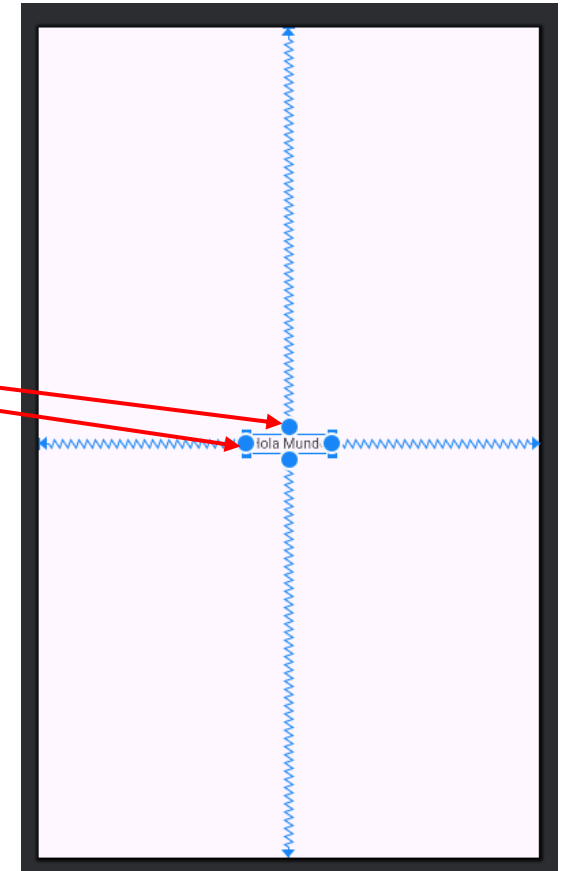


Software para Dispositivos Móviles

Miguel Sánchez Santillán
sanchezsmiguel@uniovi.es

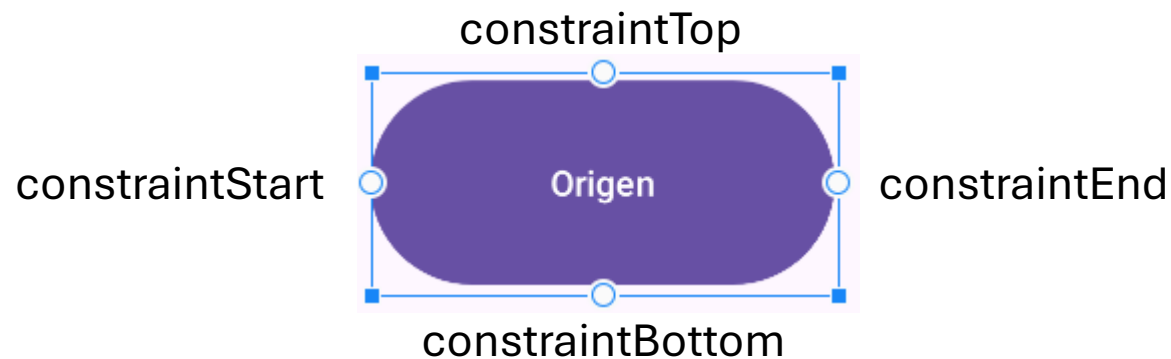
ConstraintLayout – Layout Flexible

- ~~Layout anidados~~ → **diseño adaptable** (responsive)
- **Restricción** vertical y horizontal
 - Las restricciones **tiran** del componente.
- Uso optimizado en el **editor visual**

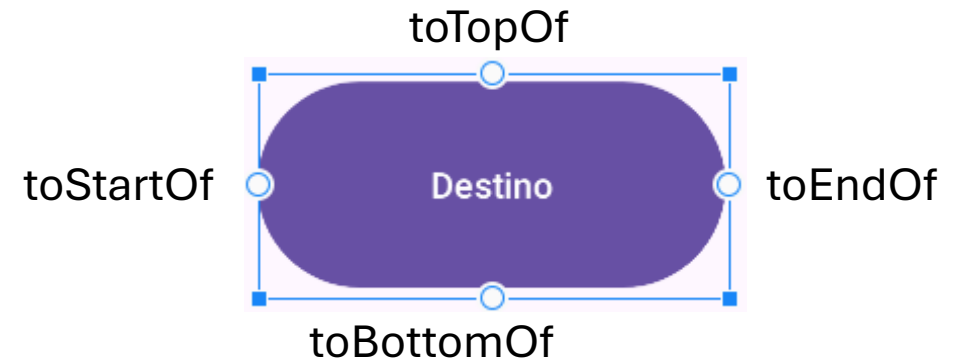


ConstraintLayout – Restricciones I

- Cuatro **puntos de origen**
 - **Mínimo** 1 horizontal y 1 vertical



- Origen **enlazado** a un **destino**
 - Mediante la *id* o *parent*



- ¿A qué elemento se refiere este código? ¿Qué refleja?

```
app:layout_constraintEnd_toStartOf="@+id/bDestino"  
app:layout_constraintTop_toTopOf="parent" />
```

ConstraintLayout – Restricciones II

- El editor visual facilita la configuración mediante un panel

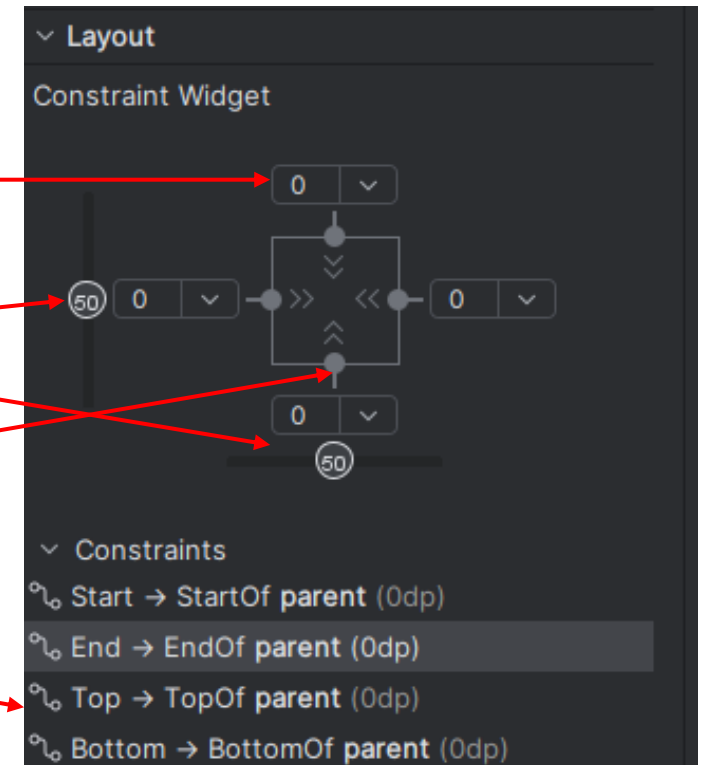
- Otras propiedades/valores interesantes son:

- Uso de **márgenes**

- **constraintHorizontalBias|VerticalBias**

- Favorecer una dirección [0.0 , 1.0]

- **Eliminar restricciones**



Ejercicio I – Interfaz básica

- Empleando **ConstraintLayout**, diseña la interfaz de la imagen
- Empieza con el **editor visual**
- Y utiliza la vista de **código** para **refinar**
- Esto es un **checkBox**

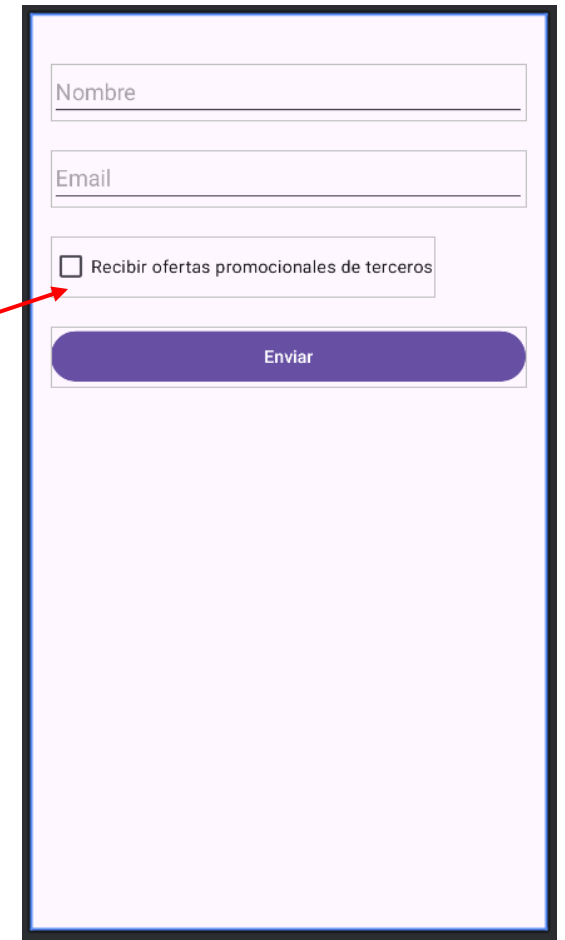


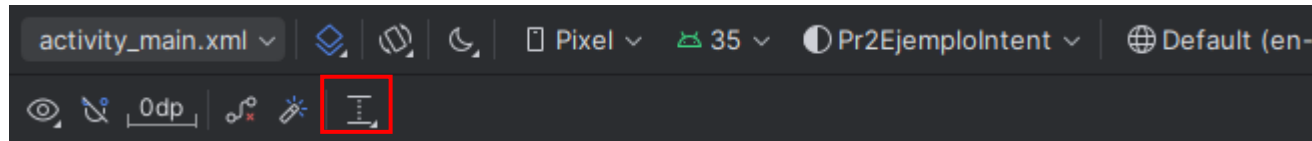
Diagrama de una interfaz de usuario básica, probablemente para un formulario de registro o suscripción. La interfaz está contenida en un recuadro vertical con un fondo claro y una sombra. Los elementos incluyen:

- Un campo de texto con el placeholder "Nombre".
- Un campo de texto con el placeholder "Email".
- Un checkbox con el texto "Recibir ofertas promocionales de terceros".
- Un botón de envío con el texto "Enviar".

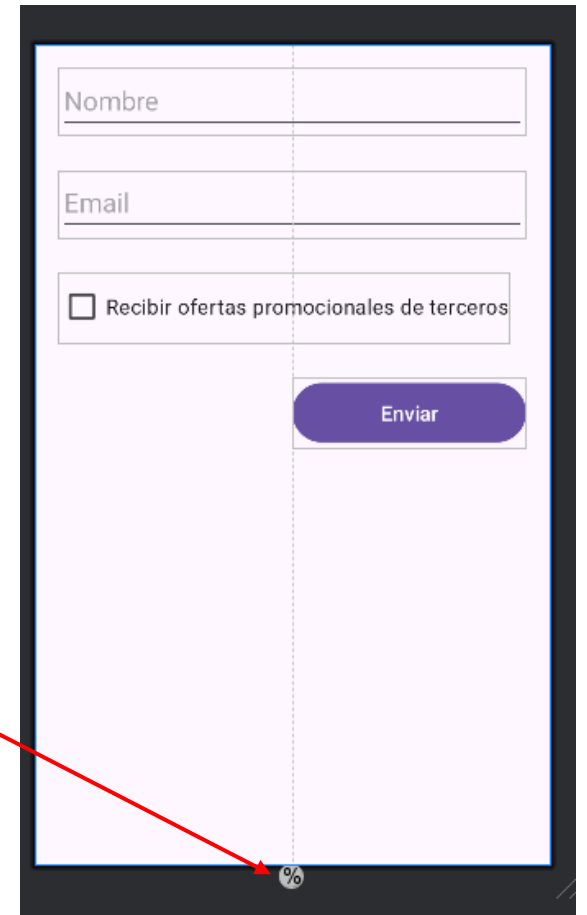
Una flecha roja apunta desde el texto "checkBox" en la lista de puntos hacia el checkbox de la interfaz.

Ejercicio II – Guías como apoyo

- Es posible añadir **guías** que te **ayuden a maquetar**



- Basadas en valores absolutos o **porcentajes**
 - Haciendo clic en el símbolo
- **Posiciona el botón como en la imagen**



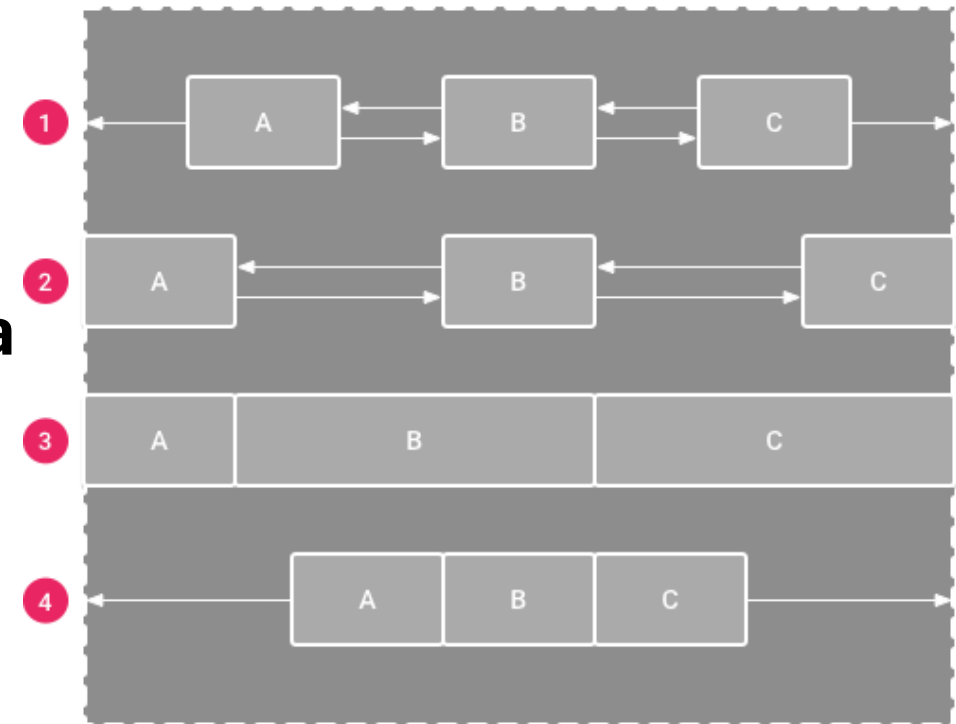
Ejercicio III – Añadiendo lógica

- Añade un **TextView** debajo del botón
- Al hacer **clic** en el botón, se **cambiará** el **texto** del TextView, incluyendo:
 - El **Nombre** tecleado en el EditTextNombre
 - El **Email** tecleado en el EditTextEmail
 - Si está **activo** o **desactivado** el checkBox

The diagram shows a mobile app interface with a light purple background. It contains three input fields: 'Nombre', 'Email', and a checkbox labeled 'Recibir ofertas promocionales de terceros'. Below the checkbox is a purple button labeled 'Enviar'. A red arrow originates from the text 'Añade un TextView debajo del botón' in the list and points to a new, empty rectangular area below the 'Enviar' button, indicating where to add the new TextView.

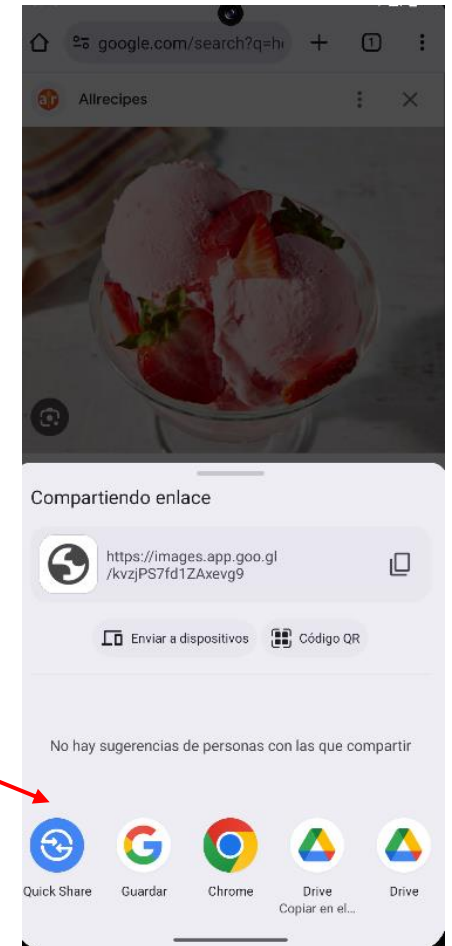
ConstraintLayout – Cadenas

- Es posible encadenar vertical o horizontalmente una serie de elementos
- La cadena se indica **partiendo del elemento situado más arriba y más a la izquierda**
 - <https://developer.android.com/develop/ui/views/layout/constraint-layout#constrain-chain>



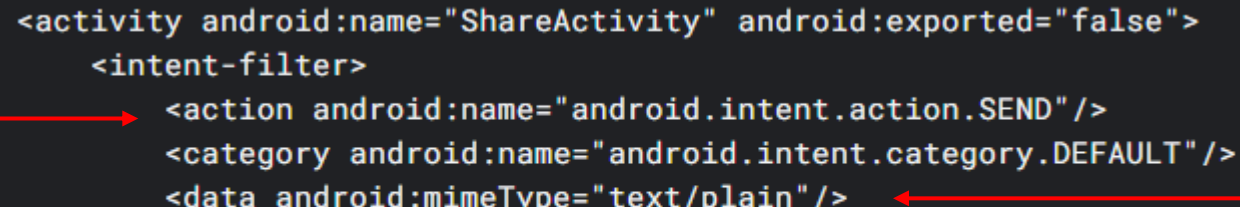
Intents – Objetos para *lanzar* componentes

- Servicio, emisión o **una activity de la app** (o de otra)
- **Implícitos:** Indicamos **la acción** a realizar y no la app
 - Ejemplo: Compartir un enlace
- **Explícitos:** Activity **específica**
 - **Conocemos** el componente



Intents – Lanzamientos implícitos

- Cuando creamos una Activity en el **manifiesto** se indicamos a qué **acciones** puede responder y el **tipo** de información

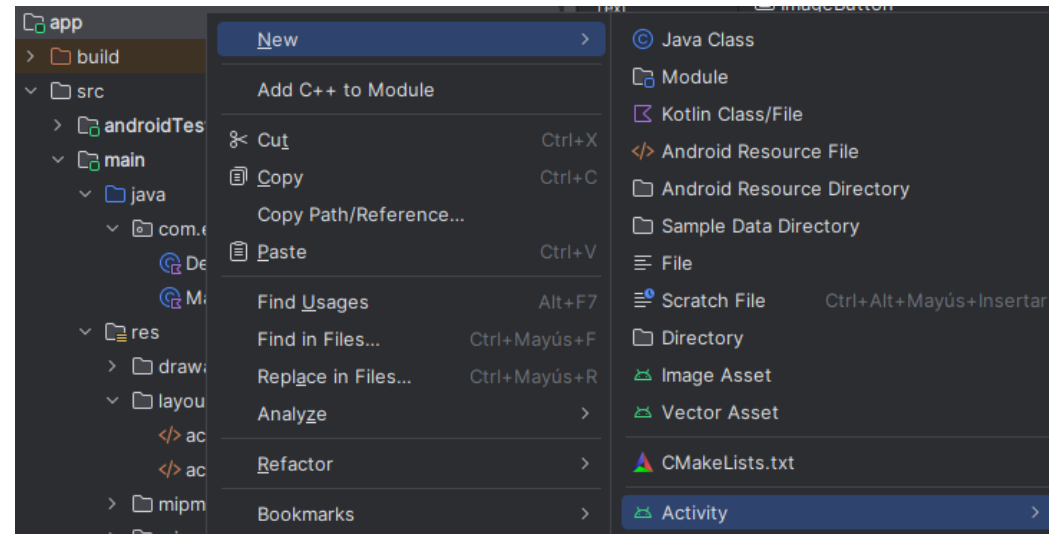


```
<activity android:name="ShareActivity" android:exported="false">
  <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
  </intent-filter>
</activity>
```

- <https://developer.android.com/guide/components/intents-filters#Receiving>

Intents – Añadir una nueva Activity I

- Al crear una Activity, es necesario declararla en el manifiesto.
- Android Studio lo hace automáticamente:
 - **Clic derecho en app** → New → Activity → **Empty Views Activity**



Intents – Añadir una nueva Activity y II

- Nombre: *DetallesActivity* . Activa opción de **generar** fichero **layout**
- Al aceptar **se añade al manifiesto:**
 - Cuidado con el copy & paste de clases ;)
- **Añade a la interfaz de esta actividad un texto/componente cualquiera.**

```
<activity  
    android:name=".DetallesActivity"  
    android:exported="false" />
```

Intents – Lanzamientos explícitos I

- Se realizan mediante la clase **Intent()**

```
val intent = Intent(applicationContext, DetallesActivity::class.java)  
startActivity(intent)
```

- Recibe el **contexto** (podrá variar cuando veamos Fragments) **y la clase** de la activity a lanzar.
- **Haz que se lance la nueva Activity al pulsar el botón Enviar**

Intents – Lanzamientos explícitos II

- Podemos enviar valores a la nueva activity.
- **Antes** de llamar a *startActivity*, es posible utilizar (n veces):
intent.putExtra(clave, valor)
- En el destino se recupera mediante:
intent.get**String**Extra(clave)
- String puede cambiarse por: Double, Integer, Boolean,

Ejercicio IV – Combinando Activities

- Al pulsar el botón *Enviar*, DetallesActivity recibirá el email, el nombre y el estado del checkbox
- Diseñar el layout de **DetallesActivity** para que tenga **tres TextView**, uno para cada valor
 - Y un **botón** que solamente **invocará al método finish()**
- Añadir validación para que solamente se lance la Activity si la longitud de los campos es mayor que 0

Kotlin – Companion object

- En Kotlin **no tenemos el *static***, algo similar sería:

```
class DetallesActivity : AppCompatActivity() {  
    companion object {  
        const val CLAVE_NOMBRE : String = "INFO_NOMBRE"  
        const val CLAVE_EMAIL : String = "INFO_EMAIL"  
    }  
}
```

- **Utiliza este enfoque en el ejercicio anterior.**
 - Es decir, no me uses DIRECTAMENTE el valor en los put / get.

Intents – Explícito sí, pero con resultado

- En ocasiones queremos **recibir un resultado** de la actividad lanzada
- Es necesario utilizar la **Activity Result API**
 - <https://developer.android.com/training/basics/intents/result> (mucho texto)
 - <https://developer.android.com/reference/androidx/activity/result/contract/ActivityResultContracts>
- Básicamente se basa en utilizar un **launcher** con el **contrato** adecuado: `StartActivityForResult`.

Intents – El resultado es un Intent

- Para devolver el resultado, se crea un **Intent**.
- Se añaden los pares clave-valor al **intent** con el **.putExtra**
- Se invoca al método setResult(**código_fin**, **intent**) y luego finish()
- **código_fin** → varios posibles: **RESULT_OK**, **RESULT_CANCELED**...

Intents – Launcher explícito inicialización

- En MainActivity se **declara** el launcher

```
private lateinit var launcher : ActivityResultLauncher<Intent>
```

- Y se **inicializa** el launcher (NO ESTAMOS LANZANDO NADA):

```
launcher = registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { resultado ->  
    procesarResultado(resultado)  
}
```

- Se ejecuta **cuando acaba** la actividad lanzada.

Intents – Launcher explícito. Resultado

- El parámetro (*resultado*) de la lambda es de tipo **ActivityResult**
- ¿Cómo finalizó?
resultado.resultCode (RESULT_OK | RESULT_CANCELED)
- Acceder a los resultados:
resultado.data?.getStringExtra(clave)

Intents – Launcher explícito. Lanzamiento

- Ya no se invoca a **startActivity()**
- Se utiliza el **launcher**:

```
launcher.launch(intent)
```

Ejercicio V – Activity con resultado

- Modifica el ejercicio anterior para que **DetallesActivity** tenga dos botones en lugar de uno:
 - **Aceptar** → Devuelve el RESULT_OK y un texto (el que quieras)
 - **Cancelar** → Devuelve el RESULT_CANCELED
- **MainActivity** empleará el **launcher** y gestionará el resultado.
 - Mostrará un Toast indicando si se canceló o se aceptó.
 - Si se aceptó, mostrará el texto en un TextView

Ejercicio – Dos Activities

- Diseña un ejercicio con **dos Activities**. Las condiciones son:
 - Debes utilizar el componente **spinner** y **otro nuevo a tu elección**
 - La primera activity debe tener validación de campos, para decidir si se lanza o no se lanza la segunda activity.
 - La segunda activity realizará algún tipo de funcionalidad en base a los valores recibidos de la primera.
 - La primera activity gestionará el resultado de la segunda activity