Android tiene licencia de código abierto, los fabricantes pueden realizar sus propias adaptaciones. ***Características de Android:***

1. ***Mensajería:*** Soporta SMS / servicio de mensajes cortos y MMS / servicio de mensajería multimedia.
2. ***Soporte de hardware:*** Acelerómetro, cámara, brújula, sensor y GPS.
3. ***Almacenamiento:*** Usa SQLite, una base de datos relacional ultraligera para el almacenamiento de datos.
4. ***Formatos:*** Soporta multitud de formatos multimedia.
5. ***Conectividad:*** Soporta Bluetooth, Wi-fi, WiMAX, GSM, LTE, etc…

El sistema operativo de ***Android está dividido en cinco secciones:***

1. ***Kernel de Linux:*** Android está basado en este kernel. Esta capa contiene todos los drivers de dispositivos a bajo nivel para los componentes del dispositivo.
2. ***Librerías:*** Contienen el código de las características principales de Android.
3. ***Android Runtime:*** Está al mismo nivel que las librerías, proporciona librerías fundamentales para escribir aplicaciones. Este también incluye una máquina virtual Dalvik que permite que las aplicaciones se ejecuten en su propio proceso.
4. ***Framework de aplicaciones:*** Proporciona las características de Android para usarlas en aplicaciones.
5. ***Aplicaciones:*** Capa más alta. En esta se encuentran todo tipo de aplicaciones.

Un dispositivo móvil tiene buena movilidad por su peso y tamaño, tiene conexión WiFi o permanente usando redes de telefonía; tiene alta capacidad de interacción con la pantalla táctil y tiene capacidad de procesamiento y memoria interna. Además de móviles y tablets, existen dispositivos wearables que se conectan mediante bluetooth a un dispositivo android.

Estos ***tienen varias limitaciones:*** Capacidad de procesamiento y RAM reducida, dimensiones reducidas y debe tener buena seguridad.

La programación para dispositivos móviles tiene unas características particulares que la diferencian de la programación tradicional, estas son debidas a las limitaciones de los dispositivos móviles. El sistema operativo y la API / Interfaz de Programación de Aplicaciones que se usan para las aplicaciones del sistema son únicos.

***PARTES DE UNA APLICACIÓN ANDROID:***

1. La carpeta Java contiene las clases .java con el código del proyecto, en especial de las actividades.
2. El directorio /res contiene los recursos que se usan en la aplicación, en uno de sus subdirectorios se genera un .xml con interfaz gráfica por cada actividad que hayamos creado. Los elementos que albergue harán referencia a otros elementos ubicados en subdirectorios de la carpeta /res.
3. El directorio /assets contiene otros recursos como archivos o bases de datos. Habrá que crearlo manualmente.
4. En el archivo AndroidManifest.xml del directorio /manifest se especifican los permisos y otras características de la aplicación.

***BLOQUES PARA CONSTRUIR APLICACIONES ANDROID:***

1. ***Actividades:*** Componentes de la aplicación que provee una pantalla con la que interactuar. En Android las actividades pueden activar actividades de otras aplicaciones.
2. ***Fragmentos:*** Es una porción de una actividad. Se pueden tener varios fragmentos y modificarlos y combinarlos en la misma actividad.
3. ***Servicios:*** Es un componente que se ejecuta en segundo plano / background. Estos pueden hacer lo mismo que una actividad pero no cuentan con interfaz gráfica. Se pueden poner en marcha hasta que termine su ejecución.
4. ***Proveedores de contenido / content providers:*** Son interfaces para compartir datos entre aplicaciones, ya que estos se encuentran aislados del resto. Los intentos traspasan pequeñas cantidades de datos entre aplicaciones, los proveedores de contenidos están más capacitados para compartir conjuntos de datos. Esta información se puede insertar, borrar, actualizar y consultar desde un proveedor de contenidos. Se usan sobre todo para leer y escribir datos que son privados en la aplicación.
5. ***Receptores de broadcast / broadcast receivers:*** Responden a anuncios del sistema dirigidos a todas las aplicaciones del dispositivo. Este espera latente para activarse cuando ocurra un evento al que está suscrito. El sistema está continuamente enviando mensajes de broadcast continuamente. Las aplicaciones también pueden enviar estos mensajes. Estos no tienen interfaz gráfica, pero al ejecutar un código pueden crear una notificación avisando del broadcast.

Todos estos se activan por intentos, estos atan los componentes individuales entre sí durante la ejecución.

Los intentos son mensajes que se envían entre sí los distintos componentes. Se crean con objetos de la clase Intent, y se pueden inicializar de dos maneras:

1. Explícita: Activa un componente específico.
2. Implícita: Activa un tipo de componente.

Respecto a las actividades y servicios, los intentos definen una acción que debe realizarse y puede especificar el URI de los datos sobre los que se debe actuar.

***URI / Identificador de recursos universal:*** Es una cadena de caracteres que sirve para referenciar recursos en el sistema. Esta identificación permite la interacción con recursos del sistema. El método Uri.parse(String uri) sirve para analizar y codificar un String en forma de URI.

Para los receptores de broadcast, los intentos definen el anuncio que se está transmitiendo; como un mensaje de batería baja; este simplemente sería un String.

Los proveedores de contenidos no se activan por intentos, sino con peticiones de ContentResolver / resolutor de contenidos. Este trata todas las transacciones directas con el proveedor de contenido. Se emplea una capa de abstracción entre el proveedor de contenidos y el componente que solicita la información.

Métodos para activar cada componente:

1. **Actividades:** startActivity(Intent i) / startActivityForResult(Intent i, int rC) si se desea que devuelva un resultado.
2. ***Servicio:*** startService(Intent s) / bindService(Intent s, ServiceConnection c, int f) para enlazar un servicio.
3. ***Mensaje de broadcast:*** sendBroadcast(Intent i) / sendOrderedBroadcast(Intent i, String r) / sendStickyBroadcast(Intent i).
4. ***Proveedor de contenidos:*** query(Uri u, String[]x, String s, String[]sA, String so) de un ContentResolver.

Antes de que el sistema Android inicie un componente de la aplicación, este debe saber que existe. Para anunciar al sistema los componentes de la aplicación existe el archivo ***AndroidManifest.xml,*** que estará en la raíz del proyecto. Se hace un registro de todos los componentes. También contiene las siguientes funciones:

1. Identifica los permisos que se necesitan.
2. Declara los requisitos hardware y software necesarios.
3. Declara librerías de aplicación además de las API de Android.

En el elemento <application> se pueden encontrar atributos específicos. El elemento <activity> sirve para declarar una actividad.

Los componentes que no se encuentren en el documento no se podrán ejecutar, ya que no son visibles por el sistema. La tarea principal del manifiesto es informar al sistema de los componentes.

Las actividades tienen un ciclo de vida que consta de varias fases. Cada vez que se lanza una actividad, la anterior se detiene y el sistema la guarda en la ***actividad pila.*** Cuando hay un cambio de estado en la actividad, esta queda notificada a través de los métodos de callback y se liberan recursos reservados.

—------------------------------------------------------------------------------------------------------------------------

onCreate(): El sistema llama a este método cuando se crea la actividad. Dentro se inicializan los componentes de la aplicación. Debe llamar a setContentView() para definir la distribución de la interfaz de la actividad.

onPause(): El sistema llama a este método para indicar que el usuario está saliendo de la actividad, este guarda todos los cambios por si no se vuelve a abrir la aplicación.

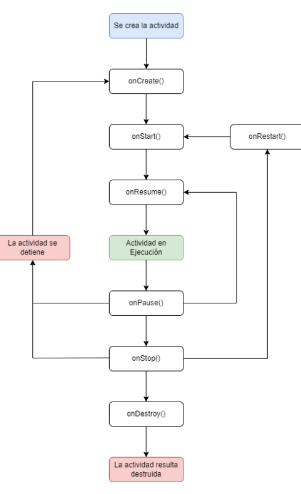
La interfaz de usuario está formada por jerarquía de vistas (objetos derivados de la clase View). Los widgets, layouts son vistas, esta última contiene más vistas con distribución específica. Los layouts están en /res/layout.

***CICLO DE VIDA DE UNA ACTIVIDAD:***

1. ***onCreate():*** Se llama cuando la actividad se ha creado por primera vez.
2. ***onStart():*** Se llama cuando la actividad es visible para el usuario.
3. ***onResume():*** Se llama cuando la actividad comienza a interaccionar con el usuario.
4. ***onPause():*** Se llama cuando la actividad actual se pausa y la actividad anterior es

reanudada.

1. ***onStop():*** Se llama cuando la actividad ya no es visible para el usuario.
2. ***onDestroy():*** Se llama antes de que la actividad sea destruida por el sistema.
3. ***onRestart():*** Se llama cuando la actividad se ha detenido y se ha vuelto a iniciar.



……………………………………………………………………………………………………………

Los ***intentos son mensajes enviados entre diferentes elementos*** que conforman las aplicaciones. Indican a una actividad que se inicie o a un servicio que empiece o se detenga, comunicando los componentes.

***Iniciar una actividad:*** startActivity(Intent i) / Activity.startActivityForResult(Intent i, int rC).

***Iniciar un servicio o enviar información a un servicio:*** Context.startService(Intent service) / Context.bindService().

***Intento para todos los receptores de broadcast:*** Context.sendBroadcast(Intent intent), Context.sendOrderedBroadcast(), Context.sendStickBroadcast(Intent intent).

***CARACTERÍSTICAS DE UN INTENTO:***

Nombre del componente: Objeto ComponentName (Es un atributo opcional). Si se especifica, el intento se envía a la instancia de la clase, en caso contrario el sistema encontrará un componente para tratarlo. (setComponent(), setClass() o setClassName()).

Acción: Es un String que especifica la acción a realizar o una acción que ha tenido lugar y se está anunciando. (setAction() y getAction())

Datos: Información relativa a los datos sobre los que se debe realizar la acción con un URI y el tipo MIME de los datos. MIME / extensiones de correo de Internet multipropósito es un estándar de Internet que se diseñó para especificar el tipo de datos.

Categoría: Es una cadena que contiene información adicional sobre el tipo de componente que va a tratar el intento. Hay 4 categorías:

1. CATEGORY\_PREFERENCE: La actividad es un panel de preferencias.
2. CATEGORY\_APP\_MUSIC: La actividad debe reproducir o manipular música.
3. CATEGORY\_APP\_MESSAGING: La aplicación debe enviar y recibir mensajes.
4. CATEGORY\_APP\_BROWSER: La actividad debe navegar por internet.

addCategory(String category), removeCategory(String category) y getCategories() para añadir, borrar y obtener una lista de las categorías.

Extras: Son pares clave-valor que aportan información adicional que debe enviarse al componente para que trate el intento. Se usan objetos de la clase Bundle para añadir extras, estos permiten almacenar una serie de datos de diferente tipo.

Flags: Indican al sistema cómo debe lanzar la actividad y cómo tratarla una vez iniciada.

***FILTROS DE INTENTOS:***

***Explícitos:*** Donde se especifica el componente por su nombre. Generalmente no conocemos el nombre específico de los componentes de otras aplicaciones, por lo que estos sólo se usan para llamar componentes de nuestra aplicación.

***Implícitos:*** Éstos no tienen definido el nombre del componente al que van dirigidos, se usan para activar componentes de otras aplicaciones. El sistema compara el contenido del intento con los posibles componentes que puedan ejecutarlo (filtros de intentos objetos de la clase IntentFilter).