

**Sistema Operativo**

**Android**

**Grupo:** T.2.2

**Autores:** Gabriel García Martínez

Daniel Andina Pailos

Ángel Moltó Álvarez

Jonathan Moreno Añón

**Fecha:** 24/04/2023

**Tabla de contenidos**

[1. Introducción 3](#_Toc133164646)

[2. Características generales de Android 3](#_Toc133164647)

[3. Breve historia 4](#_Toc133164648)

[3.1. Anterior a la compra de Google 4](#_Toc133164649)

[3.2. Posterior a la compra de Google 4](#_Toc133164650)

[4. Versiones de Android 5](#_Toc133164651)

[5. Visión general de la estructura interna del Sistema 5](#_Toc133164652)

[5.1. Kernel de Linux 6](#_Toc133164653)

[5.2. Android HAL 6](#_Toc133164654)

[5.3. Librerías / Runtime 7](#_Toc133164655)

[5.4. Framework de aplicaciones 7](#_Toc133164656)

[5.4.1. Ejemplos APIs 7](#_Toc133164657)

[5.5. Aplicaciones 8](#_Toc133164658)

[6. Visión general del aspecto de las APIs proporcionadas por Android 8](#_Toc133164659)

[6.1. Sistema de vistas 8](#_Toc133164660)

[6.2. Administrador de recursos 9](#_Toc133164661)

[6.3. Administrador de notificaciones 9](#_Toc133164662)

[6.4. Administrador de actividad 10](#_Toc133164663)

[6.5. Proveedores de contenido 10](#_Toc133164664)

[6.6. Administrador de telefonía 10](#_Toc133164665)

[6.7. Administrador de localización 11](#_Toc133164666)

[6.8. Administrador de ventanas 11](#_Toc133164667)

[7. Interfaces de usuario proporcionadas por Android. 11](#_Toc133164668)

[7.1. CLI 11](#_Toc133164669)

[7.1.1. Dispositivo 11](#_Toc133164670)

[7.1.2. Ordenador 12](#_Toc133164671)

[7.2. UI 13](#_Toc133164672)

[7.2.1. ¿Qué es una interfaz de usuario? 13](#_Toc133164673)

[7.2.2. Tipos de interfaces de usuario 13](#_Toc133164674)

[7.2.3. Interfaz gráfica de usuario 14](#_Toc133164675)

[7.2.4. Interfaz controlada por voz 14](#_Toc133164676)

[7.2.5. Interfaz basada en gestos 15](#_Toc133164677)

[8. Conclusiones 15](#_Toc133164678)

[9. Glosario 16](#_Toc133164679)

[10. Referencias 16](#_Toc133164680)

# Introducción

El objetivo de este trabajo es ampliar el poco conocimiento que, en general, se suele tener de los sistemas operativos, más concretamente de Android, uno de los más usados a nivel mundial en dispositivos móviles.

A lo largo de este se abordarán diferentes aspectos sobre el sistema operativo Android, tales como:

* Características generales de Android.
* Breve historia.
* Versiones de Android.
* Visión general de la estructura interna del sistema
* Visión general del aspecto de la(s) API(s) proporcionadas por Android.
* Interfaces de usuario proporcionadas por Android.

# Características generales de Android

Android es un sistema operativo móvil basado principalmente en Linux y también en otros softwares de código abierto. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, aunque en la actualidad también es usado en otros dispositivos, como televisores.

Icono

Descripción generada automáticamenteSu logotipo se caracteriza por ser un androide de color verde.

El almacenamiento se realiza en SQLite, que es una base de datos para almacenamiento local. La mayoría de las aplicaciones están escritas en Java, pero Android no cuenta con una máquina virtual, por lo que utilizan Android Runtime (ART), que crea un archivo de compilación cuando se instala, para evitar que se compile cada vez que se inicia la aplicación. Para el desarrollo de aplicaciones, anteriormente era utilizado el IDE Eclipse. En la actualidad, el IDE usado es Android Studio. Android presenta soporte para formatos Streaming, formatos multimedia, para hardware adicional, como GPS y para Bluetooth. Otra de las características de Android es que presenta multitarea real de las aplicaciones, cuando no están en ejecución, sus ciclos de reloj continúan. Android soporta el Tethering3, que posibilita al dispositivo a ser un punto de acceso inalámbrico.

Los componentes principales de Android son:

* Núcleo Linux: se encarga de la seguridad, la gestión de memoria y procesos, etc. Además, actúa como capa de abstracción entre la pila de software y el hardware.
* Android Runtime (ART): se encarga de la ejecución y compilación de las aplicaciones.
* Aplicaciones: están escritas en Java y se suelen descargar en la Google Play Store, que es la tienda propia de Android.
* Bibliotecas: incluye su propio conjunto de bibliotecas escritas en C/C++.

Android incorpora protección en la mayoría de las acciones posibles, tanto en el navegador, como en las descargas, etc. En contrapartida, al ser software de código abierto, pueden existir vulnerabilidades de código y es más fácil eliminar restricciones de seguridad o instalar programas no autorizados. Esto también hace que las vulnerabilidades sean más fácilmente solucionadas. En la actualidad, Android es menos seguro que su gran competencia, IOS.

# Breve historia

La historia de Android se divide en dos partes: previa a la compra de Google y posterior a la compra de Google.

## Anterior a la compra de Google

Android Inc fue creada en 2003 en California por Rich Miner, Andy Rubin, Chris White y Nick Sears con el objetivo de desarrollar un sistema operativo para móviles basado en Linux. Tres años después, en 2006, la compañía fue comprada por Google.

## Posterior a la compra de Google

Tras la compra de Google, a finales de 2007 Android lanzó su primera versión Beta pública. Esta primera versión incluía el primer SDK público de Android, aunque no incluía aplicaciones de terceros. Esta primera versión ya incluía la barra de notificaciones, similar a la actual, el dock, o fila inferior de iconos y el escritorio. Era una versión poco pulida gráficamente. Esta versión apenas tenía aplicaciones, incluía Google Maps y la app de Contactos y llamadas.

También, en 2007, Google, HTC, Samsung, Intel, Motorola, T-Mobile, entre otras empresas., fundaron OHA, Open Handset Alliance, con el objetivo de promover y desarrollar el sistema operativo Android. El objetivo principal de esta alianza fue crear una plataforma de software abierta y estándar para dispositivos móviles, que permitiera a los desarrolladores crear aplicaciones y servicios sin restricciones y sin tener que pagar derechos de licencia. Además, les proporciona a las empresas las herramientas de desarrollo y los recursos para crear dispositivos compatibles. Desde su creación, la OHA ha trabajado para mejorar y evolucionar Android.

Un año después, Google lanzó la primera versión estable, Android 1.0. El primer móvil que usó este sistema operativo fue el HTC Dream. Esta primera versión de Android no incluía teclado virtual. Esta versión fue actualizada un mes después con el objetivo de corregir errores. Su mayor novedad es que era una actualización over-the-air, lo cual era revolucionario en ese momento.

En el año 2010, Xiaomi lanzó el primer sistema operativo basado en Android. Esto supuso una revolución que posteriormente fue copiado por otras empresas. MIUI es una versión modificada de Android, que cuenta con una interfaz de usuario similar a la de IOS.

En el año 2013, Oppo Electronics, copiando lo realizado por Xiaomi, lanzó su propio sistema operativo basado en Andoid. Es utilizado en los móviles Oppo y Realme. Se caracteriza por ofrecer pantalla dividida y la navegación por gestos.

También han sacado sus propios sistemas operativos basados en Android otras empresas, como OnePlus, con su sistema OxygenOS, en 2015. Por su parte, Samsung tiene su propia interfaz de usuario para Android, llamada OneUI. Está diseñado para facilitar el uso de dispositivos más grandes.

Android ha lanzado un gran número de versiones, aproximadamente un año, y la última es Android 13, que fue lanzada el 16 de agosto de 2022. Posteriormente analizaremos las diferentes versiones.

Actualmente, Android es el sistema operativo móvil más utilizado en el mundo, con una cuota de mercado del 71% (dato Statisa, marzo 2023).

# Versiones de Android

El Sistema operativo Android tiene diecinueve versiones, y desde Android 1.5 hasta Android 9.0, se caracterizan por su nombre de postre, estas son las más importantes:

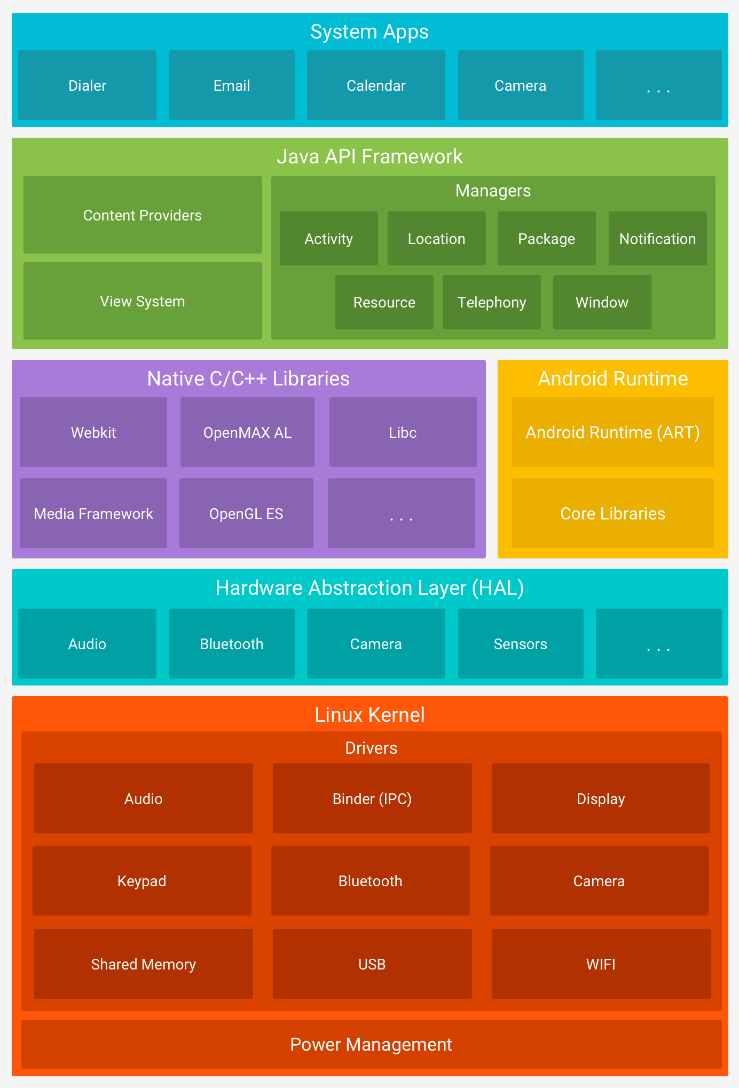
* **Android 1.0** (2008): Fue la primera versión pública de Android, que presentó características básicas como la integración de Google Maps, el navegador web y la capacidad de instalar aplicaciones desde Google Play Store.
* **Android 2.2** (2010): También conocido como Froyo, introdujo la función de hotspot Wi-Fi, mejoras en el rendimiento y la capacidad de instalar aplicaciones en la tarjeta SD.
* **Android 2.3** Gingerbread (2010): La principal novedad es la API para juegos y el soporte para NFC1.
* **Android 4.0** (2011): Conocida como Ice Cream Sandwich, fue una de las versiones más importantes de Android, ya que unificó la experiencia de usuario en tabletas y teléfonos inteligentes. Además, presentó mejoras significativas en la interfaz de usuario, el rendimiento y la seguridad.
* **Android 4.4** (2013): Conocida como KitKat, presentó mejoras en el rendimiento y la estabilidad, así como la integración de Google Now.
* **Android 5.0** (2014): Conocida como Lollipop, presentó la nueva interfaz de usuario Material Design, que ofrecía una apariencia más moderna y limpia. Además, mejoró la duración de la batería y presentó una nueva función de notificaciones.
* **Android 6.0** (2015): Conocida como Marshmallow, presentó mejoras en la privacidad y la seguridad, así como la integración de Google Now on Tap.
* **Android 7.0** (2016): Conocida como Nougat, presentó mejoras en la duración de la batería, la multitarea y la integración de realidad virtual.
* **Android 8.0** (2017): Conocida como Oreo, presentó mejoras en el rendimiento y la seguridad, así como la función Picture-in-Picture para ver videos mientras se realizan otras tareas.
* **Android 9.0** (2018): Conocida como Pie, presentó mejoras en la navegación gestual, la inteligencia artificial y la duración de la batería.
* **Android 10** (2019): Presentó mejoras en la privacidad, la navegación gestual y la compatibilidad con dispositivos plegables.
* **Android 11** (2020): Presentó mejoras en las notificaciones, la privacidad y la compatibilidad con dispositivos 5G.

# Visión general de la estructura interna del Sistema

Android es un sistema operativo de basado en el núcleo Linux. Su estructura interna se compone de las siguientes capas:

* Kernel de Linux
* Android HAL
* Librerías / Runtime
* Framework de aplicaciones
* Aplicaciones

## Kernel de Linux

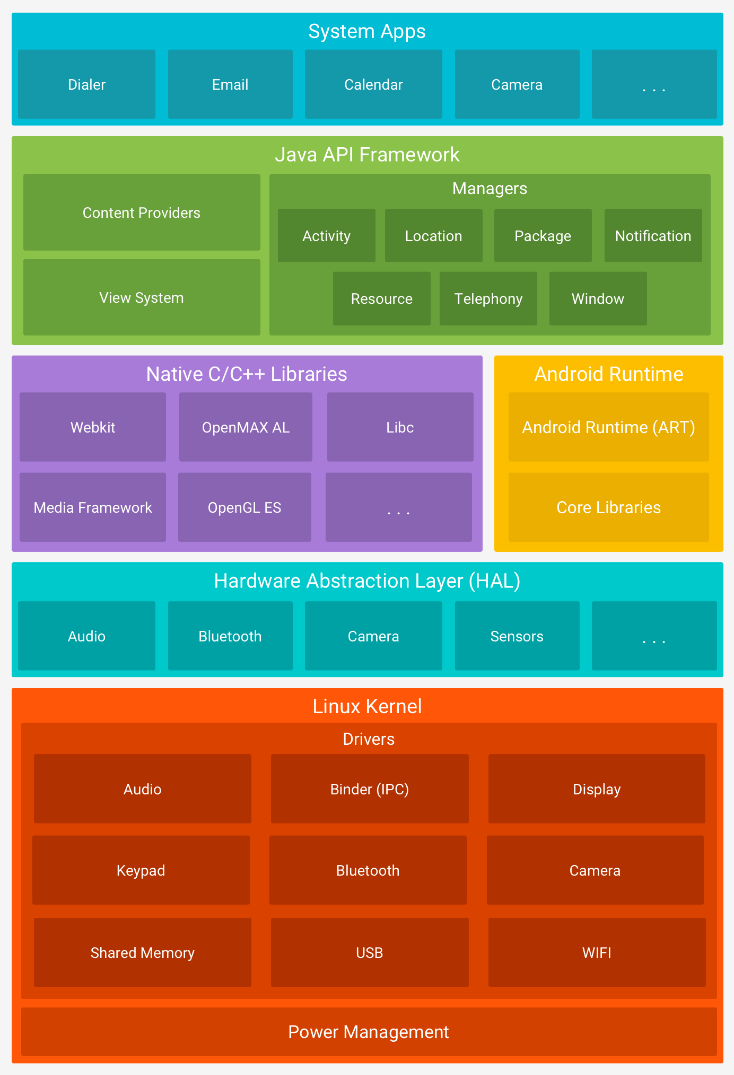


El **Kernel** **de** **Linux** es la base del sistema operativo y proporciona los servicios esenciales del sistema, como la gestión de memoria y procesos, el acceso a los dispositivos hardware y la seguridad del sistema.

Esta versión del kernel de Linux en Android ha sido personalizada y optimizada para su uso en dispositivos móviles, lo que ha permitido la incorporación de nuevas funciones como el soporte para pantallas táctiles, la conectividad inalámbrica y la gestión de energía.

El uso del kernel de Linux permite a los fabricantes de dispositivos desarrollar controladores hardware para un kernel popular. Además, proporciona una interfaz común para los diferentes componentes de hardware en un dispositivo Android lo que permite que el HAL (Hardware Abstraction Layer) actúe como una capa de abstracción para los controladores hardware.

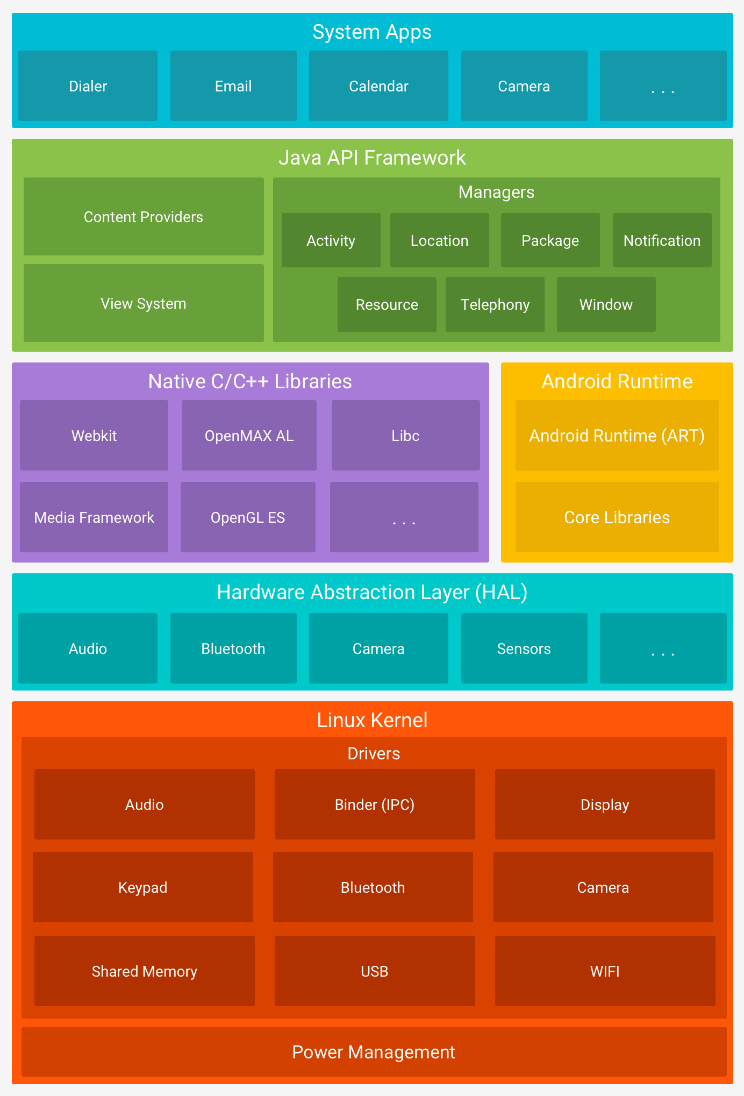
## Android HAL



El **HAL** de Android (Hardware Abstraction Layer) es una capa de software que se encuentra entre el kernel de Linux y el Framework de Aplicaciones de Android en la estructura interna de Android. Su función es proporcionar una interfaz estándar para los controladores de los dispositivos de hardware del sistema, como la cámara, el sensor de huellas dactilares, el acelerómetro, entre otros. Esto permite que el sistema operativo Android sea compatible con una amplia variedad de dispositivos de hardware sin necesidad de modificar el kernel de Linux.

El HAL de Android también es importante porque permite que los desarrolladores de hardware proporcionen controladores específicos para sus dispositivos, sin tener que preocuparse por la compatibilidad con Android. Además, esto permite que los fabricantes de dispositivos Android puedan personalizar y adaptar el sistema operativo a sus propios dispositivos sin afectar la compatibilidad con las aplicaciones existentes.

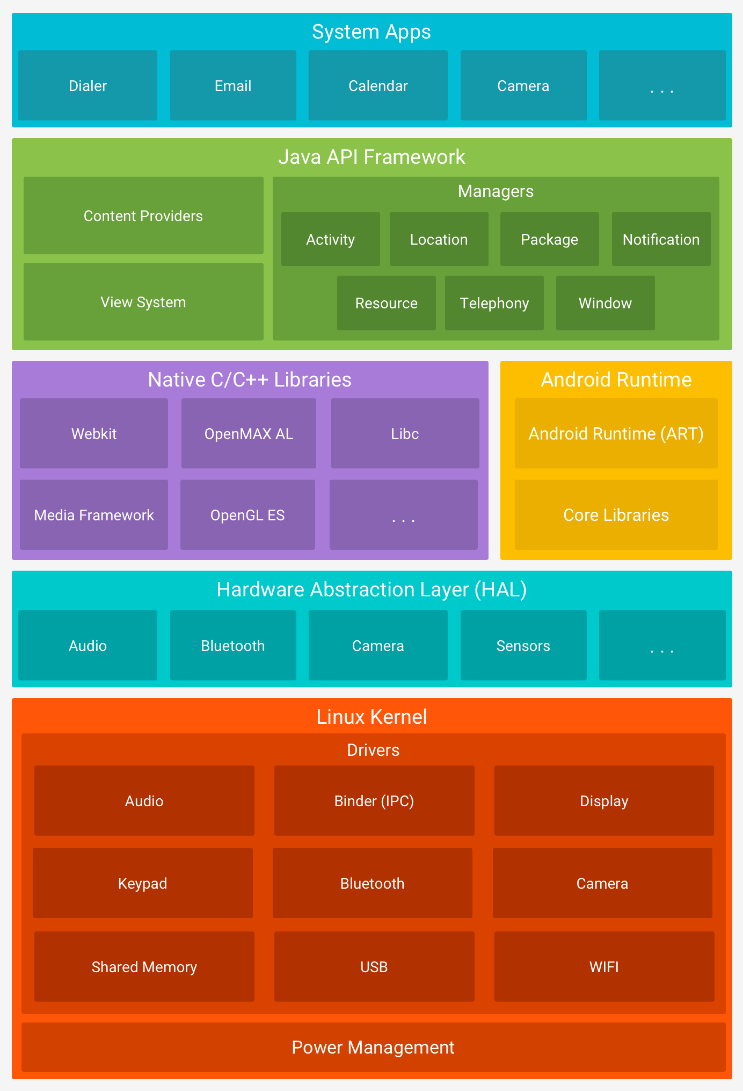
## Librerías / Runtime



La capa de **librerías** de Android es una colección de librerías escritas en C y C++ que ofrecen una variedad de funciones del sistema, como la gestión de gráficos, multimedia y bases de datos. Estas librerías proporcionan una interfaz de programación de aplicaciones (API) para que los desarrolladores de aplicaciones puedan acceder a estas funciones de manera fácil y eficiente.

Por otro lado, el **ART** (Android Runtime) es el entorno de ejecución para las aplicaciones de Android y se ejecuta en la máquina virtual de Java de Android. El ART es responsable de la compilación de las aplicaciones de Android en código nativo para mejorar el rendimiento y la eficiencia energética de las aplicaciones. Además, utiliza técnicas de optimización para reducir el tiempo de inicio de las aplicaciones y mejorar la experiencia del usuario.

## Framework de aplicaciones

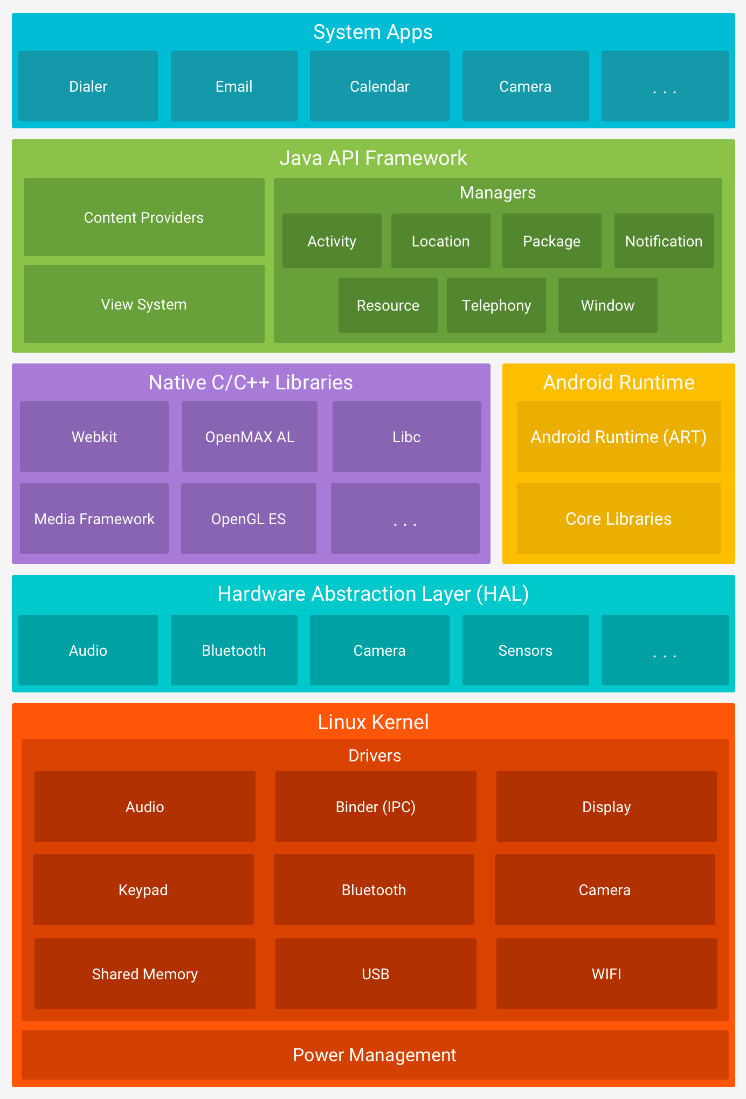


La capa **Framework** de aplicaciones de Android es una parte importante de la arquitectura de Android que proporciona una plataforma para que las aplicaciones accedan a las funciones del sistema operativo. Está construida sobre la capa de librerías de Android y proporciona un conjunto de APIs2 para que los desarrolladores creen aplicaciones que interactúen con el hardware y software del dispositivo con una funcionalidad completa y accesible para los usuarios.

### Ejemplos APIs

* **Activity** **Manager**: Gestiona el ciclo de vida de las aplicaciones.
* **Content** **Provider**: Permite que las aplicaciones accedan a datos de otros programas.
* **Package** **Manager**: Se encarga de instalar y actualizar aplicaciones en el dispositivo.
* **Window** **Manager**: Gestiona las ventanas y la interfaz de usuario.
* **Location** **Manager**: Proporciona información sobre la ubicación del dispositivo.
* **Notification** **Manager**: Gestiona las notificaciones de las aplicaciones.
* **Telephony** **Manager**: Permite acceder a información relacionada con la telefonía móvil.

## Aplicaciones



La capa de **aplicaciones** de Android es la capa superior de la arquitectura de Android que incluye todas las aplicaciones preinstaladas y descargables en un dispositivo Android. Estas aplicaciones están escritas en Java o en lenguajes compatibles con la máquina virtual de Java.

La capa de aplicaciones de Android incluye aplicaciones de mensajería, de correo electrónico, navegadores web, reproductores multimedia, juegos y mucha más variedad. También incluye aplicaciones de sistema, como la aplicación de configuración y la cámara.

Además, los usuarios pueden descargar e instalar aplicaciones adicionales desde la tienda de aplicaciones de Google Play. Los desarrolladores pueden utilizar las APIs de Android y los componentes de la capa framework para crear aplicaciones personalizadas y hacer uso de las funciones del sistema operativo.

# Visión general del aspecto de las APIs proporcionadas por Android

## Sistema de vistas

Encargado del procesamiento y compilación de las Interfaces de Usuario de una aplicación.

Diagrama

Descripción generada automáticamente

**ViewGroup:** constituyen los layouts, contienen Views y ViewGroups y determinan su distribución y diseño.

**View:** constituyen los componentes (widgets) Son la clase base para todo componente interactivo de la interfaz de usuario

La clase View ofrece además manejo de eventos y de foco, estilos (themes), cambio entre navegación con teclado a navegación táctil, opciones de seguridad…

La **comunicación** entre los componentes View y los ViewGroups que los contienen se realiza a través de instancias de **ViewGroup.LayoutParams**, que son usadas por las Views para indicar a los ViewGroups cómo quieren ser representados.

La **implementación** de la interfaz de usuario puede hacerse mediante uso de XML mediante escritura directa o mediante uso de editores de diseño (drag and drop), o bien puede hacerse programáticamente, creando los respectivos componentes de la UI en tiempo de ejecución. Usar XML separa interfaz de gestión de eventos, siendo una opción superior.

## Administrador de recursos

Se encarga de proporcionar al programador estructuramiento y acceso a los recursos de la aplicación fácilmente. Son **recursos** de la aplicación todo lo necesario para el correcto funcionamiento de la aplicación, sin ser código necesariamente. (imágenes, audio, datos, animaciones, traducciones de texto…)

Todo tipo de recurso debe colocarse en el subdirectorio res/ del proyecto. Dentro del mismo, se distinguen varios subdirectorios, entre otros:

* **color/,** que contiene archivos XML que describen selectores de colores (ColorStateList), los cuales cambian de color al objeto View que tengan asignado en función de su estado. Ejemplo: botón verde sin presionar, blanco presionado.
* **drawable/,** contiene archivos de mapas de bits (.png, 9.png, .jpg, .gif) además de recursos varios de diseño.
* **mipmap/,** usado para contener iconos de la aplicación. No se guardan en /drawable ya que los elementos guardados en drawable/ pueden ser escalados en función a la resolución que soporta el dispositivo, causando que los iconos pierdan calidad.
* **layout/, menu/** contienen archivos XML que describen el diseño de la interfaz de usuario y los menús que tiene respectivamente.
* **values/** contiene archivos XML que definen grupos de valores. Conviene clasificar los archivos por el tipo de datos que definen (Colores.XML, Dimensiones.XML, Strings.XML…)
* **xml/ y font/** almacenan XMLs misceláneos y fuentes, respectivamente.

**Imagen que contiene Texto

Descripción generada automáticamente**Se pueden añadir calificadores de diferenciación a los directorios creados con intención de tener diferentes recursos disponibles en función de las necesidades de la aplicación. Hay muchos calificadores, entre ellos:

* **Idioma y región**: (-<idioma>-r<país>) -en-rUS. (inglés, en EE. UU)
* **Dirección de diseño**: usado para distinguir en el diseño entre lenguajes leídos de izquierda a derecha (-ldltr) o de derecha a izquierda. (ldrtl)
* **Orientación:** vertical (-port) u horizontal (-land)
* **Modos de IU:** -car, -desk, -television, -appliance, -watch, -vrheadset.
* **Densidad de píxeles en pantalla:** -ldpi, -mdpi, -hdpi, -xhdpi, -xxhdpi…
* **Anchura/alturas disponibles:** -w<N>dp, -h<N>dp

Ejemplo de estructura de res/

## Administrador de notificaciones

Las notificaciones envían avisos breves y concisos al usuario sobre el estado de la aplicación. El sistema soporta varias plantillas de formato de aplicaciones, aunque el programador puede crear las suyas propias. Un ejemplo:

**1, 2, 3**) icono pequeño, nombre de app y tiempo de creación.

**4**) (Opcional) Imagen

Texto

Descripción generada automáticamente**5, 6**) (Opcionales) Título y texto

Para crear las notificaciones, será necesario crear un builder, al que asignarle los atributos que se deseen.

* Pueden añadirse componentes (botones, barras de progreso…)
* Toda notificación requiere estar asociada a un canal de notificaciones.
* Pueden establecerse categorías del sistema para notificaciones, de tal forma que se pueda determinar cuándo estas deben aparecer, (Modo No Molestar) además de establecer niveles de prioridad que determinan cuánto espacio ocupan en pantalla, si emiten sonido… (alarmas, llamadas)

## Administrador de actividad

Diagrama

Descripción generada automáticamente**OnCreate():** Inicialización, tareas que solo deben hacerse una vez.

**onStart()**: Preparación de la app para comenzar a ser usada.

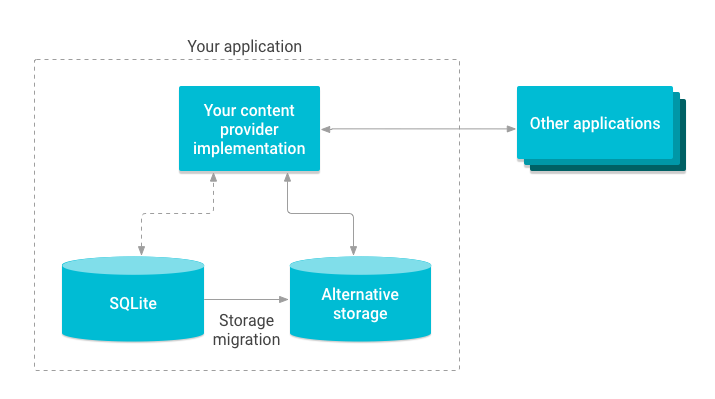
**onResume()**: Uso directo de la aplicación, bien sea desde su creación o porque ha sido restaurada de un bloqueo. Dado que el sistema puede desechar partes de la actividad que estén bloqueadas, es importante restaurarlas en esta llamada.

**onPause()**: Ocurre cuando eventos interrumpen la actividad, o cuando otra actividad reclama el foco, dejando parcialmente visible a la actual.

**onStop()**: Actividad a punto de finalizar, o cubierta totalmente por otra. Desecha gran parte de los recursos, pero aún pueden ser recuperados. Pensado para operaciones pesadas.

**onDestroy()**: Actividad finalizada. Libera todos los recursos.

## Proveedores de contenido

Proporcionan formas de gestión de datos de una aplicación dada y de otras, además de soportar el traspaso de información entre ellas. Son una abstracción entre la aplicación y el almacenamiento de sus datos. También son muy útiles en la regulación de permisos de acceso a los datos. Necesarios para el copiado y pegado de datos o archivos complejos.

## Administrador de telefonía

Procesa los servicios de telefonía (llamadas) además de canales de comunicación por subscripción. Gestionado a través de la clase TelephonyManager.

## Administrador de localización

Puede accederse a la ubicación del dispositivo a través de un FusedLocationProviderClient, obtenible de la clase LocationServices a través de getters. Usando su método getLastLocation(), se obtiene un objeto Location con la información de la ubicación deseada.

Texto

Descripción generada automáticamenteTambién hay formas de implementar un flujo constante de obtención de ubicaciones en tiempo de ejecución, usando el método requestLocationUpdates(), lo cual es útil en aplicaciones como GPS (Google Maps) Ejemplo:

## Administrador de ventanas

Servicio ofrecido por un WindowManager que se encarga de la gestión de ventanas. Usado para determinar su orden en “profundidad”, crear transiciones entre ventanas, y mostrar las ventanas que estén “encima”. Por defecto, estas ventanas cubren toda la pantalla, pero puede hacerse que ocupen solo parte de ella configurando WindowManager.

# Interfaces de usuario proporcionadas por Android.

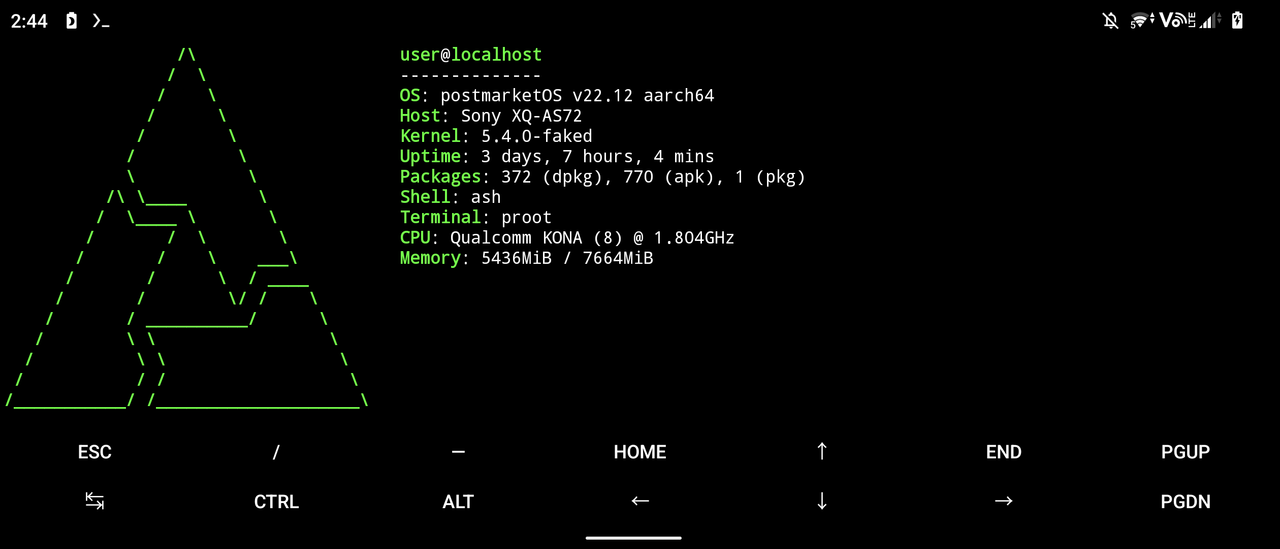
En este apartado distinguimos en dos las formas del usuario para comunicarse con el Sistema Operativo, la interfaz de línea de comandos y la interfaz de usuario, “CLI” y “UI” por sus siglas en inglés.

## CLI

En el contexto de la línea de comandos en Android, es importante distinguir entre las opciones que se ejecutan directamente en el dispositivo y aquellas que se conectan a través de una conexión USB a un ordenador.

### Dispositivo

En el caso de las opciones que se ejecutan directamente en el dispositivo, es cierto que Android no cuenta con una interfaz de línea de comandos integrada de forma predeterminada. Sin embargo, esto no significa que los usuarios no puedan acceder a la línea de comandos en sus dispositivos Android. Hay varias aplicaciones de terminal disponibles en Google Play Store que permiten a los usuarios interactuar con su dispositivo mediante la introducción de comandos de texto. Algunas de las aplicaciones más populares incluyen Termux, Terminal Emulator for Android y Console.



Estas aplicaciones de terminal ofrecen una variedad de comandos que pueden ser ejecutados por los usuarios, como comandos de Unix, comandos de red y comandos de sistema. Aunque la interfaz de usuario puede parecerse a una terminal de Linux, es importante tener en cuenta que el nivel de acceso y control que se le da al usuario en un dispositivo Android es mucho más limitado en comparación con un sistema operativo de escritorio.

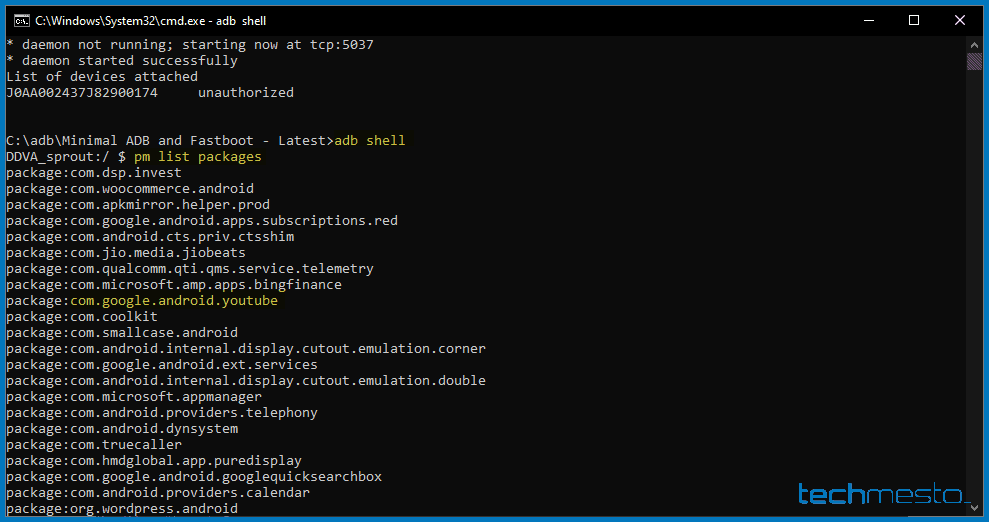
A pesar de esto, el uso de una aplicación de terminal en Android puede ser útil para realizar tareas de mantenimiento o solucionar problemas en el dispositivo. Es importante tener en cuenta que el acceso a la línea de comandos en un dispositivo Android puede requerir ciertos conocimientos técnicos y puede ser potencialmente peligroso si no se sabe lo que se está haciendo.

### Ordenador

En este caso, nos encontramos con las herramientas instalables en un ordenador, ofrecidas principalmente por Google desde su página para desarrolladores. La utilidad más relevante es sin duda el “Android Debug Bridge” (ADB) dado que es una herramienta de CLI que permite la comunicación con el dispositivo mediante una shell de Unix. Se compone de:

* **Cliente**: Ejecutado en la máquina de desarrollo.
* **Daemon**: Ejecuta los comandos (ejecutado en segundo plano en cada dispositivo)
* **Servidor**: Gestiona la comunicación cliente y servidor (ejecutado en la máquina de desarrollo).

Entre algunas de las razones más habituales por las que usuarios técnicos usan esta herramienta, de las más potentes en su ámbito, son la depuración de aplicaciones, el inicio del dispositivo usando diferentes modos, como el “sideloading”, lo que suele hacerse para cargar una ROM personalizada y diferente a la proporcionada con el fabricante. En la siguiente imagen se muestra un ejemplo de cómo se inicia el servidor previamente visto, y luego se listan los paquetes instalados en el dispositivo.



También se ofrecen herramientas para el desarrollador que son comúnmente ejecutadas sobre el ya mencionado ADB, como “apkanalyzer”,” avdmanager”,” sqlite3”,” retrace” y” sdkmanager”. Estos programas tienen utilidades como escanear código e identificar problemas de calidad en la estructura del mismo, instalar y desinstalar paquetes del “Software Development Kit” (SDK) y gestionar las BBDD de SQLite creadas por aplicaciones Android.

También merece la pena comentar, que ninguno de los métodos anteriores para iniciar una shell te convierte en root de forma predeterminada, pero si el dispositivo tiene acceso root (lo cual escapa de las competencias de este trabajo), puedes usar el comando "su" de la manera habitual para convertirte en root. Cualquier gestor de root que tengas en tu teléfono (Superuser, SuperSU, etc.) probablemente aparecerá en la pantalla para que confirmes el acceso root.

## UI

### ¿Qué es una interfaz de usuario?

La interfaz de usuario es la herramienta que el usuario utiliza para interactuar con diferentes componentes del sistema operativo Android. Es la cara visible de la aplicación y su diseño debe ser intuitivo para facilitar la experiencia del usuario. Para ello, se presenta de diferentes formas, con el objetivo de adaptarse a las necesidades y preferencias de los usuarios. Es a través de la UI que los usuarios pueden interactuar con los distintos elementos de una aplicación.

### Tipos de interfaces de usuario

**-Interfaz Gráfica de Usuario (GUI)**: Esto ayuda a los usuarios a interactuar con representaciones visuales en dispositivos digitales o smartphones.

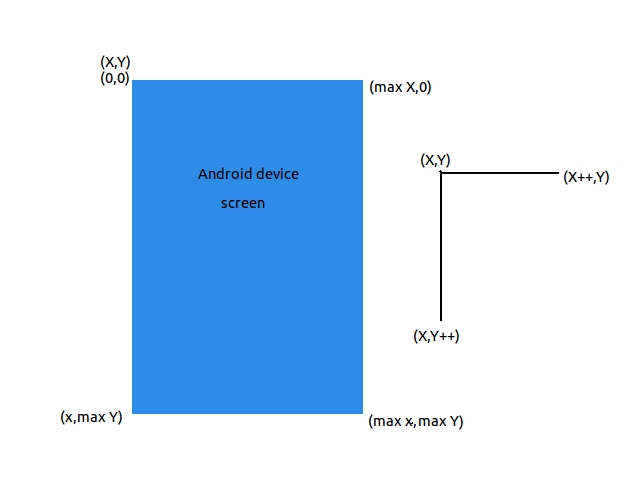
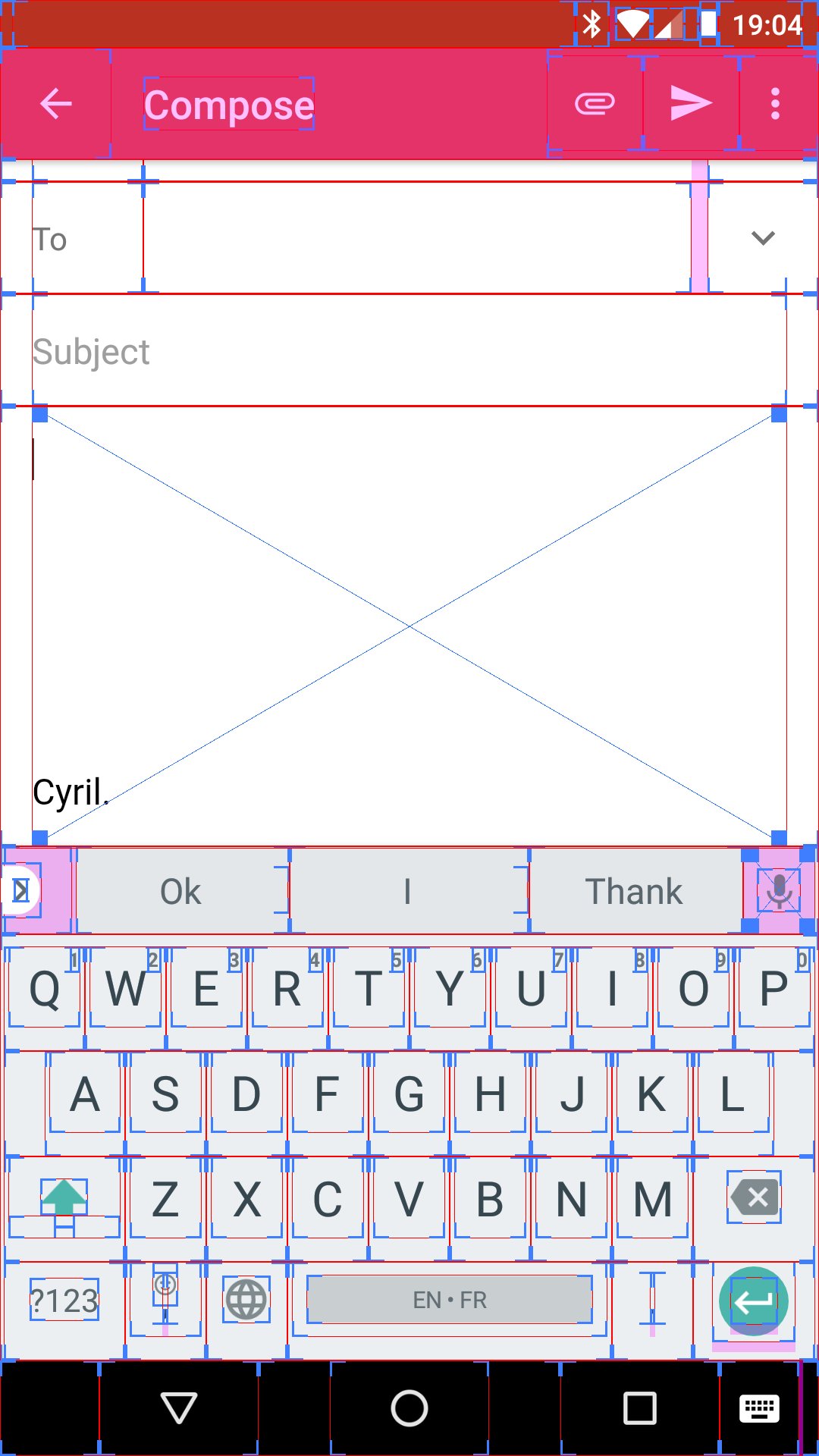
**-Interfaz controlada por voz**: Permite al usuario interactuar a través de comandos de voz.

**-Interfaz basada en gestos**: El usuario puede interactuar con la interfaz a través de gestos tales como deslizar, arrastrar…

### Interfaz gráfica de usuario

El sistema operativo Android tiene una interfaz gráfica de usuario (GUI) nativa que se utiliza en la mayoría de los dispositivos Android. Puede ser modificada por el fabricante del dispositivo, para lo que se usa el framework de aplicación de Android, que proporciona una amplia variedad de herramientas y recursos para diseñar y construir interfaces gráficas de usuario, con lo que existen multitud de interfaces diferentes.

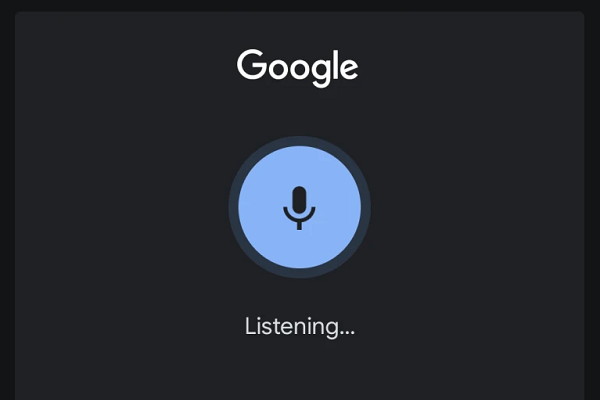
Android utiliza vistas, controles y diseños para crear elementos visuales en la pantalla. Las vistas definen la apariencia y comportamiento de los elementos visuales, mientras que los controles manejan las interacciones del usuario. Los diseños organizan las vistas en la pantalla y se definen en archivos XML. Android utiliza un sistema de coordenadas en píxeles para posicionar los elementos en la pantalla.



### Interfaz controlada por voz

Las interfaces controladas por voz son una tecnología que permite a los usuarios interactuar con sus dispositivos mediante comandos de voz para realizar ciertas tareas como escribir texto e incluso controlar el dispositivo. Para implementar esta tecnología en aplicaciones, los desarrolladores pueden utilizar herramientas y lenguajes de programación específicos, como el Android SDK, la API de Google Cloud Speech-to-Text y TensorFlow.

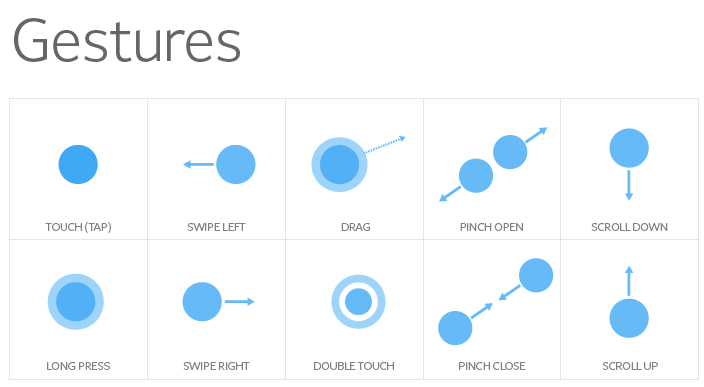
El Android SDK es una herramienta que proporciona una serie de recursos y bibliotecas para el desarrollo de aplicaciones en dispositivos Android. Entre estas herramientas se encuentra una API de reconocimiento de voz que permite integrar la tecnología de reconocimiento de voz en una aplicación. Por otro lado, la API de Google Cloud Speech-to-Text utiliza técnicas avanzadas de aprendizaje automático para transcribir el habla en texto, enviándolo a un servidor y procesándolo en la nube. Por último, TensorFlow es una plataforma de aprendizaje automático de código abierto desarrollada por Google, que se utiliza ampliamente en aplicaciones de reconocimiento de voz y otras aplicaciones de procesamiento de señales de audio. Proporciona una amplia variedad de herramientas y bibliotecas para el procesamiento de señales de audio y el aprendizaje automático.



### Interfaz basada en gestos

Las interfaces controladas por gestos en Android son una forma intuitiva de interactuar con los dispositivos móviles mediante movimientos físicos. Son una alternativa a la interacción basada en botones y menús, y pueden mejorar significativamente la usabilidad de las aplicaciones y sistemas operativos. Sin embargo, las interfaces controladas por gestos también presentan algunos desafíos para los usuarios. Es importante que los desarrolladores ofrezcan opciones para la interacción basada en botones y menús para aquellos que prefieren una experiencia más tradicional.

En Android, los gestos se implementan mediante diversas tecnologías y herramientas. El sistema operativo Android ofrece una variedad de gestos integrados, como el deslizamiento hacia arriba para desbloquear, el deslizamiento hacia abajo para acceder al panel de notificaciones y el pellizco para hacer zoom. Los desarrolladores también pueden aprovechar esta tecnología para crear experiencias de usuario más fluidas y atractivas.



# Conclusiones

Es importante conocer este sistema operativo por su prevalencia en los dispositivos móviles y por la gran presencia en la vida de las personas ya que cada vez más tienen acceso a uno de estos dispositivos.

Esperamos que con la información recogida en este documento el lector haya conseguido tener una idea general del sistema operativo, su estructura y su historia.

# Glosario

* **NFC1:** Tecnología inalámbrica que permite la comunicación e intercambio de datos entre dos dispositivos.
* **API2:** Parte expuesta del framework, deiseñada para modularizar y simplificar en gran medida las operaciones que pueda hacer un programador en el desarrollo de aplicaciones.
* **Tethering3:** permite a un dispositivo electrónico ofrecer conexión a internet a otro, mediante cable, WiFi o Bluetooth.

# Referencias

* Android Developers. "Versiones de plataforma". En: Android Developers. [en línea]. [Consulta: 18 de abril de 2023]. Disponible en: <https://developer.android.com/about/dashboards>
* Wikipedia. "Historial de versiones de Android". En: Wikipedia, la enciclopedia libre. [en línea]. [Consulta: 18 de abril de 2023]. Disponible en: <https://es.wikipedia.org/wiki/Historial_de_versiones_de_Android>
* Wikipedia. Android - Historia [en línea]. Consultado el 18 de abril de 2023. Disponible en: <https://es.wikipedia.org/wiki/Android#Historia>
* Developer.android.com. "Plataforma Android". En: Android Developers. [en línea]. [Consulta: 18 de abril de 2023]. Disponible en: <https://developer.android.com/guide/platform?hl=es-419>
* Lozano, J. "Arquitectura de Android". En: Source.android.com. [en línea]. [Consulta: 18 de abril de 2023]. Disponible en: <https://source.android.com/docs/core/architecture>
* Developer.android.com. "Herramientas de línea de comandos". En: Android Developers. [en línea]. [Consulta: 18 de abril de 2023]. Disponible en: <https://developer.android.com/studio/command-line/adb?hl=es-419>
* Support.google.com. "Utilizar gestos de accesibilidad en Android - Ayuda de Accesibilidad". En: Google Support. [en línea]. [Consulta: 18 de abril de 2023]. Disponible en: <https://support.google.com/accessibility/android/answer/6151848?hl=es>
* Source.android.com. "Guía de interacción de voz | Dispositivos Automotrices". En: Source.android.com. [en línea]. [Consulta: 18 de abril de 2023]. Disponible en: <https://source.android.com/docs/devices/automotive/voice/voice_interaction_guide>
* Developer.android.com. "Plataforma Android". En: Android Developers. [en línea]. [Consulta: 18 de abril de 2023]. Disponible en: <https://developer.android.com/guide/platform?hl=es-419>
* Developer.android.com. "Declarar la IU en Android". En: Android Developers. [en línea]. [Consulta: 18 de abril de 2023]. Disponible en: <https://developer.android.com/guide/topics/ui/declaring-layout?hl=es-419>
* Developer.android.com. "Guía del programador para Android". En: Android Developers. [en línea]. [Consulta: 18 de abril de 2023]. Disponible en: <https://developer.android.com/guide?hl=es-419>
* Developer.android.com. "Proveedores de contenido". En: Android Developers. [en línea]. [Consulta: 18 de abril de 2023]. Disponible en: <https://developer.android.com/guide/topics/providers/content-providers?hl=es-419>