Comunicando actividades

Intent explícito

Intent

- Es uno de los **componentes básicos** que conforman las aplicaciones Android. **Sirven para invocar desde un componente a otro componente**.
- Permiten llamar a aplicaciones externas a la nuestra, lanzar eventos a los que otras aplicaciones puedan responder, lanzar alarmas etc.

- Por ejemplo, un *Intent* permite indicarle a Android que se desea hacer algo a través de otra actividad que se construyó específicamente para esa situación, pudiendo ser esa otra actividad de otra aplicación o no.
- **Permiten enviar o recibir información desde y hacia otros componentes** (ya que no se comparten las variables).

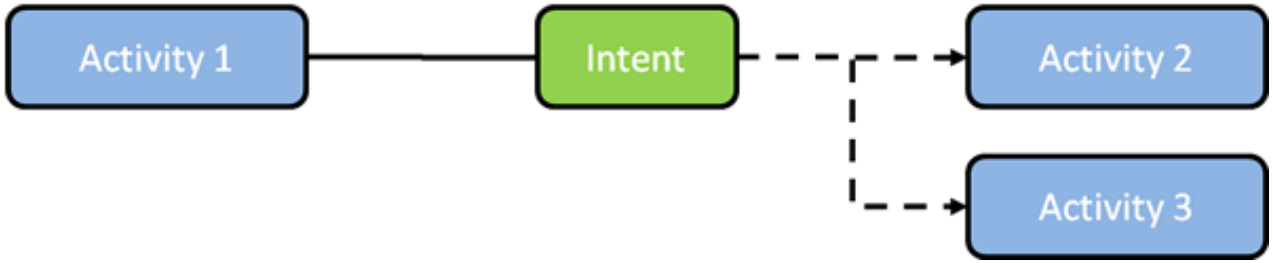
Uso de Intents

- Hay 3 principales usos de las intenciones:
  - Para **iniciar una actividad**: facilitan la navegación entre actividades. Para iniciar una instancia de una nueva actividad se usa el método **startActivity(Intent)**, que recibe como parámetro una intención con todos los datos necesarios para iniciar esta nueva actividad (nombre de la nueva actividad, datos extra que queremos pasar entre actividades…) o el método **startActivityForResult(Intent)**, que permite recibir resultados en el método **onActivityResult()**

  - Para **iniciar un servicio**: Un servicio es un componente que realiza operaciones en segundo plano sin necesidad de una interfaz gráfica. Se inician con el método **startService(Intent)**.
  - Para **enviar un anuncio ("broadcast")**: Un anuncio es enviar un mensaje sin destinatario fijo, es decir, cualquier app puede recibirlo. El sistema Android envía anuncios para eventos del sistema, por ejemplo, para indicar batería baja o que el móvil esta empezando a cargar…

Tipos de Intent

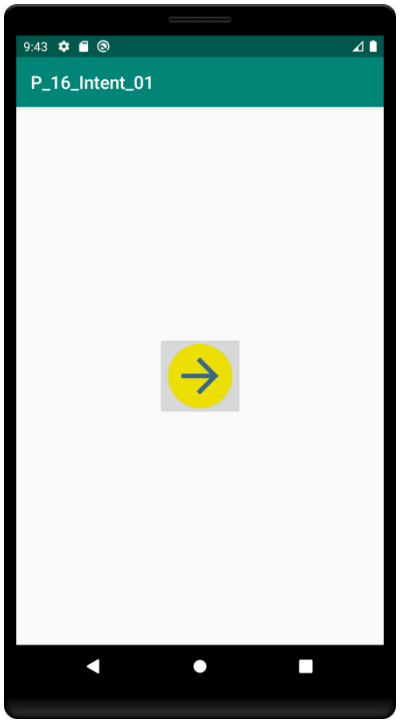
- Existen dos tipos de Intents:
  - Los Intents **explícitos** que **nombran directamente el componente que se necesita**, por ejemplo, especifican la clase que será utilizada para realizar una determinada tarea. Su utilización típica es la de ir ejecutando los diferentes componentes de una aplicación.

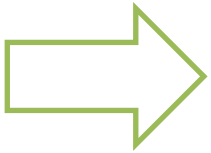


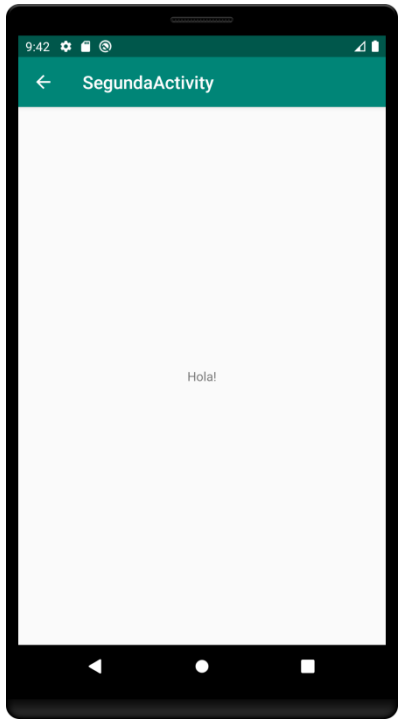
  - Los Intents **implícitos**, al realizar una petición, **dejan la selección del componente a utilizar en manos del sistema.** Por ejemplo, se podría indicar la intención de enviar un email y el sistema en tiempo de ejecución busca que aplicaciones tiene instaladas para poder hacerlo.

Ejemplo de Intent explícito básico

- Deseamos una aplicación que al arrancar muestre una pantalla con un botón (la MainActivity) y al pulsar muestre otra pantalla con el mensaje "Hola!" (la SegundaActivity):

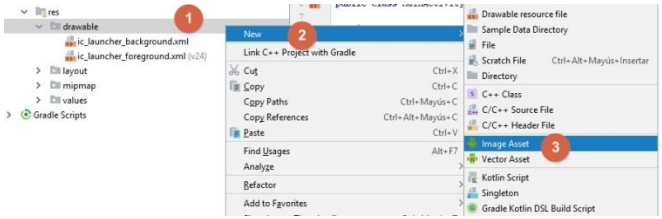


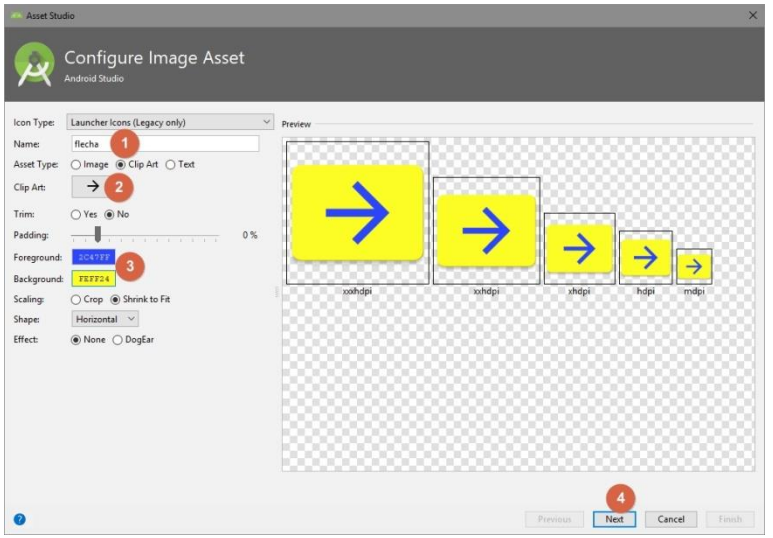




Trabajando con ImageButton

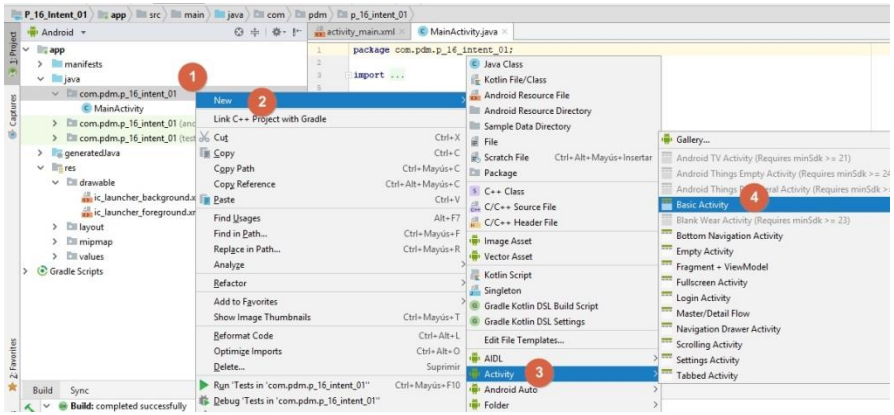
- La propiedad src adjudica la imagen del control ImageButton. Dicha imagen:
  - se puede copiar a la carpeta drawable (recuerda el plugin "Android drawable Importer" ya que debes tenerla en varios tamaños para las distintas densidades)
  - utilizar una del sistema
  - o crearla desde el asistente:

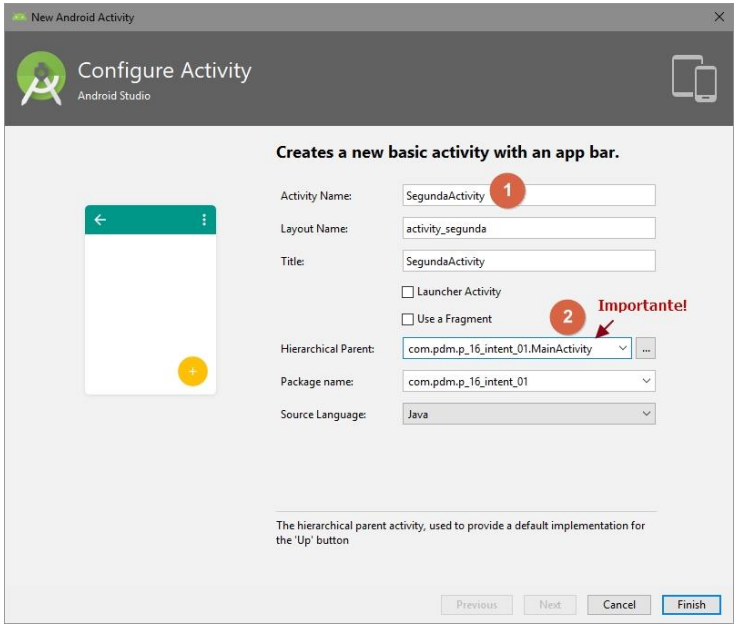




Segunda Pantalla

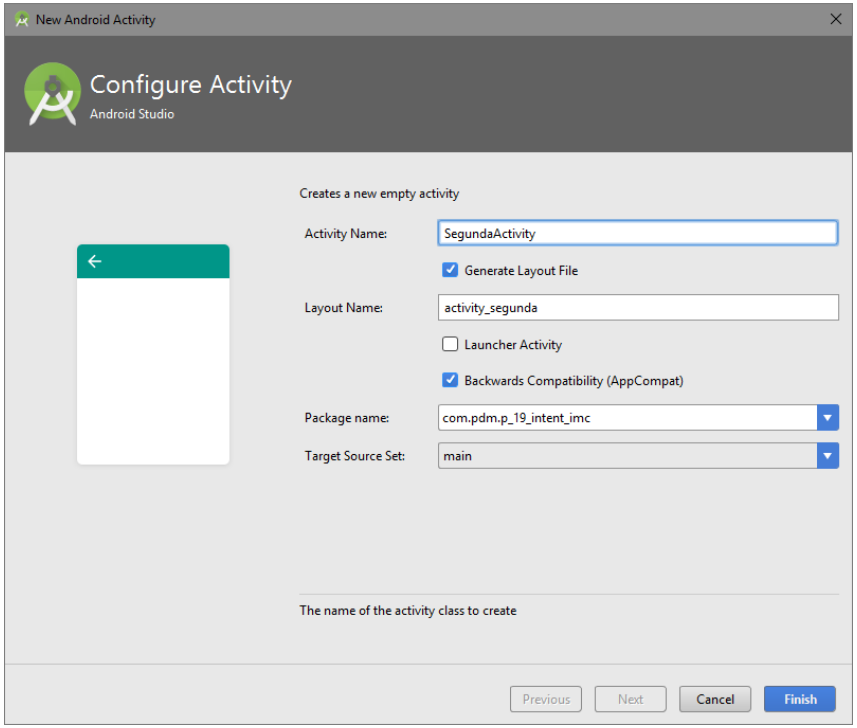
- Obviamente, para poder llamar a otra actividad, dicha actividad debe existir.
- Podemos usar el asistente ya que se encarga de crear el layout, el código básico y de incluirla en el Manifest.
- Nuestro trabajo será retocar el layout y poco más!



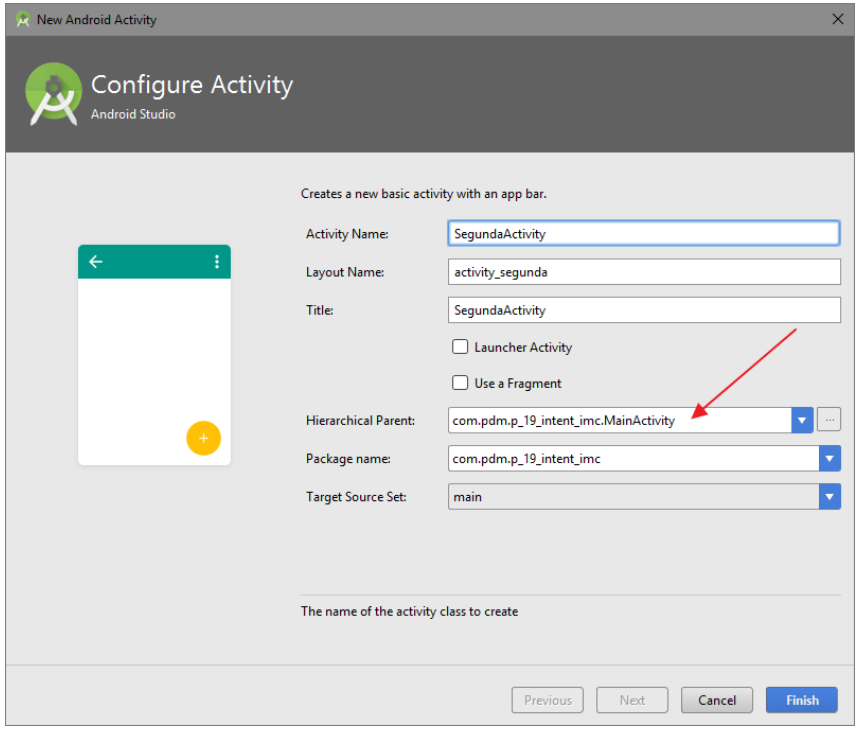


Plantilla para 2ª actividad?

Empty



Basic



Características de la plantilla de 2ª actividad?

Empty

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
…
   <application
…
      <activity android:name=".SegundaActivity"></activity>
   </application>

</manifest>
```

Podremos volver a la actividad anterior a la que estábamos:

Basic

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
…
   <application
…
      <activity
         android:name=".SegundaActivity"
         android:label="@string/title_activity_segunda"
         android:parentActivityName=".MainActivity"
         android:theme="@style/AppTheme.NoActionBar">
         <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.pdm.p_19_intent_imc.MainActivity" />
      </activity>
   </application>

</manifest>
```

Podremos volver a la actividad "padre" de la aplicación desde la "toolbar":

Intent explícito básico


```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ImageButton imageButton=findViewById(R.id.imageButton);
        imageButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent=new Intent(getApplicationContext(),SegundaActivity.class);
                startActivity(intent);
            }
        });
    }
```

Utilizamos el ***Intent*** explícito para construir la intención de llamar a una actividad desde otra actividad de la misma aplicación, para lo que pasaremos a su constructor una referencia a la propia *actividad llamadora* (getApplicationContext()) y la clase de la *actividad llamada* (SegundaActivity.class).

Con el método **startActivity()** se pone en marcha la actividad fijada en el Intent como parámetro.

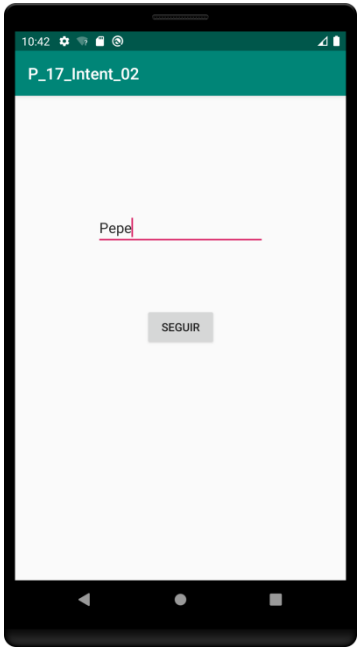
Manifest

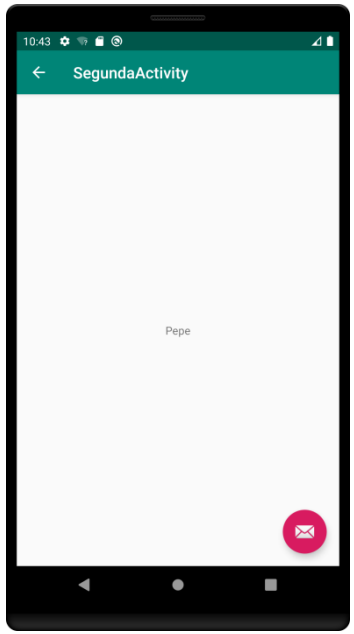

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.pdm.p_16_intent_01">

  <application
    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/AppTheme">
    <activity
      android:name=".MainActivity"
      android:label="@string/app_name"
      android:theme="@style/AppTheme.NoActionBar">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity
      android:name=".SegundaActivity"
      android:label="@string/title_activity_segunda"
      android:parentActivityName=".MainActivity"
      android:theme="@style/AppTheme.NoActionBar">
      <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="com.pdm.p_16_intent_01.MainActivity" />
    </activity>
  </application>

</manifest>
```

Si no te gustan los nombres que aparecen en la ActionBar (el del proyecto y el de la segunda actividad) pueden cambiarse desde los atributos android:label de las declaraciones de las respectivas actividades. Para ello crea los títulos deseados en el fichero strings.xml y aplícalos.

Ejemplo de Intent explícito con envío de datos

- La primera actividad nos solicita teclear un nombre y en la segunda se visualiza.
- La primera actividad necesita enviar a la segunda el dato del nombre tecleado (con objetos Bundle).





Clase Bundle

- Proporciona un contenedor para el almacenamiento de datos utilizando un mecanismo de pares clave-valor.
- Las claves tienen la forma de valores de cadena, mientras que los valores asociados pueden ser en forma de un valor simple o cualquier objeto que implemente la interfaz Parcelable.
- La clase Bundle también contiene un conjunto de métodos que se pueden utilizar para obtener y definir pares clave-valor para una variedad de tipos de datos que incluye los tipos primitivos (incluyendo valores Boolean, char, double y float) y objetos (tales como Strings y CharSequences).
- Ejemplo:

```
Bundle bundle = new Bundle();
bundle.putString("nombre", txtNombre.getText().toString());
bundle.putInt("edad",8);

bundle.getString("nombre");
```

Envío y recepción de datos entre actividades

- La clase Intent tiene el método **putExtra()** que permite **añadir 1 dato** al *intent:*

```
intent.putExtra("nombreTecleado", et1.getText().toString());
```

Clave (nombre) del dato enviado

Valor del dato enviado

- Si la actividad "emisora" de datos necesita pasar **más de uno**, se trabaja con objetos **Bundle** (conjunto de datos) y sus métodos **putString(), putInt(),…** y se envía con el método **putExtras()** de la clase Intent:

```
Bundle bundle = new Bundle();
bundle.putString("nombreTecleado", txtNombre.getText().toString());
bundle.putString("direccion", et1.getText().toString());
intent.putExtras(bundle);
```

- La actividad "receptora" de los datos **siempre** lo hace de la siguiente manera:
  - Recibe el conjunto de datos: Bundle bundle = getIntent().getExtras();
  - Obtiene un determinado dato: bundle.getString("nombreTecleado");

Código clases

- Código **MainActivity.java**

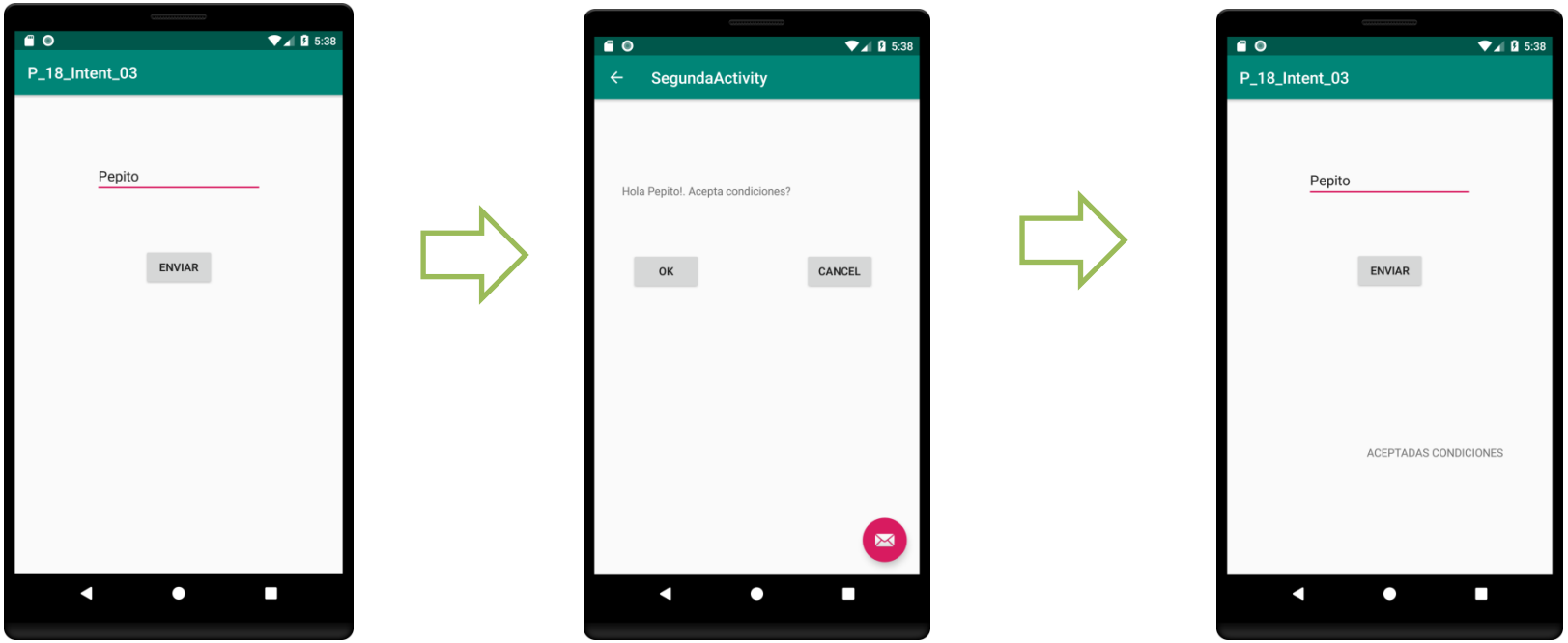
```
public class MainActivity extends AppCompatActivity  {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      final EditText editText= findViewById(R.id.editText);
      Button button= findViewById(R.id.button);
      button.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            Intent intent = new Intent(getApplicationContext(), SegundaActivity.class);
            intent.putExtra("nombreTecleado", editText.getText().toString());
            startActivity(intent);
         }
      });
   }
}
```

- Código **SegundaActivity.java**

```
public class SegundaActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_segunda);
      …
      TextView textview=findViewById(R.id.textview);
      Bundle bundle = getIntent().getExtras();
      textview.setTextl(bundle.getString("nombreTecleado"));
   }
}
```

Ejemplo Intent explícito con envío y devolución de datos

- La 1ª actividad envía datos a la segunda que a su vez devuelve otros datos a la primera.



Actividades intercomunicadas


```
public class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      final EditText editText=findViewById(R.id.editText);
      Button button=findViewById(R.id.button);
      button.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            Intent intent=new Intent(getApplicationContext(),SegundaActivity.class);
            intent.putExtra("nombre", editText.getText().toString());
            startActivityForResult(intent,1234);
         }
      });
   }

   protected void onActivityResult(int requestCode,int resultCode, Intent intent) {
      String res="Sin resolver";
      if(requestCode==1234 && resultCode==RESULT_OK){
         res=intent.getExtras().getString("resultado");
      }
      TextView textView=findViewById(R.id.textView);
      textView.setText(res);
   }

}
```

```
public class SegundaActivity extends AppCompatActivity implements
        View.OnClickListener {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_segunda);
      String nombre= getIntent().getStringExtra("nombre");
      TextView textView=findViewById(R.id.textView2);
      textView.setText(String.format("Hola %s!. %s", nombre,
         getString(R.string.texto4)));
      Button button=findViewById(R.id.button2);
      button2.setOnClickListener(this);
      Button button3=findViewById(R.id.button3);
      button3.setOnClickListener(this);
   }

   @Override
   public void onClick(View v) {
      String resultado = "RECHAZADAS CONDICIONES";
      if (v.getId()== R.id.button2)
         resultado = "ACEPTADAS CONDICIONES";
      Intent intent = new Intent();
      intent.putExtra("resultado", resultado);
      setResult(RESULT_OK, intent);
      finish();
   }
}
```

1ª actividad


```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText editText=findViewById(R.id.editText);
        Button button=findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent=new
                Intent(getApplicationContext(),SegundaActivity.class);
                intent.putExtra("nombre", editText.getText().toString());
                startActivityForResult(intent,1234);
            }
        });
    }

    protected void onActivityResult(int requestCode,int resultCode, Intent
            intent) {
        String res="Sin resolver";
        if(requestCode==1234 && resultCode==RESULT_OK){
            res=intent.getExtras().getString("resultado");
        }
        TextView textView=findViewById(R.id.textView2);
        textView.setText(res);
    }

}
```

- El método **startActivityForResult()** arranca la siguiente actividad pasando como parámetro el intent y un código de petición que se utilizará a la vuelta para identificar qué subactividad es la que ha terminado.

  startActivityForResult(intent,1234);

  En este ejemplo, el valor 1234 servirá para identificar la actividad que devuelve el resultado (es un switch asociado).

- Para la recuperación de los resultados en la actividad "llamadora" se redefine el manejador **onActivityResult**

Código de petición: el código que se utilizó para arrancar la subactividad. Ten en cuenta que se pueden realizar varias llamadas a distintas "activities" y el resultado de todas esas llamadas se obtendrá a través de un único método *onActivityResult*.

protected void onActivityResult(int **requestCode**,
int **resultCode**, Intent **intent**) {

Código de resultado: el código que devuelve la subactividad. Si se aborta anormalmente la subactividad devuelve Activity.RESULT_CANCELED

Intent con los resultados devueltos por la subactividad.

Otros comentarios del código de la 1ª actividad


```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText editText=findViewById(R.id.editText);
        Button button=findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent=new
                Intent(getApplicationContext(),SegundaActivity.class);
                intent.putExtra("nombre", editText.getText().toString());
                startActivityForResult(intent,1234);
            }
        });
    }

    protected void onActivityResult(int requestCode,int resultCode, Intent
            intent) {
        String res="Sin resolver";
        if(requestCode==1234 && resultCode==RESULT_OK){
            res=intent.getExtras().getString("resultado");
        }
        TextView textView=findViewById(R.id.textView2);
        textView.setText(res);
    }

}
```

- Como el tratamiento del "listener" del botón, no hemos querido hacerlo en un método aparte de la MainActivity (como en otros ejercicios) para que no necesite implementarlo la propia actividad, necesitamos:
  - Declarar la instancia del objeto que necesitamos en el método on Click de tipo final
  - Dado que en el método onClick, "this" como contexto de la aplicación es desconocido, es necesario usar el método **getApplicationContext()**
- Si te molesta que se siga viendo el botón, dentro del "if" puede añadirse en el método onActivityResult:

  button.setVisibility(View.INVISIBLE);

  ya que el método setVisibility() de la clase View, permite fijar la visibilidad.

2ª actividad

- Observa como la 2º actividad ha recogido y visualizado los datos enviados por la primera:

```
String nombre= getIntent().getStringExtra("nombre");
...
textView.setText(String.format("Hola %s!. %s", nombre,
        getString(R.string.texto4)));
```

- Para que la segunda actividad devuelva un resultado se llama al método **setResult()** antes de la llamada OBLIGATORIA al método finish() de la clase Activity.
- El método setResult() acepta dos parámetros: un código de resultado (RESULT_OK, RESULT_CANCELED o cualquier entero) y un Intent con los datos que se desea devolver.

```
setResult(RESULT_OK, intent);
```

```
public class SegundaActivity extends AppCompatActivity
        implements View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_segunda);
        String nombre= getIntent().getStringExtra("nombre");
        TextView textView=findViewById(R.id.textView);
        textView.setText(String.format("Hola %s!. %s", nombre,
            getString(R.string.texto4)));
        Button button2=findViewById(R.id.button2);
        button2.setOnClickListener(this);
        Button button3=findViewById(R.id.button3);
        button3.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        String resultado = "RECHAZADAS CONDICIONES";
        if (v.getId()== R.id.button2)
            resultado = "ACEPTADAS CONDICIONES";
        Intent intent = new Intent();
        intent.putExtra("resultado", resultado);
        setResult(RESULT_OK, intent);
        finish();
    }
}
```

Clase Toast

- Un toast es un **mensaje que se muestra en pantalla durante unos segundos al usuario para luego volver a desaparecer automáticamente sin requerir ningún tipo de actuación** por su parte, y sin recibir el foco en ningún momento (o dicho de otra forma, sin interferir en las acciones que esté realizando el usuario en ese momento).
- Aparecen por defecto en la **parte inferior de la pantalla, sobre un rectángulo gris ligeramente translúcido** (aunque más adelante veremos que son personalizables).
- Por sus propias características, este tipo de notificaciones son ideales para mostrar mensajes rápidos y sencillos al usuario y como desarrolladores las utilizaremos para efectuar pruebas de traza de nuestra aplicación (es decir, las quitaremos cuando comprobemos que funciona!)
- La clase Toast dispone de un método **makeText()** al que deberemos pasar como parámetro el contexto de la actividad, el texto a mostrar y la duración del mensaje (que puede tomar los valores LENGTH_LONG o LENGTH_SHORT, dependiendo del tiempo que queramos que la notificación aparezca en pantalla). Tras obtener una referencia al objeto Toast a través de este método, ya sólo nos quedaría mostrarlo en pantalla mediante el método **show()**:

```
Toast.makeText(this,"Programador: Yo",Toast.LENGTH_SHORT).show();
```

Prácticas propuestas

- Realiza la hoja de ejercicios Ejercicios_04_Comunicando_actividades