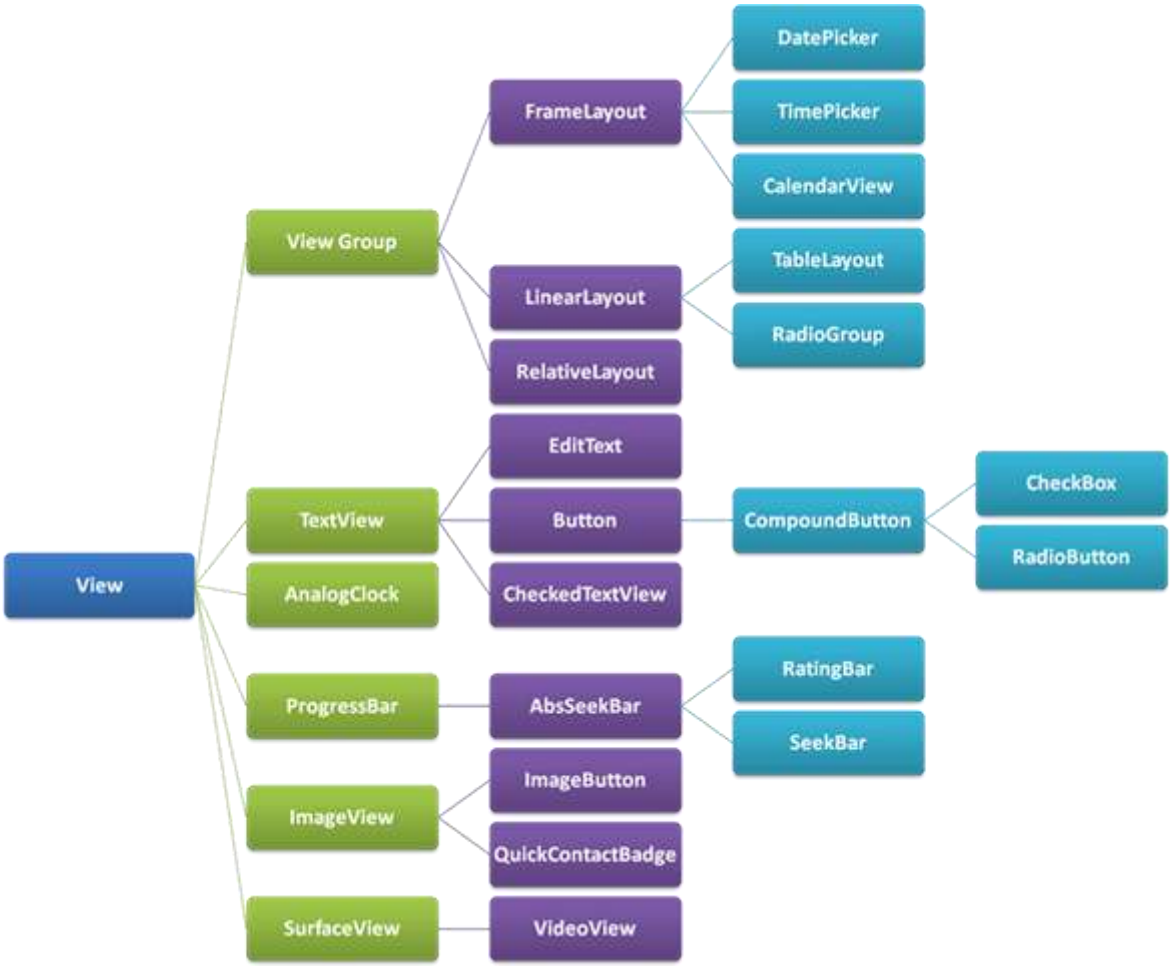
INDICE

1.1. Introducción

- Los botones (botón) permiten al Usuario indicar a la aplicación que realiza una acción.
- Los botones pueden tener texto, una imagen o ambos:
- O el texto o la imagen comunican claramente al usuario cuál es la función del botón.

- Estes controles son subclases de:
  - ✓ Button de TextView
  - ✓ ToogleButton de CompoundButton
  - ✓ ImageButton de ImageView (que se verá proximamente)



- Observe cómo se definen los tres tipos de botones anteriores:
  - ✓ **Botón con texto**:

```
<Button
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="@string/button_text"
     ... />
```

  - ✓ **Botón con imagen**

```
<ImageButton
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:src="@drawable/button_icon"
     ... />
```

- ✓ **Botón con texto e imagen.** La propiedad de **Android: drawableLeft** indica dónde se encuentra la imagen.

```
<Botón
    android: layout_width = "wrap_content"
    android: layout_height = "wrap_content"
    android: text = "@ string / button_text"
    android: drawableLeft = "@ drawable / button_icon"
    ...  />
```

- Los botones tienen una propiedad de **Android: onClick = "MyMethod"** que nos permite llamar al método indicado cuando lo presionamos.
  - ✓ Ese método debe declararse en Java como **public void oMyMethod (View v) {...}**
  - ✓ Recibe el objeto View que lo llamo.
- Los botones son subclases de **TextView**.
- **Referencias**:
  - ✓ El Control botón: http://developer.android.com/reference/android/widget/Button.html
  - ✓ Introducción a los botones: http://developer.android.com/design/building-blocks/buttons.html
  - ✓ Para programadores: http://developer.android.com/guide/topics/ui/controls/button.html
  - ✓ Para descargar iconos: http://www.iconarchive.com
  - ✓ Para descargar imágenes: http://www.openclipart.org

1.1.1.Botones de 2 estados: ToggleButton / Switch

- Hay otro tipo de botón de 2 estados (ON / OFF).
- Uno más básico: **<ToggleButton>**

- Y otro más avanzado (Android versión 4.0 o superior): **<Switch />**

  - ✓ Te permiten cambiar su estado al pasar de un estado a otro. La operación es similar al control ToogleButton.
- Un objeto ToogleButton / Switch hereda de la clase **CompoundButton**, que a su vez hereda de la clase **Button**.
- Por tanto funcionan de la misma forma, pero además de este tipo de botones:
  - ✓ tiene 2 estados (Verdadero / Falso), que podemos verificar con el método **isChecked ()**
  - ✓ Para cada estado, podemos mostrar un texto diferente en el botón: **android: TextOn** y **android: TextOFF**.

Para que aparezca el texto, la propiedad showText debe cambiarse a true. A nivel de diseño, no se recomienda el uso de texto si estamos usando un Material Theme.

- **Referencias:**
  - ✓ El control ToggleButton: https://developer.android.com/reference/android/widget/ToggleButton.html
  - ✓ El control switch: https://developer.android.com/reference/android/widget/Switch.html
  - ✓ Introducción a los botones de 2 estados: https://developer.android.com/guide/topics/ui/controls/togglebutton.html

1.2. Casos prácticos

- Partimos del proyecto inicial que hemos creado en unidades anteriores.
- Recordar que si se está utilizando bibliotecas de compatibilidad (predeterminadas), **vuestra activity se derivará de AppCompactActivity** y no de Activity.
- En los siguientes ejemplos, toda activity deriva de Activity, pero lo normal es que se utilice AppCompactActivity (cuidado al copiar los ejemplos).
- Si no lo tenemos creado antes, crear un paquete llamado **UI** como un subpaquete de su paquete principal.
- Dentro del paquete de IU, crearemos un nuevo paquete llamado: **Buttons**.
- Dentro del paquete Buttons crea una nueva *'Empty Activity*' llamada: **UD02_01_Botones** de tipo Launcher y sin compatibilidad.
- Modificar el archivo **AndroidManifiest.xml** y agregar una etiqueta a la activity como hemos hecho en casos anteriores.
- Creemos 3 botones:
  - ✓ 1 Botón con texto.
  - ✓ 1 ToogleButton.
  - ✓ 1 botón con imagen.

Tres Botones

Cuando entremos en la aplicación este será su estado.

Si pulsamos el primer botón (Texto) se mostrará un texto en pantalla.

Al presionar el segundo botón (ToogleButton) se mostrará otro texto en la pantalla y el botón cambiará su estado y el texto del mismo.

Si lo volvemos a presionar nos muestra otro texto y vuelve a cambiar el estado.

Si pulsamos el tercer botón (Imagen) se volverá a visualizar un texto diferente.

1.2.1.Creación del layout

- El layout XML tiene 3 botones y 1 TextView.
- Observar como hay un layout dentro de otro: uno dispone los elementos en vertical y otro en horizontal.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

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

        <Button
            android:id="@+id/btnBoton_UD02_01_buttons"
            android:layout_width="150sp"
            android:layout_height="wrap_content"
            android:text="Pulsar aquí" >
        </Button>

        <ToggleButton
            android:id="@+id/tbtnDosEstados_UD02_01_buttons"
            android:layout_width="100dp"
            android:layout_height="match_parent"
            android:textOff="Apagado"
            android:textOn="Acceso" >
        </ToggleButton>

        <ImageButton
            android:id="@+id/ibtnImagen_UD02_01_buttons"
            android:layout_width="55sp"
            android:layout_height="50sp"
            android:scaleType="fitXY"
            android:src="@drawable/ic_icono1" >
        </ImageButton>
    </LinearLayout>

    <TextView
        android:id="@+id/txtTv_accion_UD02_01_Buttons"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/texto_tv_string" />

</LinearLayout>
```

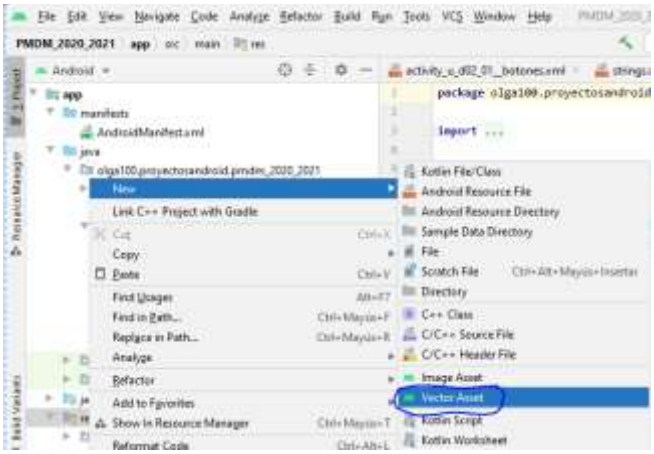
- Si cargamos este layout y no realizamos ninguna codificación en Java, no pasará nada con los botones:

  Esto se debe a que no hemos codificado ninguna acción para cuando se haga clic en cada uno de ellos.

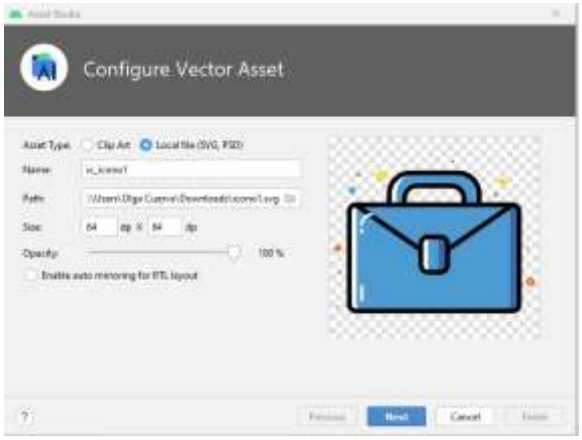
1.2.2.Usar gráficos SVG con un ImageButton

- Si queremos usar gráficos vectoriales podemos 'cargarlos' en nuestro proyecto con la herramienta Vector Asset:

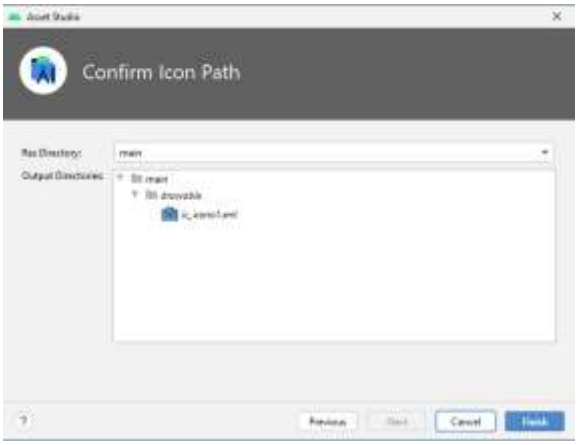
Usar gráficos svg



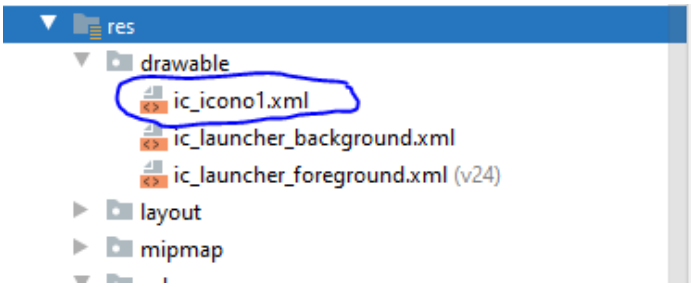
Lanzamos la herramienta Vector Asset



Escogemos el archivo svg descargado. Indicamos el tamaño del gráfico en dp's y el nombre del mismo.



Indicar dónde se va a guardar el gráfico y con qué archivo (fijarse que genera un archivo XML).



El archivo XML generado en la carpeta '*drawable*'.

Una ImageButton con el icono asociado.

1.3. Métodos a conocer en el manejo de los Button´s y variantes

1.3.1.Todos los Buttons: Button, FloatActionButton, ToogleButton, Switch

- Referenciar a un button con el método findViewById.
- Recuperar el contenido del texto.
- Cambiar el contenido del texto, pudiendo emplear recursos guardados en **/res/values**.
- Modificar propiedades básicas, como color, tamaño, visibilidad,...(y los métodos getZZZZZ para obtener sus valores).

Ejemplos con un Button, pero aplicables a cualquier tipo de Button:

```
Button btn = findViewById(R.id.btnBoton); // Referenciamos a un botón que esta en el Layout da Activity

String cadeaet = btn.getText().toString(); // Fijarse que getText() devuelve un objeto Editable, no un String.

btn.setText("Cambiamos elvtexto directamente");// Cambiamos el contenido por una cadena concreta
String textoRes = getResources().getString(R.string.app_name);
btn.setText(textoRes + " texto que viene de la BD");// Si queremos concatenar texto de una base de datos con un texto que pueda ser traducido

btn.setTextSize(20);                                    // Ajustar tamaño del texto

btn.setTextColor(Color.BLUE);                           // Cambia el color
btn.setVisibility(View.VISIBLE);                        // Las constantes aparecen en Android Studio al teclear. GONE hace que no ocupe espacio en el Layout.
```

1.3.2.Switch - ToogleButton

- Obtener el estado del botón (ON / OFF).
- Cambiar el estado del botón (ON / OFF)
- Cambiar el texto del SwitchButton
- Capturar el evento que se produce cuando hay un cambio de estado (diferente a hacer un click, podemos hacer click sobre el mismo estado muchas veces y no hay cambio de estado).

```
ToggleButton tbtn = findViewById(R.id.tbtnToogleBoton); // Referenciamos a un botón que este en el Layout de la Activity
Toast.makeText(this,String.valueOf(tbtn.isChecked()),Toast.LENGTH_LONG).show();
// isChecked() devuelve true/false

tbtn.setChecked(true); // Cambiamos el estado por programación

tbtn.setText("NUEVO TEXTO TOGGLE");                 // Cambiamos el texto
```

```
tbtn.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {   // Al cambiar el estado ON-OFF llama al método onCheckedChanged
@Override
public void onCheckedChanged(CompoundButton compoundButton, boolean b) {     // Al ser una interface anónima, compoundButton es tbtn. Se escogemos la opción de gestionar los eventos en la Activity, empleando dicho objeto.getId() podemos identificar el botón
Toast.makeText(getApplicationContext(),String.valueOf(b),Toast.LENGTH_LONG).show();  // El parámetro b indica el nuevo estado del ToggleButton
}
});
```

Para el Switch es exactamente igual.

```
Switch sw = findViewById(R.id.swbSwitchButton);
// Referenciamos a un botón que este en el Layout da Activity
        Toast.makeText(this,String.valueOf(sw.isChecked()),Toast.LENGTH_LONG).show();
// isChecked() devuelve true/false

        sw.setChecked(true);
// Cambiamos el estado por programación

        sw.setText("NUEVO TEXTO SWITCH");
// Nuevo texto del switch

        sw.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
// Al cambiar el estado ON-OFF llama al método onCheckedChanged
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
// Al ser una interface anónima, compoundButton es sw. Se escogemos la opción de gestionar los eventos en la Activity, empleando dicho objeto.getId() podemos identificar el botón

Toast.makeText(getApplicationContext(),String.valueOf(b),Toast.LENGTH_LONG).show(); // El parámetro b indica el nuevo estado del ToggleButton
}
});
```