de este curso.

Interfaz de usuario gráfica, Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Analicemos las diferentes capas comenzando desde abajo:

1. **Kernel de Linux**: por ejemplo, el tiempo de ejecución de Android (ART, acrónimo de Android Run Time), se basa en el kernel de Linux para funcionalidades subyacentes, como la generación de subprocesos y la administración de memoria de bajo nivel. Ventajas del kernel Linux:
   1. **Funciones de seguridad claves**.
   2. Permite a los fabricantes de dispositivos **desarrollar controladores de hardware** para un kernel conocido.
2. **Capa de abstracción de hardware (HAL)**: brinda interfaces estándares que exponen las capacidades de hardware del dispositivo al [framework de la Java API](https://developer.android.com/guide/platform/index.html?hl=es-419" \l "api-framework" \o "Abrir guia de la plataforma de la Arquitectura Android developer.android.com (se abre en ventana nueva)" \t "_blank)[[](https://developer.android.com/guide/platform/index.html?hl=es-419#api-framework)](https://developer.android.com/guide/platform/index.html?hl=es-419" \l "api-framework" \o "Abrir guia de la plataforma de la Arquitectura Android developer.android.com (se abre en ventana nueva)" \t "_blank) de nivel más alto.
   1. La **HAL** consiste en **varios módulos de biblioteca** y cada uno de estos implementa una interfaz para un tipo específico de componente de hardware, como el módulo de la [cámara[](https://source.android.com/devices/camera/index.html?hl=es-419)](https://source.android.com/devices/camera/index.html?hl=es-419) o de [bluetooth](https://source.android.com/devices/bluetooth.html?hl=es-419)
   2. Cuando el framework de una API realiza una llamada para acceder a hardware del dispositivo, el sistema Android carga el módulo de biblioteca para el componente de hardware en cuestión.
3. **Tiempo de ejecución de Android (ART)**: para los dispositivos con Android 5.0 (nivel de API 21) o versiones posteriores, **cada app ejecuta sus propios procesos** con sus propias instancias del tiempo de ejecución de Android (ART).
   1. ART **está escrito para ejecutar varias máquinas virtuales** en dispositivos de memoria baja ejecutando archivos DEX, un formato de código de bytes diseñado especialmente para Android y optimizado para ocupar un espacio de memoria mínimo.
   2. Crea cadenas de herramientas, como [Jack[](https://source.android.com/source/jack.html?hl=es-419)](https://source.android.com/source/jack.html?hl=es-419), y **compila fuentes de Java en código de bytes DEX** que se pueden ejecutar en la plataforma Android.
   3. Estas son algunas de las funciones principales del ART:
      1. **Compilación ahead-of-time** (AOT, compilación anticipada) y just-in-time (JIT, compilación en tiempo real).
      2. **Recolección** de elementos no usados (GC) optimizada
      3. **Mejor compatibilidad con la depuración**, como un generador de perfiles de muestras dedicado, excepciones de diagnóstico detalladas e informes de fallos, y la capacidad de establecer puntos de control para controlar campos específicos.
      4. **Antes de Android 5.0 (nivel de API 21)**, Dalvik **era el tiempo de ejecución del sistema operativo**. Por tanto ART es una evolución de Dalvik mejorando su rendimiento, en cuanto a velocidad de ejecución y tiempo que tarda en abrir una aplicación. Si tu app se ejecuta bien en el ART, también debe funcionar en Dalvik, pero es posible que no suceda lo contrario.
      5. En Android también se incluye un conjunto de bibliotecas de tiempo de ejecución centrales que proporcionan la mayor parte de la funcionalidad del lenguaje de programación Java; se incluyen algunas [funciones del lenguaje Java 8[](https://developer.android.com/studio/write/java8-support?hl=es-419)](https://developer.android.com/studio/write/java8-support?hl=es-419), que el framework de la Java API usa.
4. **Bibliotecas de C/C++ nativas**: muchos componentes y servicios centrales, como el ART y la HAL; se basan en código nativo que requiere bibliotecas nativas escritas en C y C++. La plataforma Android proporciona la **API del framework de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las apps**.
   1. **Por ejemplo**: puedes acceder a la [guía[](https://developer.android.com/guide/topics/graphics/opengl.html?hl=es-419)](https://developer.android.com/guide/topics/graphics/opengl.html?hl=es-419) de [OpenGL ES](http://localhost/Proyectos/2DAM/Programaci%c3%b3n%20multimedia%20y%20dispositivos%20m%c3%b3viles/Evaluacion%201/Tema%201/Apuntes/PMDM01/WEB/HTML/33_EL_EN.HTM#tbcd74002-38a9-8c41-5edc-1ef3f974e4e7) a través de la [Java OpenGL API[](https://developer.android.com/reference/android/opengl/package-summary.html?hl=es-419)](https://developer.android.com/reference/android/opengl/package-summary.html?hl=es-419) del framework de Android para agregar a tu app compatibilidad con los dibujos y la manipulación de gráficos 2D y 3D.
      1. Si desarrollas una app que requiere C o C++, puedes usar el [NDK de Android[](https://developer.android.com/ndk/index.html?hl=es-419)](https://developer.android.com/ndk/index.html?hl=es-419) para acceder a algunas de estas [bibliotecas de plataformas nativas[](https://developer.android.com/ndk/guides/stable_apis.html?hl=es-419)](https://developer.android.com/ndk/guides/stable_apis.html?hl=es-419) directamente desde tu código nativo
5. **Framework de la API de Java**: Todo el conjunto de funciones del SO Android está disponible mediante API escritas en el lenguaje Java. **Estas API son los cimientos que necesitas para crear apps de Android** **simplificando la reutilización de componentes** del sistema y servicios centrales y modulares, como los siguientes:
   1. Un **sistema de vista enriquecido** y extensible que puedes usar para compilar la IU de una app; se incluyen listas, cuadrículas, cuadros de texto, botones e incluso un navegador web integrable.
   2. Un **administrador de recursos** que te brinda acceso a recursos sin código, como strings localizadas, gráficos y archivos de diseño.
   3. Un **administrador de notificaciones** que permite que todas las apps muestren alertas personalizadas en la barra de estado.
   4. Un **administrador de actividad** que administra el ciclo de vida de las apps y proporciona una pila de retroceso de navegación común.
   5. **Proveedores de contenido** que permiten que las apps accedan a datos desde otras apps, como la app de Contactos, o compartan sus propios datos.
   6. (Para saber más) Los desarrolladores tienen acceso total a las mismas [API del framework[](https://developer.android.com/reference/packages.html?hl=es-419)](https://developer.android.com/reference/packages.html?hl=es-419) que usan las apps del sistema Android.
6. **Apps del sistema**: en Android se incluye un conjunto de apps centrales para correo electrónico, mensajería SMS, calendarios, navegación en Internet y contactos, entre otros elementos.
   1. Las apps incluidas en la plataforma no tienen un estado especial entre las apps que el usuario elije instalar; por ello, una app externa se puede convertir en el navegador web, el sistema de mensajería SMS o, incluso, el teclado predeterminado del usuario (existen algunas excepciones, como la app Settings del sistema).
   2. Las apps del sistema funcionan como apps para los usuarios y brindan capacidades claves a las cuales los desarrolladores pueden acceder desde sus propias apps. Por ejemplo, si en tu app se intenta entregar un mensaje SMS, no es necesario que compiles esa funcionalidad tú mismo; como alternativa, puedes invocar la app de SMS que ya está instalada para entregar un mensaje al receptor que especifiques.

## 3.4.- Máquinas virutales.

Una máquina virtual es una aplicación software que emula el comportamiento de otra máquina.

En el caso de una máquina virtual de **Java** (JVM – Java Virtual Machine), se trata de un programa encargado de interpretar código intermedio (bytecode en el caso de Java) de los programas precompilados (en lenguaje Java) a código máquina ejecutable por el hardware (instrucciones máquina), realizar las llamadas al sistema operativo necesarias, velar por la seguridad e integridad del código ejecutado, etc.

De esta manera, la JVM proporciona al programa compilado en Java independencia de la plataforma hardware y del sistema operativo que corra sobre él.

De esta manera, la **JVM** proporciona al programa compilado en **Java** independencia de la plataforma hardware y del sistema operativo que corra sobre él. Las pantallas de los móviles serían incapaces de soportar toda la funcionalidad proporcionada por la **AWT** (principal interfaz gráfica de usuario que tiene Java). Es por esta razón que la plataforma **Java ME** ha propuesto otras máquinas virtuales bastante más ligeras.

Por otro lado, una única plataforma de Java podría no encajar adecuadamente con todos los modelos de dispositivos posibles. Es por ello por lo que **Java ME** introdujo los conceptos de configuración y de perfil.

Las dos máquinas virtuales disponibles en **Java ME** son la **KVM** (**K Virtual Machine**, se le ha dado ese nombre porque sólo ocuparía unos pocos Kilobytes de memoria) y la **CVM** (**Compact Virtual Machine**), algo más pesada.

De manera similar, para Android, [Dalvik](https://es.wikipedia.org/wiki/Dalvik" \o "Abrir artículo sobre Dalvik en wikipedia (se abre en nueva ventana)" \t "_blank)[[](https://es.wikipedia.org/wiki/Dalvik)](https://es.wikipedia.org/wiki/Dalvik" \o "Abrir artículo sobre Dalvik en wikipedia (se abre en nueva ventana)" \t "_blank), que es la [máquina virtual[](https://es.wikipedia.org/wiki/M%C3%A1quina_virtual)](https://es.wikipedia.org/wiki/M%C3%A1quina_virtual) utilizada originalmente por Android, y lleva a cabo la transformación de la aplicación en instrucciones de máquina, que luego son ejecutadas por el [entorno de ejecución[](https://es.wikipedia.org/wiki/Entorno_de_ejecuci%C3%B3n)](https://es.wikipedia.org/wiki/Entorno_de_ejecuci%C3%B3n) nativo del dispositivo. En versiones más modernas de **Android**, **ART** ha reemplazado a **Dalvik**.