Activitys e intents

Introducción

Una actividad representa una sola pantalla de la aplicación con la que el usuario puede realizar una única tarea específica, como tomar una foto, enviar un correo electrónico o ver un mapa. Una actividad generalmente se presenta al usuario como una ventana de pantalla completa.

Una aplicación generalmente consta de varias pantallas que están débilmente unidas entre sí. Cada pantalla es una actividad. Normalmente, una actividad de una aplicación se especifica como la actividad "principal" (), que se presenta al usuario cuando se inicia la aplicación. La actividad principal puede iniciar otras actividades para realizar diferentes acciones.MainActivity.java

Cada vez que se inicia una nueva actividad, la actividad anterior se detiene, pero el sistema conserva la actividad en una pila (la "pila posterior"). Cuando comienza una nueva actividad, esa nueva actividad se envía a la pila posterior y se centra en el usuario. La pila trasera sigue la lógica básica de pila de "último en entrar, primero en salir". Cuando el usuario termina con la actividad actual y presiona el botón Atrás, esa actividad se saca de la pila y se destruye, y la actividad anterior se reanuda.

Una actividad se inicia o activa con una *intent*. Una intent es un mensaje asincrónico que puede usar en su actividad para solicitar una acción de otra actividad o de algún otro componente de la aplicación. Utiliza una intent para iniciar una actividad a partir de otra actividad y para pasar datos entre actividades.

Un intent puede ser *explícito* o *implícito*

Una *intent explícita* es aquella en la que conoces el objetivo de esa intent. Es decir, ya conoce el nombre de clase completo de esa actividad específica.

Una *intent implícita* es aquella en la que no tiene el nombre del componente de destino, pero tiene una acción general que realizar.

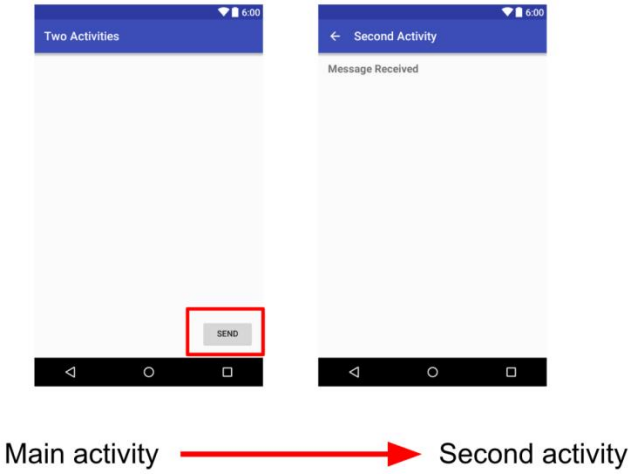
Descripción general de la aplicación

Se creará una aplicación llamada **TwoActivities** que contendrá dos implementaciones.

La aplicación se crea en tres etapas.

1° Parte: Activity

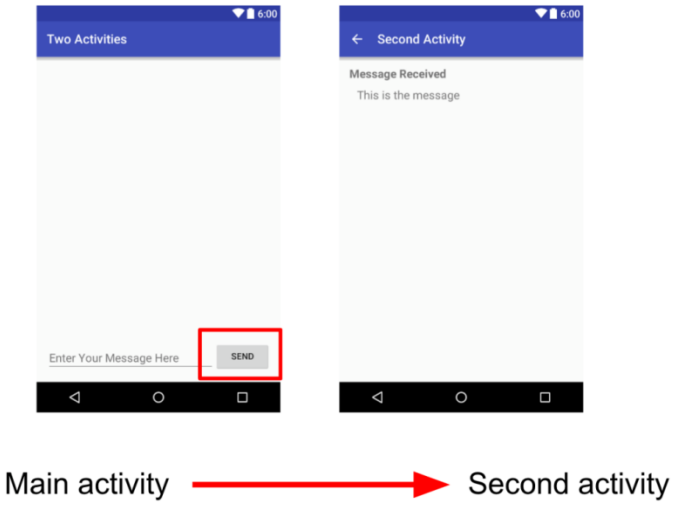
En la primera etapa, crea una aplicación cuya actividad principal contiene un botón, **Enviar**. Cuando el usuario hace clic en este botón, la actividad principal utiliza una **intent** para iniciar la segunda actividad.



2° Parte:

En la segunda etapa, agrega una vista a la actividad principal. El usuario escribe un mensaje y hace clic en **Enviar**.

- La actividad principal utiliza una **intent** para iniciar la segunda actividad y enviar el mensaje del usuario a la segunda actividad.
- La segunda actividad muestra el mensaje que recibió.EditText

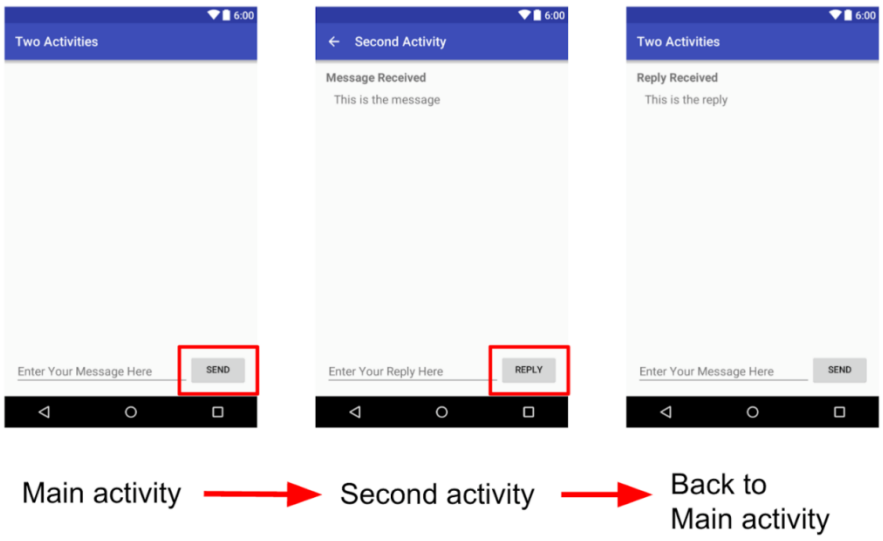


3° Parte:

En la etapa final de creación de la aplicación **TwoActivities**, agrega un botón **Responder** a la segunda actividad.

- El usuario ahora puede escribir un mensaje de respuesta y tocar **Responder**, y la respuesta se muestra en la actividad principal.

- En este punto, utiliza una intent para pasar la respuesta de la segunda actividad a la actividad principal.EditText



Desarrollo

Tarea 1: Crear el proyecto TwoActivities

Configure el proyecto inicial con una actividad principal, defina el diseño y defina un método básico para el evento del botón onClick.

Pasos:

1. Inicie Android Studio y cree un nuevo proyecto de Android Studio.

- Asigne el nombre TwoActivities a su aplicación
- La carpeta del proyecto se llama automáticamente TwoActivities y el nombre de la aplicación que aparece en la barra de aplicaciones será "Two Activities".

2. Elija Actividad vacía para la plantilla de actividad. Haga clic en Siguiente.

3. Acepte el nombre de actividad predeterminado (MainActivity). Asegúrese de que las opciones Generar archivo de diseño y Compatibilidad con versiones anteriores (AppCompat) estén marcadas.

4. Haga clic en Finalizar.

Definir el diseño de la actividad principal

1. Abra **res > diseño > actividad_principal.xml** en el panel **Proyecto > Android**. Aparece el editor de diseño.

2. Haga clic en la pestaña Diseño y elimine el **TextView** (que dice "Hola mundo") en el panel Árbol de componentes.

3. Con Conexión automática activada (la configuración predeterminada),

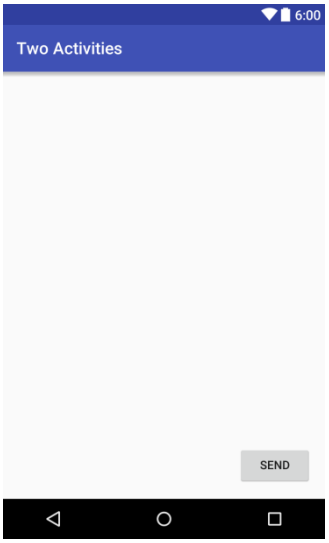
- arrastre un Botón desde el panel Paleta hasta la esquina inferior derecha del diseño. La conexión automática crea restricciones para el botón**.**

4. En el panel Atributos, establezca:

- ID en **button_main**
- **layout_width** y **layout_height** en **wrap_content**

- ingrese **Enviar** para el campo Texto.

El diseño ahora debería verse así:



5. Haga clic en la pestaña Texto para editar el código XML. Agregue el siguiente atributo al Botón:

android:onClick="launchSecondActivity"

El valor del atributo está subrayado en rojo porque aún no se ha creado el método **launchSecondActivity()**.

- Ignore este error por ahora; lo arreglas en la siguiente tarea.

6. Extraiga el recurso de cadena, para "**Enviar**" y use el nombre **button_main** para el recurso.

//

Extraer recursos de cadena

En lugar de codificar cadenas de forma rígida, se recomienda utilizar recursos de cadena, que representan las cadenas.

- Tener las cadenas en un archivo separado facilita su administración, especialmente si usa estas cadenas más de una vez.
- Además, los recursos de cadena son obligatorios para traducir y localizar su aplicación, ya que necesita crear un archivo de recursos de cadena para cada idioma.

-Haga clic una vez en la palabra " **Enviar** ", haga clic en la bombilla que aparece al principio y seleccione **Extraer recurso de cadena** en el menú emergente.

-Ingrese **button_label_main** para el nombre del recurso.
-Haga clic en Aceptar.
Se crea un recurso de cadena en el archivo res/values/strings.xml y la cadena en su código se reemplaza con una referencia al recurso:
@string/button_label_main

En el panel **Proyecto > Android**, expanda los valores dentro de **res** y luego haga doble clic en **strings.xml** para ver sus recursos de cadena en el archivo **strings.xml**
//

El código XML para el botón debe tener el siguiente aspecto:

```
<Button
    android:id="@+id/button_main"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/button_main"
    android:onClick="launchSecondActivity"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent" />
```

Definir la acción del botón

Implementar el método **launchSecondActivity**() al que se refirió en el diseño del atributo **android:onClick.**

1. Haga clic en "**launchSecondActivity**" en el código XML **activity_main.xml**.

2. Presione **Alt+Enter** y seleccione Create '**launchSecondActivity**(View)' en '**MainActivity**'.

Se abre el archivo **MainActivity** y Android Studio genera un método básico para el controlador **launchSecondActivity**().

3. Dentro de **launchSecondActivity**(), agregue una declaración de registro que diga **"¡Botón presionado!"**

Log.d(LOG_TAG, "¡Botón presionado!");

LOG_TAG se mostrará en rojo. Porque no se ha declarado la variable todavía...

4. En la parte superior de la clase MainActivity, agregue una constante para la variable LOG_TAG:

```
private static final String LOG_TAG =
                    MainActivity.class.getSimpleName();
```

Esta constante usa el nombre de la clase como etiqueta.

5. Ejecute su aplicación.
Cuando haces clic en el **botón Enviar**, ves el mensaje **"¡Haz clic en el botón!"** mensaje en el panel de **Logcat**.
Si hay demasiados resultados en el monitor, escriba MainActivity en el cuadro de búsqueda y el panel de Logcat solo mostrará líneas que coincidan con esa etiqueta.

El código de MainActivity debería tener el siguiente aspecto:

```
package com.example.android.twoactivities;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;

public class MainActivity extends AppCompatActivity {
  private static final String LOG_TAG =
                      MainActivity.class.getSimpleName();

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

  public void launchSecondActivity(View view) {
     Log.d(LOG_TAG, "Button clicked!");
  }
}
```

Tarea 2: Crear y lanzar la segunda Actividad

Cada nueva actividad que agregue a su proyecto tiene su propio diseño y archivos Java, separados de los de la actividad principal. También tienen sus propios elementos <**actividad**> en el archivo **AndroidManifest.xml.**
Al igual que con la actividad principal, las implementaciones de actividades nuevas que creas en Android Studio también se extienden desde la clase **AppCompatActivity**.
Cada actividad en su aplicación solo está vagamente conectada con otras actividades. Sin embargo, puede definir una actividad como principal de otra actividad en el archivo **AndroidManifest.xml.**
Esta relación padre-hijo permite que Android agregue sugerencias de navegación, como flechas hacia la izquierda en la barra de título para cada actividad.
Una actividad se comunica con otras actividades (en la misma aplicación y en diferentes aplicaciones) con una Intent. Un Intent puede ser explícito o implícito:

- Una explicit intent es aquella en la que conoce el objetivo de esa intención; es decir, ya conoce el nombre de clase completo de esa actividad específica.
- Una implicit intent es aquella en la que no tiene el nombre del componente de destino, pero tiene una acción general que realizar.

En esta tarea, agregará una segunda actividad a la aplicación, con su propio diseño. Modificará el archivo **AndroidManifest.xml** para definir la actividad principal como principal de la segunda actividad. Luego, modificará el método **launchSecondActivity**() en **MainActivity** para incluir una intención que inicie la segunda actividad cuando haga clic en el botón.

Crear la segunda Actividad

1. Haga clic en la carpeta de la aplicación para su proyecto y seleccione **Archivo > Nuevo > Actividad >** Actividad vacía.

2. Asigne el nombre **SecondActivity** a la nueva actividad. Asegúrese de **que Generar archivo de diseño y Compatibilidad con versiones anteriores** (**AppCompat**) estén marcados. El nombre del diseño se completa como **activity_second**.

No marque la opción de iniciar Activity

3. Haga clic en Finalizar.

Android Studio agrega un nuevo diseño de actividad (**activity_second.xml**) y un nuevo archivo Java (**SecondActivity.java**) a su proyecto para la nueva actividad. También actualiza el archivo **AndroidManifest.xml** para incluir la nueva actividad.

Modificar el archivo AndroidManifest.xml

1. Abra **manifests > AndroidManifest.xml**.

2. Busque el elemento **<actividad>** que creó Android Studio para la segunda actividad.

```
<activity android:name=".SecondActivity"></activity>
```

Reemplace el elemento <activity> por lo siguiente:

```
<activity android:name=".SecondActivity"
   android:label = "Second Activity"
   android:parentActivityName=".MainActivity">
   <meta-data
      android:name="android.support.PARENT_ACTIVITY"
      android:value=
            "com.example.android.twoactivities.MainActivity" />
</activity>
```

El atributo **label** agrega el título de la Actividad a la barra de la aplicación.
Con el atributo **parentActivityName**, indica que la actividad principal es la principal de la segunda actividad. Esta relación se usa para la navegación hacia arriba en su aplicación: la barra de la aplicación para la segunda actividad tendrá una flecha hacia la izquierda para que el usuario pueda navegar "hacia arriba" a la actividad principal.
Con el elemento **<meta-data>,** proporciona información arbitraria adicional sobre la actividad en forma de pares clave-valor. En este caso, los atributos de metadatos hacen lo mismo que el atributo **android:parentActivityName:** definen una relación entre dos actividades para la navegación ascendente. Estos atributos de metadatos son necesarios para las versiones anteriores de Android, porque el atributo **android:parentActivityName** solo está disponible para los niveles de **API 16** y superiores.
4. Extraiga un recurso de cadena para " Second Activity " en el código anterior y use activity2_name como nombre del recurso.

Definir el diseño de la segunda Actividad

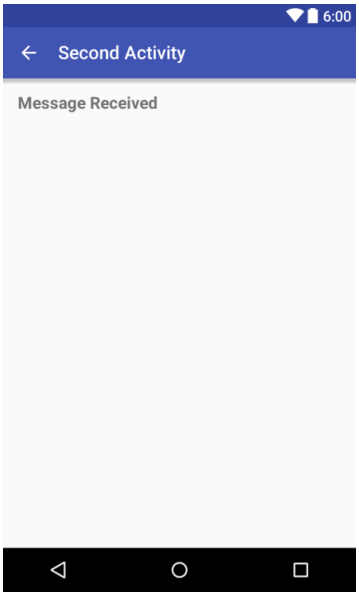
1. Abra activity_second.xml y haga clic en la pestaña Diseño si aún no está seleccionada.

2. Arrastre un **TextView** desde el panel Pallete hasta la esquina superior izquierda del diseño y agregue restricciones a los lados superior e izquierdo del diseño. Establezca sus atributos en el panel Atributos de la siguiente manera:

Attribute	**Value**
id	text_header
Top margin	16
Left margin	8
layout_width	wrap_content
layout_height	wrap_content
text	Message Received
textAppearance	AppCompat.Medium
textStyle	B (bold)

El valor de **textAppearance** es un atributo de tema especial de Android que define los estilos de fuente básicos

El diseño ahora debería verse así:



3. Haga clic en la pestaña Texto para editar el código XML y extraiga la cadena " Message Received " en un recurso llamado text_header.

4. Agregue el atributo **android:layout_marginLeft="8dp"** a **TextView** para complementar el atributo **layout_marginStart** para versiones anteriores de Android.

El código XML para actividad_segundo.xml debe ser el siguiente:

```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="com.example.android.twoactivities.SecondActivity">

   <TextView
      android:id="@+id/text_header"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginStart="8dp"
      android:layout_marginLeft="8dp"
      android:layout_marginTop="16dp"
      android:text="@string/text_header"
      android:textAppearance=
                  "@style/TextAppearance.AppCompat.Medium"
      android:textStyle="bold"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Añadir un Intent a la Actividad principal

Agregará una **Intent** explícita a la actividad principal. Esta **Intent** se usa para activar la segunda actividad cuando se hace clic en el botón Enviar.

1. Abra Actividad principal.

2. Cree una nueva **Intent** en el método **launchSecondActivity**().

El constructor de **Intent** toma dos argumentos para una **Intent** explícita: un contexto de aplicación y el componente específico que recibirá esa intención. Aquí debe usar **this** como Contexto y **SecondActivity.class** como la clase específica:

```
Intent intent = new Intent(this, SecondActivity.class);
```

3. Llamar al método **startActivity** () con la nueva **Intent** como argumento.

```
startActivity(intent);
```

4. Ejecute la aplicación.

When you click the **Send** button, MainActivity sends the Intent and the Android system launches SecondActivity, which appears on the screen. To return to MainActivity, click the **Up** button (the left arrow in the app bar) or the Back button at the bottom of the screen

- Cuando hace clic en el botón Enviar, **MainActivity** envía la **Intent** y el sistema Android inicia **SecondActivity**, que aparece en la pantalla.

- Para volver a **MainActivity**, haga clic en el botón Arriba (la flecha izquierda en la barra de la aplicación) o en el botón Atrás en la parte inferior de la pantalla

Tarea 3: Enviar datos de la actividad principal a la segunda actividad

- En tarea anterior, agregó una Intent explícita a MainActivity que lanzó SecondActivity.
- También puede usar una Intent para enviar datos de una actividad a otra mientras la inicia.

Su objeto **Intent** puede pasar datos a la actividad de destino de dos maneras: en el campo de datos o en los extras de intención.

Los datos de la Intent son un URI que indica los datos específicos sobre los que se va a actuar. Si la información que desea pasar a una actividad a través de una indica los datos no es un URI, o tiene más de una información que desea enviar, puede colocar esa información adicional en los extras.

Los extras de **Intent** son pares clave/valor en un paquete. Un paquete es una colección de datos, almacenados como pares clave/valor. Para pasar información de una actividad a otra, coloca claves y valores en el paquete adicional de intención de la actividad de envío y luego los vuelve a sacar en la actividad de recepción.

En esta tarea, modificará la **Intent** explícita en MainActivity para incluir datos adicionales (en este caso, una cadena ingresada por el usuario) en el paquete extra de **Intent**. Luego,

modifica **SecondActivity** para recuperar esos datos del paquete adicional de **Intent** y mostrarlos en la pantalla.

Agregue un EditText al diseño MainActivity

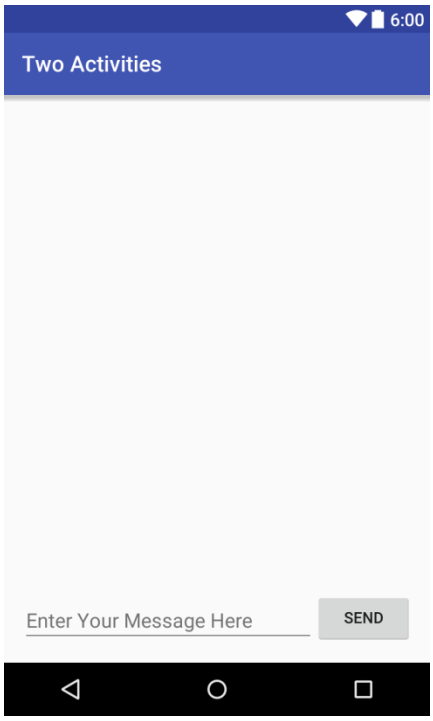
1. Abra actividad_principal.xml.

2. Arrastre un elemento de Texto sin formato (EditText) desde el panel Paleta hasta la parte inferior del diseño y agregue restricciones al lado izquierdo del diseño, la parte inferior del diseño y el lado izquierdo del botón Enviar. Establezca sus atributos en el panel Atributos de la siguiente manera:

1.

Attribute	Value
id	editText_main
Right margin	8
Left margin	8
Bottom margin	16
layout_width	match_constraint
layout_height	wrap_content
inputType	textLongMessage
hint	Enter Your Message Here
text	(Delete any text in this field)

El nuevo diseño en activity_main.xml se ve así:



3. Haga clic en la pestaña Texto para editar el código XML y extraiga la cadena " **Enter Your Message Here** " en un recurso llamado **editText_main**.

El código XML para el diseño debería parecerse a lo siguiente.

```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.MainActivity">

    <Button
        android:id="@+id/button_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginRight="16dp"
        android:text="@string/button_main"
        android:onClick="launchSecondActivity"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintRight_toRightOf="parent" />

    <EditText
        android:id="@+id/editText_main"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:ems="10"
        android:hint="@string/editText_main"
        android:inputType="textLongMessage"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button_main"
        app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Agregue una cadena a los extras de Intent

Los extras de **Intent** son pares **clave/valor** en un paquete. Un paquete es una colección de datos, almacenados como pares clave/valor. Para pasar información de una actividad a otra, coloca claves y valores en el paquete adicional de **Intent** de la actividad que envía y luego los vuelve a sacar en la actividad que recepta.

1. Abrir **MainActivity**.
2. Agreagar una constant publica en la parte superior de la clase para definir la llave para el Intent extra:

```
public static final String EXTRA_MESSAGE =
        "com.example.android.twoactivities.extra.MESSAGE";
```

3. Agregar una variable privada en la parte superior de la clase para contener el EditText:

```
private EditText mMessageEditText;
```

4. En el método onCreate(), use findViewById() obtener la referencia al EditText y asignarlo a la variable privada.

```
mMessageEditText = findViewById(R.id.editText_main);
```

5. En el método launchSecondActivity(), justo debajo de la nueva Intent, obtenga el texto de EditText como una cadena:

```
String message = mMessageEditText.getText().toString();
```

6. Agregue esa cadena al Intent como un extra con la constante EXTRA_MESSAGE como clave y la cadena como valor.

```
intent.putExtra(EXTRA_MESSAGE, message);
```

El método **onCreate**() in **MainActivity** debería verse asi:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mMessageEditText = findViewById(R.id.editText_main);
}
```

El método **launchSecondActivity**() en **MainActivity** debería verse asi:

```
public void launchSecondActivity(View view) {
    Log.d(LOG_TAG, "Button clicked!");
    Intent intent = new Intent(this, SecondActivity.class);
    String message = mMessageEditText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

Agregue un TextView a SecondActivity para el mensaje

1. Abra **activity_second.xml**.
2. Arrastre otro **TextView** al diseño debajo de text_header TextView y agregue restricciones al lado izquierdo del diseño y en la parte inferior de text_header.
3. Establezca los nuevos atributos de TextView en el panel Atributos de la siguiente manera:

1.

Attribute	Value
id	text_message
Top margin	8

Left margin	8
layout_width	wrap_content
layout_height	wrap_content
text	(Delete any text in this field)
textAppearance	AppCompat.Medium

El nuevo diseño tiene el mismo aspecto que en la tarea anterior, porque el nuevo TextView (todavía) no contiene ningún texto y, por lo tanto, no aparece en la pantalla.

El código XML para el diseño de actividad_segundo.xml debería tener un aspecto similar al siguiente:

```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.SecondActivity">

    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:text="@string/text_header"
        android:textAppearance=
                    "@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/text_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintStart_toStartOf="parent"
```

```
    app:layout_constraintTop_toBottomOf="@+id/text_header" />
</android.support.constraint.ConstraintLayout>
```

Modifique SecondActivity para obtener los extras y mostrar el mensaje

1. Abra **SecondActivity** para agregar código al método **onCreate**().
2. Obtenga la **Intent** que activó esta actividad:

```
Intent intent = getIntent();
```

3. Obtenga la cadena que contiene el mensaje de los extras de Intent utilizando la variable estática **MainActivity.EXTRA_MESSAGE** como clave:

```
String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

4.Use **findViewByID**() para obtener una referencia aTextView para el mensaje del diseño.

```
TextView textView = findViewById(R.id.text_message);
```

5.Establezca el texto de TextView en la cadena del Intent extra:

```
textView.setText(message);
```

6.Ejecute la app.

Cuando escribe un mensaje en MainActivity y hace clic en Enviar, SecondActivity se inicia y muestra el mensaje.

El método onCreate() de SecondActivity debería tener el siguiente aspecto:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_second);
   Intent intent = getIntent();
   String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
   TextView textView = findViewById(R.id.text_message);
   textView.setText(message);
}
```

Tarea 4: devolver los datos a la actividad principal

Hasta ahora tiene una aplicación que inicia una nueva actividad y le envía datos, el paso final es devolver los datos de la segunda actividad a la actividad principal. También usa una **Intent** y **extras** de intención para esta tarea**.**

Agregue un EditText y un Button al diseño de SecondActivity

1. Abra **strings.xml** y agregue recursos de cadena para el texto del botón y la sugerencia para EditText que agregará a SecondActivity:

```
<string name="button_second">Reply</string>
<string name="editText_second">Enter Your Reply Here</string>
```

Abra activity_main.xml and activity_second.xml.

2. Copie EditText y Button del archivo de diseño **activity_main.xml** y péguelos en el diseño **activity_second.xml.**

3. En activity_second.xml, modifique los valores de atributo para el Botón de la siguiente manera:

Old attribute value	New attribute value
android:id="@+id/button_main"	android:id="@+id/button_second"
android:onClick= "launchSecondActivity"	android:onClick="returnReply"
android:text= "@string/button_main"	android:text= "@string/button_second"

4. En **activity_second.xml**, modifique los valores de atributo para **EditText** de la siguiente manera:

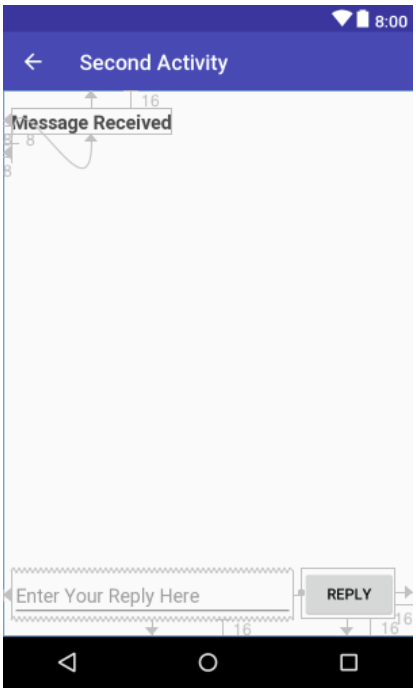
Old attribute value	New attribute value
android:id="@+id/editText_main"	android:id="@+id/editText_second"
app:layout_constraintEnd_toStartOf="@+id/button"	app:layout_constraintEnd_toStartOf="@+id/button_second"

android:hint= "@string/editText_main"	android:hint= "@string/editText_second"

5. En el editor de diseño XML, haga clic en **returnReply**, presione Alt+Enter y seleccione **Create 'returnReply(View)'** en '**SecondActivity**'.

Android Studio genera un método esqueleto para el controlador **returnReply().** Implementará este método en la siguiente tarea.

El nuevo diseño para activity_second.xml se ve así:



El código XML para el diseño de activity_second.xml se ve como lo siguiente:

```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.SecondActivity">

    <TextView
        android:id="@+id/text_header"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="16dp"
    android:text="@string/text_header"
    android:textAppearance="@style/TextAppearance.AppCompat.Medium"
    android:textStyle="bold"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/text_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="8dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text_header" />

<Button
    android:id="@+id/button_second"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/button_second"
    android:onClick="returnReply"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent" />

<EditText
    android:id="@+id/editText_second"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:ems="10"
    android:hint="@string/editText_second"
    android:inputType="textLongMessage"
    app:layout_constraintBottom_toBottomOf="parent"
```

```
    app:layout_constraintEnd_toStartOf="@+id/button_second"
    app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Crear una Intent de respuesta en la segunda actividad

Los datos de respuesta de la segunda Actividad a la Actividad principal se envían en un Intent adicional. Usted construye esta **Intent** de devolución y coloca los datos en él de la misma manera que lo hace para el **Intent** de envío.

1. Abra Segunda actividad.

2. En la parte superior de la clase, agregue una constante pública para definir la clave para el Intent adicional:

```
public static final String EXTRA_REPLY =
        "com.example.android.twoactivities.extra.REPLY";
```

3. Agregue una variable privada en la parte superior de la clase para contener EditText.

```
private EditText mReply;
```

4. En el método **onCreate**(), antes del código de Intent, use **findViewByID**() para obtener una referencia al **EditText** y asignarlo a esa variable privada:

```
mReply = findViewById(R.id.editText_second);
```

5. En el método **returnReply**(), obtenga el texto de **EditText** como una cadena:

```
String reply = mReply.getText().toString();
```

6. En el método **returnReply**(), crea una nueva **Intent** para la respuesta; no reutilices el objeto Intent que recibiste de la solicitud original.

```
Intent replyIntent = new Intent();
```

7. Agregue la cadena de respuesta de **EditText** a la nueva **Intent** como una **Intent** adicional. Como los extras son pares clave/valor, aquí la clave es **EXTRA_REPLY** y el valor es la respuesta:

```
replyIntent.putExtra(EXTRA_REPLY, reply);
```

8. Establezca el resultado en **RESULT_OK** para indicar que la respuesta fue exitosa. La clase de actividad define los códigos de resultado, incluidos RESULT_OK y RESULT_CANCELLED.

```
setResult(RESULT_OK,replyIntent);
```

9. Llame a **finish**() para cerrar la actividad y volver a MainActivity.

```
finish ();
```

El código para SecondActivity ahora debería ser el siguiente:

```
public class SecondActivity extends AppCompatActivity {
    public static final String EXTRA_REPLY =
                    "com.example.android.twoactivities.extra.REPLY";
    private EditText mReply;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        mReply = findViewById(R.id.editText_second);
        Intent intent = getIntent();
        String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
        TextView textView = findViewById(R.id.text_message);
        textView.setText(message);
    }

    public void returnReply(View view) {
        String reply = mReply.getText().toString();
        Intent replyIntent = new Intent();
        replyIntent.putExtra(EXTRA_REPLY, reply);
        setResult(RESULT_OK, replyIntent);
        finish();
    }
}
```

Agregar elementos TextView para mostrar la respuesta

MainActivity necesita una forma de mostrar la respuesta que envía **SecondActivity**. En esta tarea, agregará elementos **TextView** al diseño **activity_main.xml** para mostrar la respuesta en **MainActivity**.

Para facilitar esta tarea, copia los elementos de **TextView** que usaste en **SecondActivity**.

1. Abra **strings.xml** y agregue un recurso de cadena para el encabezado de respuesta:
<string name="text_header_reply">Reply Received</string>

2. Abra **activity_main.xml** y **activity_second.xml**.

3. Copie los dos elementos **TextView** del archivo de diseño de **activity_second.xml** y péguelos en el diseño de **activity_main.xml** sobre el Botón.

4. En **activity_main.xml**, modifique los valores de atributo para el primer TextView de la siguiente manera:

Old attribute value	**New attribute value**
android:id="@+id/text_header"	android:id="@+id/text_header_reply"
android:text= "@string/text_header"	android:text= "@string/text_header_reply"

5. En activity_main.xml, modique los valores de atributo para el Segundo TextView con lo siguiente:

Old attribute value	**New attribute value**
android:id="@+id/text_message"	android:id= "@+id/text_message_reply"
app:layout_constraintTop_toBottomOf="@+id/text_ header"	app:layout_constraintTop_toBottomOf="@+id/text_heade r_reply"

7. Agregue el atributo **android:visibility** a cada **TextView** para hacerlos inicialmente invisibles. (Tenerlos visibles en la pantalla, pero sin ningún contenido, puede resultar confuso para el usuario).

android:visibility="invisible"

Hará que estos elementos de TextView sean visibles después de que los datos de respuesta se transfieran desde la segunda Actividad.

El diseño **activity_main.xml** tiene el mismo aspecto que en la tarea anterior, aunque ha agregado dos nuevos elementos **TextView** al diseño. Debido a que configura estos elementos como invisibles, no aparecen en la pantalla.

El siguiente es el código XML para el archivo **activity_main.xml**:

```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.MainActivity">

    <TextView
        android:id="@+id/text_header_reply"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="16dp"
        android:text="@string/text_header_reply"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        android:visibility="invisible"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/text_message_reply"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:visibility="invisible"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/text_header_reply" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
```

```
        android:layout_marginRight="16dp"
        android:text="@string/button_main"
        android:onClick="launchSecondActivity"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent" />

    <EditText
        android:id="@+id/editText_main"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:ems="10"
        android:hint="@string/editText_main"
        android:inputType="textLongMessage"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button2"
        app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Obténer la respuesta del Intent extra y mostrarla

Cuando usa una **Intent** explícita para iniciar otra actividad, es posible que no espere recuperar ningún dato, solo está activando esa actividad. En ese caso, usa **startActivity()** para iniciar la nueva actividad, como lo hizo anteriormente en esta práctica. Sin embargo, si desea recuperar los datos de la actividad activada, debe iniciarla con **startActivityForResult().**

En esta tarea, modificará la aplicación para iniciar **SecondActivity** esperando un resultado, para extraer los datos devueltos del **Intent** y para mostrar esos datos en los elementos de **TextView** que creó en la última tarea.

1. Abra **MainActivity**.

2. Agregue una constante pública en la parte superior de la clase para definir la clave para un tipo particular de respuesta que le interese:

```
public static final int TEXT_REQUEST = 1;
```

3. Agregue dos variables privadas para contener el encabezado de respuesta y los elementos TextView de respuesta:

```
private TextView mReplyHeadTextView;
private TextView mReplyTextView;
```

4. En el método **onCreate**(), use **findViewByID()** para obtener referencias desde el diseño hasta el encabezado de respuesta y los elementos TextView de respuesta. Asigne esas instancias de vista a las variables privadas:

El método **onCreate**() completo ahora debería verse así:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      mMessageEditText = findViewById(R.id.editText_main);
      mReplyHeadTextView = findViewById(R.id.text_header_reply);
      mReplyTextView = findViewById(R.id.text_message_reply);
}
```

5. En el método **launchSecondActivity**(), cambie la llamada a **startActivity**() para que sea **startActivityForResult**() e incluya la clave **TEXT_REQUEST** como argumento:

```
startActivityForResult(intención, TEXT_REQUEST);
```

6. Sobrescriba el método de callback **onActivityResult**() con esta firma:

```
@Override
public void onActivityResult(int requestCode,
                    int resultCode, Intent data) {
}
```

Los tres argumentos de **onActivityResult**() contienen toda la información que necesita para manejar los datos devueltos: el código de solicitud que estableció cuando inició la actividad con **startActivityForResult**(), el código de resultado establecido en la actividad iniciada (generalmente uno de RESULT_OK o RESULT_CANCELED), y los datos de intención que contienen los datos devueltos por la actividad de lanzamiento.

7. Dentro de **onActivityResult**(), llame a **super.onActivityResult():**

```
super.onActivityResult(requestCode, resultCode, data);
```

8. Agregue código para probar **TEXT_REQUEST** para asegurarse de procesar el resultado de **Intent** correcto, en caso de que haya varios. También pruebe para RESULT_OK, para asegurarse de que la solicitud fue exitosa:

```
if (requestCode == TEXT_REQUEST) {
   if (resultCode == RESULT_OK) {
   }
}
```

La clase Actividad define los códigos de resultado. El código puede ser RESULT_OK (la solicitud fue exitosa), RESULT_CANCELED (el usuario canceló la operación) o RESULT_FIRST_USER (para definir sus propios códigos de resultado).

9. Dentro del bloque if interno, obtenga el Intent extra del Intent de respuesta (datos). Aquí la clave para el extra es la constante EXTRA_REPLY de SecondActivity:

```
String reply = data.getStringExtra(SecondActivity.EXTRA_REPLY);
```

10. Establezca la visibilidad del encabezado de respuesta en verdadera:

```
mReplyHeadTextView.setVisibility(View.VISIBLE);
```

11. Establezca el texto TextView de respuesta en la respuesta y establezca su visibilidad en verdadero:

```
mReplyTextView.setText(reply);
mReplyTextView.setVisibility(View.VISIBLE);
```

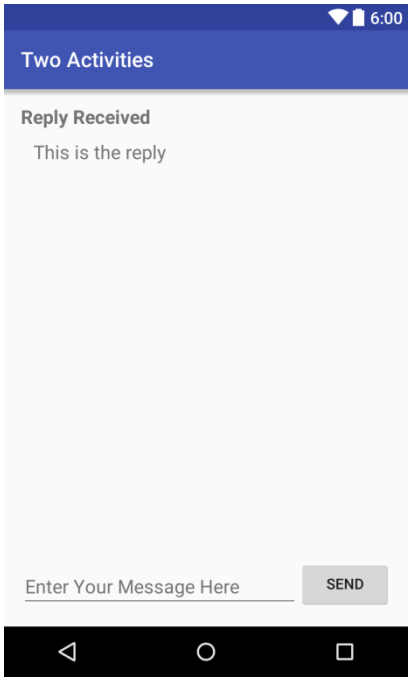
El método **onActivityResult**() completo ahora debería verse así:

```
@Override
public void onActivityResult(int requestCode,
                    int resultCode, Intent data) {
   super.onActivityResult(requestCode, resultCode, data);
   if (requestCode == TEXT_REQUEST) {
      if (resultCode == RESULT_OK) {
         String reply =
               data.getStringExtra(SecondActivity.EXTRA_REPLY);
         mReplyHeadTextView.setVisibility(View.VISIBLE);
         mReplyTextView.setText(reply);
         mReplyTextView.setVisibility(View.VISIBLE);
      }
```

```
    }
}
```

12. Ejecute la app.

    Ahora, cuando envía un mensaje a la segunda actividad y obtiene una respuesta, la actividad principal se actualiza para mostrar la respuesta.



Resumen

Descripción general:

• Una Actividad es un componente de aplicación que proporciona una sola pantalla enfocada en una sola tarea de usuario.

• Cada actividad tiene su propio archivo de diseño de interfaz de usuario.

• Puede asignar a sus implementaciones de actividad una relación padre/hijo para habilitar la navegación hacia arriba dentro de su aplicación.

• Una Vista puede hacerse visible o invisible con el atributo **android:visibility**.

Para implementar una Actividad:

• Elija **Archivo > Nuevo > Actividad** para comenzar desde una plantilla y realice los siguientes pasos automáticamente.

• Si no comienza desde una plantilla, cree una clase Java de actividad, implemente una interfaz de usuario básica para la actividad en un archivo de diseño XML asociado y declare la nueva actividad en **AndroidManifest.xml.**

Intent:

• Un Intent le permite solicitar una acción de otro componente en su aplicación, por ejemplo, para iniciar una Actividad desde otra. Un Intent puede ser explícito o implícito.

• Con un Intent explícito, indica el componente de destino específico para recibir los datos.

• Con una intent implícita, especifica la funcionalidad que desea, pero no el componente de destino.

• Un Intent puede incluir datos sobre los cuales realizar una acción (como un URI) o información adicional como extras del Intent.

• Los extras de intención son pares clave/valor en un paquete que se envían junto con la intención.