Cómo iniciar otra actividad

Cuando hayas completado la [lección anterior](https://developer.android.com/training/basics/firstapp/building-ui), tendrás una app que mostrará una actividad que consiste en una sola pantalla con un campo de texto y un botón **Enviar**. En esta lección, podrás agregar código a MainActivity con el objetivo de iniciar una nueva actividad para mostrar un mensaje cuando el usuario presione el botón **Enviar**.

**Nota:** En esta lección, se asume que usas Android Studio 3.0 o una versión posterior.

Cómo responder al botón Enviar

Sigue estos pasos para agregar un método a la clase MainActivity que se llama al presionar el botón **Enviar**:

1. En el archivo **app > java > com.example.myfirstapp > MainActivity**, agrega el siguiente método auxiliar sendMessage():

[KOTLIN](https://developer.android.com/training/basics/firstapp/starting-activity.html#kotlin)[JAVA](https://developer.android.com/training/basics/firstapp/starting-activity.html#java)

    public class MainActivity extends AppCompatActivity {  
        @Override  
        protected void onCreate(Bundle savedInstanceState) {  
            super.onCreate(savedInstanceState);  
            setContentView(R.layout.activity\_main);  
        }  
  
        **/\*\* Called when the user taps the Send button \*/  
        public void sendMessage(View view) {  
            // Do something in response to button  
        }**  
    }

Es posible que veas un error porque Android Studio no puede resolver la clase View que se usa como argumento del método. Para borrar el error, haz clic en la declaración View, coloca el cursor sobre ella y presiona Alt + Intro (o, en Mac, Opción + Intro) a fin de realizar una corrección rápida. Si aparece un menú, selecciona **Import class**.

1. Regresa al archivo **activity\_main.xml** para llamar al método desde el botón, de la siguiente manera:
   1. Selecciona el botón en el editor de diseño.
   2. En la ventana **Attributes**, ubica la propiedad **onClick** y selecciona **sendMessage [MainActivity]** en la lista desplegable.

De esta manera, cuando se presione el botón, el sistema llamará al método sendMessage().

Toma nota de los detalles de este método. Son necesarios para que el sistema reconozca el método como compatible con el atributo [android:onClick](https://developer.android.com/reference/android/view/View#attr_android:onClick). Específicamente, el método tiene las siguientes características:

* 1. Acceso público.
  2. Un valor de retorno [unit](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-unit/index.html) vacío o, en Kotlin, implícito.
  3. Un objeto [View](https://developer.android.com/reference/android/view/View) como el único parámetro. Este es el objeto [View](https://developer.android.com/reference/android/view/View) en el que hiciste clic al final del paso 1.

1. A continuación, deberás completar este método para leer el contenido del campo de texto y proporcionar dicho texto a otra actividad.

Cómo compilar un intent

Un [Intent](https://developer.android.com/reference/android/content/Intent) es un objeto que proporciona vinculación en tiempo de ejecución entre componentes separados, como dos actividades. El [Intent](https://developer.android.com/reference/android/content/Intent) representa la intención de realizar una tarea que tiene una app. Puedes usar los intents para varias tareas; pero, en esta lección, tu intent inicia otra actividad.

En MainActivity, agrega la constante EXTRA\_MESSAGE y el código sendMessage(), como se muestra a continuación:

[KOTLIN](https://developer.android.com/training/basics/firstapp/starting-activity.html#kotlin)[JAVA](https://developer.android.com/training/basics/firstapp/starting-activity.html#java)

    public class MainActivity extends AppCompatActivity {  
        **public static final String EXTRA\_MESSAGE = "com.example.myfirstapp.MESSAGE";**  
        @Override  
        protected void onCreate(Bundle savedInstanceState) {  
            super.onCreate(savedInstanceState);  
            setContentView(R.layout.activity\_main);  
        }  
  
        /\*\* Called when the user taps the Send button \*/  
        public void sendMessage(View view) {  
            **Intent intent = new Intent(this, DisplayMessageActivity.class);  
            EditText editText = (EditText) findViewById(R.id.editText);  
            String message = editText.getText().toString();  
            intent.putExtra(EXTRA\_MESSAGE, message);  
            startActivity(intent);**  
        }  
    }

Puedes esperar que Android Studio vuelva a mostrar errores **Cannot resolve symbol**. Para borrar los errores, presiona Alt + Intro o, en Mac, Opción + Regresar. Deberías terminar con las siguientes importaciones:

[KOTLIN](https://developer.android.com/training/basics/firstapp/starting-activity.html#kotlin)[JAVA](https://developer.android.com/training/basics/firstapp/starting-activity.html#java)

    import androidx.appcompat.app.AppCompatActivity;  
    import android.content.Intent;  
    import android.os.Bundle;  
    import android.view.View;  
    import android.widget.EditText;

Todavía queda un error para DisplayMessageActivity, pero está bien. Puedes arreglarlo en la siguiente sección.

Lo que sucede en sendMessage() es lo siguiente:

* El constructor [Intent](https://developer.android.com/reference/android/content/Intent) toma dos parámetros, [Context](https://developer.android.com/reference/android/content/Context) y [Class](https://developer.android.com/reference/java/lang/Class).

El parámetro [Context](https://developer.android.com/reference/android/content/Context) se usa primero, porque la clase [Activity](https://developer.android.com/reference/android/app/Activity) es una subclase de [Context](https://developer.android.com/reference/android/content/Context).

El parámetro [Class](https://developer.android.com/reference/java/lang/Class) del componente de la app, al que el sistema entrega el [Intent](https://developer.android.com/reference/android/content/Intent),, es, en este caso, la actividad que va a comenzar.

* El método [putExtra()](https://developer.android.com/reference/android/content/Intent#putExtra(java.lang.String,%20java.lang.String)) agrega el valor de EditText al intent. Un Intent puede transportar tipos de datos como pares clave-valor llamados *extras*.

Tu clave es una constante pública EXTRA\_MESSAGE porque la actividad siguiente usa la clave para obtener el valor de texto. Te recomendamos definir las claves para extras de intents con el nombre del paquete de la app como prefijo. De esta manera, se garantiza que las claves sean únicas en el caso de que tu app interactúe con otras apps.

* El método [startActivity()](https://developer.android.com/reference/android/app/Activity#startActivity(android.content.Intent)) inicia una instancia de DisplayMessageActivity que especifica el [Intent](https://developer.android.com/reference/android/content/Intent). Luego, deberás crear esa clase.

**Nota:** El componente de la arquitectura de navegación te permite usar el editor de navegación para asociar una actividad con otra. Una vez que se establezca la relación, podrás usar la API para iniciar la segunda actividad cuando el usuario active la acción asociada, por ejemplo, cuando haga clic en un botón. Para obtener más información, consulta [Navegación](https://developer.android.com/topic/libraries/architecture/navigation/).

Cómo crear la segunda actividad

Para crear la segunda actividad, sigue estos pasos:

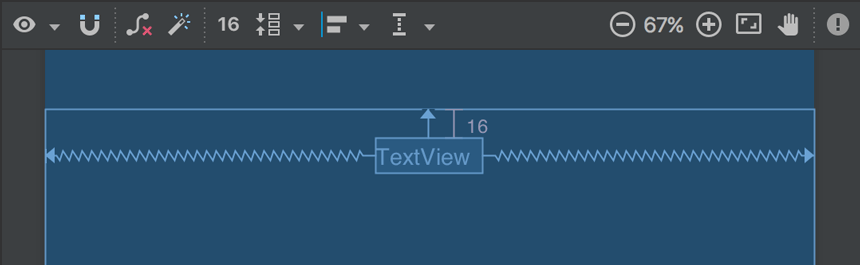
1. En la ventana **Project**, haz clic con el botón derecho en la carpeta de la **app** y selecciona **New > Activity > Empty Activity**.
2. En la ventana **Configure Activity**, ingresa "DisplayMessageActivity" en **Activity Name**. Deja todas las demás propiedades con sus valores predeterminados y haz clic en **Finish**.

Android Studio realiza tres acciones automáticamente:

* Crea el archivo DisplayMessageActivity.
* Crea el archivo de diseño activity\_display\_message.xml, que se corresponde con el archivo DisplayMessageActivity.
* Agrega el elemento [<activity>](https://developer.android.com/guide/topics/manifest/activity-element) correspondiente en AndroidManifest.xml.

Si ejecutas la app y presionas el botón en la primera actividad, se iniciará la segunda, pero estará vacía porque usa el diseño vacío proporcionado por la plantilla.

Cómo agregar una vista de texto

**Figura 1:** La vista de texto centrada en la parte superior del diseño

La nueva actividad incluye un archivo de diseño en blanco. Sigue estos pasos para agregar una vista de texto al lugar donde aparece el mensaje:

1. Abre el archivo **app > res > diseño > activity\_display\_message.xml**.
2. Haz clic en **Enable Autoconnection to Parent**  en la barra de herramientas De esta manera, se habilita la conexión automática. Consulta la figura 1.
3. En el panel **Palette**, haz clic en **Text**, arrastra un elemento **TextView** al diseño y suéltalo cerca de la parte central superior del diseño de manera que se acople a la línea vertical que aparecerá. La opción de conexión automática agrega restricciones a la derecha y a la izquierda para ubicar la vista en el centro horizontal.
4. Crea una restricción más desde la parte superior de la vista de texto hasta la parte superior del diseño para que se vea como en la figura 1.

De manera opcional, puedes realizar algunos ajustes en el estilo de texto si expandes **textAppearance** en el panel **Common Attributes** de la ventana **Attributes** y cambias los atributos como **textSize** y **textColor**.

Cómo mostrar el mensaje

En este paso, puedes modificar la segunda actividad para mostrar el mensaje que pasó la primera actividad.

1. En DisplayMessageActivity, agrega el siguiente código al método onCreate():

[KOTLIN](https://developer.android.com/training/basics/firstapp/starting-activity.html#kotlin)[JAVA](https://developer.android.com/training/basics/firstapp/starting-activity.html#java)

    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_display\_message);  
         **// Get the Intent that started this activity and extract the string  
        Intent intent = getIntent();  
        String message = intent.getStringExtra(MainActivity.EXTRA\_MESSAGE);  
  
        // Capture the layout's TextView and set the string as its text  
        TextView textView = findViewById(R.id.textView);  
        textView.setText(message);**  
    }

1. Presiona Alt + Intro o, en Mac, Opción + Regresar para importar estas otras clases necesarias:

[KOTLIN](https://developer.android.com/training/basics/firstapp/starting-activity.html#kotlin)[JAVA](https://developer.android.com/training/basics/firstapp/starting-activity.html#java)

    import androidx.appcompat.app.AppCompatActivity;  
    import android.content.Intent;  
    import android.os.Bundle;  
    import android.widget.TextView;

Cómo agregar navegación ascendente

En cada pantalla de tu app que no sea el punto de entrada principal, es decir, todas las pantallas que no sean la "pantalla principal", se debe proporcionar navegación para que el usuario pueda regresar a la pantalla superior lógica en la jerarquía de la app. Para ello, agrega un botón **Arriba** en la [barra de la app](https://developer.android.com/training/appbar/index).

Para agregar un botón **Arriba**, debes declarar qué actividad es la lógica principal en el archivo [AndroidManifest.xml](https://developer.android.com/guide/topics/manifest/manifest-intro). Abre el archivo en **app > manifests > AndroidManifest.xml**, busca la etiqueta <activity> para DisplayMessageActivity y reemplázala con lo siguiente:

    <activity android:name=".DisplayMessageActivity"  
              android:parentActivityName=".MainActivity">  
        <!-- The meta-data tag is required if you support API level 15 and lower -->  
        <meta-data  
            android:name="android.support.PARENT\_ACTIVITY"  
            android:value=".MainActivity" />  
    </activity>

El sistema Android agregará automáticamente el botón **Arriba** en la barra de app.

Cómo ejecutar la app

Haz clic en **Apply Changes** , en la barra de herramientas, para ejecutar la app. Cuando se abra, escribe un mensaje en el campo de texto y presiona **Send** para que el mensaje aparezca en la segunda actividad.

**Figura 2:** Imagen de la app abierta, con texto que se ingresa a la izquierda de la pantalla y se muestra a la derecha

Eso es todo. Compilaste tu primera app para Android.

Si quieres continuar aprendiendo los conceptos básicos sobre el desarrollo de apps para Android, regresa a [Cómo compilar tu primera app](https://developer.android.com/training/basics/firstapp/) y sigue los otros vínculos que se proporcionan allí.