Juan Menduiña

Trabajo Práctico N° 2

Ejercicio 1.

Crear un nuevo proyecto en Android Studio con una Actividad vacía y modificar el contenido del archivo .xml de la actividad creada (en res/layouts), colocar el siguiente código en él y probar en el dispositivo o emulador:

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

  <TextView
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:text="Hola Mundo" />
</RelativeLayout>
```

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:text="Hola Mundo" />
</RelativeLayout>
```

Ejercicio 2.

Investigar la utilidad de los atributos "android:layout_width" y "android:layout_height".

Los atributos *android:layout_width* y *android:layout_height* son fundamentales en *Android XML layouts*, ya que determinan el ancho y alto, respectivamente, de una vista o contenedor.

(a) *¿Qué sucede si se omite alguno de los dos en el elemento RelativeLayout?*

Si se omite uno de estos atributos en un *layout* como *RelativeLayout*, el compilador lanzará un error y la aplicación no se podrá compilar correctamente. *Android* requiere, explícitamente, que se definan ambos atributos (*layout_width* y *layout_height*) para cualquier vista o *layout*.

(b) *¿Qué sucede si, en lugar de "match_parent", se establece como valor "wrap_content"?*

Si, en lugar de *"match_parent"*, se establece como valor *"wrap_content"*, lo que sucede es que la vista se ajustará al contenido que tenga dentro, en lugar de ocupar todo el espacio disponible en su contenedor padre.

Ejercicio 3.

Añadir un color de fondo al TextView y modificar el color de las letras.

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:text="Hola Mundo"
        android:textColor="#FFFFFF"
        android:background="#FF0000" />
</RelativeLayout>
```

Ejercicio 4.

Modificar el ancho del TextView para que siempre se adapte al ancho del contenedor. Luego, centrar el texto horizontalmente.

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:text="Hola Mundo"
        android:textColor="#FFFFFF"
        android:background="#FF0000"
        android:gravity="center_horizontal" />
</RelativeLayout>
```

Ejercicio 5.

Cambiar el tamaño de la tipografía (utilizar la unidad de medida sp).

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:text="Hola Mundo"
        android:textColor="#FFFFFF"
        android:textSize="24sp"
        android:background="#FF0000"
        android:gravity="center_horizontal" />
</RelativeLayout>
```

Ejercicio 6.

Crear un TextView debajo del "Hola Mundo" con el texto "Aquí Estoy". ¿Qué ocurrió? ¿Por qué?

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:text="Hola Mundo"
        android:textColor="#FFFFFF"
        android:textSize="24sp"
        android:background="#FF0000"
        android:gravity="center_horizontal" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:text="Aquí Estoy"
        android:textColor="#FFFFFF"
        android:textSize="20sp"
        android:background="#0000FF"
        android:gravity="center_horizontal" />
</RelativeLayout>
```

Lo que ocurre es que el segundo *TextView* "pisa" al primer *TextView* porque no se especifica dónde colocar el segundo *TextView* respecto del primero.

Ejercicio 7.

Reemplazar el código creado RelativeLayout por un LinearLayout de la siguiente manera y ver el resultado. ¿Para qué sirve especificar la orientación del LinearLayout?

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

En *activity_main.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:text="Hola Mundo"
        android:textColor="#FFFFFF"
        android:textSize="24sp"
        android:background="#FF0000"
        android:gravity="center_horizontal" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Aquí Estoy"
        android:textColor="#FFFFFF"
        android:textSize="20sp"
        android:background="#0000FF"
        android:gravity="center_horizontal" />
</LinearLayout>
```

Especificar la orientación del *LinearLayout* sirve para definir cómo se van a organizar los elementos hijos: uno debajo del otro (vertical) o uno al lado del otro (horizontal).

Ejercicio 8.

Resolver el problema visual que surgió en el Ejercicio 6 sin utilizar LinearLayout. Nota: Utilizar la propiedad layout_below.

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:text="Hola Mundo"
        android:textColor="#FFFFFF"
        android:textSize="24sp"
        android:background="#FF0000"
        android:gravity="center_horizontal" />
    <TextView
        android:id="@+id/textView2"
        android:layout_below="@id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Aquí Estoy"
        android:textColor="#FFFFFF"
        android:textSize="20sp"
        android:background="#0000FF"
        android:gravity="center_horizontal" />
</RelativeLayout>
```

Ejercicio 9.

Generar una disposición de componentes visuales como la siguiente:

En *activity_main.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Práctica 2"
        android:textColor="#FFFFFF"
        android:textSize="30sp"
        android:textStyle="bold"
        android:background="#3F51B5"
        android:gravity="center_vertical"
        android:padding="16dp" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Arriba"
        android:textSize="20sp" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Abajo"
        android:textSize="20sp" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Izquierda"
            android:textSize="20sp" />
        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
```

```
            android:layout_weight="1"
            android:text="Derecha"
            android:textSize="20sp" />
    </LinearLayout>
</LinearLayout>
```

Ejercicio 10.

A partir de la experiencia obtenida con el uso de Layouts.

(a) *¿Por qué son importantes los Layouts?*

Los *Layouts* son importantes porque:

- Organizan, visualmente, los elementos de la interfaz (*TextViews*, *Buttons*, etc.).
- Permiten crear interfaces que se adaptan, automáticamente, a diferentes tamaños de pantalla, resoluciones y orientaciones.
- Separan la lógica visual de la lógica del programa (buena práctica de diseño).
- Hacen que la *app* sea responsiva y mantenible.

Sin ellos, habría que posicionar todo a mano y eso no es escalable.

(b) *¿Cuál hubiera sido la problemática de definir la posición/tamaño de las componentes visuales a través de coordenadas absolutas?*

Usar coordenadas absolutas sería problemático porque:

- No se adapta a distintos tamaños de pantalla o resoluciones.
- Si el usuario cambia de orientación (por ejemplo, de vertical a horizontal), la UI se rompe.
- Es difícil de mantener y escalar, ya que cualquier cambio requiere recalcular todo.
- No considera accesibilidad ni escalado de texto (por ejemplo, si el usuario aumenta el tamaño de fuente).

En resumen, la interfaz sería rígida y frágil.

(c) *¿Esta problemática sólo existe en dispositivos móviles?*

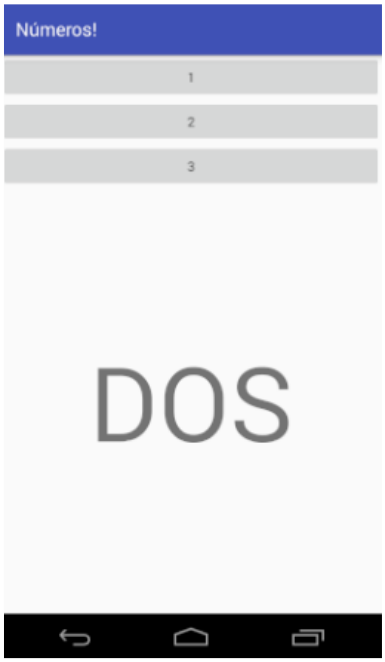
No, esta problemática existe en cualquier sistema con múltiples resoluciones. Por ejemplo:

- En computadoras: Distintas resoluciones de monitor y ventanas redimensionables generan el mismo problema.
- En aplicaciones *web*: Es lo que motivó el diseño *responsive*.
- En TVs, *tablets*, relojes inteligentes, etc.

Por eso, los sistemas modernos (*Android*, iOS, HTML/CSS, etc.) usan *Layouts* o sistemas de diseño responsivo para evitar esas limitaciones.

Ejercicio 11.

Implementar una aplicación con un layout como el siguiente:



Cuando se haga click sobre uno de los botones, la aplicación deberá mostrar en pantalla de forma centrada y con una tipografía más grande la representación en letras mayúsculas del número sobre el cual se hizo click. El ancho de los botones deberá adaptarse al ancho de la pantalla. El espacio en donde se visualiza el texto deberá adaptarse al espacio disponible y el texto se mostrará centrado horizontal y verticalmente.

En *activity_main.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Números!"
        android:textColor="#FFFFFF"
        android:textSize="20sp"
        android:textStyle="bold"
        android:background="#3F51B5"
        android:gravity="center_vertical"
```

```
        android:padding="16dp"
        android:layout_marginBottom="6dp" />
    <Button
        android:id="@+id/btn1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="1"
        android:textColor="#000000"
        android:background="#888888"
        android:layout_marginBottom="12dp" />
    <Button
        android:id="@+id/btn2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="2"
        android:textColor="#000000"
        android:background="#888888"
        android:layout_marginBottom="12dp" />
    <Button
        android:id="@+id/btn3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="3"
        android:textColor="#000000"
        android:background="#888888" />
    <TextView
        android:id="@+id/txtViewResult"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:textColor="#888888"
        android:textSize="100sp"
        android:gravity="center" />
</LinearLayout>
```

(a) *Implementar utilizando un manejador de click para cada botón.*

En *MainActivity.kt*:

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

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        val txtViewResult =
findViewById<TextView?>(R.id.txtViewResult)
        val btn1 = findViewById<Button?>(R.id.btn1)
        val btn2 = findViewById<Button?>(R.id.btn2)
        val btn3 = findViewById<Button?>(R.id.btn3)
```

```
        btn1.setOnClickListener { txtViewResult.setText("UNO") }
        btn2.setOnClickListener { txtViewResult.setText("DOS") }
        btn3.setOnClickListener { txtViewResult.setText("TRES") }
    }
}
```

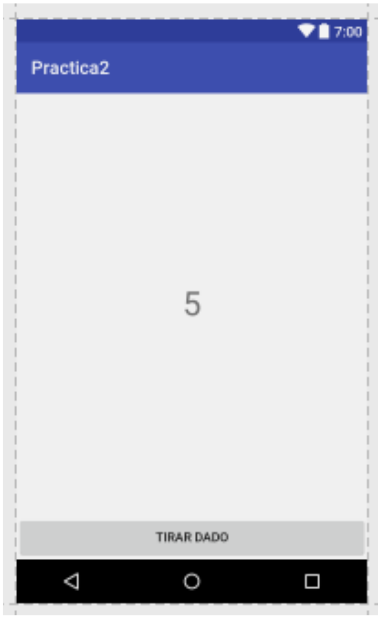
(b) *Implementar utilizando un único manejador de click para todos los botones.*

En *MainActivity.kt*:

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

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        val txtViewResult =
findViewById<TextView?>(R.id.txtViewResult)
        val btn1 = findViewById<Button?>(R.id.btn1)
        val btn2 = findViewById<Button?>(R.id.btn2)
        val btn3 = findViewById<Button?>(R.id.btn3)
        val clickListener = View.OnClickListener { v ->
            when (v?.id) {
                R.id.btn1 -> txtViewResult.text = "UNO"
                R.id.btn2 -> txtViewResult.text = "DOS"
                R.id.btn3 -> txtViewResult.text = "TRES"
            }
        }
        btn1.setOnClickListener(clickListener)
        btn2.setOnClickListener(clickListener)
        btn3.setOnClickListener(clickListener)
    }
}
```

Ejercicio 12.

Implementar una aplicación con un layout como el siguiente:



Cada vez que se hace click en el botón "Tirar Dado", la aplicación deberá generar, aleatoriamente, un valor entre 1 y 6, mostrándolo en pantalla. Para la generación de números aleatorios, investigar el uso de la clase java.util.Random.

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Practica2"
        android:textColor="#FFFFFF"
        android:textSize="20sp"
        android:textStyle="bold"
        android:background="#3F51B5"
        android:gravity="center_vertical"
        android:padding="16dp" />
    <TextView
        android:id="@+id/txtViewResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#888888"
        android:textSize="50sp"
```

```
        android:layout_centerInParent="true" />
    <Button
        android:id="@+id/btnTirarDado"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="TIRAR DADO"
        android:textColor="#000000"
        android:background="#888888"
        android:layout_alignParentBottom="true" />
</RelativeLayout>
```

En *MainActivity.kt*:

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

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        val txtViewResult =
findViewById<TextView?>(R.id.txtViewResult)
        val btnTirarDado = findViewById<Button?>(R.id.btnTirarDado)
        btnTirarDado.setOnClickListener {
            val num = (1..6).random()
            txtViewResult.setText(num.toString())
        }
    }
}
```

Ejercicio 13.

Usando un layout similar al desarrollado en el Ejercicio 12, agregar una imagen usando el componente <ImageView>. La imagen deberá incluirse en el directorio /proyecto/app/src/main/res/drawable/.

Nota: Se puede usar el siguiente código:

```
<ImageView
        android:src="@drawable/imagen"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:scaleType="center" />
```

En *activity_main.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:animateLayoutChanges="true"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/title"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Práctica 2"
        android:textColor="#FFFFFF"
        android:textSize="20sp"
        android:textStyle="bold"
        android:background="#800000"
        android:gravity="center_vertical"
        android:padding="16dp"
        android:layout_marginBottom="6dp" />
    <TextView
        android:id="@+id/txtViewImage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Facultad de Informática"
        android:textColor="#888888"
        android:textSize="30sp"
        android:textStyle="bold"
        android:gravity="center"
        android:padding="16dp"
        android:layout_marginBottom="6dp" />
    <ImageView
        android:id="@+id/imgViewFacultad"
        android:src="@drawable/facultad_informatica"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:scaleType="centerCrop"
        android:layout_marginBottom="6dp" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/btnOcultar"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="OCULTAR"
            android:textColor="#000000"
            android:textSize="20sp"
            android:background="#888888"
            android:layout_marginStart="12dp" />
        <Button
            android:id="@+id/btnMostrar"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="MOSTRAR"
            android:textColor="#000000"
            android:textSize="20sp"
            android:background="#888888"
            android:layout_marginStart="12dp"
            android:layout_marginEnd="12dp" />
    </LinearLayout>
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
```

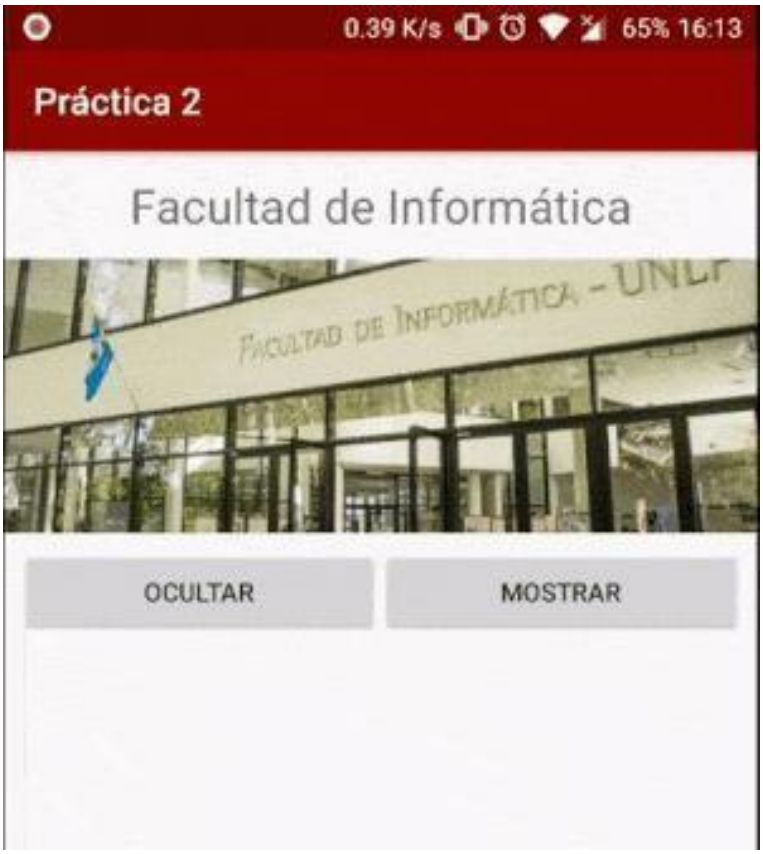
```
        setContentView(R.layout.activity_main)

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        val imgViewFacultad =
findViewById<ImageView?>(R.id.imgViewFacultad)
        val btnOcultar = findViewById<Button?>(R.id.btnOcultar)
        val btnMostrar = findViewById<Button?>(R.id.btnMostrar)
        btnOcultar.setOnClickListener {
            imgViewFacultad.visibility = View.GONE
            //imageViewFacultad.visibility = View.INVISIBLE
        }
        btnMostrar.setOnClickListener {
            imgViewFacultad.visibility = View.VISIBLE
        }
    }
}
```

(a) *¿Qué sucede al cambiar el valor del atributo scaleType por "centerCrop"? ¿Y "fitXY"?*

Lo que sucede al cambiar el valor del atributo *scaleType* por:

- *"centerCrop"* es que la imagen ya no se muestra en su tamaño original (como sucedía con "*center"*), sino que, ahora, se muestra escalada proporcionalmente para que llene todo el *ImageView*, manteniendo su relación de aspecto.
- *"fitXY"* es que la imagen ya no se muestra en su tamaño original (como sucedía con "*center"*), sino que, ahora, se muestra para que llene todo el *ImageView*, sin mantener su relación de aspecto.

(b) *Al hacer click sobre los botones, deberá ocultarse o mostrarse la imagen usando el método setVisiblity(View.GONE) y setVisibility(View.VISIBLE) sobre la imagen.*



¿Qué cambia si, en vez de ocultar la imagen usando View.GONE, se usa View.INVISIBLE?

Lo que cambia si, en vez de ocultar la imagen usando *View.Gone*, se usa *View.INVISIBLE* es que, cuando se oculta la imagen, ésta desaparece pero su espacio ya no desaparece, sino que, ahora, sigue reservado en pantalla.

(c) *Agregar el atributo android:animateLayoutChanges=”true” en el LinearLayout que contiene a los elementos. ¿Qué diferencia hay al cambiar la visibilidad de la imagen?*

Al agregar el atributo *android:animateLayoutChanges=“true”* en el *LinearLayout* que contiene a los elementos, la diferencia que hay es que cuando se oculta o se muestra la imagen el cambio ya no es inmediato y brusco, es decir, la imagen no desaparece o aparece de golpe, sino que, ahora, el cambio se anima suavemente.