Instituto Politécnico Nacional

Omar Montoya Romero

7CM1

Desarrollo de Aplicaciones Móviles Nativas

Sistemas Computacionales

Practica III.- Aplicaciones nativas

Mtro. Efraín Arredondo Morales

28/09/2023

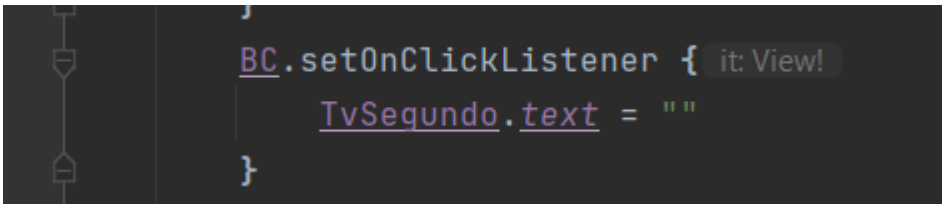
Primero se crearon botones con los respectivos nombres de los botones clásicos de una calculadora, despues de crear dichos botones se hace referencia a los botones de la vista.

```
21      lateinit var BC: Button
22      lateinit var BCE: Button
23      lateinit var BDivision: Button
24      lateinit var BSuma: Button
25      lateinit var BResta: Button
26      lateinit var BMultiplicacion: Button
27      lateinit var BPunto: Button
28      lateinit var BIgual: Button
29      lateinit var TvSegundo: TextView
30
31      override fun onCreate(savedInstanceState: Bundle?) {
32          super.onCreate(savedInstanceState)
33          setContentView(R.layout.activity_main)
34          requestedOrientation = ActivityInfo.SCREEN_ORIENTATION_PORTRAIT
35
36          B0= findViewById(R.id.button0)
37          B1= findViewById(R.id.button1)
38          B2= findViewById(R.id.button2)
39          B3= findViewById(R.id.button3)
40          B4= findViewById(R.id.button4)
41          B5= findViewById(R.id.button5)
42          B6= findViewById(R.id.button6)
```

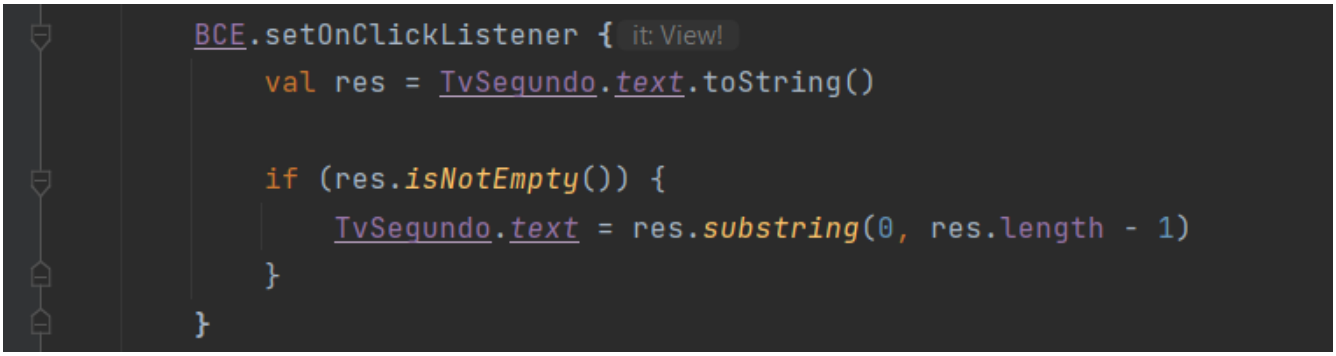
En los Botones del 0 al 9 es la misma función, esta recibe la cadena de texto que se encuentra en el textView una vez que lo recibe se le asigna al mismo textView, para asignárselo se hace una concatenación del valor que se encontraba en en el textview mas el texto del botón.

```
        B9.setOnClickListener { it: View!
            val res = TvSegundo.text.toString()
            TvSegundo.text = res + B9.text.toString()
        }
```

El botón C este tiene la funcionalidad de borrar todo el contenido el textView, donde se iguala a vacio esto para poder limpiaro.



Lo contrario del botón C es el CE donde este solo se encarga de borrar un digito del contenido del textView, hace la comparación de si esta vacio no hace nada, mientras tenga una cantidad este podrá hacer efecto de quitar un solo dijito.



El botón punto recibe la cadena del textView, hace la comparación que no exista otro botón, esto para evitar errores de exceso de puntos, no hay ninguno agrega el punto el cual es el texto del botón, pero donde ya se encuentre un botón este no agrega el botón.

El botón suma recibe el texto del text view, y recibe el ultimo digito del textView eexistente, hace la comparación de que no se repita ningún signo, esto para evitar repeticiones donde no tenga números en medio de cada signo. Esta lógica aplica para los otros botones de las operaciones restantes.

```
BSuma.setOnClickListener { it: View!
    val res = TvSegundo.text.toString()
    val last = res.lastOrNull()
    if (last != '-' && last != '+' && last != '*' && last != '/'&& last != '%') {
        TvSegundo.text = res + BSuma.text.toString()
    }
}
```

El botón igual recibe la expresión que se encuentra en el textView despues se manda llamar una función que recibe la expresión anteriormente obtenida. El código de las funciones del igual fue obtenido con ayuda del chat gpt ya que se intento de muchas maneras pero no funcionaban, crasheaba la app directamente, en este caso fue necesario apoyarse de esta materia para que la app pudiera jalar.

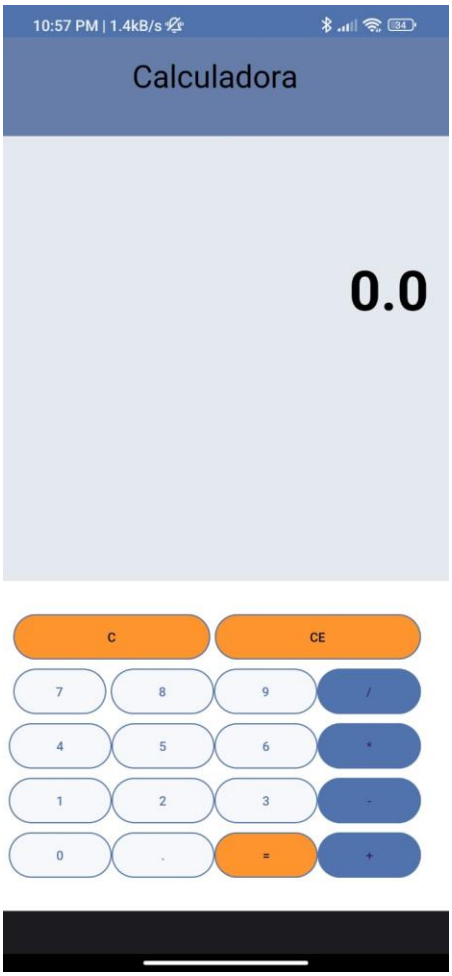
```
}
BIgual.setOnClickListener { it: View!
    val expresion = TvSegundo.text.toString()
    try {
        val resultado = evaluarExpresion(expresion)
        TvSegundo.text = resultado.toString()
    } catch (e: ArithmeticException) {
        TvSegundo.text = "Error: ${e.message}"
    } catch (e: Exception) {
        TvSegundo.text = "Error"
    }
}
```

V

```
fun evaluarExpresion(expresion: String): Double {
    val operadores = mutableListOf<Char>() // Lista de operadores
    val numeros = mutableListOf<Double>()  // Lista de números

    var numTemporal = ""  // Variable temporal para construir números
    for (caracter in expresion) { // Iterar sobre cada caracter de la expresión
        if (caracter.isDigit() || caracter == '.') { // Si el caracter es dígito o punto decimal
            numTemporal += caracter  // Construir el número temporal
        } else { // Si el caracter es un operador
            if (numTemporal.isNotEmpty()) {
                numeros.add(numTemporal.toDouble()) // Agregar el número construido a la lista de números
                numTemporal = ""  // Reiniciar el número temporal
            }

            // Resolver operaciones con prioridad mayor o igual
            while (operadores.isNotEmpty() && prioridadOperador(operadores.last()) >= prioridadOperador(caracter)) {
                val operador = operadores.removeAt( index: operadores.size - 1)
                val num2 = numeros.removeAt( index: numeros.size - 1)
                val num1 = numeros.removeAt( index: numeros.size - 1)
                numeros.add(aplicarOperacion(operador, num1, num2))
            }
            operadores.add(caracter) // Agregar operador a la lista de operadores
        }
    }
    if (numTemporal.isNotEmpty()) {
        numeros.add(numTemporal.toDouble()) // Agregar el último número construido
    }
    // Resolver cualquier operación restante
    while (operadores.isNotEmpty()) {
        val operador = operadores.removeAt( index: operadores.size - 1)
        val num2 = numeros.removeAt( index: numeros.size - 1)
        val num1 = numeros.removeAt( index: numeros.size - 1)
        numeros.add(aplicarOperacion(operador, num1, num2))
    }

    return numeros[0] // Devolver el resultado final
}

// Función para aplicar una operación a dos números
fun aplicarOperacion(operador: Char, num1: Double, num2: Double): Double {
    if (operador == '/' && num2 == 0.0) {
        throw ArithmeticException("No se puede dividir entre 0")
    }
    return when (operador) { // Determinar el tipo de operación
        '+' -> num1 + num2
        '-' -> num1 - num2
        '*' -> num1 * num2
        '/' -> num1 / num2
        else -> throw IllegalArgumentException("Operador no válido") // Lanzar error si el operador es inválido
    }
}
```

```
// Función para determinar la prioridad de un operador
fun prioridadOperador(operador: Char): Int {
    return when (operador) { // Asignar prioridades a los operadores
        '+', '-' -> 1
        '*', '/' -> 2
        else -> 0
    }
}
```

Resultado de la aplicación:
Este es el diseño final de la calculadora, los colores están basados a la imagen que se encuentra en el Moodle, los botones tienen un fondo especial, Este diseño fue complicado de armar ya que no se quería acomodar correctamente, además de que no se adapta al dispositivo se tuvo que modificar mucho para que le quede a un celular en especifico.



Este diseño fue complicado de hacer y de acomodar los diferentes layout para los botones y el text view, y el problema mas grande es el igual ya que no salió y se tuvo que pedir ayuda externa al chatGPT.