



El futuro digital  
es de todos

MinTIC

# Unidad 3

## Aplicaciones móviles





El futuro digital  
es de todos

MinTIC

# Tema 4

## RelativeLayout





# RelativeLayout

Es un grupo de vistas que muestra vistas secundarias en posiciones relativas. La posición de cada vista puede especificarse como relativa a elementos del mismo nivel (como a la izquierda de otra vista o por debajo de ella) o en posiciones relativas al área RelativeLayout superior (como alineada a la parte inferior, izquierda o central).





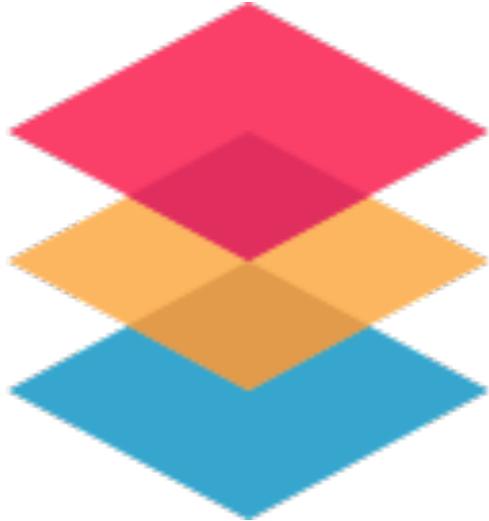
# RelativeLayout

RelativeLayout es una utilidad muy eficaz para diseñar una interfaz de usuario porque puede eliminar grupos de vistas anidados y conservar la estabilidad de la jerarquía de diseño, lo que mejora el rendimiento. Si se usan varios grupos de LinearLayout anidados, quizás se puedan reemplazar por un grupo RelativeLayout individual.





# Cómo posicionar vistas



RelativeLayout permite que vistas secundarias especifiquen su posición relativa a la vista superior o entre sí (especificada por ID). De esta manera, se puede alinear dos elementos por el borde derecho o hacer que uno esté por debajo del otro, en el centro de la pantalla, en el centro a la izquierda, y así sucesivamente. De manera predeterminada, todas las vistas secundarias se dibujan en la esquina superior izquierda del diseño, por lo que se debe definir la posición de cada vista utilizando las diversas propiedades de diseño disponibles en RelativeLayout.LayoutParams



# Propiedades de diseño

Estos son solo algunos ejemplos. Todos los atributos de diseño están documentados en `RelativeLayout.LayoutParams`.



`android:layout_alignParentTop`

Si el valor es "true", el borde superior de esta vista coincidirá con el del elemento superior.



`android:layout_centerVertical`

Si el valor es "true", centra este elemento secundario en posición vertical dentro de su elemento superior.



# Propiedades de diseño



android:layout\_below

Posiciona el borde superior de esta vista debajo de la vista especificada con un ID de recurso.



android:layout\_toRightOf

Posiciona el borde izquierdo de esta vista a la derecha de la vista especificada con un ID de recurso.



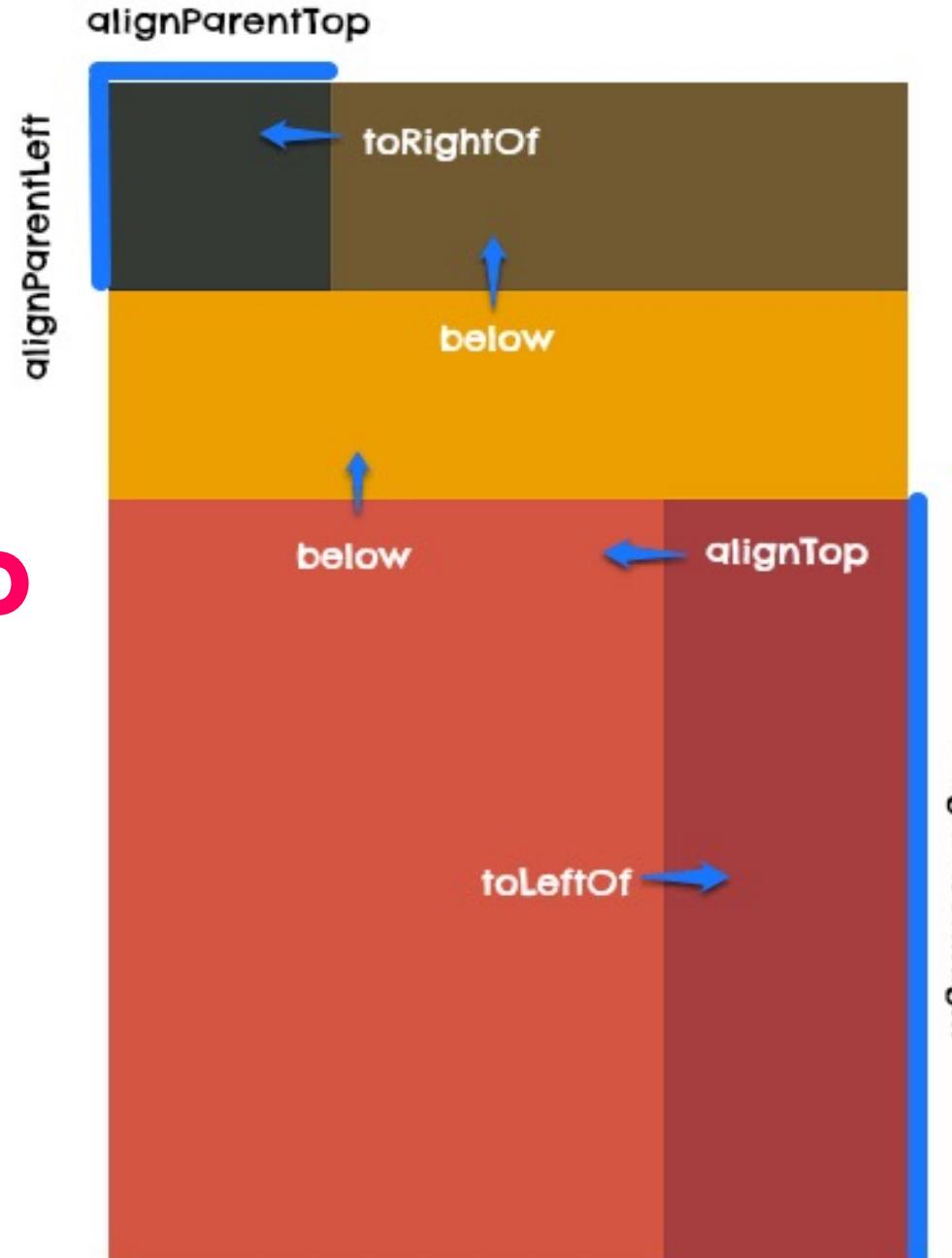
# Propiedades de diseño



El valor de cada propiedad de diseño es un valor booleano que habilita una posición de diseño relativa al elemento `RelativeLayout` superior o un ID que hace referencia a otra vista en el diseño en el que se debe posicionar la vista.

En un diseño XML, las dependencias frente a otras vistas en el diseño se pueden declarar en cualquier orden. Por ejemplo, se puede declarar que "view1" se posicione debajo de "view2", incluso si "view2" es la última vista declarada de la jerarquía. El siguiente ejemplo demuestra una situación de este tipo.

# Posicionando elementos



Con el `RelativeLayout` se puede pensar en cómo alinear los bordes de cada view con otros.

En una sentencia como «el botón estará por debajo del texto» o «la imagen se encuentra a la derecha de la descripción», en ninguno de los casos se refiere a una posición absoluta o un espacio determinado.

Simplemente se describe la ubicación y el framework de Android computará el resultado final.



# Ejemplo RelativeLayout

## Paso 1

Crea otro layout dentro de **res/layout** llamado **ejemplo\_relative\_layout.xml**.

Esta vez crearemos un pequeño formulario con cuatro campos de una persona. Se usará un edit text para los nombres y otro para los apellidos. Por debajo tendremos dos spinners que permitirán seleccionar el estado civil y el cargo actual. Todos van alineados dentro de un relative layout

Implementa la siguiente definición:



# Ejemplo RelativeLayout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="@dimen/activity_horizontal_margin">

    <EditText
        android:id="@+id/input_nombre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:ems="10"
        android:hint="Nombres"
        android:inputType="textPersonName" />
```



# Ejemplo RelativeLayout

```
<EditText  
    android:id="@+id/input_apellido"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentStart="true"  
    android:layout_below="@+id/input_nombre"  
    android:ems="10"  
    android:hint="Apellidos"  
    android:inputType="textPersonName" />  
<TextView  
    android:id="@+id/texto_estado_civil"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentStart="true"  
    android:layout_below="@+id/input_apellido"  
    android:layout_marginRight="48dp"  
    android:paddingBottom="8dp"  
    android:paddingTop="16dp"  
    android:text="Estado civil"  
    android:textAppearance="?android:attr/textAppearanceMedium" />
```



# Ejemplo RelativeLayout

```
<Spinner  
    android:id="@+id/spinner_estado_civil"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentLeft="true"  
    android:layout_below="@+id/texto_estado_civil"  
    android:layout_toLeftOf="@+id/spinner_cargo"  
    android:entries="@array/lista_estado_civil" />
```

```
<TextView  
    android:id="@+id/texto_cargo"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/input_apellido"  
    android:layout_centerHorizontal="true"  
    android:layout_toRightOf="@+id/texto_estado_civil"  
    android:paddingBottom="8dp"  
    android:paddingTop="16dp"  
    android:text="Cargo"  
    android:textAppearance="?android:attr/textAppearanceMedium" />
```



# Ejemplo RelativeLayout

```
<Spinner  
    android:id="@+id/spinner_cargo"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignLeft="@+id/texto_cargo"  
    android:layout_alignParentRight="true"  
    android:layout_alignStart="@+id/texto_cargo"  
    android:layout_below="@+id/texto_cargo"  
    android:entries="@array/lista_cargo" />
```

```
</RelativeLayout>
```



# Ejemplo RelativeLayout

## Paso 2

Cambia el parámetro de `setContentView()` por `R.layout.ejemplo_relative_layout`.

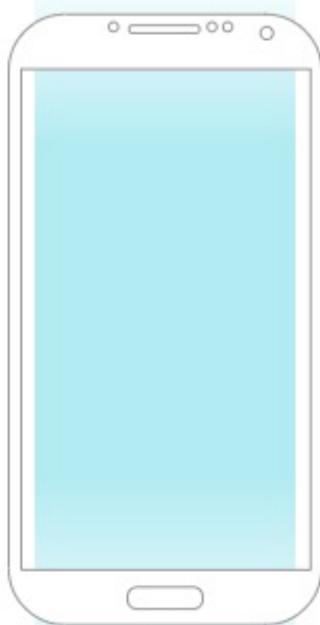
## Paso 3

Visualiza el resultado.





# El control ScrollView, barra de desplazamiento



El ScrollView en Android te permite albergar una jerarquía de views con el fin de desplazar su contenido a lo largo de la pantalla, cuando sus dimensiones exceden el tamaño de la misma.

El usuario hará scroll a través de un gesto de swipe vertical u horizontal para revelar la información limitada por el tamaño del contenedor.



# Ejemplo desplazamiento horizontal

```
<?xml version="1.0" encoding="utf-8"?>
<HorizontalScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/scroll"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:padding="16dp">
        <TextView
            android:id="@+id/piece1"
            android:layout_width="90dp"
            android:layout_height="90dp"
            android:background="@color/blue1"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
```



# Ejemplo desplazamiento horizontal

<TextView

```
    android:id="@+id/block2"  
    android:layout_width="180dp"  
    android:layout_height="0dp"  
    android:layout_marginStart="16dp"  
    android:layout_marginLeft="16dp"  
    android:background="@color/blue2"  
    app:layout_constraintBottom_toBottomOf="@+id/piece1"  
    app:layout_constraintStart_toEndOf="@+id/piece1"  
    app:layout_constraintTop_toTopOf="parent" />
```

<TextView

```
    android:id="@+id/block3"  
    android:layout_width="40dp"  
    android:layout_height="0dp"  
    android:layout_marginStart="16dp"  
    android:layout_marginLeft="16dp"  
    android:background="@color/blue3"  
    app:layout_constraintBottom_toBottomOf="@+id/piece1"  
    app:layout_constraintStart_toEndOf="@+id/block2"  
    app:layout_constraintTop_toTopOf="parent" />
```

# Ejemplo desplazamiento horizontal

<TextView

```
    android:id="@+id/block4"  
    android:layout_width="40dp"  
    android:layout_height="40dp"  
    android:layout_marginStart="16dp"  
    android:layout_marginLeft="16dp"  
    android:background="@color/orange1"  
    app:layout_constraintStart_toEndOf="@+id/block3"  
    app:layout_constraintTop_toTopOf="parent" />
```

<TextView

```
    android:id="@+id/block5"  
    android:layout_width="40dp"  
    android:layout_height="0dp"  
    android:layout_marginStart="16dp"  
    android:layout_marginLeft="16dp"  
    android:layout_marginTop="16dp"  
    android:background="@color/orange2"  
    app:layout_constraintBottom_toBottomOf="@+id/block3"  
    app:layout_constraintStart_toEndOf="@+id/block3"  
    app:layout_constraintTop_toBottomOf="@+id/block4" />
```





# Ejemplo desplazamiento horizontal

<TextView

```
    android:id="@+id/block6"
    android:layout_width="80dp"
    android:layout_height="0dp"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:background="@color/orange3"
    app:layout_constraintBottom_toBottomOf="@+id/block4"
    app:layout_constraintStart_toEndOf="@+id/block4"
    app:layout_constraintTop_toTopOf="@+id/block4"
    app:layout_constraintVertical_bias="0.0" />
```

<TextView

```
    android:id="@+id/block7"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="@color/orange4"
    app:layout_constraintBottom_toBottomOf="@+id/block5"
    app:layout_constraintEnd_toEndOf="@+id/block8"
    app:layout_constraintStart_toStartOf="@+id/block6"
    app:layout_constraintTop_toTopOf="@+id/block5" />
```



# Ejemplo desplazamiento horizontal

```
<TextView
```

```
    android:id="@+id/block8"  
    android:layout_width="80dp"  
    android:layout_height="40dp"  
    android:layout_marginStart="16dp"  
    android:layout_marginLeft="16dp"  
    android:layout_marginBottom="5dp"  
    android:background="@color/blue3"  
    app:layout_constraintBottom_toTopOf="@+id/block7"  
    app:layout_constraintStart_toEndOf="@+id/block6"  
    app:layout_constraintTop_toTopOf="parent"  
    app:layout_constraintVertical_bias="0.0" />
```

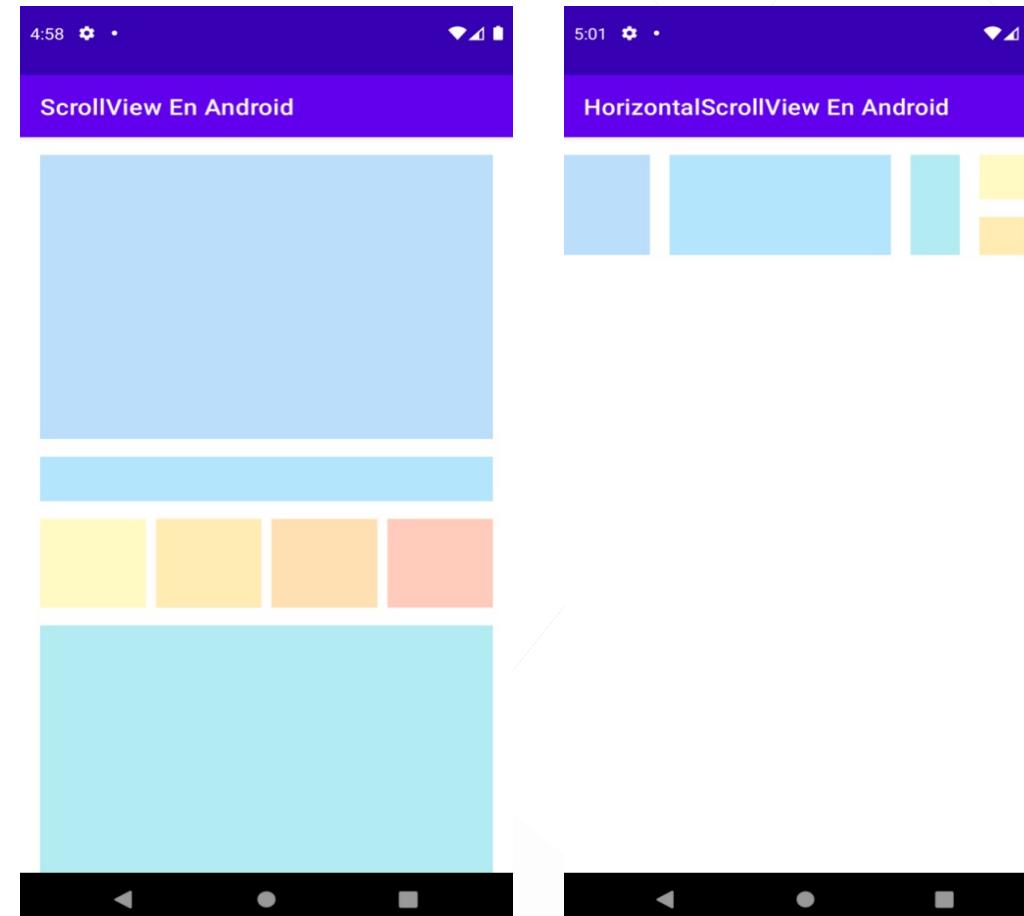
```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
</HorizontalScrollView>
```



# HorizontalScrollView

La siguiente imagen muestra la App de ilustración creada para comprender los conocimientos expuestos

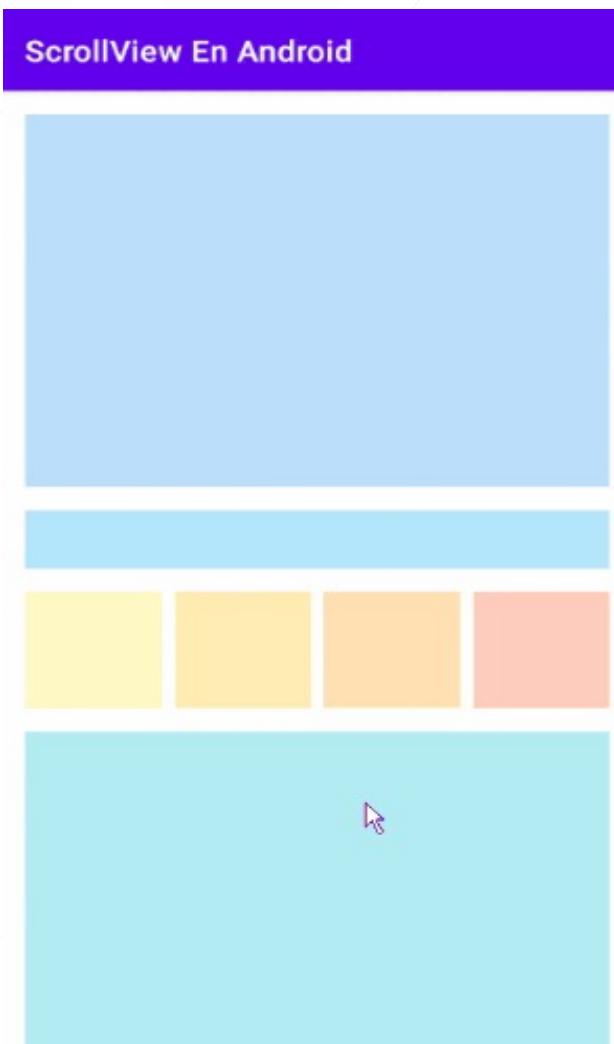




# Scrolling Vertical

Si la información que deseas proyectar verticalmente es más grande que la orientación de tu pantalla móvil, entonces envuelve el ViewGroup del contenido con un objeto de la clase ScrollView

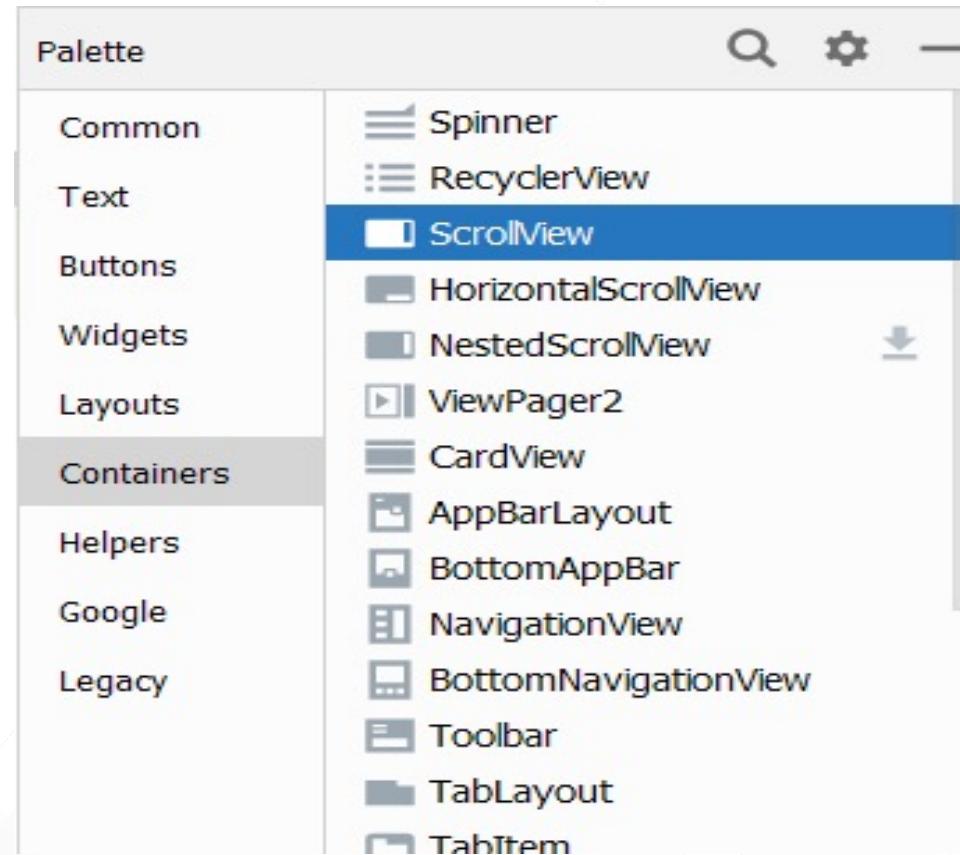
ScrollView soporta un solo hijo como contexto de desplazamiento, por lo que la estructura general de tu diseño debe encontrarse en él.





# Scrolling Vertical

Puedes añadir este elemento desde Android Studio yendo a **Palette** > **Containers** > **ScrollView**, o se puede recubrir el ViewGroup principal desde el layout con la etiqueta **<ScrollView>**





# Ejemplo desplazamiento vertical

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    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:id="@+id/scrollView"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_pare
        android:layout_height="wrap_content"
        android:padding="16dp"
        tools:context=".MainActivity">
```



# Ejemplo Desplazamiento vertical

<TextView

```
    android:id="@+id/child_scroll"  
    android:layout_width="0dp"  
    android:layout_height="256dp"  
    android:background="@color/blue2"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

<TextView

```
    android:id="@+id/headline"  
    android:layout_width="0dp"  
    android:layout_height="40dp"  
    android:layout_marginTop="16dp"  
    android:background="@color/blue3"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/child_scrol  
    />
```



# Ejemplo Desplazamiento vertical

```
<TextView  
    android:id="@+id/indicator_1"  
    android:layout_width="0dp"  
    android:layout_height="80dp"  
    android:layout_marginTop="16dp"  
    android:layout_marginEnd="8dp"  
    android:layout_marginRight="8dp"  
    android:background="@color/orange"  
    app:layout_constraintEnd_toStartOf="@+id/indicator_2"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/headline" />  
  
<TextView  
    android:id="@+id/indicator_2"  
    android:layout_width="0dp"  
    android:layout_height="80dp"  
    android:layout_marginTop="16dp"  
    android:layout_marginEnd="8dp"  
    android:layout_marginRight="8dp"  
    android:background="@color/orange2"  
    app:layout_constraintEnd_toStartOf="@+id/indicator_3"  
    app:layout_constraintHorizontal_bias="0.5"  
    app:layout_constraintStart_toEndOf="@+id/indicator_1"  
    app:layout_constraintTop_toBottomOf="@+id/headline" />
```



# Ejemplo Desplazamiento vertical

```
<TextView  
    android:id="@+id/indicator_3"  
    android:layout_width="0dp"  
    android:layout_height="80dp"  
    android:layout_marginTop="16dp"  
    android:layout_marginEnd="8dp"  
    android:layout_marginRight="8dp"  
    android:background="@color/orange3"  
    app:layout_constraintEnd_toStartOf="@+id/indicator_4"  
    app:layout_constraintHorizontal_bias="0.5"  
    app:layout_constraintStart_toEndOf="@+id/indicator_2"  
    app:layout_constraintTop_toBottomOf="@+id/headline" />  
  
<TextView  
    android:id="@+id/indicator_4"  
    android:layout_width="0dp"  
    android:layout_height="80dp"  
    android:layout_marginTop="16dp"  
    android:background="@color/orange4"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.5"  
    app:layout_constraintStart_toEndOf="@+id/indicator_3"  
    app:layout_constraintTop_toBottomOf="@+id/headline" />
```



# Ejemplo desplazamiento vertical

```
<TextView
```

```
    android:id="@+id/body_text"  
    android:layout_width="0dp"  
    android:layout_height="512dp"  
    android:layout_marginTop="16dp"  
    android:background="@color/blue4"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/indicator_1" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

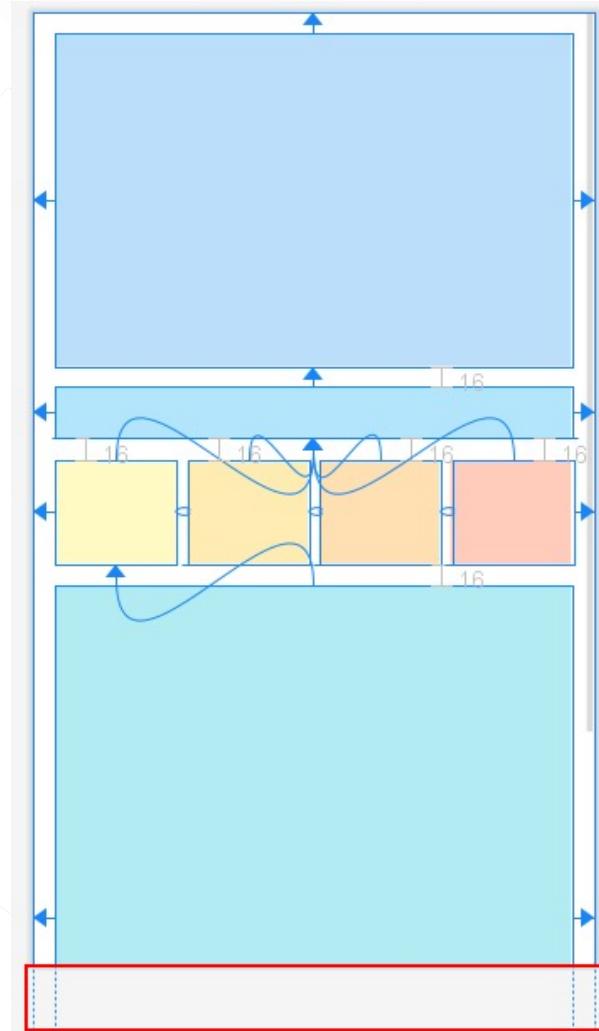
```
</ScrollView>
```



# Scrolling Vertical

El diseño anterior usa un tamaño de 512dp en el último elemento `body_text` con el fin de desbordar la capacidad de la pantalla. Al conectar elementos en el editor de layouts en Android Studio se verá una zona punteada consciente del exceso.

Es necesario que uses `wrap_content` en `android:layout_height` del contenedor directo para ajustar la zona al scrolling.





# Scrolling Horizontal

## HorizontalScrollView En Android



El mismo criterio aplica para los contenidos que abruman horizontalmente la pantalla del dispositivo. Usa la clase `HorizontalScrollView` para proveer desplazamiento entre los límites del contenedor.

Al igual que `ScrollView`, `HorizontalScrollView` hereda de la clase `FrameLayout`, por lo que la jerarquía a scrollear debe ser parte de un solo nodo.