Introducción a la programación en ANDROID

Segunda parte

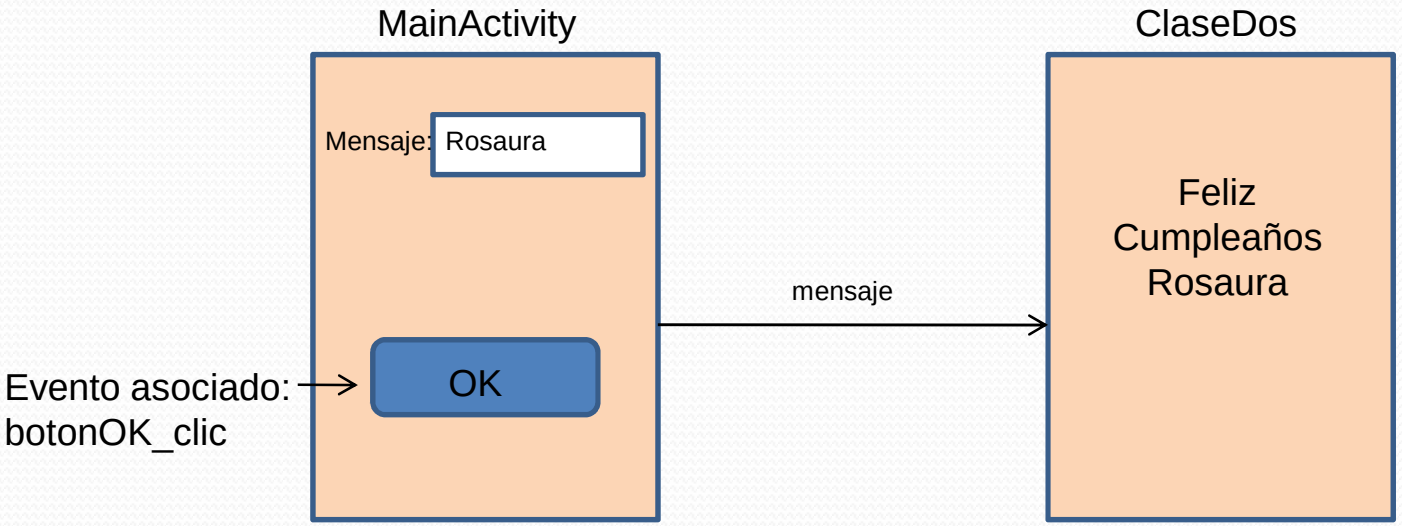
Laboratorio de Programación Avanzada
Facultad de Informática
Universidad Nacional del Comahue

Actividades

- En una aplicación pueden definirse muchas actividades
- Solo una actividad puede especificarse como actividad principal
- Cada actividad se encarga de recrear una ventana de UI por defecto (layout)
- Una actividad puede retornar un resultado a la actividad previa
- En el archivo AndroidManisfest se declaran las actividades que pueden ser iniciadas

Intents - Intenciones

- Una actividad puede iniciar otra, para esto se utilizan las intenciones (Intent)
- Las intenciones son un mecanismo de comunicación



Actividades e intenciones

- En la Actividad Principal en el método OnClick:

```
….
Switch (view.getId() ){
        case R.id.boton_saludo:
                Intent in= Intent(this, ClaseDos.class);
                in.putExtra(“mensaje”, getString(R.string.msj)+nombre);
                startActivity( in);
                break;
```

- En la Segunda Actividad en el método OnCreate:

```
…
                Intent in= getIntent( );
                String texto_msj = in.getStringExtra(“mensaje”);
                TextView  tv = (TextView) findViewById (R.id.textMessage);
                tv.setText (texto_msj);
```

Funciones de los Intents

- Se usan para iniciar una Actividad
  - startActivity(intent) o startActivityForResult(intent)
- Se usan para iniciar un servicio, un servicio realiza una actividad en background (ej. Descargar un archivo)
  - startService(intent) y bindService(intent)
- Se usan para entregar un broadcast, un broadcast es un mensaje que cualquier aplicación puede recibir
  - sendBroadcast(intent), sendOrderedBroadcast(intent) o sendStickyBroadcasr(intent)

Clase Intent

- La información fundamental es: la **acción** y los **datos**
- También se puede especificar: category, type, component y extras

Intent() → Crea un Intent vacío
Intent(String action) → Crea un Intent con una acción
Intent(Context packageContext, Class cls) → Crea un Intent para un componente específico
Intent(String action, Uri uri) → Crea un intent con una acción y datos
Intent(String action, Uri uri, Context packageContext, Class cls) → Crea un Intent para un componente con una acción y datos

putExtra (nombre parámetro, valor) → Usado para pasar información
setType (MIME type) → Usado para especificar el tipo que maneja el Intent cuando no retorna datos

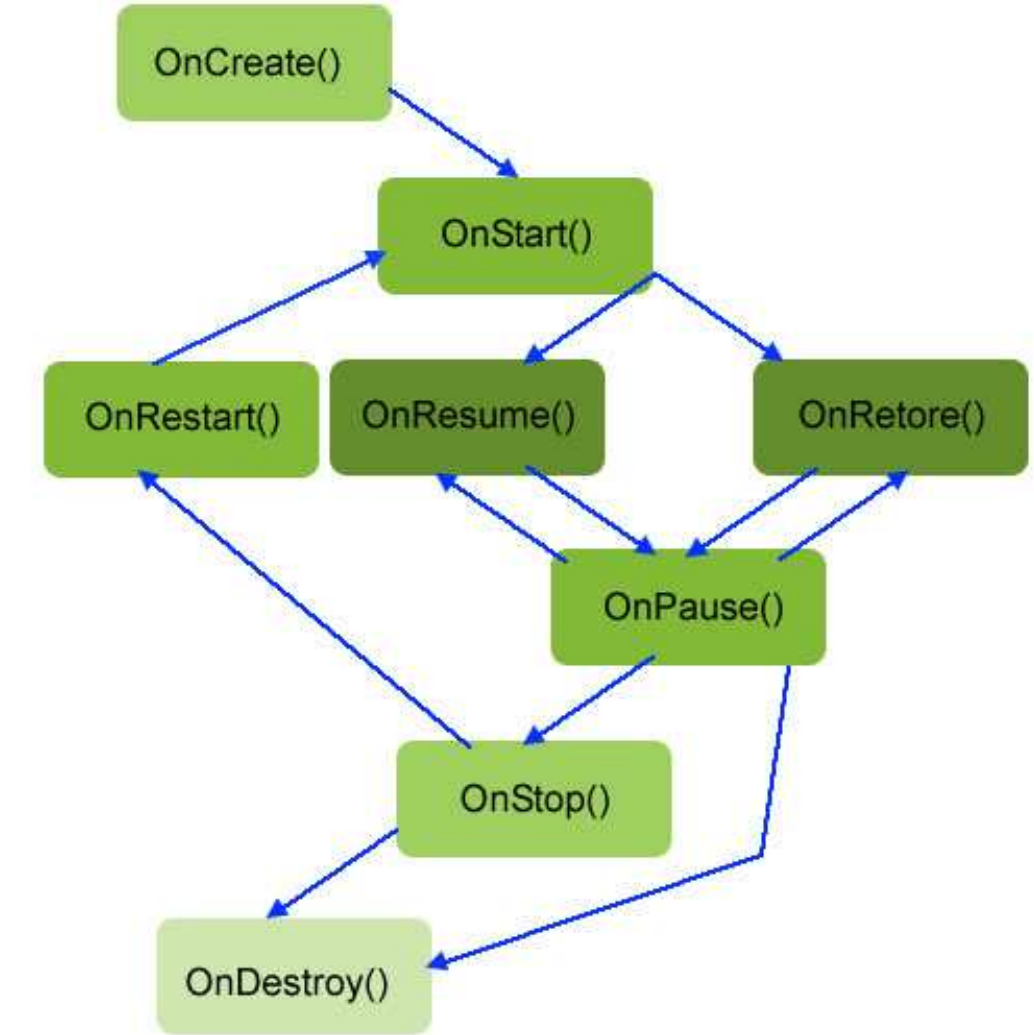
http://developer.android.com/intl/es/reference/android/content/Intent.html#setType(java.lang.String)

Ciclo de vida de las Actividades

- El ciclo de vida se inicia cuando el componente es creado y termina cuando es destruido
- En el medio puede estar activo o inactivo
- Una vez que una actividad es creada puede estar en 3 estados:
  - Resumed: la actividad esta en el primer plano de la pantalla, se puede interactuar con ella
  - Paused: la actividad perdió el foco pero aun es visible al usuario
  - Stopped: la actividad ha sido ocultada por otra actividad, aun retiene su estado y la información

Ciclo de vida

- El sistema puede terminar una actividad que está en alguno de los dos últimos estados (invocando al método finish() de la actividad o matando el proceso)



Escenarios Comunes

- OnPause → OnResume
- OnPause →OnStop → OnRestart →OnStart → OnResume

```
public class MainActivity extends Activity {
	private void log(String text) {
		Log.d(“Test del Ciclo de Vida: ", text);
	}
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		log(“Creada");
	}
	public void onStart() {
		super.onStart();
		log(“Iniciada");
	}
```

La clase Log

- android.util.Log
- Los mensajes incluyen: fecha y hora, criticidad, PID, Tag y mensaje
- Según su criticidad: error, warning, info, debug y verbose
- Para cada uno de estos tipos de mensaje existe un método estático que permite añadirlo al log
- Para cada una de las categorías éstos métodos son: e(), w(), i(), d() y v().
- En la perspectiva de Eclipse llamada DDMS podemos ver los mensajes en la vista llamada *LogCat*.

Rotación del dispositivo

- Los datos almacenados que el sistema usa para restaurar el estado previo son guardados como pares clave-valor en un objeto Bundle.
- El sistema guarda cada valor de los view de la aplicación
- Podemos guardar información adicional usando los métodos put

```
public void onSaveInstanceState (Bundle outState) {
      outState.putString("GRID", game.gridToString());
      super.onSaveInstanceState(outState);
}
public void onRestoreInstanceState (Bundle savedInstanceState){
      super.onRestoreInstanceState(savedInstanceState);
      String grid = savedInstanceState.getString("GRID");
      game.stringToGrid(grid);
      setFigureFromGrid();
}
private void setFigureFromGrid (){
    //restaurar el estado del juego en la pantalla

}
```

Implementar en la Clase Game

Intenciones y Aplicaciones Nativas

- Las Intenciones también sirven para abrir aplicaciones como el navegador, una aplicación de email o el teléfono

```
New Intent(“android.intent.action.VIEW”), URI.parse(“http:// …. “ ));

New Intent(“android.intent.action.VIEW”), URI.parse(“mailto: ….“ ));
```

- En el archivo AndroidManifest especificar los permisos:

<uses_permission android.name=“android.permission.CALL_PHONE”/>

Ejemplo envío mensaje

```
public class MainActivity extends Activity implements OnClickListener {
@Override
public void onCreate(Bundle savedInstanceState) {
…                    //asigno como escuchador al boton send y al boton call
    button = (Button) findViewById(R.id.call);
    button.setOnClickListener(this);
}
@Override
public void onClick(View v) {
    if (v.getId() == R.id.send) {
        Intent intent = new Intent(android.content.Intent.ACTION_SEND);
        intent.setType("text/plain");
        intent.putExtra(Intent.EXTRA_SUBJECT, "Probando desde Android");
        intent.putExtra(Intent.EXTRA_TEXT, "Hola, este mensaje te lo envio desde mi aplicación ");
        startActivity(intent);
    } else
          if (v.getId() == R.id.call) {       // Incluir android.permission.CALL_PHONE en el manifest
              Intent intent = new Intent(android.content.Intent.ACTION_CALL);
              intent.setData(Uri.parse("tel:+620222333"));
              startActivity(intent);
          }     }     }
```

Multimedia

- La clase MediaPlayer permite reproducir audio y video
  - setLooping(true): ejecución infinita
  - start(): arranca o reanuda la reproducción (llamarlo en OnResume)
  - stop(): detiene la reproducción (llamarlo en onPause). Tras parar la ejecución hay que llamar a release para liberar los recursos
- El archivo (MP3) debe ubicarse en /res/raw

Ejemplo: aplicación con sonido

```
public class Musica {
    private static MediaPlayer player;

    public static void play (Context context, int id){
        player = MediaPlayer.create(context, id);
        player.setLooping(true);
        player.start();
   }

    public static void stop (Context context){
        if(player != null){
            player.stop();
            player.release();
            player = null;
        }
    }  }
```

- Desde mainActivity invocar a play en el método onResume y a stop en el método onStop
- Incluir botones “reproducir” y “detener”

Diseño de UI - Colores

- Se definen como recursos (RGB) y pueden aplicarse a cualquier vista

miLayout.xml

```
<linearlayout
        …
        android:background="#FF9E49“
        … >
        <button
                …
                android:textColor="#FF9E49"
                …/>
        <button
                …
                android:background="@color/micolor“
                …>
</linearlayout>
```

Colors.xml

```
<resources>
    <color name="micolor">#551100</color>
</resources>
```

Diseño de UI - Estilos

- Define un conjunto de propiedades que pueden ser aplicados a una vista, como tamaño, márgenes, color, fuentes, etc.

```
/res/values/styles.xml
<resources>
    <style name="MiEstilo"
    parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>

En el layout
…
<TextView
      style="@style/MiEstilo"
      android:text="Este texto tiene el estilo que defini" />
```

Diseño de UI - Temas

- Temas: son estilos aplicados a una Actividad o a una Aplicación. Cada elemento del estilo se aplica solo a aquellos elementos donde sea posible. Por ej. CodeFont solo afectará al texto.
- Editar el *AndroidManifest.xml* :
  - <application android:theme="@style/MiTema">
  - <activity android:theme="@style/MiTema">

Especifico un tema predefinido en AndroidManifest.xml
:
<activity android:theme="@android:style/Theme.Translucent">

Creo un nuevo tema a partir de un predefinido:

<style name="MiTema" parent="android:Theme.Light">
  <item name="android:colorBackground">@color/mi_color</item>
</style>

Luego lo especifico en AndroidManifest.xml

<activity android:theme="@style/MiTema">

Diseño de UI – Gráficos 2D

- Drawable
  - Es una clase que representa un tipo de recurso que permite dibujar algo en la pantalla, dentro de una vista. No provee manejo de eventos, no provee formas de interactuar
  - Es adecuado cuando no se requiere modificarlo dinámicamente y cuando no forman parte de la animación en un juego
- Canvas
  - Es una clase que permite dibujar algo directamente llamando al método draw deseado, de esta manera se tiene mayor control de la animación

Ejemplo usando Drawable

En MainActivity:

```
…
 LinearLayout   mLinearLayout;

  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

  // Crea un LinearLayout  en el cual poner el ImageView
      mLinearLayout = new LinearLayout(this);

  // Instancia un ImageView y define sus propiedades
      ImageView i = new ImageView(this);
      i.setImageResource(R.drawable.circulo);

  // Agrega el ImageView al layout que lo contendrá
      mLinearLayout.addView(i);
      setContentView(mLinearLayout);
  }
```

Material Design

- Es un conjunto de guías el diseño visual y de interacción en distintas plataformas y dispositivos
- Disponible a partir del API 18

- https://developer.android.com/intl/es/design/material/index.html

Fragmentos

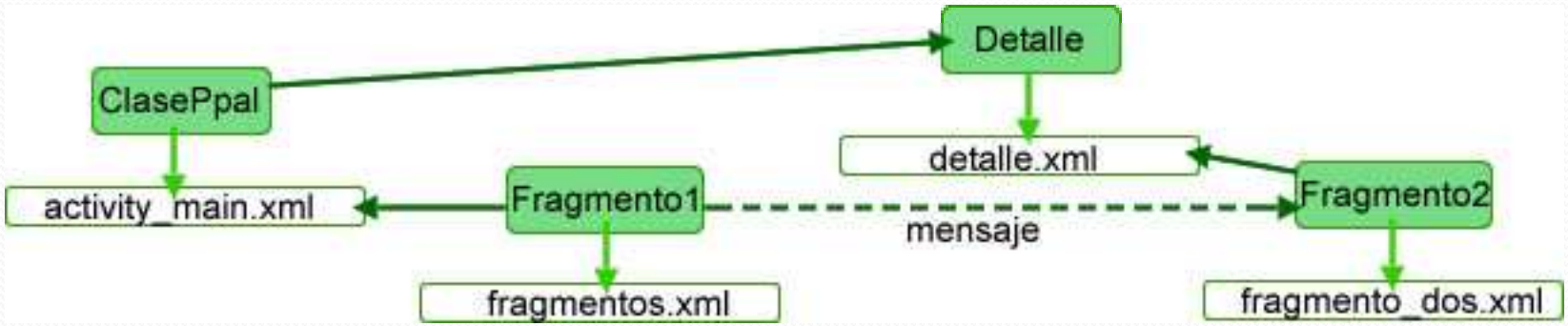
- El uso de fragmentos permite que una aplicación se adapte a diferentes pantallas
- Una aplicación puede usar un fragmento para mostrar un menú de opciones y otro para mostrar acciones
- En una pantalla grandes ambos fragmentos son visibles a la vez
- En un teléfono celular se necesitan dos actividades, una ligada al primer fragmento y otra ligada al segundo, la cual será visible cuando se presione algún botón

Imagen extraída de http://joomla.probando-cosas.com.ar/index.php/item/72-trabajando-con-fragments-interfaces-de-usuarios-ui-dinamicas-parte-1

Fragmentos

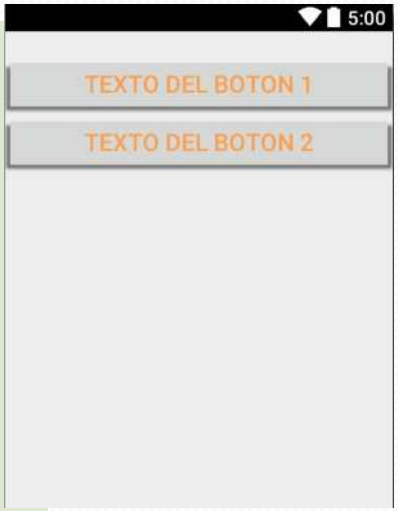
- Un fragmento es una subactividad que está ligado a una actividad, y a su ciclo de vida
  - En el archivo de diseño de la actividad se especifica en la línea *class* el Fragmento que se liga a ella.
- Cada fragmento requiere
  - Un archivo de diseño xml
  - Un archivo Java en el que la clase extiende la clase Fragment, sobreescribir los métodos onCreatView (que retorna algo de tipo view)

Clases Fragmentos y actividades



fragmento1 y fragmento2

```
<LinearLayout
    …
    android:orientation="vertical">
   <Button
        android:id="@+id/fragment1Button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:textColor="#FF9E49"
        android:text="@string/first_button_text"
        android:textSize="20sp" />
   <Button
        android:id="@+id/fragment1Button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#FF9E49"
        android:text="@string/second_button_text"
        android:textSize="20sp" />
</LinearLayout>
```




```
<LinearLayout   … >
    <TextView
        android:id="@+id/fragment2TextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:textSize="20sp" />
</LinearLayout>
```

Activity_main y detalle

```
<RelativeLayout  … >

   <fragment
         android:id="@+id/fragment1"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         class="com.example.pruebafragmentos.Fragmento1" />

</RelativeLayout>
```

Ligan dos clases que extienden Fragment

```
<RelativeLayout  … >

   <fragment
         android:id="@+id/fragment2"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         class="com.example.pruebafragmentos.Fragmento2" />

</RelativeLayout>
```

MainActivity y Fragmento1 en java

```
public class MainActivity extends Activity implements OnClickListener{
    Fragmento1 fragment1;
    Fragmento2 fragment2;
    TextView textView;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Button button1 = (Button) findViewById(R.id.fragment1Button1);
    Button button2 = (Button) findViewById(R.id.fragment1Button2);
    button1.setOnClickListener(this);
    button2.setOnClickListener(this);
} …….
}
```

```
public class Fragmento1 extends Fragment {

  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState){
       View view = inflater.inflate(R.layout.fragmento1, container);
       return view;
   }
}
```

MainActivity.java (continuación)

```
public class MainActivity extends Activity implements OnClickListener{

……

@Override
public void onClick(View v) {
    String str;
    if (v.getId() == R.id.fragment1Button1)
        str = "Presionaste el primer boton";
    else
        str = "Presionaste el segundo boton";

    Intent intent = new Intent (getApplicationContext(), Detalle.class);
    intent.putExtra("mensaje", str);
    startActivity(intent);
  }
}
```

Detalle y Fragmento2 en java

```
public class Detalle extends Activity{

    public void onCreate (Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.detalle);
        Bundle bundle = getIntent().getExtras();

       if (bundle != null){
          String str = bundle.getString("mensaje");
          TextView textView = (TextView) findViewById(R.id.fragment2TextView);
          textView.setText(str);
        }
    }
}
```

```
public class Fragmento2 extends Fragment {

  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState){
       View view = inflater.inflate(R.layout.fragmento2, container);
       return view;
   }
}
```

Manos a la Obra!