UI: Parte I

Desarrollo para plataformas móviles

Borja Martin Herrera
Borja.herrera@u-tad.com

CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

Elementos de un Layout

- TextView:
  - Define un texto en el Layout.
  - Su XML es:

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="TextView" />
```

  - Podemos controlar su tamaño usando la propiedad textSize, que se mide en sp (scale independent pixels).
  - Los sp tienen en cuenta las opciones de tamaño de textos que tenga seleccionado el ususario.
  - Podemos cambiar el texto en ejecución usando el método setText del objeto TextView.

Elementos de un Layout

- EditText:
  - Define un campo de texto editable en el Layout.
  - Su XML es:

```
<EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="textPersonName"
    android:text="Name" />
```

  - InputType define el tipo de campo de texto editable.
    - Lista completa: https://developer.android.com/reference/android/text/InputType
  - Ems es el ancho del campo en número de caracteres.
  - Text es el texto que se muestra antes de escribir.
  - Usamos getText en el objeto editText para leer lo escrito por el usuario.

Elementos de un Layout

- ImageView:
  - Define un imagen en un Layout.
  - Su XML es:

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/utadlogo" />
```

  - La propiedad srcCompat define la imagen que queremos usar.
  - La imagen tiene que estar en la carpeta *drawable.*
  - Podemos cambiar de imagen usando el método setImageResource del objeto ImageView

Elementos de un Layout

- Button:
  - Define un botón.
  - Su XML es:

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button" />
```

Elementos de un Layout

Para darle funcionalidad al botón podemos decirle que hacer usando el método setOnClickListener:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)

    binding.botonCambio.setOnClickListener {   it: View!
        // ejecucion al pulsar el botón
    }

}
```

En el caso de existir más de un método en el listener, es necesario declararlo con el tipo concreto.
Otra de las posibilidades es implementar la interfaz y pasarlo como parámetros

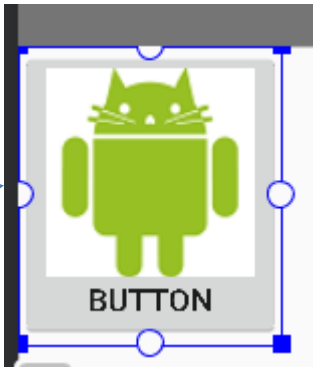
Elementos de un Layout

```
class MainActivity : AppCompatActivity(), OnClickListener {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.botonCambio.setOnClickListener(this)
        binding.botonEnvio.setOnClickListener(this)

    }

    override fun onClick(p0: View?) {
        when(p0.id){
            binding.botonEnvio.id->{}
            binding.botonEnvio.id->{}
        }
    }
}
```

Elementos de un Layout

Podemos añadir una imagen a un botón usando la propiedad drawableLeft, drawableRight, drawableTop, drawableBottom.

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button"
    android:drawableTop="@drawable/androidlogomin"/>
```



Elementos de un Layout

- ImageButton:
  - Define un botón en una imagen.
  - Su XML es:

```
<ImageButton
    android:id="@+id/imageButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/androidlogomin" />
```

Elementos de un Layout

- ToggleButton:

  - Define un botón con dos estados en un Layout.
  - Su XML es:

```
<ToggleButton
    android:id="@+id/toggleButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="On"
    android:textOff="Off"/>
```

  - Las propiedades textOn y textOff indican el texto que se va a mostrar en cada estado del botón.
  - Funciona igual que los botones, podemos asociar un método de la actividad que es llamado cuando se hace click en el botón.

Elementos de un Layout

- ToggleButton:

  - Ejemplo de código del evento onClick para saber si está on u off:

```
public void clickInToggleButton (View v)
{
    boolean isOn = ((ToggleButton)v).isChecked();

    if (isOn)
    {
        //....
    }
    else
    {
        //....
    }
}
```

Elementos de un Layout

- Switch:

  - Al igual que ToggleButton define un botón con dos estados en un Layout.
  - Su XML es:

```
<Switch
    android:id="@+id/switch1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Switch" />
```

  - Funciona exactamente igual que ToggleButton a nivel de código, podemos asociar un método de la actividad que es llamado cuando se hace click en el botón y se puede comprobar si está en on u off.

Elementos de un Layout

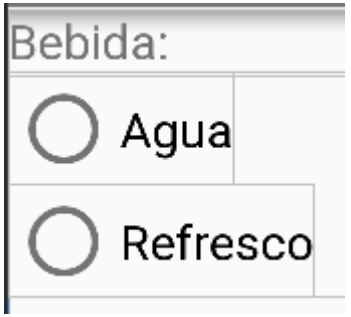
- Checkboxes:
  - Al igual que ToggleButton y Switch define un botón con dos estados en un Layout.
  - Su XML es:

```
<CheckBox
    android:id="@+id/checkBox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="CheckBox" />
```

  - Funciona exactamente igual que ToggleButton/Switch a nivel de código, podemos asociar un método de la actividad que es llamado cuando se hace click en el botón y se puede comprobar si está en on u off.

Elementos de un Layout

- RadioButtons:
  - Al igual que ToggleButton, Switch y CheckBox define un botón con dos estados en un Layout.
  - Tiene la particularidad de que podemos agruparlos y el usuario sólo puede seleccionar uno en concreto.



  - Funciona exactamente igual que ToggleButton/Switch a nivel de código, podemos asociar un método de la actividad que es llamado cuando se hace click en el botón y se puede comprobar si está en on u off.

Elementos de un Layout

- RadioButtons:
  - Para conseguir este comportamiento tenemos que agruparlos en un RadioGroup.

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <RadioButton
        android:id="@+id/radioButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Agua" />

    <RadioButton
        android:id="@+id/radioButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Refresco" />
</RadioGroup>
```

Elementos de un Layout

- RadioButtons:
  - En código podemos distinguir la opción seleccionada preguntando por su id.

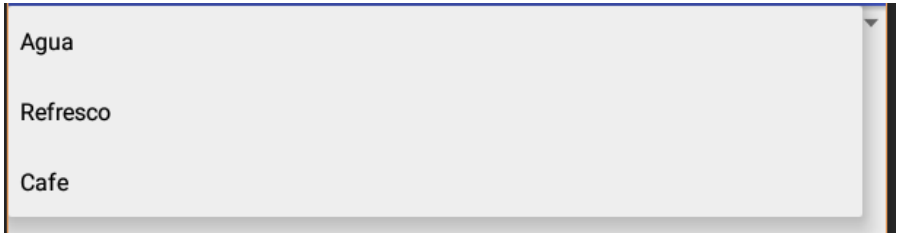
```
lateinit var radioSeleccioando: RadioButton;
radioSeleccioando = findViewById(grupoRadios.checkedRadioButtonId);
if (grupoRadios.checkedRadioButtonId >= 0) {
   Snackbar.make(p0, radioSeleccioando.text, Snackbar.LENGTH_SHORT).show()

    when (radioSeleccioando.id){
        R.id.radio_uno->{}
        R.id.radio_dos->{}
        R.id.radio_tres->{}
    }
}
```

- O bien ponerle un escuchador al grupo

```
grupoRadios.setOnCheckedChangeListener { radioGroup, i ->
    var radioSeleccioando: RadioButton = findViewById(i);
    Snackbar.make(grupoRadios,radioSeleccioando.text,Snackbar.LENGTH_SHORT).show()
}
```

Elementos de un Layout

- Spinner:



  - Define una *drop-down list* de opciones.
  - Su XML:

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

  - Para rellenar la lista usamos la propiedad *entries*:

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/listaBebidas"/>
```

```
<string-array name="listaBebidas">
    <item>Agua</item>
    <item>Refresco</item>
    <item>Cafe</item>
</string-array>
```

Strings.xml

Elementos de un Layout

- Spinner:
  - Podemos leer la opción seleccionada poniendo un escuchador al spinner:

```
binding.spinnerComplejo.onItemSelectedListener = object : OnItemSelectedListener{
    override fun onItemSelected(p0: AdapterView<*>?, p1: View?, p2: Int, p3: Long) {
        Log.v( tag: "testSpinner", adaptadorSencillo.getItem(p2).toString())
    }

    override fun onNothingSelected(p0: AdapterView<*>?) {
        TODO( reason: "Not yet implemented")
    }

}
```

Elementos de un Layout

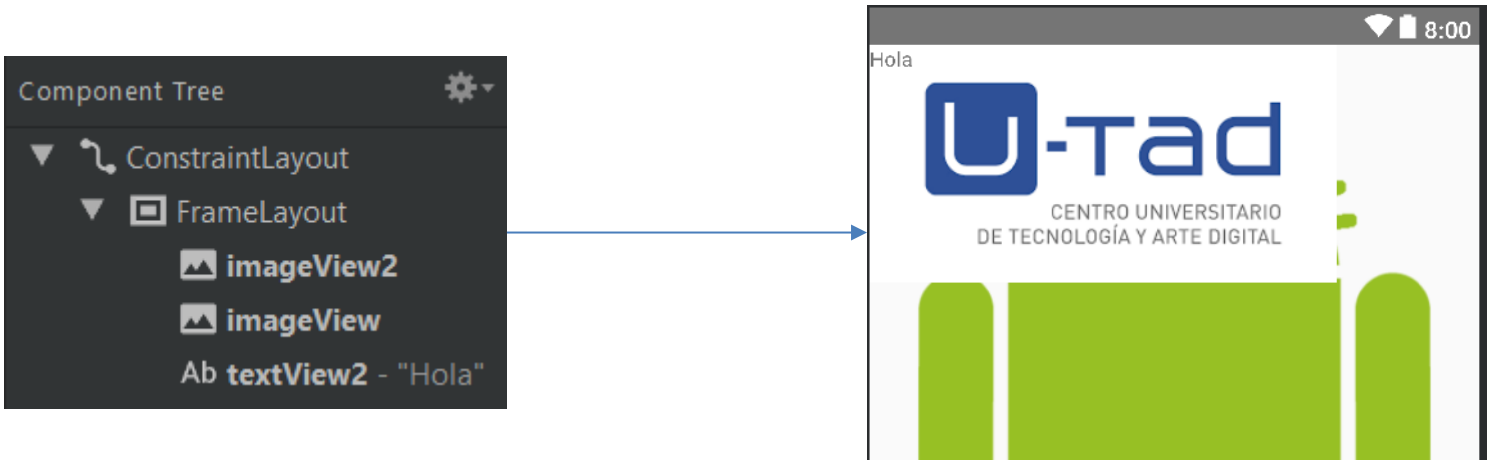
- LinearLayout:
  - Nos permite dividir el Layout en vertical u horizontal en un número N de elementos del mismo tamaño.
  - Tiene el campo orientation para definir el tipo de *LinearLayout*

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
</LinearLayout>
```

  - Se pueden usar de manera anidada para definir el *Layout*.

Elementos de un Layout

- FrameLayout:
  - Nos permite solapar elementos en el mismo espacio.



Elementos de un Layout

- FrameLayout:
  - En XML:

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/androidlogo" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/utadlogo" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hola" />
</FrameLayout>
```

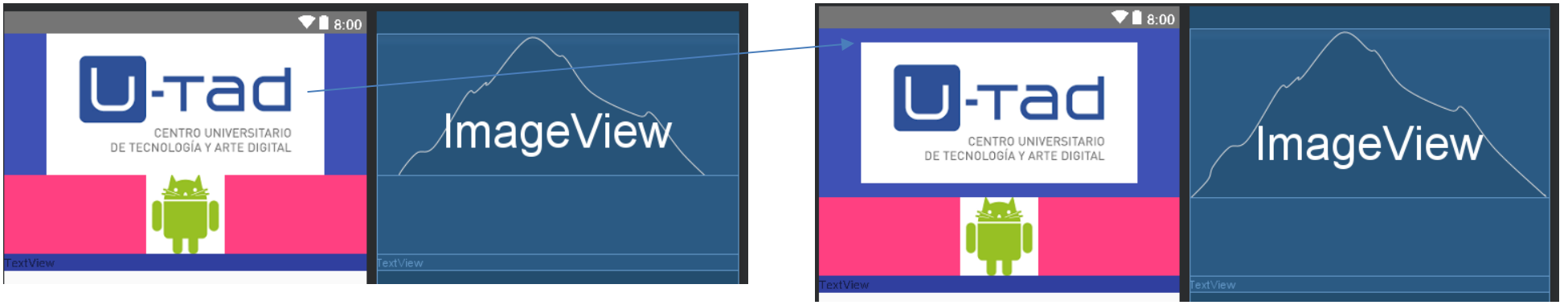
Elementos de un Layout

- FrameLayout:
  - Podemos cambiar la posición de los elementos usando la propiedad gravity:

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/androidlogo" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/utadlogo"
        android:layout_gravity="center" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hola"
        />
</FrameLayout>
```

Elementos de un Layout

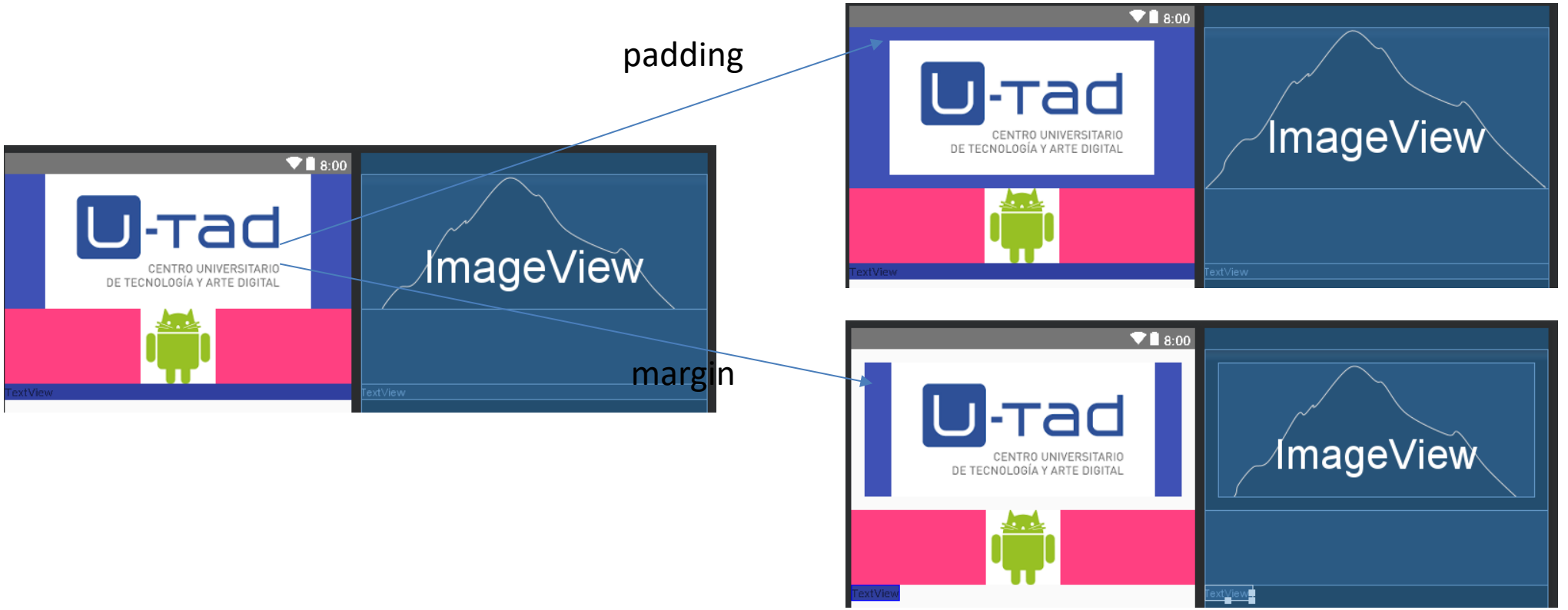
- Padding:
  - La propiedad Padding nos permite añadir un margen entre el elemento y su padre. Se puede especificar si el padding es en todos los bordes o en alguno/algunos en concreto (usando paddingLeft, paddingRight, etc)




```
<ImageView
    android:id="@+id/imageView3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="fill_horizontal"
    app:srcCompat="@drawable/utadlogo"
    android:background="@color/colorPrimary"
    android:padding="16dp"/>
```

Elementos de un Layout

- Margin:
  - Mientras el *padding* es un espacio “interno”, de los bordes de la vista hacia “dentro”, el *margin* es un espacio “externo”

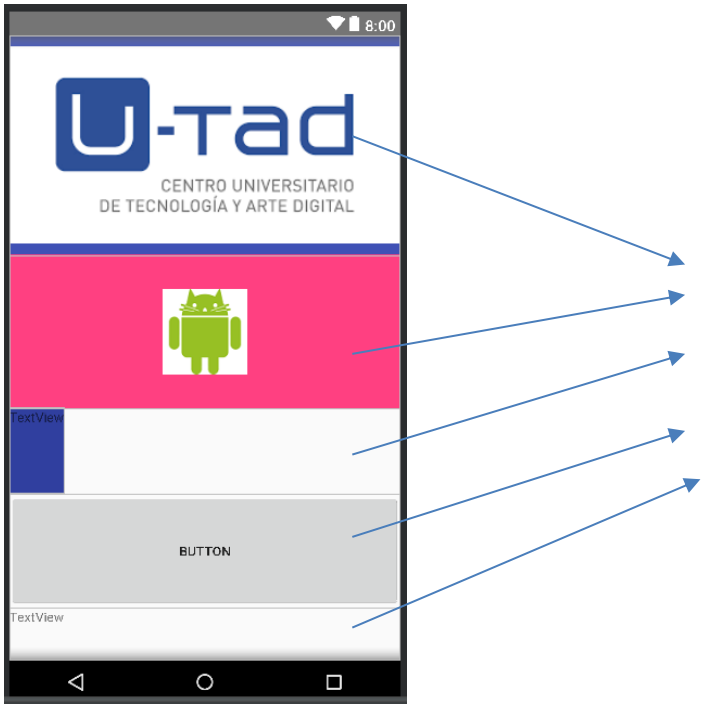


Elementos de un Layout

- Weight:
  - La propiedad weight nos permite distribuir el espacio dentro de un Layout dándole “porcentajes” de cuanto queremos que ocupen las cosas.
  - El “porcentaje” se define por un número entero (1, 2, 3…) que asignamos a cada elemento y se calcula sobre la suma de todos.

Elementos de un Layout

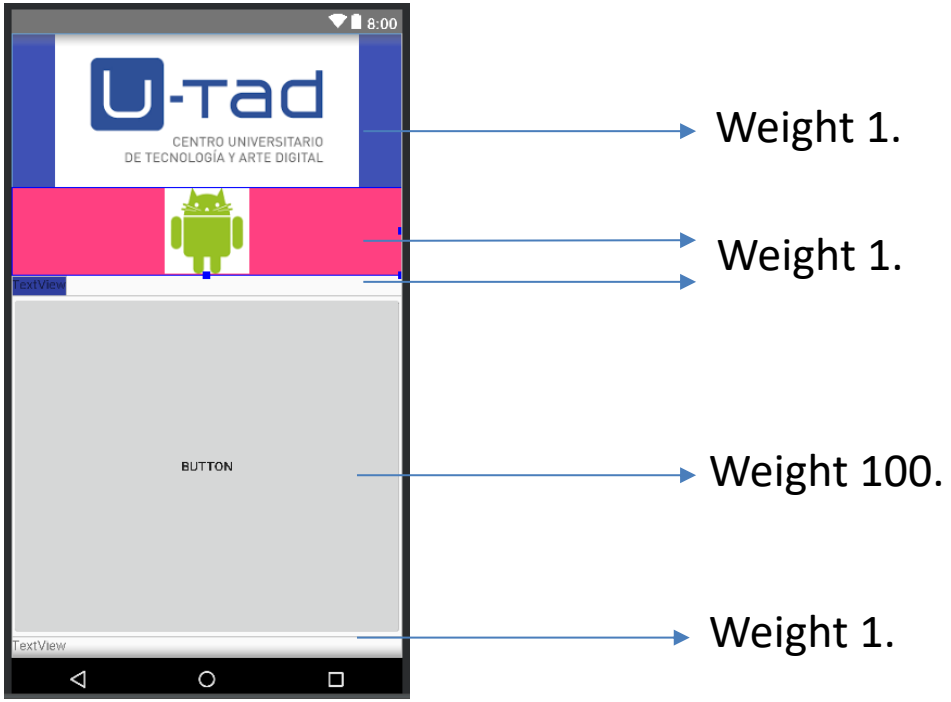
- Weight:
  - Por ejemplo: 5 elementos con valor 2 en la propiedad weight. Se calcula sobre 10 (5 elementos * valor 2). Cada uno es 2/10 partes -> 1/5 parte de la pantalla.



Cada elemento weight 2.

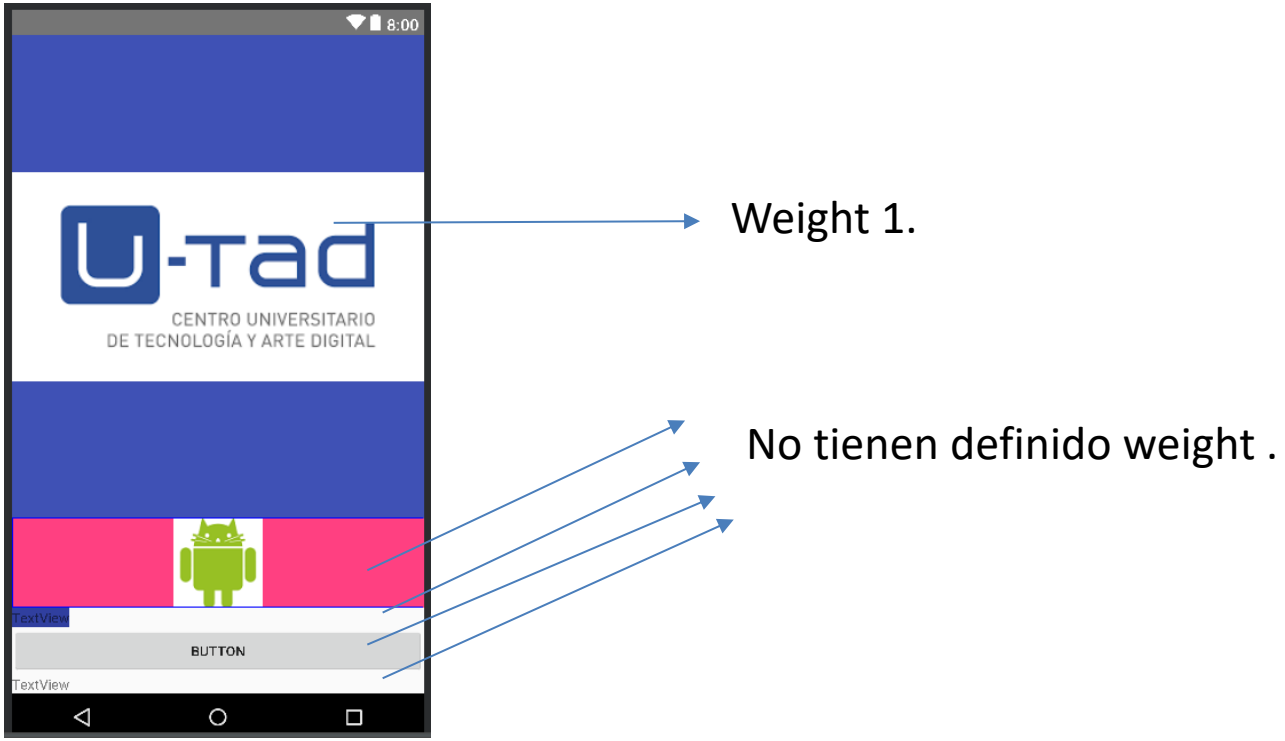
Elementos de un Layout

- Weight:
  - Primero se pintan todos los elementos ocupando el espacio que tienen que ocupar, lo que se reparte es el espacio extra que sobra. Ejemplo:



Elementos de un Layout

- Weight:
  - Si sólo le ponemos weight a un elemento, se expande hasta ocupar todo el espacio posible.



Ejemplo de App

Ejercicio: Crear una app que me permita seleccionar un menú en un restaurante de comida rápida y mandarlo en texto plano a la app de recibir mensajes que realizamos en el tema 2 (hay que adaptarla para que pueda mostrar el menú en texto). Tiene que sobrevivir a los cambios de orientación e incluir una imagen de fondo para el formulario.

Características del formulario (Metedlo todo en un ScrollView):

Nombre de cliente (Edit Text)
Número de teléfono (Edit text numérico)
Pedido a domicilio (ToggleButton):
Tipo Hamburguesa (Spinner): Normal, Con Queso, con Bacon…
Ingredientes Extra (Checkbox): Pepinillos, extra de salsa, etc…
Tipo de Bebida (RadioButton): Refresco, agua, etc.
Tipo de patatas (Spinner): Normal, Deluxe, etc
Tipo de envío de formulario (Switch): o texto plano compatible con nuestra app o por correo.
Botón de enviar.