

UD 03

CONTROL DE FLUJO DE ACTIVIDADES

CONCEPTOS BÁSICOS SOBRE LA ESTRUCTURA DE LA APLICACIÓN

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES 22/23

CFGS DAM

Autor: Carlos Espinosa
c.espinosamoreno@edu.gva.es
Fecha: 2022/2023
Licencia Creative Commons
versión 4.0



Reconocimiento – NoComercial – CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Índice

1.Objetivos de la unidad.....	1
2.Introducción	2
3.Introducción a las Activities	2
4.Flujo de actividades	2
5.Conjunto de estados de una aplicación: Ciclo de vida	4
6.Comunicación entre actividades.....	5
7.La Barra de Herramientas (TOOL BAR)	5
8.Bibliografía	6

1. OBJETIVOS DE LA UNIDAD

- Conocer la clase Activity
- Comprender el ciclo de vida de una aplicación de Android y sus métodos asociados.
- Conocer el funcionamiento de la pila en Android (Back Stack)
- Valorar el uso de una librería en Android
- Conocer los “intents” y trabajar con ellos y con sus filtros.
- Conocer los services, sus diferentes tipos y trabajar con ellos.

2. INTRODUCCIÓN

En esta unidad de aprendizaje vamos a continuar conociendo la forma de crear una aplicación en AndroidStudio. Para ello, es imprescindible conocer el flujo de una aplicación desarrollada en Android Studio.

Como recomendación, antes de comenzar con el apartado de prácticas/actividades estudia a fondo la teoría y asegúrate de comprender la lógica de lo que aquí se explica. En aulas se dejarán también recursos adicionales como URL's interesantes que seguro son buenos complementos cuando estés programando.

3. INTRODUCCIÓN A LAS ACTIVITIES

Una “Activity” es el principal componente de una aplicación de Android y gestionará una parte importante de las interacciones del usuario. A nivel de programación podemos decir que cada “Activity” se traduce en una pantalla, así pues, nuestra aplicación tendrá tantas pantallas como “activities” y viceversa.

Toda actividad debe tener una vista asociada, que se usará como interfaz. La vista puede ser simple, pero suele ser de tipo layout.

Cuando creamos una app, la primera actividad se crea de forma automática, pero el resto debe crearse. Estos son los pasos:

1. Crear el **layout** de la aplicación
2. Crear una nueva clase descendiente de **Activity**, indicando que se visualizará el anterior layout.
3. Activar la aplicación desde otra actividad.
4. Registrarlo todo en **AndroidManifest.xml**

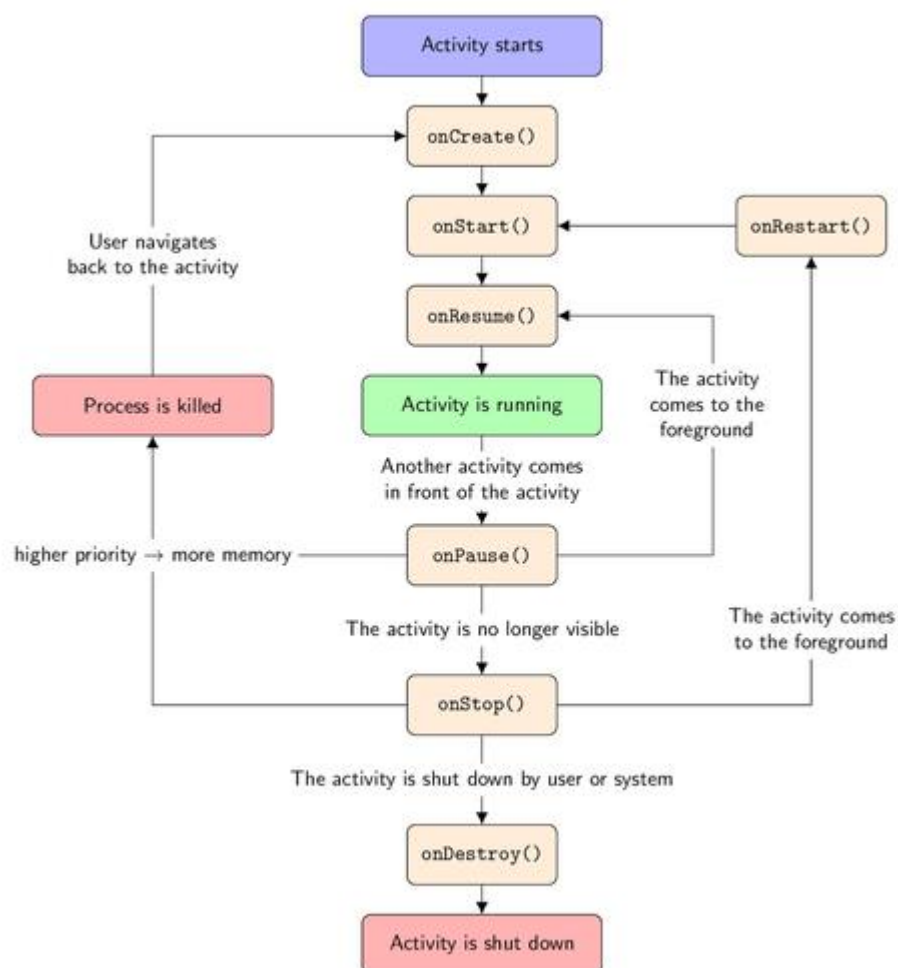
4. FLUJO DE ACTIVIDADES

A la hora de programar nuestra aplicación deberemos definir el flujo de nuestras actividades, lo cual fijará la lógica de su funcionamiento. Desde la pantalla principal, podremos navegar por el resto de pantallas, siendo el dispositivo el que “recuerda” las actividades previas, permitiéndonos navegar mediante el retroceso de pantallas.

Android almacena la posición de cada una de las pantallas en una pila (STACK) de tipo LIFO (Last In, First Out). El Stack funciona de tal manera que cuando se inicia otra actividad, el sistema se centra en ella y la anterior va a la pila, quedando en un estado de “hibernación” (para que cuando el usuario vuelva hacia atrás, el foco se cambie a esta y la que tenía el foco se elimine). En la pila podemos encontrar varias actividades y al conjunto de estas actividades que guardan una relación entre ellas se les denomina TASK.

Importante: Puede pasar que, si se requieren muchos recursos, la actividad se guarde en el su lugar en la pila pero que no tenga contenido, por lo que el sistema debe crearla de nuevo. Para evitar las pérdidas en este caso se implementa *onSaveInstanceState()*

A continuación, tenemos un esquema general de una aplicación basada en diferentes Activities:



Como se puede ver en la imagen anterior, dentro del ciclo de vida de la aplicación hay diferentes métodos y es necesario conocerlos y saber manejarlos para lograr robustez en nuestra aplicación. Los métodos más importantes son:

- **onCreate()** : es el método de la primera invocación a la actividad. Se crean las vistas, es decir la interfaz y se hace la estructura de lo que será la actividad, obteniendo las referencias mediante `findViewById()`, reservándose la memoria, etc.
- **onStart()** : Sucede a `onCreate()`. Aquí solemos ubicar las instrucciones asociadas a nuestra interfaz. La actividad es visible pero NO es interactiva. Debe seguir con `onResume()`.
- **onResume()** : Sucede a `onStart()` y precede a `onPause()`, durante este método se activan los sensores, se actualiza la info, etc. Es decir, es el método en el que una actividad es interactiva.
- **onPause()** : En este método la actividad es parcialmente visible y desde aquí puede volver a primer plano mediante `onResume()` o hacerse invisible mediante `onStop()`. Es el método en el que está una actividad que pasa a segundo plano.
- **onStop()** : Sirve para pausar cosas minimizando los recursos que está utilizando. Desde `onStop()` podemos pasar a `onRestart()` si queremos reactivarla o a `onDestroy()` si queremos que sea eliminada.
- **onRestart()** : Va SIEMPRE seguido de un `onStart()`, y se llama cuando una actividad va a volver a estar activa.
- **onDestroy()** : Sirve para limpiar y liberar recursos en el dispositivo. Se llama cuando la actividad se destruye definitivamente y se elimina de la pila.

```
import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        Log.i(tag: "cicloVida", msg: "Ingresa a onCreate()")
    }

    override fun onStart() {
        super.onStart()
        Log.i(tag: "cicloVida", msg: "Ingresa a onStart()")
    }

    override fun onResume() {
        super.onResume()
        Log.i(tag: "cicloVida", msg: "Ingresa a onResume()")
    }
}
```

5. CONJUNTO DE ESTADOS DE UNA APLICACIÓN: CICLO DE VIDA

A diferencia de otros sistemas operativos, en Android cada aplicación es un proceso separado, el usuario no decide cuándo se finaliza una aplicación, esta decisión reside en la gestión de la pila de aplicaciones del propio sistema Android, en función de las necesidades de memoria en cada momento. Del mismo modo, cuando el usuario vuelve a solicitar la aplicación, el sistema se encarga de crear el proceso para mostrar la aplicación en pantalla.

Dependiendo del estado en el cual se encuentre el proceso, el sistema asigna una prioridad que determina la probabilidad de que la aplicación sea cerrada. Podemos verlo en la siguiente tabla:

Prioridad	Estado de proceso	Estado de la actividad
Baja o nula	Primer plano	Created, Started, Resumed
Media	Segundo plano	Paused
Alta	Segundo plano no visible	Stopped, Destroyed

Es muy importante comprender el ciclo de vida de una aplicación, para crear aplicaciones robustas, durante el desarrollo debemos ejecutar el código en el momento indicado para que la aplicación funcione correctamente

Como hemos visto, una actividad puede tener diferentes estados, esto es, periodos de funcionamiento, que dependen de la interacción del usuario, de las necesidades de recursos que tenga el sistema y del flujo de la propia aplicación. Al conjunto de estados en los que puede estar una app se le llama “Ciclo de vida”. Los diferentes estados que puede tener una app son:

Activa (Running): La actividad es visible y tiene el foco.

- Visible (Paused): La actividad es visible pero no tiene el foco. Sería cuando otra actividad con transparencias o que no ocupa toda la pantalla está activa.
- Parada (Stopped): La actividad no es visible. Se guardará el estado de la interfaz de usuario y las preferencias.
- Destruída (Destroyed): La actividad termina → finish(), o es matada por el sistema. Los cambios de estado se controlan con métodos de tipo callback.

6. COMUNICACIÓN ENTRE ACTIVIDADES

Una clase **Activity** es un componente clave de una app para Android, y la forma en que se inician y se crean las actividades es una parte fundamental del modelo de aplicación de la plataforma.

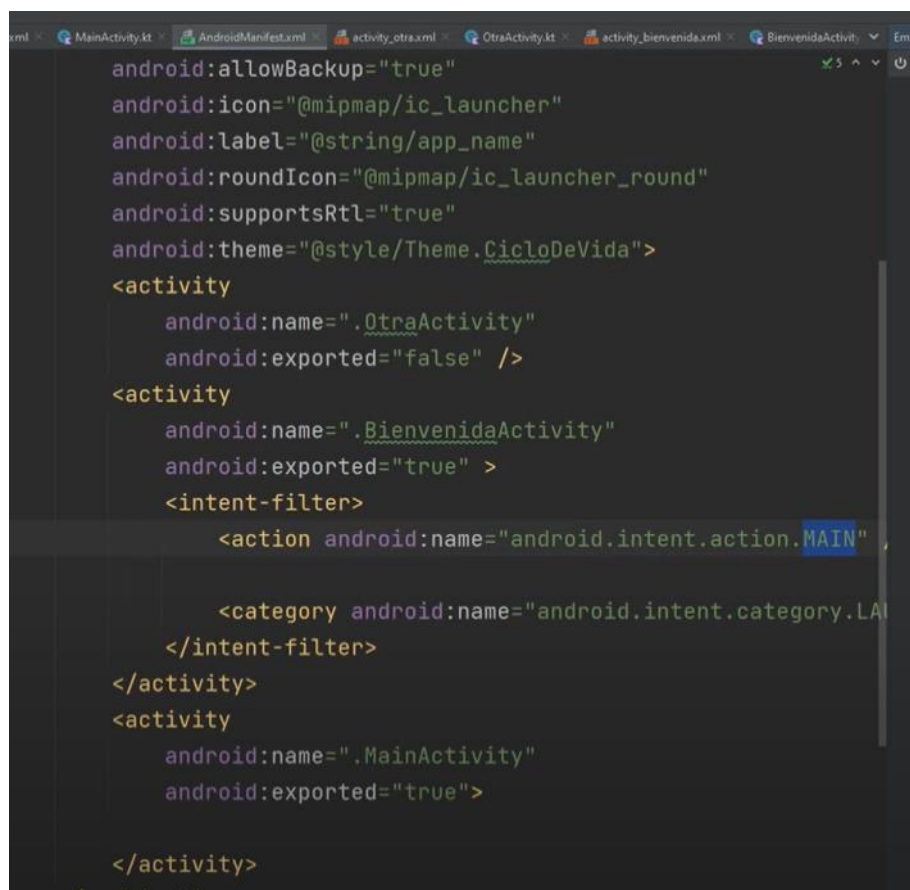
A diferencia de los paradigmas de programación en los que las apps se inician con un método main(), el sistema Android inicia el código en una instancia de Activity invocando métodos de devolución de llamada específicos que corresponden a etapas específicas de su ciclo de vida.

Las Activities son independientes entre sí por ello se gestionan mediante el ciclo de vida comentado anteriormente que donde se controla el comportamiento de la actividad desde que se inicia la actividad hasta que termina. Estos estados van cambiando según la interacción con esta actividad, este funcionamiento interno y no es un elemento que se deba implementar pero si conviene utilizarlo y saber su funcionamiento para determinados aspectos de nuestra app, como por ejemplo, el uso de recursos.

Para gestionar qué actividad se utilizará mediante su declaración en **Android Manifest**. Este es un archivo de configuración que nos va a permitir parametrizar información de nuestro sistema. Las declaramos ahí y mediante filtros vamos a indicar que Activity se va a lanzar en cada momento.

Información general declarada en el Android Manifest:

- Icono del proyecto
- el nombre de la App
- paquete donde se encuentran las funciones principales
- Las Actividades
- Servicios o providers, permisos



```
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/Theme.CicloDeVida">
<activity
    android:name=".OtraActivity"
    android:exported="false" />
<activity
    android:name=".BienvenidaActivity"
    android:exported="true" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" ,
            <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".MainActivity"
    android:exported="true">
</activity>
</manifest>
```

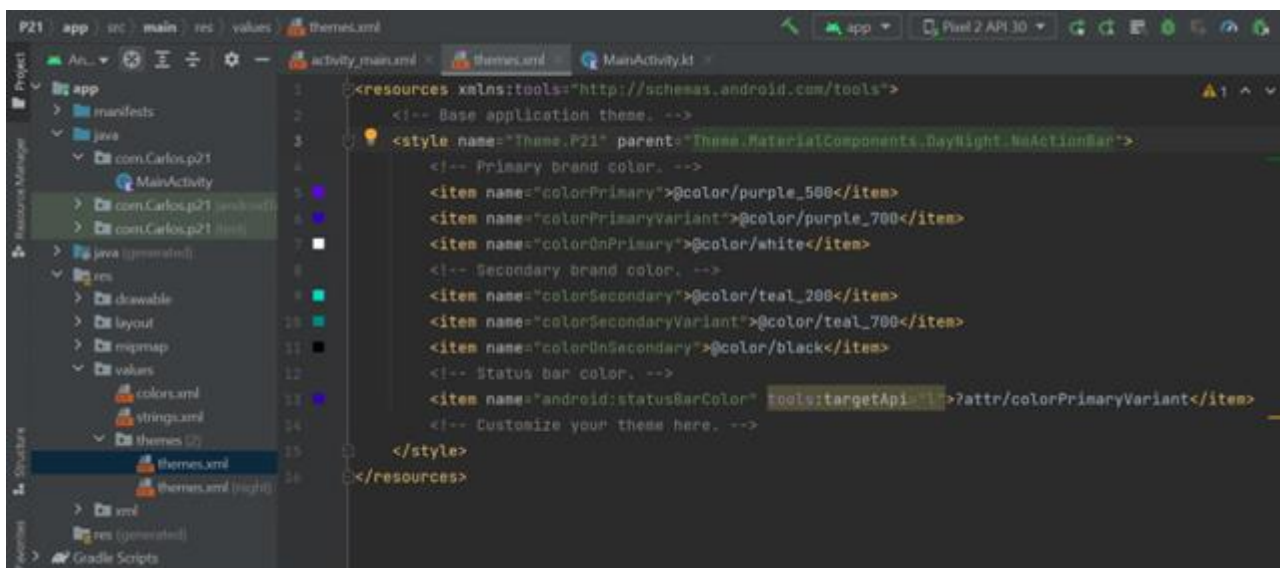
7. LA BARRA DE HERRAMIENTAS (TOOL BAR)

La barra de herramientas es un componente de barra de acciones que nos permite tener más flexibilidad en nuestra aplicación, ya que puede colocarse en cualquier lugar, puede cambiarse su estilo (color, tamaño, etc) y permite que cada Activity tenga su propia toolbar (incluso varias).

Esta opción no estará disponible en versiones de Android anteriores a Android 3.0. Dentro de la barra de acciones, encontramos elementos principales que suelen estar siempre visibles, como por ejemplo el nombre de nuestra aplicación y el de desplegar el menú y otros que podemos desplegar desde el botón Overflow (el de los 3 puntos en vertical).

Para añadir una Toolbar podemos hacerlo mediante dos clases:

- **ActionBar:** Aparece por defecto en todas las Actividades, si queremos que no se muestre hay que asignarle un tema que acabe en .NoActionBar. ActionBar está disponible desde Android3.0
- **ToolBar:** está disponible desde Android5.0 y cambia el diseño para adaptarse a Material Design, pero puede usarse en otras versiones. En este caso, la barra no se incorpora de forma automática si no que hay que incluirla en el layout con , lo que hace que nosotros decidamos su posición.



8. BIBLIOGRAFÍA

- [1] Android. Programación Multimedia y de dispositivos móviles. Garceta.
- [2] Programación Multimedia y Dispositivos Móviles. Editorial Síntesis
- [3] Android App Development. For Dummies
- [4] Beginning Android Programming with Android Studio. Wrox
- [5] Java Programming for Android Developers. For Dummies
- [6] Android Programming with Android Studio
- [7] <https://academiaandroid.com/>
- [8] <https://developer.android.com/>
- [9] <https://www.redhat.com>
- [10] <https://abhiandroid.com/>
- [11] <https://desarrolloweb.com>