Juan Menduiña

Trabajo Práctico N° 3

Ejercicio 1.

Modificar el Ejercicio 12 de la Práctica 2 para que, al girar el dispositivo, se mantenga el número mostrado en pantalla. Utilizar los métodos onSaveInstanceState/onRestoreInstanceState.

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private var num = 0
    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(object :
View.OnClickListener {
            override fun onClick(v: View?) {
                num = (1..6).random()
                txtViewResult!!.setText(num.toString())
            }
        })
        if (savedInstanceState != null) {
            num = savedInstanceState.getInt("num")
            txtViewResult.text = num.toString()
        }
    }
    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putInt("num", num)
    }
}
```

Ejercicio 2.

Implementar la siguiente Activity con cuatro imágenes y un título, usando un ScrollView.

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    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">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <TextView
            android:id="@+id/title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Práctica 3"
            android:textColor="#FFFFFF"
            android:textSize="20sp"
```

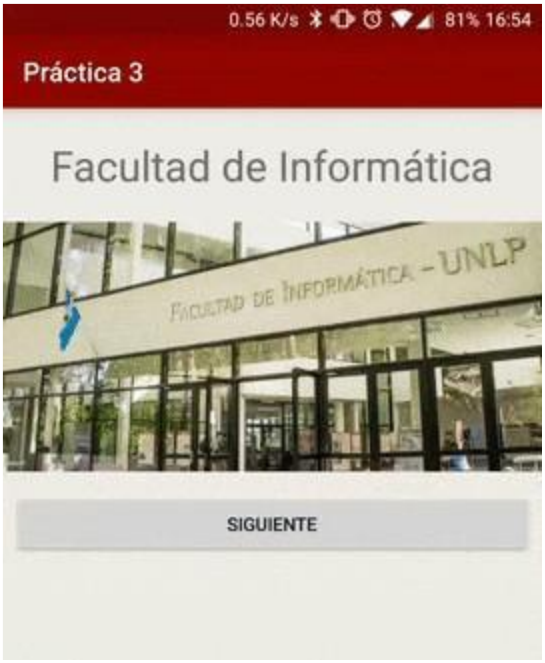
```
            android:textStyle="bold"
            android:background="#800000"
            android:gravity="center_vertical"
            android:padding="16dp"
            android:layout_marginBottom="6dp" />
        <TextView
            android:id="@+id/txtViewImage"
            android:layout_width="match_parent"
            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/imgViewFacultad1"
            android:src="@drawable/facultad_informatica_1"
            android:layout_width="match_parent"
            android:layout_height="190dp"
            android:scaleType="centerCrop"
            android:layout_marginBottom="6dp" />
        <ImageView
            android:id="@+id/imgViewFacultad2"
            android:src="@drawable/facultad_informatica_2"
            android:layout_width="match_parent"
            android:layout_height="190dp"
            android:scaleType="centerCrop"
            android:layout_marginBottom="6dp" />
        <ImageView
            android:id="@+id/imgViewFacultad3"
            android:src="@drawable/facultad_informatica_3"
            android:layout_width="match_parent"
            android:layout_height="190dp"
            android:scaleType="centerCrop" />
    </LinearLayout>
</ScrollView>
```

Ejercicio 3.

Modificar el ejercicio anterior para que muestre una única imagen. Cada vez que se clickea el botón "Siguiente", se deberá reemplazar la imagen. Investigar el uso del mensaje setImageResource de la clase ImageView. Al girar la pantalla, no debe cambiarse la imagen y el layout debe visualizarse correctamente.

Nota: Se puede almacenar las referencias a las imágenes en un arreglo de la siguiente manera:

```
private int[] imagenes = {R.drawable.portada, R.drawable.interior, R.drawable.foto3};
```



En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    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">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:id="@+id/title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Práctica 3"
            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:layout_width="match_parent"
            android:layout_height="250dp"
            android:scaleType="centerCrop"
            android:layout_marginBottom="12dp" />
        <Button
            android:id="@+id/btnSiguiente"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="SIGUIENTE"
            android:textColor="#000000"
            android:textSize="20sp"
            android:background="#888888"
            android:layout_marginStart="12dp"
            android:layout_marginEnd="12dp" />
    </LinearLayout>
</ScrollView>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private val imagenes = arrayOf(
        R.drawable.facultad_informatica_1,
        R.drawable.facultad_informatica_2,
        R.drawable.facultad_informatica_3
    )
    override fun onCreate(savedInstanceState: Bundle?) {
        val hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY)
        if (hour in 10..22) {
            setTheme(R.style.Theme_Day)
        }
        else {
            setTheme(R.style.Theme_Night)
        }
        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)
```

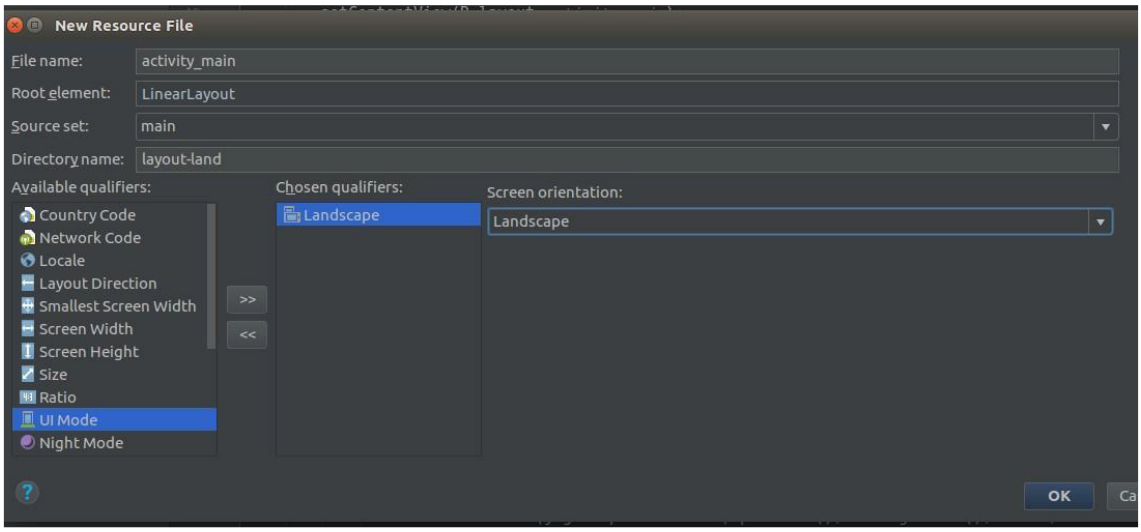
```
            insets
        }
        val imgViewFacultad =
findViewById<ImageView?>(R.id.imgViewFacultad)
        val btnSiguiente = findViewById<Button?>(R.id.btnSiguiente)
        var index = 0
        imgViewFacultad?.setImageResource(imagenes[index])
        btnSiguiente?.setOnClickListener {
            index = (index + 1) % imagenes.size
            imgViewFacultad?.setImageResource(imagenes[index])
        }
    }
}
```

Ejercicio 4.

Modificar el ejercicio anterior de manera que, al girar la pantalla, se modifique el layout de la siguiente forma:

Implementar la grilla de botones usando GridLayout y un XML diferenciado según la orientación.

Nota: Para incluir un layout dependiente de la orientación, desde Android Studio, se debe hacer click derecho en el directorio layout ⟶ New ⟶ Layout resource file. Se deberá tener el mismo nombre de archivo que el layout que se usó para la versión vertical. Luego, se elige "Orientation" en el cuadro de "Available qualifiers" y, luego, la orientación Landscape:



Por defecto, cuando la aplicación esté en orientación Portrait, usará el layout que se había definido originalmente y, ahora, la orientación Landscape usará este nuevo layout.

En *activity_main.xml (land)*:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
```

```
    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">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <TextView
            android:id="@+id/title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Práctica 3"
            android:textColor="#FFFFFF"
            android:textSize="20sp"
            android:textStyle="bold"
            android:background="#800000"
            android:gravity="center_vertical"
            android:padding="16dp"
            android:layout_marginBottom="6dp" />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="horizontal">
            <ImageView
                android:id="@+id/imgViewFacultad"
                android:layout_width="600dp"
                android:layout_height="300dp"
                android:scaleType="centerCrop" />
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical">
                <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="4dp"
                    android:layout_marginBottom="20dp" />
                <GridLayout
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:columnCount="2"
                    android:rowCount="2"
                    android:alignmentMode="alignMargins"
                    android:layout_gravity="center"
                    android:padding="4dp">
                    <Button
                        android:id="@+id/btn1"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="IMAGEN 1"
                        android:textColor="#000000"
```

```
                        android:textSize="16sp"
                        android:background="#888888"
                        android:layout_margin="4dp" />
                    <Button
                        android:id="@+id/btn2"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="IMAGEN 2"
                        android:textColor="#000000"
                        android:textSize="16sp"
                        android:background="#888888"
                        android:layout_margin="4dp" />
                    <Button
                        android:id="@+id/btn3"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="IMAGEN 3"
                        android:textColor="#000000"
                        android:textSize="16sp"
                        android:background="#888888"
                        android:layout_margin="4dp" />
                    <Button
                        android:id="@+id/btn4"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="IMAGEN 4"
                        android:textColor="#000000"
                        android:textSize="16sp"
                        android:background="#888888"
                        android:layout_margin="4dp" />
                </GridLayout>
            </LinearLayout>
        </LinearLayout>
    </LinearLayout>
</ScrollView>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private val imagenes = arrayOf(
        R.drawable.facultad_informatica_1,
        R.drawable.facultad_informatica_2,
        R.drawable.facultad_informatica_3
    )
    private var index = 0
    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 btnSiguiente = findViewById<Button?>(R.id.btnSiguiente)
```

```
        val btn1 = findViewById<Button?>(R.id.btn1)
        val btn2 = findViewById<Button?>(R.id.btn2)
        val btn3 = findViewById<Button?>(R.id.btn3)
        if (savedInstanceState != null) {
            index = savedInstanceState.getInt("index")
        }
        imgViewFacultad?.setImageResource(imagenes[index])
        btnSiguiente?.setOnClickListener {
            index = (index + 1) % imagenes.size
            imgViewFacultad?.setImageResource(imagenes[index])
        }
        btn1?.setOnClickListener {
            index = 0
            imgViewFacultad?.setImageResource(imagenes[index])
        }
        btn2?.setOnClickListener {
            index = 1
            imgViewFacultad?.setImageResource(imagenes[index])
        }
        btn3?.setOnClickListener {
            index = 2
            imgViewFacultad?.setImageResource(imagenes[index])
        }
    }
    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putInt("index", index);
    }
}
```

Ejercicio 5.

Investigar sobre los recursos de elementos de diseño de Android (Recursos drawable).

(a) *¿Para qué sirven?*

Los recursos *drawable* en *Android* sirven para definir elementos gráficos que se pueden mostrar en la interfaz de usuario, como imágenes, formas vectoriales, bordes y fondos. Estos recursos permiten separar los elementos visuales del código de la aplicación, facilitando su reutilización y la adaptación a diferentes resoluciones de pantalla, densidades o temas.

(b) *¿Qué tipos de elementos de diseño están disponibles?*

Los tipos de elementos de diseño que están disponibles son:

- Imágenes *bitmap*: Archivos .png, .jpg, .webp, etc.
- Vectores: Archivos .xml con formato *VectorDrawable* (como ic_launcher.xml).
- *Shapes*: Archivos .xml que definen formas (rectángulos, óvalos, etc.) con bordes, colores, gradientes.
- *Selectors*: Archivos .xml que permiten definir diferentes estados de un componente gráfico (por ejemplo, botón presionado/no presionado).
- *State lists*: Similares a *selectors*, pero para controlar cambios visuales según el estado del componente.
- *Layer lists*: Combinación de múltiples *drawables* en capas.
- *Nine-patch* (.9.png): Imágenes con áreas elásticas para adaptarse al contenido.

(c) *¿Qué tipo de elemento de diseño se utilizaría para mostrar una imagen?*

El tipo de elemento de diseño que se utilizaría para mostrar una imagen es *bitmap*.

(d) *¿Qué tipo de elemento de diseño se utilizaría para mostrar un rectángulo con colores?*

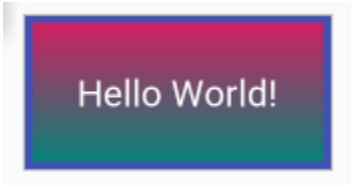
El tipo de elemento de diseño que se utilizaría para mostrar un rectángulo con colores es *shape*.

(e) *¿Qué tipo de elemento de diseño se utilizaría para mostrar los estados "presionado" y "suelto" de un botón?*

El tipo de elemento de diseño que se utilizaría para mostrar los estados “presionado” y “suelto” de un botón es *selector*.

Ejercicio 6.

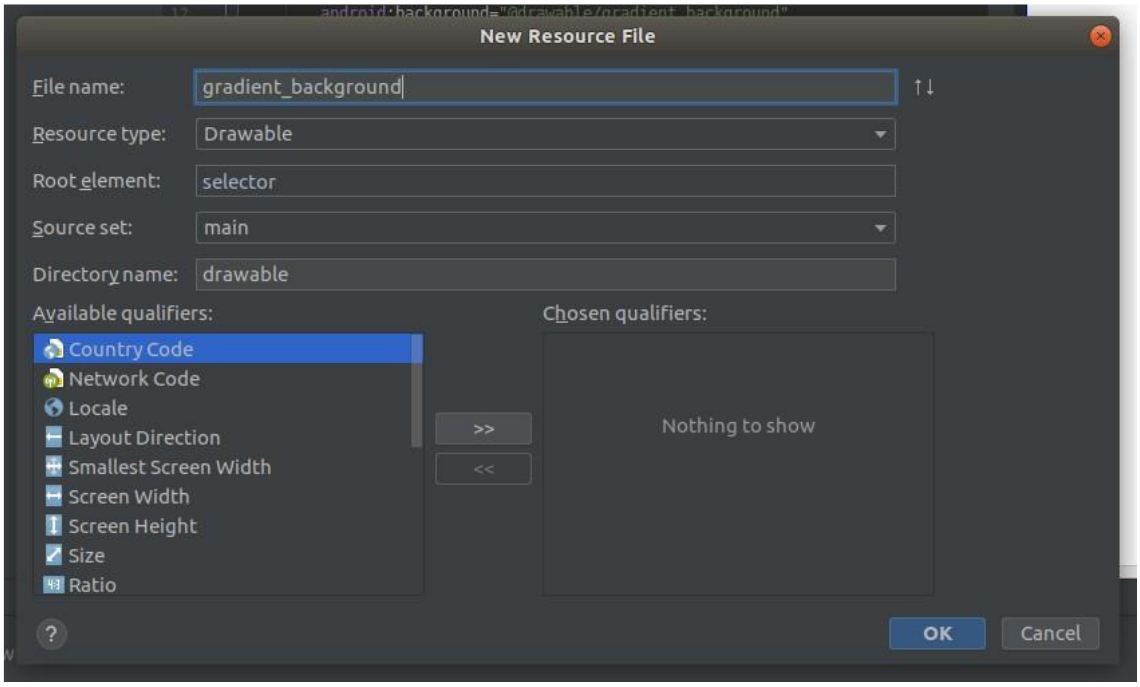
Utilizar un recurso drawable para definir el fondo de un TextView que se visualice de la siguiente manera:



Para ello, se deberá:

(a) *Generar un recurso drawable denominado a modo de ejemplo: gradient_background.*

Click derecho ⟶ New ⟶ Android Resource File
Notar que Resource type debe ser Drawable.



(b) *En el archivo .xml generado, se deberá definir un diseño gradiente vertical lineal entre dos colores. Al mismo tiempo, se debe definir un borde de 3dp de ancho con un color diferente al del gradiente. Se deberá utilizar un Shape. Se puede obtener información sobre la sintaxis a utilizar en: https://developer.android.com/guide/topics/resources/drawable-resource?hl=es-419#Shape.*

(c) *Una vez definido el recurso drawable, se puede asignar a un TextView, simplemente, estableciendo el atributo android:background de la siguiente manera:* <TextView android:background="@drawable/gradient_background".

En *gradient_background.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <stroke
        android:width="3dp"
        android:color="#3F51B5" />
    <gradient
        android:startColor="#E91E63"
        android:endColor="#009688"
        android:angle="270" />
</shape>
```

En *activity_main.kt*:

```
<?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:gravity="center"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:textColor="@color/white"
        android:background="@drawable/gradient_background"
        android:padding="16dp" />
</LinearLayout>
```

Ejercicio 7.

Generar un Botón que utilice como background el gradiente definido en el ejercicio anterior. Cuando el usuario presiona el botón, deberá cambiar el gradiente a un color diferente. Para ello, generar un nuevo recurso similar al del ejercicio anterior, pero con colores diferentes para el gradiente con nombre: gradient_background_pressed. A continuación, definir un nuevo recurso drawable que defina los dos estados del botón y que haga referencia a los drawables previamente definidos. Se deberá utilizar un StateList. Se puede obtener información de la sintaxis a utilizar en: https://developer.android.com/guide/topics/resources/drawable-resource?hl=es-419#StateList.

En *gradient_background.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <stroke
        android:width="3dp"
        android:color="#3F51B5" />
    <gradient
        android:startColor="#E91E63"
        android:endColor="#009688"
        android:angle="270" />
</shape>
```

En *gradient_background_pressed.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <stroke
        android:width="3dp"
        android:color="#FF9800" />
    <gradient
        android:startColor="#FFEB3B"
        android:endColor="#F57C00"
        android:angle="270" />
</shape>
```

En *gradient_background_selector.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<selector
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:state_selected="true"
        android:drawable="@drawable/gradient_background_pressed" />
    <item
        android:drawable="@drawable/gradient_background" />
</selector>
```

En *activity_main.xml*:

```
<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:gravity="center"
    tools:context=".MainActivity">
    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cambiar Background"
        android:textColor="@color/white"
        android:background="@drawable/gradient_background_selector"
        android:padding="16dp"
        android:layout_marginBottom="16dp" />
</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 btn = findViewById<Button?>(R.id.btn)
        btn.setOnClickListener {
            Toast.makeText(this, "¡Botón presionado!",
Toast.LENGTH_SHORT).show()
            btn.isSelected = !btn.isSelected
        }
    }
}
```

Ejercicio 8.

Intentar utilizar el drawable StateList definido en el ejercicio anterior en otra componente visual como un ImageView o un TextView. ¿Funciona el cambio de estado al presionar sobre la componente? Establecer el atributo "clickable" de la componente en true y volver a intentarlo.

Al intentar utilizar el *drawable StateList* definido en el ejercicio anterior en otra componente visual como un *TextView*, el cambio de estado no funciona al presionar sobre la componente. Al establecer el atributo "*clickable*" de la componente en *true*, sí funciona.

En *activity_main.xml*:

```
<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:gravity="center"
    tools:context=".MainActivity">
    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cambiar Background"
        android:textColor="@color/white"
        android:background="@drawable/gradient_background_selector"
        android:padding="16dp"
        android:layout_marginBottom="16dp" />
    <TextView
        android:id="@+id/txt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:textColor="@color/white"
        android:background="@drawable/gradient_background_selector"
        android:padding="16dp"
        android:clickable="true" />
</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 btn = findViewById<Button?>(R.id.btn)
        val txt = findViewById<TextView?>(R.id.txt)
        btn.setOnClickListener {
            Toast.makeText(this, "¡Botón presionado!",
Toast.LENGTH_SHORT).show()
            btn.isSelected = !btn.isSelected
        }
        txt.setOnClickListener {
            Toast.makeText(this, "¡TextView presionado!",
Toast.LENGTH_SHORT).show()
            txt.isSelected = !txt.isSelected
        }
    }
}
```