



**UNIVERSIDAD
DE ANTIOQUIA**
1803

Sesión . Actividades

Ing. Edwin Andrés Cubillos Vega Msc.



A large light blue circle on the left side of the slide, with a grey arc extending from its right edge. Five colored circles (yellow, green, blue, purple, orange) are positioned along this arc, each corresponding to a text box on the right.

Principales Bloques de una App

Actividad

Multiples Actividades

Menu

Paso de datos entre Actividades

A diagram on the left side of the slide features a large light blue circle. A grey arc extends from its right edge, connecting to five horizontal rounded rectangular boxes. Each box is preceded by a small colored circle: yellow, green, blue, purple, and orange.

Principales Bloques de una App

Actividad

Multiples Actividades

Menu

Paso de datos entre Actividades



Principales Bloques de una App

- ❖ Android provee unos componentes principales que fundamentan el desarrollo de apps
- ❖ Estos componentes son implementados como clases de Java
- ❖ Los componentes principales de una app de Android son:
 - Activity
 - Services
 - Broadcast Receivers
 - Content Providers



Principales Bloques de una App

Actividades (Activities)

- ❖ Una actividad es usualmente una pantalla que el usuario ve en el dispositivo en un momento dado.
- ❖ Una aplicación tiene típicamente múltiples actividades.
- ❖ Una actividad es la parte más visible de la aplicación, contiene un layout
- ❖ Ejemplo: Las actividades son análogas a un sitio web:
 - Un sitio web tiene múltiples páginas, una aplicación android consiste de múltiples actividades.
 - Los sitios web tienen un home page, android tiene una actividad principal (main).
 - Así como un sitio web tiene que proveer alguna forma de navegación entre las páginas, una aplicación Android también.



Principales Bloques de una App

Que pasa cuando se lanza una actividad?

- ❖ Se crea un proceso nuevo en Linux.
- ❖ Se asigna memoria para todos los objetos de la interfaz de usuario.
- ❖ La información en los archivos XML es leída y cargada en el runtime de java.
- ❖ Se configura la pantalla.
- ❖ Los ESTADOS de una actividad son manejados por el Activity Manager.
- ❖ El comportamiento en las transiciones los define el programador.



Principales Bloques de una App

Servicios (Service)

- ❖ Los servicios corren en background (segundo plano) y no tiene ningún componente de interfaz de usuario.
- ❖ Es usual que los servicios corran en un hilo diferente al principal (interfaz de usuario).
- ❖ Realizan cualquier tipo de acciones: Actualizar datos, lanzar notificaciones, o mostrar elementos visuales.
- ❖ Pueden realizar tareas muy largas



Principales Bloques de una App

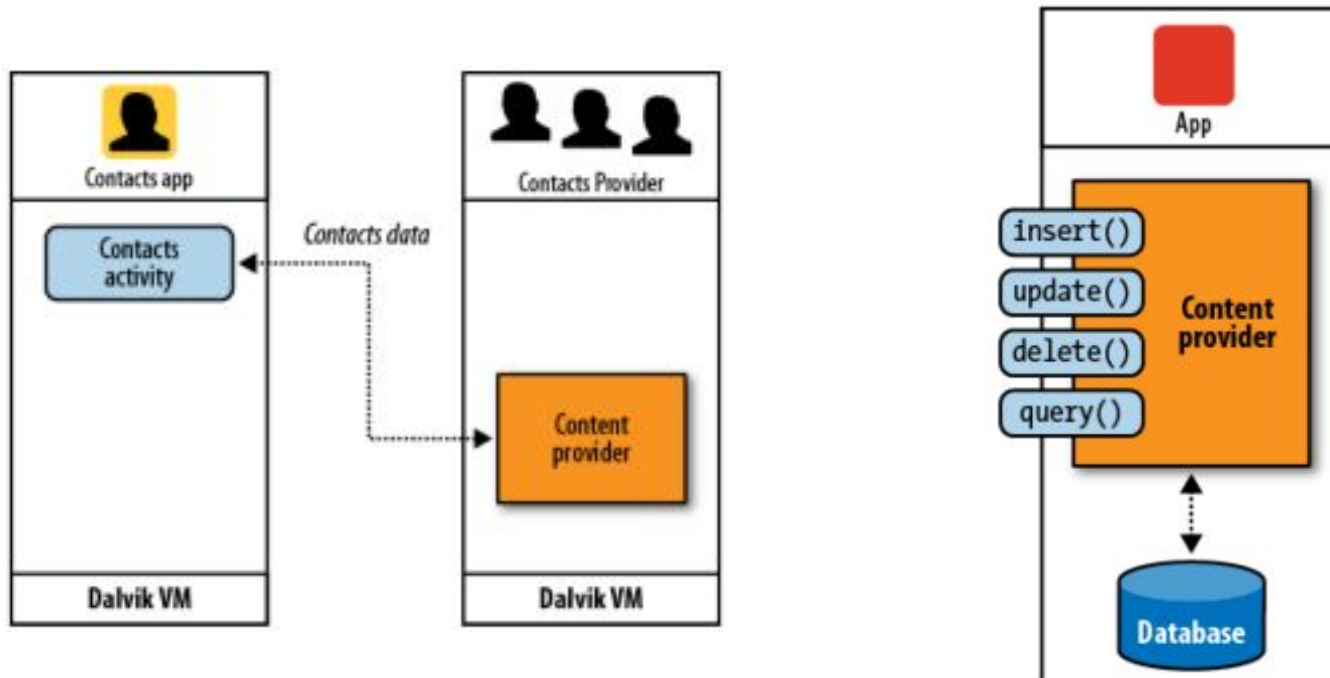
Broadcast Receivers

- ❖ Componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema o por otras aplicaciones
- ❖ Ejemplo: el sistema hace Broadcast de múltiples eventos, tales cómo:
 - Llegada de un SMS,
 - llamada entrante,
 - batería baja,
 - sistema cargado (system get booted).

Principales Bloques de una App

Proveedores de Contenido (Content Providers)

- ❖ Implementan un patrón de diseño denominado proxy: sirven como interfaz para permitir el intercambio de datos entre aplicaciones.
- ❖ Gracias a este se puede compartir datos sin mostrar detalles del almacenamiento interno, estructura o implementación



A large light blue circle on the left side of the slide, with a grey arc extending from its right edge to connect to a series of five horizontal rounded rectangles. Each rectangle contains a colored circle and a text label.

Principales Bloques de una App

Actividad

Multiples Actividades

Menu

Paso de datos entre Actividades

- ❖ La clase Activity se utiliza para presentar una GUI
- ❖ Captura la interacción con el usuario a través de la interface.
- ❖ Se crea en el archivo .kt
- ❖ Una aplicación Android está formada por varias actividades.
- ❖ Las actividades son las que controlan el ciclo de vida de una aplicación porque el usuario no cambia de aplicación sino de actividad
- ❖ El Sistema mantiene una pila con las actividades previamente visualizadas.
- ❖ El usuario puede regresar a la actividad anterior presionando la tecla retorno



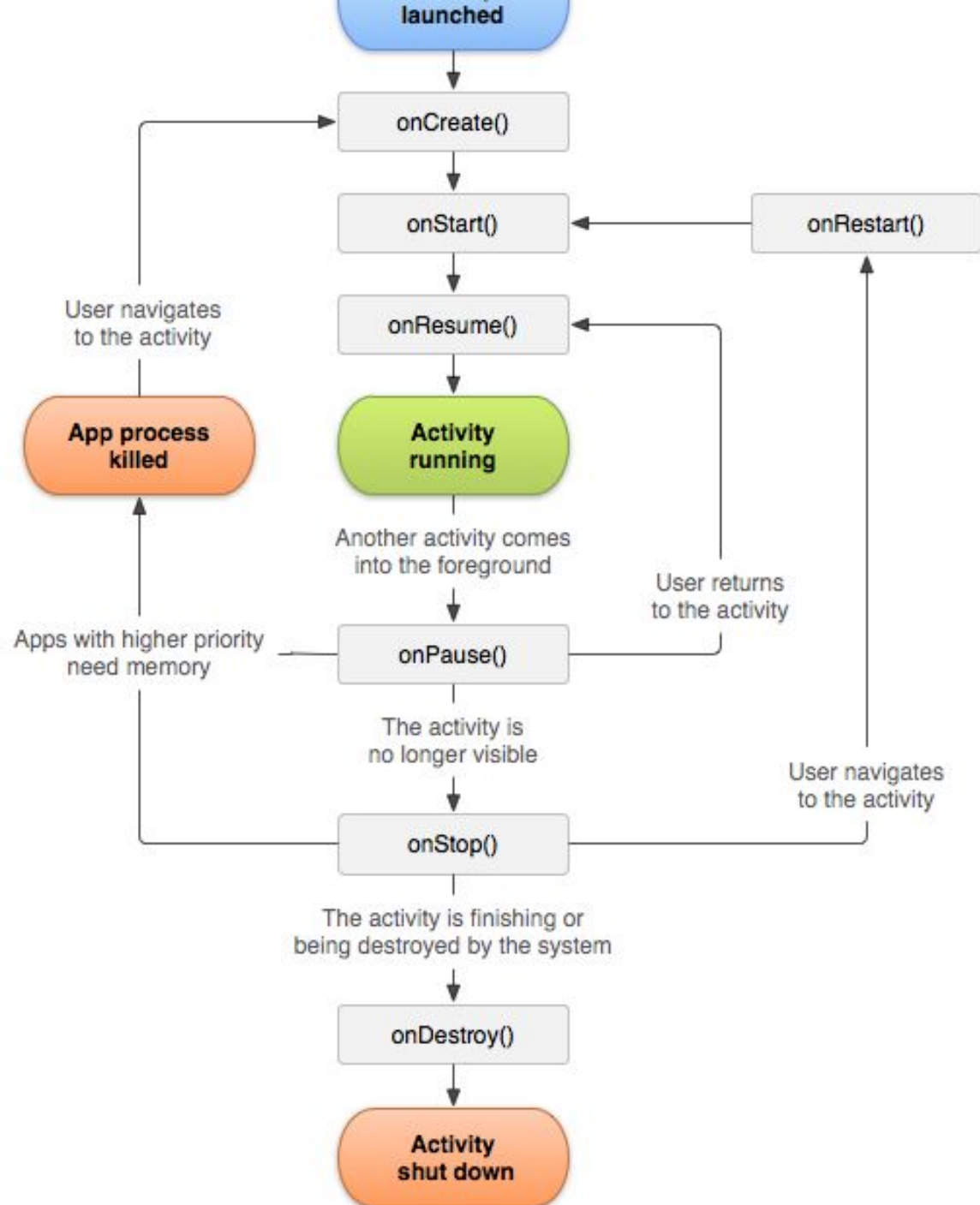
Activity

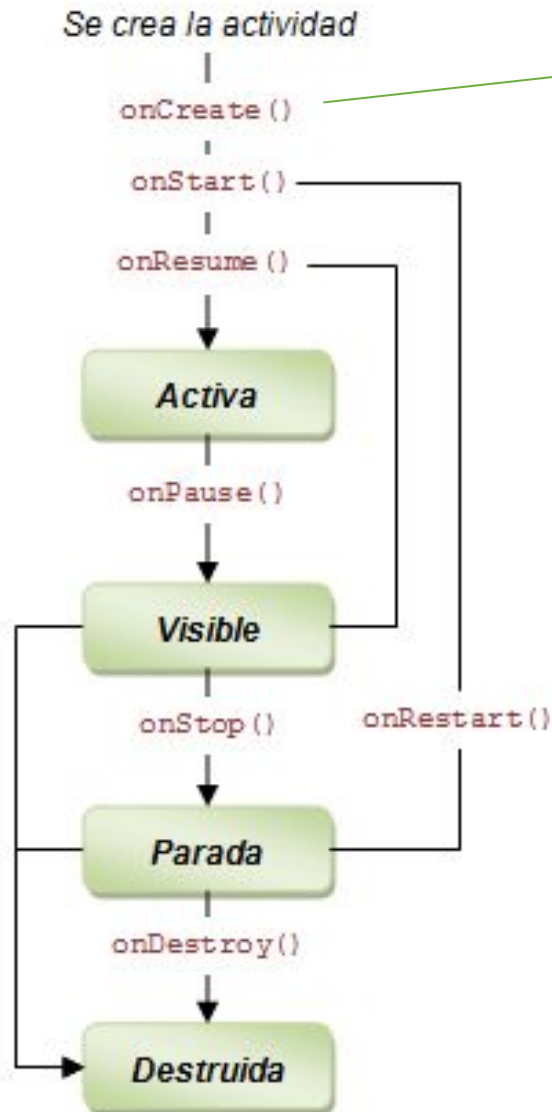
- ❖ Una aplicación Android corre dentro de su propio proceso Linux.
- ❖ Este proceso es creado con la aplicación y continuará vivo hasta que ya no sea requerido y el sistema reclame su memoria para asignarla a otra aplicación.
- ❖ En Android un proceso no es controlado por la aplicación sino por el sistema dependiendo de que se esta ejecutando (actividades o servicios)
- ❖ Si después de eliminar el proceso el usuario vuelve a ella, se crea de nuevo el proceso, pero se pierde el estado que tenía la app.
- ❖ Es necesario almacenar manualmente el estado de las actividades si así se requiere.

- ❖ Android es sensible al ciclo de vida de una actividad.
- ❖ Es necesario comprender y manejar los eventos relacionados con el ciclo de vida si se desea crear aplicaciones estables:
 - **Activa** (*Running*): La actividad está encima de la pila, lo que quiere decir que es visible y tiene el foco.
 - **Visible** (*Paused*): La actividad es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra actividad con alguna parte transparente o que no ocupa toda la pantalla. Cuando una actividad está tapada por completo, pasa a estar parada.
 - **Parada** (*Stopped*): Cuando la actividad no es visible. El programador debe guardar el estado de la interfaz de usuario, preferencias, etc.
 - **Destruída** (*Destroyed*): Cuando la actividad termina al invocarse el método *finish()*, o es matada por el sistema.



- ❖ Cada vez que una actividad de estado se va a generar eventos que ser capturados por ciclo de la actividad





`onCreate(Bundle)`

- Se llama en la creación de la actividad.
- Se utiliza para realizar todo tipo de inicializaciones
- como la creación de la interfaz de usuario o la inicialización de estructuras de datos.
- Puede recibir información de estado de la actividad (en una instancia de la clase Bundle), por si se reanuda desde una actividad que ha sido destruida y vuelta a crear.

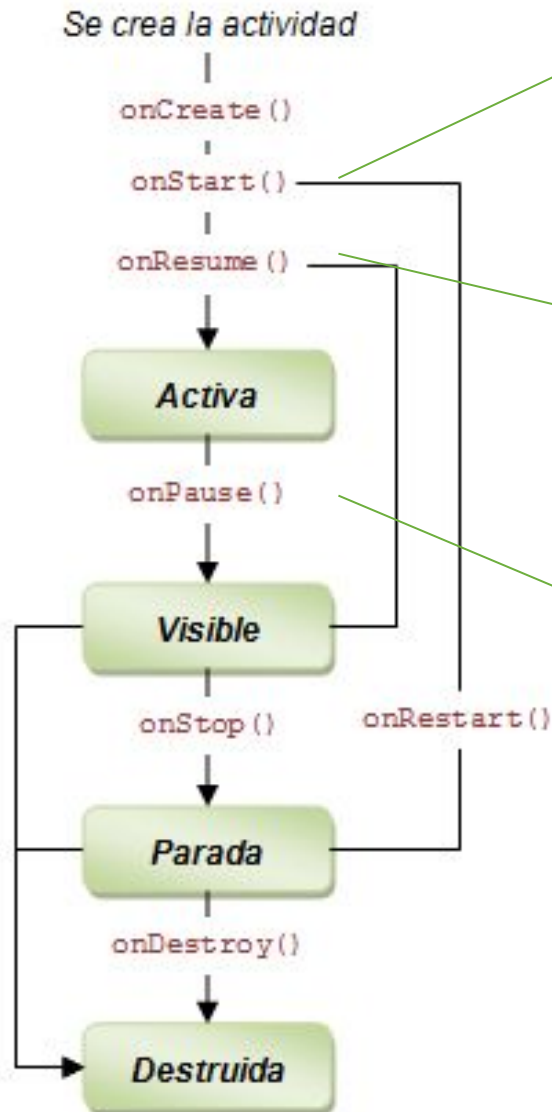
```
package com.edwinacubillos.sesion2_activities
```

```
import ...
```

```
class MainActivity : AppCompatActivity() {
```

```
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
    }
```

```
}
```

onStart():

- ❖ Nos indica que la actividad está a punto de ser mostrada al usuario.

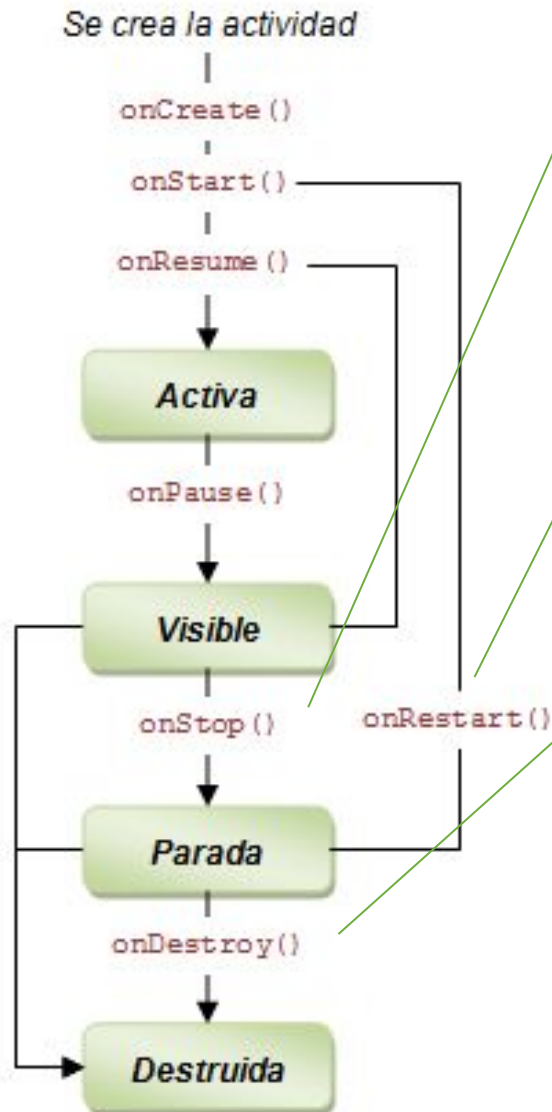
onResume():

- ❖ Se llama cuando la actividad va a comenzar a interactuar con el usuario. Es un buen lugar para lanzar las animaciones y la música.

onPause():

- ❖ Indica que la actividad está a punto de ser lanzada a segundo plano, normalmente porque otra actividad es lanzada.
- ❖ Es el lugar adecuado para detener animaciones, música o almacenar los datos que estaban en edición.

Activity



onStop():

- ❖ La actividad ya no va a ser visible para el usuario. Ojo si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método.

onRestart():

- ❖ Indica que la actividad va a volver a ser representada después de haber pasado por *onStop()*.

onDestroy():

- ❖ Se llama antes de que la actividad sea totalmente destruida.
- ❖ Por ejemplo, cuando el usuario pulsa el botón <volver> o cuando se llama al método *finish()*.
- ❖ Ojo si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método

❖ Ejercicio: Crear una app Sesión 6

1. Método onCreate
2. Agregar los 7 métodos
3. Probar la app

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
        Log.d( tag: "Método", msg: "onCreate")  
    }  
  
    override fun onStart() {  
        super.onStart()  
        Log.d( tag: "Método", msg: "onStart")  
    }  
  
    override fun onResume() {  
        super.onResume()  
        Log.d( tag: "Método", msg: "onResume")  
    }  
  
    override fun onPause() {  
        super.onPause()  
        Log.d( tag: "Método", msg: "onPause")  
    }  
  
    override fun onStop() {  
        super.onStop()  
        Log.d( tag: "Método", msg: "onStop")  
    }  
  
    override fun onRestart() {  
        super.onRestart()  
        Log.d( tag: "Método", msg: "onRestart")  
    }  
  
    override fun onDestroy() {  
        super.onDestroy()  
        Log.d( tag: "Método", msg: "onDestroy")  
    }  
}
```

Activity












Logcat

 Sony C6903 Android 5.1.1, API ▾

com.edwinacubillos.sesion2_acl ▾

Verbose ▾

 Método

Icon	Time	Process	Package	Class	Method
	07-16 15:14:07.531	16911-16911	com.edwinacubillos.sesion2_activities	D/Método:	onCreate
	07-16 15:14:07.532	16911-16911	com.edwinacubillos.sesion2_activities	D/Método:	onStart
	07-16 15:14:07.533	16911-16911	com.edwinacubillos.sesion2_activities	D/Método:	onResume
	07-16 15:14:09.444	16911-16911	com.edwinacubillos.sesion2_activities	D/Método:	onPause
	07-16 15:14:09.811	16911-16911	com.edwinacubillos.sesion2_activities	D/Método:	onStop
	07-16 15:14:12.901	16911-16911	com.edwinacubillos.sesion2_activities	D/Método:	onRestart
	07-16 15:14:12.905	16911-16911	com.edwinacubillos.sesion2_activities	D/Método:	onStart
	07-16 15:14:12.905	16911-16911	com.edwinacubillos.sesion2_activities	D/Método:	onResume
	07-16 15:14:15.079	16911-16911	com.edwinacubillos.sesion2_activities	D/Método:	onPause
	07-16 15:14:15.174	16911-16911	com.edwinacubillos.sesion2_activities	D/Método:	onStop
	07-16 15:14:15.174	16911-16911	com.edwinacubillos.sesion2_activities	D/Método:	onDestroy



A diagram showing the components of an app. On the left, a large light blue circle is partially visible. To its right, a vertical grey line acts as a spine. Five colored circles (yellow, green, blue, purple, orange) are positioned along this spine, each corresponding to a text box on the right. The text boxes are rounded rectangles with grey borders. The third box, 'Múltiples Actividades', is highlighted with a bold font. The background of the slide features a blue header with the word 'Contenido' and a bottom section with a blurred image of a newspaper and a keyboard.

Principales Bloques de una App

Actividad

Múltiples Actividades

Menu

Paso de datos entre Actividades



Múltiples Actividades

- ❖ Al crear una nueva aplicación por defecto se crea una actividad que tiene asociado un layout
- ❖ Pero y si queremos agregar múltiples actividades cómo lo hacemos?
 1. Crear un Nuevo layout para una actividad
 2. Crear una nueva clase descendiente de activity, la cual abrirá el layout que queremos visualizar.
 3. Programar la app para que desde la actividad inicial se pueda abrir la nueva actividad
 4. Registrar la nueva actividad en el android Manifest.
- ❖ Android Studio realiza los pasos 1, 2 y 4 de forma automática



A diagram illustrating the components of an app. On the left, a large light blue circle is partially visible. To its right, a vertical grey line acts as a spine, with five colored circles (yellow, green, blue, purple, orange) attached to it. Each colored circle is followed by a rounded rectangular box containing text. The text boxes are arranged vertically, with the first one at the top and the last one at the bottom.

Principales Bloques de una App

Actividad

Múltiples Actividades

Menú

Paso de datos entre Actividades

Menu Overflow

Es un menú que se ubica en el ActionBar



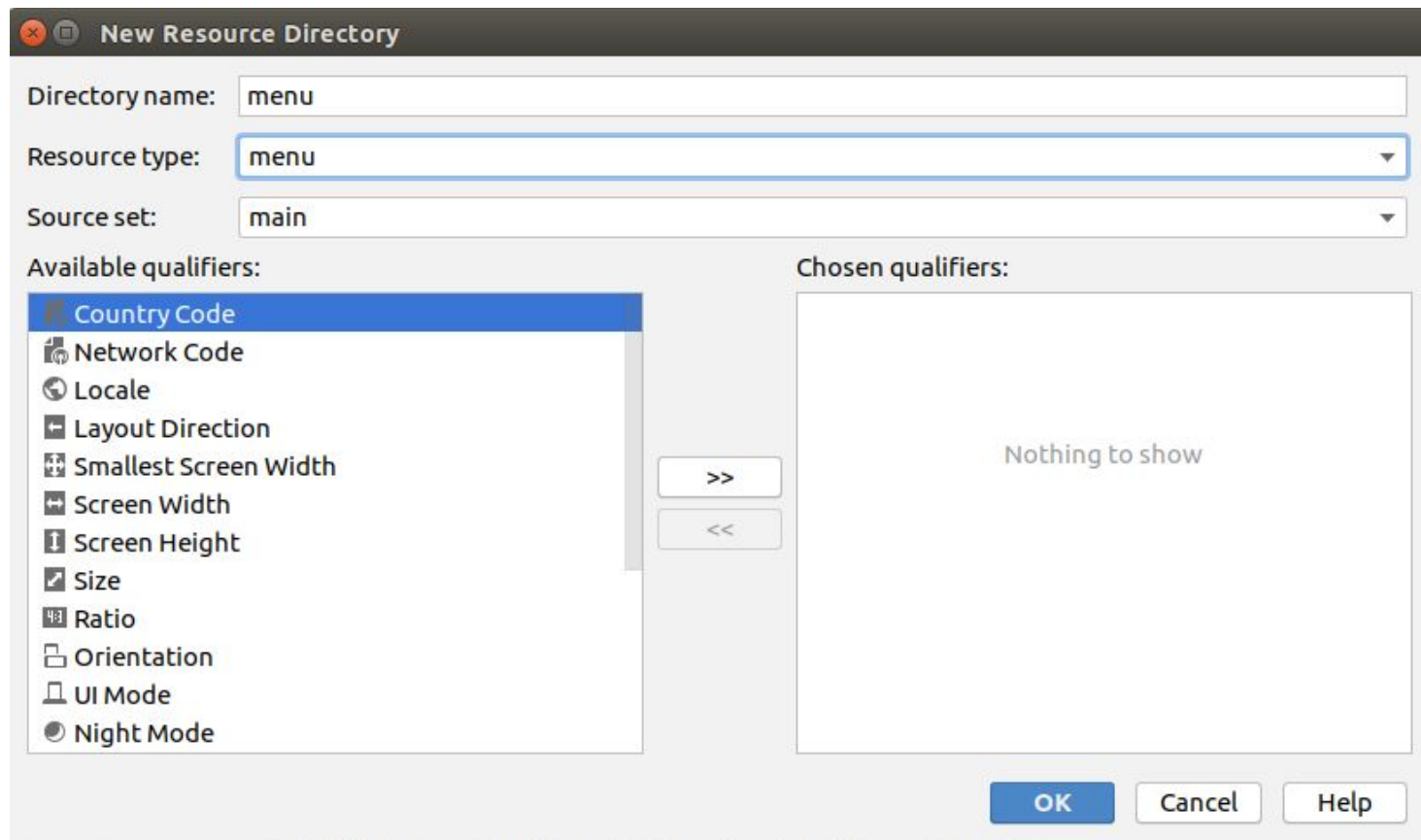
Menu Overflow

Inicialmente el directorio res no contiene un directorio menu, por lo que se debe crear



Menu Overflow

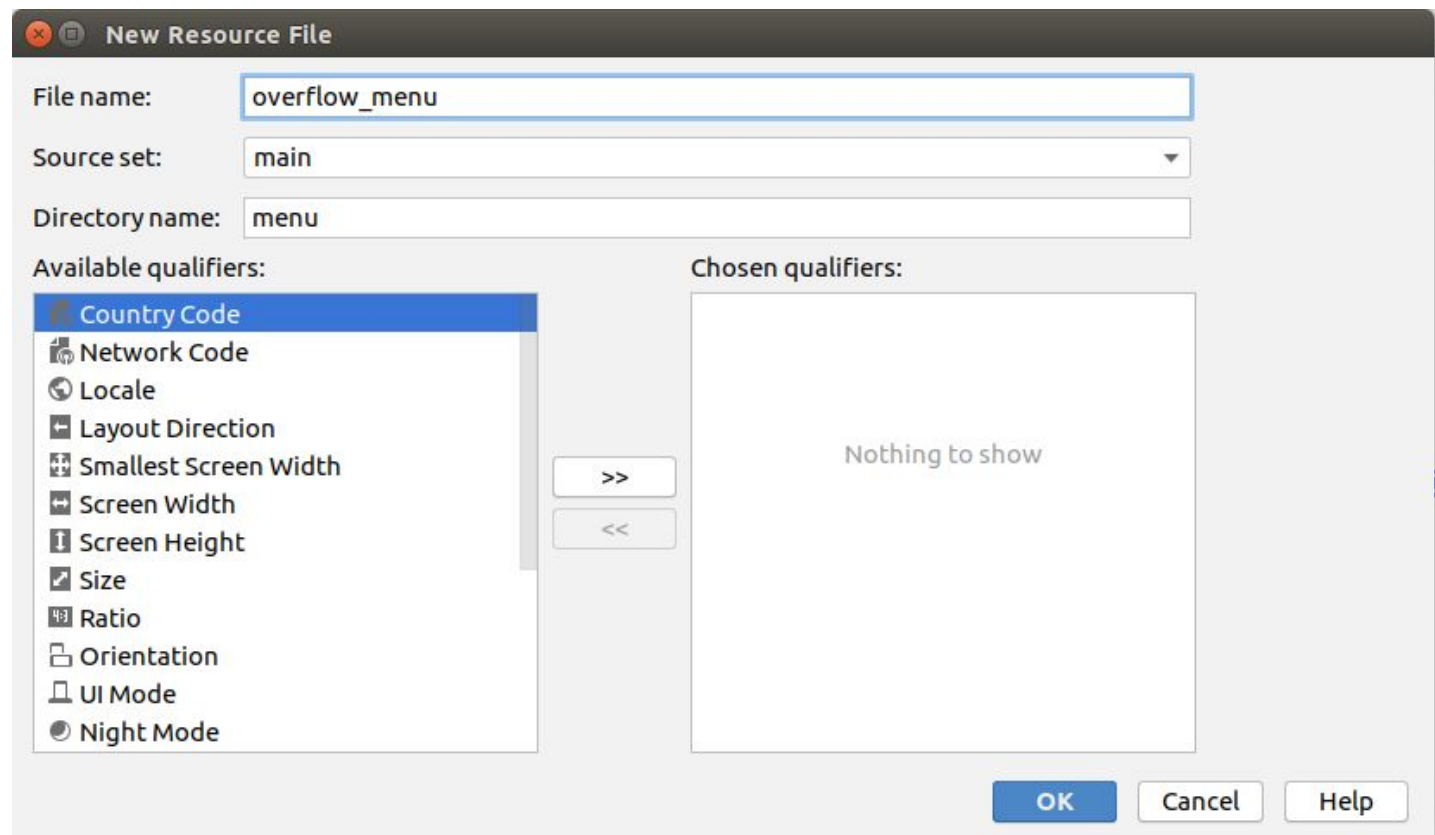
Click derecho res -> New -> Android Resource Directory



Menu Overflow

Ahora se crea un archivo xml con el nombre deseado en este caso menu.xml

Click derecho en menu -> New -> New Resource File



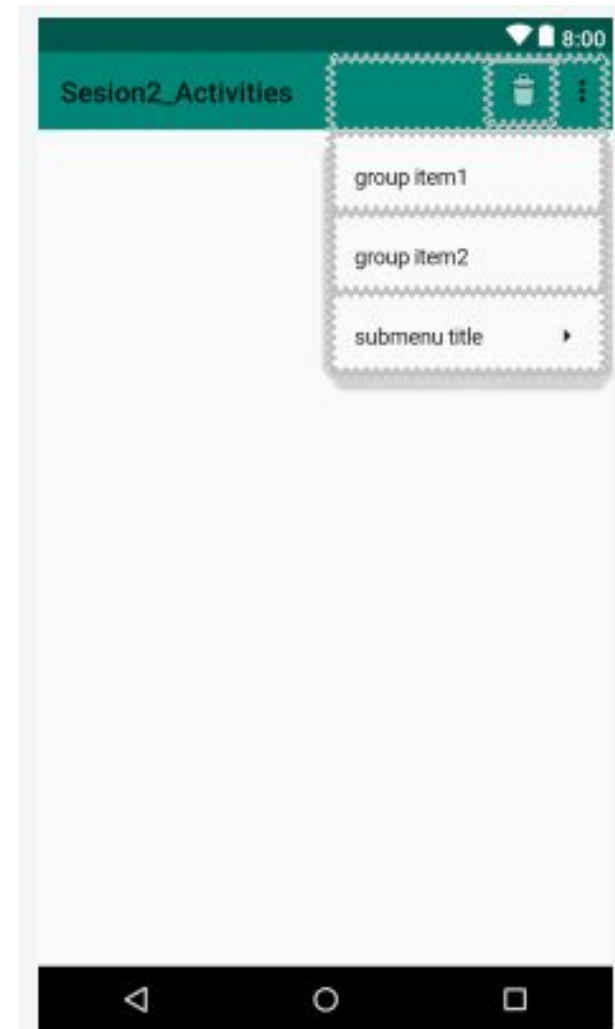
Propiedades del Menú

- ❖ **android:id**. El ID identificativo del elemento, con el que podremos hacer referencia dicha opción.
- ❖ **android:title**. El texto que se visualizará para la opción.
- ❖ **android:icon**. El icono asociado a la acción.
- ❖ **android:showAsAction**. Si se está mostrando una action bar (último tema de la clase), este atributo indica si la opción de menú se mostrará como botón de acción o como parte del menú de overflow. Puede tomar varios valores:
 - **ifRoom**. Se mostrará como botón de acción sólo si hay espacio disponible.
 - **withText**. Se mostrará el texto de la opción junto al icono en el caso de que éste se esté mostrando como botón de acción.
 - **never**. La opción siempre se mostrará como parte del menú de overflow.
 - **always**. La opción siempre se mostrará como botón de acción. Este valor puede provocar que los elementos se solapen si no hay espacio suficiente para ellos. [Más Métodos](#)

Menu Overflow

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
  <item android:id="@+id/item1"
        android:title="actividad 1"
        android:icon="@android:drawable/ic_menu_delete"
        app:showAsAction="ifRoom|withText"/>

  <group android:id="@+id/group">
    <item android:id="@+id/group_item1"
          android:onClick="onGroupItemClick"
          android:title="group item1"
          android:icon="@android:drawable/btn_dialog"/>
    <item android:id="@+id/group_item2"
          android:onClick="onGroupItemClick"
          android:title="group item2"
          android:icon="@android:drawable/btn_star_big_on"/>
  </group>
  <item android:id="@+id/submenu"
        android:title="submenu title" >
    <menu>
      <item android:id="@+id/submenu_item1"
            android:title="submenu item1"
            android:icon="@android:drawable/checkbox_on_background"/>
    </menu>
  </item>
</menu>
```



Menu Overflow

overflow_menu.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android">
3   <item android:id="@+id/mActivity2"
4         android:title="@string/actividad_2"/>
5   <item android:id="@+id/mActivity3"
6         android:title="@string/actividad_3"/>
7 </menu>
```



- ❖ No inflar el menú generamos el método `onCreateOptionsMenu`

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {  
    menuInflater.inflate(R.menu.example_menu, menu)  
    return true  
}
```


- ❖ Una vez creado el menú se le debe dar funcionalidad
- ❖ Dos opciones:
 - a. Opción 1

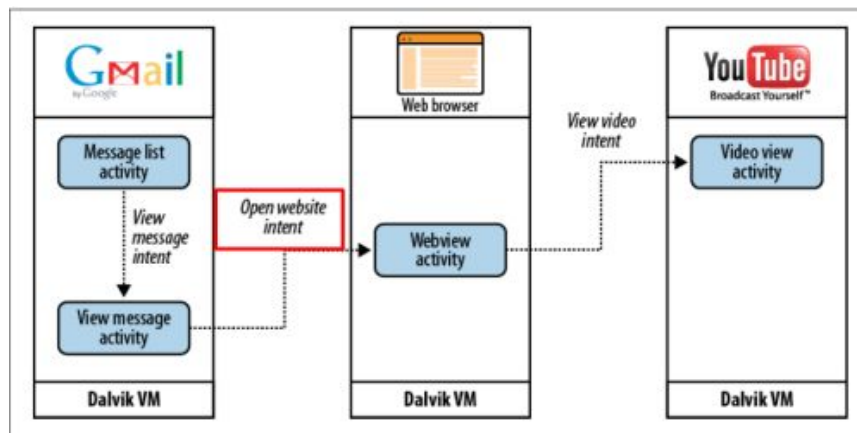
```
fun onGroupItemClick(item: MenuItem) {  
    when (item.itemId) {  
        R.id.group_item1 -> Toast.makeText( context: this, text: "menu group_item1", Toast.LENGTH_SHORT).show()  
    }  
}
```

b. Opción 2

```
override fun onOptionsItemSelected(item: MenuItem?): Boolean {  
    when (item!!.itemId) {  
        R.id.group_item1 -> Toast.makeText( context: this, text: "menu group_item1", Toast.LENGTH_SHORT).show()  
    }  
    return super.onOptionsItemSelected(item)  
}
```

Intents

- ❖ Es una descripción abstracta de una operación a ser realizada como:
- ❖ Disparan otras actividades.
- ❖ Inician o detienen servicios.
- ❖ Envío de broadcast
- ❖ Permite compartir información entre actividades



[Mas Información](#)

Contexto de la Aplicación (Application Context)

- ❖ Se refiere al proceso en el cual “viven” todos sus componentes tales como:
 - Actividades,
 - Servicios,
 - Proveedores de contenido, y
 - Broadcast receivers.
- ❖ El proceso como “contenedor” le permite a las diferentes partes compartir datos y recursos.

```
var intent = Intent( packageContext: this, DataActivity::class.java)  
startActivity(intent)
```



Actividad a abrir

```
override fun onOptionsItemSelected(item: MenuItem?): Boolean {  
    when (item!!.itemId) {  
        R.id.group_item1 -> {  
            Toast.makeText(context: this, text: "menu group_item1", Toast.LENGTH_SHORT).show()  
            var intent = Intent(packageContext: this, DataActivity::class.java)  
            startActivity(intent)  
        }  
    }  
    return super.onOptionsItemSelected(item)  
}
```

❖ Probar su funcionamiento

A diagram on the left side of the slide features a large light blue circle. A grey arc extends from its right edge, connecting to five horizontal rounded rectangular boxes. Each box is preceded by a small colored circle: yellow, green, blue, purple, and orange. The text inside the boxes lists the components of an app.

Principales Bloques de una App

Actividad

Múltiples Actividades

Menú

Paso de datos entre Actividades



Paso de datos entre Actividades

- ❖ Para realizar envío y recepción de datos entre actividades se utilizan los intents
- ❖ Como ejemplo crearemos una actividad para la aplicación de notas en la que podamos configurar los porcentajes de las notas
- ❖ Al dar clic en el menú configuración se debe abrir esta actividad
- ❖ El MainActivity le pasará el porcentaje actual de las notas y se visualizará en los EditText para edición
- ❖ Inicialmente la aplicación no hará nada más, solo visualizar, luego configuraremos la aplicación para que retorne los nuevos porcentajes al MainActivity

Paso de datos entre Actividades

- ❖ Después de crear el objeto intent y antes de iniciar la actividad se colocan los datos que se quieren enviar a la nueva actividad

```
override fun onOptionsItemSelected(item: MenuItem?): Boolean {  
    when (item!!.itemId) {  
        R.id.mActividadDos -> {  
            Toast.makeText( context: this, text: "ir a actividad 2", Toast.LENGTH_SHORT).show()  
            var intent = Intent( packageContext: this, SegundaActividad::class.java)  
            intent.putExtra( name: "usuario", value: "edwinacubillos")  
            intent.putExtra( name: "cedula", value: 986)  
            startActivity(intent)  
        }  
    }  
    return super.onOptionsItemSelected(item)  
}
```

Paso de datos entre Actividades

- ❖ Para recibir los datos en la actividad settings se debe crear un objeto tipo Bundle

```
var datosRecibidos : Bundle? = intent.extras  
Toast.makeText( context: this,datosRecibidos.getString( key: "usuario"), Toast.LENGTH_SHORT).show()  
Log.d( tag: "cedula",datosRecibidos.getInt( key: "cedula").toString())
```


Paso de datos entre Actividades

- ❖ Si quisiéramos recibir datos de la segunda actividad (similar a una respuesta) el método cambia

```
var intent = Intent( packageContext: this, SegundaActividad::class.java)
intent.putExtra( name: "usuario", value: "edwinacubillos")
intent.putExtra( name: "cedula", value: 986)
startActivityForResult(intent, requestCode: 1234)
```

- ❖ 1234 se llama RequestCode y sirve para identificar quien lanza la actividad

```
override fun onOptionsItemSelected(item: MenuItem?): Boolean {
    when (item!!.itemId) {
        R.id.mActividadDos -> {
            Toast.makeText( context: this, text: "ir a actividad 2", Toast.LENGTH_SHORT).show()
            var intent = Intent( packageContext: this, SegundaActividad::class.java)
            intent.putExtra( name: "usuario", value: "edwinacubillos")
            intent.putExtra( name: "cedula", value: 986)
            startActivityForResult(intent, requestCode: 1234)
        }
    }
    return super.onOptionsItemSelected(item)
}
```

Paso de datos entre Actividades

- ❖ Y como pasamos los datos de la actividad settings a la actividad principal?
- ❖ Configuramos el botón, creamos el intent y agregamos los datos, luego se envía el resultCode y terminamos con finish()

```
button.setOnClickListener{ it: View!  
    var intent = Intent()  
    intent.putExtra( name: "username", value: "Edwin")  
    intent.putExtra( name: "password", value: "123456")  
    setResult(Activity.RESULT_OK, intent)  
    finish()  
}
```


Paso de datos entre Actividades

- ❖ Finalmente en la clase MainActivity se debe definir un método llamado onActivityResult

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  
    super.onActivityResult(requestCode, resultCode, data)  
}
```

- ❖ requestCode: permite identificar quien envía los datos
- ❖ resultCode: resultado de la actividad, si es correcta o no
- ❖ data: Intent con datos recibidos

Paso de datos entre Actividades

- ❖ Para obtener los datos enviados desde la actividad Settings

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  
    super.onActivityResult(requestCode, resultCode, data)  
    if (requestCode == 1234 && resultCode == Activity.RESULT_OK){  
        Log.d( tag: "username", data!!.extras.getString( key: "username"))  
        Log.d( tag: "password", data!!.extras.getString( key: "password"))  
    }  
}
```