
GALAXY
TRAINING

ANDROID DESDE 0

Sesión 2

Parte 2

Ing. Marco Estrella
Instructor en Tecnologías Java y Android
Github @jmarkstar

TEMAS A TRATAR

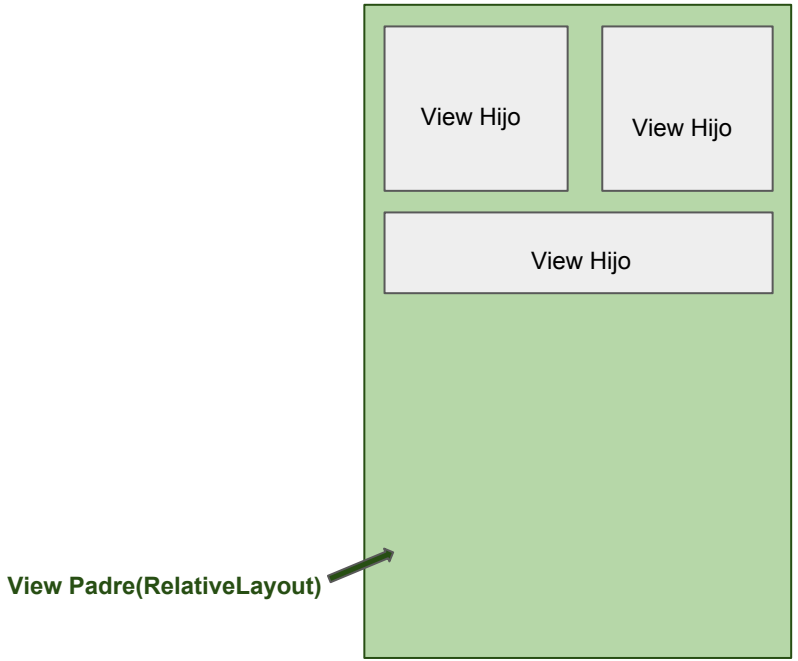
RelativeLayout

Más Componentes View

Recursos estáticos(iconos, imágenes, texto, color, enteros, etc)

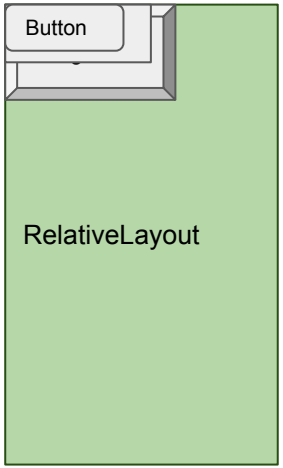
Introducción a Adapters

Es un ViewGroup donde podemos posicionar un componente View hijo en relación a otros Views Hijos o al View padre.



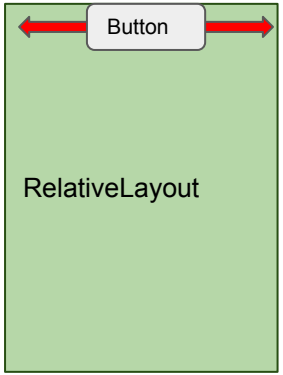
POSICIONAMIENTO CON RELACIÓN AL PADRE

Si creamos algunos Views dentro del RelativeLayout, estos se posicionarán a lado izquierdo superior de manera automática.

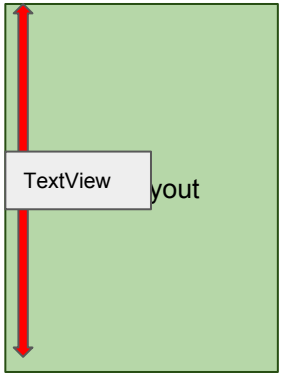


Algunos Atributos para posicionar los Views con relación al Padre:

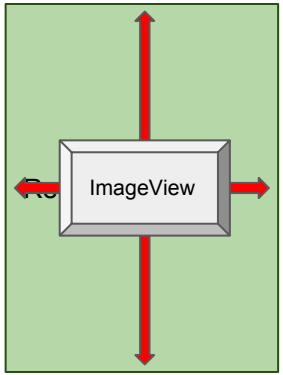
android:layout_centerHorizontal = “true”



android:layout_centerVertical = “true”



android:layout_centerInParent = “true”



Nota: Estos atributos son asignados al Hijo.

RELATIVELAYOUT - EJERCICIO

SOMOS PARTNER ORACLE

Usar RelativeLayout como Layout raíz de la pantalla.

```
<RelativeLayout
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- AGREGAR VIEW HIJOS AQUÍ -->

</RelativeLayout>
```

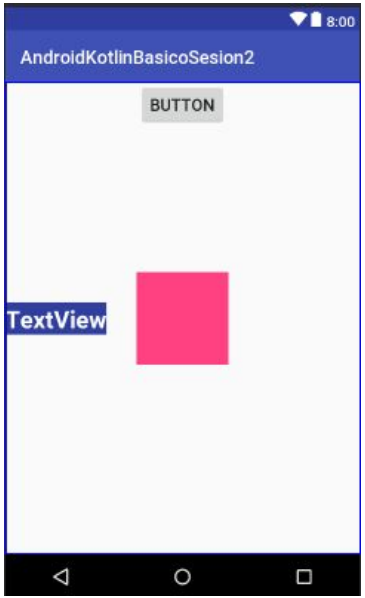
Agregar Hijos

```
<ImageView android:id="@+id/ivFoto"
    android:layout_centerInParent="true"
    android:background="@color/colorAccent"
    android:layout_width="100dp"
    android:layout_height="100dp" />
```

```
<TextView android:id="@+id/tvTexto"
    android:layout_centerVertical="true"
    android:text="TextView"
    android:textSize="26sp"
    android:textColor="#FFFFFF"
    android:background="@color/colorPrimaryDark"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

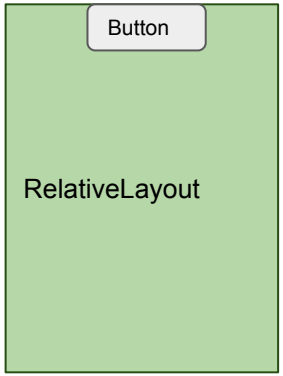
```
<Button android:id="@+id/btnClick"
    android:layout_centerHorizontal="true"
    android:text="Button"
    android:textSize="18sp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Resultado

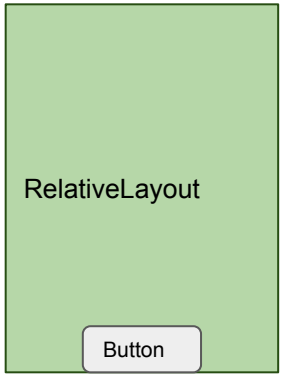


POSICIONAMIENTO CON RELACIÓN AL PADRE

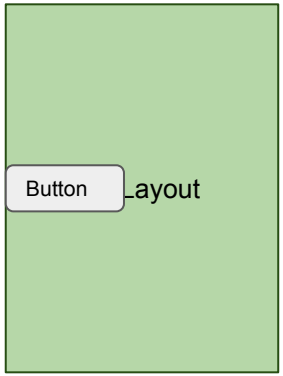
android:layout_alignParentTop = “true”



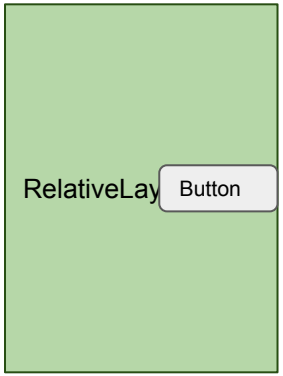
android:layout_alignParentBottom = “true”



android:layout_alignParentStart = “true”



android:layout_alignParentEnd = “true”



Nota: Para poder tener una mejor visualización del efecto de estos atributos, también se está usando el atributo **android:layout_centerInParent = “true”**.

POSICIONAMIENTO - EJERCICIO

SOMOS PARTNER ORACLE

Crear el RelativeLayout

```
<RelativeLayout
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- AGREGAR VIEW HIJOS AQUÍ -->

</RelativeLayout>
```

Agregar los View Hijos

```
<TextView
    android:layout_centerInParent="true"
    android:layout_alignParentTop="true"
    android:text="TextView 1"
    android:textSize="26sp"
    android:textColor="#FFFFFF"
    android:background="@color/colorPrimaryDark"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```
<TextView
    android:layout_centerInParent="true"
    android:layout_alignParentBottom="true"
    android:text="TextView 4"
    android:textSize="26sp"
    android:textColor="#FFFFFF"
    android:background="@color/colorPrimaryDark"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

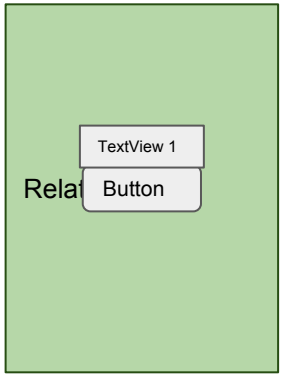
```
<TextView
    android:layout_centerInParent="true"
    android:layout_alignParentStart="true"
    android:text="TextView 3"
    android:textSize="26sp"
    android:textColor="#FFFFFF"
    android:background="@color/colorPrimaryDark"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```
<TextView
    android:layout_centerInParent="true"
    android:layout_alignParentEnd="true"
    android:text="TextView 2"
    android:textSize="26sp"
    android:textColor="#FFFFFF"
    android:background="@color/colorPrimaryDark"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

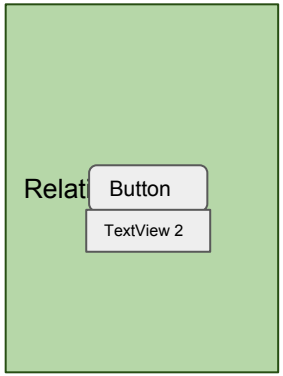
Resultado

POSICIONAMIENTO CON RELACIÓN A OTROS VIEWS

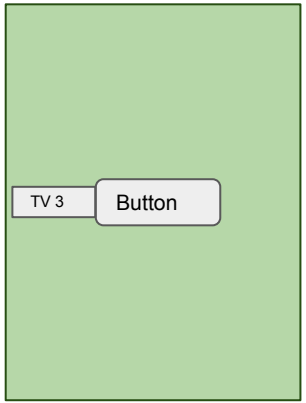
android:layout_above = "@+id/buttonId"



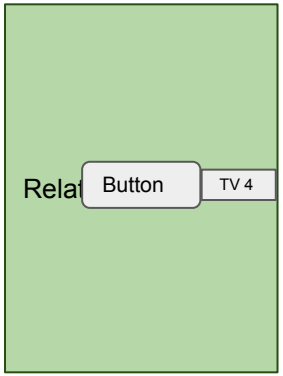
android:layout_below = "@+id/buttonId"



android:layout_toStartOf = "@+id/buttonId"



android:layout_toEndOf = "@+id/buttonId"



Nota: Para poder tener una mejor visualización del efecto de estos atributos, también se está usando el atributo **android:layout_centerInParent = "true"** en todos los Views..

POSICIONAMIENTO CON RELACIÓN A OTROS VIEWS

SOMOS PARTNER ORACLE

1- Crear el RelativeLayout

```
<RelativeLayout
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- AGREGAR VIEW HIJOS AQUÍ -->

</RelativeLayout>
```

2 - Agregar Botón

```
<Button android:id="@+id/button"
    android:text="Button"
    android:layout_centerInParent="true"
    android:textSize="18sp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

3 - Agregar los TextViews

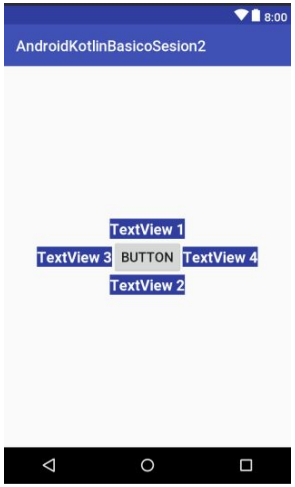
```
<TextView
    android:layout_centerInParent="true"
    android:layout_above="@+id/button"
    android:text="TextView 1"
    android:textSize="20sp"
    android:textColor="#FFFFFF"
    android:background="@color/colorPrimaryDark"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```
<TextView android:layout_toStartOf="@+id/button"
    android:layout_centerInParent="true"
    android:text="TextView 3"
    android:textSize="20sp"
    android:textColor="#FFFFFF"
    android:background="@color/colorPrimaryDark"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```
<TextView
    android:layout_below="@+id/button"
    android:layout_centerInParent="true"
    android:text="TextView 2"
    android:textSize="20sp"
    android:textColor="#FFFFFF"
    android:background="@color/colorPrimaryDark"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```
<TextView android:layout_toEndOf="@+id/button"
    android:layout_centerInParent="true"
    android:text="TextView 4"
    android:textSize="20sp"
    android:textColor="#FFFFFF"
    android:background="@color/colorPrimaryDark"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

4 - Resultado



RECURSOS ESTÁTICOS

Son recursos de diferentes tipos que son guardados en la carpeta **res.** Estos pueden ser usando en los activities, tanto en el archivo kotlin como en el layout xml.

Estos son algunos de los recursos más usados:

layout.- Son archivos xml que nos permite diseñar una pantalla.

menu.- Son archivos xml que nos permite agregar opciones de menú en el lado superior derecho de una pantalla.

drawable-XXX.- Son las carpetas donde guardaremos imágenes e iconos para nuestra aplicación..

mipmap-XXX.-Son las carpetas donde solo guardamos el icono para la aplicación.

values.- Es la carpeta donde podemos guardar datos estáticos como strings, integers, arrays, booleanos, dimensiones, colores y estilos.

string.- Es una forma de crear constantes, por ejemplo mensajes de error, hints, textos, etc. Se encuentra en el archivo **string.xml.**

string-array.- Es una forma de crear array con valores que no cambiarán.

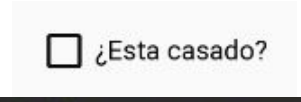
color.- Provee datos de colores en hexadecimal, se encuentra en el archivo **colors.xml**.

styles.- En este archivo podemos configurar el tema para nuestro app completo, se encuentra en el archivo **styles.xml**.

RadioButton


```
<RadioButton android:id="@+id/rn_option_1"
    android:text="Opción 1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

CheckBox

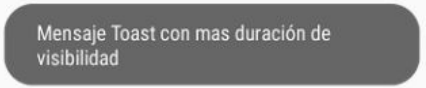



```
<CheckBox android:id="@+id/cb_mi_checkbox"
    android:text="¿Esta casado?"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

ImageView

```
<ImageView android:id="@+id/iv_my_drawable"
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:src="@drawable/artwork"
    android:layout_marginTop="10dp"
    android:layout_marginBottom="10dp"/>
```

Toast




```
toast("Mensaje toast con mas duración de visibilidad")
```

Nota: Se usará la librería anko.

RADIOBUTTON

1. Agregar RadioGroup y RadioButton en el Layout

```
<RadioGroup android:id="@+id/rg_genero"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <RadioButton android:id="@+id/rb_varon"
        android:text="Varon"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <RadioButton android:id="@+id/rb_mujer"
        android:text="Mujer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</RadioGroup>
```

El **RadioButton** es un componente view de tipo selección, tiene solo 2 estados seleccionado o no seleccionado.
Si mostramos varios Radiobuttons y lo queremos agrupar, tenemos que usar **RadioGroup**, este hará que el usuario solo pueda seleccionar uno de los ítems agrupados.

2. Agregar OnCheckedChangeListener() en la clase Kotlin.

El listener retornar 2 parámetros(group y checkedId), el importante es el parámetro **checkedId**, ya que es el **id del RadioButton seleccionado** por el usuario. Para saber cual se la seleccionado, tenemos que igualar el id con todas las opciones.

Ejemplo usando **when**

```
rg_genero.setOnCheckedChangeListener { group, checkedId ->

    when(checkedId){
        R.id.rb_varon -> toast("La opción Varon fue seleccionado.")
        R.id.rb_mujer -> toast("La opción Mujer fue seleccionado.")
        else -> toast("La opción no existe")
    }
}
```

Ejemplo usando **if else**

```
rg_genero.setOnCheckedChangeListener { group, checkedId ->
    if(checkedId == R.id.rb_varon){

        toast("La opción Varon fue seleccionado.")
    }else if(checkedId == R.id.rb_mujer){

        toast("La opción Mujer fue seleccionado.")
    }
}
```

CHECKBOX

1. Agregar el CheckBox en el Layout.

```
<CheckBox android:id="@+id/cb_terminos"
    android:layout_marginTop="6dp"
    android:text="Acepta los terminos y condiciones"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

2. Leer sus datos en la clase kotlin.

```
//Asignando valor al checkbox desde código
cb_terminos.isChecked = true

//Obteniendo el valor del checkbox
val terminosAceptados = cb_terminos.isChecked

//Mostrando el valor en el log.
Log.d( tag: "FormularioActivity",  msg: "terminosAceptados: $terminosAceptados")
```

INTRODUCCIÓN A ADAPTERS

Adapter

Un objeto adapter actúa como puente entre un AdapterView y los datos a mostrar.

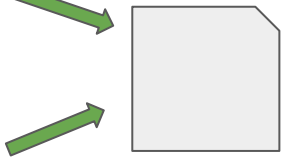
El adapter provee acceso a los elementos de los datos.

El adapter es responsable de hacer un View por cada ítem de la lista de datos.

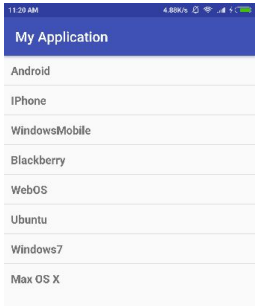
Modelo View para cada ítem

```
private val listaSO = arrayOf("Android", "IPhone",
                              "WindowsMobile", "Blackberry",
                              "WebOS", "Ubuntu", "Windows7",
                              "Max OS X")
```

Lista de objetos(datos)



Adapter



AdapterView

AdapterView

Un AdapterView es un View cuyo hijos están determinados por un Adapter.

Algunos AdapterView

- ~~ListView~~
- ~~GridView~~
- Spinner
- ~~Gallery~~
- RecyclerView

SPINNER

SOMOS PARTNER ORACLE

1. Crear Spinner en el Layout

```
<Spinner
    android:id="@+id/spRole"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

El spinner es un componente similar al conocido ComboBox.

2. Código en la clase Kotlin

```
//LISTA DE OBJETOS
val roles = arrayOf("Asistente", "Jefe", "Gerente")

//ADAPTER
val adapter = ArrayAdapter<String>( context: this, android.R.layout.simple_spinner_item, roles)
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

//CARGANDO ADAPTER EN EL SPINNER
spRole.adapter = adapter

//LISTENER PARA DETECTAR EL ITEM SELECCIONADO
spRole.onItemSelectedListener = object: OnItemSelectedListener {

    override fun onItemSelected(parent: AdapterView<*>?, view: View?, position: Int, id: Long) {
        toast("Item seleccionado: ${roles[position]}")
    }

    override fun onNothingSelected(parent: AdapterView<*>?) {}
}
```

Thank you!
Questions?