

## **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>
```

Ver proyecto “*TP1\_E1aE14*”.

## **Ejercicio 8.**

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

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

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

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?

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 *NuevaActivity.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 completamente 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 ocultada completamente 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?

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.

(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 `MainActivity.kt`:

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

En *activity\_main.xml*:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp">
    <TextView
        android:id="@+id/textoIdioma"
        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>
```

## **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 actividades 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 *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private var contador = 0
    private lateinit var txtContador: TextView
    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 *activity\_main.xml*:

```
<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>
```

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()
            }
        }
    }
}
```

En *activity\_operaciones.xml*:

```
<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>
```

## **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:

```
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*:

```
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)
    contador = savedInstanceState?.getInt("contador") ?: 0
    txtContador.text = contador.toString()
    btnOperacion.setOnClickListener {
        //val intent = Intent(this, Operaciones::class.java)
        val intent = Intent("com.example.tp1_e19ae22.OPERACIONES")
        startActivityForResult(intent, REQUEST_CODE)
    }
}

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)  
}
```

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.

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

```
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 textoDesdeIntent = intent.getStringExtra("texto")
        val textoRecuperado = savedInstanceState?.getString("texto")
        texto.text = textoRecuperado ?: textoDesdeIntent ?: 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 `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"
        }
    }
}

```

En *activity\_main.xml*:

```

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="24dp">
    <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*:

```

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

```

```
        android:orientation="vertical"
        android:padding="24dp">
        <TextView
            android:id="@+id/txtDia"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Esperando día..."
            android:textSize="20sp" />
    </LinearLayout>
```



## 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:background="#FF0000"
        android:textColor="#FFFFFF" />
</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:background="#FF0000"
        android:textColor="#FFFFFF"
        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:background="#FF0000"
        android:textColor="#FFFFFF"
        android:gravity="center_horizontal"
        android:textSize="24sp" />
</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:background="#FF0000"
        android:textColor="#FFFFFF"
        android:gravity="center_horizontal"
        android:textSize="24sp" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:text="Aquí Estoy"
        android:background="#0000FF"
        android:textColor="#FFFFFF"
        android:gravity="center_horizontal"
        android:textSize="20sp" />
</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:background="#FF0000"
        android:textColor="#FFFFFF"
        android:gravity="center_horizontal"
        android:textSize="24sp" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Aquí Estoy"
        android:background="#0000FF"
        android:textColor="#FFFFFF"
        android:gravity="center_horizontal"
        android:textSize="20sp" />
</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:background="#FF0000"
        android:textColor="#FFFFFF"
        android:gravity="center_horizontal"
        android:textSize="24sp" />
    <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:background="#0000FF"
        android:textColor="#FFFFFF"
        android:gravity="center_horizontal"
        android:textSize="20sp" />
</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:textSize="30sp"
        android:textColor="#FFFFFF"
        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?*

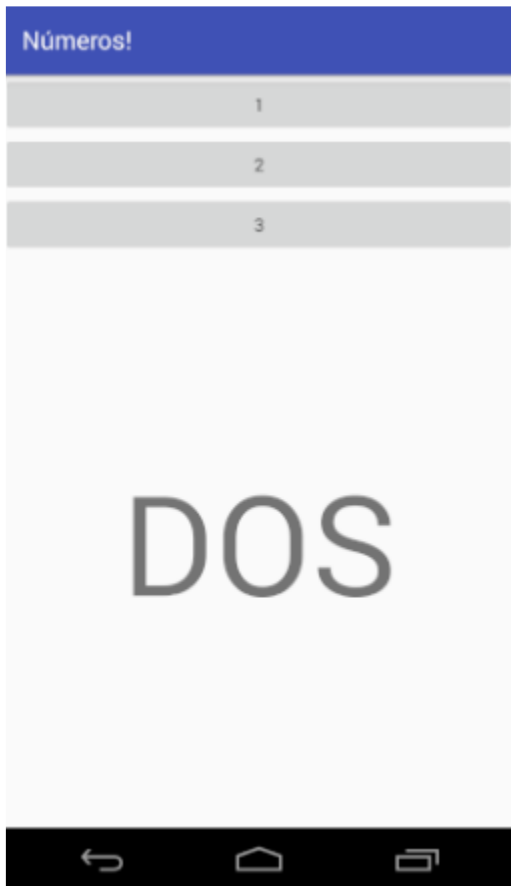
No, esta problemática existe en cualquier sistema con múltiples resoluciones. Por ejemplo:

- 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:textSize="20sp"
        android:textColor="#FFFFFF"
        android:background="#3F51B5"
        android:gravity="center_vertical"
        android:padding="16dp"
        android:layout_marginBottom="6dp" />
    <Button
        android:id="@+id/button1"
        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/button2"
        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/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="3"
        android:textColor="#000000"
        android:background="#888888" />
    <TextView
        android:id="@+id/textViewResult"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:textSize="100sp"
        android:textColor="#888888"
        android:gravity="center" />
</LinearLayout>
```

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

En *MainActivity.kt*:

```
package com.example.tp2_e11

import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.TextView
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

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 textViewResult =
                    findViewById<TextView?>(R.id.textViewResult)
                val button1 = findViewById<Button?>(R.id.button1)
                val button2 = findViewById<Button?>(R.id.button2)
                val button3 = findViewById<Button?>(R.id.button3)
                button1.setOnClickListener(object : View.OnClickListener {
                    override fun onClick(v: View?) {
                        textViewResult.setText("UNO")
                    }
                })
                button2.setOnClickListener(object : View.OnClickListener {
                    override fun onClick(v: View?) {
                        textViewResult.setText("DOS")
                    }
                })
                button3.setOnClickListener(object : View.OnClickListener {
                    override fun onClick(v: View?) {
                        textViewResult.setText("TRES")
                    }
                })
            }
        }
    }
}

```

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

En MainActivity.kt:

```

package com.example.tp2_e11

import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.TextView
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

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

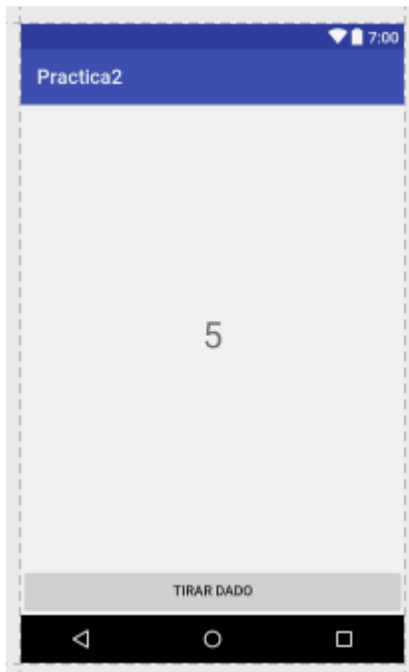
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {

```

```
v, insets ->
    val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
    v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
    insets
}
val textViewResult =
findViewById<TextView?>(R.id.textViewResult)
val button1 = findViewById<Button?>(R.id.button1)
val button2 = findViewById<Button?>(R.id.button2)
val button3 = findViewById<Button?>(R.id.button3)
val clickListener = View.OnClickListener { v ->
    when (v?.id) {
        R.id.button1 -> textViewResult.text = "UNO"
        R.id.button2 -> textViewResult.text = "DOS"
        R.id.button3 -> textViewResult.text = "TRES"
    }
}
button1.setOnClickListener(clickListener)
button2.setOnClickListener(clickListener)
button3.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:textSize="20sp"
        android:textColor="#FFFFFF"
        android:background="#3F51B5"
        android:gravity="center_vertical"
        android:padding="16dp" />
    <TextView
        android:id="@+id/textViewResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="50sp"
```



```

        android:textColor="#888888"
        android:layout_centerInParent="true" />
    <Button
        android:id="@+id/buttonTirarDado"
        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*:

```

package com.example.tp2_e12

import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.TextView
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import java.util.Random

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 textViewResult =
                findViewById<TextView?>(R.id.textViewResult)
            val buttonTirarDado =
                findViewById<Button?>(R.id.buttonTirarDado)
            val random = Random()
            buttonTirarDado!!.setOnClickListener(object :
                View.OnClickListener {
                    override fun onClick(v: View?) {
                        val num: Int = (1..6).random()
                        textViewResult!!.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:textSize="20sp"
        android:textColor="#FFFFFF"
        android:background="#800000"
        android:gravity="center_vertical"
        android:padding="16dp"
        android:layout_marginBottom="6dp" />
    <TextView
        android:id="@+id/textViewImage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Facultad de Informática"
        android:textSize="30sp"
        android:textColor="#888888"
        android:textStyle="bold"
        android:gravity="center"
        android:layout_marginBottom="6dp" />
    <ImageView
        android:id="@+id/imageViewFacultad"
        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/buttonOcultar"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="OCULTAR"
            android:textSize="20sp"
            android:textColor="#000000"
            android:background="#888888"
            android:layout_marginStart="12dp" />
        <Button
            android:id="@+id/buttonMostrar"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="MOSTRAR"
            android:textSize="20sp"
            android:textColor="#000000"
            android:background="#888888"
            android:layout_marginStart="12dp"
            android:layout_marginEnd="12dp" />
    </LinearLayout>
</LinearLayout>

```

En *MainActivity.kt*:

```

package com.example.tp2_e13

import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.ImageView
import androidx.activity.enableEdgeToEdge

```

```

import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

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 imageViewFacultad =
findViewById<ImageView?>(R.id.imageViewFacultad)
            val buttonOcultar = findViewById<Button?>(R.id.buttonOcultar)
            val buttonMostrar = findViewById<Button?>(R.id.buttonMostrar)
            buttonOcultar.setOnClickListener {
                imageViewFacultad.visibility = View.GONE
                // imageViewFacultad.visibility = View.INVISIBLE
            }
            buttonMostrar.setOnClickListener {
                imageViewFacultad.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 *setVisibility(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.

## **Trabajo Práctico N° 3**

### **Ejercicio 1.**

## **Ejercicio 2.**

### **Ejercicio 3.**



## **Ejercicio 4.**

## **Ejercicio 5.**

## **Ejercicio 6.**

## **Ejercicio 7.**

## **Ejercicio 8.**

## **Ejercicio 9.**

## **Ejercicio 10.**

## **Ejercicio 11.**



## **Ejercicio 12.**

### **Ejercicio 13.**

## **Ejercicio 14.**

## **Ejercicio 15.**

## **Ejercicio 16.**

## **Ejercicio 17.**

## **Ejercicio 18.**

## **Ejercicio 19.**



## **Ejercicio 20.**

## **Ejercicio 21.**

## **Ejercicio 22.**

## **Ejercicio 23.**

## **Ejercicio 24.**

## **Ejercicio 25.**

## **Trabajo Práctico N° 4**

### **Ejercicio 1.**

## **Ejercicio 2.**



### **Ejercicio 3.**

## **Ejercicio 4.**

## **Ejercicio 5.**

## **Ejercicio 6.**

## **Ejercicio 7.**

## **Ejercicio 8.**

## **Ejercicio 9.**

## **Ejercicio 10.**



## **Ejercicio 11.**

## **Ejercicio 12.**

### **Ejercicio 13.**

## **Ejercicio 14.**

## **Ejercicio 15.**

## **Ejercicio 16.**

## **Ejercicio 17.**

## **Ejercicio 18.**



## **Ejercicio 19.**

## **Ejercicio 20.**

## **Ejercicio 21.**

## **Ejercicio 22.**

## **Ejercicio 23.**

## **Ejercicio 24.**

## **Ejercicio 25.**

## **Trabajo Práctico N° 5**

### **Ejercicio 1.**



## **Ejercicio 2.**

### **Ejercicio 3.**

## **Ejercicio 4.**

## **Ejercicio 5.**

## **Ejercicio 6.**

## **Ejercicio 7.**

## **Ejercicio 8.**

## **Ejercicio 9.**



## **Ejercicio 10.**

## **Ejercicio 11.**

## **Ejercicio 12.**

### **Ejercicio 13.**

## **Ejercicio 14.**

## **Ejercicio 15.**

## **Ejercicio 16.**

## **Ejercicio 17.**



## **Ejercicio 18.**

## **Ejercicio 19.**

## **Ejercicio 20.**

## **Ejercicio 21.**

## **Ejercicio 22.**

## **Ejercicio 23.**

## **Ejercicio 24.**

## **Ejercicio 25.**