EL CICLO DE VIDA DE UNA "ACTIVITY" O ACTIVIDAD EN ANDROID

Diplomado en desarrollo de aplicaciones moviles

Universidad de Córdoba
Facultad de Ingenierías
Dpto. de Ing. de Sistemas
2016

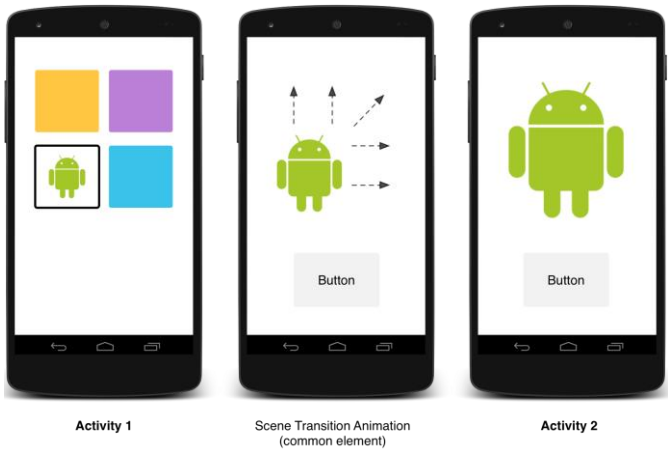
Consideraciones para el desarrollo

- ✓ Pequeña capacidad de procesamiento
- ✓ Memoria RAM limitada
- ✓ Memoria permanente de poca capacidad
- ✓ Pantallas pequeñas de poca resolución
- ✓ Transferencias de datos costosa (en términos de energía y económicos) y lenta
- ✓ Inestabilidad de las conexiones de datos
- ✓ Batería muy limitada
- ✓ Necesidad de terminar la aplicación en cualquier momento

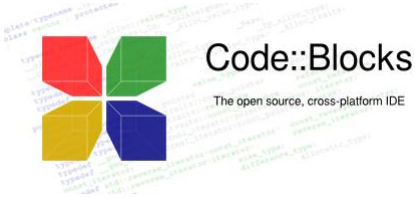
Consideraciones para el desarrollo

- ✓ Ser eficiente
  - ❑ CPU
  - ❑ Memoria
  - ❑ Recursos y red
- ✓ Respetar al usuario
  - ❑ No robar el foco
  - ❑ Pocos avisos
  - ❑ Interfaz intuitiva y coherente con Android

El ciclo de vida de una "activity" o actividad en Android es de lo primero que debemos conocer si queremos hacer el uso correcto de la misma en nuestra aplicación.



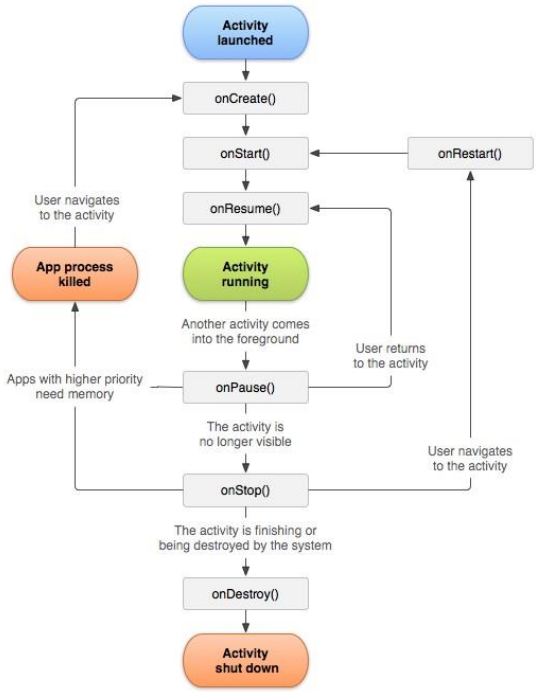
A diferencia de muchos sistemas o aplicaciones que se inician con un método "main()", que si vienes del mundo Java o C++ me imagino que te sonará



Activity lifecycle es el ciclo de vida de una **activity**.

- ✓ Describe los estados y transiciones en los que puede encontrarse una *activity*.
- ✓ Una *Activity* es un "objeto" cuya finalidad es hacer lo que el usuario le indica
- ✓ Activity: tarea destinada a mostrar una interfaz gráfica al usuario.
- ✓ Sólo podemos ver en pantalla una actividad a la vez.
- ✓ Una aplicación suele estructurarse en un conjunto de actividades.
- ✓ Una aplicación puede mostrar actividades de otras aplicaciones o actividades nativas del sistema (por ejemplo, la de enviar SMS).

Organigrama fundamental del ciclo de una activity



Estados de las activities:

Una *activity*, puede encontrarse en tres estados:

- ✓ **Running** (Ejecutándose) Interactúa con el usuario.
- ✓ **Paused** (Pausado) Cuando la *activity* está todavía en la pantalla, pero en estado latente, pausada y oscurecida, bien porque es otra actividad la que está en ejecución o porque aparece un diálogo. También ocurre cuando la pantalla del Smartphone o Tablet, se encuentra bloqueada. A la hora de programar, es importante tenerlo en cuenta, ya que puede eliminarse en cualquier momento, por ejemplo si Android se encuentra con problemas de memoria. En este estado, la *activity* todavía se encuentra viva, en la **heap memory** de la máquina virtual, esperando volver al estado Running.
- ✓ **Stopped** (Detenido) Cuando la *activity* está completamente oscurecida por otra *activity*.

Estados de las activities:

Cuando se pulsa la tecla Home y la *activity* se encuentra en estado *Paused* o *Stopped*, **Android** puede decidir eliminar la *activity*, informando a la propia *activity* o matando directamente el proceso. Hasta que no sea totalmente eliminada, una *activity* que se encuentre en estos estados, pueden volver al estado Running.

Métodos protegidos

Las *activities* cuentan con métodos protegidos para estar informados de los cambios de estado. Estos métodos protegidos, se pueden sobreescribir para estar informados de los cambios de una *activity*.

- ✓ **oncreate()** Este método es llamado cuando la *activity* es iniciada por primera vez.
- ✓ **onStart()** Llamado a continuación del método *oncreate().* También puede llamarse cuando la *activity* es recomenzada desde *stopped*.
- ✓ **onRestart()** Se recupera una *activity* que se encuentra en estado Stopped (Detenida) y se devuelve al primer plano *onStart*()
- ✓ **onResume**() Llamado después de onStart o cuando la activity es recomenzada desde Paused

Métodos protegidos

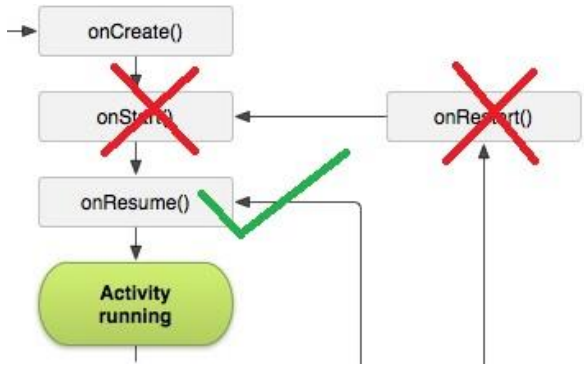
- ✓ **onPause()** Llamado cuando la *activity* entra en el estado pausado. Puede ser el último estado. Debido a ello, es conveniente salvar cualquier estado existente aquí.

- ✓ **onStop()** Cuando la *activity* entra en el estado detenido. Precedido por onPause(). Por lo tanto una *activity*, antes de estar detenida, debe estar pausada.

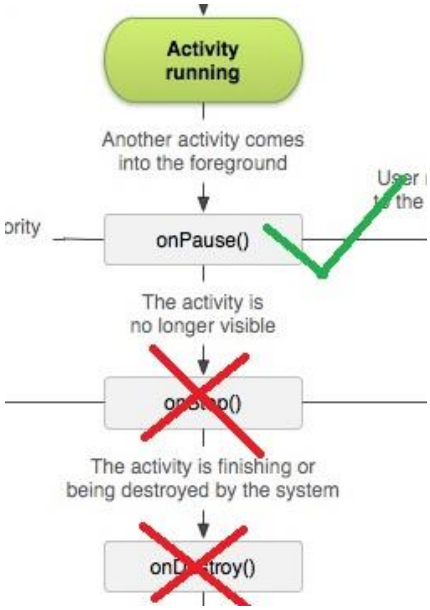
- ✓ **onDestroy()** Llamado al final del ciclo de vida de la *Activity*. Es destruida de forma irreversible.

Hay que tener en cuenta unas ideas principales sobre el ciclo de vida de la Activity:

El método **onResume()** es llamado siempre, antes de que la **activity** entre en el estado **Running**. Por lo tanto, podemos ignorar los métodos **onStart()** y **onRestart().** No importa si la **activity** se ha recuperado de un estado detenido o pausado. Lo importante es saber que la **activity** se está ejecutando (Running). Si observas el organigrama principal, verás que no hay forma de llegar a "**Activity Running**", ejecución de la activity sin pasar por **onResume()**

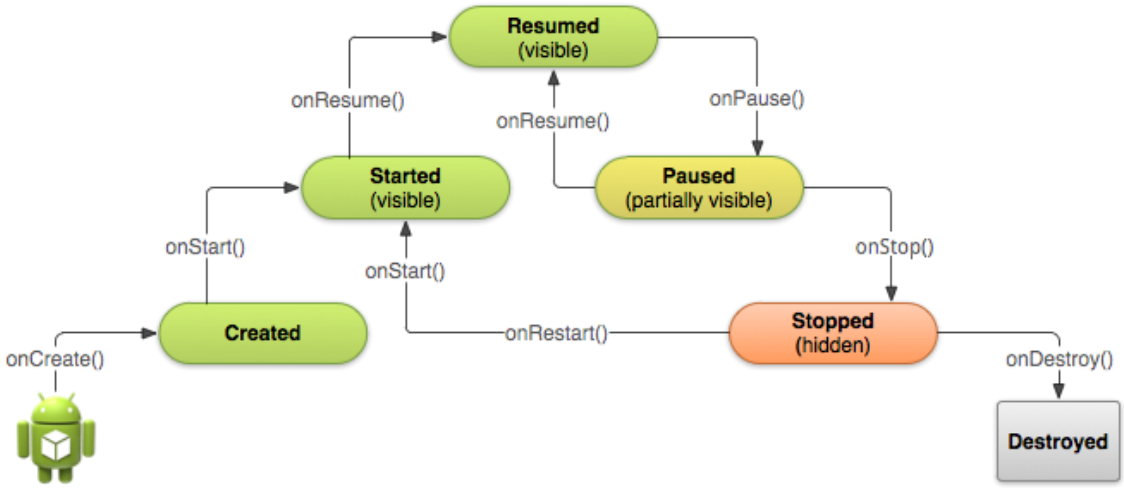


La **Activity** puede ser destruida silenciosamente después de **onPause()**. Por lo tanto, podemos ignorar **onStop()** y **onDestroy()** y guardar los datos en **onPause()** (por ejemplo, puntuaciones más altas o nivel de progreso)



Por lo tanto, se puede decir que los métodos más importantes son *onCreate(), onResume() y onPause().* Con cada uno de ellos ejecutaremos distintas acciones:

- ✓ ***onCreate()*** Con este método, se crea la ventana, el elemento **UI** y los **Inputs** de la aplicación.
- ✓ ***onResume()*** Se inicia o reinicia el hilo **loop Main,** que permite que esta se ejecute.
- ✓ ***onPause()*** Se detiene el hilo *loop Main. Si isFinishing()* devuelve *true*, te permite guardar cualquier estado que quieras que persista al disco para que no se pierda



El archivo AndroidManifest.xml

El archivo más importante en cada proyecto y que deberá de estar presente en la raíz del proyecto: **AndroidManifest.xml**.

Dicho archivo es generado automáticamente y modificable gráficamente o programando. Y por lo tanto es importante conocerlo. Ya que el archivo presenta información esencial sobre la aplicación al sistema operativo Android. Información necesaria para que pueda ejecutar la aplicación.

Principales tareas del AndroidManifest.xml

- ✓ Utiliza el nombre de **paquete Java como identificador** único de la aplicación.
- ✓ Describe los **componentes de la aplicación**: Actividades, servicios, proveedores de contenido... Para ello utiliza el nombre de las clases que implementan cada uno de estos componentes y publica sus capacidades. Esto permite al sistema operativo conocer que componentes tiene y bajo que condiciones pueden ser lanzados.
- ✓ Especifica que **permisos tiene la aplicación** para acceder a partes protegidas del API que proporciona el sistema Android.
- ✓ Declara la **mínima versión del sistema operativo** en el que funcionará la aplicación.
- ✓ Indica las **librerías** que utiliza el proyecto y por lo tanto tienen que ser empaquetadas al crear la aplicación.
- ✓ Permite declarar una clase 'Instrumentation' que sirve para **monitorizar** la interacción de la aplicación con el sistema. Esta declaración solo estará presente mientras se desarrolla y prueba la aplicación. Ya que será borrada antes de que la aplicación se vaya a publicar.

Ejemplo básico AndroidManifest.xml

Cosas a tener en cuenta:

- ✓ El elemento padre (manifest) del archivo debe de contener una declaración del espacio de nombres y del nombre que asigna al paquete que forma la aplicación.
- ✓ Cada manifiesto incluye un único elemento que contendrá información básica para la aplicación como el icono, nombre o tema que utiliza.
- ✓ Cada una de las actividades (controladores de cada pantalla de la interfaz) que aparecerán en la aplicación deberán de aparecer en el manifiesto.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.loginandregisterapi"
  android:versionCode="1"
  android:versionName="1.0" >
  <uses-sdk
    android:minSdkVersion="10"
    android:targetSdkVersion="17" />
  <uses-permission android:name="android.permission.INTERNET"/>
  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
      android:name="com.example.loginandregisterapi.LoginActivity"
      android:label="@string/app_name" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Descripción de los elementos básicos:

1. Etiqueta manifest

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.loginandregisterapi"
  android:versionCode="1"
  android:versionName="1.0" >
```

Este el elemento raíz del manifiesto y sus dos atributos principales y obligatorios son:

- ✓ xmlns:android: Define el espacio de nombres de Android y siempre debe de ser el mismo
- ✓ package: El nombre del paquete define la aplicación y actúa como identificador único de la misma.

Por lo que si has publicado una aplicación con un nombre y luego se cambia, los usuarios de la primera versión no podrán actualizar a la siguiente.

Descripción de los elementos básicos:

2. Etiqueta uses-sdk

```
<uses-sdk
  android:minSdkVersion="10"
  android:targetSdkVersion="17" />
```

Elemento (etiqueta) de segundo nivel obligatorio que determina la compatibilidad de la aplicación con una o más versiones del sistema operativo. Esta compatibilidad viene expresada en base al nivel de API del sistema Android que soporta. Sus dos atributos principales son:

- ✓ android:minSdkVersion (obligatorio): Determina el mínimo nivel de API que debe de tener el sistema operativo Android que pretenda ejecutar la aplicación. El sistema evitará que la aplicación se instale en un sistema que tenga un nivel de API inferior del especificado.
- ✓ android:targetSdkVersion (opcional): Si no se especifica se toma el valor de minSdkVersion. Determina el nivel del API con el que fue construida la aplicación. Por lo que se espera que tome ventajas del nivel de API especificado pero es totalmente retro-compatible con versiones antiguas hasta la indicada mediante minSdkVersion.

Descripción de los elementos básicos:

3. Etiqueta uses-permission

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Etiqueta opcional de segundo nivel que sirve para indicar un permiso necesario que requiere la aplicación para acceder a alguna parte protegida del API que proporciona el sistema Android. Esta declaración alertará a los usuarios que la aplicación utilizará ciertos permisos. Evidentemente crearemos tantas etiquetas como permisos necesitemos. El único atributo disponible y obligatorio es android:name. El cual indica un permiso que necesita la aplicación.

Para ver la lista total de permisos podemos recurrir a la documentación de Google. http://developer.android.com/reference/android/Manifest.permission.html

Descripción de los elementos básicos:

4. Etiqueta application

```
<application
  android:allowBackup="true"
  android:icon="@drawable/ic_launcher"
  android:label="@string/app_name"
  android:theme="@style/AppTheme" >
```

Etiqueta de segundo nivel que contendrá otras etiquetas que declararán cada uno de los componentes de la aplicación. Además permite atributos que pueden afectar a todos los citados componentes de la aplicación. Solo se puede declarar una vez este elemento en el manifiesto. Y admite multitud de atributos aunque los más importantes y utilizados son los siguientes:

- ✓ **android:allowBackup**(opcional). Valor por defecto a true. Indica que si se hace un backup del sistema Android, las aplicaciones que tengan este valor a true se guarden junto con el backup del sistema.
- ✓ **android:description**: Descripción larga de la aplicación y sus funcionalidades.
- ✓ **android:icon**: identificador del recurso que será el icono de la aplicación.
- ✓ **android:label**. Identificador de la cadena de texto que dará nombre a la aplicación y que será el que verá el usuario en el sistema operativo.
- ✓ **android:permission.** Especificamos el nombre de un permiso que será necesario si otras aplicaciones hacen uso de partes de tu aplicación. También podemos definir el permiso necesario para cada una de las actividades (siguiente etiqueta) utilizando este atributo en cada una de ella.
- ✓ **android:theme.** Identificador al recurso que especifica el tema por defecto de todas las actividades de la aplicación. Las actividades pueden sobreescribir individualmente el tema general con sus respectivos atributos android:theme.

Descripción de los elementos básicos:

5. Etiqueta activity

```
<activity
  android:name="com.example.loginandregisterapi.LoginActivity"
  android:label="@string/app_name" >
</activity>
```

Etiqueta de tercer nivel y que es uno de los sub-elementos de la etiqueta application. Una actividad es el controlador que va a interactuar con una pantalla de la interfaz gráfica. Y por lo tanto debemos de especificar cada actividad del proyecto con su etiqueta activity correspondiente. Si una actividad no esta especificada en el manifiesto, esta no podrá lanzarse.

Descripción de los elementos básicos:

6. Etiqueta intent-filter

```
<activity
  android:name="com.example.loginandregisterapi.LoginActivity"
  android:label="@string/app_name" >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

Etiqueta que se sitúa como sub-elemento de la anterior etiqueta (activity). Esta etiqueta sirve para agrupar el número de acciones que concretarán el ámbito en el que se va a ejecutar la actividad. Las actividades pueden declarar el tipo de acciones que pueden llevar a cabo y los tipos de datos que pueden gestiona.

Descripción de los elementos básicos:

7. Etiqueta action

```
<activity
  android:name="com.example.loginandregisterapi.LoginActivity"
  android:label="@string/app_name" >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

Una acción que el 'intent-filter' soporta. Las acciones son cadenas de texto estándar que describen lo que que la actividad puede hacer. El único y obligatorio atributo es android:name. En el cual indicaremos el nombre de la acción. En el anterior ejemplo, indicamos que esta es la actividad principal de la aplicación y por lo tanto la que controlará el inicio de la aplicación

Descripción de los elementos básicos:

8. Etiqueta category

Básicamente indica si la actividad va a ser lanzada desde el lanzador de aplicaciones, desde el menú de otra aplicación, directamente desde otra actividad...

Lista de categorías:
http://developer.android.com/reference/android/content/Intent.html

9. Etiqueta data

Mediante esta etiqueta añadiremos una especificación de datos para las acciones especificadas. Puede ser un tipo de datos o una URI.

(identificador de recursos uniforme o URI —del inglés uniform resource identifier—)

GRACIAS !!!