También puede crear una clave de firma por línea de comandos mediante el ejecutable **keytool**.

```
[arnau@tulkass ~]$ keytool -genkey -keystore claves -alias "clave firma android" -keyalg RSA -validity 36500
Introduzca la contraseña del almacén de claves:
Volver a escribir la contraseña nueva:
¿Cuáles son su nombre y su apellido?
  [Unknown]:  Arnau
¿Cuál es el nombre de su unidad de organización?
  [Unknown]:
¿Cuál es el nombre de su organización?
  [Unknown]:
¿Cuál es el nombre de su ciudad o localidad?
  [Unknown]:
¿Cuál es el nombre de su estado o provincia?
  [Unknown]:
¿Cuál es el código de país de dos letras de la unidad?
  [Unknown]:  ES
¿Es correcto CN=Arnau, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=ES?
  [no]:  si

Introduzca la contraseña de clave para <clave firma android>
        (INTRO si es la misma contraseña que la del almacén de claves):
Volver a escribir la contraseña nueva:
[arnau@tulkass ~]$
```

En Windows, el ejecutable keytool se encuentra en la carpeta bin de la carpeta de instalación de Java.

También puede firmar aplicación a posteriori mediante el ejecutable **jarsigner**.

```
jarsigner -verbose -verify -certs miAplicacion.apk
```

Ver los siguientes documentos:

- Cómo exportar un APK en Android Studio.doc
- Formato APK.doc

Clase del 27 de Febrero 2023

Componentes Android

El framework Android se compone de algunos elementos esenciales para la creación de aplicaciones.

1. Activity (actividad)

Una actividad es el componente principal de una aplicación Android. Representa la implementación y las interacciones de sus interfaces.

Tomemos, por ejemplo, una aplicación que enumere todas las imágenes que hay en teléfono. El proyecto podría descomponerse de la siguiente manera:

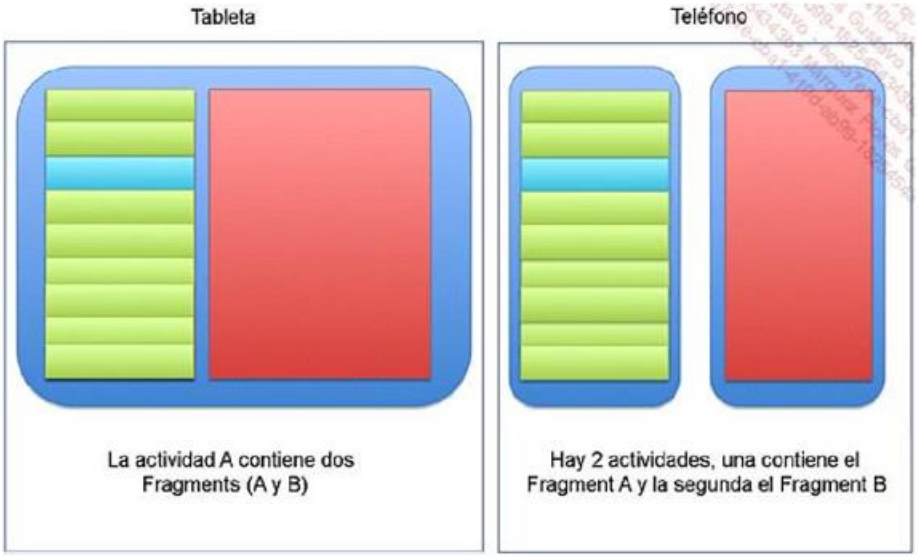
- Una vista para mostrar la lista de imágenes.
- Una actividad para implementar la interfaz y las interacciones del usuario con la lista de imágenes.

El framework Android puede decidir "matar" una actividad para liberar recursos para otras aplicaciones (sólo si actividad ya no está en primer plano) (sección Ciclo de vida de una actividad, más adelante).

2. Fragment (fragmento)

Este concepto se introdujo a partir de la versión 3.0 de Android. Permite construir interfaces más flexibles (smartphone/tableta), dinámicas y fáciles de utilizar.

Un fragmento puede considerarse como una parte de una interfaz. Por lo tanto, una interfaz (actividad) puede componerse de uno o varios fragmentos.



Tomemos una interfaz compuesta de dos vistas. En vez de implementar dos actividades distintas, se podrían utilizar los fragmentos. Esto facilita la creación de dos interfaces diferenciadas para teléfono o tableta.

- **Para teléfono**: los fragmentos se separan en dos actividades diferentes. La selección de un elemento de la actividad A (el fragmento A se incluye en la actividad A) mostrará la actividad B (el fragmento B se incluye en la actividad B).
- **Para tableta**: su interfaz se compondrá de una sola actividad, pero ésta estará separada en dos fragmentos. Cuando el usuario pulse un elemento del fragmento A, actualizará todo el contenido del fragmento B sin iniciar una actividad nueva.

7. La clase Application

Cada aplicación Android tiene una clase Application. Esta clase se mantiene instanciada a lo largo de todo el ciclo de vida de aplicación. Permite mantener el estado global de su aplicación. Puede sobrecargar esta clase para:

- Salvaguardar variables globales de aplicación.
- Gestionar los recursos que utiliza aplicación.
- Declarar constantes globales para aplicación.

Una vez se haya sobrescrito esta clase, debe declararla en archivo de manifiesto para que se tome en consideración.

```
<application android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:name=".MyApplication">
```

La clase Application se instanciará en el arranque de la aplicación.

Esta clase se programa en el archivo manifiesto.

El ciclo de vida de una actividad se refiere a las etapas o estados por las cuales pasa una aplicación en Android (actividad) . Por ejemplo: en ejecución, en espera, terminada.

Ciclo de vida de una actividad

Por defecto, cada aplicación Android se ejecuta en un proceso separado. Android gestiona los recursos disponibles en el dispositivo y puede, si fuera necesario, cerrar aplicaciones para liberar recursos (excepto aplicaciones en ejecución).

La elección de la aplicación que se cerrará depende fundamentalmente del estado del proceso en el que se encuentra. Si Android debe elegir entre dos aplicaciones que se encuentran en el mismo estado, elegirá la que se encuentre en este estado desde hace más tiempo.

1. Estado de una actividad

Una actividad puede encontrarse en cuatro estados distintos:

- **En ejecución**: la actividad se encuentra en primer plano y recibe las interacciones del usuario. Si el dispositivo necesita recursos, se matará la actividad que se encuentra en el fondo del back stack (Back stack).
- **Pausada**: la actividad está visible pero el usuario no puede interactuar con ella (oculta por un cuadro de diálogo, por ejemplo). La única diferencia con el estado anterior es la no recepción de eventos de usuario.

- **Parada**: la actividad ya no es visible pero sigue en ejecución. Todos los datos relativos a su ejecución se conservan en memoria. Cuando una actividad pasa a estar parada, debe guardar los datos importantes y detener todos los tratamientos en ejecución.
- **Muerta**: se ha matado la actividad, ya no está en ejecución y desaparece de la **back stack**. Todos los datos presentes en caché que no se hayan guardado se pierden.

2. Back stack

Como ocurre con un sitio de Internet, en una pantalla de Android se muestra una vista cada vez. La sucesión de páginas se almacena en una pila llamada back stack, lo que permite volver a la página anterior cuando se utiliza el botón "volver" del teléfono (similar al botón de volver atrás en un navegador web).

Todas las actividades iniciadas se almacenan en la **back stack**. Cuando se inicia una actividad nueva, ésta se añade en la cima de la back stack.

Si el usuario pulsa el botón "atrás" de dispositivo, la actividad en curso se cierra y la que se encontraba inmediatamente debajo en la back stack se abre.

Dependiendo del estado en que se encuentre la aplicación, se ejecuta determinado método

3. Ciclo de vida

El ciclo de vida de una actividad es bastante complejo y su comprensión es indispensable en el desarrollo para Android. El siguiente esquema resume este ciclo de vida:

Ver el siguiente documento:

El ciclo de vida de una actividad en Android.doc

Cuando se inicia la actividad, se invoca al método **onCreate**. En este método, debe inicializar vista y vincular los datos a una lista. Este método recibe como parámetro un Bundle que contiene el estado anterior de la actividad.

A esta llamada le sigue el método **onStart**, que permite indicar el inicio efectivo de la aplicación (ahora ya es visible).

A continuación, se invoca al método **onResume** para ejecutar todos los tratamientos necesarios para el funcionamiento de la actividad (thread, proceso, tratamiento), inicializar variables y listeners. Estos tratamientos deberán detenerse cuando se invoque al método **onPause** y relanzarse, si fuera necesario, cuando se produzca una futura llamada al método **onResume**.

Después de estas tres llamadas, la actividad se considera utilizable y puede recibir interacciones de los usuarios.

Si otra actividad pasa a primer plano, la actividad en ejecución pasará a estar pausada. Justo antes de la llamada al método **onPause**, se invocará al método **onSaveInstanceState** para permitirle guardar los datos importantes de la actividad. Estos datos podrán aplicarse a futuras ejecuciones de la actividad con la llamada al método **onRestoreInstanceState** o la llamada a **onCreate**.

El método **onPause** permite detener los tratamientos realizados (tratamiento no necesario si la actividad no es visible) por la actividad (tratamiento, thread, proceso).

Si actividad pasa a estar visible de nuevo, se realizará una llamada al método **onResume**.

El paso de la actividad al estado "parada" tiene asociado una llamada al método **onStop**. En este método hay que detener todos los tratamientos restantes.

Una vez parada, actividad puede:

- **Volver a iniciarse**: acción realizada por una llamada al método **onStart** siguiendo el ciclo de vida normal de la actividad.
- **Terminar**: se realiza con una llamada al método **onDestroy**, en el que deberá parar todos los tratamientos restantes, cerrar todas las conexiones a la base de datos, todos los threads, todos los archivos abiertos, etc. Puede provocar la llamada al método **onDestroy** utilizando el método **finish**.

Es posible sobrecargar todos los métodos del ciclo de vida de una actividad.

Manifiesto

El archivo **AndroidManifest.xml**, que se encuentra en la raíz de cualquier proyecto Android, es lo más parecido a una descripción completa de la aplicación (componentes, actividades, servicios, permisos, proveedores de contenido, etc.).

Vamos a describir las secciones generales e indispensables de un archivo de manifiesto. El comienzo de un archivo de manifiesto es, con algunos detalles, casi idéntico al de cualquier otra aplicación.

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

Esta primera parte contiene:

- La declaración xmlns (obligatoria).
- El package (identificador) de aplicación (elegido en la creación de proyecto).

Si utiliza Eclipse, la declaración de una aplicación (versionName y versionCode) se realiza en el archivo de manifiesto de la aplicación.

El contenido del manifiesto varía en función de la aplicación.

El manifiesto contiene varias secciones (tags).

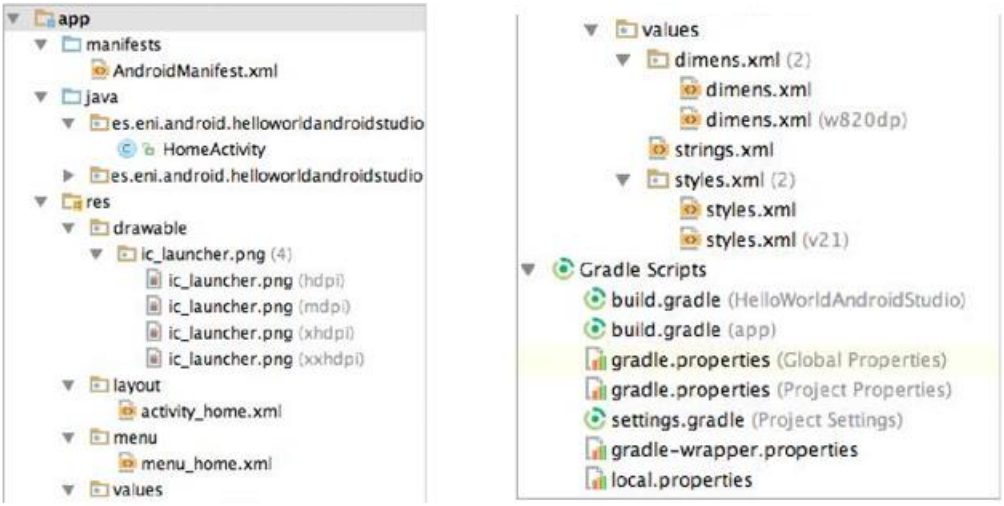
- **La aplicación**: define la estructura y el contenido de la aplicación (actividad, servicio, etc.).

En la sección application del manifiesto tendrá que describir todas las características específicas y todos los componentes de aplicación:

- Nombre de la aplicación.
- Icono de la aplicación.
- Tema utilizado por la aplicación.
- Los distintos componentes de la aplicación (Actividad, Servicio, Proveedor de contenido, Receptor de eventos) así como las características específicas de cada componente (componente principal, presencia en el inicio de la aplicación, uso de datos).

Arquitectura del proyecto

Debería obtener la siguiente arquitectura en Android Studio:



Una aplicación Android en Android Studio (vista **Android**) se compone de varias carpetas:

- **La carpeta raíz**: representa aplicación (llamada, generalmente, "app").
- **La carpeta manifests**: contiene el archivo de manifiesto de aplicación (archivo que describe aplicación).
- **java**: contiene dos sub-packages (paquetes), el primero representa el código fuente de aplicación y el segundo el código de pruebas.
  - **Paquete fuente de su aplicación**: tiene, en general, el identificador que se ha indicado durante la creación del proyecto seguido del nombre de la aplicación. Esta carpeta contiene el código fuente de aplicación (actividad, servicio, código de negocio, etc.).
- **res**: esta carpeta sirve para almacenar todos los recursos que utilice en aplicación. Puede tener recursos diferentes en función de la resolución, de la orientación o del idioma del dispositivo. Los recursos se precompilan y se añaden al archivo **R.java** (Creación de interfaces sencillas - Recursos).
  - **drawable**: contiene todas las imágenes en las distintas resoluciones aceptadas.
  - **layout**: contiene todas las vistas (modos horizontal, vertical y tableta).
  - **values**: contiene todos los archivos que contienen datos o valores (strings, arrays, colors, dimensions, styles, etc.).

    - **menu**: contiene las barras de acciones de aplicación.

- **Gradle Scripts**: contiene los scripts y las configuraciones Gradle de proyecto.

Estas carpetas se corresponden con la vista **Android** del proyecto, pero puede cambiar el tipo de visualización del proyecto mediante el menú situado sobre su arquitectura de proyecto.