Evidencia dia 2 semana 14

Hilos en Android (threads)

Una aplicación Android se ejecuta siempre en un proceso del sistema operativo, todos los componentes de la App (por defecto) se ejecutan dentro de ese mismo proceso.

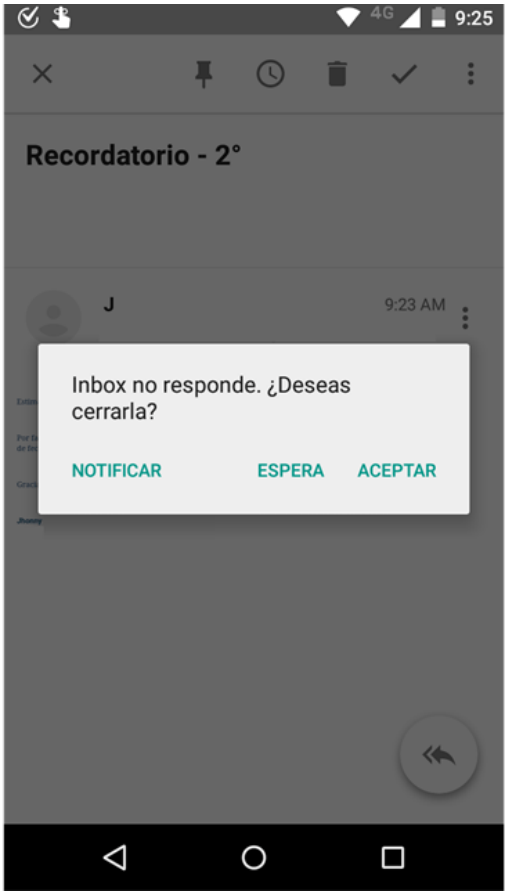
Una aplicación Android crea siempre un subproceso o un hilo principal de ejecución. Todo el código de nuestra aplicación se ejecuta dentro de este subproceso, esto incluye dibujar y actualizar la interfaz de usuario.

El hilo principal es el encargado de enviar los eventos a los widgets apropiados de la interfaz de usuario, así como la comunicación entre cada uno de los componentes.

Para mantener la respuesta de la aplicación, es esencial evitar el uso del hilo principal para realizar cualquier operación que pueda terminar bloqueandola.

Procesos y subprocesos en Android

¿Qué sucede si se ejecuta una operación muy pesada, de muy larga duración dentro del hilo principal?: La aplicación se bloquea y no responde a la interacción del usuario (error ANR «aplicación no responde»), y se muestra un mensaje similar a esta imagen:



.

Para evitar estos errores, seguimos dos reglas:

No bloquear el hilo principal.

Todas las operaciones de interfaz gráfica se hacen en el hilo principal.

Android ofrece muchas maneras para crear y administrar hilos, y existen librerías de terceros que hacen que la gestión de hilos en Android sea mucha más sencilla y agradable para el programador.

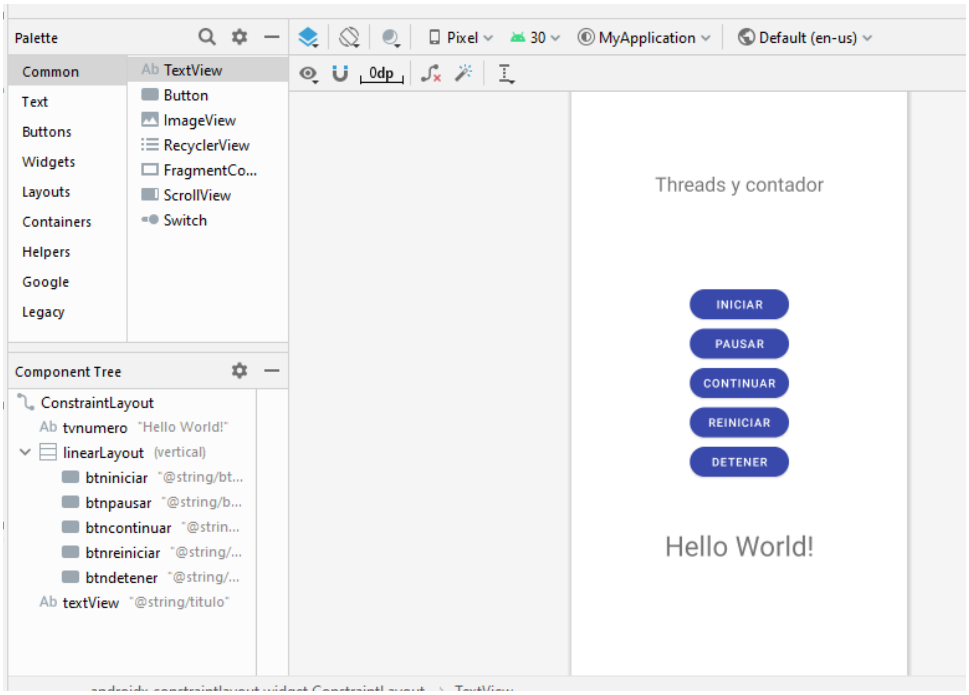
Las operaciones de red, y las llamadas a la base de datos, servicios web y la carga de algunos componentes (Ej. imágenes de internet), son ejemplos comunes de operaciones que uno como programador debe evitar hacer en el hilo principal.

En Android, se puede categorizar todos los componentes de subprocesamiento en dos categorías:

1. **Subprocesos que están adjuntos a una actividad / fragmento:** estos subprocesos están vinculados al ciclo de vida de la actividad / fragmento y finalizan tan pronto como se destruye la actividad /fragmento.
2. **Subprocesos que no están adjuntos a ninguna actividad / fragmento:** estos subprocesos pueden seguir ejecutándose más allá de la duración de la actividad / fragmento (si corresponde) de la que se generaron.

Android permite crear hilos o subprocesos adicionales, para ejecutar tareas de larga duración. Una vez terminada la tarea podemos regresar al hilo principal para actualizar la interfaz de usuario.

Diseño Interfaz Threads



Layout ActivityMain.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tvnumero"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:textSize="34sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout"
        app:layout_constraintVertical_bias="0.275" />

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <Button
            android:id="@+id/btniniciar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:backgroundTint="#3949AB"
            android:text="@string/btn1"
            app:cornerRadius="20dp" />

        <Button
            android:id="@+id/btnpausar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:backgroundTint="#3949AB"
            android:text="@string/btn2"
            app:cornerRadius="20dp" />

        <Button
            android:id="@+id/btncontinuar"
            android:layout_width="match_parent"
```

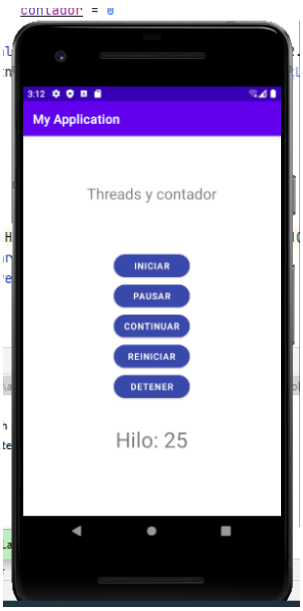
```
            android:layout_height="wrap_content"
            android:backgroundTint="#3949AB"
            android:text="@string/btn3"
            app:cornerRadius="20dp" />

        <Button
            android:id="@+id/btnreiniciar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:backgroundTint="#3949AB"
            android:text="@string/btn4"
            app:cornerRadius="20dp" />

        <Button
            android:id="@+id/btndetener"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:backgroundTint="#3949AB"
            android:text="@string/btn5"
            app:cornerRadius="20dp" />
    </LinearLayout>

    <TextView
        android:id="@+id/textView"
        android:layout_width="321dp"
        android:layout_height="156dp"
        android:gravity="center"
        android:text="@string/titulo"
        android:textSize="24sp"
        android:textStyle="normal"
        app:layout_constraintBottom_toTopOf="@+id/linearLayout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Método kotlin ActivityMain

```
class MainActivity : AppCompatActivity() {

    var contador = 0
    var estado = true
    var pausa = false

    override fun onCreate(savedInstanceState: Bundle?) {

        val hilo = Hilo(this)

        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

```
        val tvnumero = findViewById<TextView>(R.id.tvnumero)
        val btniniciar = findViewById<Button>(R.id.btniniciar)
        btniniciar.setOnClickListener {
            hilo.start()
        }
        val btnpausar = findViewById<Button>(R.id.btnpausar)
        btnpausar.setOnClickListener {
            pausa = true
        }
        val btncontinuar = findViewById<Button>(R.id.btncontinuar)
        btncontinuar.setOnClickListener {
            pausa = false
        }
        val btnreiniciar = findViewById<Button>(R.id.btnreiniciar)
        btnreiniciar.setOnClickListener {
            contador = 0
        }
        val btndetener = findViewById<Button>(R.id.btndetener)
        btndetener.setOnClickListener setOnclikListener@{
            if (hilo.isAlive) {
                estado = false
                return@setOnclikListener
            }
        }
    }

    class Hilo(activity: MainActivity) : Thread() {
        var act = activity
        override fun run() {
            super.run()
            while (act.estado) {
                while (act.pausa == true) {
                    sleep(100)
                }
                sleep(100)
                act.runOnUiThread {
                    act.tvnumero.setText("Hilo: " + act.contador)
                }
                act.contador++
            }
        }
    }
}
```

Emulador



Reflexión: El día de hoy seguimos revisando nuevo contenido (Threads), un poco complicado de entender, pero que espero poder profundizar o comprender a medida que implementemos más métodos.