Lidando com sensores em dispositivos móveis

Pablo Emannoel Da Silva Santos, 202307014796

Polo de São Sebastião DF

Objetivo da Prática

-Instalação do Android Studio e do emulador;

-Criar um app para Wear OS;

-Executar um app no emulador;

-Fazer capturas de telas no Android Studio;

-Fazer capturas de telas com app complementar.

-3º Missão Prática

Resultados

Método onCreate que é o método principal, é chamado quando a atividade é criada , definindo a interface com o usuário e inicializando com os componentes como o audioHelper, audioManager, audioPlayer além de fazer uma verificação de dispositivos de audio e mostrando o resultado no console.

```
class MainActivity : AppCompatActivity() {                                    3  18
    override fun onCreate(savedInstanceState: Bundle?) {
        playButton.setOnClickListener {
            if (!mediaPlayer.isPlaying) {
                mediaPlayer.start()
            }
        }


        val isSpeakerAvailable = audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BUILTIN_SPEAKER)
        if (isSpeakerAvailable) {
            println("Alto-falante embutido disponível.")
        } else {
            println("Alto-falante embutido não disponível.")
        }


        val isBluetoothHeadsetConnected = audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_A2DP)
        if (isBluetoothHeadsetConnected) {
            println("Fone de ouvido Bluetooth conectado.")
        } else {
            println("Fone de ouvido Bluetooth não conectado.")
        }
```

Declaração do método registerAudioDeviceCallback para monitorar quando houver conexão de dispositivos de audio (onAudioDeviceAdded) e quando houver desconexão de dispositivos de audio(onAudioDeviceRemoved).

```
    private fun registerAudioDeviceCallback() {
        audioManager.registerAudioDeviceCallback(object : AudioDeviceCallback() {

            override fun onAudioDevicesAdded(addedDevices: Array<out AudioDeviceInfo>) {
                super.onAudioDevicesAdded(addedDevices)
                if (audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_A2DP)) {
                    println("Fone de ouvido Bluetooth foi conectado.")
                }
            }

            override fun onAudioDevicesRemoved(removedDevices: Array<out AudioDeviceInfo>) {
                super.onAudioDevicesRemoved(removedDevices)
                if (!audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_A2DP)) {
                    println("Fone de ouvido Bluetooth foi desconectado.")
                }
            }
        }, handler: null)
    }
```

Método onDestroy que é chamado quando a atividade é finalizada quando por exemplo fechamos o aplicativo.Sendo usado para liberar os recursos do media player.

```
    override fun onDestroy() {
        super.onDestroy()

        if (::mediaPlayer.isInitialized) {
            mediaPlayer.release()
        }
    }
}
```

Componente LinearLayout contendo um componente Button para quando o usuario clicar reproduzir o audio

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.co
    xmlns:app="http://schemas.android.com/apk/res-auto
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="reproduzir audio"
    android:id="@+id/button_play_audio"
    />
</LinearLayout>
```

Conclusão

No desenvolvimento do aplicativo AudioHelper foi possivel trabalhar com diversas tecnologias para o desenvolvimento de aplicativo android com Java e Kotlin, como componentes de audio (AudioManager e MediaPlayer), interface com usuario, monitoramento de dispositivo de audio e gerenciamento de recursos.