tema2_Componentes_basicos

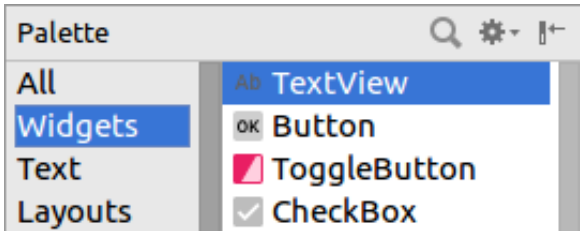
Tema 2. Componentes básicos

ÍNDICE.

1.1 Editores

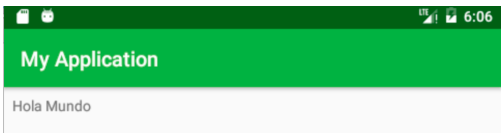
TextView

TextView es un control que muestra un texto, generalmente es informativo.



En el layout añadiremos el código indicado y el resultado será:

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

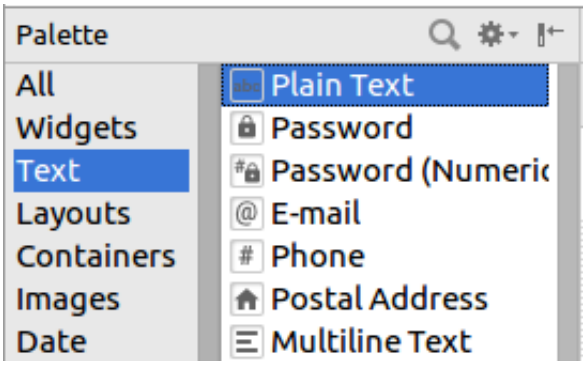


No debemos confundir la propiedad **gravity** con **layout:gravity,** esto suele crear bastante confusión sobre todo cuando uno se inicia en android.

Gravity hace referencia a la posición del contenido del control, mientras que layout:gravity hace referencia a la posición del control dentro del layout. Es decir, si queremos alinear el texto dentro del control hacia la derecha, utilizaremos gravity, y marcaremos la opción de right, mientras que si queremos alinear todo el control a la derecha de la pantalla(activity), utilizaremos layout:gravity y también marcaremos right.

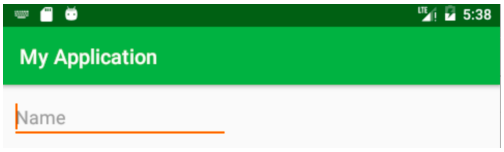
EditText

EditText es un control que permite introducir un texto mediante teclado.
El control básico es Plain Text, aunque existen muchas otras opciones que permiten mostrar el teclado correspondiente, cuando el usuario pulsa sobre el control.



En el layout añadiremos el código indicado y el resultado será:

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



Para obtener el texto que el usuario introduce en el EditText, lo primero es declarar el control en nuestra clase de la siguiente forma:
En Java:

```
EditText etTextoIntroducido = findViewById(R.id.etEntradaTexto);

String texto = etTextoIntroducido.getText().toString();
```

En Kotlin:

```
val etTextoIntroducido = findViewById(R.id.etEntradaTexto) as EditText

val texto = etTextoIntroducido.text
```

Podemos añadir **etiquetas flotantes** en los EditText, para ello debemos incluir el EditText dentro de un control **TextInputLayout**. Este control forma parte de la librería **material** y por tanto, para poder utilizarlo, habrá previamente que incluir la librería en nuestro proyecto.

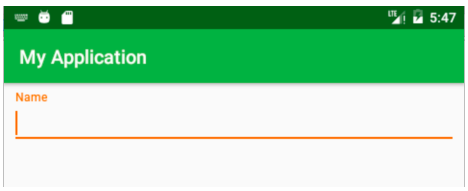
Para enlazar la librería accedemos al fichero **build.gradle** de la **app** y en dependencias la enlazamos, a continuación sincronizamos el proyecto (Sync Now):

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test:runner:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
    implementation 'com.google.android.material:material:1.0.0'
}
```

Nota: Tened en cuenta que las versiones de las librerías deben coincidir.

En el layout insertamos el control TextInputLayout y en su interior el EditText, como se indica:

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/TiLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

<com.google.android.material.textfield.TextInputEditText

        android:id="@+id/textInputEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:hint="Name" />

</com.google.android.material.textfield.TextInputLayout>
```



1.2 Botones

Tenemos varios tipos de botones (button) disponibles que se adaptarán a la mayoría de nuestras necesidades:

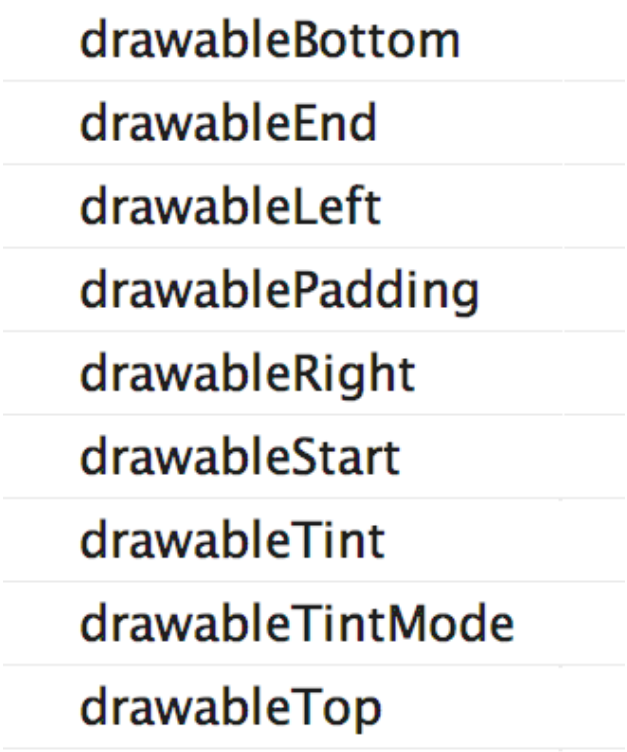
- **Button**: El control mas básico, botón rectangular con texto.
- **ImageButton**: Igual que el anterior, pero en lugar del típico texto tenemos una imagen.
- **FloatingActionButton** (FAB): botón circular.
- **Chip**: botón redondeado que incluye una etiqueta, un icono opcional y un icono de cierre también opcional.
- **ChipGroup**: Contenedor de un grupo de Chips.
- **MaterialButton**: nuevo botón incorporado en la librería material, con un diseño más atractivo.

Button

Botón rectangular con texto, aunque se puede acompañar de una imagen.

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

Para añadir una imagen al botón, accederemos a las propiedades siguientes:



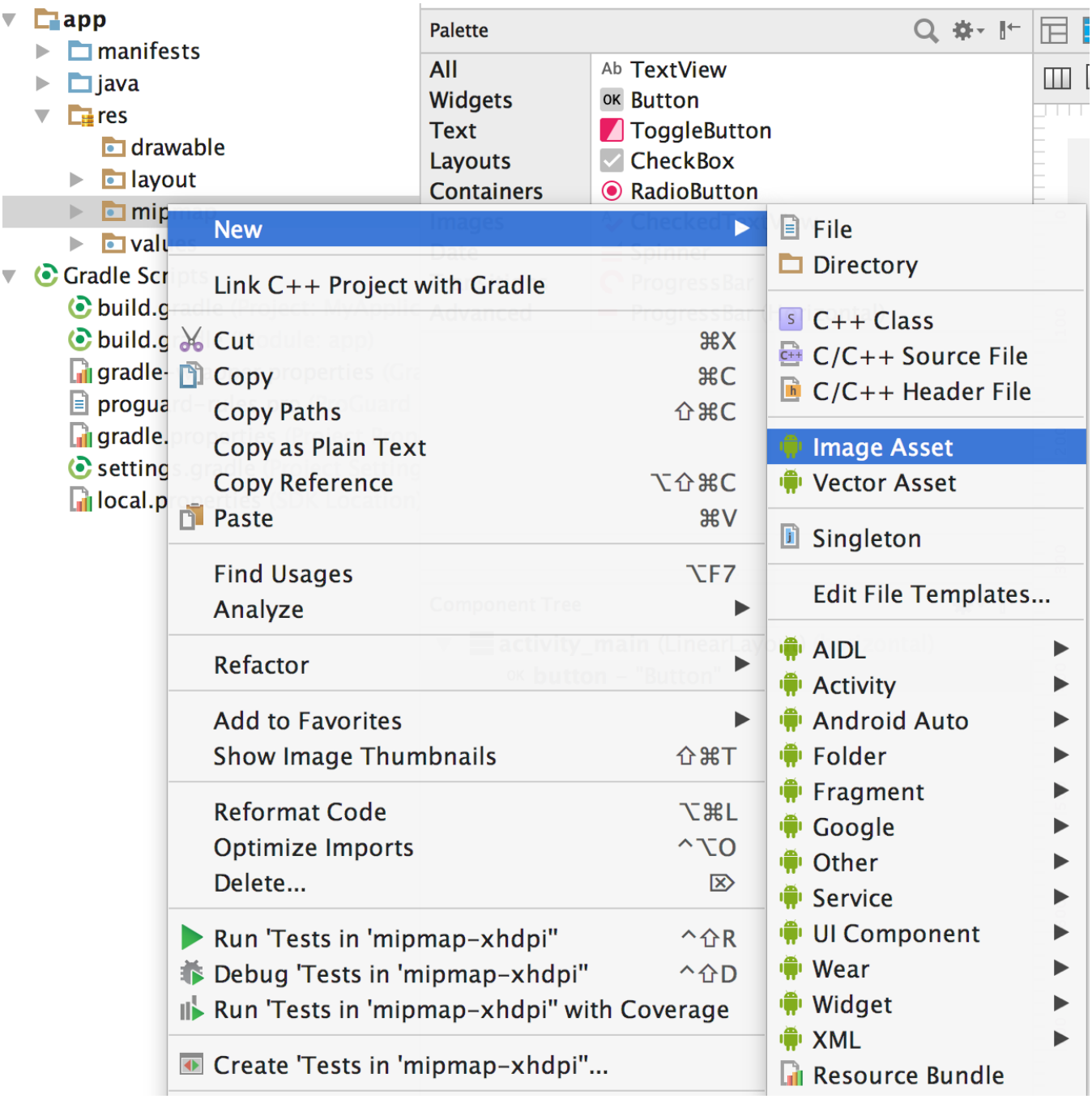
Ejemplo. Botón con una imagen a la izquierda del texto.

En la propiedad drawableLeft indicaremos la ruta de la imagen a añadir. En este caso, la imagen o icono (@android:drawable/ic_lock_lock) está predefinida por Android.

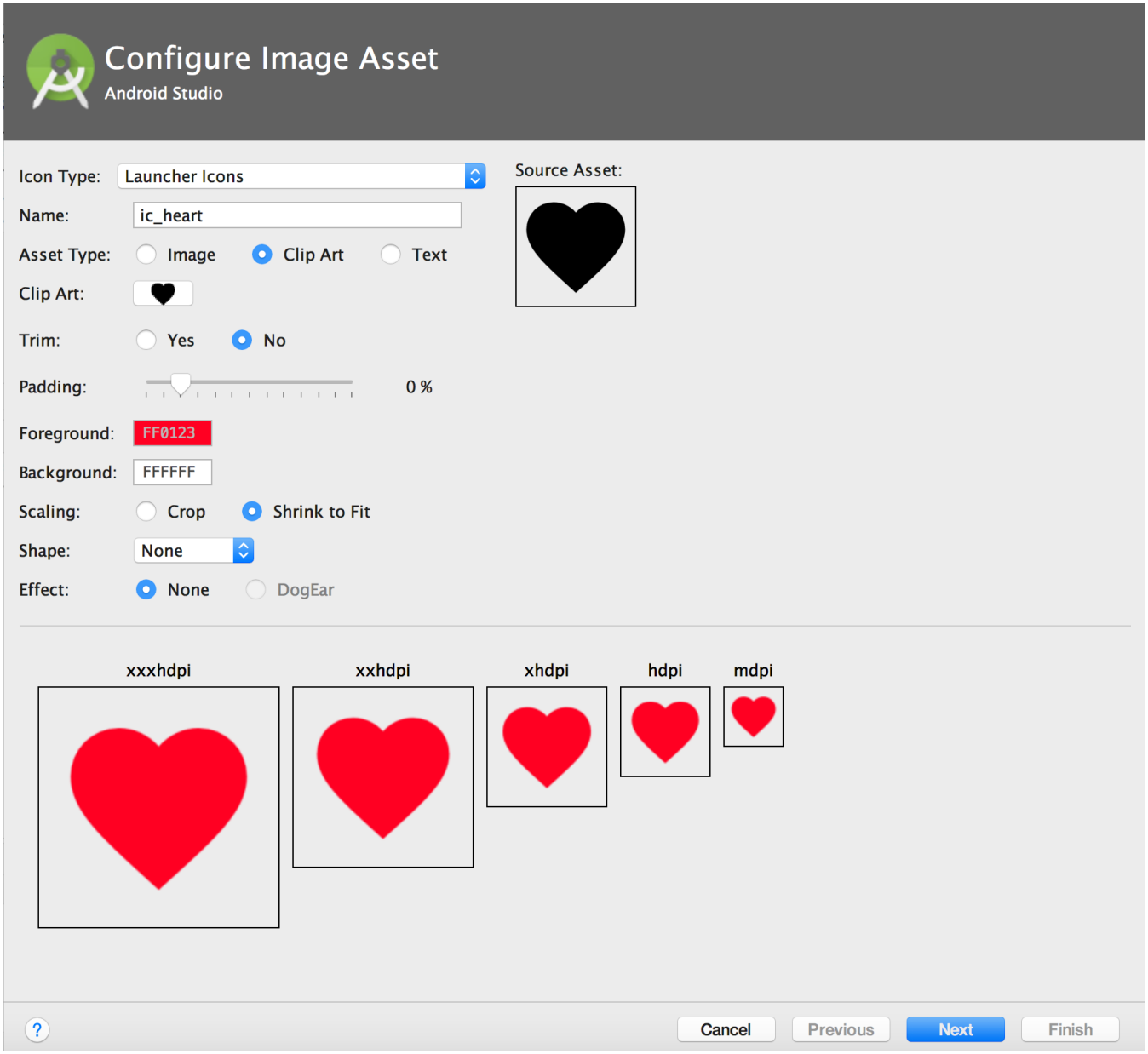
```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawableLeft="@android:drawable/ic_lock_lock"
    android:text="Button" />
```

Para crear una imagen o icono que no esté predefinido por Android, utilizaremos la herramienta Image Asset proporcionada por Android Studio.

Para ello, seleccionamos la carpeta mipmap y con el botón derecho accedemos a la utilidad:



Se despliega la utilidad y elegimos en clipart el icono, por ejemplo, del corazón. Le ponemos de nombre ic_heart, sin forma de fondo (shape) y de color rojo (foreground). Se creará para cada densidad de pantalla el icono correspondiente en la carpeta mipmap.



El resultado será el siguiente:

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

ImageButton

Como el control Button, pero en lugar del típico texto tenemos una imagen.

Utilizamos en el ejemplo una imagen definida por Android.

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

FloatingActionButton (FAB)

Botón circular, incluido en la librería material, por tanto para poder utilizarlo recordad tened incluida dicha librería en el fichero build.gradle de la aplicación.
En el layout añadiremos el código indicado y el resultado será:

```
<com.google.android.material.floatingactionbutton.FloatingActionB
   android:id="@+id/fab"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   app:srcCompat="@mipmap/ic_add"
   app:fabSize="normal" />
```

La propiedad **src** indica la ruta donde se encuentra la imagen, y **fabSize** el tamaño del botón que puede tomar dos valores: normal o mini.

Chip / ChipGroup

Botón redondeado que puede tomar cuatro tipos de estilos distintos: **Entry (Input), Filter, Choice y Action**. Normalmente se engloban dentro de un ChipGroup.
En el layout añadiremos el código indicado y el resultado será:

```
<com.google.android.material.chip.ChipGroup
    android:id="@+id/chip_group"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
    app:singleSelection="true">

  <com.google.android.material.chip.Chip
    android:id="@+id/chip"
    style="@style/Widget.MaterialComponents.Chip.Entry"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Chip 1"
    android:textAppearance="?android:textAppearanceMedium"/>

  <com.google.android.material.chip.Chip
    android:id="@+id/chip2"
    style="@style/Widget.MaterialComponents.Chip.Filter"
    android:layout_width="wrap_content"
    android:layout_height="35dp"
    android:text="Chip 2"android:textAppearance="?
    android:textAppearanceMedium"/>

  <com.google.android.material.chip.Chip
    android:id="@+id/chip3"
    style="@style/Widget.MaterialComponents.Chip.Choice"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Chip 3"
    android:textAppearance="?android:textAppearanceMedium"/>

  <com.google.android.material.chip.Chip
    android:id="@+id/chip4"
    style="@style/Widget.MaterialComponents.Chip.Action"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Chip 4"
    android:textAppearance="?android:textAppearanceMedium"/>

 </com.google.android.material.chip.ChipGroup>
```

Para saber que Chip ha seleccionado el usuario, podemos utilizar un listener directamente sobre el ChipGroup, como veremos en el código:

En Java:

```
ChipGroup chipGroup = findViewById(R.id.chip_group);

chipGroup.setOnCheckedChangeListener(new OnCheckedChangeListener(

        public void onCheckedChanged(ChipGroup group, int checked
                Chip checkedChip = group.findViewById(checkedId);
                option = checkedChip.getText().toString();
            }
});
```

En Kotlin:

```
val chipGroup = findViewById(R.id.chip_group) as ChipGroup

chipGroup.setOnCheckedChangeListener(object : ChipGroup.OnChecked

    override fun onCheckedChanged(group: ChipGroup, checkedId: In
        val checkedChip = group.findViewById<Chip>(checkedId)
        option = checkedChip.text as String
    }
})
```

MaterialButton

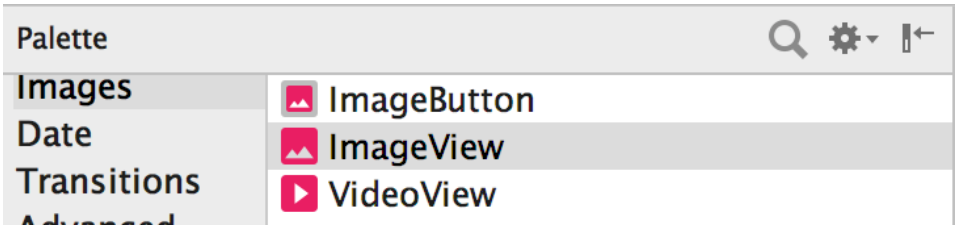
Nuevo tipo de botón con un diseño más atractivo.
En el layout añadiremos el código indicado y el resultado será:

```
<com.google.android.material.button.MaterialButton
        android:id="@+id/material_button"
        style="@style/Widget.MaterialComponents.Button.Unelevated
        android:layout_width="140dp"
        android:layout_height="wrap_content"
        android:text="Button"
        android:textAppearance="@style/TextAppearance.AppCompat.L
        app:cornerRadius="20dp" />
```

1.3 Imágenes

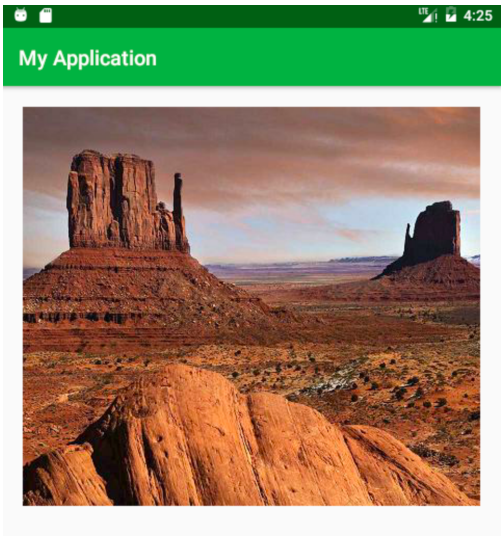
ImageView

El ImageView es un control que muestra una imagen con el escalado seleccionado.



En el layout añadiremos el código indicado o arrastraremos el control, y el resultado será:

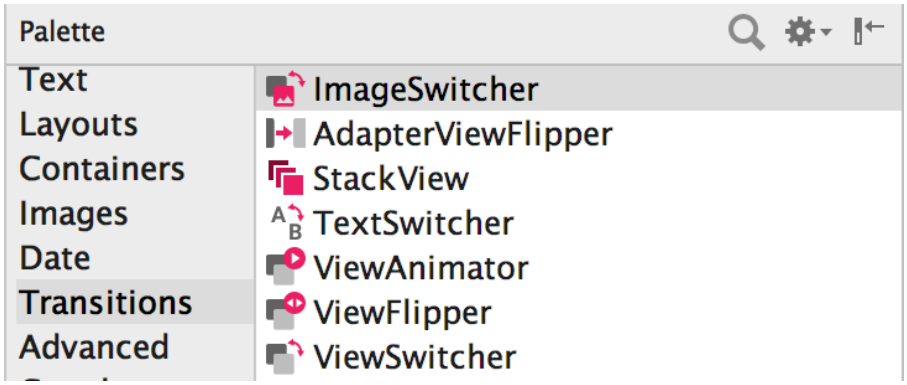
```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:scaleType="centerCrop"
    app:srcCompat="@drawable/canyon" />
```



La propiedad *app:srcCompat* indica la ruta donde se encuentra la imagen a reproducir, mientras que la propiedad *android:scaleType* define el tipo de escalado a utilizar.

ImageSwitcher

En algunas aplicaciones no siempre interesa que la imagen seleccionada por el usuario aparezca o se reproduzca de forma “brusca” sobre un *ImageView*. Puede resultar más atractivo el hecho de que la imagen seleccionada aparezca o desaparezca con una animación, dando un efecto más elegante. Para realizar este efecto de animación, es necesario utilizar una vista *ImageSwitche*r.



Al hacer click sobre un *ImageView*, vamos a reproducir la imagen de éste en un *ImageSwitcher* de manera que aparezca sobre este componente con un deslizamiento hacia la izquierda y desaparezca con un deslizamiento hacia la derecha.

El código del layout será el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="16dp"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:scaleType="centerCrop"
        app:srcCompat="@drawable/canyon" />

    <ImageSwitcher
        android:id="@+id/imageswitcher"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1" />

</LinearLayout>
```

En Java, nuestra actividad principal deberá implementar la interfaz *ViewFactory*, ya que es necesaria esta interfaz para crear las vistas que utilizará el *ImageSwitcher*. Para ello, necesitaremos además, implementar el método *makeView()*. Este método creará una nueva vista que será añadida al *ImageSwitcher*, en este caso un *ImageView*.

```
public class MainActivity extends AppCompatActivity implements Vi

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

        final ImageSwitcher imageSwitcher = (ImageSwitcher) findV
        imageSwitcher.setFactory(this);
        imageSwitcher.setInAnimation(AnimationUtils.loadAnimation
        imageSwitcher.setOutAnimation(AnimationUtils.loadAnimatio

        final ImageView imagen = (ImageView) findViewById(R.id.im
        imagen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                imageSwitcher.setImageDrawable(imagen.getDrawable
            }
        });
    }

    public View makeView() {
        ImageView imageView = new ImageView(this);
        imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
        imageView.setLayoutParams(new ImageSwitcher.LayoutParams(

        return imageView;
    }
}
```

En Kotlin:

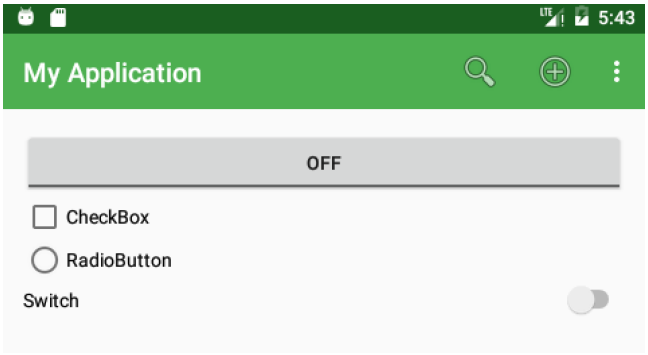
```
class MainActivity : AppCompatActivity(), ViewSwitcher.ViewFactor
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val imageSwitcher = findViewById(R.id.imageswitcher) as I
        imageSwitcher.setFactory(this)
        imageSwitcher.inAnimation = AnimationUtils.loadAnimation(
        imageSwitcher.outAnimation = AnimationUtils.loadAnimation

        val imagen = findViewById(R.id.imageView) as ImageView
        imagen.setOnClickListener(object : View.OnClickListener {
            override fun onClick(v: View) {
                imageSwitcher.setImageDrawable(imagen.getDrawable
            }
        })
    }

    override fun makeView(): View {
        val imageView = ImageView(this)
        imageView.scaleType = ImageView.ScaleType.FIT_CENTER
        imageView.layoutParams = FrameLayout.LayoutParams(ViewGro
        return imageView
    }
}
```

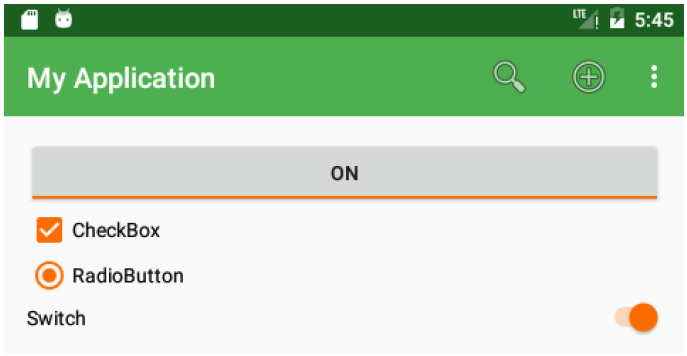
1.4 Controles de selección

- **ToggleButton**: botón con dos estados off / on.
- **CheckBox**: botón que puede estar seleccionado (checked) o sin seleccionar (unchecked).
- **RadioButton**: control que permite al usuario elegir una opción entre varias, si va incluido dentro de un RadioGroup. También tiene dos estados: seleccionado o sin seleccionar.
- **Switch:** el nuevo botón para versiones superiores a Android 4, viene a sustituir a ToggleButton.

Controles de selección "unchecked"



Controles de selección "checked"



ToggleButton, CheckBox y Switch

Son un tipo de botones booleanos, que puede contener dos estados: "checked" o "unchecked". El método que indica el estado del botón es isChecked().

Por ejemplo:

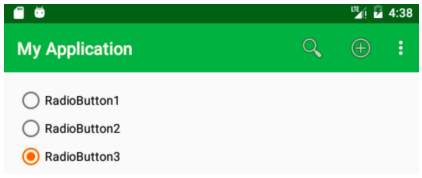
```
if (checkBox.isChecked()) {
                // el botón ha sido activado
}
```

RadioButton

Cada RadioButton es un control que puede tener 2 estados, es similar al CheckBox pero en este caso el usuario sólo puede pulsar sobre una de las opciones.

Los RadioButton deben estar contenidos dentro de un RadioGroup (Android Studio no coloca por defecto un identificador para el RadioGroup):

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

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RadioButton1"
        android:id="@+id/radioButton"
        android:checked="false" />

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RadioButton2"
        android:id="@+id/radioButton2"
        android:checked="false" />

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RadioButton3"
        android:id="@+id/radioButton3"
        android:checked="false" />

</RadioGroup>
```



Para saber que RadioButton ha seleccionado el usuario, podemos utilizar un listener directamente sobre el RadioGroup, como veremos en el código:

```
RadioGroup rGroup = findViewById(R.id.radioGroup);

rGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeL

        public void onCheckedChanged(RadioGroup rGroup, int check
```

```
            {
                RadioButton checkedRadioButton = (RadioButton)rGr
                option = checkedRadioButton.getText().toString();
            }
});
```

En Kotlin:

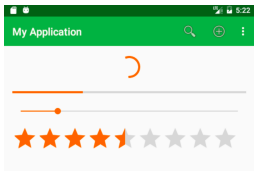
```
val rGroup = findViewById(R.id.radioGroup) as RadioGroup

rGroup.setOnCheckedChangeListener(object : RadioGroup.OnCheckedCh

    override fun onCheckedChanged(group: RadioGroup, checkedId: I
        val checkedRadioButton = rGroup.findViewById<RadioButton>
        option = checkedRadioButton.text as String
    }
})
```

1.5 Controles de progreso

Disponemos de tres barras para poder indicar el progreso de una operación o selección del usuario:

- **ProgressBar**: barra de progreso en formato circular con animación u horizontal. El usuario no puede seleccionar el progreso deseado.
- **SeekBar**: barra de progreso horizontal, permite que el usuario haga su selección.
- **RatingBar:** barra de progreso horizontal con estrellas. También permite que el usuario haga su selección.

```
<ProgressBar
    android:id="@+id/progressBar"
    style="?android:attr/progressBarStyle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"/>

<ProgressBar
    android:id="@+id/progressBar2"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:progress="30" />
```




```
<SeekBar
    android:id="@+id/seekBar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"/>

<RatingBar
    android:id="@+id/ratingBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numStars="9" />
```

ProgressBar, SeekBar y RatingBar

Los métodos en Java que permiten situar o conseguir el progreso en los controles son:

ProgressBar	SeekBar	RatingBar
setProgress(int)	setProgress(int)	setRating(float)
getProgress(int)	getProgress(int)	getRating(float)

En Kotlin, el progreso en los controles se obtiene con los métodos: ***progress*** (ProgressBar y SeekBar) y ***rating*** (RatingBar).