

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

U.T. 3. INTERFAZ DE USUARIO

PRÁCTICA 3.1, 3.2, 3.3 y 3.4:

ANDROID STUDIO

AUTOR: Vincenzo Rojas Carrera

CICLO FORMATIVO DE GRADO SUPERIOR DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA

[3.1 - Interfaces de usuario. ConstraintLayout 3](#_cbvgc1ke0vv4)

[Nuevo proyecto 3](#_vujfrmqn64ag)

[Layout: usar ConstraintLayout y desactivar Autoconnect 4](#_3l1gbs5wsv42)

[Borra el TextView “Hello World” 4](#_gkrnoih3uw4r)

[Insertar un ImageView y configurarlo 4](#_zg6m5eg7qpm4)

[Añadir los dos botones (ocultar / ver) 6](#_fml5jex06num)

[Acceder a los controles desde Java y programar los listeners 8](#_tz7kgwalopj)

[3.2 - Interfaces de usuario. ConstraintLayout 10](#_q90c2cqafwm6)

[Crea un nuevo proyecto 10](#_3cery5pbzxl0)

[Abrir y modificar el TextView principal 10](#_nqloof8va67o)

[Crear recurso de tipo cadena 10](#_nvo2odckqp56)

[Fondo temporal para el TextView 11](#_guym9m1173mv)

[Ajustar tamaño y estilo del texto 11](#_usndtop2ty1s)

[Añadir un segundo TextView 12](#_zfbg38vopfrz)

[Ejecutar en AVD 12](#_rsfhbasoi6ld)

[Modificar desde Java 13](#_hnln7wxjxnwk)

[Ejecutar y probar 13](#_nsgia8svrdn)

[3.3 - Interfaces de usuario. ConstraintLayout 14](#_degy4v9ftoa3)

[Crear un nuevo proyecto 14](#_vg1lo8voe0m6)

[Abrir y modificar activity\_main.xml 14](#_qitgy7srehp)

[Definir cadenas en strings.xml 15](#_ff22xvxugna2)

[Visualización en Design y AVD 15](#_g5fvq6ylz4n1)

[Observar estructura en árbol 15](#_i1dpyxd1n1i9)

[Probar layout\_width y layout\_height 16](#_ocfrutwla34v)

[Probar orientación del LinearLayout 17](#_8xn4qby5w3rs)

[Probar padding y margin 18](#_6u475dof37zn)

[Probar gravity y layout\_gravity 18](#_rjtjw1xqdjnl)

[Combinar orientación horizontal y wrap\_content 19](#_2ugqzbdrem69)

[Usar layout\_weight para repartir espacio 19](#_ntoh2i895rjq)

[3.4 - Interfaces de usuario. ConstraintLayout 21](#_kgei9geniq0c)

[Crear un nuevo proyecto 21](#_z548ljd4grtf)

[Diseñar el layout activity\_main.xml 21](#_mumr8mt54e6n)

[Definir los textos en res/values/strings.xml 22](#_ao61gb5py010)

[Programar la lógica en MainActivity.java 22](#_rrpgfjhnu38m)

[Explicación rápida 24](#_7hn3aqm43j44)

[Prueba del programa 24](#_8gk5maoc47zs)

[Eliminar el botón y detectar la tecla ENTER 25](#_vpxhxe4r2nko)

[Contar los intentos 26](#_nwpq1xhlk30y)

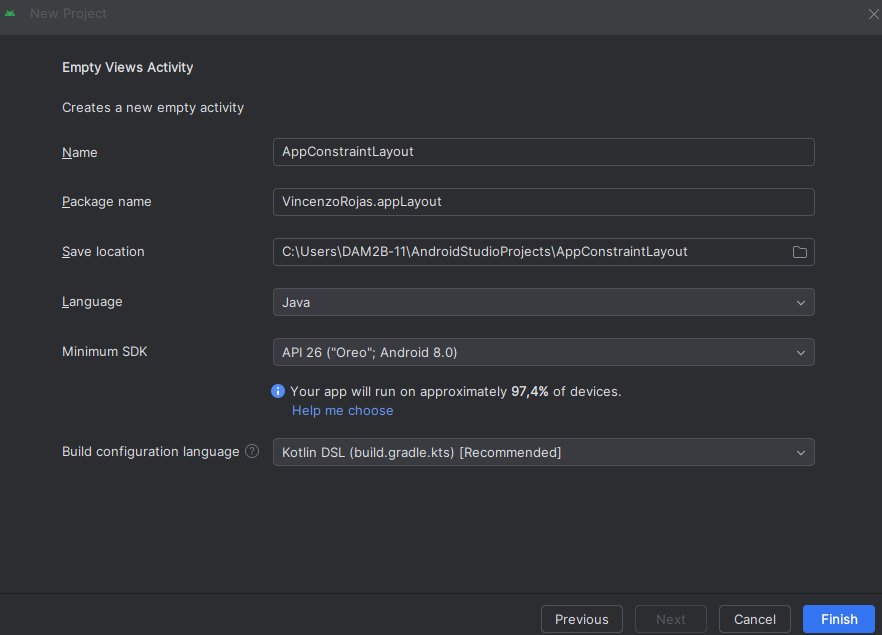
[Gestionar el final de la partida 27](#_lc533p9fsedm)

## 

# 3.1 - Interfaces de usuario. ConstraintLayout

## Nuevo proyecto

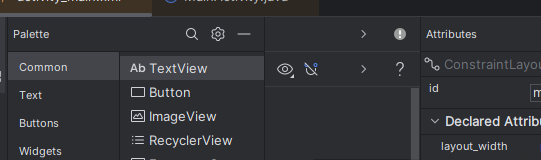
1. Abre **Android Studio** → **New Project**.
2. Plantilla: **Empty View Activity**.
3. Application name: AppConstraintLayout
4. Package name: apel1apel2.nombre.appLayout (sustituye apel1 y apel2 por tus apellidos).
5. Language: **Java**.
6. Minimum SDK: deja la recomendada por Android Studio (por compatibilidad).



## 

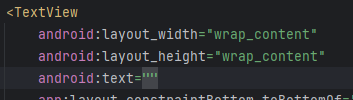
## Layout: usar ConstraintLayout y desactivar Autoconnect

1. Abre app/src/main/res/layout/activity\_main.xml en el **Design Editor**.
2. Asegúrate de que el contenedor raíz sea ConstraintLayout (si no, cámbialo).
3. En la barra superior del editor de ConstraintLayout, localiza el icono **Autoconnect** (imán) y **desactívalo** (segundo icono). Esto evita que Android Studio auto-conecte constraints. (el enunciado lo pide expresamente).



## Borra el TextView “Hello World”

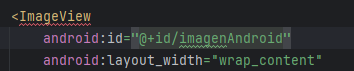
1. En la vista de diseño o en el XML, selecciona y elimina el TextView que viene por defecto.

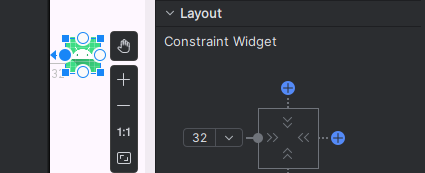


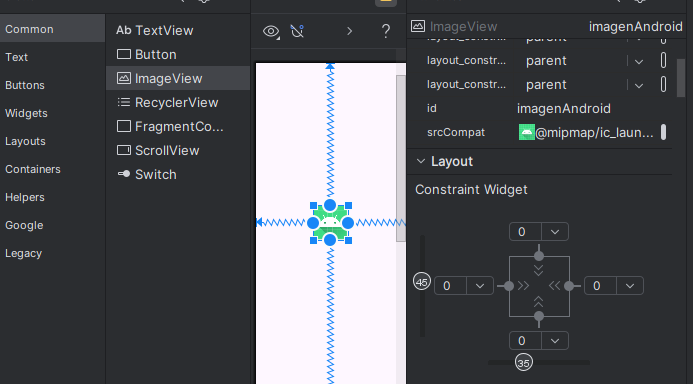
## Insertar un ImageView y configurarlo

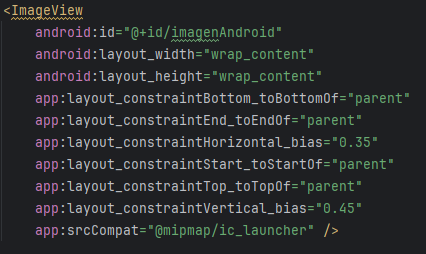
1. En la paleta, arrastra un **ImageView** al área de diseño.
2. Al añadirlo, en la ventana emergente elige el recurso (Mipmap o Drawable). Puedes añadir la imagen a app/src/main/res/mipmap-... o drawable por **Right-click → New → Image Asset** o copiar una imagen en res/drawable.
3. Selecciona el ImageView y **cámbiale el id** (en la vista XML o en Attributes) a:

android:id="@+id/imagenAndroid"



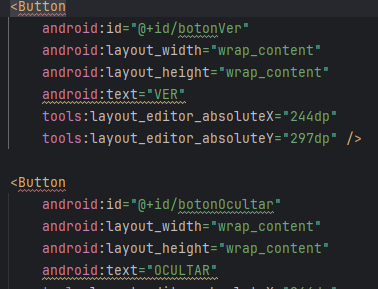
1. Crea constraints manuales:  
   * Ancla el punto de anclaje izquierdo del ImageView al borde izquierdo del ConstraintLayout y fija **margin-left = 32dp**.
   * Ancla también right y bottom (según el enunciado pide anclar izquierda, derecha e inferior).
   * Ajusta el **Horizontal Bias** y **Vertical Bias** si quieres centrar levemente (ej.: 45% vertical).

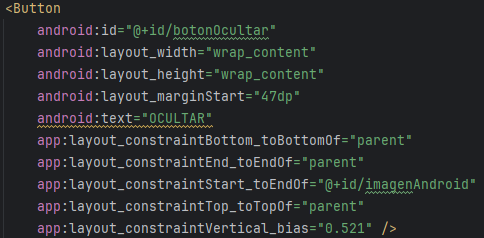


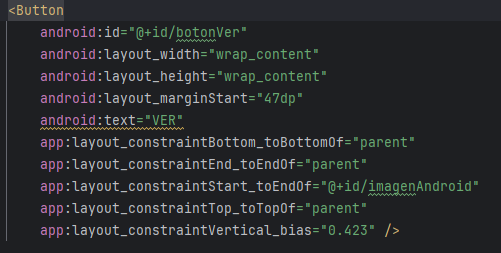
* + Observa en la vista XML las constraints generadas.

## Añadir los dos botones (ocultar / ver)

1. Arrastra dos Button desde la paleta y colócalos **a la derecha** del ImageView.
2. Asigna:
   * @+id/botonVer
   * @+id/botonOcultar
   * Texto de los botones: Ver y Ocultar



1. Crea constraints para que ambos botones queden siempre a la derecha de la imagen (por ejemplo: Start de los botones al End de la imagen con margin = 32dp; y alínealos verticalmente).



**Texto de los botones OCULTAR y VER**

En el panel lateral **Project**, asegúrate de estar en vista **Android**. Navega a:

app → res → values → strings.xml

Dentro del archivo encontrarás algo así:

<resources>

<string name="app\_name">AppConstraintLayout</string>

</resources>

Debajo de esa línea, **añade las dos nuevas cadenas**:

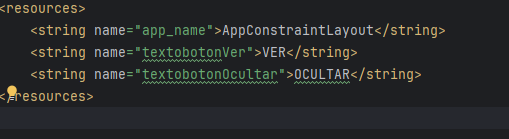
<resources>

<string name="app\_name">AppConstraintLayout</string>

<string name="textobotonVer">VER</string>

<string name="textobotonOcultar">OCULTAR</string>

</resources>



Abre el layout: app → res → layout → activity\_main.xml

1. Localiza las líneas donde definiste los botones: android:text="Ver" y android:text="Ocultar"
2. Sustituir por las referencias a las cadenas:  
    android:text="@string/textobotonVer"  
    android:text="@string/textobotonOcultar"
3. Cuando empieces a escribir @string/ puedes pulsar **Ctrl + Espacio** para que Android Studio te muestre una lista de todas las cadenas disponibles y autocompletar.

## Acceder a los controles desde Java y programar los listeners

Abre MainActivity.java y reemplaza/añade el código siguiente:

package VincenzoRojas.appLayout;

import android.os.Bundle;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

import android.view.View;

import android.widget.Button;

import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

private ImageView imagenAndroid;

private Button botonOcultar, botonVer;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main); // asegúrate del nombre

// Obtener referencias

imagenAndroid = (ImageView) findViewById(R.id.imagenAndroid);

botonOcultar = (Button) findViewById(R.id.botonOcultar);

botonVer = (Button) findViewById(R.id.botonVer);

// Listener para ocultar

botonOcultar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

ocultarImagen();

}

});

// Listener para ver (mostrar)

botonVer.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

verImagen();

}

});

}

private void ocultarImagen() {

imagenAndroid.setVisibility(View.INVISIBLE); // o View.GONE si quieres que libere espacio

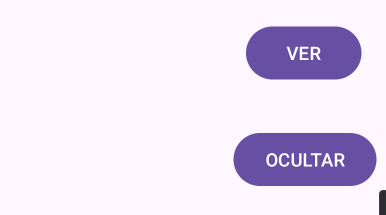
}

private void verImagen() {

imagenAndroid.setVisibility(View.VISIBLE);

}

}



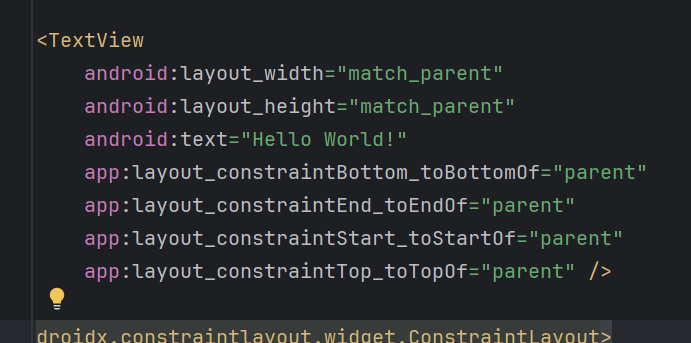
# 3.2 - Interfaces de usuario. ConstraintLayout

## Crea un nuevo proyecto

* **Application name:** AppEtiquetas
* **Package Name:** apel1.nombre.appEtiquetas
* **Min SDK:** Android 7.0 - API 24
* **Tipo de Activity:** Empty View Activity

## Abrir y modificar el TextView principal

* Abre res/layout/activity\_main.xml.
* Localiza el TextView con el texto Hello World.
* Cambia **layout\_width y layout\_height** a match\_parent usando autocompletado (Ctrl + Espacio):

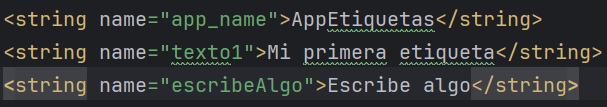


## Crear recurso de tipo cadena

* Abre res/values/strings.xml y añade:

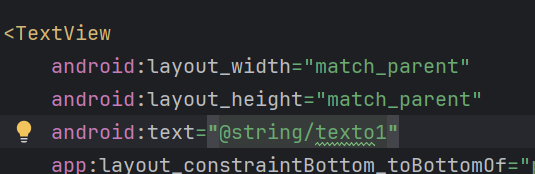
<string name="texto1">Mi primera etiqueta</string>

<string name="escribeAlgo">Escribe algo</string>



* Vuelve a activity\_main.xml y reemplaza el texto del TextView por:

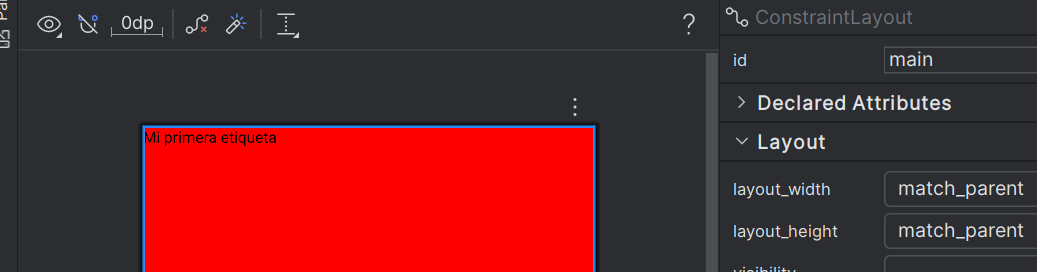
android:text="@string/texto1"



## Fondo temporal para el TextView

* Añade un color de fondo temporal para verlo mejor y cambia su **id** a:

android:background="#FF0000"  
android:id="@+id/etiqueta1"

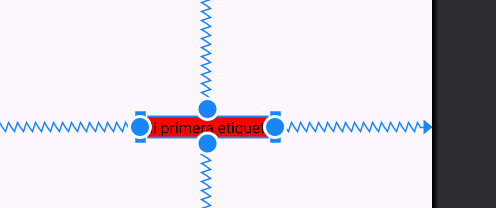


## Ajustar tamaño y estilo del texto

* Modifica ancho y alto a wrap\_content:

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

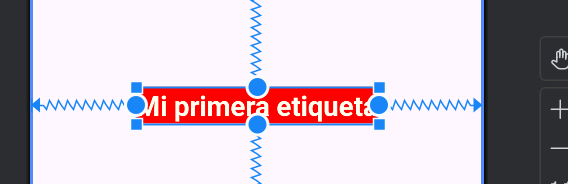


* Añade propiedades de estilo:

android:textSize="25sp"

android:textColor="#FFFFF0"

android:textStyle="bold"



## Añadir un segundo TextView

* En la paleta de **Design**, arrastra un TextView debajo del primero.
* Centra horizontalmente y ajusta constraints para que quede debajo del primer TextView.



## Ejecutar en AVD

* Abre un **AVD** (emulador) y ejecuta la aplicación para ver los cambios.

Añadir un EditText

* En activity\_main.xml añade:

<EditText

android:id="@+id/editText"

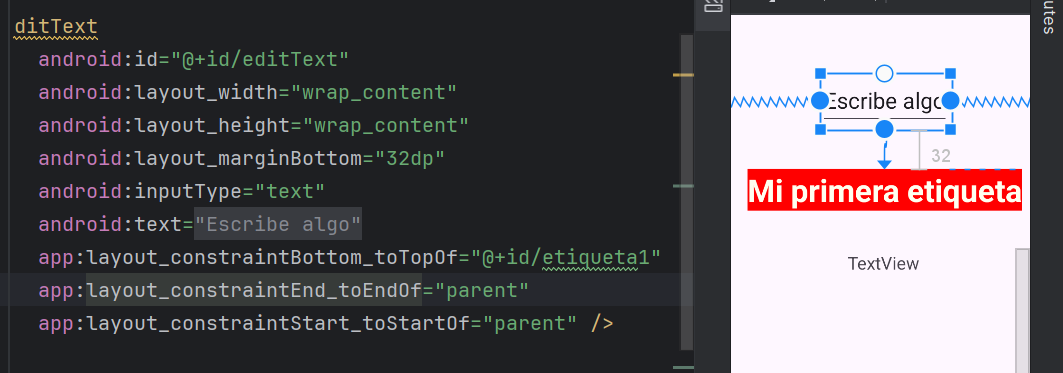
android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:inputType="text"

android:text="@string/escribeAlgo" />

* Esto crea un campo para introducir texto.
* Centra horizontalmente y coloca encima del TextView etiqueta1.



## Modificar desde Java

* En MainActivity.java, dentro de **onCreate()**, añade:

TextView etiqueta1 = findViewById(R.id.etiqueta1);

EditText cajaTexto = findViewById(R.id.cajaTexto);

cajaTexto.setOnKeyListener((v, keyCode, event) -> {

if (keyCode == KeyEvent.KEYCODE\_ENTER && event.getAction() == KeyEvent.ACTION\_UP) {

cambiarTexto(etiqueta1, cajaTexto.getText().toString());

return true;

}

return false;

});

* Crea el método **cambiarTexto()** fuera de onCreate():

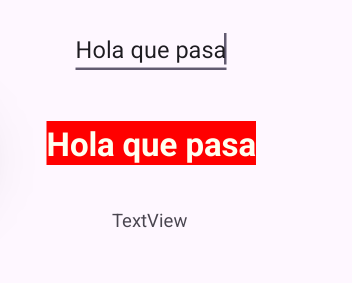
public void cambiarTexto(TextView etiqueta, String texto) {

etiqueta.setText(texto);

}

## Ejecutar y probar

* Ejecuta la app en el emulador.
* Escribe algo en el EditText y pulsa Enter.
* El texto aparecerá en el TextView etiqueta1.

# 

# 3.3 - Interfaces de usuario. ConstraintLayout

## Crear un nuevo proyecto

* **Application name:** AppLinearLayout
* **Package Name:** VincenzoRojas.appLinearLayout
* **Min SDK:** el recomendado por Android Studio
* **Tipo de Activity:** Empty View Activity

## Abrir y modificar activity\_main.xml

Sustituye todo el contenido del archivo por el siguiente código:

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/main"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#E0E0E0"

android:orientation="vertical"

tools:context=".MainActivity">

<TextView

android:id="@+id/text1"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="#80FF80"

android:text="@string/etiqueta1" />

<TextView

android:id="@+id/text2"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="#FF8080"

android:text="@string/etiqueta2" />

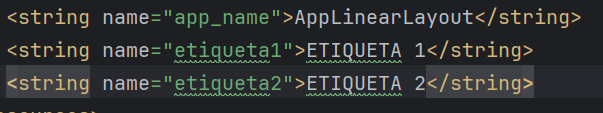
</LinearLayout>

## Definir cadenas en strings.xml

Abre res/values/strings.xml y añade:

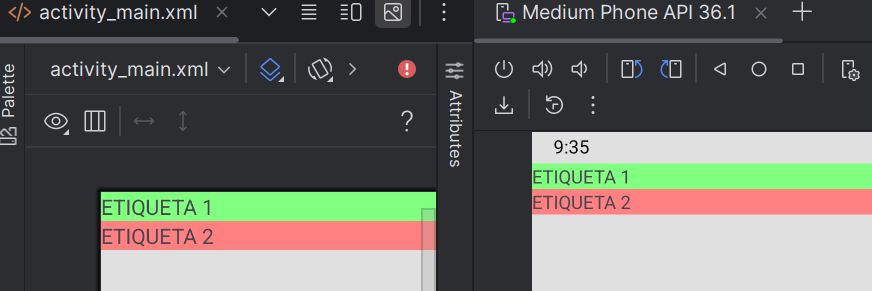
<string name="etiqueta1">ETIQUETA 1</string>

<string name="etiqueta2">ETIQUETA 2</string>



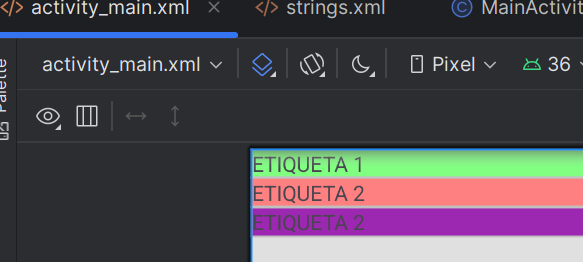
## Visualización en Design y AVD

Ve a la vista **Design** para comprobar cómo se muestran las etiquetas. Ejecuta la app en un **AVD** para verlo también.



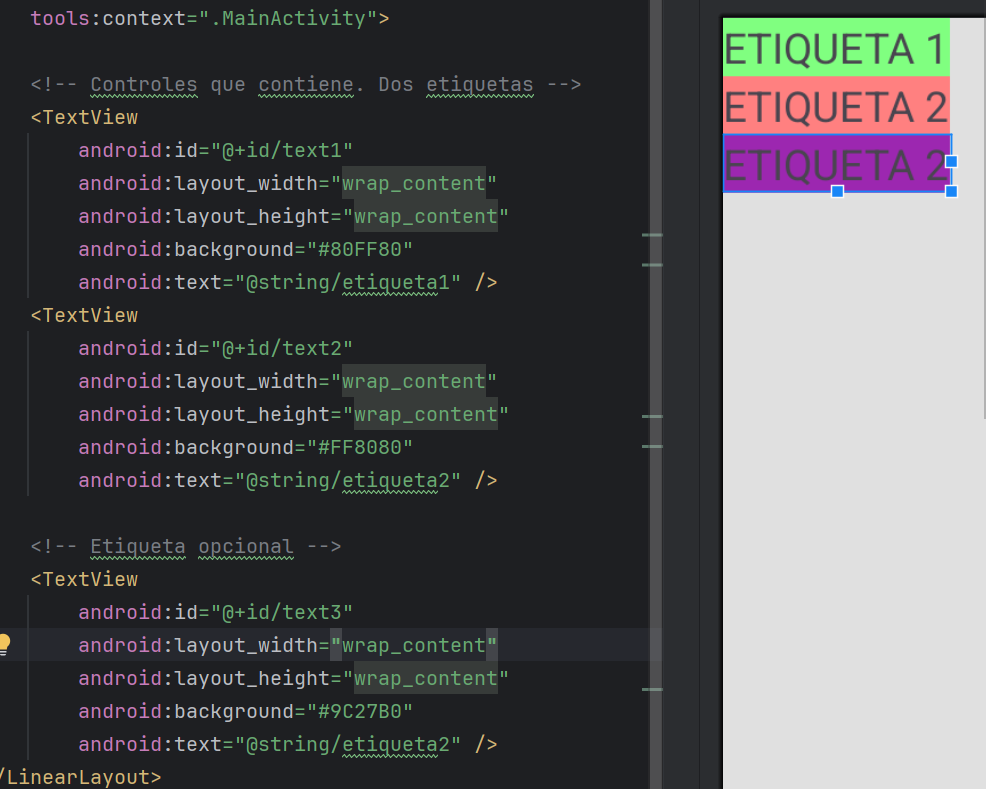
## Observar estructura en árbol

En **Component Tree** verás que dentro del LinearLayout están los dos TextView. Puedes incluir otro LinearLayout dentro del primero si quieres.

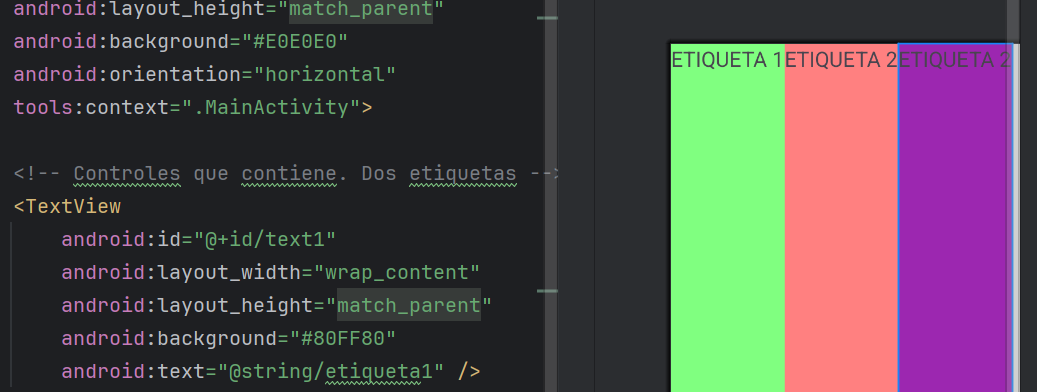


## Probar layout\_width y layout\_height

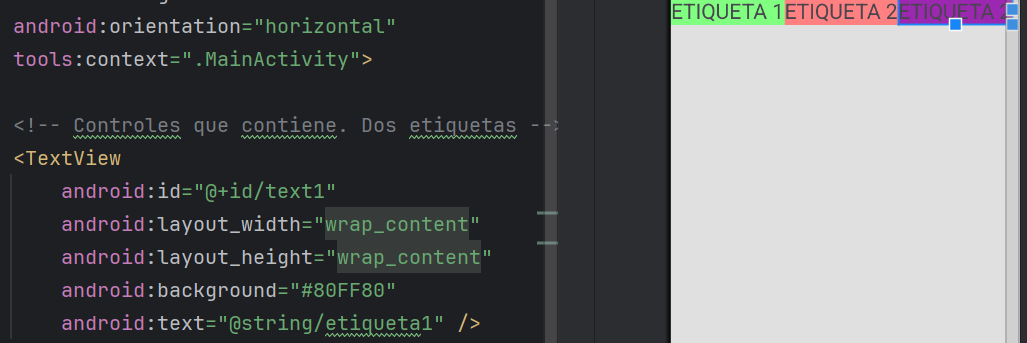
Selecciona etiqueta1 en **Component Tree** y cambia layout\_width y layout\_height a wrap\_content. Haz lo mismo con etiqueta2. Observa cómo ahora ocupan solo el espacio necesario para el contenido.



## Probar orientación del LinearLayout

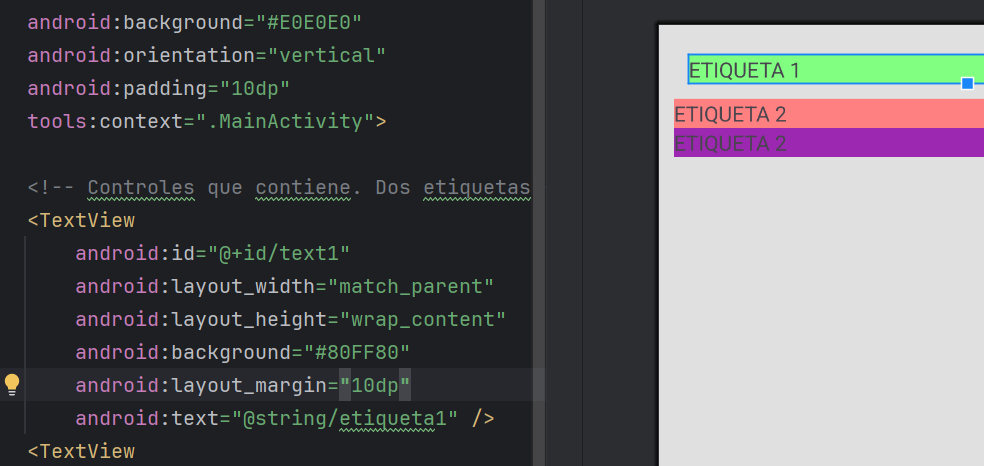
Selecciona el LinearLayout y cambia **orientation** a horizontal. Observa cómo se colocan las etiquetas en línea.

Cambia layout\_width de etiqueta1 a wrap\_content. Observa el efecto.

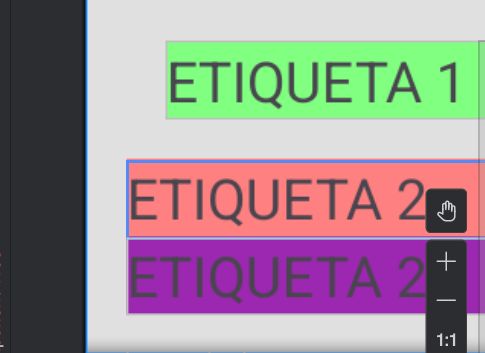


Restaura layout\_width de etiqueta1 a match\_parent. Vuelve a poner la orientación en vertical. Comprueba el comportamiento.

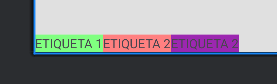
## Probar padding y margin

Añade padding al LinearLayout: android:padding="10dp". Observa el efecto interior. Añade margin a etiqueta1: android:layout\_margin="10dp". Observa la diferencia: margin afecta al espacio exterior del control.

## Probar gravity y layout\_gravity

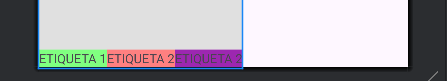
Modifica gravity del LinearLayout a bottom (android:gravity="bottom") y observa cómo se colocan las etiquetas dentro del layout. Cambia layout\_margin del LinearLayout a 10dp y observa el efecto.

## Combinar orientación horizontal y wrap\_content

Pon LinearLayout en horizontal y quita el margen. Cambia layout\_width de etiqueta1 y etiqueta2 a wrap\_content. Ahora ocupan solo el espacio necesario y están juntos.

## Usar layout\_weight para repartir espacio

Añade android:layout\_weight="1" en cada TextView y observa cómo se reparte el espacio sobrante.

Cambia layout\_width del LinearLayout a wrap\_content y comprueba el efecto.

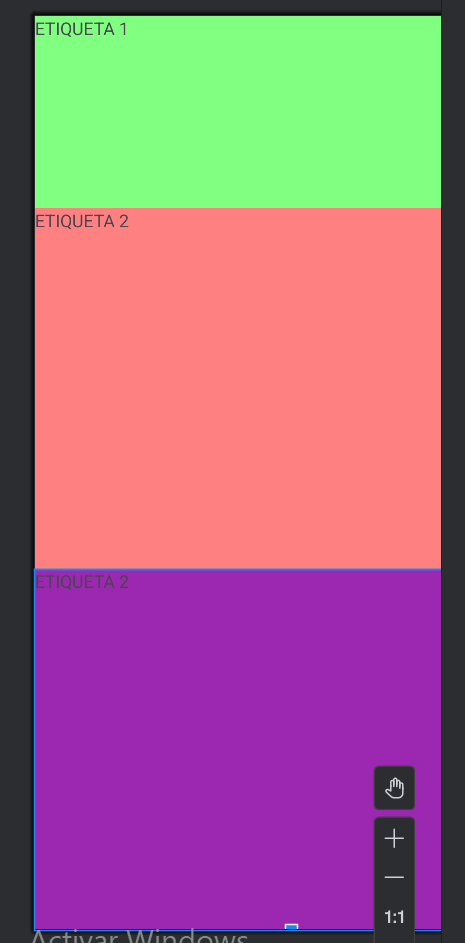
Vuelve a orientación vertical.



Cambia layout\_weight de etiqueta1 a 0.5. Observa cómo se reparte el alto del layout entre las etiquetas según el peso.



Vuelve a poner layout\_width del LinearLayout a match\_parent y de las etiquetas a match\_parent. Comprueba cómo se distribuye el espacio ahora.



# 3.4 - Interfaces de usuario. ConstraintLayout

## Crear un nuevo proyecto

* **Application name:** AppAdivina
* **Package Name:** VincenzoRojas.appAdivina
* **Min SDK:** la recomendada por Android Studio
* **Tipo de Activity:** Empty View Activity

## Diseñar el layout activity\_main.xml

Vamos a crear un diseño vertical con un texto superior, una zona inferior con un EditText, un TextView con el mensaje “¡Inténtalo!” y un Button para probar.

<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"

android:padding="16dp"

tools:context=".MainActivity">

<TextView

android:id="@+id/tInicio"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@string/textoInicio"

android:textSize="20sp"

android:gravity="center"

android:paddingBottom="40dp"/>

<EditText

android:id="@+id/editText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:inputType="number"

android:hint="@string/textoIntroduceNumero"/>

<TextView

android:id="@+id/tIntenta"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@string/textoIntentalo"

android:gravity="center"

android:paddingTop="10dp"/>

<Button

android:id="@+id/bProbar"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/textoBoton"

android:onClick="comprobarNumero"

android:layout\_gravity="center"

android:paddingTop="20dp"/>

</LinearLayout>

## Definir los textos en res/values/strings.xml

<string name="app\_name">AppAdivina</string>

<string name="textoInicio">He pensado un número entre 1 y 100. ¿Adivina cuál es?</string>

<string name="textoIntroduceNumero">Introduce un número</string>

<string name="textoIntentalo">¡Inténtalo!</string>

<string name="textoBoton">Probar</string>

<string name="textoMayor">¿%d? ¡Uy! El número que he pensado es mayor.</string>

<string name="textoMenor">¿%d? ¡Uy! El número que he pensado es menor.</string>

<string name="textoAcierto">¡¡Has acertado!!</string>

## Programar la lógica en MainActivity.java

Abre el archivo MainActivity.java y añade el siguiente código:

package ApelApel2.nombre.appAdivina;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import java.util.Random;

public class MainActivity extends AppCompatActivity {

TextView tInicio, tIntenta;

EditText et;

Button bProbar;

int numeroOculto;

int numeroJugado;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

tInicio = findViewById(R.id.tInicio);

tIntenta = findViewById(R.id.tIntenta);

et = findViewById(R.id.editText);

bProbar = findViewById(R.id.bProbar);

// Generar número aleatorio entre 1 y 100

// Para pruebas, puedes poner numeroOculto = 50;

Random dado = new Random();

numeroOculto = dado.nextInt(100) + 1;

}

public void comprobarNumero(View v) {

numeroJugado = Integer.parseInt(et.getText().toString());

if (numeroJugado < numeroOculto) {

String mensaje = getResources().getString(R.string.textoMayor);

String mensajeFinal = String.format(mensaje, numeroJugado);

tInicio.setText(mensajeFinal);

} else if (numeroJugado > numeroOculto) {

String mensaje = getResources().getString(R.string.textoMenor);

String mensajeFinal = String.format(mensaje, numeroJugado);

tInicio.setText(mensajeFinal);

} else {

tInicio.setText(R.string.textoAcierto);

}

}

}

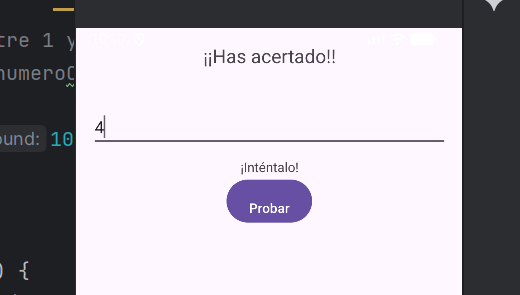
## 

## Explicación rápida

* El TextView superior muestra el mensaje inicial.
* El usuario escribe un número en el EditText.
* Al pulsar el botón **Probar**, se ejecuta comprobarNumero().
* Se compara el número introducido con el número aleatorio.
* El TextView superior se actualiza con el mensaje correspondiente.

## Prueba del programa

* Ejecuta la aplicación en un emulador o dispositivo físico.
* Introduce números hasta adivinar el correcto.
* Cuando aciertes, aparecerá “¡¡Has acertado!!”.



## 

## Eliminar el botón y detectar la tecla ENTER

En lugar de usar el botón “Probar”, capturaremos cuando el usuario pulse **Enter** en el campo de texto.

Borra por completo estas dos líneas: *Button bProbar;*

y dentro de onCreate(): *bProbar = findViewById(R.id.bProbar);*

También puedes quitar el <Button> del archivo XML (activity\_main.xml), ya no se usará.

Agrega este import arriba (es necesario para poder detectar la tecla Enter):

import android.view.KeyEvent;

Después de: et = findViewById([R.id](http://r.id/).editText); Se añade este bloque:

et.setOnKeyListener(new View.OnKeyListener() {

@Override

public boolean onKey(View v, int keyCode, KeyEvent event) {

if ((event.getAction() == KeyEvent.ACTION\_DOWN) &&

(keyCode == KeyEvent.KEYCODE\_ENTER)) {

comprobarNumero(); // Llamamos al método sin el parámetro View

return true;

}

return false;

}

});

Ya no recibe un parámetro View, porque no lo llama un botón. Cámbialo a:

public void comprobarNumero()

### 

## 

## Contar los intentos

Declara al principio de la clase:

int numIntentos = 0;

TextView tvIntentos;

En tu onCreate() vincúlalo: *tvIntentos = findViewById(R.id.tvIntentos);*

En tu método comprobarNumero(), justo antes de comprobar si es mayor/menor, añade, porque cada vez que el jugador pruebe un número, **incrementa** y **actualiza** el texto:

numIntentos++;

actualizarIntentos();

Crea el método:

private void actualizarIntentos() {

String intentosTexto = getResources().getQuantityString(

R.plurals.intentos, numIntentos, numIntentos);

tvIntentos.setText(intentosTexto);

}

En tu strings.xml crea un recurso *plurals*:

<plurals name="intentos">

<item quantity="one">Llevas %d intento</item>

<item quantity="other">Llevas %d intentos</item>

</plurals>

Y en activity\_main.xml, añade:

<TextView

android:id="@+id/tvIntentos"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Llevas 0 intentos"

android:textSize="16sp"

android:layout\_marginTop="16dp"/>

## 

## Gestionar el final de la partida

1️⃣ Añadir un botón para “Nueva Partida”

En tu activity\_main.xml, añade un Button que empezará **oculto**:

<Button

android:id="@+id/btnNuevaPartida"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Nueva Partida"

android:visibility="gone"/>

Esto permitirá reiniciar el juego.

2️⃣ Declarar el botón en la clase

Button btnNuevaPartida;

Y en onCreate() vincúlalo:

btnNuevaPartida = findViewById(R.id.btnNuevaPartida);

btnNuevaPartida.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

nuevaPartida();

}

});

3️⃣ Crear el método finalPartida()

Dentro de la clase MainActivity:

private void finalPartida() {

et.setVisibility(View.GONE);

tvIntentos.setVisibility(View.GONE);

btnNuevaPartida.setVisibility(View.VISIBLE);

}

4️⃣ Modificar comprobarNumero()

Cuando el usuario acierte, cambia a:

else {

tInicio.setText(getString(R.string.hasAcertado));

finalPartida(); }

5️⃣ Crear método para iniciar nueva partida

private void nuevaPartida() {

numeroOculto = new Random().nextInt(100) + 1;

numIntentos = 0;

actualizarIntentos();

tInicio.setText(R.string.textoInicio); // un mensaje de bienvenida, opcional

et.setText("");

et.setVisibility(View.VISIBLE);

tvIntentos.setVisibility(View.VISIBLE);

btnNuevaPartida.setVisibility(View.GONE);

}

Esto reinicia el juego como al principio.

