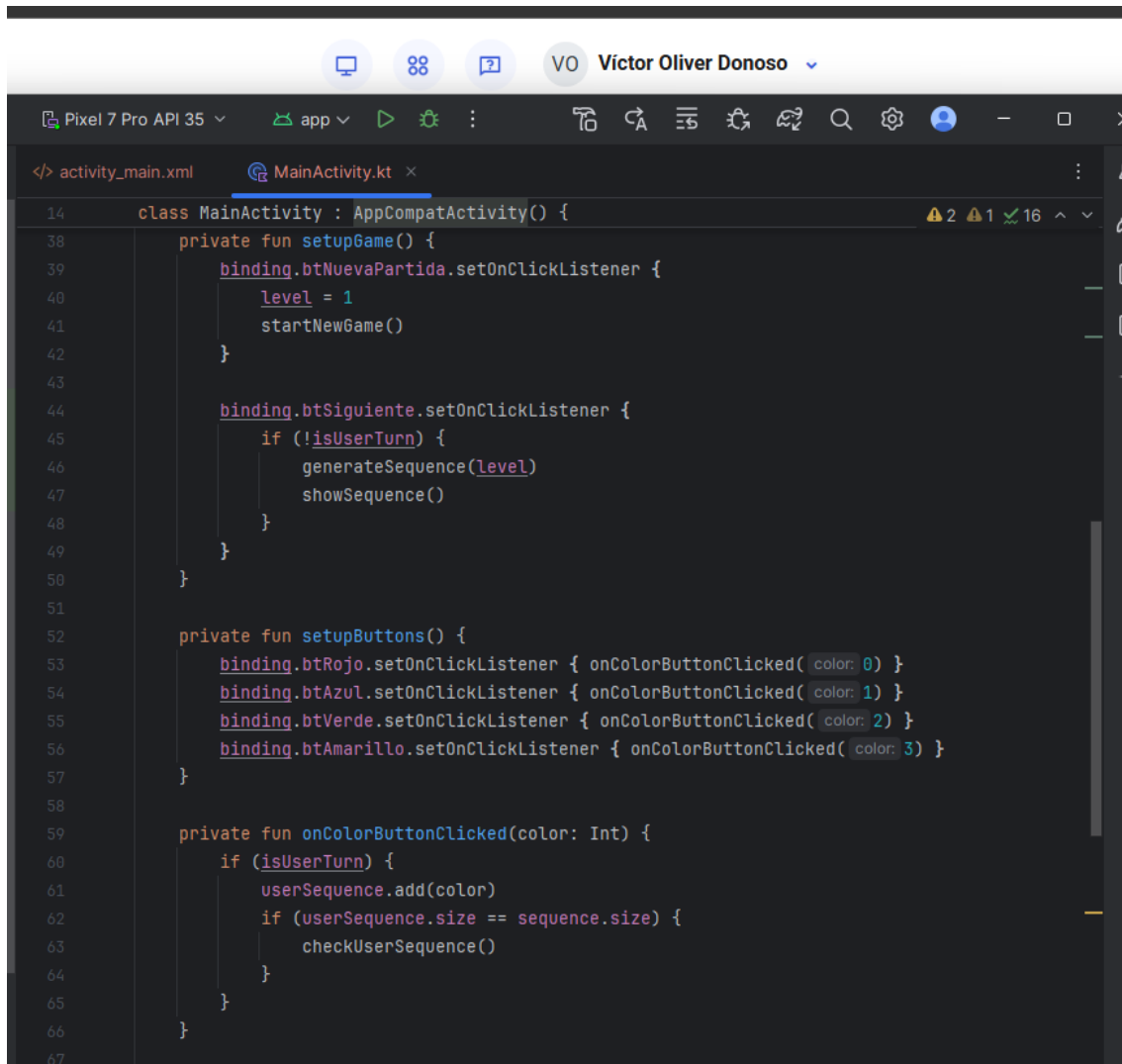


SIMON DICE

MAIN

```
1 package com.example.simondice
2
3 import android.graphics.Color
4 import android.os.Bundle
5 import android.os.Handler
6 import android.widget.Toast
7 import androidx.activity.enableEdgeToEdge
8 import androidx.appcompat.app.AppCompatActivity
9 import androidx.core.view.ViewCompat
10 import androidx.core.view.WindowInsetsCompat
11 import com.example.simondice.databinding.ActivityMainBinding
12 import kotlin.random.Random
13
14 class MainActivity : AppCompatActivity() {
15     lateinit var binding: ActivityMainBinding
16     private val sequence = mutableListOf<Int>()
17     private val userSequence = mutableListOf<Int>()
18     private var level = 1
19     private var isUserTurn = false
20
21     override fun onCreate(savedInstanceState: Bundle?) {
22         super.onCreate(savedInstanceState)
23         enableEdgeToEdge()
24
25         binding = ActivityMainBinding.inflate(layoutInflater)
26         setContentView(binding.root)
27
28         ViewCompat.setOnApplyWindowInsetsListener(binding.main) { v, insets ->
29             val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
30             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
31             insets
32         }
33
34         setupGame()
35         setupButtons()
36     }
37 }
```



```
14 class MainActivity : AppCompatActivity() {
38     private fun setupGame() {
39         binding.btNuevaPartida.setOnClickListener {
40             level = 1
41             startNewGame()
42         }
43
44         binding.btSiguiente.setOnClickListener {
45             if (!isUserTurn) {
46                 generateSequence(level)
47                 showSequence()
48             }
49         }
50     }
51
52     private fun setupButtons() {
53         binding.btRojo.setOnClickListener { onColorButtonClicked(color: 0) }
54         binding.btAzul.setOnClickListener { onColorButtonClicked(color: 1) }
55         binding.btVerde.setOnClickListener { onColorButtonClicked(color: 2) }
56         binding.btAmarillo.setOnClickListener { onColorButtonClicked(color: 3) }
57     }
58
59     private fun onColorButtonClicked(color: Int) {
60         if (isUserTurn) {
61             userSequence.add(color)
62             if (userSequence.size == sequence.size) {
63                 checkUserSequence()
64             }
65         }
66     }
67 }
```

VO Victor Oliver Donoso

Pixel 7 Pro API 35 app

activity_main.xml MainActivity.kt

```
class MainActivity : AppCompatActivity() {  
    private fun startNewGame() {  
        sequence.clear()  
        userSequence.clear()  
        isUserTurn = false  
        Toast.makeText(context: this, text: "Nueva Partida Iniciada", Toast.LENGTH_SHORT).show()  
    }  
  
    private fun generateSequence(level: Int) {  
        val colorsToShow = level + 2 //botones en cada nivel  
        for (i in 1..colorsToShow) {  
            sequence.add(Random.nextInt(until: 4)) //para la aleatoriedad de los botones  
        }  
    }  
  
    private fun showSequence() {  
        isUserTurn = false  
        userSequence.clear()  
  
        val handler = Handler()  
        var delay = 0L  
  
        sequence.forEach { color ->  
            handler.postDelayed({  
                flashButton(color)  
            }, delay)  
            delay += 1000  
        }  
  
        handler.postDelayed({  
            isUserTurn = true  
        }, delay)  
    }  
}
```

```
SD SimonDice main Pixel 7 Pro API 35 app
activity_main.xml MainActivity.kt
class MainActivity : AppCompatActivity() {
    private fun showSequence() {
        // delay
    }

    private fun flashButton(color: Int) {
        val button = when (color) {
            0 -> binding.btRojo
            1 -> binding.btAzul
            2 -> binding.btVerde
            else -> binding.btAmarillo
        }

        val originalColor = button.solidColor
        button.setBackgroundColor(Color.WHITE)
        Handler().postDelayed({
            button.setBackgroundColor(originalColor)
        }, delayMillis: 500)
    }

    private fun checkUserSequence() {
        isUserTurn = false
        if (userSequence == sequence) {
            Toast.makeText(context: this, text: ";Correcto! Nivel $level superado", Toast.LENGTH_SHORT).show()
            level++
            startNewGame()
            generateSequence(level)
            showSequence()
        } else {
            Toast.makeText(context: this, text: "Error en la secuencia. Inténtalo de nuevo.", Toast.LENGTH_SHORT).show()
            startNewGame()
        }
    }
}
```

FUNCIONANDO

