



Estácio

CAMPUS POLO AUSTIN - NOVA IGUAÇU – RJ

DESENVOLVIMENTO FULL STACK

Nível 3: Lidando Com Sensores em Dispositivos Móveis

Tutor(a): Altamira de Souza Queiroz

RPG0025 9001

2025/1

SALOMÃO ISAAC CARVALHO GARCIA

Lidando Com Sensores em Dispositivos Móveis

Nova Iguaçu

01/02/2025

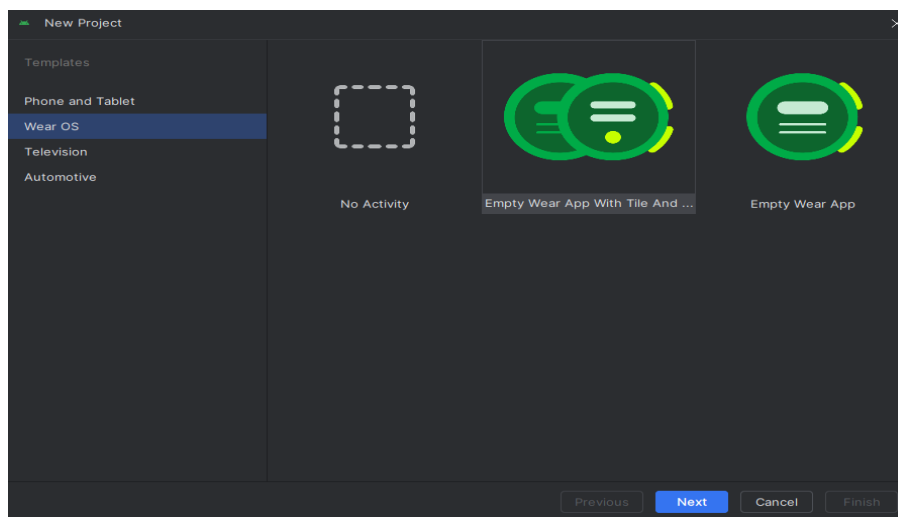
Lidando Com Sensores em Dispositivos Móveis

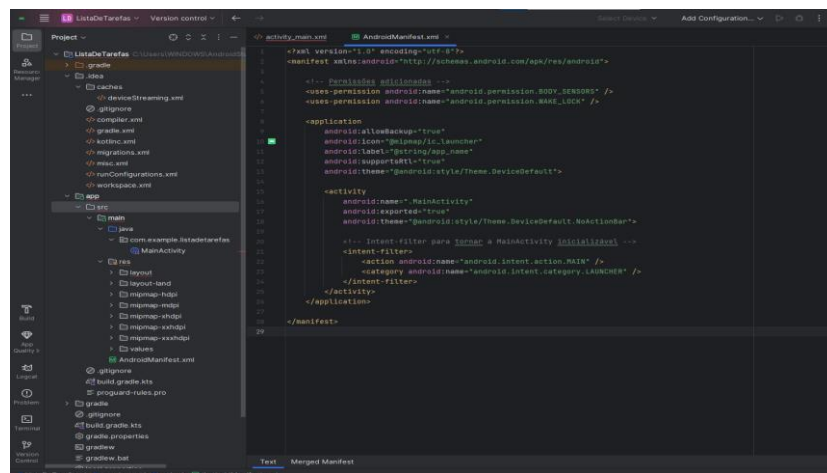
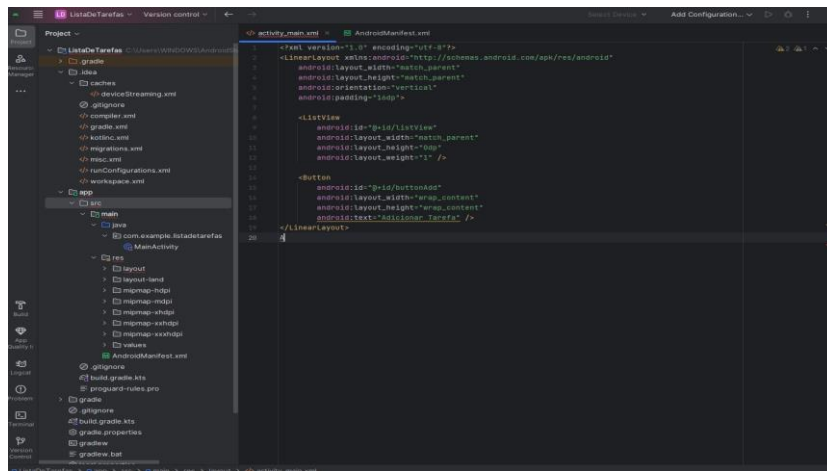
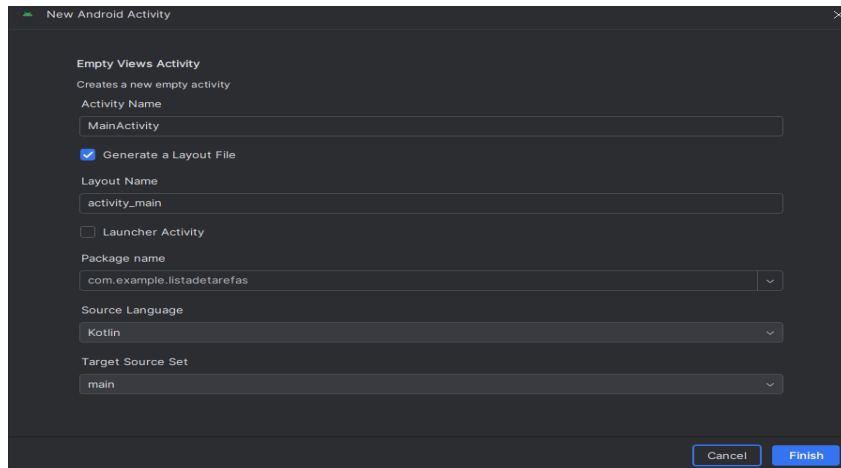
Objetivo da Prática: O objetivo desta prática é desenvolver habilidades práticas no uso de sensores em dispositivos móveis, focando na criação de aplicativos para Wear OS. A prática abrange a instalação do Android Studio, a configuração de emuladores e a construção de um aplicativo simples para Wear OS. Além disso, inclui a realização de capturas de tela tanto no emulador quanto no dispositivo real, utilizando um app complementar para testar e validar o aplicativo. A prática visa proporcionar uma compreensão dos processos de desenvolvimento de aplicativos móveis, como a interação com sensores e o teste em dispositivos virtuais, essenciais para a criação de soluções eficientes e funcionais no contexto de dispositivos móveis wearables.

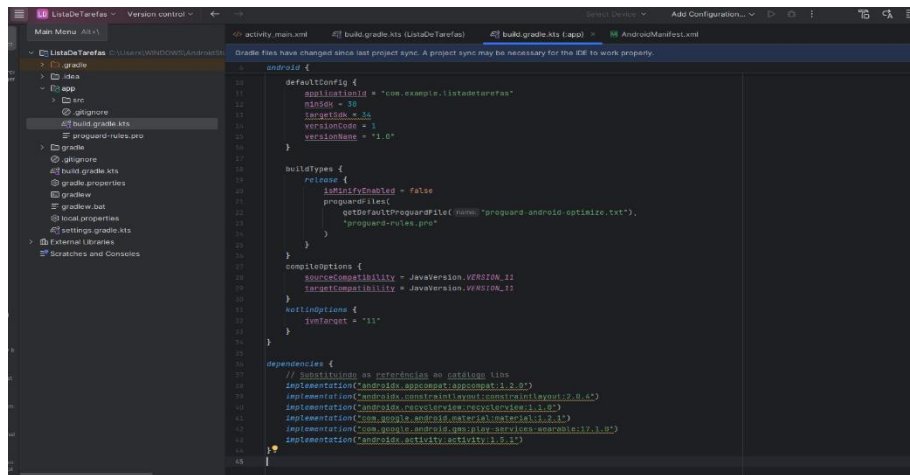
<https://github.com/SaloGarcia/Miss-o-Pr-tica-N-vel-3-Mundo-4.git>

CÓDIGOS

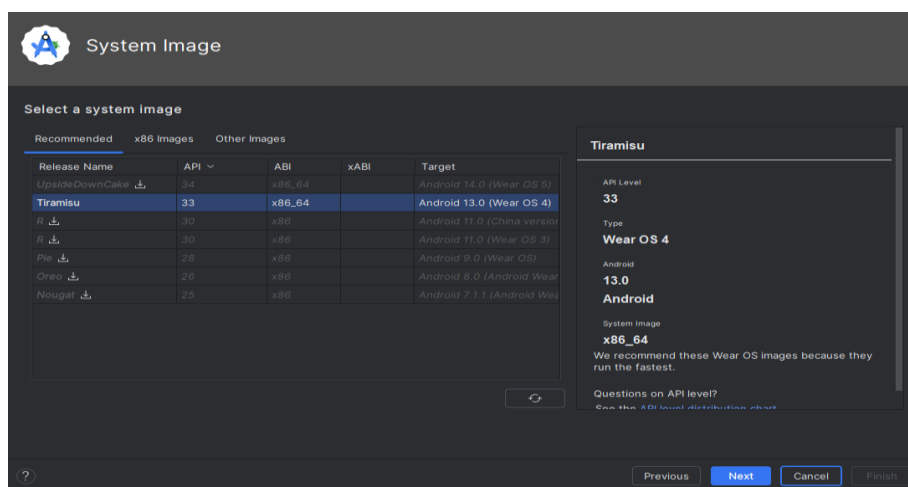
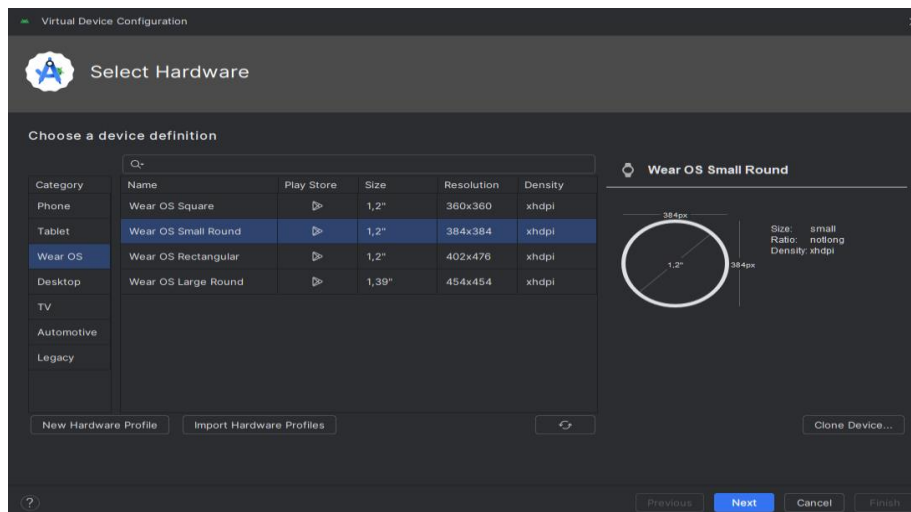
Criando um novo projeto no Android Studio

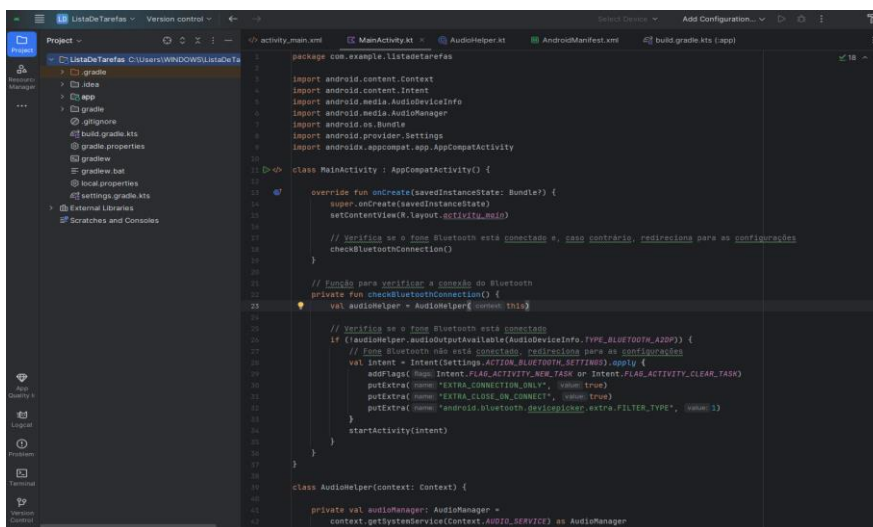
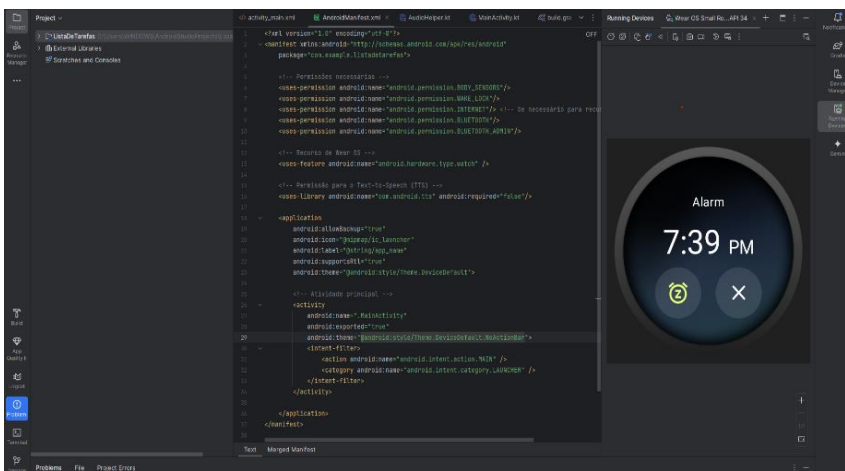
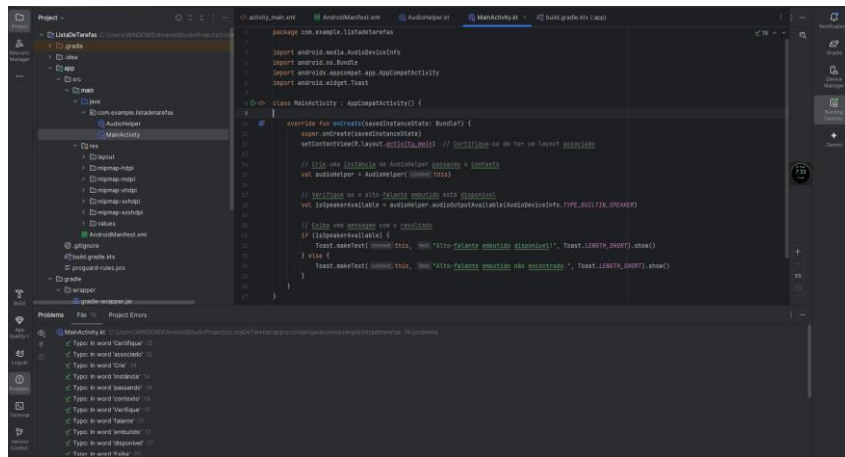


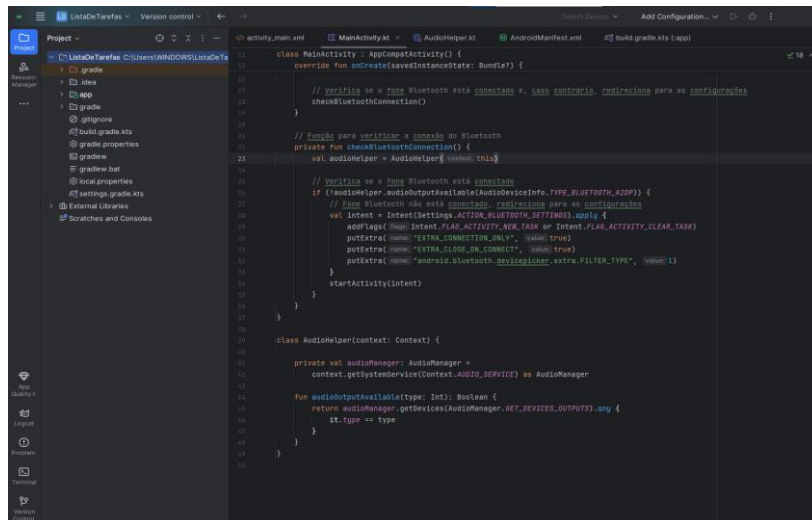




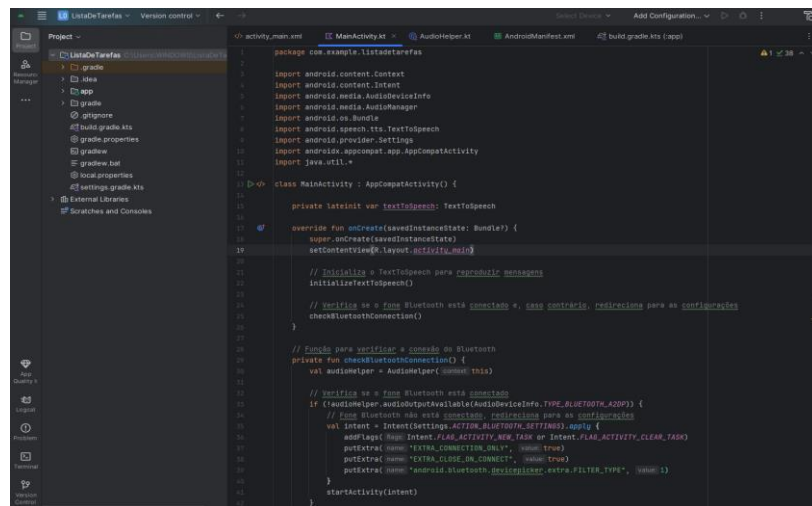
Criando um emulador



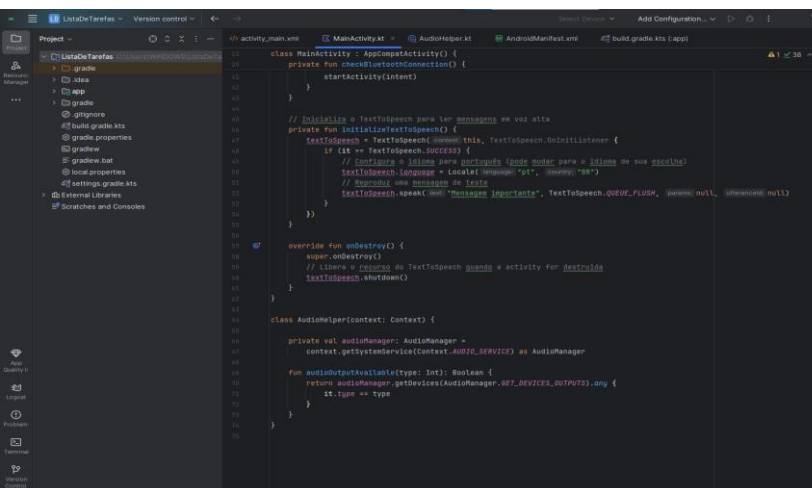




```
1 class MainActivity : AppCompatActivity() {
2     override fun onCreate(savedInstanceState: Bundle?) {
3         // Verifica se o fone Bluetooth está conectado e, caso contrário, redireciona para as configurações
4         checkBluetoothConnection()
5     }
6
7     // Função para verificar o conexão do Bluetooth
8     private fun checkBluetoothConnection() {
9         val audioHelper = AudioHelper()
10
11         // Verifica se o fone Bluetooth está conectado
12         if (audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_A2DP)) {
13             // Fone Bluetooth está conectado, redirecionando para as configurações
14             val intent = Intent(Settings.ACTION_BLUETOOTH_SETTINGS).apply {
15                 addFlags(Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK)
16                 putExtra(Intent.EXTRA_CONNECTION_ONLY, true)
17                 putExtra(Intent.EXTRA_CLOSE_ON_CONNECT, true)
18                 putExtra(Intent.EXTRA_Bluetooth_Developer.extra.FILTER_TYPE, 0)
19             }
20             startActivity(intent)
21         }
22     }
23
24     class AudioHelper(context: Context) {
25         private val audioManager: AudioManager =
26             context.getSystemService(Context.AUDIO_SERVICE) as AudioManager
27
28         fun audioOutputAvailable(type: Int): Boolean {
29             return audioManager.getDevices(AudioManager.GET_DEVICES_OUTPUTS).any {
30                 it.type == type
31             }
32         }
33     }
34 }
```



```
1 package com.example.listaatarefas
2
3 import android.content.Context
4 import android.content.Intent
5 import android.media.AudioDeviceInfo
6 import android.media.AudioManager
7 import android.os.Bundle
8 import android.speech.tts.TextToSpeech
9 import android.provider.Settings
10 import androidx.appcompat.app.AppCompatActivity
11 import java.util.*
12
13 class MainActivity : AppCompatActivity() {
14     private lateinit var textToSpeech: TextToSpeech
15
16     override fun onCreate(savedInstanceState: Bundle?) {
17         super.onCreate(savedInstanceState)
18         setContentView(R.layout.activity_main)
19
20         // Inicializa o TextToSpeech para reproduzir mensagens
21         initializeTextToSpeech()
22
23         // Verifica se o fone Bluetooth está conectado e, caso contrário, redireciona para as configurações
24         checkBluetoothConnection()
25     }
26
27     // Função para verificar o conexão do Bluetooth
28     private fun checkBluetoothConnection() {
29         val audioHelper = AudioHelper()
30
31         // Verifica se o fone Bluetooth está conectado
32         if (audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_A2DP)) {
33             // Fone Bluetooth está conectado, redirecionando para as configurações
34             val intent = Intent(Settings.ACTION_BLUETOOTH_SETTINGS).apply {
35                 addFlags(Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK)
36                 putExtra(Intent.EXTRA_CONNECTION_ONLY, true)
37                 putExtra(Intent.EXTRA_CLOSE_ON_CONNECT, true)
38                 putExtra(Intent.EXTRA_Bluetooth_Developer.extra.FILTER_TYPE, 0)
39             }
40             startActivity(intent)
41         }
42     }
43 }
```



```
1 class MainActivity : AppCompatActivity() {
2     private fun checkBluetoothConnection() {
3         startActivity(intent)
4     }
5
6     // Inicializa o TextToSpeech para ler mensagens em voz alta
7     private fun initializeTextToSpeech() {
8         textToSpeech = TextToSpeech(this, TextToSpeech.OnInitListener {
9             if (it == TextToSpeech.SUCCESS) {
10                 // Configura o idioma para português (pode mudar para o idioma de sua escolha)
11                 textToSpeech.language = Locale("pt", "BR")
12                 // Recupera um gerador de fala
13                 textToSpeech.speak(R.string.mensagem_importante, TextToSpeech.QUEUE_FLUSH, null, null)
14             }
15         })
16     }
17
18     override fun onDestroy() {
19         super.onDestroy()
20         // Libera o recurso de TextToSpeech quando a activity for destruída
21         textToSpeech.shutdown()
22     }
23
24     class AudioHelper(context: Context) {
25         private val audioManager: AudioManager =
26             context.getSystemService(Context.AUDIO_SERVICE) as AudioManager
27
28         fun audioOutputAvailable(type: Int): Boolean {
29             return audioManager.getDevices(AudioManager.GET_DEVICES_OUTPUTS).any {
30                 it.type == type
31             }
32         }
33     }
34 }
```