

Trabajo Práctico N° 1

Ejercicio 1.

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

Ejercicio 2.

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

Ejercicio 3.

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

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 piensa 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>
```

Ejercicio 8.

Añadir el siguiente código en la clase MainActivity.java, 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.

- Se inicia la *app* en el emulador.
- Se muestra su *MainActivity* con un botón que dice “Click”.
- Se toca 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.

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?

(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.

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>
```

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.

Ejercicio 21.

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

Ejercicio 22.

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

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

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?

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

(c) ¿Qué sucede con la 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.