



Creación de una primera aplicación Android

Índice



Creación de una primera aplicación Android

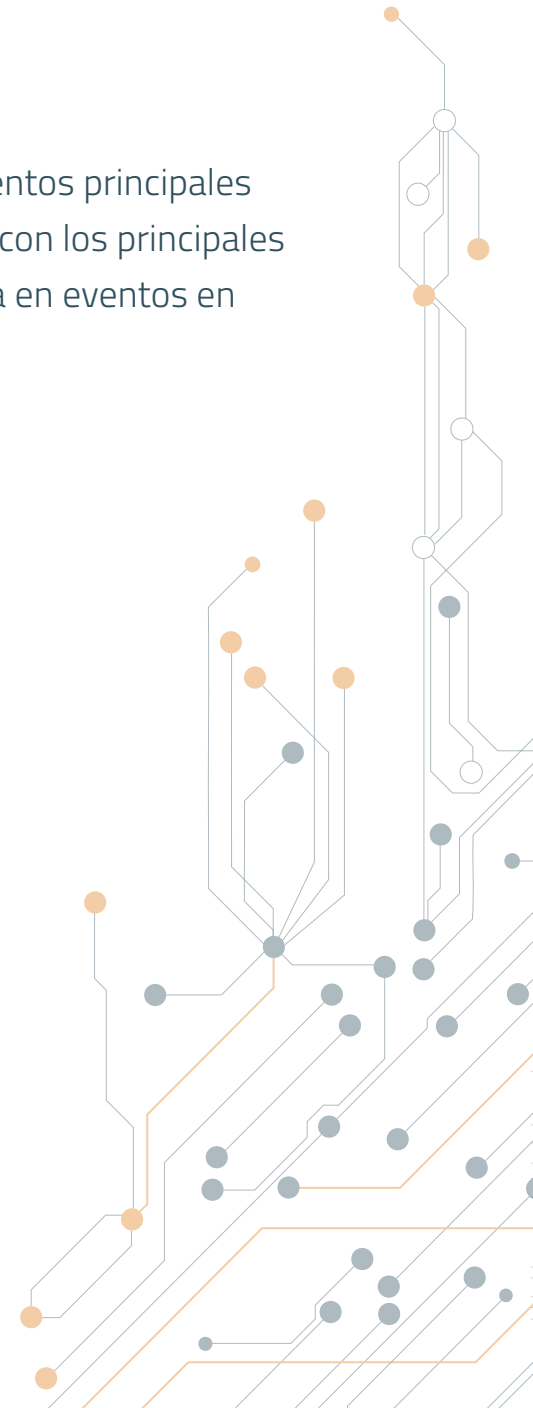
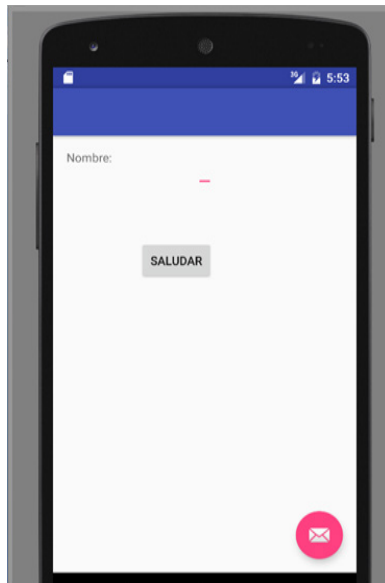
1 Introducción	3
2 Presentación de la aplicación	3
3 Diseño de la interfaz gráfica	5
4 Código de la aplicación	10
5 Probando la aplicación	14

1. Introducción

Una vez que hemos analizado la estructura de un proyecto Android Studio y presentado los elementos principales del mismo, vamos a crear una primera aplicación Android que nos va a servir para familiarizarnos con los principales controles gráficos y el diseño de interfaces, así como con los principios de la programación basada en eventos en Android.

2. Presentación de la aplicación

El objetivo es realizar un sencillo programa que tenga el siguiente aspecto:

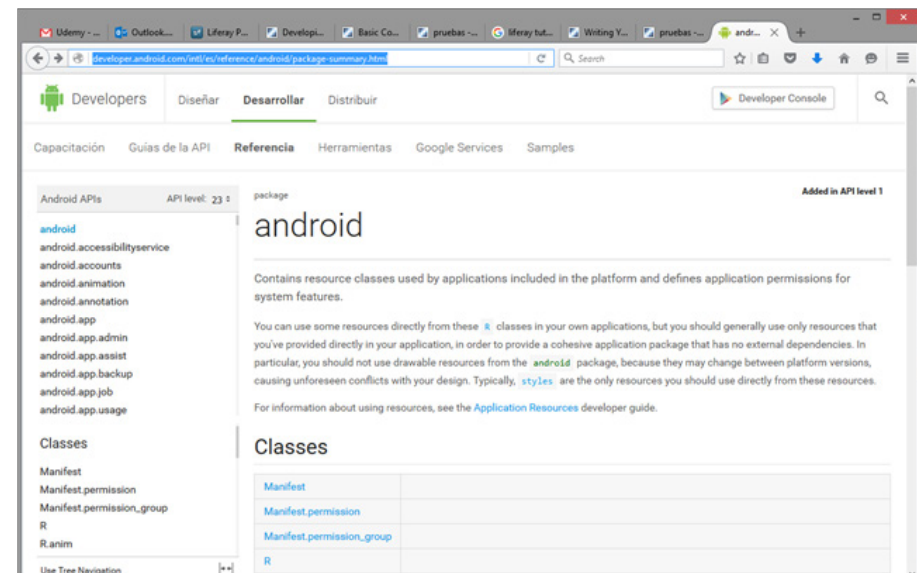


Se solicitará al usuario la introducción de un nombre y al pulsar el botón "Saludar", se mostrará un mensaje personalizado para el usuario justo debajo del botón de pulsación:



Antes de nada, será conveniente tener a mano el javadoc o ayuda de las clases de Android. La dirección es:

<http://developer.android.com/intl/es/reference/android/package-summary.html>

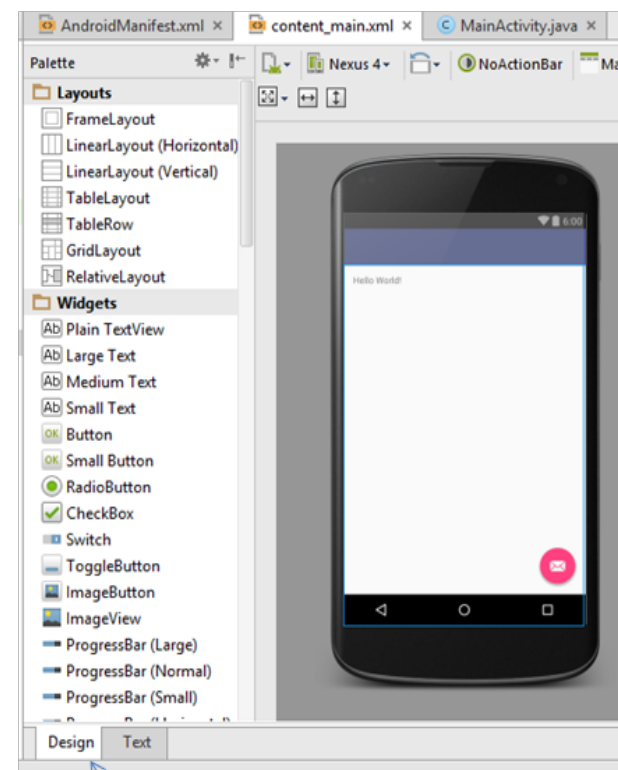


Aquí encontramos tanto las clases básicas Java o clases core, como las específicas de Android, por lo que es un importante recurso de ayuda al programador durante el desarrollo de las aplicaciones.

3. Diseño de la interfaz gráfica

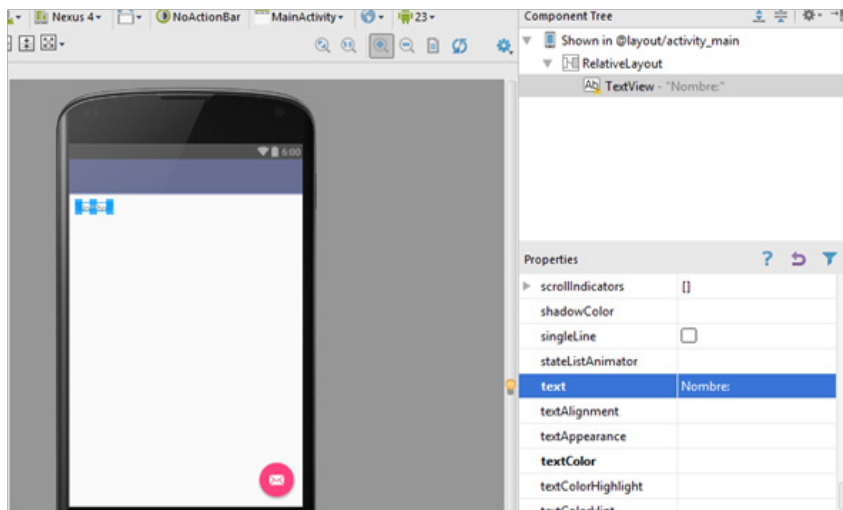
Vamos a partir del proyecto ejemplo01 creado anteriormente y lo primero será realizar el diseño de la interfaz gráfica de la aplicación, que estará formada por cuatro widgets: dos TextView, en uno de ellos se mostrará el texto "Nombre:" y en otro el mensaje de saludo generado por la aplicación, un EditText o caja de texto donde el usuario introducirá el nombre solicitado y un Button que servirá para generar la acción sobre la aplicación.

Para realizar el diseño de la aplicación, nos desplazamos hasta el archivo de plantilla main_content.xml y nos situamos dentro de la vista de diseño pulsando la pestaña design que se encuentra debajo de la paleta de controles:

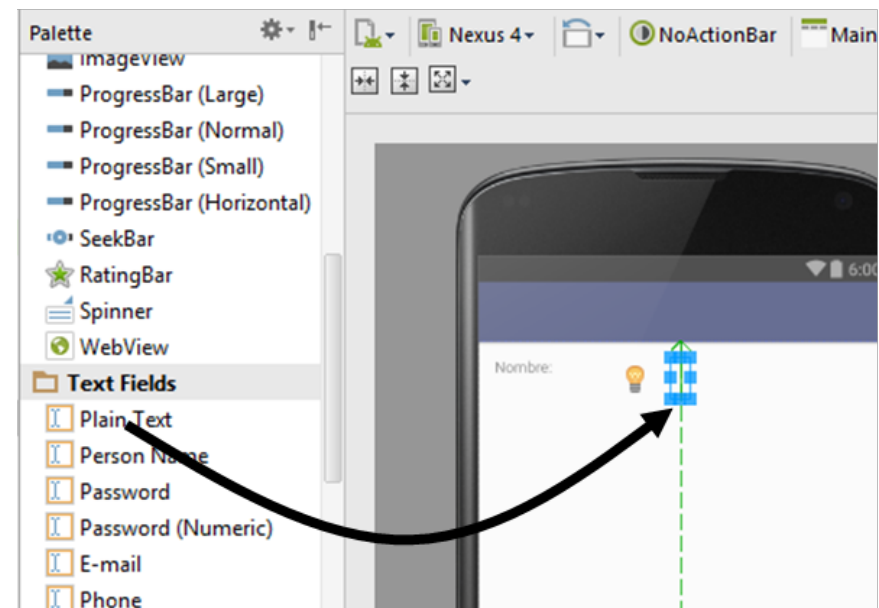


Vista diseño

Vamos a reutilizar el TextView que nos incorpora por defecto el proyecto en la actividad. Nos colocamos con el ratón sobre el control en la vista de diseño y hacemos clic sobre el mismo para seleccionarlo, después, en la ventana de propiedades que se encuentra a la derecha del diseñador nos desplazamos hasta la propiedad "text" y cambiamos el valor actual por "Nombre:"



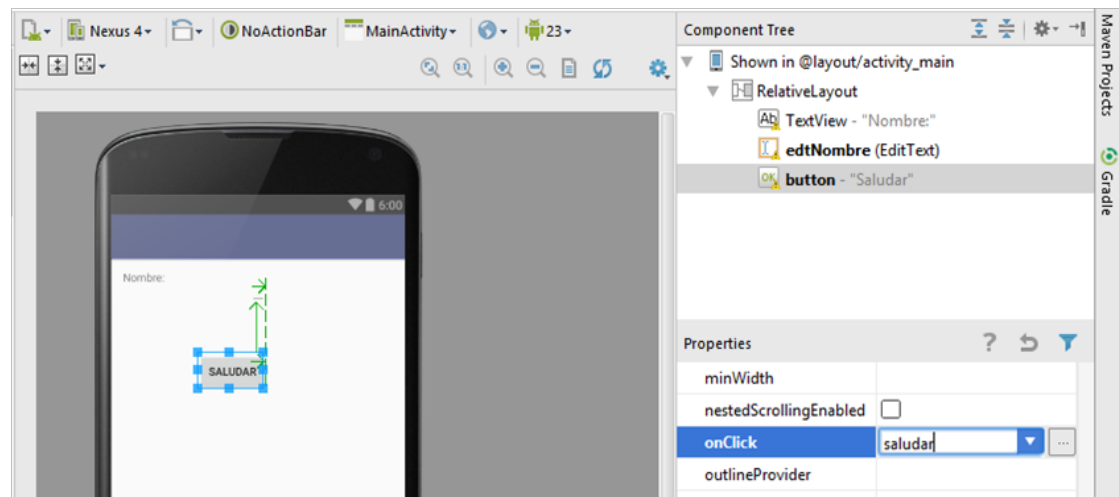
A continuación, añadiremos el control EditText al diseñador, para ello, nos desplazamos en la paleta hasta la sección Text Fieles y arrastraremos el control Plain Text hacia el diseñador, colocándolo a la derecha del TextView:



Dentro de la ventana de propiedades, dejaremos la propiedad text de este control en blanco y como valor de propiedad id estableceremos "getNombre". Este valor será el utilizado por Android para definir el identificador del control en la clase R.

Seguidamente, utilizando la misma técnica de arrastrar y soltar, añadiremos el botón de pulsación a la interfaz gráfica, debajo de los dos controles anteriores. Además de la propiedad text, que establecemos al valor "Saludar", también modificaremos la propiedad enclocó al valor "saludar". Esta propiedad contiene el nombre del método que se ejecutará al pulsar el botón, método que definiremos más adelante en la clase de actividad.

Durante el arrastre y colocación de los controles en el diseñador, hay que tener cuidado al intentar mover o cambiar el tamaño de cualquier control, ya que puede descuadrarse todo el conjunto de la interfaz. El motivo es el layout utilizado pues, al ser de tipo RelativeLayout, la colocación de los controles es siempre relativa a los demás y un cambio de posición o tamaño en alguno de ellos puede afectar al resto.



Finalmente, arrastramos un nuevo TextView y lo colocamos debajo del botón. Su propiedad text la dejaremos en blanco, mientras que en la propiedad id pondremos valor "tv Saludo".

El código final del documento content_main.xml, que veremos en la vista de texto, quedará más o menos como se muestra en el siguiente listado:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main" tools:context=".MainActivity">
    <TextView android:text="Number:" android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView2" />
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/edtNombre"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Saludar"
        android:id="@+id/button"
        android:layout_below="@+id/edtNombre"
        android:layout_alignRight="@+id/edtNombre"
        android:layout_alignEnd="@+id/edtNombre"
        android:layout_marginTop="61dp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/tvSaludo"
        android:layout_centerVertical="true"
        android:layout_toRightOf="@+id/textView2"
        android:layout_toEndOf="@+id/textView2" />
</RelativeLayout>
```


Los valores de las propiedades de los controles se definen como atributos de las etiquetas de cada control. Seguidamente, comentaremos algunos de los atributos que podemos encontrar en la mayoría de componentes gráficos Android:

- **id.** Identificador del control, es un valor que permite identificar al control de forma única dentro de la aplicación y que se emplea para poder obtener una referencia al mismo desde código. Observa como el valor de la propiedad id (atributo android:id) sigue en xml el formato “@+id/id_control”.
- **text.** Representa el texto que muestra el control. Encontramos esta propiedad en controles que van a mostrar un texto en la interfaz gráfica, como etiquetas (TextView), cajas de texto y botones.
- **layout_width.** Ancho del componente. Encontramos esta propiedad no solo en los controles, también en los componentes de tipo Layout. Sus posibles valores son:
 - **wrap_content.** El ancho del componente se adapta a su contenido.
 - **match_parent.** El ancho se adapta al del componente que lo contiene. En el caso de los controles, será el layout y en el caso de estos su contenedor será la propia actividad.
- **layout_height.** Alto del componente. Puede adquirir los mismos valores que layout_width



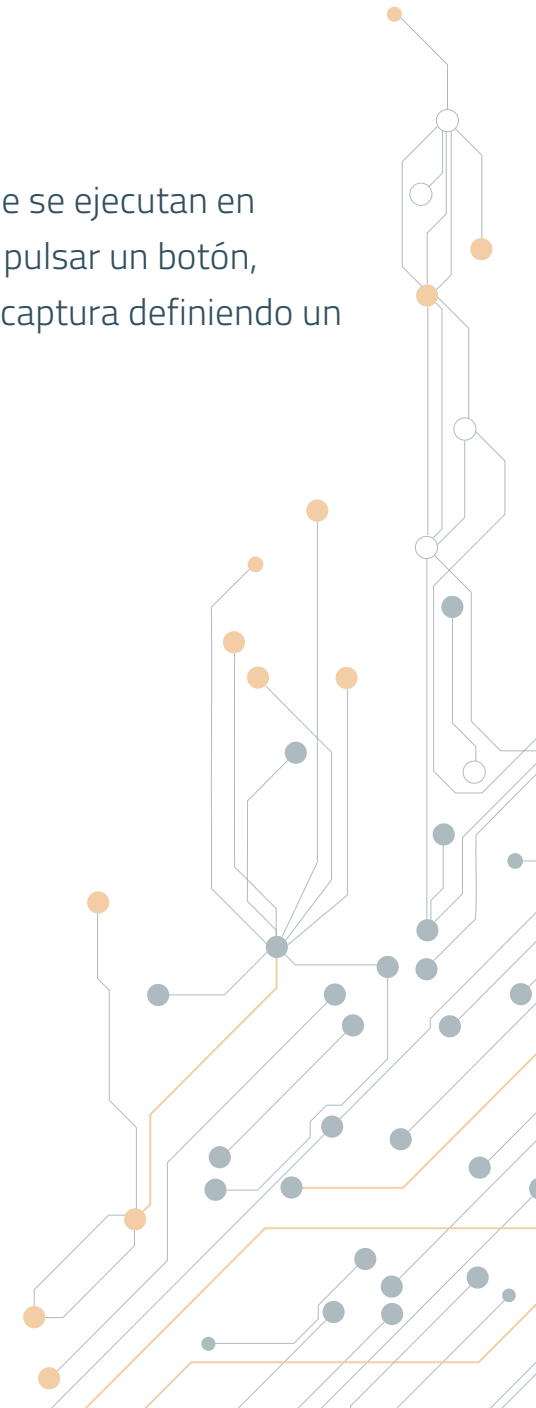
4. Código de la aplicación

Las aplicaciones gráficas Android son conducidas por eventos, lo que significa que las acciones que se ejecutan en la misma tienen lugar cuando el usuario provoca algún suceso en la interfaz de la actividad, como pulsar un botón, seleccionar un elemento de una lista, etc. En el caso de los botones, la respuesta al evento clic se captura definiendo un método en la actividad con el formato siguiente:


```
public void nombreMetodo(View v){  
:  
}
```

El método podrá llamarse de cualquier manera, pero debe ser public y void y declarar un parámetro de tipo View. Este parámetro representa el objeto sobre el que se producirá el evento, y es que **todas las clases de objetos gráficos Android heredan View**.

Para enlazar el método con el control, se deberá especificar el nombre de dicho método en la propiedad onClick del botón. En el ejemplo que estamos desarrollando, establecimos el valor "saludar" en la propiedad onClick de nuestro botón, luego así tendrá que llamarse el método de respuesta al evento.



Este será el código completo de la actividad con la funcionalidad del método saludar:



```
package android.moviles.ejemplo01;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

    public void saludar(View v) {

        //obtenemos una referencia al control campo de texto
        EditText nombre = (EditText) this.findViewById(R.id.edtNombre);
        //obtenemos una referencia al control que
        //mostrará el mensaje de saludo
        TextView saludo = (TextView) this.findViewById(R.id.tvSaludo);
        //recuperamos el contenido del campo de texto
        //y formamos el mensaje de saludo
        String texto = "Bienvenido Sr./a " + nombre.getText();
        //volcamos el mensaje en el TextView
        saludo.setText(texto);
    }
}
```

Pasamos a comentar a continuación el código que te acabamos de mostrar.

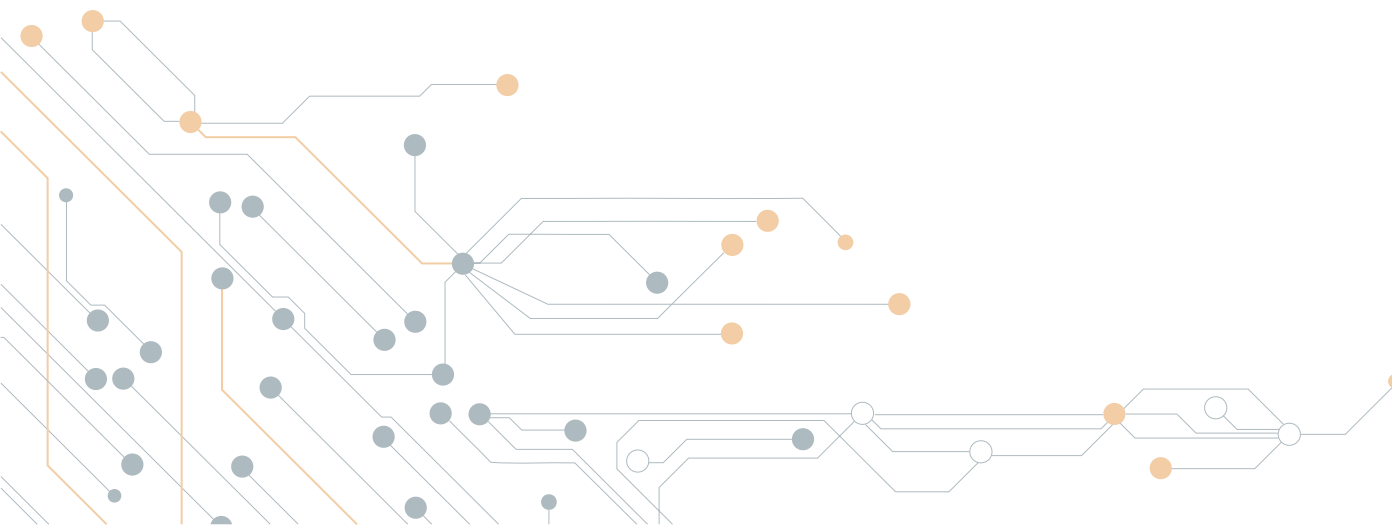
En las dos primeras instrucciones, utilizando el método `findViewById()` heredado de `Activity`, obtenemos una referencia a los controles que vamos a manipular desde código a partir del identificador de los mismos. Como el método devuelve un tipo `View` y cada widget o control pertenece a una subclase de `View`, debemos realizar una conversión o casting al tipo de control correspondiente. Las clases de controles se encuentran definidas dentro del paquete **`android.widget`**.

Los identificadores de los controles están definidos en la clase interna `id` de `R`. El método *`findViewById()`* devuelve el objeto como tipo `View`, de ahí que tengamos que hacer una conversión al tipo específico de widget.

Tanto el control `TextView` como `EditText`, disponen de los métodos `getText()` y `setText()` que nos permiten recuperar y modificar su contenido, respectivamente.

Un detalle a tener en cuenta durante el desarrollo de aplicaciones con Android Studio es que **podemos incluir automáticamente las importaciones** de las clases utilizadas pulsando la combinación de teclas **`Alt+Enter`**.

Las aplicaciones gráficas Android son conducidas por eventos, lo que significa que las acciones que se ejecutan en la misma tienen lugar cuando el usuario provoca algún suceso en la interfaz de la actividad, como pulsar un botón, seleccionar un elemento de una lista, etc. En el caso de los botones, la respuesta al evento clic se captura definiendo un método en la actividad.

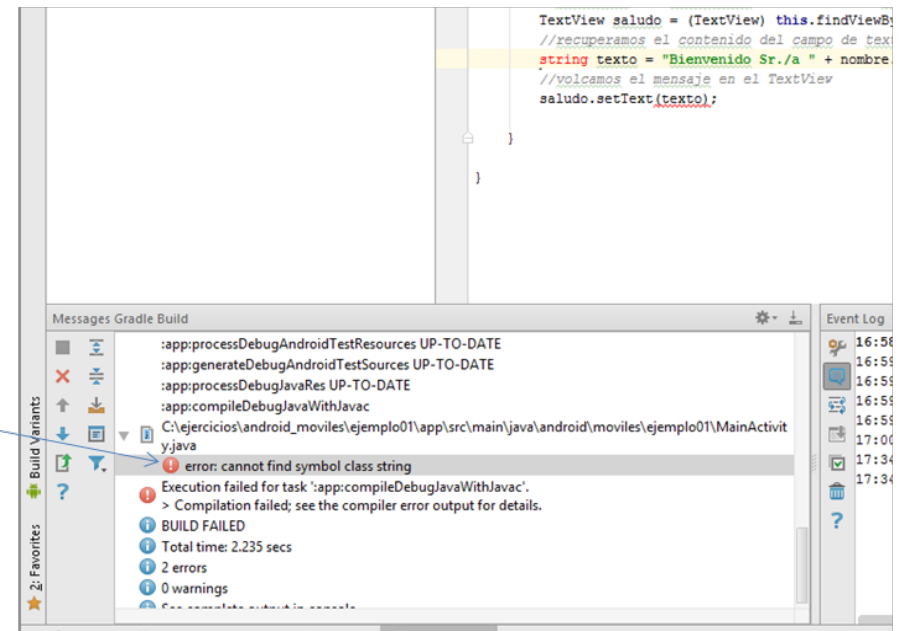


A diferencia de otros entornos de desarrollo como eclipse, Android Studio no resulta muy intuitivo a la hora de mostrar los errores de compilación de nuestros programas, tendremos que fijarnos en la columna de la derecha del editor, donde mediante una línea horizontal de color rojo se nos avisa de la existencia de errores en el código. Al pulsar sobre la línea, nos aparece un mensaje descriptivo del error y el cursor se sitúa en la instrucción errónea:



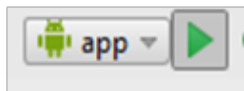
También podemos forzar la compilación del código con la opción de menú Build-> Make Project (combinación de teclas Ctrl+F9). Si se producen errores durante la compilación, estos se mostrarán en la ventana Messages Gradle Build que aparece en la parte inferior izquierda del escritorio:

Error de compilación

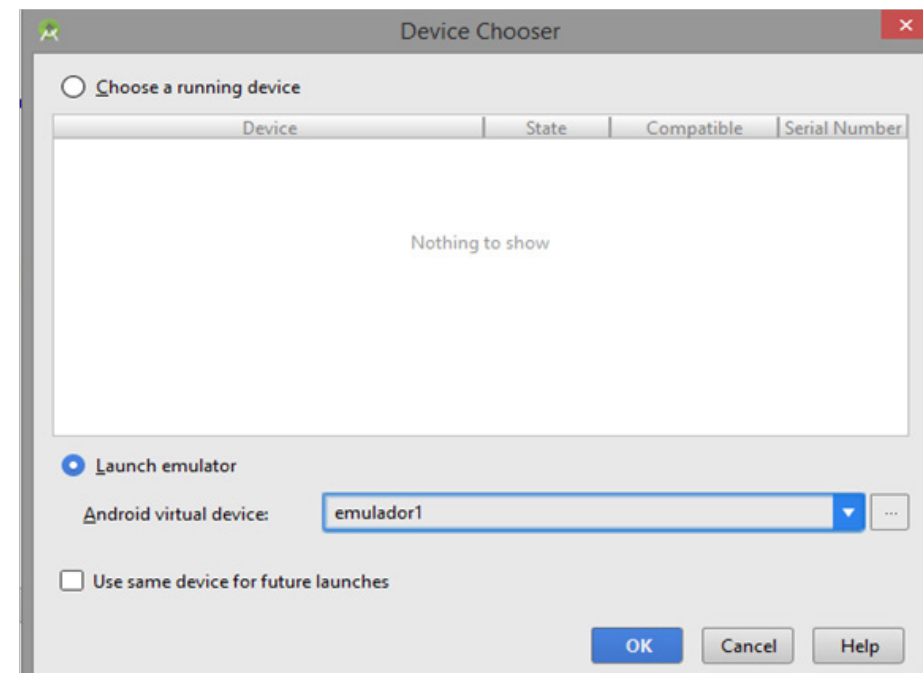


5. Probando la aplicación

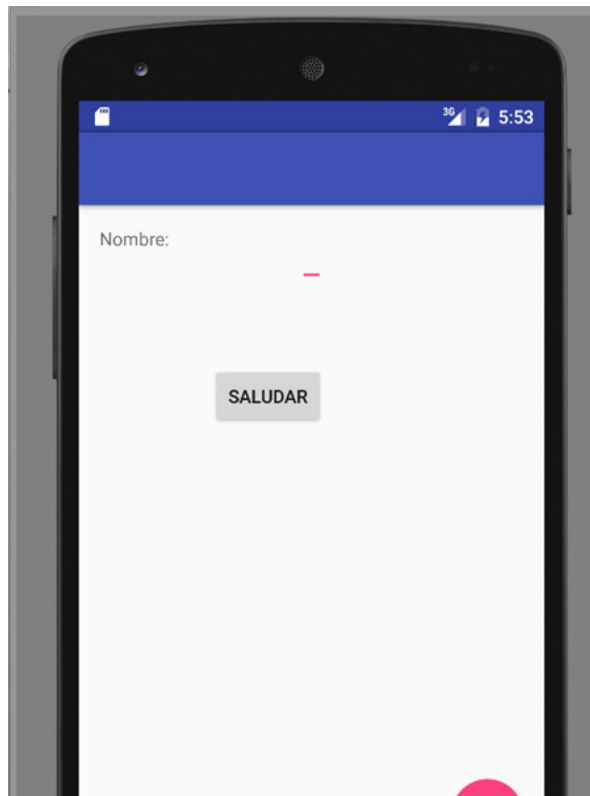
Una vez codificado el método de respuesta al evento clic del botón y corregidos los posibles errores de compilación, pasamos a probar la aplicación en el emulador para comprobar su funcionamiento. Para llevar esta operación a cabo, pulsamos el botón de ejecución que se encuentra en la barra de herramientas:



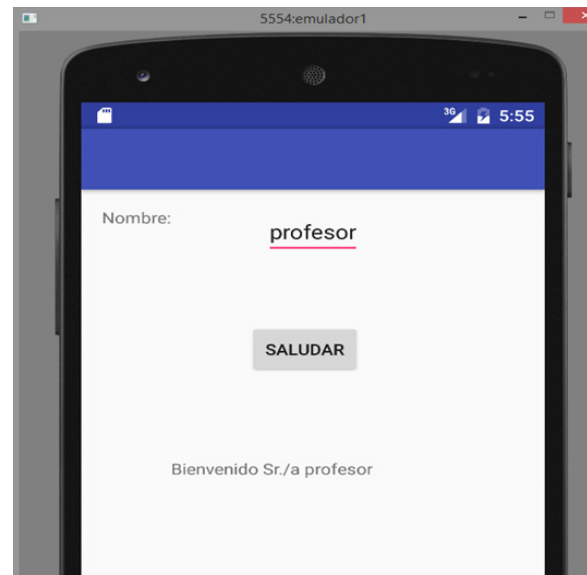
Al cabo de unos segundos, aparecerá un cuadro de diálogo para que elijamos el dispositivo en el que queremos probar nuestra aplicación:



Por defecto, nos aparece la opción “Launch emulador” marcada y el emulador que añadimos al principio seleccionado. Pulsamos el botón “Ok” tras unos minutos, nos aparecerá el emulador con la actividad cargada:

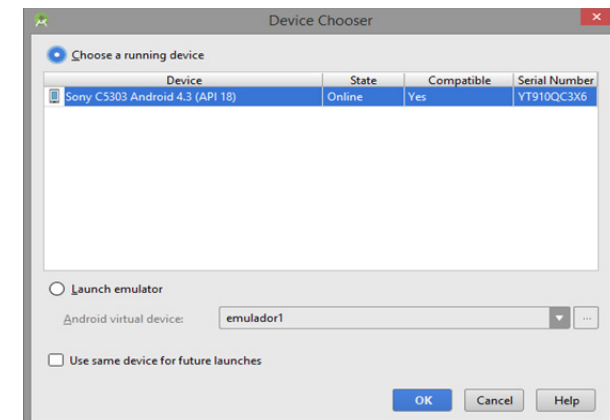


Si introducimos un texto y pulsamos el botón, el mensaje de saludo se mostrará debajo del botón:



Podemos probar también nuestra aplicación en un dispositivo real. Para ello, conectaremos el terminal al puerto USB de nuestro equipo y dentro de los ajustes del dispositivo, en la sección de opciones para el desarrollador, activaremos **la depuración USB**. Es posible que con algún modelo de teléfono o tablet tengamos que instalar algún software adicional en el equipo, pero por regla general, las acciones anteriores bastarán para que el terminal sea reconocido por Android Studio.

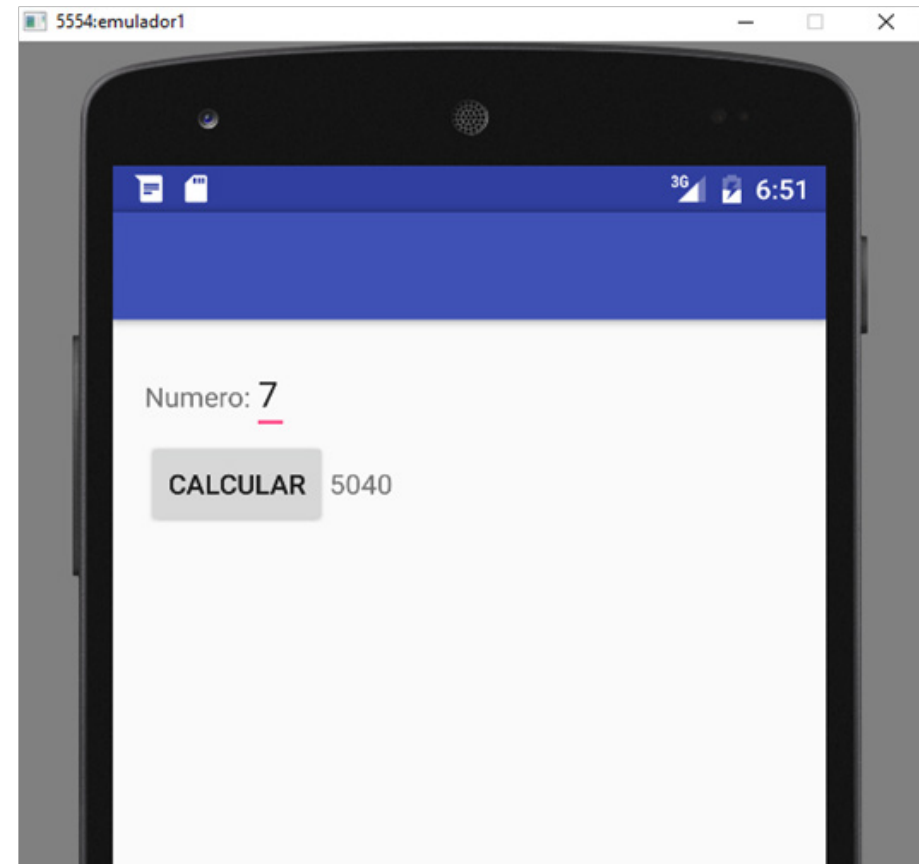
Si el dispositivo es reconocido, aparecerá en la lista de dispositivos disponibles cuando ejecutemos la aplicación:



Ejercicio resuelto

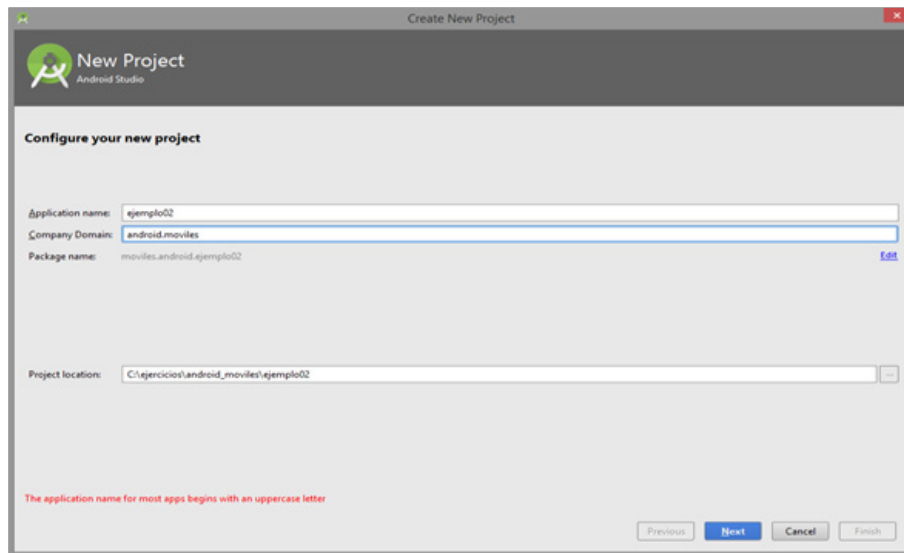
Vamos a realizar un segundo ejercicio para resumir los pasos seguidos en la creación de una aplicación Android. Aprovecharemos este ejercicio para explicar una forma alternativa de diseñar la interfaz gráfica de la aplicación.

El ejercicio consiste en realizar un programa que realice el cálculo del factorial de un número. Mediante un campo de texto de edición se solicitará al usuario el número y al pulsar el botón calcular, se mostrará el resultado del factorial en un TextView. El aspecto del programa será el que se indica en la siguiente imagen:




En primer lugar, crearemos el nuevo proyecto Android Studio seleccionando la opción de menú File -> New -> New Project. Le pondremos como nombre de proyecto ejemplo02 y para el resto de datos seguiremos los mismos pasos que en el ejercicio anterior:

Como hemos comentado, vamos a realizar el diseño de la interfaz gráfica de forma ligeramente diferente a como lo hicimos en el ejemplo anterior. En este caso, vamos a utilizar layouts de tipo LinearLayout en vez de RelativeLayout, los cuales nos permiten colocar los controles de izquierda a derecha (LinearLayout horizontal) y de arriba hacia abajo (LinearLayout vertical).



Para ello, una vez creado el proyecto, nos iremos a la vista de texto del archivo content-main.xml y cambiaremos la etiqueta RelativeLayout por LinearLayout:

Cambio de layout

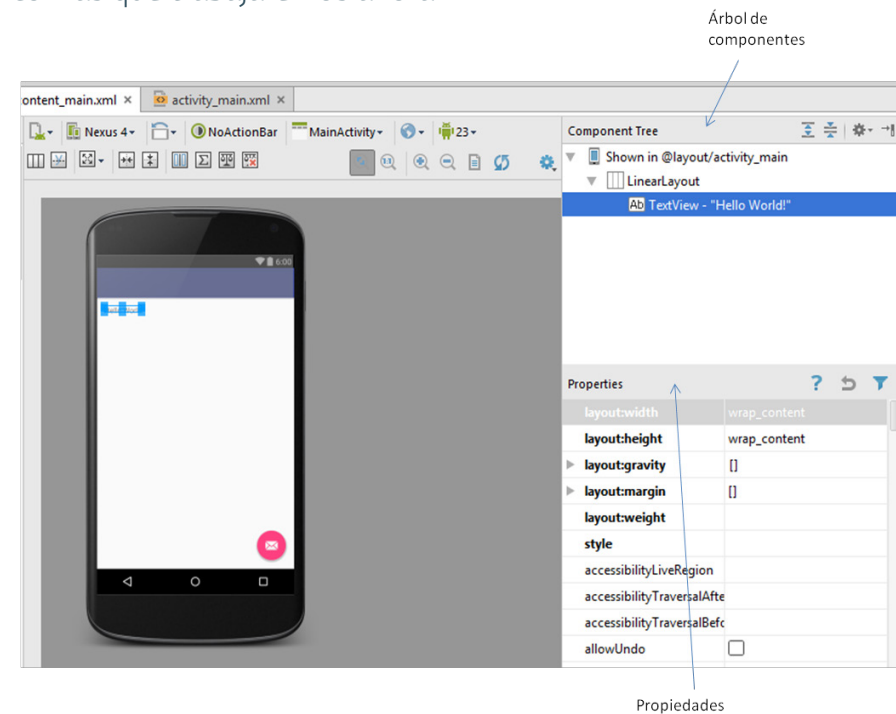


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res/app"
    android:layout_height="match_parent"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main"
    tools:layout_editor_previous_version="23.0.0"
    android:orientation="vertical">

    <TextView android:text="Hello World!"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_margin="10dp">

</LinearLayout>
```

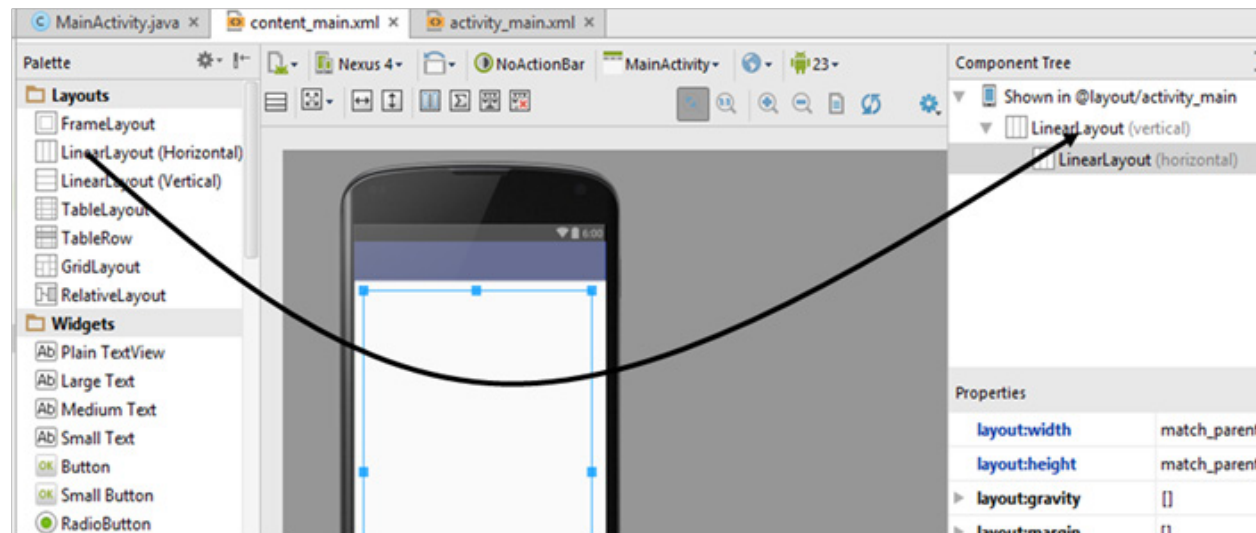
Seguidamente, cambiamos a la vista de diseño de la plantilla y nos fijamos en las ventanas "Component tree" y "Properties", que son con las que trabajaremos ahora:



La ventana “component tree” nos muestra el árbol de objetos gráficos que forman la plantilla. Lo primero que haremos aquí será borrar el componente TextView que nos incorpora de forma automática el IDE, para lo cual, nos situamos con el ratón sobre el objeto TextView y en el menú que aparece al pulsar el botón derecho elegimos la opción Delete.

Seguidamente, en la ventana de propiedades modificaremos la propiedad orientation del LinearLayout y le pondremos el valor vertical. La idea es definir un layout principal con orientación vertical y cada grupo de controles que tengan que situarse horizontalmente se incluyan dentro de un sublayout con orientación horizontal.

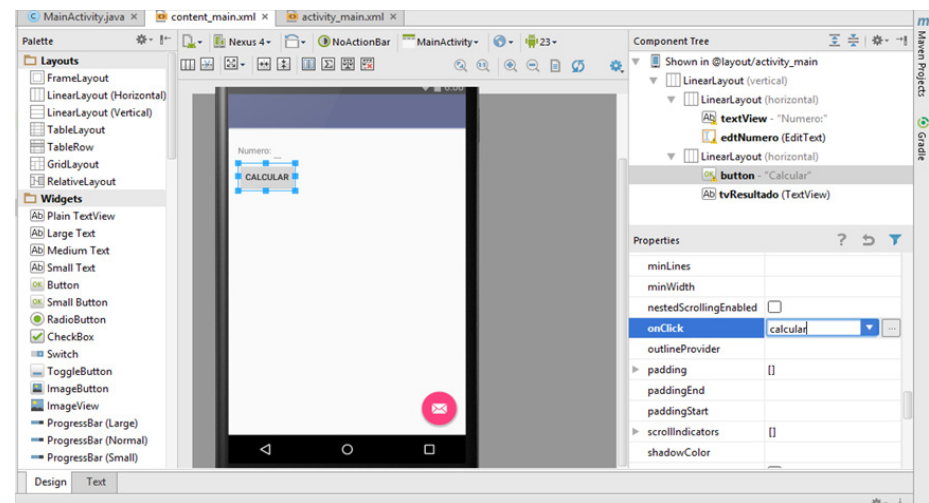
Así pues, antes de colocar la primera fila de controles añadiremos un LinearLayout horizontal dentro del layout principal, y lo haremos arrastrando el control desde la paleta hasta la ventana del árbol de componentes, sobre el LinearLayout principal:




En este LinearLayout que acabamos de incorporar, modificaremos el valor de su propiedad `layout_height` al valor `"wrap_content"` para que el alto se adapte a su contenido, pues el valor `"match_parent"`, que es el que se establece por defecto, hará que ocupe todo el alto de la actividad.

Para crear la primera fila de componentes, formada por un TextView y un EditText, arrastraremos estos controles hasta el LinearLayout horizontal que acabamos de crear. La propiedad `text` del TextView la estableceremos al valor `"Número:"`, mientras que la propiedad `id` del EditText tendrá el valor `"edtNumero"`.

Seguidamente, incluiremos un nuevo LinearLayout horizontal al mismo nivel que el anterior, para ello lo arrastramos a la ventana de componentes y lo soltamos sobre el layout vertical. En este segundo LinearLayout horizontal pondremos un botón con el texto `"Calcular"` y en su propiedad `onClick` indicaremos el valor `"calcular"`, además de un TextView vacío con identificador `"tvResultado"`. El diseño quedará como se indica en la siguiente imagen:



En el método `calcular()` dentro de la actividad incluiremos el código para el cálculo de factorial, con lo que la clase nos quedará como se indica en el siguiente listado:



```
public class MainActivity extends AppCompatActivity {

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

    public void calcular(View v){
        EditText edtNumero=
            (EditText)this.findViewById(R.id.edtNumero);
        TextView tvResultado=
            (TextView)this.findViewById(R.id.tvResultado);
        //convertimos el valor de texto a número
        int num=Integer.parseInt(edtNumero.getText().toString());
        int res=1;
        //calculamos el factorial y lo mostramos en el TextView
        for(int i=1;i<=num;i++){
            res*=i;
        }
        tvResultado.setText(String.valueOf(res));
    }
}
```

Telefonica

EDUCACIÓN DIGITAL