

# SEMINARIO DE LENGUAJES

## OPCIÓN ANDROID



**Intents.**

Mg. Corbalán Leonardo, Esp. Delía Lisandro

# Activación de Componentes

- En Android una aplicación puede iniciar un componente de otra aplicación.
- Sin embargo **no** puede hacerlo **en forma directa** porque cada aplicación se ejecuta en un proceso independiente con permisos de archivos que limitan el acceso a otras aplicaciones
- Para hacerlo debe **pedírselo al S.O.** por medio de un mensaje especificando la intención (***intent***) de iniciar un componente específico. Luego, es el propio Android el que activa ese componente.

# Ejemplo práctico

- Vamos a crear un proyecto con dos *activity*.
- Al presionar un botón de la *activity* principal se mostrará información de la aplicación en una *activity* distinta.

# Ejemplo práctico

- Crear un nuevo proyecto basado en un **Empty Activity** y definir el siguiente **layout**

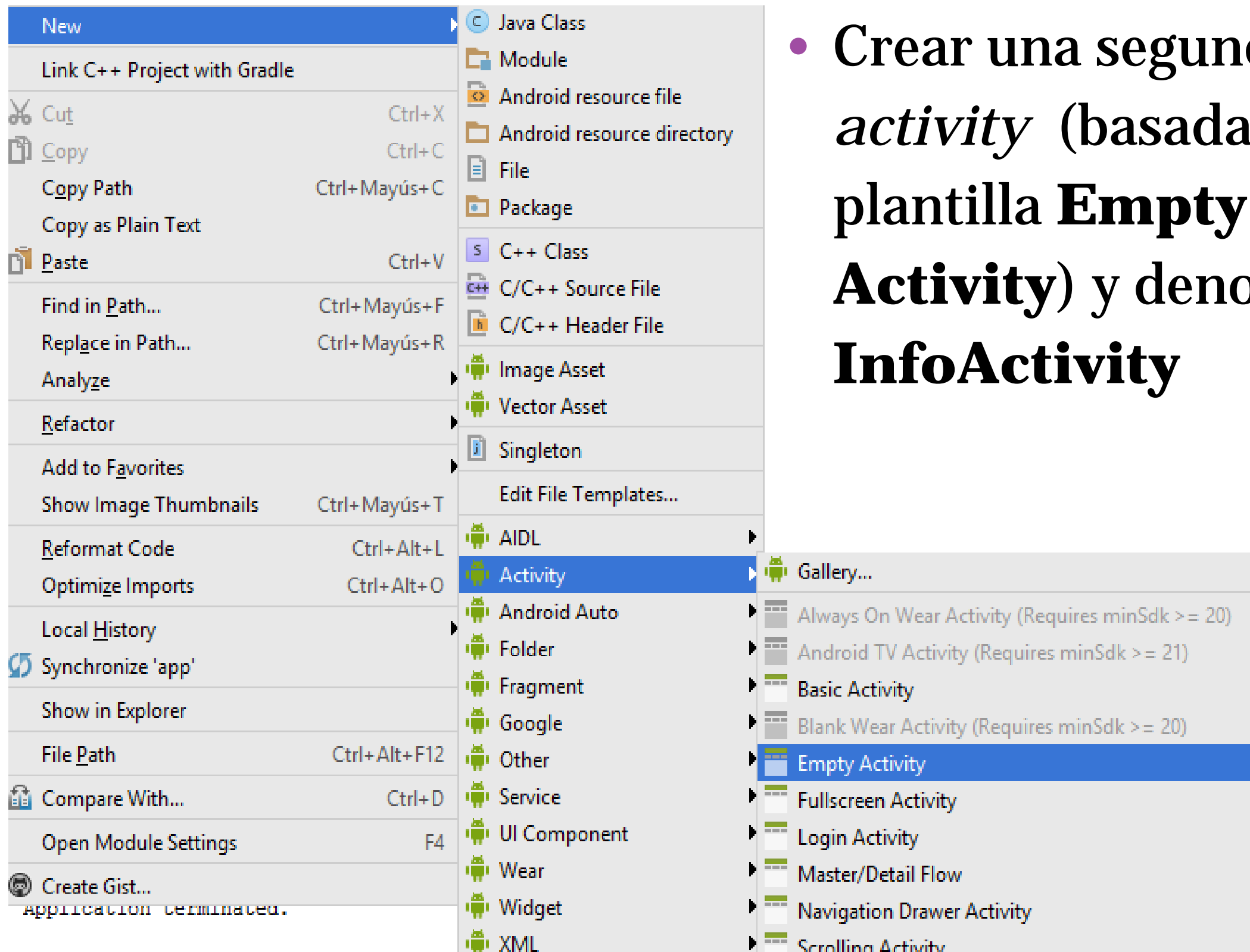
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Mostrar Información"
        android:onClick="mostrarInfo"
    />

</LinearLayout>
```

En **onClick** se especifica el nombre del método de la **activity** que se ejecutará al presionar el botón

# Ejemplo práctico



- Crear una segunda *activity* (basada en la plantilla **Empty Activity**) y denominarla **InfoActivity**

# Ejemplo práctico

- Definir este layout para **InfoActivity**

The screenshot displays the Android Studio interface with the `activity_info.xml` file open. The XML code defines a `LinearLayout` containing a `TextView`. A callout box highlights the background color `#70FF90` in the `LinearLayout` tag, with a legend showing its RGB components: R=70, G=FF, B=90. Another callout box points to the `android:id="@+id/texto"` attribute in the `TextView` tag, with the text "Importante: definir un id para el TextView". The right-hand side shows the visual preview of the layout on a Nexus 4 device, featuring a blue header "Intents" and a large green area with the text "Info del Curso de Android".

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout android:layout_width="match_parent"
3   android:layout_height="match_parent"
4   android:background="#70FF90"
5   xmlns:android="http://schemas.android.com/apk/
6
7   <TextView
8     android:layout_width="match_parent"
9     android:layout_height="wrap_content"
10    android:textSize="30sp"
11    android:text=" Info del Curso de Android "
12    android:id="@+id/texto"
13  />
14
15 </LinearLayout>
```

Callout 1: `"#70 FF 90"`  
R G B

Callout 2: Importante: definir un id para el TextView

Preview: Nexus 4, 25, 20% zoom. Text: Intents, Info del Curso de Android.

# Ejemplo práctico

- Ya tenemos la *activity* que queremos mostrar al presionar el botón "**Mostrar Información**" de la *activity* principal
- Ahora debemos codificar el manejador del evento **onClick** de dicho botón

Agregar el método **mostrarInfo** en la clase **MainActivity**  
Este método lo establecimos en el **onClick** del botón de esta *activity*  
(análisis de este código en la siguiente diapositiva)

```
Import android.view.View;
```

```
public class MainActivity extends AppCompatActivity {
```

```
    @Override
```

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

Importante: debe ser **public**

```
    public void mostrarInfo(View v)  
    {  
        Intent i = new Intent(this, InfoActivity.class);  
        startActivity(i);  
    }
```

```
}
```



Se crea el **intent** **i** que referencia a la *activity* que se va a iniciar

```
public void mostrarInfo(View v)
{
```

```
    Intent i = new Intent(this, InfoActivity.class);
```

```
    startActivity(i);
```

```
}
```

Se inicia la nueva *activity* por medio del **Intent** **i**

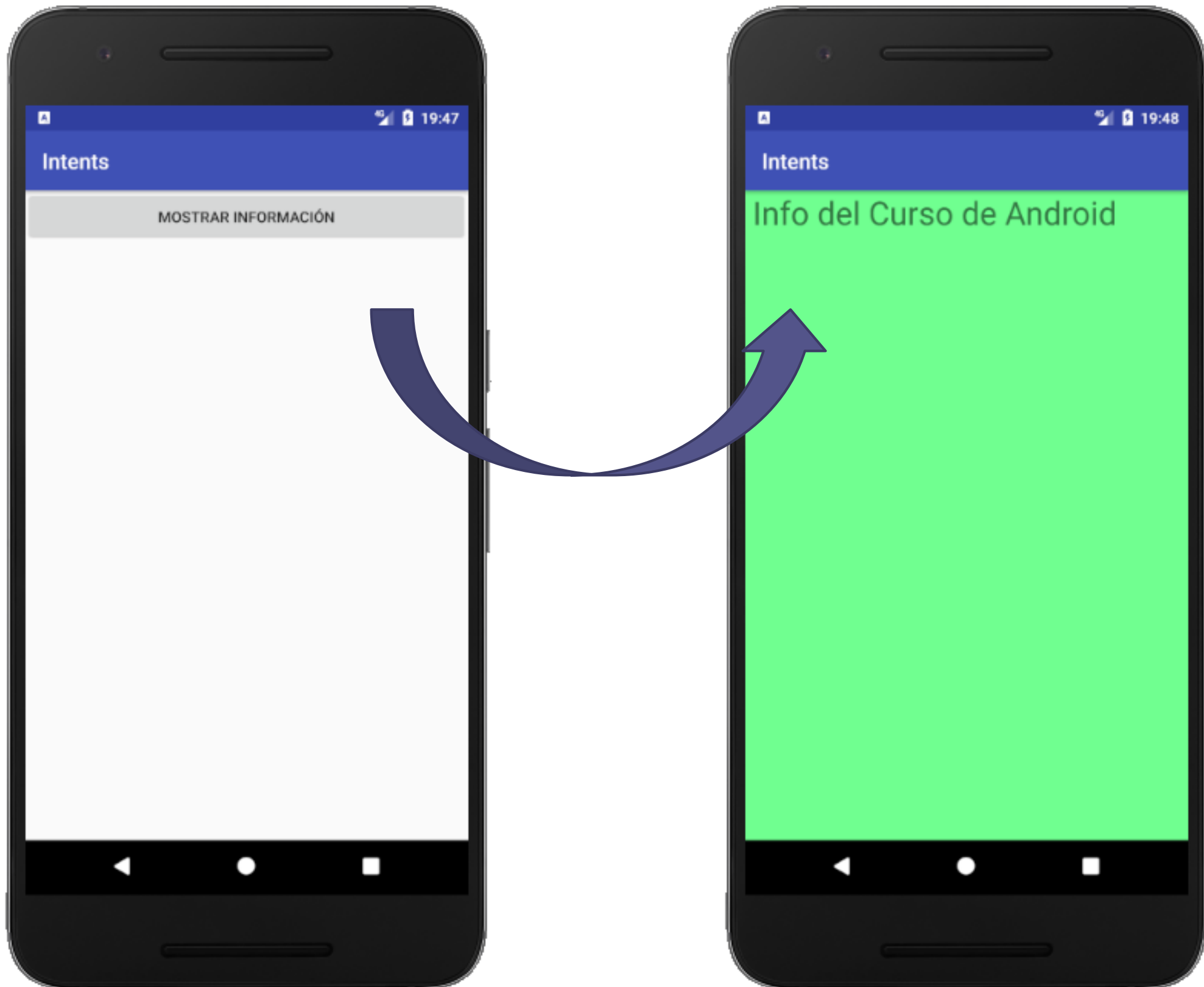
Contexto desde el que se va a iniciar **InfoActivity**, en este caso **this** hace referencia a la *activity* principal (instancia de **MainActivity**)

```
public void mostrarInfo(View v)
{
    Intent i = new Intent(this, InfoActivity.class);
    startActivity(i);
}
```

Referencia a la clase de la *activity* que se va a iniciar

Correr en el emulador  
para comprobar  
comportamiento

# Resultado en el emulador



# Pasando información a la *activity* iniciada

- Para pasar información a la *activity* se utiliza el mismo **Intent** con el que se la inicia.
- Se pueden pasar tantos datos como se requieran llamando repetidamente al método **putExtra()** del **Intent**
- **putExtra()** recibe dos parámetros: un **String** (a modo de clave) y el dato en cuestión.
- Los datos enviados podrán ser recuperados en la *activity* que se inicia por medio de un único objeto **Bundle** en el que se encuentran empaquetados todos los datos enviados.

## Modificar el método `mostrarInfo` de `MainActivity`


```
public void mostrarInfo(View v)
{
    Intent i = new Intent(this, InfoActivity.class);
    i.putExtra("dato1", "La cantidad de alumnos es: ");
    i.putExtra("dato2", 85);
    startActivity(i);
}
```

Agregar estas líneas para enviar a la *activity* que se inicia dos datos, un String y un entero

# Agregar en el método **onCreate** de **InfoActivity**

(análisis de este código en la siguiente diapositiva)

```
public class InfoActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_info);  
        Bundle datos = getIntent().getExtras();  
        if (datos != null)  
        {  
            String leyenda = datos.getString("dato1");  
            int cantidad = datos.getInt("dato2");  
            TextView texto = (TextView)findViewById(R.id.texto);  
            texto.setText(leyenda + cantidad);  
        }  
    }  
}
```



```
protected void onCreate(Bundle savedInstanceState)
```

```
super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.activity_info);
```

```
Bundle datos = getIntent().getExtras();
```

```
if (datos != null)
```

```
{
```

```
String leyenda = datos.getString("dato1");
```

```
int cantidad = datos.getInt("dato2");
```

```
TextView texto=(TextView)findViewById(R.id.texto);
```

```
texto.setText(leyenda + cantidad);
```

```
}
```

```
}
```

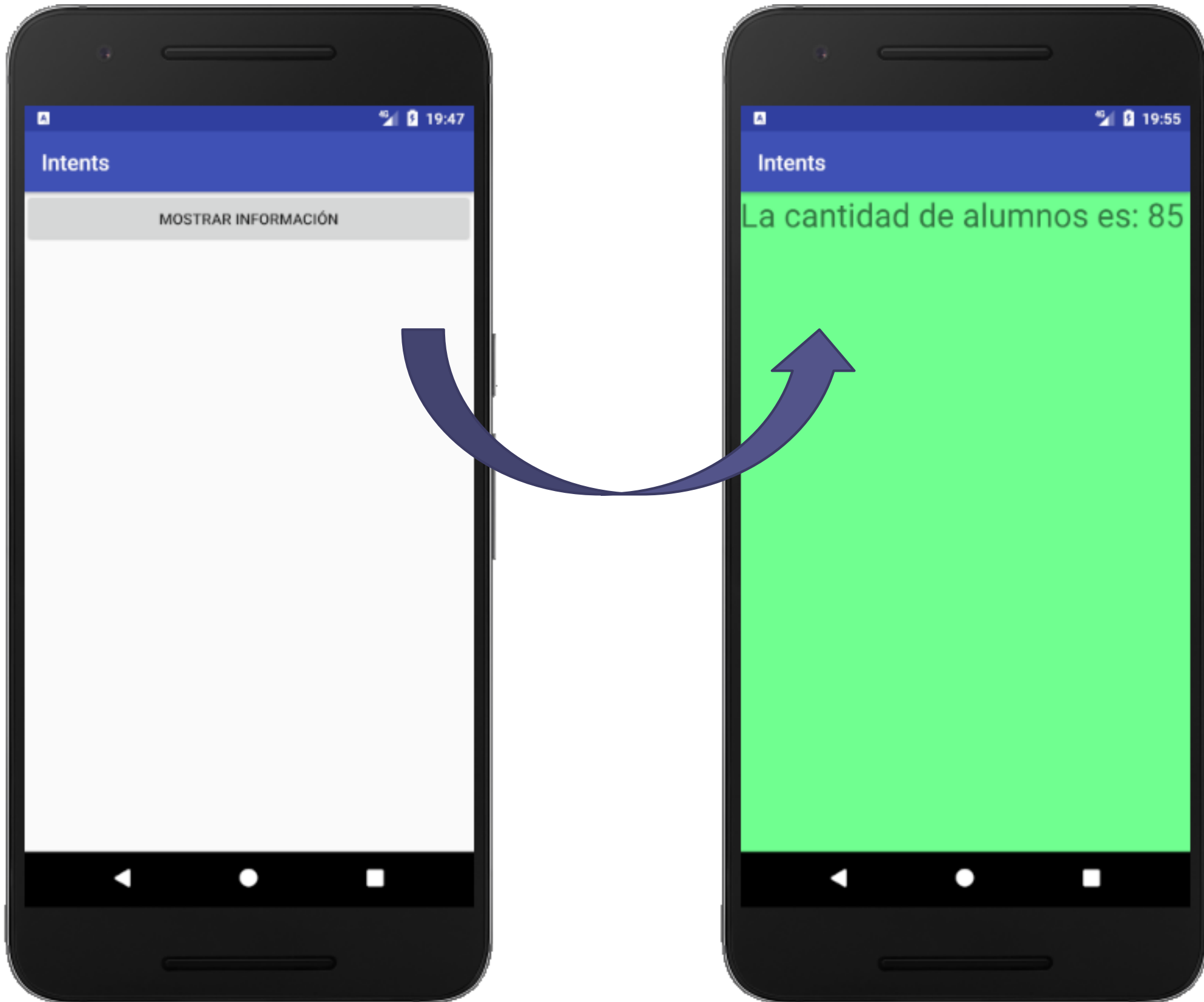
Obtenemos el **Bundle** desde el **intent** con el que se inició esta *activity*

Recuperamos del **Bundle** el **String** y el entero que se enviaron

Obtenemos la referencia al **TextView** y establecemos el texto a mostrar

Correr en el emulador

# Resultado en el emulador





# Devolviendo información a la *activity* iniciadora

- Para recibir un resultado desde la *activity* que se va a iniciar se debe utilizar el método **startActivityForResult()** en lugar de **startActivity()**
- El resultado es recibido por medio del método *callback* **onActivityResult()**

## Modificar el método `mostrarInfo` de `MainActivity`

```
public void mostrarInfo(View v)
{
    Intent i = new Intent(this, InfoActivity.class);
    i.putExtra("dato1", "La cantidad de alumnos es: ");
    i.putExtra("dato2", 85);
    startActivityForResult(i, 5);
}
```

A diagram consisting of three main parts. At the top, a dark teal box contains the title 'Modificar el método mostrarInfo de MainActivity'. A vertical arrow points down from this box to a light gray box containing the code snippet. From the bottom of the light gray box, an arrow points up to a dark teal box at the bottom containing an explanation of the 'RequestCode' parameter.

El segundo parámetro (**RequestCode**) es un entero que se utiliza para identificar la invocación y será retornado en el *callback* **onActivityResult()**, en este caso se eligió 5 arbitrariamente

## Agregar el método **onActivityResult** en **MaintActivity**



```
@Override  
  
protected void onActivityResult(int requestCode, int resultCode,  
                                Intent data) {  
  
    super.onActivityResult(requestCode, resultCode, data);  
    String leyenda = "requestCode: " + requestCode +  
                    " resultCode: " + resultCode;  
    Toast.makeText(this, leyenda, Toast.LENGTH_SHORT).show();  
}
```

Correr en el emulador y responder:

1. ¿En qué momento se invoca **onActivityResult()**?
2. ¿Cuáles son los valores devueltos en los parámetros **requestCode** y **resultCode**?

# Respuestas a las preguntas planteadas

- **onActivityResult()** es invocado cuando el usuario cierra **InfoActivity** y se reinicia **MainActivity**. Este método se ejecuta antes del método **onRestart()**.
- El parámetro **requestCode** recibido en **onActivityResult()** vale 5, y se corresponde con el valor utilizado en la llamada al método **startActivityForResult()**. El parámetro **resultCode** vale 0 (valor por defecto)

# Valor de RequestCode

- Notar que **MainActivity** podría invocar diferentes **intents** y esperar resultados de cada uno de ellos.
- Mediante el callback **onActivityResult()** se recibirán los resultados de cualquiera de los intents.
- El programador puede interpretar que el resultado recibido se corresponde a un **Intent** determinado según el **requestCode**

# Valores posibles de resultCode

- El parámetro resultCode obtenido en el método `onActivityResult()` es un entero, por lo tanto el programador puede establecer arbitrariamente tanto su valor como su interpretación.
- Sin embargo se ha convenido lo siguiente :  
**0 = Cancelado      -1 = Aceptado.**
- En `Activity.java` se encuentran definidas las siguientes constantes:

```
/** Standard activity result: operation canceled. */  
public static final int RESULT_CANCELED      = 0;  
/** Standard activity result: operation succeeded. */  
public static final int RESULT_OK            = -1;
```

# Devolviendo información a la *activity* iniciadora

- En `Activity_info.xml` establecer la orientación del `LinearLayout` en **vertical** y agregar el botón **"Aceptar"**

`<Button`

```
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:onClick="aceptarYcerrar"  
    android:text="Aceptar" />
```

- En `InfoActivity.java` codificar el método

Correr en el  
emulador

```
public void aceptarYcerrar(View v) {  
    setResult(RESULT_OK) ;  
    this.finish() ;  
}
```

Establece resultado

Cierra la *activity*

# Devolviendo información a la *activity* iniciadora

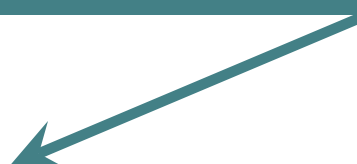
- Vamos a modificar la aplicación que estamos construyendo agregando un **EditText** en **InfoActivity** para que el usuario pueda tipear un texto en él.
- Sólo si el usuario presiona el botón "Aceptar", una vez cerrada **InfoActivity**, **MainActivity** debe mostrar en un mensaje **Toast** el texto tipeado por el usuario.



# Devolviendo información a la *activity* iniciadora

- Agregar el **EditText** en **Activity\_info.xml**

Es necesario establecer un **id** para luego poder referenciarlo desde el código java



```
<EditText  
    android:id="@+id/editor"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
/>
```

# Devolviendo información a la *activity* iniciadora

- En **InfoActivity.java** modificar el método

```
public void aceptarYcerrar(View v) {
```

```
    EditText editor = (EditText) findViewById(R.id.editor);
```

```
    String texto = editor.getText().toString();
```

```
    Intent i = new Intent();  
    i.putExtra("info", texto);
```

```
    setResult(RESULT_OK, i);
```

```
    this.finish();
```

```
}
```

Obtiene referencia al editor

Obtiene el texto  
ingresado

Crea un **intent** con la información  
para pasar

Establece resultado y el **intent** con  
la información

# Devolviendo información a la *activity* iniciadora

- En **MainActivity.java** modificar el método

@Override

```
protected void onActivityResult(int requestCode, int resultCode,  
                                Intent data) {
```

```
    super.onActivityResult(requestCode, resultCode, data);  
    if (resultCode == RESULT_OK )  
    {  
        String leyenda = data.getExtras().getString("info");  
        Toast.makeText(this, leyenda, Toast.LENGTH_SHORT).show();  
    }
```

```
}
```

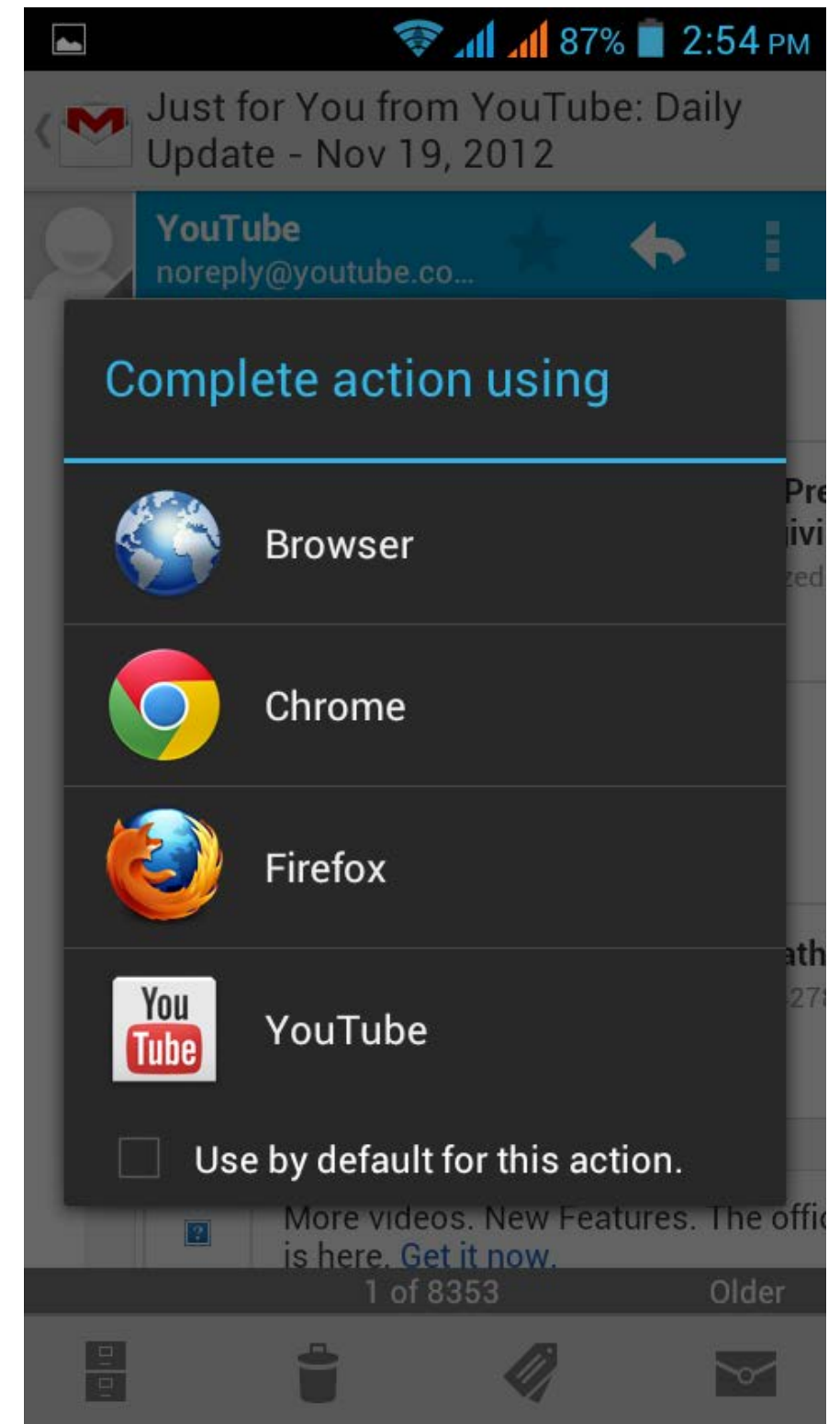
Correr en el  
emulador

# Intent para llamar *activities* de otras aplicaciones

- Hemos utilizado un objeto **Intent** para iniciar una *activity* de nuestra aplicación de manera **explícita** (**InfoActivity.class**)
- También existen los **Intent implícitos** que NO especifican el componente que se desea iniciar sino que sólo describen una **acción** a realizar y opcionalmente los datos sobre los que trabajar

# Intent para llamar *activities* de otras aplicaciones

- Los Intent **implícitos** delegan en Android la búsqueda de un componente en el dispositivo que puedan realizar la acción especificada. Si hay más de un componente en esas condiciones se presenta un diálogo al usuario para que elija cuál de todos iniciar.



# Intent para llamar *activities* de otras aplicaciones

- Android identifica los componentes que pueden responder a un **intent** comparando el contenido del **intent** con los *filtros de intents* que se proporcionan en el archivo de manifiesto de otras aplicaciones instaladas en el sistema

# Contenido de los *intents* utilizado para comparar con los filtros de *intents*

- **Acción:** un string que especifica una acción genérica. Dependiendo de la acción es posible determinar si son necesarios datos extras.
- **Datos:** El objeto **Uri** que hace referencia a los datos en que se debe realizar la acción o el tipo **MIME** de esos datos.
- **Categoría:** un string que contiene información adicional sobre el tipo de componente que debe iniciarse. Un **intent** puede tener varias categorías pero la mayoría de ellos no requieren ninguna

# Ejemplo de código que utiliza un **Intent** implícito para iniciar una *activity* de otra aplicación

```
Intent i = new Intent();
```

Se establece la acción del Intent

```
i.setAction(Intent.ACTION_SEND);
```

```
i.setType("text/plain");
```

Se establece el tipo MIME de los datos que se enviarán

```
i.putExtra(Intent.EXTRA_TEXT, "texto que se envía");
```

Se agrega al Intent el texto que se enviará a la *activity*

```
if (i.resolveActivity(getPackageManager()) != null) {  
    startActivity(i);  
}
```

Se verifica que exista alguna *activity* en el sistema que pueda resolver el **intent** antes de iniciarla, de lo contrario **startActivity()** fallará



# Ejemplo de código que utiliza un **Intent** implícito para iniciar una *activity* de otra aplicación

```
Intent i = new Intent();  
i.setAction(Intent.ACTION_SEND);  
i.setType("text/plain");  
i.putExtra(Intent.EXTRA_TEXT, "texto que se envía");  
  
if (i.resolveActivity(getPackageManager()) != null) {  
    startActivity(i);  
}
```

Es una constante de tipo string  
"android.intent.action.SEND"

En la clase **Intent** existen muchas  
otras constantes de acciones  
definidas

Es una constante de tipo string  
"android.intent.extra.TEXT"

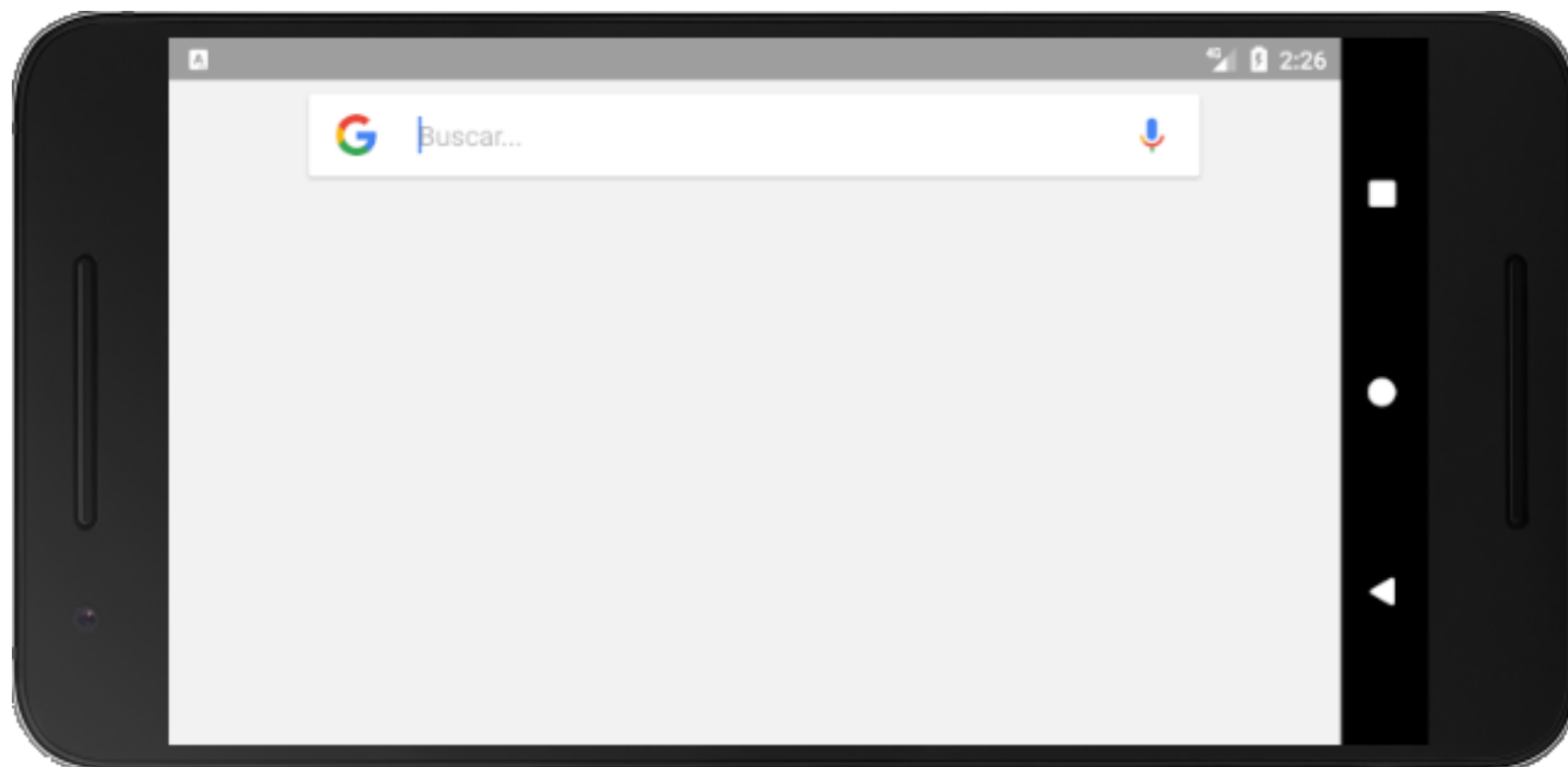
La clase **Intent** especifica muchas constantes  
EXTRA\_\* para tipos de datos estandarizados

# *Intents* comunes

- Existe una gran cantidad de *intents* implícitos que pueden usarse para realizar acciones comunes.
- Modifique la aplicación para implementar los ejemplos que se muestran a partir de la próxima diapositiva.

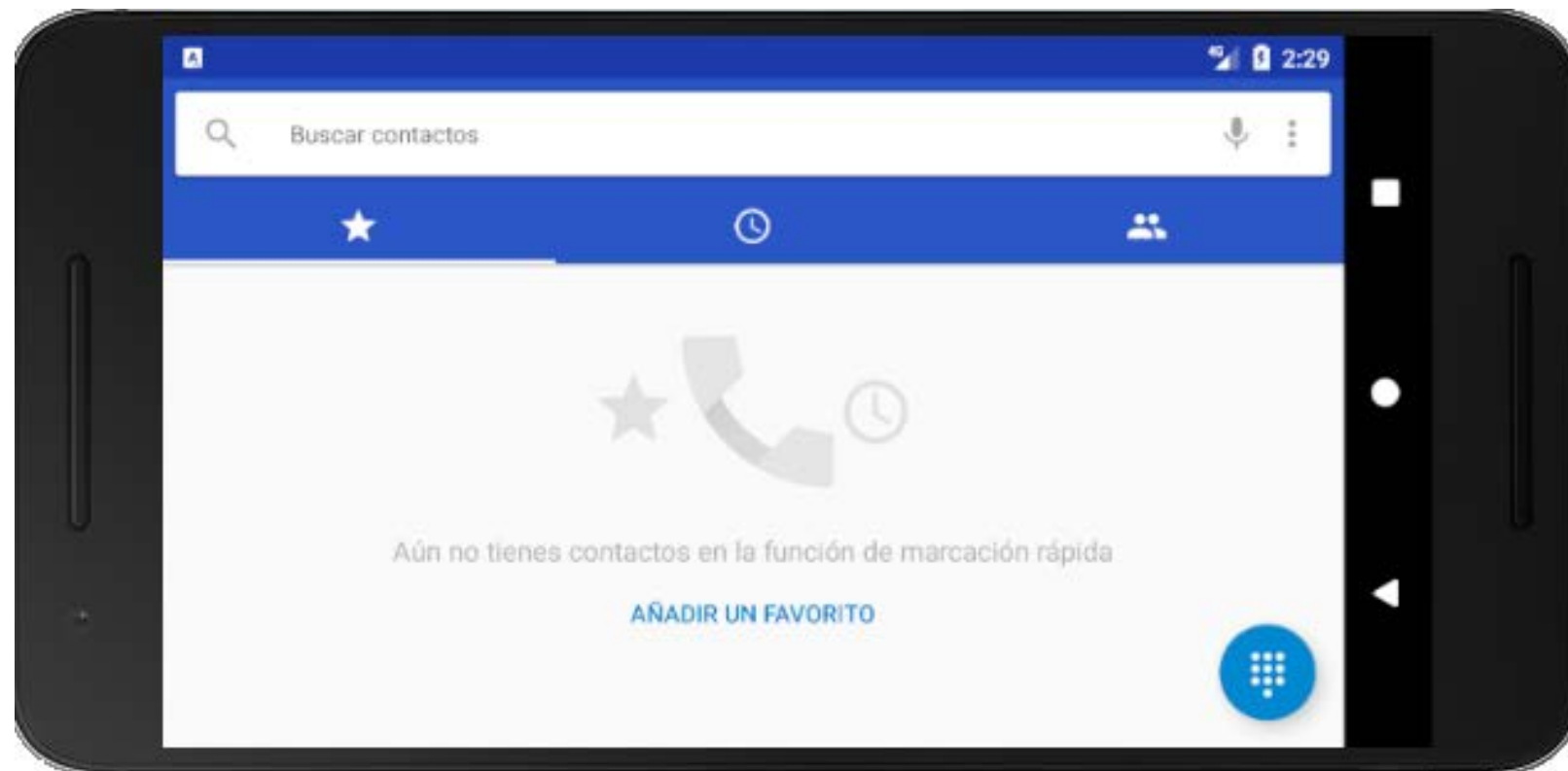
# *Intents* comunes

```
Intent i = new Intent();  
i.setAction(Intent.ACTION_WEB_SEARCH);  
if (i.resolveActivity(getPackageManager()) != null) {  
    startActivity(i);  
}
```



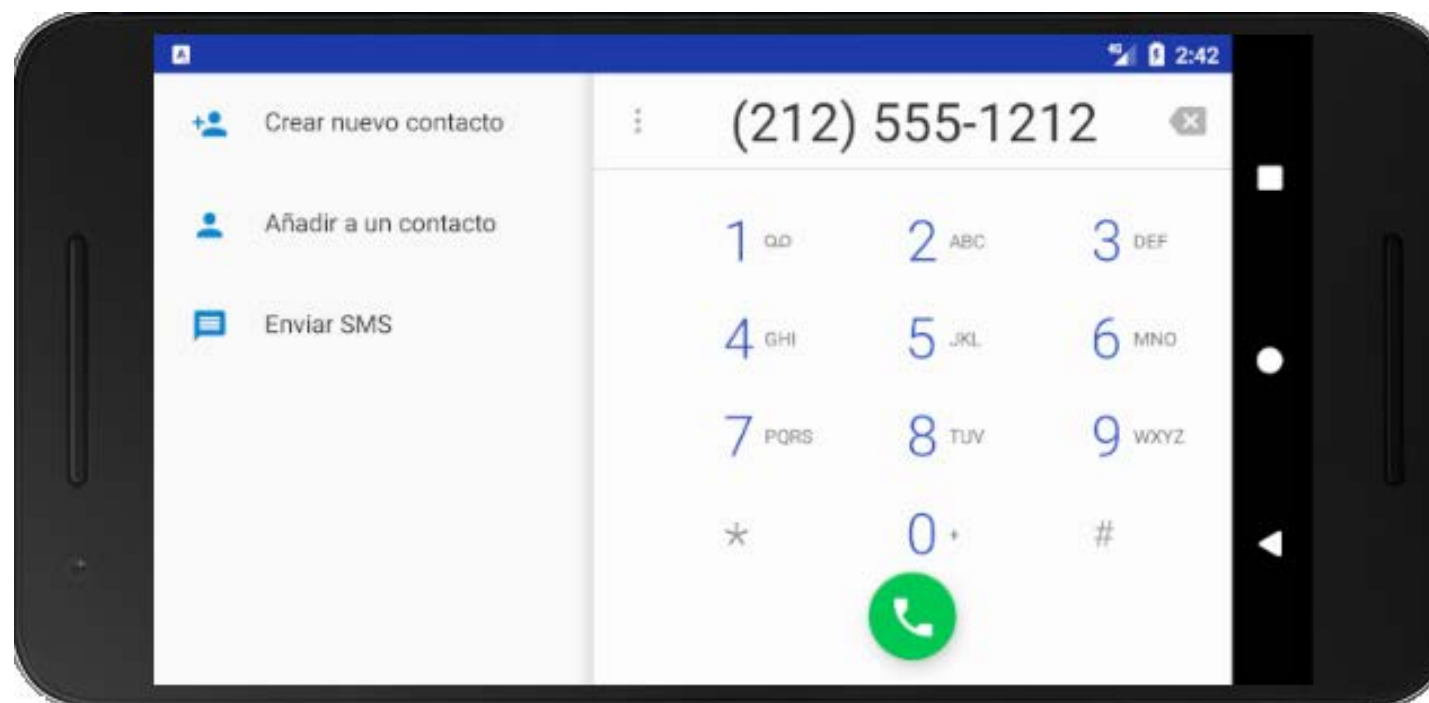
# Intents comunes

```
Intent i = new Intent();  
i.setAction(Intent.ACTION_DIAL);  
if (i.resolveActivity(getPackageManager()) != null) {  
    startActivity(i);  
}
```



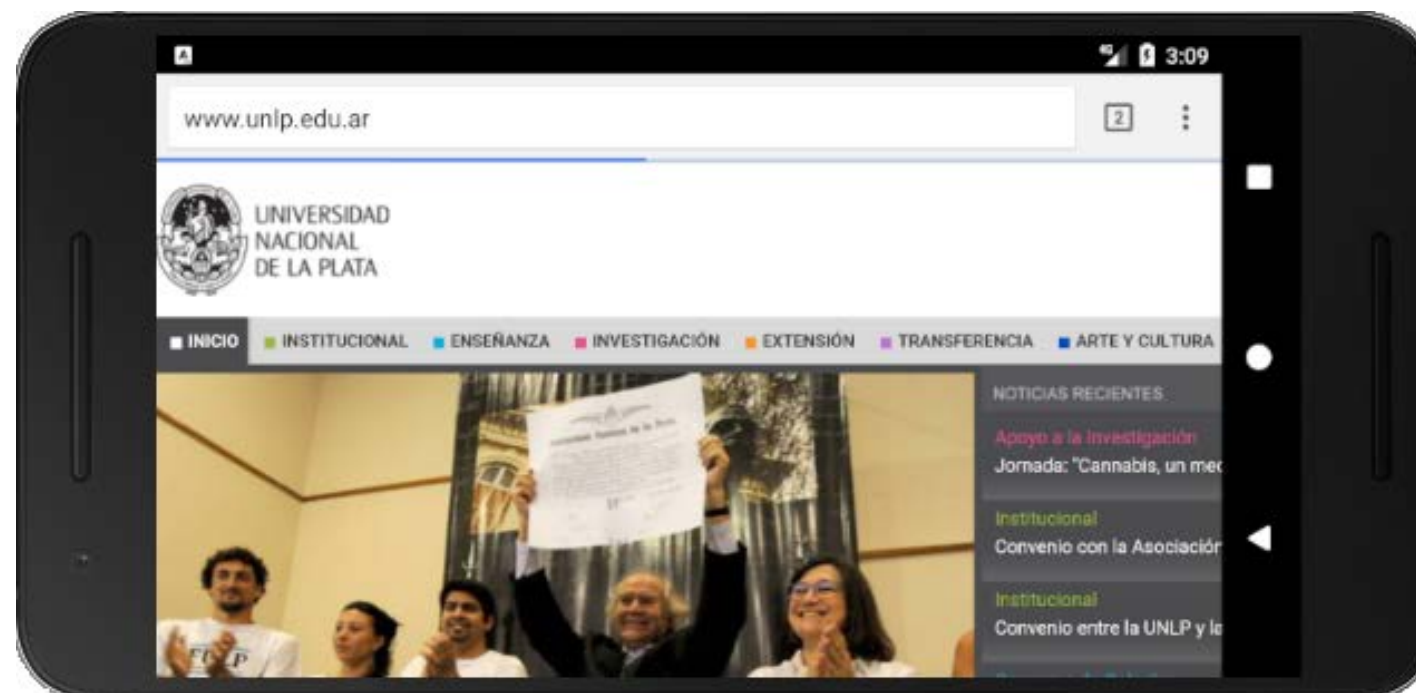
# Intents comunes

```
Intent i = new Intent();  
i.setAction(Intent.ACTION_DIAL);  
i.setData(Uri.parse("tel:2125551212"));  
if (i.resolveActivity(getPackageManager()) != null) {  
    startActivity(i);  
}
```



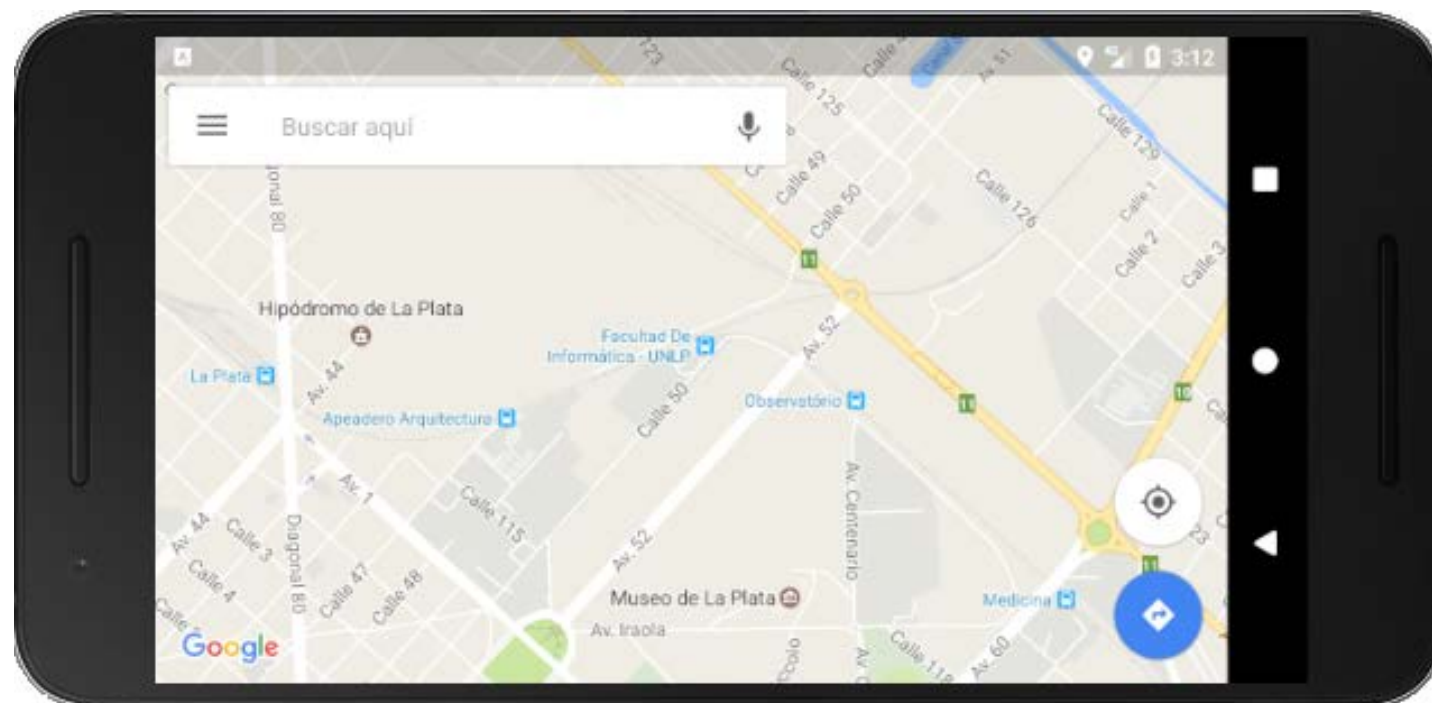
# Intents comunes

```
Intent i = new Intent();  
i.setAction(Intent.ACTION_VIEW);  
i.setData(Uri.parse("http://www.unlp.edu.ar"));  
if (i.resolveActivity(getPackageManager()) != null) {  
    startActivity(i);  
}
```



# Intents comunes

```
Intent i = new Intent();  
i.setAction(Intent.ACTION_VIEW);  
i.setData(Uri.parse("geo:-34.903735,-57.938081"));  
if (i.resolveActivity(getPackageManager()) != null) {  
    startActivity(i);  
}
```



# Intents comunes

```
Intent i = new Intent();  
i.setAction(Intent.ACTION_SEND);  
i.setType("text/plain");  
i.putExtra(Intent.EXTRA_TEXT, "texto que se envía");  
if (i.resolveActivity(getPackageManager()) != null) {  
    startActivity(i);  
}
```

La acción SEND también se la conoce como compartir.  
Dependiendo de las aplicaciones que se encuentren instaladas en el dispositivo, se mostrará al usuario un cuadro de diálogo con una lista de posibles elecciones



# Modificando **InfoActivity** para que pueda iniciarse desde otra app

- Vamos a crear un filtro de *intents* adecuado para la *activity* **InfoActivity** para que otra aplicación pueda iniciarla
- Vamos a definir nuestra propia acción con el string "GESTION\_INFO"
- Luego crearemos otra aplicación que utilizará un Intent implícito para solicitar a Android que inicie la activity que pueda realizar la función "GESTION\_INFO"

# Modificando **InfoActivity** para que pueda iniciarse desde otra app

En **androidManifest.xml** buscar la etiqueta correspondiente a **infoActivity** y agregar el siguiente filtro:

Correr en el emulador para actualizar la app en el dispositivo virtual

```
<activity android:name=".InfoActivity">  
  <intent-filter>  
    <action android:name="GESTION_INFO" />  
    <category android:name="android.intent.category.DEFAULT" />  
  </intent-filter>  
</activity>
```

Debe incluir la categoría **CATEGORY\_DEFAULT** en el filtro de *intents* porque los métodos **startActivity()** y **startActivityForResult()** tratan todos los *intents* como si pertenecieran a la categoría **CATEGORY\_DEFAULT**. Si no declara esta categoría en el filtro de *intents*, no se aplicará ningún intent implícito a la *activity*

# Creando una nueva aplicación

Crear una nueva aplicación basada en la plantilla **Empty Activity** y definir la siguiente interface

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:orientation="vertical"
    >
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Ver Info"
        android:onClick="verInfo"
        />
</LinearLayout>
```

# Creando una nueva aplicación

Codificar el método **verInfo()** de la siguiente manera:

```
public void verInfo(View v) {  
    Intent i = new Intent();  
    i.setAction("GESTION_INFO");  
    i.putExtra("dato1", "El año mostrado es: ");  
    i.putExtra("dato2", 2000);  
    if (i.resolveActivity(getPackageManager()) != null) {  
        startActivity(i);  
    }  
}
```

Correr en el  
emulador