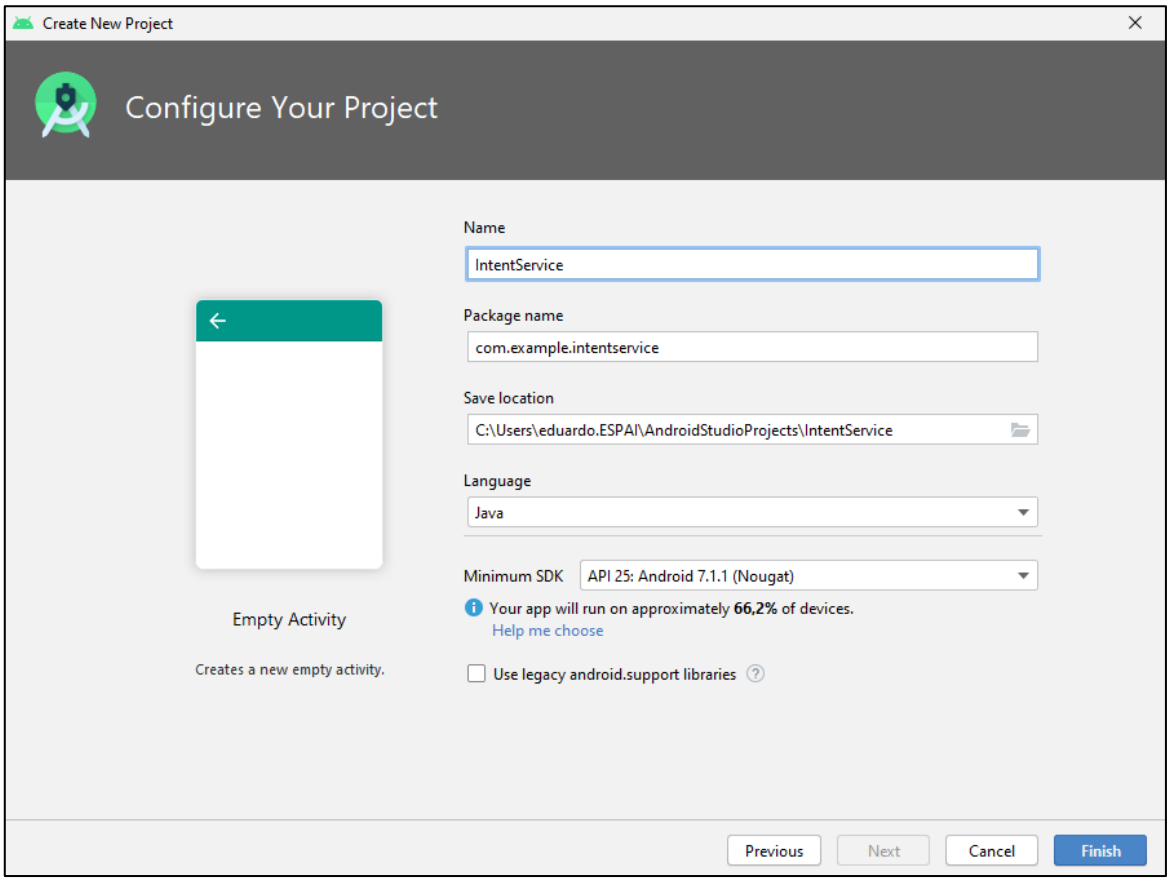
PRACTICA 16: INTENTSERVICE

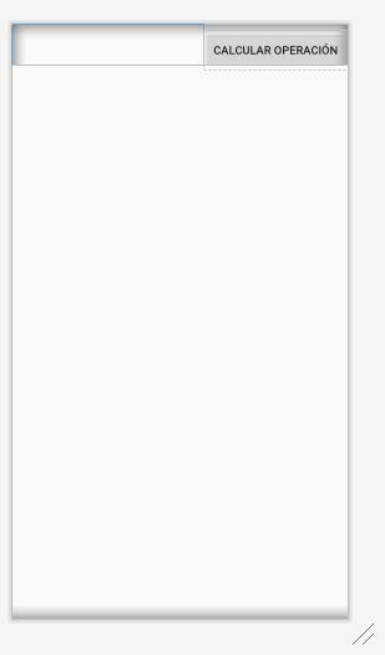
Parte I: *Un servicio que bloquea el hilo principal*

Muchos servicios han de realizar costosas operaciones o han de esperar a que concluyan lentas operaciones en la red. En ambos casos hay que tener la precaución de no bloquear el hilo principal. De hacerlo el resultado puede ser catastrófico, como se muestra en este ejercicio.

Paso 1. Crea un nuevo proyecto, de nombre IntentService, con los siguientes datos:



Paso 2. Reemplaza el código del *layout* principal por:



activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <EditText
            android:id="@+id/entrada"
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="numberDecimal"
            android:text="2.2" >
            <requestFocus/>
        </EditText>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="calcularOperacion"
            android:text="Calcular operación"/>
    </LinearLayout>
    <TextView
        android:id="@+id/salida"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text=" "
        android:textAppearance="?android:attr/textAppearanceMedium"/>
</LinearLayout>
```

Paso 3. Reemplaza el código de MainActivity por el siguiente:

MainActivity.java

```java
package com.example.intentservice;

import ...

public class MainActivity extends AppCompatActivity {

    private EditText entrada;
    public static TextView salida;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        entrada= (EditText) findViewById(R.id.entrada);
        salida= (TextView) findViewById(R.id.salida);
    }

    public void calcularOperacion(View view) {
        double n = Double.parseDouble(entrada.getText().toString());
        salida.append(n +"^2 = ");
        Intent i = new Intent( packageContext: this, ServicioOperacion.class);
        i.putExtra( name: "numero", n);
        startService(i);
    }
}
```

Observa como la variable salida ha sido declarada como **public static**. Esto nos permitirá acceder a esta variable desde otras clases. El método calcularOperacion() será llamado cuando se pulse el botón. Comienza obteniendo el valor real introducido en entrada. Se muestra la operación a realizar por salida. Luego, se crea una nueva intención con nuestro contexto y la clase que acabamos de crear. A continuación y se le añade un extra con el valor introducido. Finalmente, se arranca el servicio.

Paso 4. Crea la clase ServicioOperacion con el siguiente código:

```
public class ServicioOperacion extends Service {
    @Override
    public int onStartCommand(Intent i, int flags, int idArranque) {
        double n = i.getExtras().getDouble( key: "numero");
        SystemClock.sleep( ms: 5000);
        MainActivity.salida.append(n * n + "\n");
        return START_STICKY;
    }

    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }
}
```

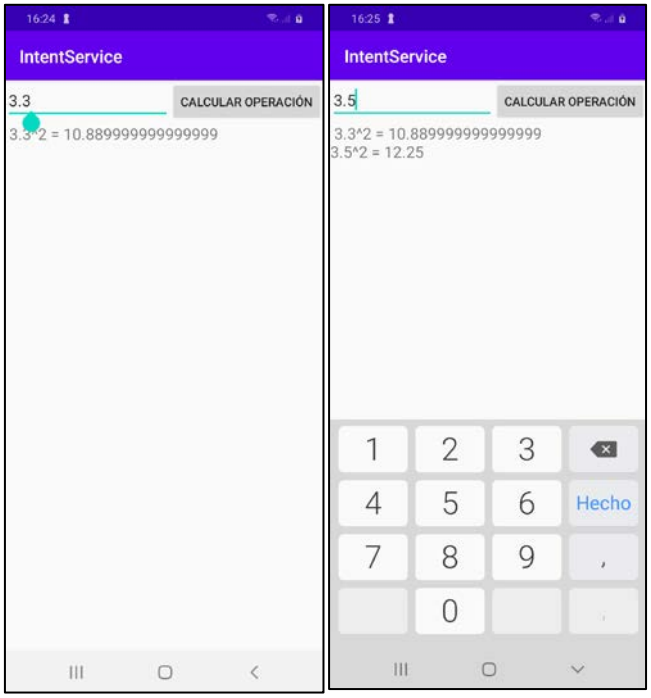
Cuando se arranca el servicio se llamará al método onStartCommand. Este comienza obteniendo el valor a calcular a partir de un extra. Luego vamos a simular que se realizan un gran número de operaciones para el vamos a bloquear el hilo durante 5000 ms (5 segundos) utilizando el método sleep. Una vez terminado el resultado se muestra directamente en el TextView salida. Esta forma de trabajar no resulta muy recomendable. Se ha realizado así, para ilustrar cómo es posible acceder al sistema gráfico de Android dado que estamos en el hilo principal. Finalmente, devolvemos START_NOT_STICKYpara indicar al sistema que si por fuerza mayor ha de destruir el servicio, no hace falta que lo vuelva a crear.

Paso 5. Edita el fichero AndroidManifest.xml y añade la siguiente línea dentro de la etiqueta <application>, para registrar el servicio:

AndroidManifest.xml

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

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

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

Paso 6. Ejecuta la aplicación. El resultado ha de ser similar al siguiente:

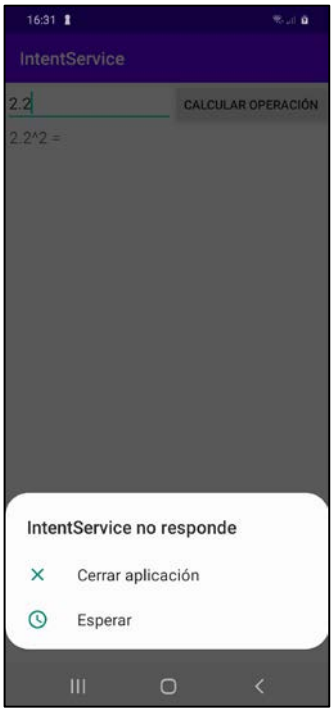


Observa cómo mientras se realiza la operación el usuario no puede pulsar el botón ni modificar el EditText. El usuario tendrá la sensación de que la aplicación está bloqueada.

Paso 7. Modifica el tiempo de retardo para que este sean 25 seg. (sleep(25000)).Ejecuta de nuevo la aplicación y observa como el sistema nos mostrará en siguiente error:

Se debe incrementar un poco más el tiempo de sleep (50 o 100) para que salga ese error

```
public class ServicioOperacion extends Service {
    @Override
    public int onStartCommand(Intent i, int flags, int idArranque) {
        double n = i.getExtras().getDouble( key: "numero");
        SystemClock.sleep( ms: 100000);
        MainActivity.salida.append(n * n + "\n");
        return START_STICKY;
    }

    @Override
    public IBinder onBind(Intent arg0) { return null; }
}
```



Para que esto no bloquee el hilo principal podemos utilizar un IntentService. En el siguiente ejercicio mostraremos como realizarlo.

Parte II: *Un servicio en su propio hilo*

En este ejercicio aprenderemos a crear servicios que se ejecutan en un hilo de ejecución diferente al principal utilizando la clase IntentService. Además veremos algunas limitaciones de este tipo de servicios, como la imposibilidad de acceder al sistema gráfico.

Paso 1. Abre el proyecto *IntentService* creado en el ejercicio anterior.

Paso 2. Crea la clase IntentServiceOperacion con el siguiente código:

```
public class IntentServiceOperacion extends IntentService {
    public IntentServiceOperacion() {
        super( name: "IntentServiceOperacion");
    }

    @Override
    protected void onHandleIntent(@Nullable Intent intent) {
        double n = intent.getExtras().getDouble( key: "numero");
        SystemClock.sleep( ms: 5000);
        MainActivity.salida.append(n*n + "\n");
    }
}
```

Paso 3. Abre MainActivityy modifica la línea:

Intent i = **new** Intent(**this**, ServicioOperacion.**class**);

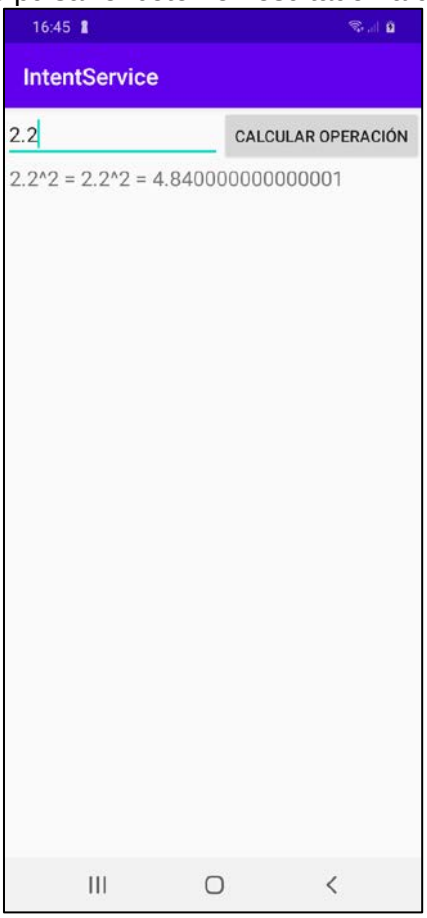
por:

Intent i = **new** Intent(**this**, IntentServiceOperacion.**class**);

```
public void calcularOperacion(View view) {
    double n = Double.parseDouble(entrada.getText().toString());
    salida.append(n +"^2 = ");
    Intent i = new Intent( packageContext: this, IntentServiceOperacion.class);
    i.putExtra( name: "numero", n);
    startService(i);
}
```

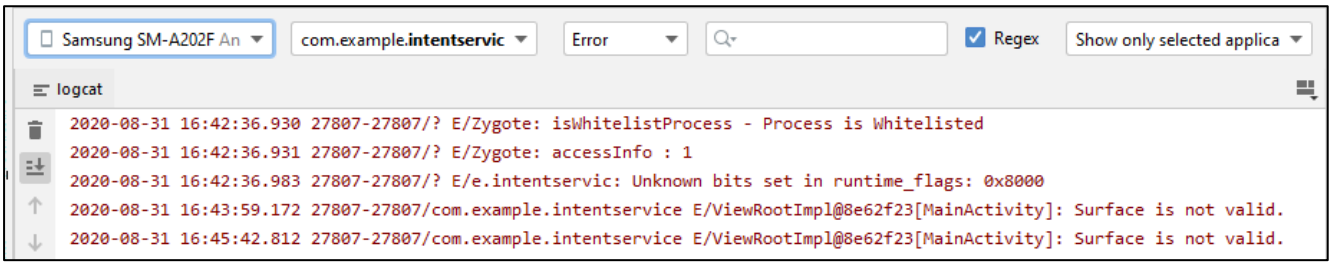
```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="IntentService"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>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

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

Paso 4. Ejecuta la aplicación. Tras pulsar el botón el resultado ha de ser:



En esta versión de Android no se produce ningun error.

Paso 5. Abre la vista *LogCat* y busca el siguiente Error:



No existe ningun error.

Te indica que solo desde el hilo principal se va a poder interactuar con las vistas del interfaz de usuario. También está prohibido desde otros hilos usar la clase Toast.

Como un hilo que hemos creado pertenece al mismo proceso que el hilo principal, compartimos con este todas las variables. Para devolver el valor calculado, podríamos implementar un método o variable público, tanto en la clase del servicio, como de la actividad. No obstante, vamos a resolver este problema utilizando un mecanismo más elegante, los receptores de anuncios. Se explica en el siguiente apartado.

Parte III: Creación de un nuevo anuncio broadcast

En el ejercicio anterior hemos creado un servicio desde una actividad para realizar una operación matemática. Una vez que el servicio ha concluido su trabajo queríamos que avisara a la actividad y le devolviera el valor resultante. En este ejercicio realizaremos este trabajo por medio de un anuncio broadcast.

Paso 1. Abre el proyecto *IntentService* creado en el apartado anterior.

Paso 2. Añade el siguiente código dentro de la clase MainActivity:

```
public class MainActivity extends AppCompatActivity {
    private EditText entrada;
    public static TextView salida;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        entrada= (EditText) findViewById(R.id.entrada);
        salida= (TextView) findViewById(R.id.salida);
    }

    public void calcularOperacion(View view) {
        double n = Double.parseDouble(entrada.getText().toString());
        salida.append(n +"^2 = ");
        Intent i = new Intent( packageContext: this, IntentServiceOperacion.class);
        i.putExtra( name: "numero", n);
        startService(i);
    }
    public class ReceptorOperacion extends BroadcastReceiver {
        public static final String ACTION_RESP=
                "com.example.intentservice.intent.action.RESPUESTA_OPERACION";
        @Override
        public void onReceive(Context context, Intent intent) {
            Double res = intent.getDoubleExtra( name: "resultado",  defaultValue: 0.0);
            salida.append(" "+ res);
        }
    }
}
```

Esta nueva clase solo va a utilizarse en esta actividad por lo que puede ser definida dentro de la clase MainActivity, en lugar de en un fichero independiente. Se trata de un receptor broadcast, que cada vez que llegue un nuevo anuncio, leerá un valor enviado en el extra "resultado", y lo añadirá al TextView salida.

Paso 3. Añade las siguientes líneas al método onCreate()

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    entrada= (EditText) findViewById(R.id.entrada);
    salida= (TextView) findViewById(R.id.salida);

    IntentFilter filtro = new IntentFilter(ReceptorOperacion.ACTION_RESP);
    filtro.addCategory(Intent.CATEGORY_DEFAULT);
    registerReceiver(new ReceptorOperacion(), filtro);
}
```

Con este código hemos asociado un anuncio broadcast a nuestro receptor de anuncios. Como vimos en otro apartado esta tarea también puede realizarse por medio de *AndroidManifest.xml*. El programador puede escoger uno u otro modo según le convenga. Al tratarse de un anuncio para una comunicación interna a nuestra aplicación, parece más conveniente realizarlo así que publicarlo por *AndroidManifest*.

Paso 4. Nos queda lanzar el anuncio broadcast. Para ello añade las siguientes líneas de IntentServiceOperacion.onHandleIntent ():

MainActivity.*salida*.append(n*n + "\n");

por:

Intent i = **new** Intent();

i.setAction(MainActivity.ReceptorOperacion.*ACTION_RESP*);

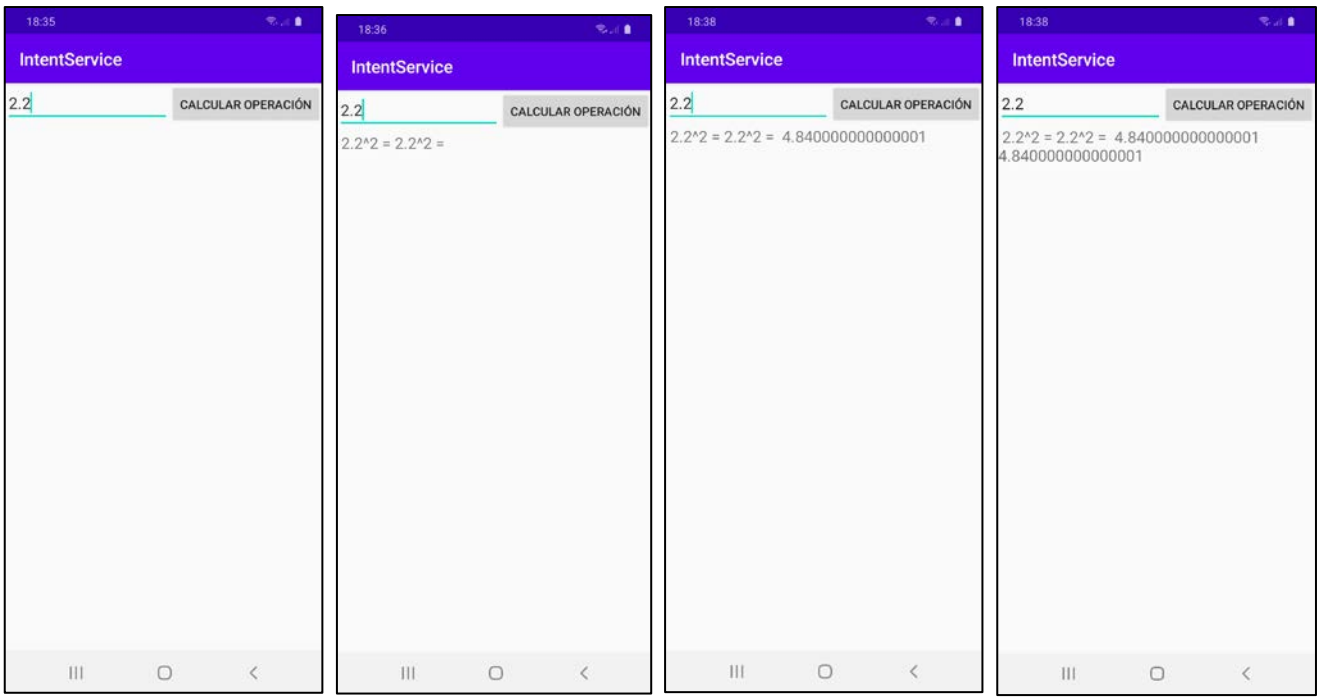
i.addCategory(Intent.*CATEGORY_DEFAULT*);

i.putExtra("resultado", n*n);

sendBroadcast(i);

```
public class IntentServiceOperacion extends IntentService {
    public IntentServiceOperacion() { super( name: "IntentServiceOperacion"); }

    @Override
    protected void onHandleIntent(@Nullable Intent intent) {
        double n = intent.getExtras().getDouble( key: "numero");
        SystemClock.sleep( ms: 5000);
        //MainActivity.salida.append(n*n + "\n");

        Intent i = new Intent();
        i.setAction(MainActivity.ReceptorOperacion.ACTION_RESP);
        i.addCategory(Intent.CATEGORY_DEFAULT);
        i.putExtra( name: "resultado",  value: n*n);
        sendBroadcast(i);
    }
}
```

Paso 5. Verifica que la aplicación funciona perfectamente. Pulsa repetidas veces el botón y verifica cómo esta no se bloquea mientras se calculan las operaciones. Advierte cómo, aunque se pulse tres veces seguidas, no comienzan las tres operaciones a la vez. Estas serán realizadas de una en una, de manera que irán apareciendo los resultados a intervalos de 5 segundos.



Paso 6. Modifica el tiempo de retardo para que este sean 25 seg. (sleep(25000)). Ejecuta de nuevo la aplicación y observa cómo el sistema no nos muestra ningún error.

```
@Override
protected void onHandleIntent(@Nullable Intent intent) {
    double n = intent.getExtras().getDouble( key: "numero");
    SystemClock.sleep( ms: 25000);
    //MainActivity.salida.append(n*n + "\n");

    Intent i = new Intent();
    i.setAction(MainActivity.ReceptorOperacion.ACTION_RESP);
    i.addCategory(Intent.CATEGORY_DEFAULT);
    i.putExtra( name: "resultado",  value: n*n);
    sendBroadcast(i);
}
```

Todo funciona bien igual que antes