Tema 5. Interfaz de Usuario I

Descargar estos apuntes

Índice

ViewBinding

Si ya eres conocedor de Android, conocerás findViewById, que permite vincular las vistas de la aplicación con las clases, y así poder acceder a determinadas propiedades de los widgets de las vistas.

Desde la versión 3.6 de Android Studio, es posible utilizar View Binding para enlazar los elementos de los layouts con las clases.

Si estamos trabajando con una versión de Android Studio superior a la 4.0 la configuración del archico `build.gradle` a nivel de `Module:app` es la siguiente:

```
    android {
        ...
        viewBinding {
            enabled = true
        }
    }
```

Una vez habilitada la vinculación de vista para un proyecto, por cada archivo xml se generará una clase de vinculación al mismo, que será utilizado para hacer refrencias a las vistas del mismo.

si nuestro archivo se llamma `activity_secundaria.xml` la clase de vinculación generada se llamará ActivitySecundariaBinding.

Para poder utilizar la vinculación de vistas hay que hacer lo siguiente en el método `onCreate()` de la actividad:

```
private lateinit var binding: ActivitySecundariaBinding

    override fun onCreate(savedInstanceState: Bundle) {
        super.onCreate(savedInstanceState)
        binding = ActivitySecundariaBinding.inflate(layoutInflater)
        val view = binding.root
        setContentView(view)
    }
```

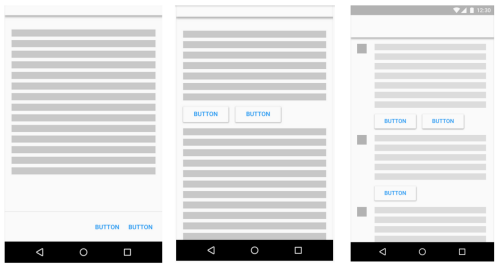
A partr de ahora podemos hacer referencia a las vistas de nuestro xml. Lo veremos a continuación tal y como vayamos viendo distintos componentes.

Botones

Los botones forman una parte funcional muy importante en cualquier aplicación. Existen tres tipos estándar de botonaes: `Floating Action Button` (botón circular con una acción muy concreta en nuestra aplicación), `Raised Button` (botón con relieve con efecto de pulsación) y `Flat Button` (botón sin relieve ni efecto de pulsación).

La elección de un botón u otro va a depender de la importancia que se le quiera dar al mismo y de la disposición del mismo.

Se recomienda utilizar los `Flat Button` en los cuadros de diálogo, en botones en línea al finalizar la pantalla o en botones que son persistentes o que estarán siempre disponibles en nuestra aplicación. Los `Raise Buttob` se utilizan más cuando están en medio de dos vistas de la interfaz.



> 📌 Aquí tenemos un enlace donde se encuentra información sobre este componente que ofrece Material Design

Raised Buttons

Flat Buttons

Floating Action Buttons

ImageView

Patrones MD

Text Fields

TextView

EditText

Floating Labels

Filled text fields
Outlines text fields

AutoComplete TextView

CheckBox

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

> 📌 Aquí tenemos un enlace donde se encuentra información sobre la gestión de colores que ofrece Material Design

RadioButton

Switches

Sliders

Chips

Toast

SnackBar

SnackBar con acción

Para poder comprender como funciona este layout, primero debemos de conocer como se integran las vistas o componentes en los layouts. Veamos la siguiente imagen:

Cada lateral de un componente viene referenciado por un nombre específico que permite hacer referencia al mismo. `Top` es la parte superior, `Bottoom` es la parte inferior, `Start` es la parte izquierda y `End` es la parte derecha.

Para poder añadir componentes debemos restringir su posición definiendo unos anclajes basados en los laterales de cada componente. Son necesarios al menos dos anclajes.

Veamos un ejemplo práctico. Vamos a dieñar la siguiente interfaz:

Cremaos un proyecto con el nombre **ejemploConstraintLayout** e integraremos los siguientes componentes:

Como vemos insertamos en el contenedor una `ImageView`, dos `TextView` una `RatingBar` y dos `Button`. No os preocupéis porque más adelante explicaremos con más detalle estos componentes, ahora solamene queremos utilizarlos para explicar `ConstraintLayout`.

El archivo XML correspondiente es el siguiente:

```
<?xml version="1.0" encoding="utf-8"? >
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.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"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imagen"
        android:layout_width="match_parent"
        android:layout_height="720px"
        android:layout_marginTop="0dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/sanfrancisco" />

    <TextView
        android:id="@+id/nombreCiudad"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/imagen"
        android:text="@string/name_city"
        android:textAlignment="center"
        android:textSize="15dp"
        android:textColor="@color/black"/>

    <TextView
        android:id="@+id/valorar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/nombreCiudad"
        android:text="@string/name_rate"
        android:textAlignment="center"
        android:textSize="30dp"
        android:textColor="@color/black"
        android:layout_marginTop="10dp"/>

    <RatingBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/valorar"
        app:layout_constraintEnd_toEndOf="parent"
        android:numStars="5"/>

    <Button
        android:id="@+id/buttonVotar"
        android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="@string/text_Votar"
            app:layout_constraintBottom_toBottomOf="parent"
54          app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toStartOf="@id/buttonSalir"
56          android:layout_marginBottom="45dp"/>

        <Button
            android:id="@+id/buttonSalir"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/text_Salir"
64          app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
66          app:layout_constraintStart_toEndOf="@id/buttonVotar"
            android:layout_marginBottom="45dp"/>
    </androidx.constraintlayout.widget.ConstraintLayout>
```

📌 Aclaraciones:

- **ImageView** líneas 10..16:
  Línea 10 con la propiedad **`android:id="@+id/…"`** damos un nombre a la vista para posteriormente poder hacer referencia a ella.
  Línea 11 con la propiedad **`android:width`** indicamos el ancho de esta view, el valor **`match_parent`** ajusta el ancho al ancho disponible en el contenedor que recibe la view, el valor **`wrap_content`** ajusta el ancho al tamaño de la imagen.
  Línea 12 con la propiedad **`android:height`** se define la altura de la view, la definimos en este caso indicando la altura en dp para que puedan reescalarse correctamente en diferentes dispositivos.
  Línea 13 la propiedad **`android:marginTop`** permite definir el margen superior respecto al contenedor principal, como queremos que esté lo más arriba posible, le quitamos el margen superior.
  Ahora viene la parte que más nos interesa, los anclajes de nuestra imagen.
  Línea 14 con la propiedad **`constraintStart_toStartOf:`** especificamos que el Start de nuestro componente estará anclado al comienzo de su padre, con eso conseguimos mantenerlo a la izquierda.
  Línea 15 con la propiedad **`constraintTop_toTopOf:`** especificamos que la parte superior de nuestro componente estará anclada a la parte superior del padre.
  Línea 16 por último, necesitamos cargar la imagen en la view, utilizamos para ello la propiedad **`app:srcCompat:`** .
- **TextView** líneas 22..27
  En la línea 22 y 23 se definen los anclajes, como se puede apreciar, el texto está de izquierda a derecha así pues, anclaremos el Start al Start del padre, igual que se hizo en

la imagen. La parte superior del texto, tendremos que anclarla a la parte inferior de la imagen para que nos quede justo debajo.
En la propiedad `android:text:` referenciamos un recurso string donde tenemos el texto que queremos que aparezaca en el **TextView**.
Con `android:textAlignment:` centramos el texto independientemente de lo grande o pequeña que sea la pantalla del terminal.

- El siguiente **TextView** tiene que ir anclado a su izquierda con el padre y estar por debajo al texto que acabamos de colocar previamente, líneas 33 y 34.
  La única propiedad que no hemos visto anteriormente es `android:marginTop` línea 39 y se utiliza para dejar un margen superior respecto a otros componentes, en este caso, 10dp.
- El siguiente componente es una **RatingBar** tiene que ir anclado a su izquierda con el padre y estar por debajo al **TextView** que tiene como id **valorar** líneas 44..46.
  Con la propiedad `android:numStars:` en la línea 47 elegimos el número de estrellas que deseamos que aparezcan en la **RatingBar**.
  ✋ Con la propiedad `android:stepSize: 0.5` nos permitiría seleccionar media estrella(probar vosotros)
- Veamos el código para incorporar los dos **Button** a la vista.
  Buscaremos que ambos botones estén a la misma altura y lo conseguiremos dando un margen inferior. Además, los colocaremos anclados entre ellos en cuanto al eje horizontal se refiere y anclados a la vez al padre, en este caso cada uno en el lateral que corresponda y respecto a la parte inferior para poder ajustarlo más cómodamente.

Descartar SnackBar

📌 ünicamente hemos modificado el `androidx.core.widget.NestedScrollView` añadiendo los elementos que se ven. El resto del diseño es idéntico al anterior.

``

✋ EjercicioPropuestoCardView

📌 Aclaraciones: