Entendiendo una Aplicación en Android

Capitulo 1, segunda parte.

Agenda

- Componentes de una Aplicación Android
- Ciclo de vida de una Aplicación Android.
- Archivo Android Manifest.
- Ejecución de una Aplicación Móvil
  - Activity
  - Fragments
  - Content Providers
  - Broadcast Provider
  - Services
- Ciclo de Vida de una Aplicación Android
- Archivo Android Manifest XML
  - Función del Android Manifest
  - Elementos del Android Manifest
- Ejecución de una Aplicación Móvil
  - En el Emulador
  - En el Dispositivo

Componentes de una Aplicación Android

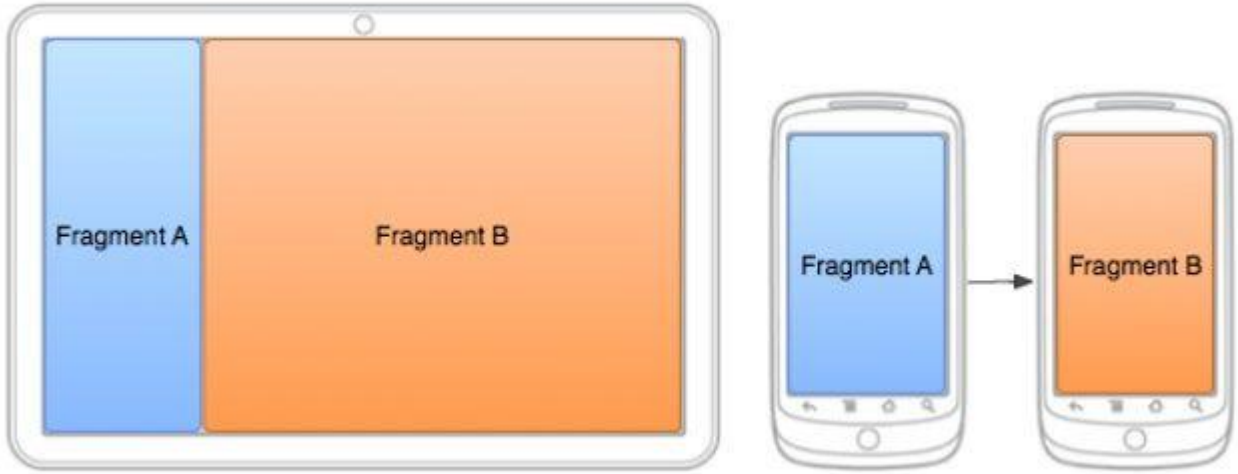
- Activity
- View
- Service
- Content Provider
- Broadcast Receiver
- Widget
- Intent

Activity - View

- Podemos decir que todas las pantallas de una aplicación son una “activity”. Más adelante vamos a ver que existen algunas variaciones, pero por ahora digamos que todas lo son. Es decir, que si una aplicación tiene cinco pantallas, tiene 5 “Actividades” o activities (puede no ser asi).
- Las activities están conformadas por dos partes: la parte lógica (java) y la parte gráfica (xml).
- Para la parte grafica, podemos colocar un nombre descriptivo al archivo .xml (login.xml), sin embargo, para la parte lógica debemos seguir un conjunto de buenas practicas y debemos llamar el archivo LoginActivity.java y esta clase siempre debe heredar de la clase Activity.
- Android pone a nuestra disposición una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, aunque también existe la posibilidad de extender la funcionalidad de estos controles básicos o crear nuestros propios controles personalizados.

Fragments

- Un *fragment* no puede considerarse ni un *control* ni un *contenedor*, aunque se parecería más a lo segundo. Un *fragment* podría definirse como una porción de la interfaz de usuario que puede añadirse o eliminarse de la interfaz de forma independiente al resto de elementos de la actividad, y que por supuesto puede reutilizarse en otras actividades.



login.xml | LoginActitvity.java

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".LoginActitvity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

login.xml | LoginActitvity.java

```java
package com.example.jotaz.applifecycle;

import ...

public class LoginActitvity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);
    }
}
```

Service

- Los servicios (*service*) son componentes sin interfaz gráfica que se ejecutan en segundo plano. En concepto, son similares a los servicios presentes en cualquier otro sistema operativo. Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (p.ej. actividades) si se necesita en algún momento la interacción con del usuario.

Content Provider

- Un proveedor de contenidos (*content provider*) es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones. Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación. De la misma forma, nuestra aplicación podrá acceder a los datos de otra a través de los *content provider* que se hayan definido.

Broadcast Receiver

- Un *broadcast receiver* es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: “Batería baja”, “SMS recibido”, “Tarjeta SD insertada”, …) o por otras aplicaciones (cualquier aplicación puede generar mensajes (*intents*, en terminología Android) broadcast, es decir, no dirigidos a una aplicación concreta sino a cualquiera que quiera escucharlo).

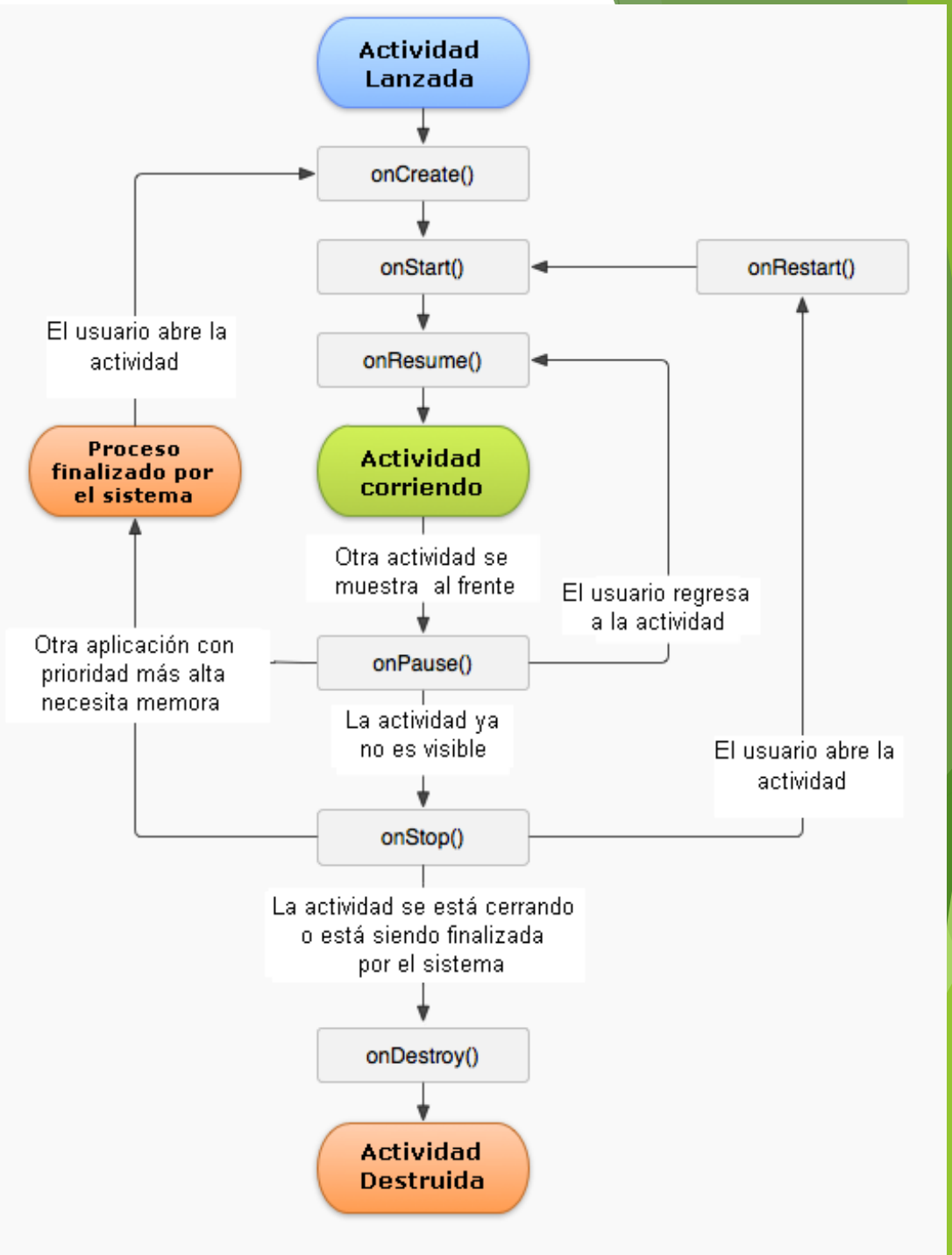
Widget

- Los *widgets* son elementos visuales, normalmente interactivos, que pueden mostrarse en la pantalla principal (*home screen*) del dispositivo Android y recibir actualizaciones periódicas. Permiten mostrar información de la aplicación al usuario directamente sobre la pantalla principal.

Ciclo de vida de un Activi

- Cuando estamos desarrollando una aplicación móvil para Android, debemos tomar en consideración los diferentes métodos que se ejecutan durante la ejecución de la misma. Los métodos que se ejecutan son los siguientes:
  - onCreate()
  - onRestart()
  - onStart()
  - onResume()
  - onPause()
  - onStop()
  - onDestroy()

- Expliquemos cada uno…



onCreate()

- Se llama ha este método cuando el activity se crea, ósea, que es lo primero que se ejecuta al abrir una aplicación.
- Cosas que se deben hacer aquí
  - Cargar el ContentView del layout correspondiente
  - Declarar e Inicializar los controles que se estarán utilizando en esta clase.
  - Obtener los datos que se envían de una actividad a otra.

onStart() y onRestart()

- Ambos se llaman inmediatamente después del onCreate. Si nuestra aplicación estaba en segundo plano, onStart o onRestart será llamado cuando la aplicación vuelva a estar en primer plano.
- Se debe utilizar para
  - Recargar datos con los cuales estaba trabajando sobre esa pantalla,
  - Preferiblemente para cargar datos que estén corriendo en background.

onResume

- Se llama cuando la actividad comienza a interactuar con el usuario, justo al momento de estar nuevamente la pantalla cargada con la interfaz grafica
- Se debe usar para
  - Cargar datos que hayan cambiado sobre la pantalla una vez se haya recargado la pantalla al usuario.

onPause

- es el primer método que se llama cuando la aplicación se está yendo de la pantalla.
- Nos funciona para
  - Detener código syncrono que esta corriendo (peticiones, bucles, animaciones, etc)
  - Guardar la información con la que estábamos trabajando en la pantalla si no la queremos perder.

- **Este método es importante porque puede ser el único en avisarnos de que la actividad o incluso toda la aplicación se está cerrando**

onStop

- Este método se ejecuta cuando la actividad se fue, ósea que desaparece de la pantalla del usuario
- No significa que la actividad se esté apagando, aunque podría ser
- Si estás haciendo algún proceso que solo debería estar corriendo cuando la actividad está en ejecución este es un buen momento para pararla.

onDestroy

- Es el último método que se llama antes del final
- Cualquier proceso de background que la actividad puede tener corriendo debe pararse.

Sin embargo porque este método se haya llamado no significa que la actividad sea borrada. Si tienes algún hilo corriendo, este puede seguir corriendo y consumiendo recursos incluso aunque este método se llame.

En resumen

Método	Descripción	Siguiente método
onCreate ()	Llamado cuando la actividad se creó por primera vez	*onStart ()*
onRestart ()	Invocado después de detener la actividad, antes de reiniciar	*onStart ()*
onStart ()	Se llama cuando la actividad se vuelve visible para el usuario	*onResume () / onStop ()*
En resumen()	Se llama cuando la actividad comienza a interactuar con el usuario	*onPause ()*
onPause ()	Invocado cuando una actividad previa está a punto de reanudarse	*onResume () / onStop ()*
onStop ()	Se llama cuando la actividad ya no es visible para el usuario	*onRestart () / onDestroy ()*
onDestroy ()	Llamada final recibida antes de que se destruya la actividad	Nada

http://eagle.phys.utk.edu/guidry/android/applicationLifecycles.html

Archivo Android Manifest.

- El archivo de manifiesto proporciona información esencial sobre tu aplicación al sistema Android, información que el sistema debe tener para poder ejecutar el código de la app.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.jotaz.applifecycle">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="AppLifeCycle"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".LoginActitvity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419

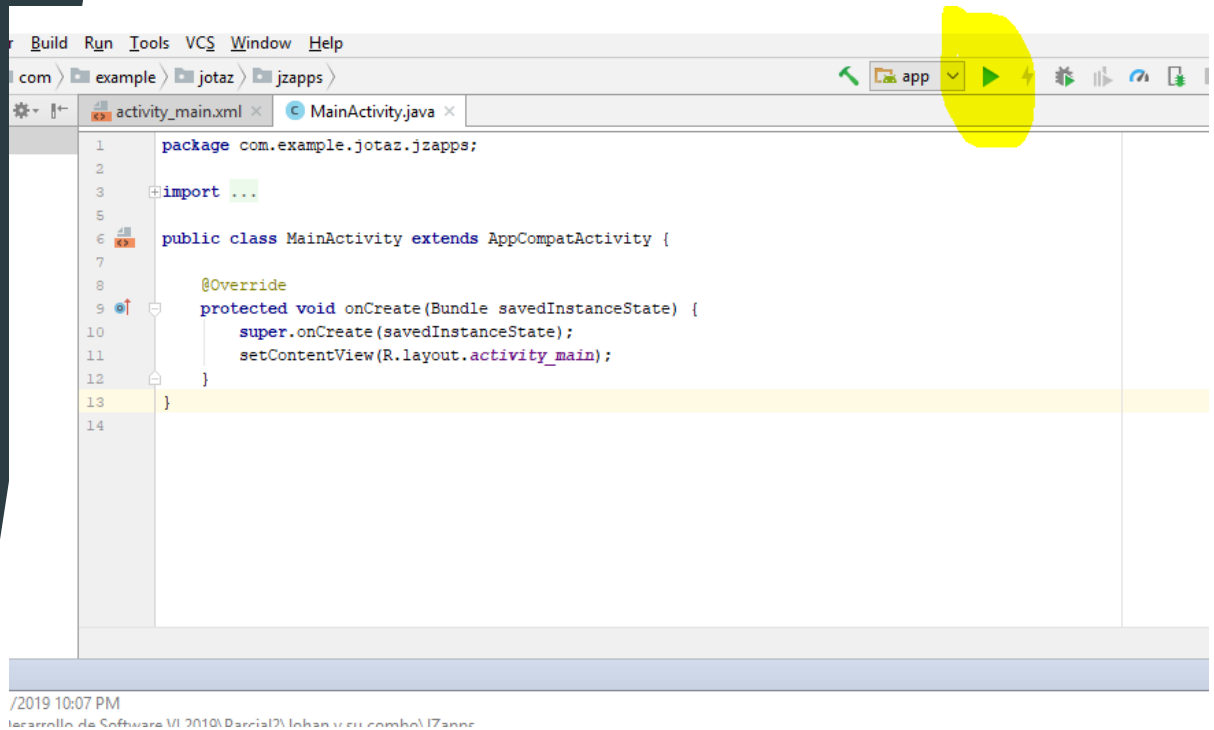
Android Manifest

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="54" android:versionName="2.1.5" package="com.smartreport.droid" android:installLocation="auto">
    <uses-sdk android:minSdkVersion="19" android:targetSdkVersion="26" />
    <application android:name=".AppBase" android:label="Smart Report Panamá" android:largeHeap="true" android:icon="@drawable/Launcher">
        <provider android:name="android.support.v4.content.FileProvider" android:authorities="com.smartreport.droid.fileprovider" android:exported="false" android:grantUriPermissions="true">
            <meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/file_paths"></meta-data>
        </provider>
    </application>
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />
</manifest>
```

- ✓ Nombra el paquete de Java para la aplicación.
- ✓ Describe los componentes de la aplicación, como las actividades, los servicios, los receptores de mensajes y los proveedores de contenido que la integran
- ✓ Determina los procesos que alojan a los componentes de la aplicación.
- ✓ Declara los permisos debe tener la aplicación para acceder a las partes protegidas de una API e interactuar con otras aplicaciones
- ✓ Declara el nivel mínimo de Android API que requiere la aplicación.
- ✓ Enumera las bibliotecas con las que debe estar vinculada la aplicación.

Ejecución de una Aplicación Móvil

- Básicamente, debemos dar clic sobre el botón triangular (como el play) para correr o en el botón de los engranajes para depurar. Ambos botones abren la siguiente pantalla.



Consideraciones para correr en teléfono Físico o Virtual

- Habilitar Instalaciones de Fuentes Externas.
- Habilitar Modo Desarrollador.
- Habilitar Depuración Externa

https://www.xatakandroid.com/sistema-operativo/opciones-de-desarrollo-de-android-para-que-sirven-y-cuales-deberiamos-activar

Físico o Virtual

- Como Podemos observar, se abre la pantalla de Selección de dispositivo disponible. En el ejemplo se ve que se encuentra disponible el “Samsung SM-N950F” que es el teléfono físico y el “Genymotion Samsung Galaxy S6” que es el teléfono virtual. Debe seleccionar 1 y darle OK.
- También, se muestra que están dos dispositivos virtuales disponibles propios del Android Studio.