



Estácio

DGT2819 - Interação com sensores de smartphones e wearebles

João Guilherme Fernandes Borba - 202208515835

Polo Ipiranga Porto Alegre - RS

Interação c/ sensores de smartphones e we – 9001 – 7

Link: [GitHub](#)

DGT2819 - Interação com sensores de smartphones e wearebles

Objetivo da Prática

O objetivo desta prática foi desenvolver um aplicativo para a plataforma Wear OS utilizando o Android Studio e a linguagem Kotlin. O projeto, contextualizado para a empresa "Doma", visou criar uma ferramenta de assistência para funcionários com necessidades especiais, focando na utilização de recursos de áudio do dispositivo. As principais competências desenvolvidas foram:

- Configuração de um projeto Wear OS: Aprender a criar um novo projeto no Android Studio, especificamente para o sistema operacional de relógios, configurando o SDK mínimo e as dependências necessárias.
- Gerenciamento de Permissões: Adicionar as permissões essenciais no arquivo AndroidManifest.xml para permitir que o aplicativo acesse os sensores do dispositivo e gerencie o estado de energia.
- Detecção de Saídas de Áudio: Implementar a lógica para enumerar e verificar a disponibilidade de diferentes saídas de áudio, como o alto-falante interno e fones de ouvido Bluetooth, utilizando o AudioManager do Android.
- Resposta a Eventos do Sistema: Utilizar callbacks (AudioDeviceCallback) para detectar dinamicamente a conexão e

desconexão de dispositivos de áudio, permitindo que o aplicativo reaja em tempo real a essas mudanças.

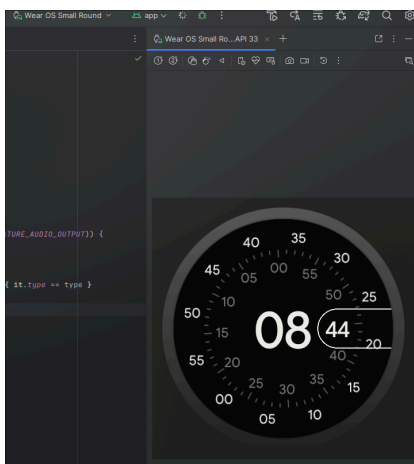
- **Interação com Configurações do Sistema:** Implementar a funcionalidade para direcionar o usuário às configurações de Bluetooth do sistema operacional a partir de um botão na interface do aplicativo, facilitando o pareamento de novos dispositivos.

A implementação do aplicativo para a empresa "Doma" foi guiada pelo roteiro prático, focando na criação da lógica de back-end para o gerenciamento de áudio, que é a base para as funcionalidades de assistência propostas. Os passos seguidos para atingir o objetivo foram:

1. **Criação e Configuração do Projeto:** O primeiro passo foi criar um novo projeto no Android Studio, selecionando o template "Wear OS" e "No Activity". Em seguida, o arquivo `AndroidManifest.xml` foi configurado com as permissões `BODY_SENSORS` e `WAKE_LOCK`, além das tags
2. **Implementação do AudioHelper:** Para organizar a lógica de detecção de áudio, foi criada uma classe auxiliar chamada `AudioHelper`. Esta classe encapsula a funcionalidade que verifica, através do `PackageManager` e `AudioManager`, se uma saída de áudio específica (como alto-falante ou Bluetooth) está disponível no dispositivo.
3. **Desenvolvimento da MainActivity:** A tela principal do aplicativo foi criada para abrigar a lógica de detecção dinâmica e a interface do usuário. No método `onCreate`, foi implementado o `registerAudioDeviceCallback`, um "ouvinte" que é notificado pelo sistema sempre que um dispositivo de áudio é conectado ou desconectado. Isso permite que o app saiba, por exemplo, quando um fone Bluetooth foi pareado para direcionar o áudio para ele.
4. **Criação da Interface de Usuário:** O layout da `MainActivity` (`activity_main.xml`) foi projetado de forma simples, contendo um único botão. Este botão foi programado para criar e disparar uma `Intent` que leva o usuário diretamente para a tela de configurações de Bluetooth do Wear OS. Essa funcionalidade é um requisito importante do projeto para facilitar a conexão de fones de ouvido.

5. Testes e Validação: Após a implementação, o aplicativo foi executado em um emulador Wear OS. Os testes confirmaram que o app instalava corretamente, que o botão abria a tela de configurações do sistema como esperado e que não ocorriam erros durante a execução, validando a implementação do código.

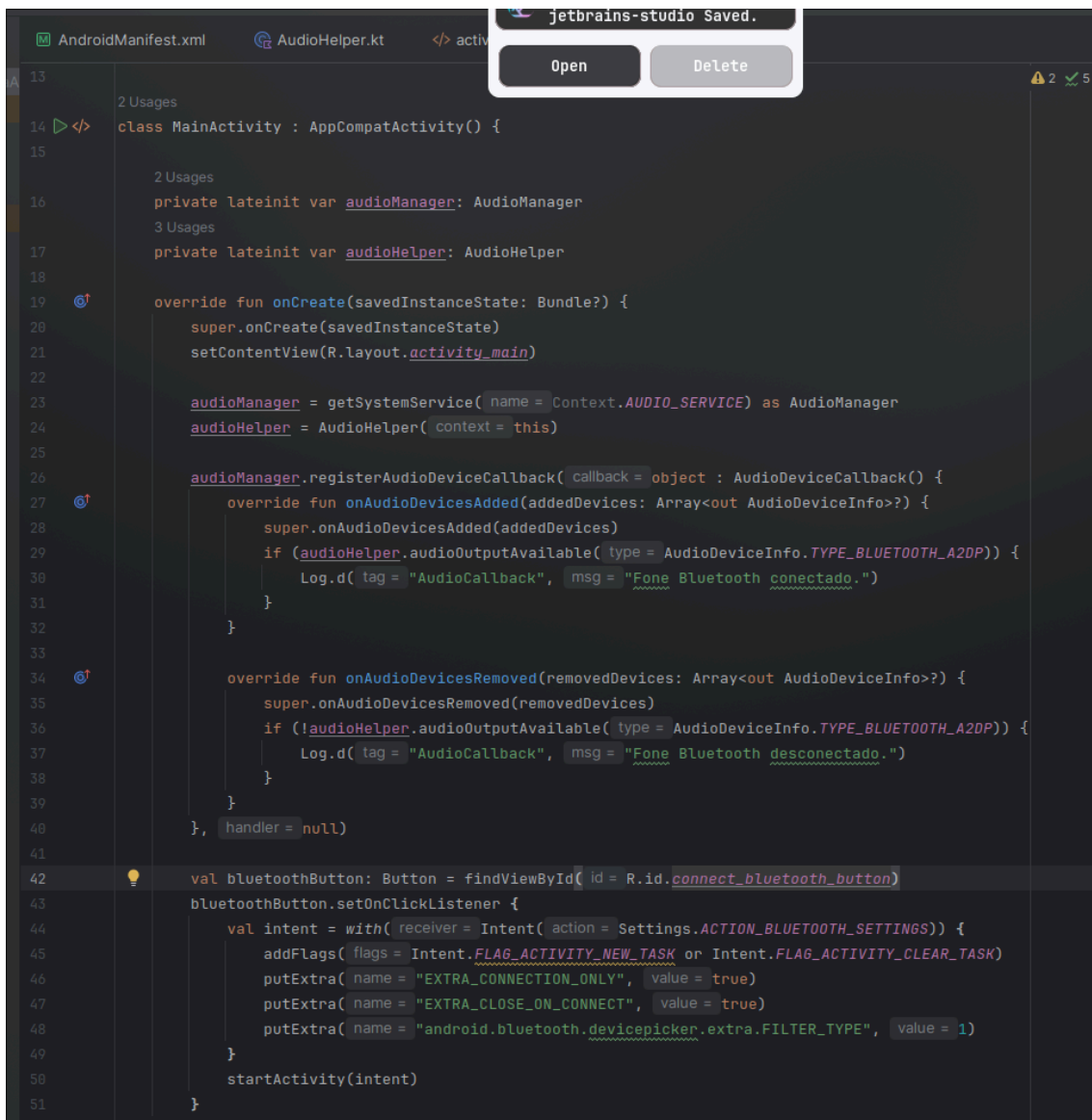
Print 1: Tela Final do Aplicativo no Emulador



Print 2: Código da Classe AudioHelper

```
AndroidManifest.xml  AudioHelper.kt  activity_main.xml  MainActivity.kt
1 package com.example.domaassistencia
2
3 import android.content.Context
4 import android.media.AudioManager
5 import android.content.pm.PackageManager
6
7 2 Usages
8 class AudioHelper(val context: Context) {
9     1 Usage
10     private val audioManager: AudioManager =
11         context.getSystemService(name = Context.AUDIO_SERVICE) as AudioManager
12
13     2 Usages
14     fun audioOutputAvailable(type: Int): Boolean {
15         if (!context.packageManager.hasSystemFeature(featureName = PackageManager.FEATURE_AUDIO_OUTPUT)) {
16             return false
17         }
18         return audioManager.getDevices(flags = AudioManager.GET_DEVICES_OUTPUTS).any { it.type == type }
19     }
20 }
```

Print 3: Implementação do AudioDeviceCallback na MainActivity



```
13 2 Usages
14 class MainActivity : AppCompatActivity() {
15
16     2 Usages
17     private lateinit var audioManager: AudioManager
18
19     3 Usages
20     private lateinit var audioHelper: AudioHelper
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24         setContentView(R.layout.activity_main)
25
26         audioManager = getSystemService(Context.AUDIO_SERVICE) as AudioManager
27         audioHelper = AudioHelper(context = this)
28
29         audioManager.registerAudioDeviceCallback(callback = object : AudioDeviceCallback() {
30             override fun onAudioDevicesAdded(addedDevices: Array<out AudioDeviceInfo>?) {
31                 super.onAudioDevicesAdded(addedDevices)
32                 if (audioHelper.audioOutputAvailable(type = AudioDeviceInfo.TYPE_BLUETOOTH_A2DP)) {
33                     Log.d(tag = "AudioCallback", msg = "Fone Bluetooth conectado.")
34                 }
35             }
36
37             override fun onAudioDevicesRemoved(removedDevices: Array<out AudioDeviceInfo>?) {
38                 super.onAudioDevicesRemoved(removedDevices)
39                 if (!audioHelper.audioOutputAvailable(type = AudioDeviceInfo.TYPE_BLUETOOTH_A2DP)) {
40                     Log.d(tag = "AudioCallback", msg = "Fone Bluetooth desconectado.")
41                 }
42             }
43         }, handler = null)
44
45         val bluetoothButton: Button = findViewById(id = R.id.connect_bluetooth_button)
46         bluetoothButton.setOnClickListener {
47             val intent = with(receiver = Intent(action = Settings.ACTION_BLUETOOTH_SETTINGS)) {
48                 addFlags(flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK)
49                 putExtra(name = "EXTRA_CONNECTION_ONLY", value = true)
50                 putExtra(name = "EXTRA_CLOSE_ON_CONNECT", value = true)
51                 putExtra(name = "android.bluetooth.devicepicker.extra.FILTER_TYPE", value = 1)
52             }
53             startActivity(intent)
54         }
55     }
56 }
```