Juan Menduiña

Trabajo Práctico N° 1

Ejercicio 1.

Abrir Android Studio y crear un nuevo proyecto con una Actividad vacía (Empty Activity).

Ver proyecto *"TP1_E1aE14"*.

Ejercicio 2.

Abrir el Android Virtual Device Manager y crear, como dispositivo virtual, un Galaxy Nexus con el nombre Nexus Seminario Android.

Ver proyecto *"TP1_E1aE14"*.

Ejercicio 3.

Probar la aplicación creada en el Ejercicio 1 en el emulador.

Ver proyecto *"TP1_E1aE14"*.

Ejercicio 4.

Describir qué representa una Activity.

En *Android Studio*, una *Activity* representa una única pantalla con una interfaz de usuario (UI) en una aplicación *Android*. Es uno de los componentes fundamentales del ciclo de vida de una *app*.

Una Activity es una clase que maneja la interacción del usuario con una pantalla. Cada vez que se abre una *app* y se ve una pantalla distinta (por ejemplo, una pantalla de *login*, una pantalla de inicio, un perfil de usuario), eso, generalmente, está representado por una *Activity* diferente.

Se puede pensar en una *Activity* como el controlador (*controller*) en el patrón MVC:

- Vista (*View*): Está definida en archivos .xml (*layouts*).
- Modelo (*Model*): Suelen ser las clases de datos o lógica de negocio.
- Controlador (*Activity*): Conecta ambos, recibe eventos (*clicks*, entradas de usuario) y responde mostrando o modificando información.

Ejercicio 5.

Abrir el archivo AndroidManifest.xml. ¿Por qué MainActivity tiene un intent-filter con action MAIN y category LAUNCHER?

En *AndroidManifest.xml*:

```
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Este bloque sirve para indicarle al sistema que esta *Activity* es el "punto de entrada" principal de la *app*.

- *<action android:name="android.intent.action.MAIN" />* ⟶ Significa que esta *Activity* es la principal, es decir, la primera que se debe ejecutar cuando se inicia la *app*.
- *<category android:name="android.intent.category.LAUNCHER" />* ⟶ Indica que esta *Activity* debe aparecer como un ícono en el *launcher* del dispositivo (el menú de *apps* del celular).

Este *intent-filter* le dice al sistema operativo *Android*: "Cuando el usuario toque el ícono de la *app*, abrí *MainActivity*". Si no se tuviera ese *intent-filter*, la *app* no aparecería en el menú de *apps* del celular y no sabría qué pantalla mostrar al iniciar.

Ejercicio 6.

Crear 2 actividades (NuevaActivity1, NuevaActivity2) y ver qué se modificó en el archivo AndroidManifest.xml.

Cuando se crean nuevas *activities* (como *NuevaActivity1* y *NuevaActivity2*), *Android Studio*, automáticamente, las registra en el archivo *AndroidManifest.xml*, porque todas las *activities* deben estar declaradas allí para que *Android* pueda reconocerlas y permitir que se ejecuten.

- *android:name=".NuevaActivity1"* ⟶ Define el nombre de la clase de la nueva *activity*. El punto al principio (.) indica que está en el mismo paquete que la *app* principal.
- *android:exported="false"* ⟶ Indica que esta *activity* no puede ser lanzada desde fuera de la *app*, sólo desde otras partes internas de la propia *app*.

Desde *Android* 12 (API 31), Google requiere, explícitamente, que todas las *activities* tengan declarado el atributo *android:exported*, para temas de seguridad. Este atributo es obligatorio si la *app* tiene *targetSdkVersion 31* o superior.

Ejercicio 7.

Pegar el siguiente código en res/layout/activity_main.xml.

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

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Click"
            android:onClick="onBtnClick"/>
</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">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Click"
        android:onClick="onBtnClick" />
</RelativeLayout>
```

Ejercicio 8.

Añadir el siguiente código en la clase MainActivity.kt, probar en el emulador y analizar el resultado:

```
fun onBtnClick(view: View?) {
   val i = Intent(this, NuevaActivity2::class.java)
   startActivity(i)
}
```

Se deben importar estos paquetes: android.view.View, android.content.Intent.

En *MainActivity.kt*:

```
fun onBtnClick(view: View?) {
    val i = Intent(this, NuevaActivity2::class.java)
    startActivity(i)
}
```

Lo que se observa al iniciar la *app* en el emulador es:

- Se muestra su *MainActivity* con un botón que dice *"Click"*.
- Al tocar el botón, se abre *NuevaActivity2*.

Eso significa que está funcionando perfecto. El botón, ahora, inicia una nueva pantalla.

Ejercicio 9.

¿Qué significa this en el código del ejercicio anterior?

En *Kotlin* (y también en *Java*), la palabra clave *this* se refiere a la instancia actual de la clase. En este caso, como estás dentro de la clase *MainActivity*, *this* representa la *activity* actual, es decir, el objeto de tipo *MainActivity*.

Ejercicio 10.

¿Lo que se utilizó fue un intent implícito o explícito? ¿Cuál es la diferencia entre ambos?

La clase *Intent* necesita dos cosas para funcionar:

Intent(context: Context, destination: Class<>).*

- El primer parámetro es el contexto (*Context*), que le dice desde qué componente del sistema se quiere hacer algo. En este caso, *this* es el contexto, ya que *MainActivity* hereda de *Context*.
- El segundo parámetro es la clase a la que se quiere ir (*NuevaActivity2::class.java*).

Un *intent* explícito es cuando se le dice al sistema, exactamente, a qué clase de *activity* se quiere ir, por lo que, en este caso, se utilizó un *intent* explícito, ya que se dice: "Quiero ir a la *activity NuevaActivity2* que está dentro de la *app*.". Un *intent* implícito es cuando no se especifica la clase exacta, sino que se dice: "Quiero hacer algo y que el sistema busque una *app* o componente que lo pueda hacer."

Ejercicio 11.

A través del AndroidManifest, modificar el nombre que se muestra al usuario para la actividad 2 para que, al hacer click, se muestre con el texto "Actividad Nueva".

En *AndroidManifest.xml*:

```
<activity
    android:name=".NuevaActivity2"
    android:exported="false"
    android:label="Actividad Nueva"
    android:theme="@style/Theme.AppCompat.Light.DarkActionBar" />
```

Ejercicio 12.

Modificar el comportamiento de onBtnClick para que, a través de un intent, abra una página web.

En *MainActivity.kt*:

```
fun onBtnClick(view: View?) {
    val url = "https://www.google.com.ar"
    val i = Intent(Intent.ACTION_VIEW)
    i.data = Uri.parse(url)
    startActivity(i)
}
```

Ejercicio 13.

En el método onCreate de la actividad nueva, añadir la siguiente línea:

```
Log.d("APP_DE_PRUEBA", "Este es mi mensaje de debug");
```

Correr la aplicación en modo debug y revisar la consola de Debug luego de abrir la actividad 2. ¿Qué ocurrió? ¿Cuál es su utilidad?

En *NuevaActivity.kt*:

```
Log.d("APP_DE_PRUEBA", "Este es mi mensaje de debug");
```

Al correr la aplicación en modo *debug*, luego de abrir la *activity* 2, lo que ocurre es que se muestra, en el *Logcat*, lo siguiente:

2025-04-11 07:25:01.391 8288-8288 APP_DE_PRUEBA com.example.tp1 D Este es mi mensaje de debug.

La utilidad de esto es poder confirmar que:

- La *activity* se abrió efectivamente.
- *onCreate()* se ejecutó correctamente.
- El botón o *intent* que se usó para abrir esa pantalla está funcionando.

Es como dejar señales en el camino para saber por dónde pasó el código.

Ejercicio 14.

Al hacer click en el botón, se debe pasar a la actividad 2 un texto como parámetro. Cuando la actividad 2 se muestra, se debe imprimir por consola el texto recibido como parámetro.

En *MainActivity.kt*:

```
fun onBtnClick(view: View?) {
    val i = Intent(this, NuevaActivity2::class.java)
    i.putExtra("mensaje", "¡Hola desde MainActivity!")
    startActivity(i)
}
```

En *NuevaActivity2.kt*:

```
val mensajeRecibido = intent.getStringExtra("mensaje")
Log.d("APP_DE_PRUEBA", "Texto recibido: $mensajeRecibido")
```

Ejercicio 15.

Describir cuáles son los estados por los que puede pasar una actividad.

Una aplicación, generalmente, consiste en múltiples *activities* vinculadas entre sí, corriendo en un único proceso del sistema operativo. Normalmente, hay una *activity* principal que se presenta al usuario cuando éste inicia la aplicación por primera vez.

Cada vez que se inicia una *activity* nueva, se detiene la anterior, se la incluye en la pila de *activities* y ésta obtiene el foco (atención del usuario). Cuando el usuario presiona el botón "Atrás", se quita de la pila, se destruye y se reanuda la *activity* anterior.

Una *activity* puede pasar por distintos estados: *created*, *resumed*, *paused*, *stopped*, *destroyed*.

- ***Created*/*Resumed* (*onCreate()*/*onRestart()*, *onStart()*, *onResume()*):** La *activity* se encuentra en el primer plano y tiene el foco (atención del usuario). También se suele denominar *running*, reanudada o en ejecución.
- ***Paused* (*onPause()*):** La *activity* está parcialmente ocultada por otra *activity* que le quitó el foco (atención del usuario). Permanece "viva" en memoria con toda su información de estado y continúa anexada al administrador de ventanas.
- ***Stopped* (*onStop()*):** La *activity* ya no está visible para el usuario, está complemente ocultada por otra *activity*. Permanece "vida" en memoria con toda su información de estado, pero no está anexada al administrador de ventanas.
- ***Destroyed* (*onDestroy()*):** La *activity* ha sido totalmente eliminada. Si se invoca nuevamente, deberá volver a crearse.

Ejercicio 16.

Describir cuáles son los eventos generados a partir de un cambio de estado de una actividad.

Los eventos generados a partir de un cambio de estado de una *activity* son los métodos del ciclo de vida que *Android* llama automáticamente cada vez que la *activity* entra o sale de un estado. Estos métodos son como "ganchos" que permiten ejecutar código cuando cambian los estados.

- ***onCreate()*:** Se ejecuta justo después del constructor. Aquí, se inicializa la *activity* y se define la vista de la misma. Este método recibe un parámetro nulo, si es la primera vez que se crea la *activity*, o con datos para recuperar el estado anterior, en caso contrario.
- ***onStart()*:** Hace que el usuario pueda ver la *activity*, mientras la *app* se prepara para que ésta entre en primer plano y se convierta en interactiva. Puede utilizarse para crear procesos cuyo objetivo es actualizar la interfaz de usuario (animaciones, temporizadores, localización GPS, etc.).
- ***onResume()*:** Se ejecuta justo después de que la *activity* sea complementa visible y obtenga el foco. Es el sitio indicado para iniciar animaciones, acceder a recursos de forma exclusiva como la cámara, etc.
- ***onPause()*:** Se ejecuta cuando la *activity* actual pierde el foco porque va a ser reemplazada por otra. Es el sitio ideal para detener todo lo que se ha activado en *onResume()* y liberar los recursos de uso exclusivo. La ejecución de este método debería ser lo más rápida posible, ya que el usuario no podrá utilizar la nueva *activity* hasta que ésta finalice.
- ***onStop()*:** Es invocado cuando la *activity* ha sido completamente ocultada por otra que ya está interactuando con el usuario. Aquí, se suelen destruir los procesos creados en el método *onStart()*.
- ***onRestart()*:** Se ejecuta cuando una *activity* que había sido ocultada (pero no destruida) tiene que mostrarse de nuevo. Es poco utilizado, pero puede ser útil en algunos casos.
- ***onDestroy()*:** El sistema ejecuta este método cuando ya no tiene intención de reutilizar más la *activity*. Aquí, ya no se puede hacer otra cosa que destruir los objetos, hilos y demás que se haya creado en *onCreate()*.

Ejercicio 17.

Crear una nueva aplicación en la que se imprima por la consola los cambios de estados de una actividad utilizando lo investigado en los Ejercicios 15 y 16.

Cada vez que la *activity* pase por un estado (crear, iniciar, reanudar, pausar, detener, reiniciar o destruir), se imprimirá un mensaje en la consola (*Logcat* en *Android Studio*) con el nombre del evento llamado.

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private val TAG = "CICLO_DE_VIDA"
    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
        }
        Log.d(TAG, "onCreate() llamado")
    }
    override fun onStart() {
        super.onStart()
        Log.d(TAG, "onStart() llamado")
    }
    override fun onResume() {
        super.onResume()
        Log.d(TAG, "onResume() llamado")
    }
    override fun onPause() {
        super.onPause()
        Log.d(TAG, "onPause() llamado")
    }
    override fun onStop() {
        super.onStop()
        Log.d(TAG, "onStop() llamado")
    }
    override fun onRestart() {
        super.onRestart()
        Log.d(TAG, "onRestart() llamado")
    }
    override fun onDestroy() {
        super.onDestroy()
        Log.d(TAG, "onDestroy() llamado")
    }
}
```

Ejercicio 18.

Generar una nueva aplicación con una actividad vacía en Android Studio. Editar el archivo AndroidManifest.xml y eliminar las siguientes líneas:

```
<intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

(a) *¿Qué error se produce en el entorno de desarrollo? ¿Cuál es el motivo del error?*

En *AndroidManifest.xml*:

```
<activity
    android:name=".MainActivity"
    android:exported="true">
</activity>
```

Al eliminar esas líneas, el error que se produce en el entorno de desarrollo es:

Error running 'app'
Default Activity not found

El motivo del error es que ese bloque de *<intent-filter>* es lo que le indica al sistema operativo *Android* cuál es la actividad principal que se debe lanzar al iniciar la *app*.

- La acción *MAIN* indica que esta actividad es el punto de entrada principal.
- La categoría *LAUNCHER* indica que esta actividad debe aparecer en el *launcher* del dispositivo, es decir, como ícono de *app*.

Al no tener una actividad con ese *intent-filter*, no se puede iniciar la aplicación automáticamente desde el ícono del *launcher* y *Android Studio* no sabe qué actividad lanzar como principal, por lo cual lanza ese error.

(b) *Volver a colocar el código eliminado para que la aplicación funcione correctamente.*

En *AndroidManifest.xml*:

```
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

(c) *Agregar un TextView a la Activity generada con el valor "Español" y un botón con el valor 'Cambiar idioma'. Al hacer click en el botón, el textview debe cambiar su texto a "Inglés". Si se presiona, nuevamente, sobre el botón, el textview debe contener el valor "Español" nuevamente.*

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:gravity="center"
    android:padding="16dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txtIdioma"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Español"
        android:textSize="20sp"
        android:layout_marginBottom="20dp" />
    <Button
        android:id="@+id/btnCambiarIdioma"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cambiar idioma"
        android:textSize="20sp" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private var enEspañol = false
    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 txtIdioma = findViewById<TextView?>(R.id.txtIdioma)
        val btnCambiar = findViewById<Button?>(R.id.btnCambiarIdioma)
        btnCambiar.setOnClickListener {
            enEspañol = !enEspañol
            if (enEspañol) {
                txtIdioma.text = "Inglés"
```

```
            }
            else {
                txtIdioma.text = "Español"
            }
        }
    }
}
```

Ejercicio 19.

Crear una nueva aplicación con 2 actividades como se ven en la figura.

Al hacer click en "Realizar operación", se debe abrir la segunda actividad. Al hacer click en los botones "Incrementar" o "Decrementar", la actividad debe cerrarse y aplicar la operación correspondiente sobre el TextView. Si se presiona "Cancelar", sólo debe cerrarse la segunda actividad sin realizar cambios sobre el TextView.

A continuación, se detallan los layouts de las activities para simplificar el ejercicio (notar el uso de LinearLayout):

Activity1:

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


        <TextView android:id="@+id/txtContador"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="0"
                android:textAlignment="center"
                android:textSize="34sp"
            />

        <Button android:id="@+id/btnRealizarOperacion"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Realizar operación"
             />
</LinearLayout>
```

Activity2:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <Button android:id="@+id/btnIncrementar"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Incrementar"
            />

        <Button android:id="@+id/btnDecrementar"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Decrementar"
            />

        <Button android:id="@+id/btnCancelar"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Cancelar"
            />

</LinearLayout>
```

En *AndroidManifest.xml*:

```
<activity
    android:name=".MainActivity"
    android:exported="true"
    android:label="Contador"
    android:theme="@style/Theme.AppCompat.Light.DarkActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".Operaciones"
    android:exported="false"
    android:label="Operaciones"
    android:theme="@style/Theme.AppCompat.Light.DarkActionBar" />
```

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/txtContador"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="0"
        android:textAlignment="center"
        android:textSize="34sp" />
    <Button
        android:id="@+id/btnRealizarOperacion"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Realizar operación" />
</LinearLayout>
```

En *activity_operaciones.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="horizontal"
    tools:context=".Operaciones">
    <Button
        android:id="@+id/btnIncrementar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Incrementar" />
    <Button
        android:id="@+id/btnDecrementar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Decrementar" />
    <Button
        android:id="@+id/btnCancelar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cancelar" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var txtContador: TextView
    private var contador = 0
    companion object {
```

```
        const val REQUEST_CODE = 1
        const val RESULT_INCREMENTAR = 1
        const val RESULT_DECREMENTAR = 2
    }
    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
        }
        txtContador = findViewById(R.id.txtContador)
        val btnOperacion =
findViewById<Button?>(R.id.btnRealizarOperacion)
        btnOperacion.setOnClickListener {
            val intent = Intent(this, Operaciones::class.java)
            startActivityForResult(intent, REQUEST_CODE)
        }
    }
    override fun onActivityResult(requestCode: Int, resultCode: Int,
data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if (requestCode == REQUEST_CODE) {
            when (resultCode) {
                RESULT_INCREMENTAR -> contador++
                RESULT_DECREMENTAR -> contador--
            }
            txtContador.text = contador.toString()
        }
    }
}
```

En *Operaciones.kt*:

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

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
        }
        findViewById<Button>(R.id.btnIncrementar).setOnClickListener {
            setResult(MainActivity.RESULT_INCREMENTAR)
            finish()
        }
        findViewById<Button>(R.id.btnDecrementar).setOnClickListener {
            setResult(MainActivity.RESULT_DECREMENTAR)
            finish()
```

```
        }
        findViewById<Button>(R.id.btnCancelar).setOnClickListener {
            finish()
        }
    }
}
```

Ejercicio 20.

Agregar un control más al ejercicio anterior para que no pueda decrementarse si el valor es 0. Para ello, al retornar a la actividad 1, verificar si el valor es 0 y desplegar un mensaje Toast informando el error.

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var txtContador: TextView
    private var contador = 0
    companion object {
        const val REQUEST_CODE = 1
        const val RESULT_INCREMENTAR = 1
        const val RESULT_DECREMENTAR = 2
    }
    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
        }
        txtContador = findViewById(R.id.txtContador)
        val btnOperacion =
findViewById<Button?>(R.id.btnRealizarOperacion)
        btnOperacion.setOnClickListener {
            //val intent = Intent(this, Operaciones::class.java)
            val intent = Intent("com.example.tp1_e19ae22.OPERACIONES")
            startActivityForResult(intent, REQUEST_CODE)
        }
    }
    override fun onActivityResult(requestCode: Int, resultCode: Int,
data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if (requestCode == REQUEST_CODE) {
            when (resultCode) {
                RESULT_INCREMENTAR -> contador++
                //RESULT_DECREMENTAR -> contador--
                RESULT_DECREMENTAR -> {
                    if (contador > 0) {
                        contador--
                    }
                    else {
                        Toast.makeText(this, "No se puede decrementar.
El contador está en 0.", Toast.LENGTH_SHORT).show()
                    }
                }
            }
            txtContador.text = contador.toString()
        }
```

```
        }
}
```

Ejercicio 21.

Modificar el ejercicio anterior para que la segunda activity (operaciones) se abra con un intent implícito.

En *AndroidManifest.xml*:

```
<activity
    android:name=".Operaciones"
    android:exported="true"
    android:label="Operaciones"
    android:theme="@style/Theme.AppCompat.Light.DarkActionBar">
    <intent-filter>
        <action android:name="com.example.tp1_e19ae22.OPERACION" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

En *MainActivity.kt*:

```
btnOperacion.setOnClickListener {
    //val intent = Intent(this, Operaciones::class.java)
    val intent = Intent("com.example.tp1_e19ae22.OPERACION")
    startActivityForResult(intent, REQUEST_CODE)
}
```

Ejercicio 22.

¿Qué sucede en el ejercicio anterior si se modifica la orientación del dispositivo (horizontal/vertical)? Solucionar el problema mediante saveInstanceState/restoreInstanceState.

Al cambiar la orientación del dispositivo (de vertical a horizontal o viceversa), *Android* destruye y vuelve a crear la actividad, lo cual reinicia las variables (como el contador en este caso).

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var txtContador: TextView
    private var contador = 0
    companion object {
        const val REQUEST_CODE = 1
        const val RESULT_INCREMENTAR = 1
        const val RESULT_DECREMENTAR = 2
    }
    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
        }
        txtContador = findViewById(R.id.txtContador)
        val btnOperacion =
findViewById<Button?>(R.id.btnRealizarOperacion)
        btnOperacion.setOnClickListener {
            //val intent = Intent(this, Operaciones::class.java)
            val intent = Intent("com.example.tp1_e19ae22.OPERACIONES")
            startActivityForResult(intent, REQUEST_CODE)
        }
        if (savedInstanceState != null) {
            contador = savedInstanceState.getInt("contador")
            txtContador.text = contador.toString()
        }
    }
    override fun onActivityResult(requestCode: Int, resultCode: Int,
data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if (requestCode == REQUEST_CODE) {
            when (resultCode) {
                RESULT_INCREMENTAR -> contador++
                //RESULT_DECREMENTAR -> contador--
                RESULT_DECREMENTAR -> {
                    if (contador > 0) {
                        contador--
                    }
```

```
                    else {
                        Toast.makeText(this, "No se puede decrementar.
El contador está en 0.", Toast.LENGTH_SHORT).show()
                    }
                }
            }
            txtContador.text = contador.toString()
        }
    }
    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putInt("contador", contador)
    }
}
```

Ejercicio 23.

Generar una actividad con nombre LifeCycleActivity y pruebe el siguiente código:

```
override fun onDestroy() {
    super.onDestroy()
    val i = Intent(this, LifeCycleActivity::class.java)
    this.startActivity(i)
}
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private val TAG = "CICLO_DE_VIDA"
    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 texto = findViewById<TextView?>(R.id.texto)
    }
    override fun onStart() {
        super.onStart()
        Log.d(TAG, "onStart() llamado")
    }
    override fun onResume() {
        super.onResume()
        Log.d(TAG, "onResume() llamado")
    }
    override fun onPause() {
        super.onPause()
        Log.d(TAG, "onPause() llamado")
    }
    override fun onStop() {
        super.onStop()
        Log.d(TAG, "onStop() llamado")
    }
    override fun onRestart() {
        super.onRestart()
        Log.d(TAG, "onRestart() llamado")
    }
    override fun onDestroy() {
        super.onDestroy()
        Log.d(TAG, "onDestroy() llamado")
        val i = Intent(this, MainActivity::class.java)
        this.startActivity(i)
    }
}
```

Ejecutar la aplicación:

(a) *Intentar destruir la actividad mediante el botón "Atrás" del dispositivo.*

Cuando se intenta destruir la *activity* mediante el botón "Atrás" del dispositivo, *Android* llama a los siguientes métodos del ciclo de vida: *onPause()*, *onStop()* y *onDestroy()*, y la *activity* se destruye.

(b) *Intentar destruir la actividad mediante el botón de intercambio de tareas (botón central del Nexus S).*

Cuando se intenta destruir la *activity* mediante el botón de intercambio de tareas, ésta no se destruye, sino que la *activity* entra en pausa (*onPause()*) y, posiblemente, en estado detenido (*onStop()*), pero no se destruye (*onDestroy()*).

Ejercicio 24.

En el ejercicio anterior, agregar, a la actividad, un TextView con id "texto". Agregar, al método onDestroy(), el siguiente código:

```
override fun onDestroy() {
    super.onDestroy()
    (findViewById<View>(R.id.texto) as TextView).text = "HOLA MUNDO!"
    val i = Intent(this, LifeCycleActivity::class.java)
    this.startActivity(i)
}
```

Intentar destruir la actividad mediante el botón "Atrás" del dispositivo.

(a) *¿Se ve el mensaje "HOLA MUNDO!" en la componente TextView? ¿Por qué?*

El mensaje "HOLA MUNDO!" no se verá en la componente *TextView*, ya que el método *onDestroy()* es llamado justo antes de que la *activity* sea destruida y, en ese momento, es posible que los cambios no se reflejen, visualmente, en la interfaz de usuario. Esto es porque el sistema operativo podría estar en el proceso de liberar los recursos y destruir la *activity*, lo que hace que el cambio en la vista no se vea.

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private val TAG = "CICLO_DE_VIDA"
    private lateinit var texto: TextView
    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
        }
        texto = findViewById(R.id.texto)
    }
    override fun onStart() {
        super.onStart()
        Log.d(TAG, "onStart() llamado")
    }
    override fun onResume() {
        super.onResume()
        Log.d(TAG, "onResume() llamado")
    }
    override fun onPause() {
        super.onPause()
        Log.d(TAG, "onPause() llamado")
    }
    override fun onStop() {
```

```
        super.onStop()
        Log.d(TAG, "onStop() llamado")
    }
    override fun onRestart() {
        super.onRestart()
        Log.d(TAG, "onRestart() llamado")
    }
    override fun onDestroy() {
        super.onDestroy()
        Log.d(TAG, "onDestroy() llamado")
        (findViewById<View>(R.id.texto) as TextView).text = "HOLA
MUNDO!"
        val i = Intent(this, MainActivity::class.java)
        i.putExtra("texto", texto.text)
        this.startActivity(i)
    }
}
```

(b) *¿Se puede resolver mediante saveInstanceState/restoreInstanceState?*

Sí, se puede resolver mediante *saveInstanceState/restoreInstanceState.*

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private val TAG = "CICLO_DE_VIDA"
    private lateinit var texto: TextView
    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
        }
        texto = findViewById(R.id.texto)
        val textoIntent = intent.getStringExtra("texto")
        val textoRecuperado = savedInstanceState?.getString("texto")
        texto.text = textoRecuperado ?: textoIntent ?: texto.text
    }
    override fun onDestroy() {
        super.onDestroy()
        Log.d(TAG, "onDestroy() llamado")
        (findViewById<View>(R.id.texto) as TextView).text = "HOLA
MUNDO!"
        val i = Intent(this, MainActivity::class.java)
        i.putExtra("texto", texto.text)
        this.startActivity(i)
    }
    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putString("texto", texto.text.toString())
```

```
    }
    override fun onRestoreInstanceState(savedInstanceState: Bundle) {
        super.onRestoreInstanceState(savedInstanceState)
        texto.text = savedInstanceState.getString("texto")
    }
}
```

(c) *¿Qué sucede con la instancia de LifeCycleActivity?*

Cuando se llama a *startActivity(i)* dentro de *onDestroy()*, lo que ocurre es:

- Justo antes de que la actividad actual se destruya, se lanza una nueva instancia de *LifeCycleActivity*.
- Esa nueva instancia se crea normalmente, como cualquier *activity* que se inicie con un *Intent*.
- No es la misma instancia que la que se está destruyendo, sino más bien una *activity* completamente nueva.

Por lo tanto, la instancia de *LifeCycleActivity* con la cual se invoca al método *onDestroy()* se destruye y se crea una nueva instancia de *LifeCycleActivity*.

Ejercicio 25.

Crear una nueva aplicación que tenga 2 actividades. En la actividad 1, reemplazar el código del archivo activity_main.xml por el presentado a continuación.

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
        <!-- Aquí van los botones →
</LinearLayout>
```

La primera actividad debe tener 2 botones. El primero debe tener el texto "¿Qué hora es?" y, al hacer click, debe imprimir en la consola de Debug la hora actual. El segundo debe contener el texto "¿Qué día es?" y, al hacer click, pasar como parámetro el día de hoy a la segunda actividad. Al abrirse la segunda actividad, debe imprimir el valor recibido.

Nota: Investigar la clase SimpleDateFormat para convertir la fecha a String en un formato específico.

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:padding="24dp"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/btnHora"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="¿Qué hora es?"
        android:textSize="20sp"
        android:layout_marginBottom="20dp" />
    <Button
        android:id="@+id/btnDia"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="¿Qué día es?"
        android:textSize="20sp" />
</LinearLayout>
```

En *activity_second.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:padding="24dp"
    tools:context=".SecondActivity">
    <TextView
        android:id="@+id/txtDia"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Esperando día..."
        android:textSize="20sp" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private val TAG = "MainActivityDebug"
    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 btnHora = findViewById<Button?>(R.id.btnHora)
        val btnDia = findViewById<Button?>(R.id.btnDia)
        btnHora.setOnClickListener {
            val horaActual = SimpleDateFormat("HH:mm:ss",
Locale.getDefault()).apply { timeZone = TimeZone.getTimeZone("GMT-3")
}.format(Date())
            Log.d(TAG, "Hora actual: $horaActual")
        }
        btnDia.setOnClickListener {
            val diaActual = SimpleDateFormat("EEEE, d 'de' MMMM 'de'
yyyy", Locale("es", "ES")).apply { timeZone =
TimeZone.getTimeZone("GMT-3") }.format(Date()).replaceFirstChar {
it.uppercaseChar() }
            val intent = Intent(this, SecondActivity::class.java)
            intent.putExtra("dia", diaActual)
            startActivity(intent)
        }
    }
}
```

En *SecondActivity.kt*:

```
class SecondActivity : AppCompatActivity() {
    private val TAG = "SecondActivityDebug"
```

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

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 diaRecibido = intent.getStringExtra("dia")
        Log.d(TAG, "Día recibido: $diaRecibido")
        val txtDia = findViewById<TextView>(R.id.txtDia)
        txtDia.text = "Hoy es: $diaRecibido"
    }
}
```

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?*

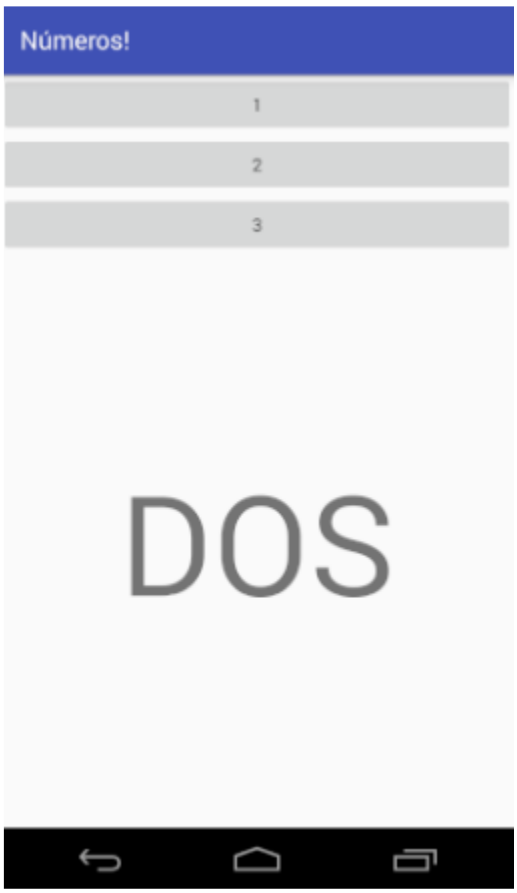
No, esta problemática existe en cualquier sistema con múltiples resoluciones. Por ejemplo:

- 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") }
    }
}
```

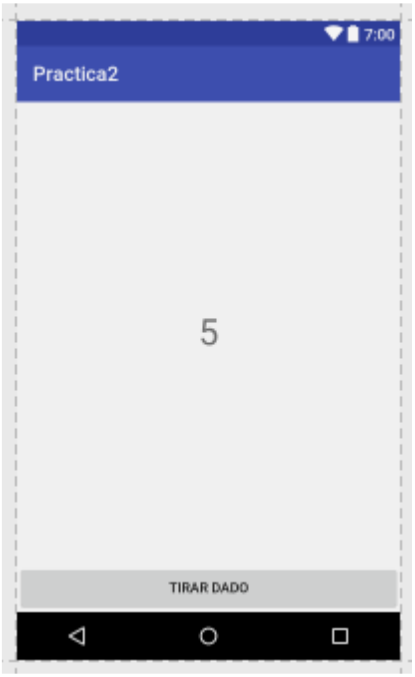
(b) *Implementar utilizando un único manejador de click para todos los botones.*

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)
```

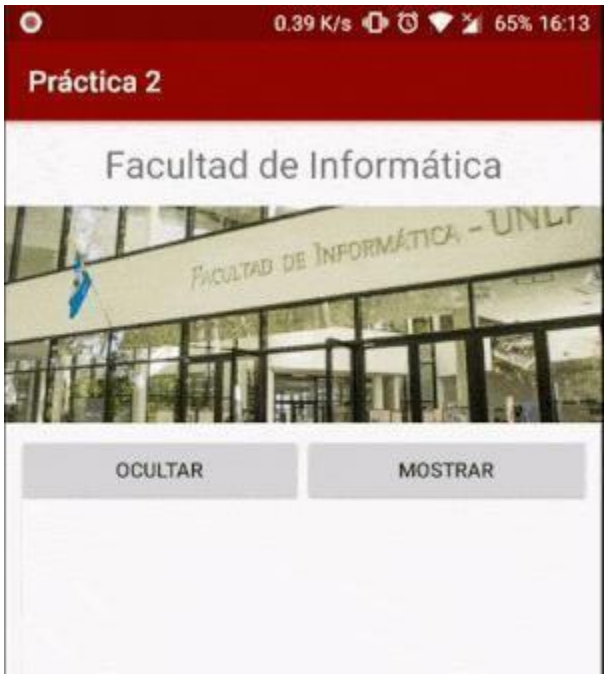
```
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        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.

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"
```

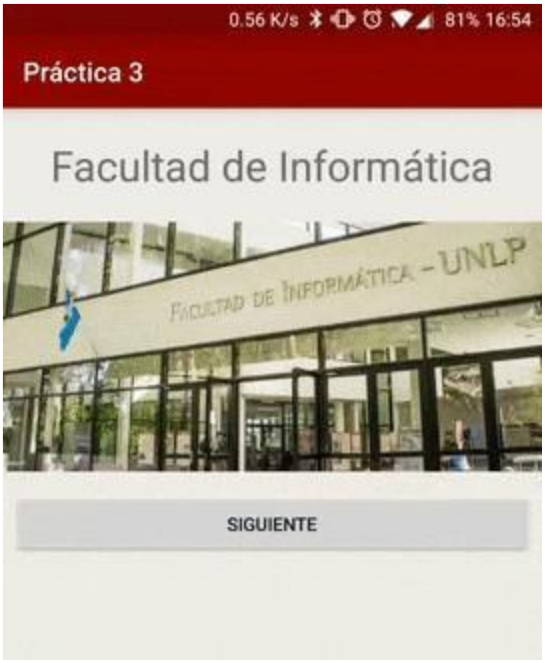
```
            android:textStyle="bold"
            android:background="#800000"
            android:gravity="center_vertical"
            android:padding="16dp"
            android:layout_marginBottom="6dp" />
        <TextView
            android:id="@+id/txtViewImage"
            android:layout_width="match_parent"
            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)
```

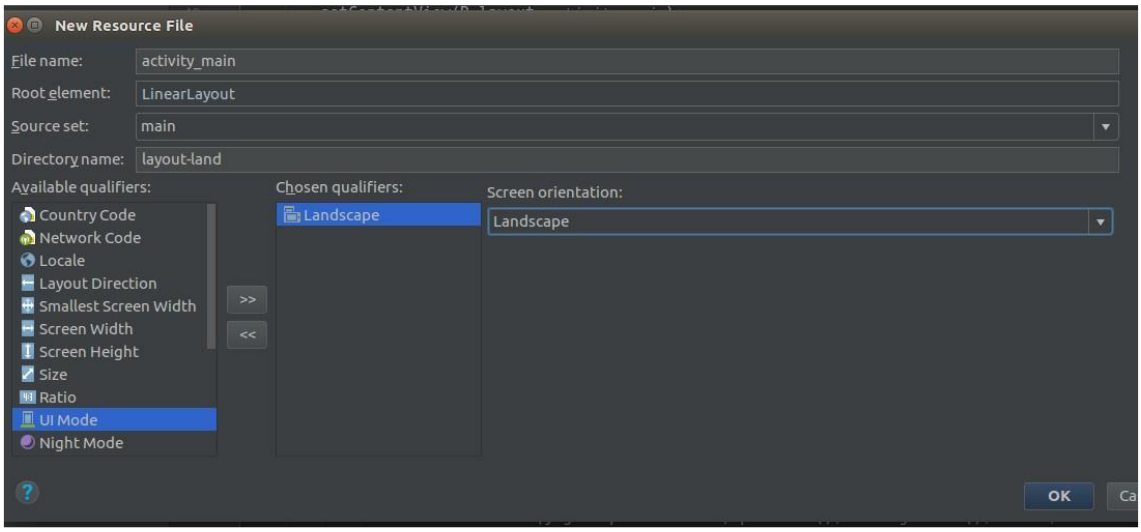
```
            insets
        }
        val imgViewFacultad =
findViewById<ImageView?>(R.id.imgViewFacultad)
        val btnSiguiente = findViewById<Button?>(R.id.btnSiguiente)
        var index = 0
        imgViewFacultad?.setImageResource(imagenes[index])
        btnSiguiente?.setOnClickListener {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

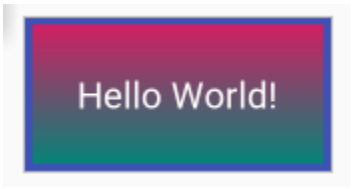
El tipo de elemento de diseño que se utilizaría para mostrar un rectángulo con colores es *shape*.

(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.

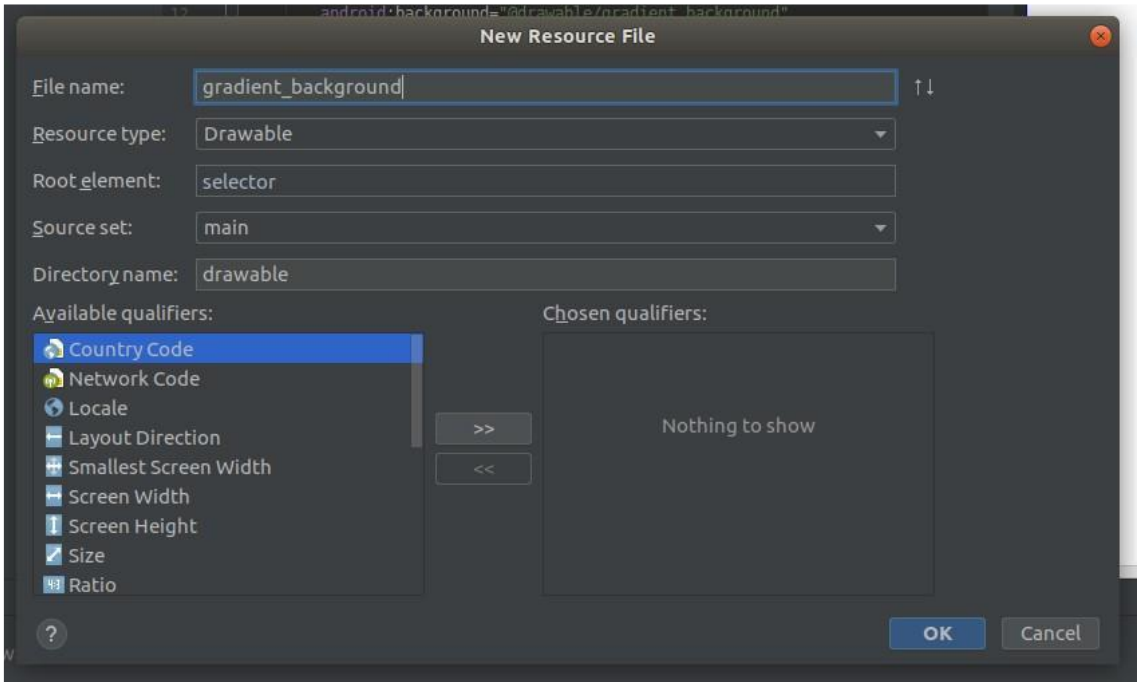
Utilizar un recurso drawable para definir el fondo de un TextView que se visualice de la siguiente manera:



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
        }
    }
}
```

Juan Menduiña

Trabajo Práctico N° 4

Ejercicio 1.

Crear una aplicación con un TextView centrado en pantalla que contenga un contador que vaya incrementándose automáticamente cada un segundo desde el momento que se abre la aplicación.

(a) *¿Se puede implementar el temporizador utilizando un for/while?*

No se puede implementar el temporizador utilizando un *for/while*, al menos no de forma segura o recomendable en *Android*, ya que se bloquea el hilo principal (*UI thread*), haciendo que la interfaz gráfica se congele y la *app* deje de responder.

(b) *Utilizar la clase Handler para resolver el temporizador. Investigar el método postDelayed.*

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:gravity="center"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textCounter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:textColor="@color/black"
        android:textSize="48sp"
        android:textStyle="bold"
        android:layout_marginBottom="16dp" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var textCounter: TextView
    private var counter = 0
    private val handler = Handler(Looper.getMainLooper())
    private val updateCounter = object : Runnable {
        override fun run() {
            counter++
            textCounter.text = counter.toString()
            handler.postDelayed(this, 1000)
```

```
        }
    }
    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
        }
        textCounter = findViewById(R.id.textCounter)
        handler.post(updateCounter)
    }
    override fun onDestroy() {
        super.onDestroy()
        handler.removeCallbacks(updateCounter)
    }
}
```

Ejercicio 2.

Implementar, en la aplicación anterior, un botón que permita detener/reanudar el contador.

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:gravity="center"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textCounter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:textColor="@color/black"
        android:textSize="48sp"
        android:textStyle="bold"
        android:layout_marginBottom="16dp" />
    <Button
        android:id="@+id/btnToggle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Detener"
        android:textColor="@color/white"
        android:textSize="16sp"
        android:textStyle="bold" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var textCounter: TextView
    private lateinit var btnToggle: Button
    private var counter = 0
    private var isRunning = true
    private val handler = Handler(Looper.getMainLooper())
    private val updateCounter = object : Runnable {
        override fun run() {
            if (isRunning) {
                counter++
                textCounter.text = counter.toString()
                handler.postDelayed(this, 1000)
            }
        }
    }
    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
        }
        textCounter = findViewById(R.id.textCounter)
        btnToggle = findViewById(R.id.btnToggle)
        handler.post(updateCounter)
        btnToggle.setOnClickListener {
            isRunning = !isRunning
            if (isRunning) {
                btnToggle.text = "Detener"
                handler.post(updateCounter)
            }
            else {
                btnToggle.text = "Reanudar"
                handler.removeCallbacks(updateCounter)
            }
        }
    }
    override fun onDestroy() {
        super.onDestroy()
        handler.removeCallbacks(updateCounter)
    }
}
```

Ejercicio 3.

Implementar un estado intermedio en el contador en donde el número cambie a color rojo 500 milisegundos antes de incrementarse. Una vez que incrementa, debe volver a su color inicial.

(a) *Implementar utilizando dos Handlers.*

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var textCounter: TextView
    private lateinit var btnToggle: Button
    private var counter = 0
    private var isRunning = true
    private val colorHandler = Handler(Looper.getMainLooper())
    private val tickHandler = Handler(Looper.getMainLooper())
    private val colorRunnable = Runnable {
        if (isRunning) {
            textCounter.setTextColor(Color.RED)
        }
    }
    private val tickRunnable = object : Runnable {
        override fun run() {
            if (isRunning) {
                counter++
                textCounter.text = counter.toString()
                textCounter.setTextColor(Color.BLACK)
                scheduleNextTick()
            }
        }
    }
    private fun scheduleNextTick() {
        colorHandler.postDelayed(colorRunnable, 500)
        tickHandler.postDelayed(tickRunnable, 1000)
    }
    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
        }
        textCounter = findViewById(R.id.textCounter)
        btnToggle = findViewById(R.id.btnToggle)
        scheduleNextTick()
        btnToggle.setOnClickListener {
            isRunning = !isRunning
            if (isRunning) {
                btnToggle.text = "Detener"
```

```
                scheduleNextTick()
            }
            else {
                btnToggle.text = "Reanudar"
                colorHandler.removeCallbacks(colorRunnable)
                tickHandler.removeCallbacks(tickRunnable)
            }
        }
    }
    override fun onDestroy() {
        super.onDestroy()
        colorHandler.removeCallbacks(colorRunnable)
        tickHandler.removeCallbacks(tickRunnable)
    }
}
```

(b) *Implementar utilizando un solo Handler.*

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var textCounter: TextView
    private lateinit var btnToggle: Button
    private var counter = 0
    private var isRunning = true
    private var isRedPhase = false
    private val handler = Handler(Looper.getMainLooper())
    private val runnable = object : Runnable {
        override fun run() {
            if (isRunning) {
                if (!isRedPhase) {
                    textCounter.setTextColor(Color.RED)
                    isRedPhase = true
                }
                else {
                    counter++
                    textCounter.text = counter.toString()
                    textCounter.setTextColor(Color.BLACK)
                    isRedPhase = false
                }
                handler.postDelayed(this, 500)
            }
        }
    }
    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
        }
        textCounter = findViewById(R.id.textCounter)
```

```
        btnToggle = findViewById(R.id.btnToggle)
        handler.post(runnable)
        btnToggle.setOnClickListener {
            isRunning = !isRunning
            if (isRunning) {
                btnToggle.text = "Detener"
                handler.post(runnable)
            }
            else {
                btnToggle.text = "Reanudar"
                handler.removeCallbacks(runnable)
            }
        }
    }
    override fun onDestroy() {
        super.onDestroy()
        handler.removeCallbacks(runnable)
    }
}
```

Ejercicio 4.

Crear una aplicación con dos TextView centrados en la pantalla, "Texto Uno" y "Texto Dos". El primer TextView deberá tener, como tamaño del texto (textSize), "18dp" y el segundo "18sp". Ejecutar en un dispositivo o emulador. Luego, desde la configuración del sistema Android, cambiar el tamaño de la fuente y verificar cómo impacta esto en la aplicación desarrollada.

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:gravity="center"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textOne"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Texto Uno"
        android:textColor="@color/black"
        android:textSize="18dp"
        android:textStyle="bold"
        android:layout_marginBottom="16dp" />
    <TextView
        android:id="@+id/textTwo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Texto Dos"
        android:textColor="@color/black"
        android:textSize="18sp"
        android:textStyle="bold" />
</LinearLayout>
```

Al cambiar al tamaño de la fuente, se observa que:

- El *TextView textTwo* con *textSize="18sp"* cambiará de tamaño, ya que *sp* responde a la escala de texto del sistema.
- El *TextView textOne* con *textSize="18dp"* no cambiará de tamaño, ya que *dp* no responde a la escala de texto del sistema.

Esto ocurre porque:

- *sp (scale-independent pixels)* es una unidad que escala con la configuración de tamaño de fuente del usuario, para respetar preferencias de accesibilidad y facilitar lectura.
- *dp (density-independent pixels)* es una unidad fija para elementos gráficos y no considera la escala de fuente configurada.

Ejercicio 5.

En base a la aplicación desarrollada en el Ejercicio 1, agregar soporte para idioma inglés y portugués, usando archivos XML de strings con calificadores de idioma. Verificar que los textos cambien de idioma al cambiar el idioma del sistema operativo desde la configuración.

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:gravity="center"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textCounter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:textColor="@color/black"
        android:textSize="48sp"
        android:textStyle="bold"
        android:layout_marginBottom="16dp" />
    <Button
        android:id="@+id/btnToggle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn_stop"
        android:textColor="@color/white"
        android:textSize="16sp"
        android:textStyle="bold" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var textCounter: TextView
    private lateinit var btnToggle: Button
    private var counter = 0
    private var isRunning = true
    private var isRedPhase = false
    private val handler = Handler(Looper.getMainLooper())
    private val runnable = object : Runnable {
        override fun run() {
            if (isRunning) {
                if (!isRedPhase) {
                    textCounter.setTextColor(Color.RED)
                    isRedPhase = true
                }
                else {
                    counter++
```

```
                    textCounter.text = counter.toString()
                    textCounter.setTextColor(Color.BLACK)
                    isRedPhase = false
                }
                handler.postDelayed(this, 500)
            }
        }
    }
    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
        }
        textCounter = findViewById(R.id.textCounter)
        btnToggle = findViewById(R.id.btnToggle)
        handler.post(runnable)
        btnToggle.setOnClickListener {
            isRunning = !isRunning
            if (isRunning) {
                btnToggle.text = getString(R.string.btn_stop)
                handler.post(runnable)
            }
            else {
                btnToggle.text = getString(R.string.btn_start)
                handler.removeCallbacks(runnable)
            }
        }
    }
    override fun onDestroy() {
        super.onDestroy()
        handler.removeCallbacks(runnable)
    }
}
```

En *strings.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">TP4_E5aE6</string>
    <string name="btn_stop">Stop</string>
    <string name="btn_start">Resume</string>
</resources>
```

En *strings.xml (es)*:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">TP4_E5aE6</string>
    <string name="btn_stop">Detener</string>
    <string name="btn_start">Reanudar</string>
</resources>
```

En *strings.xml (pt)*:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">TP4_E5aE6</string>
    <string name="btn_stop">Prender</string>
    <string name="btn_start">Retomar</string>
</resources>
```

Ejercicio 6.

Agregar a la aplicación anterior una imagen como recurso drawable llamado "bandera" que se muestre debajo de los textos. Usando calificadores de idiomas para la carpeta drawable, esta bandera deberá ser la de Argentina, Brasil o Reino Unido.

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:gravity="center"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textCounter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:textColor="@color/black"
        android:textSize="48sp"
        android:textStyle="bold"
        android:layout_marginBottom="16dp" />
    <Button
        android:id="@+id/btnToggle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn_stop"
        android:textColor="@color/white"
        android:textSize="16sp"
        android:textStyle="bold"
        android:layout_marginBottom="16dp" />
    <ImageView
        android:id="@+id/imgFlag"
        android:layout_width="240dp"
        android:layout_height="160dp"
        android:src="@drawable/bandera" />
</LinearLayout>
```

Ejercicio 7.

Cambiar la apariencia de la galería de imágenes del Ejercicio 3 de la Práctica 3, para que use un fondo oscuro y letras blancas. Aplicar un estilo con estas características, usando el atributo "theme" en la activity dentro del Manifest.

En *colors.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

En *themes.xml*:

```
<resources>
    <style name="Base.Theme.TP4_E7aE8"
parent="Theme.AppCompat.NoActionBar">
        <item name="android:windowBackground">@color/black</item>
        <item name="android:textColor">@color/white</item>
        <item name="android:textStyle">bold</item>
    </style>
    <style name="Theme.TP4_E7aE8" parent="Base.Theme.TP4_E7aE8" />
</resources>
```

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"
    android:theme="@style/Theme.TP4_E7aE8"
    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:textSize="20sp"
            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:textSize="30sp"
            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:textSize="20sp"
            android:background="#888888"
            android:layout_marginStart="12dp"
            android:layout_marginEnd="12dp" />
    </LinearLayout>
</ScrollView>
```

Ejercicio 8.

Investigar cómo aplicar un tema a la activity programáticamente desde el código Kotlin. Declarar dos estilos en XML, uno llamado "día", con colores claros, y otro "noche", con colores oscuros. Los colores deberán estar declarados en el archivo de recursos de colores XML. Según la hora actual al iniciarse la aplicación, deberá mostrar uno u otro estilo.

Nota: El método setTheme() debe ejecutarse antes del método setContentView() en el onCreate() de la Activity. De este modo, se aplican los cambios de estilo antes de cargar la vista.

En *colors.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

En *themes.xml*:

```
<resources>
    <style name="dia" parent="Theme.AppCompat.NoActionBar">
        <item name="android:windowBackground">@color/white</item>
        <item name="android:textColor">@color/black</item>
        <item name="android:textStyle">bold</item>
    </style>
    <style name="noche" parent="Theme.AppCompat.NoActionBar">
        <item name="android:windowBackground">@color/black</item>
        <item name="android:textColor">@color/white</item>
        <item name="android:textStyle">bold</item>
    </style>
    <style name="Theme.TP4_E7aE8" parent="Base.Theme.TP4_E7aE8" />
</resources>
```

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"
    android:theme="@style/Theme.TP4_E7aE8"
    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="?android:textColorPrimary"
            android:textSize="20sp"
            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:textSize="30sp"
            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:textSize="20sp"
            android:background="#888888"
            android:layout_marginStart="12dp"
            android:layout_marginEnd="12dp" />
    </LinearLayout>
</ScrollView>
```

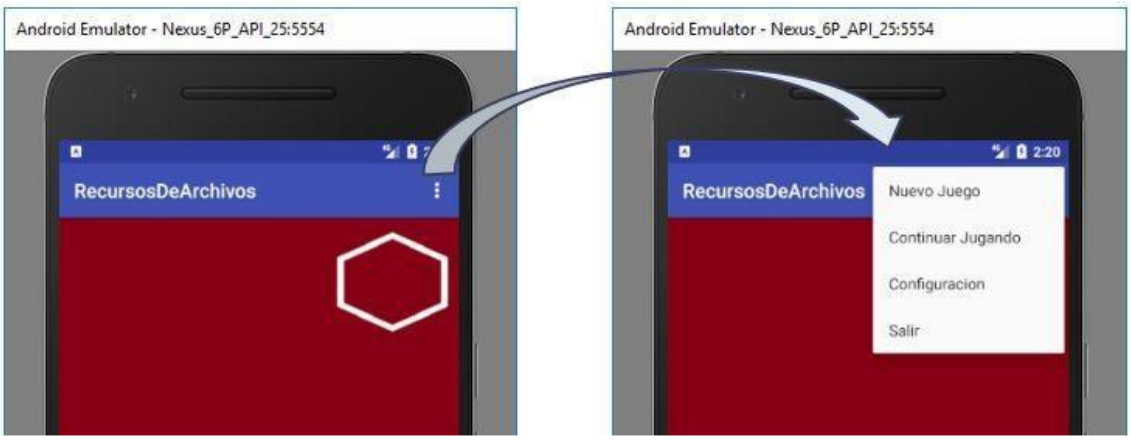
En *MainActivity.kt*:

```
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) +
1
        if (hour in 10..22) {
            setTheme(R.style.dia)
        }
        else {
            setTheme(R.style.noche)
        }
        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 9.

Agregar un menú principal, como el visto en la teoría, a la aplicación de galería de imágenes. Deberá contar con dos opciones, una "Siguiente", que adelanta una imagen, y la otra "Anterior", que vuelve a la imagen anterior.



En *menu_ppal.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/menuSiguiente"
        android:title="Siguiente" />
    <item
        android:id="@+id/menuAnterior"
        android:title="Anterior" />
</menu>
```

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"
    android:theme="@style/Theme.TP4_E9"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:background="#800000"
            android:gravity="center_vertical"
            android:padding="8dp"
```

```
            android:layout_marginBottom="6dp">
            <TextView
                android:id="@+id/title"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="Práctica 3"
                android:textSize="20sp" />
            <androidx.appcompat.widget.Toolbar
                android:id="@+id/toolbar"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1" />
        </LinearLayout>
        <TextView
            android:id="@+id/txtViewImage"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Facultad de Informática"
            android:textSize="30sp"
            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" />
    </LinearLayout>
</ScrollView>
```

En *MainActivity.xml*:

```
class MainActivity : AppCompatActivity() {
    private val imagenes = arrayOf(
        R.drawable.facultad_informatica_1,
        R.drawable.facultad_informatica_2,
        R.drawable.facultad_informatica_3
    )
    private lateinit var imgViewFacultad: ImageView
    private var index = 0
    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 toolbar = findViewById<Toolbar?>(R.id.toolbar)
        setSupportActionBar(toolbar)
        supportActionBar?.setDisplayShowTitleEnabled(false)
        imgViewFacultad = findViewById(R.id.imgViewFacultad)
        mostrarImagenActual()
        val menuHost: MenuHost = this
        menuHost.addMenuProvider(object : MenuProvider {
            override fun onCreateMenu(menu: Menu, menuInflater:
MenuInflater) {
                menuInflater.inflate(R.menu.menu_ppal, menu)
            }
            override fun onMenuItemSelected(item: MenuItem): Boolean {
                return when (item.itemId) {
                    R.id.menuSiguiente -> {
                        mostrarImagenSiguiente()
                        true
                    }
                    R.id.menuAnterior -> {
                        mostrarImagenAnterior()
                        true
                    }
                    else -> false
                }
            }
        }, this)
    }
    private fun mostrarImagenActual() {
        imgViewFacultad.setImageResource(imagenes[index])
    }
    private fun mostrarImagenSiguiente() {
        index = (index + 1) % imagenes.size
        mostrarImagenActual()
    }
    private fun mostrarImagenAnterior() {
        index = if (index - 1 < 0) imagenes.size - 1 else index - 1
        mostrarImagenActual()
    }
}
```

Juan Menduiña

Trabajo Práctico N° 5

Ejercicio 1.

Investigar el uso de los diferentes sensores del dispositivo en la documentación oficial de Android: https://developer.android.com/guide/topics/sensors/sensors_environment (Sección: Use the light, pressure and temperature sensors).

(a) *¿Cuál es la función de la clase SensorManager?*

La función de la clase *SensorManager* es responsable de acceder a los sensores físicos disponibles en un dispositivo *Android*. Proporciona métodos para:

- Obtener una lista de sensores disponibles (*getSensorList()*).
- Obtener un sensor específico (*getDefaultSensor()*).
- Registrar y eliminar *listeners* para recibir datos de sensores (*registerListener()* y *unregisterListener()*).

En resumen, gestiona el acceso a los sensores y facilita la comunicación entre la aplicación y los sensores físicos.

(b) *¿Cómo se genera una instancia de un Sensor?*

Una instancia de un *Sensor* se genera a través del *SensorManager*, usando el método *getDefaultSensor()*. Este método devuelve un objeto Sensor correspondiente al tipo de sensor solicitado (por ejemplo, luz, presión, temperatura, etc.).

(c) *¿Para qué sirven los métodos registerListener y unregisterListener de la clase Sensor?*

- *registerListener:* Registra un *listener* (*SensorEventListener*) para empezar a recibir datos del sensor seleccionado. También permite establecer la frecuencia de muestreo.
- *unregisterListener:* Detiene el envío de datos desde el sensor al *listener* registrado, lo cual es útil para ahorrar batería y recursos del sistema cuando no se necesita monitorear el sensor.

(d) *¿Por qué se utilizan las transiciones de estado onResume y onPause para registrar/eliminar el listener del Sensor?*

Las transacciones de estado *onResume* y *onPause* se utilizan para registrar/eliminar el *listener* del *Sensor* porque gestionan, eficientemente, los recursos del dispositivo. En particular:

- En *onResume()*, se registra el *listener* para empezar a recibir datos del sensor cuando la actividad está visible y en primer plano.
- En *onPause()*, se elimina el *listener* para detener la recepción de datos cuando la actividad ya no está activa, reduciendo, así, el consumo de batería y uso del procesador.

Esta práctica garantiza que los sensores sólo trabajen cuando, realmente, se necesitan.

Ejercicio 2: Sensor de Luminosidad.

(a) *Implementar una aplicación que muestre, en un TextView, el valor del Sensor en tiempo real.*

En *activity_main.xml*:

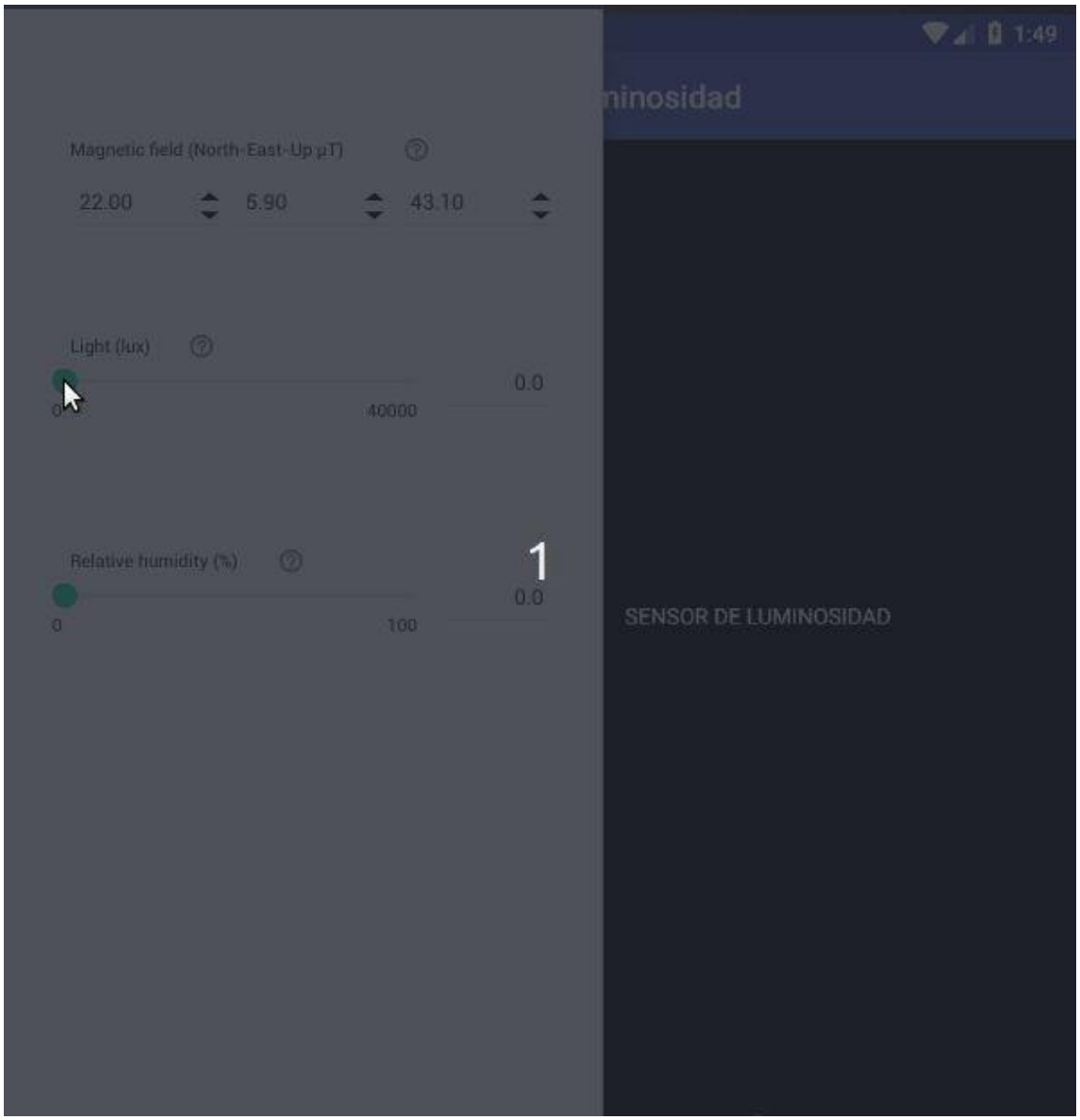
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var txtSensor: TextView
    private lateinit var sensorManager: SensorManager
    private var lightSensor: Sensor? = null
    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
        }
        txtSensor = findViewById(R.id.txtSensor)
        sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
        lightSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT)
        if (lightSensor == null) {
            txtSensor.text = "Sensor de luz no disponible"
        }
    }
    override fun onResume() {
        super.onResume()
```

```
        if (lightSensor != null) {
            sensorManager.registerListener(this, lightSensor,
SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
    override fun onSensorChanged(event: SensorEvent) {
        txtSensor.text = "SENSOR DE LUMINOSIDAD:\n${event.values[0]}
lux"
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {

    }
}
```

(b) *Implementar una aplicación con un TextView que ocupe toda la pantalla y que contenga un texto fijo. El color de fondo y el texto del TextView deberá reaccionar al nivel de luminosidad de manera que se vea el fondo blanco y letras negras en condiciones de mucha luz, mientras que, en condiciones de poca luz, se visualizará el fondo negro con letras blancas.*



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:padding="16dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:textSize="24sp"
        android:textStyle="bold"
```

```
        android:gravity="center" />
</LinearLayout>
```

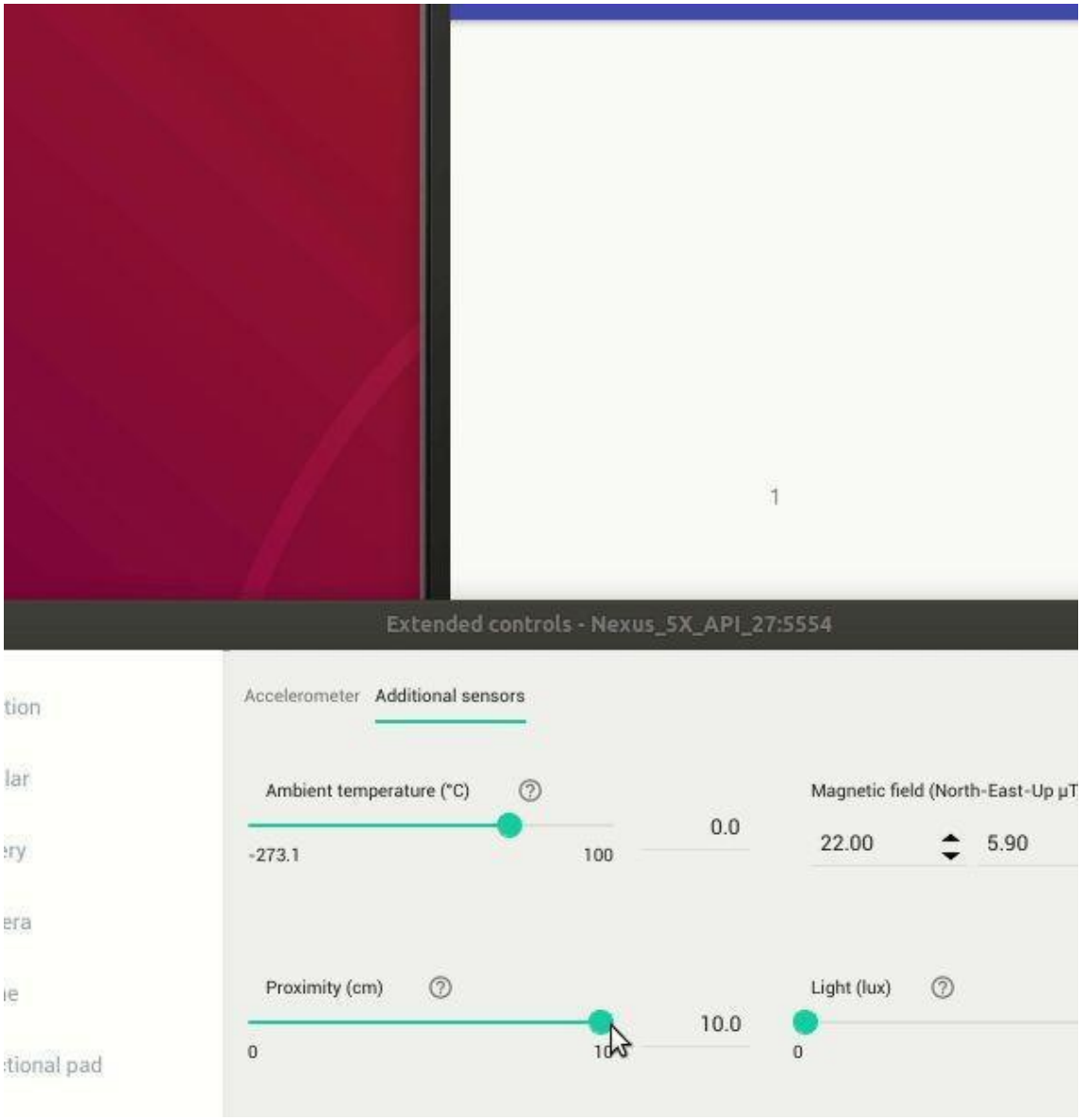
En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var layout: LinearLayout
    private lateinit var txtSensor: TextView
    private lateinit var sensorManager: SensorManager
    private var lightSensor: Sensor? = null
    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
        }
        layout = findViewById(R.id.main)
        txtSensor = findViewById(R.id.txtSensor)
        sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
        lightSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT)
        if (lightSensor == null) {
            txtSensor.text = "Sensor de luz no disponible"
        }
    }
    override fun onResume() {
        super.onResume()
        if (lightSensor != null) {
            sensorManager.registerListener(this, lightSensor,
SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
    override fun onSensorChanged(event: SensorEvent) {
        txtSensor.text = "SENSOR DE LUMINOSIDAD:\n${event.values[0]}
lux"
        if (event.values[0] > 20000) {
            layout.setBackgroundColor(Color.WHITE)
            txtSensor.setTextColor(Color.BLACK)
        }
        else {
            layout.setBackgroundColor(Color.BLACK)
            txtSensor.setTextColor(Color.WHITE)
        }
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {

    }
}
```

Ejercicio 3: Sensor de Proximidad.

Implementar una aplicación que muestre un TextView con un contador numérico. El contador debe incrementarse cada vez que el usuario pase su mano por encima del dispositivo.

Nota: Para determinar que el usuario realizó el gesto de pasar su mano por encima del dispositivo, se deberá detectar un estado de "no proximidad", seguido de un estado de "proximidad".



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:padding="16dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="SENSOR DE PROXIMIDAD:\n0"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center" />
</LinearLayout>
```

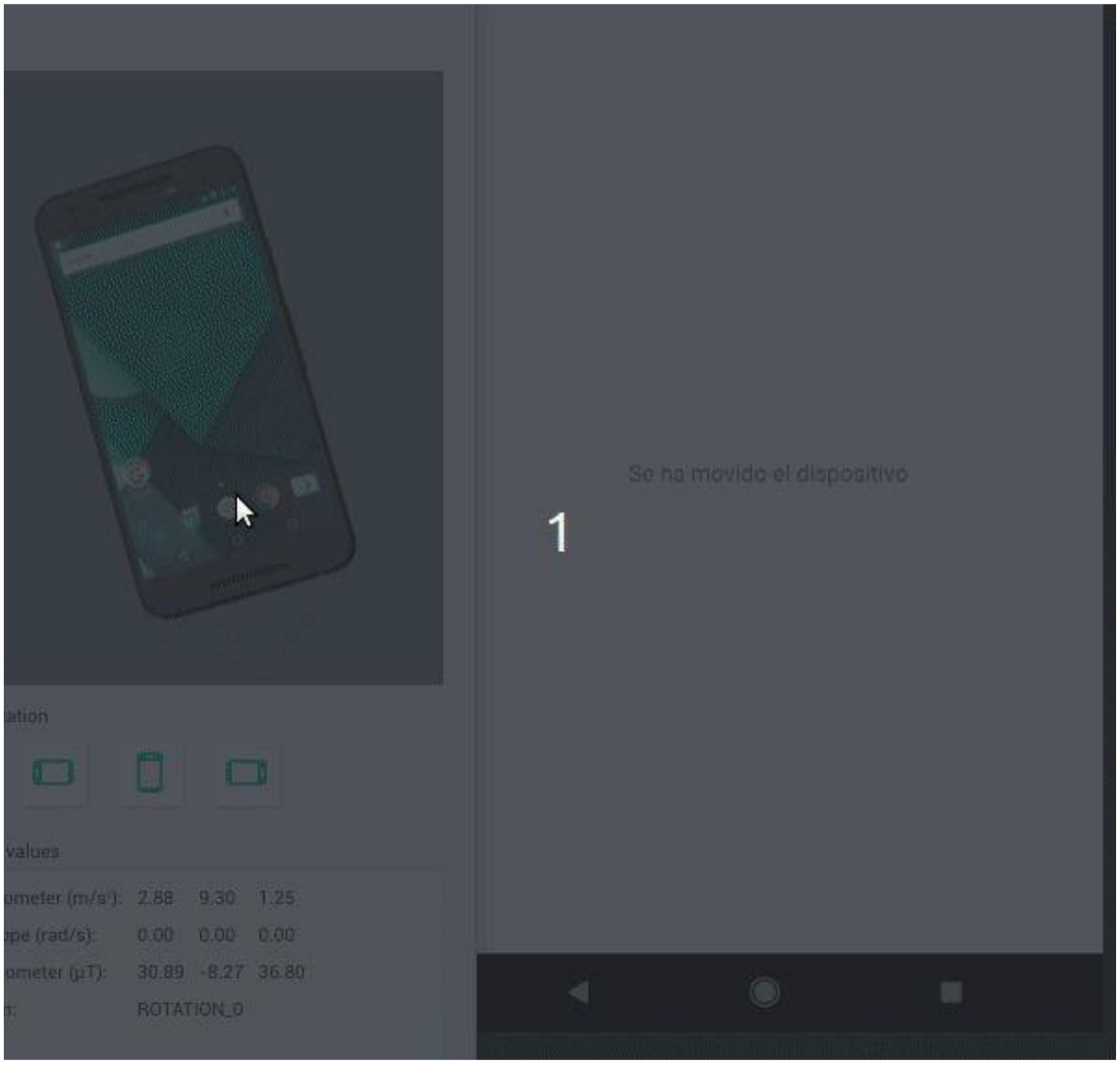
En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var txtSensor: TextView
    private lateinit var sensorManager: SensorManager
    private var proximitySensor: Sensor? = null
    private var counter = 0
    private var wasFar = false
    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
        }
        txtSensor = findViewById(R.id.txtSensor)
        sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
        proximitySensor =
sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY)
        if (proximitySensor == null) {
            txtSensor.text = "Sensor de proximidad no disponible"
        }
    }
    override fun onResume() {
        super.onResume()
        if (proximitySensor != null) {
            sensorManager.registerListener(this, proximitySensor,
SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
    override fun onSensorChanged(event: SensorEvent) {
        val distance = event.values[0]
        if (distance > 0) {
```

```
            wasFar = true
        }
        else if ((distance == 0f) && (wasFar)) {
            wasFar = false
            counter++
            txtSensor.text = "SENSOR DE PROXIMIDAD:\n$counter"
        }
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {

    }
}
```

Ejercicio 4: Acelerómetro.

Nota: Para estos ejercicios, se deberá utilizar el sensor de Aceleración LINEAL, dado que el mismo no tiene en cuenta la fuerza de gravedad.



(a) *Generar una aplicación que muestre, en un TextView, los valores de aceleración en los tres ejes X, Y, Z.*

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:padding="16dp"
    tools:context=".MainActivity">
```

```
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="ACELERACIÓN:\nX: 0.0\nY: 0.0\nZ: 0.0"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var txtSensor: TextView
    private lateinit var sensorManager: SensorManager
    private var linearAccelerationSensor: Sensor? = null
    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
        }
        txtSensor = findViewById(R.id.txtSensor)
        sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
        linearAccelerationSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION)
        if (linearAccelerationSensor == null) {
            txtSensor.text = "Sensor de aceleración lineal no
disponible"
        }
    }
    override fun onResume() {
        super.onResume()
        if (linearAccelerationSensor != null) {
            sensorManager.registerListener(this,
linearAccelerationSensor, SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
    override fun onSensorChanged(event: SensorEvent) {
        val x = event.values[0]
        val y = event.values[1]
        val z = event.values[2]
        val txt = """
            ACELERACIÓN:
            X: ${"%.2f".format(x)} m/s²
            Y: ${"%.2f".format(y)} m/s²
            Z: ${"%.2f".format(z)} m/s²
        """.trimIndent()
```

```
        txtSensor.text = txt
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {

    }
}
```

(b) *Cuando el dispositivo está en reposo, ¿los valores son, estrictamente, cero?*

Cuando el dispositivo está en reposo, los valores no son, estrictamente, cero, ya que, si bien este sensor no tiene en cuenta la fuerza de gravedad, está sujeto a:

- Ruido electrónico: Los sensores físicos tienen imprecisiones mínimas por vibraciones internas, corriente, temperatura, etc.
- Pequeñas vibraciones del entorno.
- Resolución del sensor: Algunos sensores detectan cambios tan pequeños que parecen fluctuaciones aleatorias, aunque no haya movimiento.

(c) *Generar una aplicación que sea capaz de detectar el estado de reposo (completamente quieto) del dispositivo y que genere un aviso en un TextView cuando el mismo pierda su estado de reposo (alguien mueva el dispositivo).*

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:padding="16dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="ESTADO DE REPOSO"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var txtSensor: TextView
    private lateinit var sensorManager: SensorManager
    private var linearAccelerationSensor: Sensor? = null
```

```
    private val threshold = 0.1f
    private var isAtRest = true
    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
        }
        txtSensor = findViewById(R.id.txtSensor)
        sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
        linearAccelerationSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION)
        if (linearAccelerationSensor == null) {
            txtSensor.text = "Sensor de aceleración lineal no
disponible"
        }
    }
    override fun onResume() {
        super.onResume()
        if (linearAccelerationSensor != null) {
            sensorManager.registerListener(this,
linearAccelerationSensor, SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
    override fun onSensorChanged(event: SensorEvent) {
        val x = event.values[0]
        val y = event.values[1]
        val z = event.values[2]
        val movementMagnitude = sqrt((x * x + y * y + z *
z).toDouble()).toFloat()
        if (movementMagnitude < threshold) {
            if (!isAtRest) {
                isAtRest = true
                txtSensor.text = "ESTADO DE REPOSO"
            }
        }
        else {
            if (isAtRest) {
                isAtRest = false
                txtSensor.text = "¡MOVIMIENTO DETECTADO!"
            }
        }
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {

    }
}
```

Ejercicio 5: Permisos.

(a) *¿Por qué algunos sensores requieren solicitar permisos en tiempo de ejecución y otros pueden usarse directamente? Investigar los niveles de los permisos en https://developer.android.com/guide/topics/permissions/overview#normal-dangerous.*

Algunos sensores requieren solicitar permisos en tiempo de ejecución y otros pueden usarse directamente porque los primeros tienen un nivel de permiso *Dangerous* y los otros *Normal*.

Los sensores que no comprometen información personal (como luz, proximidad o aceleración) pueden usarse directamente, mientras que los sensores que sí podrían exponer datos sensibles (como ubicación o salud) necesitan permisos en tiempo de ejecución y, por lo tanto, deben ser solicitados, explícitamente, al usuario en tiempo de ejecución.

(b) *Implementar una aplicación que solicite permisos en tiempo de ejecución para usar la ubicación del usuario. Se pueden utilizar los métodos para consultar y pedir los permisos definidos a continuación. Recordar declarar el permiso tanto en el Manifest como en la Activity.*

```
override fun onStart() {
    super.onStart()
    if (!chequearPermisosLocalizacion()) {
        pedirPermisosLocalizacion()
    } else {
        obtenerLocalizacion()
    }
}

override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<String?>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
    if (requestCode == PERMISSION_GRANTED) {
        if (grantResults.size > 0
            && grantResults[0] ==
PackageManager.PERMISSION_GRANTED
        ) {
            // El usuario concedió el permiso, ya podemos usar
la ubicación
            obtenerLocalizacion()
        }
    }
}

private fun chequearPermisosLocalizacion(): Boolean {
    val permissionState =
        ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)
    return permissionState == PackageManager.PERMISSION_GRANTED
}

private fun pedirPermisosLocalizacion() {
    ActivityCompat.requestPermissions(
        this@MainActivity,
arrayOf(Manifest.permission.ACCESS_COARSE_LOCATION),
        PERMISSION_GRANTED
    )
}
fun obtenerLocalizacion() {

    //Acá usamos el método de geolocalización que hayamos
elegido
}
```

En *build.gradle.kts*:

```
implementation("com.google.android.gms:play-services-location:21.2.0")
```

En *AndroidManifest.xml*:

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

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:padding="16dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var txtSensor: TextView
    companion object { private const val PERMISSION_REQUEST_CODE =
1001 }
    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
        }
        txtSensor = findViewById(R.id.txtSensor)
    }
    override fun onStart() {
        super.onStart()
        if (!chequearPermisosLocalizacion()) {
            pedirPermisosLocalizacion()
        }
        else {
            obtenerLocalizacion()
        }
    }
    override fun onRequestPermissionsResult(requestCode: Int,
permissions: Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
        if (requestCode == PERMISSION_REQUEST_CODE) {
            if ((grantResults.isNotEmpty()) && (grantResults[0] ==
PackageManager.PERMISSION_GRANTED)) {
                Toast.makeText(this, "Permiso de ubicación concedido",
Toast.LENGTH_SHORT).show()
                obtenerLocalizacion()
```

```
            }
            else {
                Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            }
        }
    }
    private fun chequearPermisosLocalizacion(): Boolean {
        return ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) ==
PackageManager.PERMISSION_GRANTED
    }
    private fun pedirPermisosLocalizacion() {
        ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_COARSE_LOCATION),
PERMISSION_REQUEST_CODE)
    }
    private fun obtenerLocalizacion() {
        txtSensor.text = "Consultando ubicación..."
        if (!chequearPermisosLocalizacion()) return
        val fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this)
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            return
        }
        val locationRequest =
LocationRequest.Builder(Priority.PRIORITY_HIGH_ACCURACY, 3000).build()
        val locationCallback = object : LocationCallback() {
            override fun onLocationResult(result: LocationResult) {
                val location = result.lastLocation
                if (location != null) {
                    txtSensor.text = String.format(
                        "Latitud: %.6f\nLongitud: %.6f",
                        location.latitude, location.longitude
                    )
                }
                else {
                    txtSensor.text = "Ubicación no disponible"
                }
            }
        }
        fusedLocationClient.requestLocationUpdates(
            locationRequest,
            locationCallback,
            Looper.getMainLooper()
        )
    }
}
```

Ejercicio 6: Cámara.

(a) *Implementar una aplicación que tome una fotografía usando un intent implícito y la muestre en un ImageView, tal como lo visto en la clase teórica.*

En *AndroidManifest.xml*:

```
<uses-feature android:name="android.hardware.camera"
android:required="true" />
```

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:padding="16dp"
    tools:context=".MainActivity">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="tomarFoto"
        android:text="Tomar Foto"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:backgroundTint="#888888"
        android:layout_marginBottom="6dp" />
    <ImageView
        android:id="@+id/imgFoto"
        android:layout_width="250dp"
        android:layout_height="250dp"
        android:layout_gravity="center"
        android:scaleType="centerCrop" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private val cameraLauncher =
registerForActivityResult(ActivityResultContracts.StartActivityForResu
lt())
    { result ->
        if (result.resultCode == RESULT_OK) {
            val imgFoto = findViewById<ImageView?>(R.id.imgFoto)
            val imgBitmap =
result.data?.extras?.getParcelable<Bitmap>("data")
            imgFoto.setImageBitmap(imgBitmap)
        }
    }
    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
        }
    }
    fun tomarFoto(view: View?) {
        val i = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
        cameraLauncher.launch(i)
    }
}
```

(b) *Al usar la cámara con un intent implícito, no se necesita solicitar permiso al usuario en tiempo de ejecución. ¿Qué diferencia hay si se quiere usar la cámara de manera directa dentro de la app? Investigar en https://developer.android.com/guide/topics/media/camera#manifest.*

La diferencia que hay si se quiere usar la cámara de manera directa dentro de la *app* es que sí necesita declarar el permiso en el archivo *AndroidManifest.xml* y, además, solicitarlo en tiempo de ejecución.

(c) *Si se quiere guardar la imagen en la memoria de almacenamiento, ¿alcanza con haber solicitado el permiso de uso de la cámara?*

Si se quiere guardar la imagen en la memoria de almacenamiento, no alcanza con haber solicitado el permiso de uso de la cámara, sino que también hay que considerar el permiso de acceso al almacenamiento, dependiendo de la versión de *Android*.

Ejercicio 7: Posicionamiento.

(a) *Investigar las diferentes estrategias de obtener el posicionamiento del usuario (GPS, WiFi, Celular), sus ventajas y desventajas. Ver https://developer.android.com/guide/topics/location/strategies.*

Las diferentes estrategias de obtener el posicionamiento del usuario son:

- GPS: Usa satélites para calcular la ubicación geográfica precisa.
  Las ventajas son: alta precisión (5-10 metros); ideal para navegación al aire libre.
  Las desventajas son: no funciona bien en interiores; alto consumo de batería; puede tardar varios segundos en obtener la ubicación.
- WiFi: Usa torres de telefonía móvil y puntos de acceso WiFi cercanos para estimar la ubicación.
  Las ventajas son: funciona bien en interiores; bajo consumo de batería; rápida obtención de ubicación.
  Las desventajas son: menor precisión que el GPS (entre 20-100 metros); requiere conectividad a redes disponibles.
- Celular: Escucha actualizaciones de ubicación que otras *apps* ya están solicitando.
  Las ventajas son: casi sin impacto en la batería.
  Las desventajas son: no garantiza recibir actualizaciones a tiempo; no se controla cuándo ni con qué frecuencia se actualiza la ubicación.

(b) *Implementar una aplicación que muestre la latitud y longitud del usuario en un TextView, usando las APIs de ubicación de Google Play Services. Para esto, se debe agregar Google Play Services al archivo Gradle del proyecto, ubicado en directorio principal del proyecto: classpath 'com.google.gms:google-services:4.3.15'. Debe quedar de la siguiente forma:*

```
// Top-level build file where you can add configuration options common to all
buildscript {

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.1.3'
        classpath 'com.google.gms:google-services:3.2.0'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
```

Luego, se agrega la dependencia de las APIs de ubicación en el archivo gradle de /app/build.gradle: implementation 'com.google.android.gms:play-services-location:15.0.1'.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 27
    defaultConfig {
        applicationId "com.example.alfonso.cameraexample"
        minSdkVersion 23
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-ru
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support:support-v4:27.1.1'
    implementation 'com.android.support:support-media-compat:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.1'
    implementation 'com.google.android.gms:play-services-location:15.0.1'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
}
```

Una vez hecho esto, ya se incluye la librería de ubicación de Google Play Services en la aplicación. Por lo tanto, se pueden usar las clases que provee la librería para solicitar la ubicación:

```
private var client: FusedLocationProviderClient? = null
private var locationCallback: LocationCallback? = null
```

En el método onCreate, inicializamos estas variables:

```
client = LocationServices.getFusedLocationProviderClient(this)
locationCallback = object :
    LocationCallback() {
    override fun onLocationResult(locationResult: LocationResult) {
        super.onLocationResult(locationResult)
        //usamos locationResult.getLastLocation() para tener
        //la ubicación más reciente
    }
}
```

Y, una vez que se tengan los permisos, se le solicita al cliente que envíe la información de localización al callback:

```
client?.requestLocationUpdates(LocationRequest.create(),
    locationCallback as LocationCallback, null
)
```

En *build.gradle.kts*:

```
implementation("com.google.android.gms:play-services-location:21.2.0")
```

En *AndroidManifest.xml*:

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

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:gravity="center"
    android:padding="16dp"
    tools:context=".MainActivity">
    <Button
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="obtenerUbicacion"
        android:text="Obtener ubicación"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:backgroundTint="#888888"
        android:layout_marginBottom="20dp" />
    <TextView
        android:id="@+id/txtUbicacion"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:textSize="20sp" />
 </LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var txtUbicacion: TextView
    companion object { private const val PERMISSION_REQUEST_CODE =
1001 }
    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
        }
        txtUbicacion = findViewById(R.id.txtUbicacion)
    }
    private fun checkLocationPermission(): Boolean {
        val coarseLocation = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)
        val fineLocation = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
        if ((coarseLocation != PackageManager.PERMISSION_GRANTED) ||
(fineLocation != PackageManager.PERMISSION_GRANTED)) {
            requestLocationPermission()
            return false
        }
        return true
    }
    private fun requestLocationPermission() {
        ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_COARSE_LOCATION,
Manifest.permission.ACCESS_FINE_LOCATION), PERMISSION_REQUEST_CODE)
    }
    override fun onRequestPermissionsResult(requestCode: Int,
permissions: Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
        if (requestCode == PERMISSION_REQUEST_CODE) {
            if ((grantResults.isNotEmpty()) && (grantResults[0] ==
```

```
PackageManager.PERMISSION_GRANTED)) {
                Toast.makeText(this, "Permiso de ubicación concedido",
Toast.LENGTH_SHORT).show()
                obtenerUbicacion(null)
            }
            else {
                Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            }
        }
    }
    fun obtenerUbicacion(view: View?) {
        txtUbicacion.text = "Consultando ubicación..."
        if (!checkLocationPermission()) return
        val fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this)
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            return
        }
        val locationRequest =
LocationRequest.Builder(Priority.PRIORITY_HIGH_ACCURACY, 3000).build()
        val locationCallback = object : LocationCallback() {
            override fun onLocationResult(result: LocationResult) {
                val location = result.lastLocation
                if (location != null) {
                    txtUbicacion.text = String.format(
                        "Latitud: %.6f\nLongitud: %.6f",
                        location.latitude, location.longitude
                    )
                }
                else {
                    txtUbicacion.text = "Ubicación no disponible"
                }
            }
        }
        fusedLocationClient.requestLocationUpdates(locationRequest,
locationCallback, Looper.getMainLooper())
    }
}
```

(c) *Implementar una aplicación que, al iniciar, muestre a qué distancia se encuentra el usuario de la Facultad de Informática. Investigar los métodos que provee la clase Location. La ubicación de la Facultad es -34.9037905, -57.9378442.*

En *build.gradle.kts*:

```
implementation("com.google.android.gms:play-services-location:21.2.0")
```

En *AndroidManifest.xml*:

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

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:gravity="center"
    android:padding="16dp"
    tools:context=".MainActivity">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="obtenerDistancia"
        android:text="Obtener distancia"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:backgroundTint="#888888"
        android:layout_marginBottom="20dp" />
    <TextView
        android:id="@+id/txtUbicacion"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:gravity="center" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var txtUbicacion: TextView
    companion object { private const val PERMISSION_REQUEST_CODE =
1001 }
    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
        }
        txtUbicacion = findViewById(R.id.txtUbicacion)
    }
    private fun checkLocationPermission(): Boolean {
        val coarseLocation = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)
```

```
        val fineLocation = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
        if ((coarseLocation != PackageManager.PERMISSION_GRANTED) ||
(fineLocation != PackageManager.PERMISSION_GRANTED)) {
            requestLocationPermission()
            return false
        }
        return true
    }
    private fun requestLocationPermission() {
        ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_COARSE_LOCATION,
Manifest.permission.ACCESS_FINE_LOCATION), PERMISSION_REQUEST_CODE)
    }
    override fun onRequestPermissionsResult(requestCode: Int,
permissions: Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
        if (requestCode == PERMISSION_REQUEST_CODE) {
            if ((grantResults.isNotEmpty()) && (grantResults[0] ==
PackageManager.PERMISSION_GRANTED)) {
                Toast.makeText(this, "Permiso de ubicación concedido",
Toast.LENGTH_SHORT).show()
                obtenerDistancia(null)
            }
            else {
                Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            }
        }
    }
    fun obtenerDistancia(view: View?) {
        txtUbicacion.text = "Consultando distancia..."
        if (!checkLocationPermission()) return
        val fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this)
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            return
        }
        val locationRequest =
LocationRequest.Builder(Priority.PRIORITY_HIGH_ACCURACY, 3000).build()
        val locationCallback = object : LocationCallback() {
            override fun onLocationResult(result: LocationResult) {
                val location = result.lastLocation
                if (location != null) {
                    val facultad = Location("").apply {
                        latitude = -34.9037905
                        longitude = -57.9378442
                    }
                    val distanciaMetros =
location.distanceTo(facultad)
                    txtUbicacion.text = String.format(
                        "Estás a %.2f kilómetros de la Facultad de
Informática\n(Latitud: %.6f, Longitud: %.6f)",
                        distanciaMetros/1000, location.latitude,
location.longitude
                    )
```

```
                }
                else {
                    txtUbicacion.text = "Ubicación no disponible"
                }
            }
        }
        fusedLocationClient.requestLocationUpdates(locationRequest,
locationCallback, Looper.getMainLooper())
    }
}
```