



Estácio

# Interação Com Sensores de Smartphones e Wearables

Trabalho prático Matheus Scarabelli do Nascimento -202404596761

Polo - Méier -Rio de Janeiro/RJ 9001 – 4º semestre

# Objetivo da Prática

O objetivo desta prática foi desenvolver um aplicativo Wear Os para ajudar funcionários com necessidades especiais, utilizando os recursos presentes no android studio.

Utilizando-se dos dispositivos de saída de áudio disponíveis no smartwatch , como os alto-falantes e dispositivos de Bluetooth.

Também foi implementada a detecção dinâmica de dispositivos de áudio que foram conectados ou desconectados durante o uso do app.

Projeto foi desenvolvido usando Kotlin e Jetpack Compose do Wear OS.

## Análise crítica de implementação

No começo do desenvolvimento foi requerido a configuração adequada do Wear Os, ajustes no AndroidManifest parar que a execução seja efetuada corretamente.

Depois , foi implementado uma classe auxiliar(AudioHelper) para acessar o Audiomanager e verificar os dispositivos de saída disponíveis conectados ao bluetooth.

Em terceiro foi implementado a função rememberAudiostream para interface.O audioDeviceCallback fez com que se detectasse a conexão e desconexão do dispositivo Bluetooth.

Por último , foi Feito um intent para abrir a tela de configuração Bluetooth do Wear OS.

Durante o desenvolvimento ,ocorreram varias dificuldades relacionadas a funções Composable e conflitos de componentes duplicados .

No fim, esse trabalho ajudou a entender melhor sobre aplicativos feito no Android Studio ,e ajudar a familiarizar-se com o seu uso.

# Responsável por definir a interface principal do dispositivo Wear OS

The screenshot shows the Android Studio interface with the project structure and code editor for a Wear OS application named "ListDeTarefas".

**Project Structure:**

- app/
  - manifests/
    - AndroidManifest.xml
  - kotlin/java/
    - com.example.listdetarefas/presentation/
      - MainScreen.kt
    - theme/
      - AudioState.kt
    - MainActivity.kt
    - rememberAudioState.kt
    - AudioHelper.kt
  - res/
    - (generated)
  - Gradle Scripts

**MainScreen.kt Code:**

```
1 package com.example.listdetarefas.presentation
2
3 import androidx.compose.foundation.layout.*
4 import androidx.compose.runtime.Composable
5 import androidx.compose.ui.Alignment
6 import androidx.compose.ui.Modifier
7 import androidx.compose.ui.text.style.TextAlign
8 import androidx.compose.ui.unit.dp
9 import androidx.wear.compose.material.MaterialTheme
10 import androidx.wear.compose.material.Button
11 import androidx.wear.compose.material.Text
12
13 @Composable
14 fun MainScreen(
15     hasSpeaker: Boolean,
16     hasBluetooth: Boolean,
17     onOpenBluetoothSettings: () -> Unit
18 ) {
19     Column(
20         modifier = Modifier.fillMaxSize(),
21         horizontalAlignment = Alignment.CenterHorizontally,
22         verticalArrangement = Arrangement.Center
23     ) {
24
25         Text(
26             text = """
27                 Speaker: $hasSpeaker
28                 Bluetooth: $hasBluetooth
29             """.trimIndent(),
30             textAlign = TextAlign.Center,
31             color = MaterialTheme.colors.primary
32         )
33
34         Spacer(modifier = Modifier.height(height = 8.dp))
35
36         Button(onClick = onOpenBluetoothSettings) {
37             Text(text = "Bluetooth")
38         }
39     }
40 }
```

**Device View:**

The device view shows a Wear OS Small Round device connected via USB. A message at the bottom indicates how to mirror the screen to a physical device.

To mirror a physical device, connect it via USB cable or over WiFi, click + and select the device from the list. You may also select the "Always show this screen when a physical device is connected" option in the Device Mirroring settings.

To launch a virtual device, click + and select the device from the list, or use the Device Manager.

**Bottom Bar:**

Windows taskbar showing various application icons.

Observa o estado dos dispositivos de áudio, detecta se existe um alto-falante, e Bluetooth conectado.

The screenshot shows the Android Studio interface with the code editor open to a file named `rememberAudioState.kt`. The code is written in Kotlin and uses the Compose library. It defines a function `rememberAudioState` that returns an `AudioState` object. This object is mutable and contains fields for speaker and Bluetooth connectivity. The code also registers an `AudioDeviceCallback` to listen for device additions and removals, updating the `state` variable accordingly. A tooltip on the right side of the screen provides instructions for mirroring devices via USB or connecting a virtual device.

```
import com.example.listdetarefas.AudioHelper

@Composable
fun rememberAudioState(audioHelper: AudioHelper): AudioState {
    val context = audioHelper.context
    val audioManager =
        context.getSystemService(serviceClass = AudioManager::class.java)

    var state by remember {
        mutableStateOf(
            AudioState(
                hasSpeaker = audioHelper.audioOutputAvailable(
                    type = AudioDeviceInfo.TYPE_BUILTIN_SPEAKER
                ),
                hasBluetooth = audioHelper.audioOutputAvailable(
                    type = AudioDeviceInfo.TYPE_BLUETOOTH_A2DP
                )
            )
        )
    }

    DisposableEffect(key = Unit) {
        val callback = object : AudioDeviceCallback() {
            override fun onAudioDevicesAdded(
                addedDevices: Array<out AudioDeviceInfo>
            ) {
                state = state.copy(
                    hasBluetooth = audioHelper.audioOutputAvailable(
                        type = AudioDeviceInfo.TYPE_BLUETOOTH_A2DP
                    )
                )
            }

            override fun onAudioDevicesRemoved(
                removedDevices: Array<out AudioDeviceInfo>
            ) {
                state = state.copy(
                    hasBluetooth = audioHelper.audioOutputAvailable(
                        type = AudioDeviceInfo.TYPE_BLUETOOTH_A2DP
                    )
                )
            }
        }

        audioManager.registerAudioDeviceCallback(callback, handler = null)

        onDispose {
            audioManager.unregisterAudioDeviceCallback(callback)
        }
    }

    return state
}
```

# Inicia o app,cria o audiohelper,observa mudanças de áudio,monta interface principal.

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "Android". It includes the "app" module with "manifests", "kotlin", and "com.example.listdetarefas" packages. The "com.example.listdetarefas" package contains "presentation", "theme", and "AudioHelper" components.
- MainActivity.kt:** The main code editor displays the implementation of the `MainActivity` class. It initializes `audioHelper` and `rememberAudioState`. It also handles the creation of a `WearApp` instance with `hasSpeaker` and `hasBluetooth` properties, and an `onOpenBluetoothSettings` function.
- Preview:** On the right, there's a preview window for "Wear OS Small Round" showing a minimalist UI with a speaker icon and a blue circular button.
- Bottom Status Bar:** The status bar at the bottom shows various system icons and the date "14/02/2026".

<a href="https://github.com/matheussdn/estacio/tree/main/mod4>ListDeTarefas