Trabalho Prático | DGT2816 Interação com sensores de smartphones e wearebles

Curso: Desenvolvimento Full Stack

Polo: Polo - Vila Nova - Itu - Sp

Matrícula: 202404575268

Aluno: Rafael Lima de Medeiros

Turma: 2025.4 (5º Semestre)

Data: 14/02/2026

Github:
Wear OS projeto - DOMA
https://github.com/Rafaldm/Doma-Wear-OS-Assistent

Lista de tarefas
https://github.com/Rafaldm/Lista-De-Tarefas

Objetivo: Aprender a utilizar Android Studio afim de Criar app Wear OS utilizando Android Studio, para funcionários com necessidades especiais, para melhor qualidade de vida e do dia a dia.

Como foi iniciado ?

1. Criei todo ambiente de trabalho conforme o conteúdo do Sway Cloud,

2. Fiz a instalação do Android Studio e sua configuração,

3. Criação do emulador juntamente com a criação da lista de tarefas,

4. Realizei a configuração dos arquivos e criação dos arquivos main acitivity,

5. utilizei kotlin, pois infelizmente obtive muitos problemas de compatibilidade, bugs e erros com .java,

Abaixo vão constar todos os passos, prints, codigos e problemas encontrados na criação do aplicativo.

Códigos e Prints - Lista de Tarefas

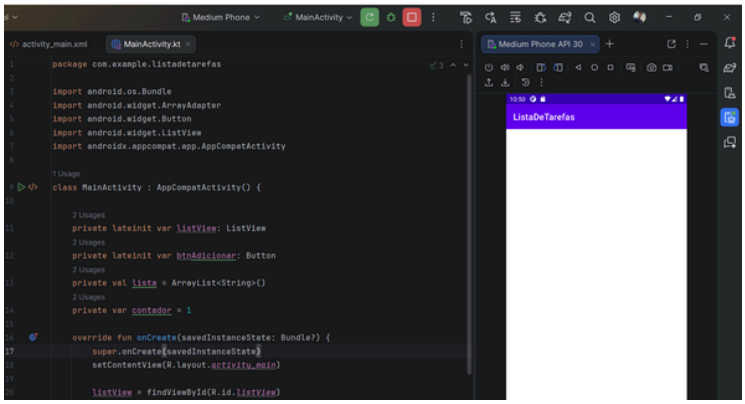
- **MainActivity.kt**

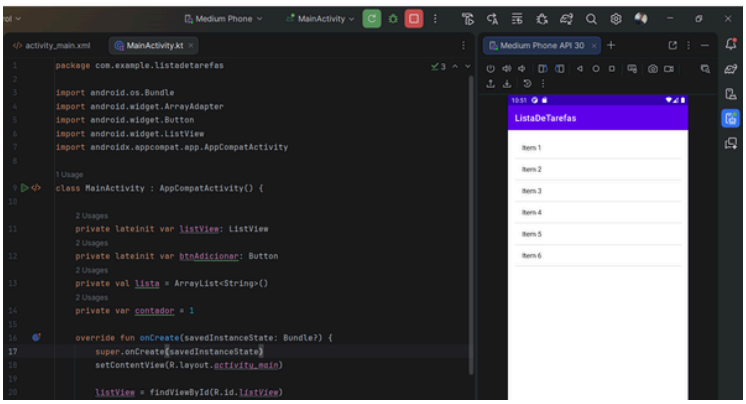
```
package com.example.listadetarefas

import android.os.Bundle
import android.widget.ArrayAdapter
import android.widget.Button
import android.widget.ListView
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private lateinit var listView: ListView
    private lateinit var btnAdicionar: Button
    private val lista = ArrayList<String>()
    private var contador = 1

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        listView = findViewById(R.id.listView)
        btnAdicionar = findViewById(R.id.btnAdicionar)

        val adapter = ArrayAdapter(this,
            android.R.layout.simple_list_item_1, lista)

        listView.adapter = adapter

        btnAdicionar.setOnClickListener {
            lista.add("Item $contador")
            contador++
            adapter.notifyDataSetChanged()
        }
    }
}
```

- **activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

  <ListView
      android:id="@+id/listView"
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_weight="1"/>

  <Button
      android:id="@+id/btnAdicionar"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Adicionar Item"/>
</LinearLayout>
```

Prints Lista de Tarefas





Capturas de tela App

- **Aqui estão as capturas de tela realizadas com o app Wear OS a partir da configuração das opções de desenvolvedor,**

Trabalho Prático - App Wear OS Doma

- **Resumo e problemas encontrados:**

Nesta etapa do projeto os desafios foram grandes, pois infelizmente os packs de SDK não tinham compatibilidade de áudio ao criar o emulador Wear OS, por falta do TTS (Text-to-Speech), tentei diversas formas para instalar outros pacotes alternativos e utilizando outras versões disponíveis, porém nenhuma delas permitia configurações e muito menos conexões pela playstore para que fosse instalado o TTS, tentei também a opção de conectar meu celular android via depuração USB, afim de instalar o app para que ele emitisse os áudios de aviso, porém sem sucesso, por falta de compatibilidade,

- **Melhor Resolução encontrada:**

Como ultima opção, resolvi prosseguir com o processo, utilizando logs, ou seja ao clicar nos botões, ações eram registradas via log no (LogCat), o que funcionou perfeitamente, validando todas as funcionalidades solicitadas, inclusive fiz a ligação das notificações diretamente com o relógio, que ao clicar no botão de notificações a mesma fica registrada na tela principal do Emulador (relógio), tambem criei um botão que direciona para as configurações de bluetooth,

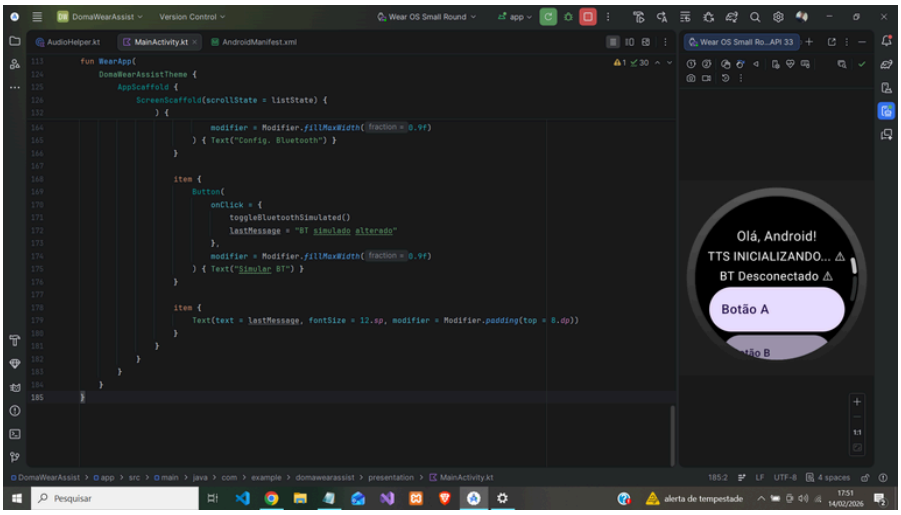
1. O Aplicativo nesta fase não reproduz os audios devido a falta de TTS, mas os logs provam que ele funciona ao registrar as ações via botão.

2. Funcionalidades de conexão bluetooth também foram implementadas via log, pois para realizar testes reais precisariamos de um aparelho real compátivel,

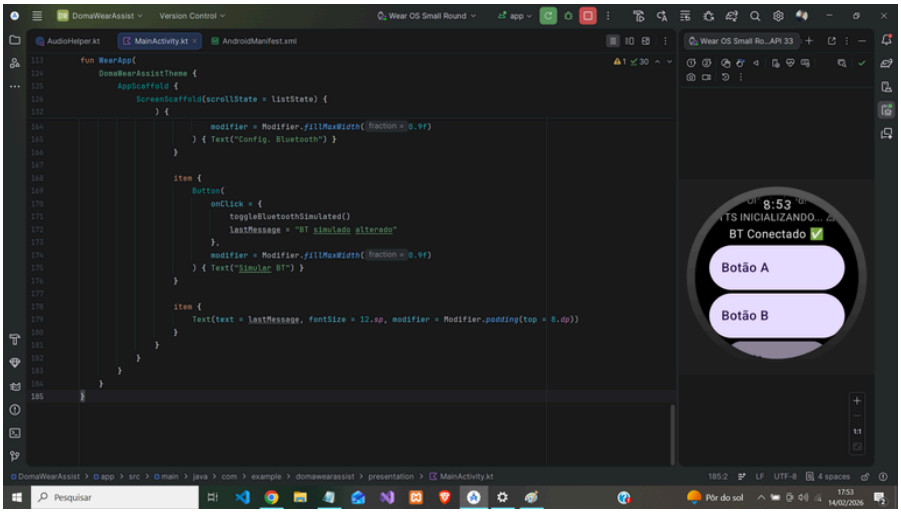
3. O App solicita alteração de permissão para que possa ser ativado bluetooth,

4. Com os registros dos Logs podemos concluir que as funcionalidades solicitadas: Bluetooth, reprodução de audio, aviso de alerta, notificações e conexão funcionam de acordo a solução encontrada sendo validada pelo **LOGCAT.**

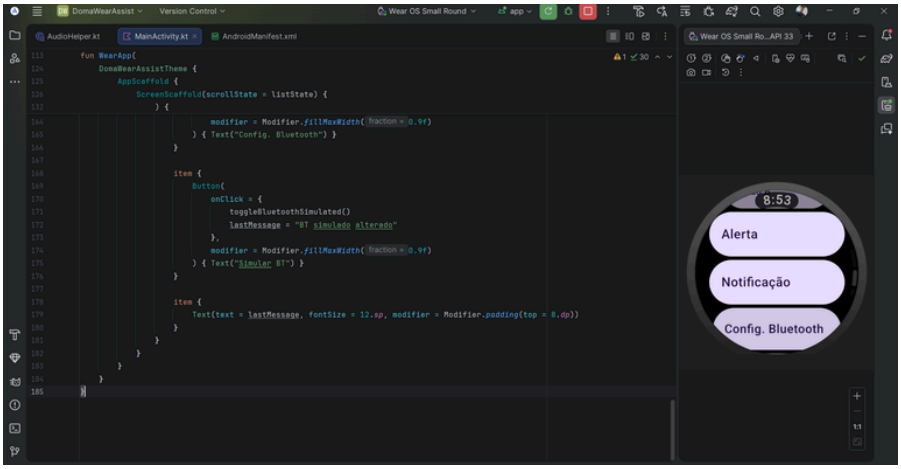
Prints do APP Funcionando



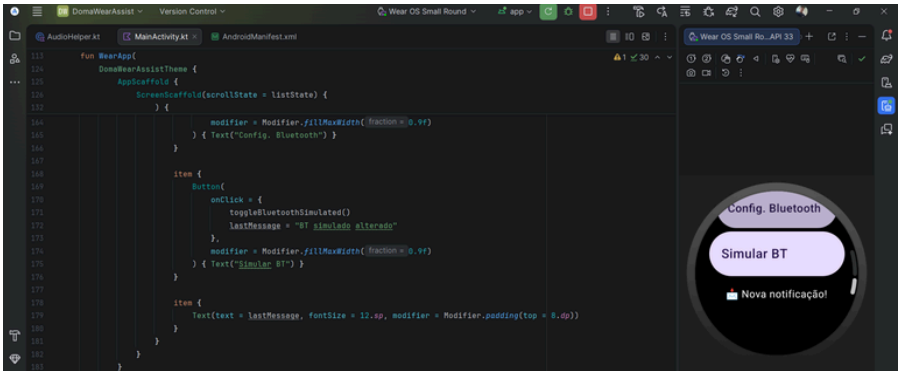
Inicialização do app mostrando os primeiros botões e BT desconectado, com TTS inicializando e mensagem de boas vindas,



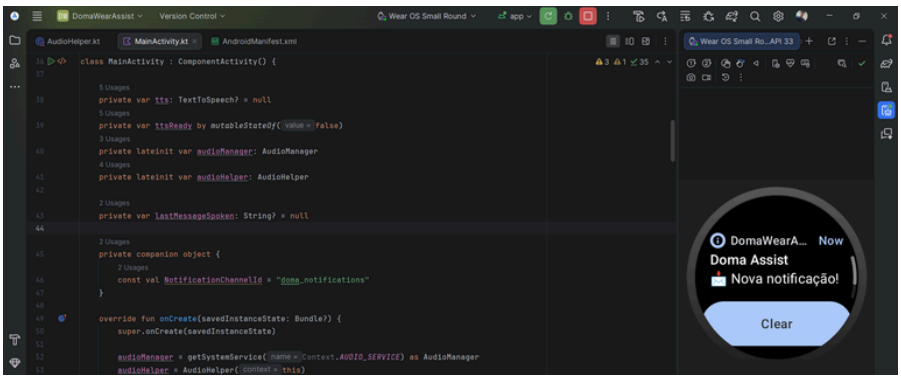
Botão do bluetooth foi pressionado, mudando o status para conectado,



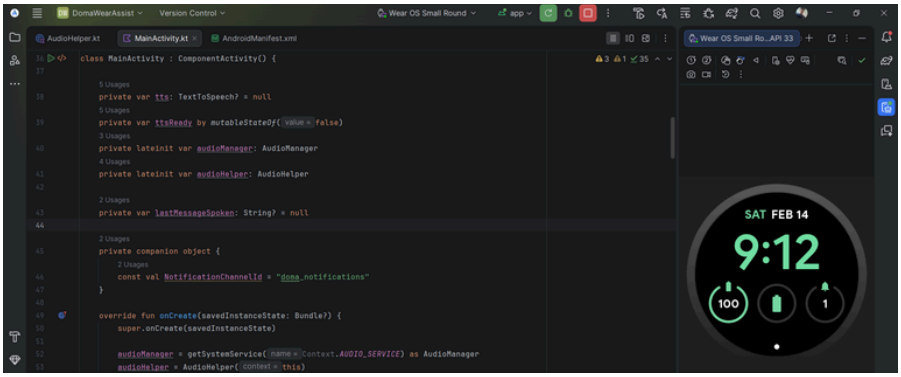
Visualização dos botões: alerta, notificação e configuração do bluetooth, que se pressionado direciona para o menu de configurações do relogio.



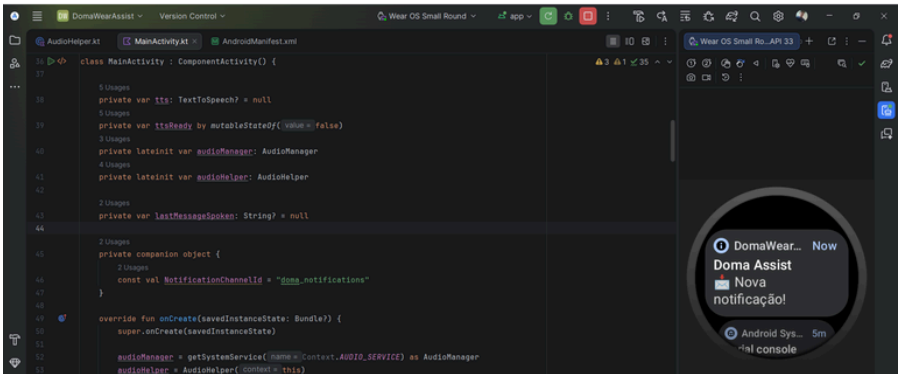
Ao pressionar o botão de notificação o mesmo emite mensagem de notificação na propria tela e também permite redirecionar,



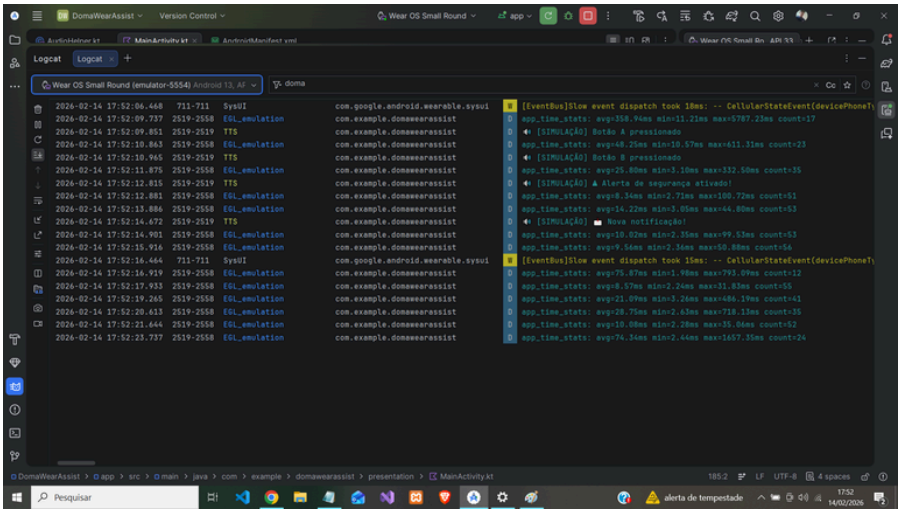
Notificação no menu de notificações.



Notificação registrada na tela principal do relogio.



Notificação pré aberta no menu de notificações.



Aqui estão os logs mostrando que o app se encontra funcional, a sistematica aplicada foi a seguinte: a leitura do TTS é feita > possui TTS ? sim não ? se sim o audio é reproduzido, se não é enviado log, exibindo em LOGCAT,

CÓDIGOS

AudioHelper.kt

```
package com.example.domawearassist.presentation

import android.content.Context
import android.media.AudioDeviceInfo
import android.media.AudioManager
import android.content.pm.PackageManager

class AudioHelper(private val context: Context) {

    private val audioManager =
        context.getSystemService(Context.AUDIO_SERVICE) as
AudioManager

    fun audioOutputAvailable(type: Int): Boolean {

        if (!context.packageManager.hasSystemFeature(
                PackageManager.FEATURE_AUDIO_OUTPUT)) {
            return false
        }

        return
audioManager.getDevices(AudioManager.GET_DEVICES_OUTPUTS)
            .any { it.type == type }
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- Permissões necessárias -->
    <uses-permission
android:name="android.permission.BLUETOOTH_CONNECT"/>
    <uses-permission
android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
    <uses-permission
android:name="android.permission.WAKE_LOCK"/>
    <uses-permission
android:name="android.permission.BODY_SENSORS"/>
    <uses-permission
android:name="android.permission.POST_NOTIFICATIONS"/>

    <!-- Wear OS Feature -->
    <uses-feature android:name="android.hardware.type.watch" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@android:style/Theme.DeviceDefault">

        <uses-library
            android:name="com.google.android.wearable"
            android:required="true" />
        <uses-library
            android:name="wear-sdk"
            android:required="false" />

        <meta-data
            android:name="com.google.android.wearable.standalone"
            android:value="true" />

        <activity
            android:name=".presentation.MainActivity"
            android:exported="true"
            android:taskAffinity=""
            android:theme="@style/MainActivityTheme.Starting">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

MainActivity.kt - Parte 1

```
package com.example.domawearassist.presentation

import android.Manifest
import android.app.NotificationChannel
import android.app.NotificationManager
import android.content.Context
import android.content.Intent
import android.content.pm.PackageManager
import android.media.AudioDeviceCallback
import android.media.AudioDeviceInfo
import android.media.AudioManager
import android.os.Build
import android.os.Bundle
import android.provider.Settings
import android.speech.tts.TextToSpeech
import android.util.Log
import androidx.activity.ComponentActivity
import
androidx.activity.compose.rememberLauncherForActivityResult
import androidx.activity.compose.setContent
import androidx.activity.result.contract.ActivityResultContracts
import androidx.compose.foundation.layout.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.app.NotificationCompat
import androidx.core.content.ContextCompat
import androidx.wear.compose.foundation.lazy.ScalingLazyColumn
import
androidx.wear.compose.foundation.lazy.rememberScalingLazyListSta
te
import androidx.wear.compose.material3.*
import
com.example.domawearassist.presentation.theme.DomaWearAssistT
heme
import java.util.Locale

class MainActivity : ComponentActivity() {

    private var tts: TextToSpeech? = null
    private var ttsReady by mutableStateOf(false)
    private lateinit var audioManager: AudioManager
    private lateinit var audioHelper: AudioHelper

    private var lastMessageSpoken: String? = null

    private companion object {
        const val NotificationChannelId = "doma_notifications"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        audioManager = getSystemService(Context.AUDIO_SERVICE) as
AudioManager
        audioHelper = AudioHelper(this)

        createNotificationChannel()
```

Parte 2

```
        tts = TextToSpeech(applicationContext) { status ->
            if (status == TextToSpeech.SUCCESS) {
                val result = tts?.setLanguage(Locale.getDefault())
                ttsReady = !(result == TextToSpeech.LANG_MISSING_DATA ||
                        result == TextToSpeech.LANG_NOT_SUPPORTED)
                if (ttsReady) speak("Sistema de áudio ativado")
            } else {
                ttsReady = false
            }
        }

        setContent {
            val context = LocalContext.current
            var hasNotificationPermission by remember {
                mutableStateOf(
                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
                        ContextCompat.checkSelfPermission(context,
Manifest.permission.POST_NOTIFICATIONS) ==
PackageManager.PERMISSION_GRANTED
                    } else true
                )
            }

            val launcher = rememberLauncherForActivityResult(
                contract = ActivityResultContracts.RequestPermission()
            ) { isGranted ->
                hasNotificationPermission = isGranted
            }

            LaunchedEffect(Unit) {
                if (!hasNotificationPermission && Build.VERSION.SDK_INT >=
Build.VERSION_CODES.TIRAMISU) {
                    launcher.launch(Manifest.permission.POST_NOTIFICATIONS)
                }
            }

            val bluetoothState = remember {

mutableStateOf(audioHelper.audioOutputAvailable(AudioDeviceInfo.TY
PE_BLUETOOTH_A2DP))
            }

            DisposableEffect(Unit) {
                val callback = object : AudioDeviceCallback() {
                    override fun onAudioDevicesAdded(addedDevices: Array<out
AudioDeviceInfo>?) {
                        bluetoothState.value =
audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_
A2DP)
                    }
                    override fun onAudioDevicesRemoved(removedDevices:
Array<out AudioDeviceInfo>?) {
                        bluetoothState.value =
audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_
A2DP)
                    }
                }
                audioManager.registerAudioDeviceCallback(callback, null)
                onDispose {
                    audioManager.unregisterAudioDeviceCallback(callback)
                }
            }
```

Parte 3

```
WearApp(
            greetingName = "Android",
            isTtsReady = ttsReady,
            bluetoothConnected = bluetoothState.value,
            onSpeak = { speak(it) },
            onSendNotification = { title, msg -> sendVisualNotification(title,
msg) },
            onOpenBluetoothSettings = { openBluetoothSettings() },
            toggleBluetoothSimulated = { bluetoothState.value =
!bluetoothState.value }
        )
    }
}

private fun createNotificationChannel() {
    val name = "Doma Assist"
    val descriptionText = "Notificações do Sistema Doma"
    val importance = NotificationManager.IMPORTANCE_DEFAULT
    val channel = NotificationChannel(NotificationChannelId, name,
importance).apply {
        description = descriptionText
    }
    val notificationManager =
getSystemService(Context.NOTIFICATION_SERVICE) as
NotificationManager
    notificationManager.createNotificationChannel(channel)
}

private fun sendVisualNotification(title: String, message: String) {
    val builder = NotificationCompat.Builder(this,
NotificationChannelId)
        .setSmallIcon(android.R.drawable.ic_dialog_info)
        .setContentTitle(title)
        .setContentText(message)
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setAutoCancel(true)

    val notificationManager =
getSystemService(Context.NOTIFICATION_SERVICE) as
NotificationManager
    notificationManager.notify(System.currentTimeMillis().toInt(),
builder.build())
}

private fun speak(message: String) {
    if (lastMessageSpoken == message) return
    lastMessageSpoken = message

    if (ttsReady) {
        tts?.speak(message, TextToSpeech.QUEUE_FLUSH, null, "tts_id")
    } else {
        Log.d("TTS", "🔊 [SIMULAÇÃO] $message")
    }
}

private fun openBluetoothSettings() {
    try {
        startActivity(Intent(Settings.ACTION_BLUETOOTH_SETTINGS))
    } catch (e: Exception) {
        Log.e("MainActivity", "Erro ao abrir configurações", e)
    }
}
```

Parte 4

```
override fun onDestroy() {
        super.onDestroy()
        tts?.stop()
        tts?.shutdown()
    }
}

@Composable
fun WearApp(
    greetingName: String,
    isTtsReady: Boolean,
    bluetoothConnected: Boolean,
    onSpeak: (String) -> Unit,
    onSendNotification: (String, String) -> Unit,
    onOpenBluetoothSettings: () -> Unit,
    toggleBluetoothSimulated: () -> Unit
) {
    var lastMessage by remember { mutableStateOf("Aguardando
ação...") }
    val listState = rememberScalingLazyListState()

    DomaWearAssistTheme {
        AppScaffold {
            ScreenScaffold(scrollState = listState) {
                ScalingLazyColumn(
                    state = listState,
                    modifier = Modifier.fillMaxSize(),
                    horizontalAlignment = Alignment.CenterHorizontally,
                    contentPadding = PaddingValues(top = 40.dp, bottom =
40.dp, start = 8.dp, end = 8.dp)
                ) {
                    item { Text(text = "Olá, $greetingName!", fontSize = 16.sp) }
                    item { Text(text = if (isTtsReady) "TTS ATIVO ✅" else "TTS
INICIANDO... ⚠", fontSize = 14.sp) }
                    item {
                        Text(
                            text = if (bluetoothConnected) "BT Conectado ✅" else
"BT Desconectado ⚠",
                            fontSize = 14.sp
                        )
                    }

                    item {
                        Button(
                            onClick = {
                                val msg = "Botão A pressionado"
                                onSpeak(msg)
                                lastMessage = msg
                            },
                            modifier = Modifier.fillMaxWidth(0.9f)
                        ) { Text("Botão A") }
                    }
```

Parte 5

```
item {
                Button(
                    onClick = {
                        val msg = "Botão B pressionado"
                        onSpeak(msg)
                        lastMessage = msg
                    },
                    modifier = Modifier.fillMaxWidth(0.9f)
                ) { Text("Botão B") }
            }

            item {
                Button(
                    onClick = {
                        val msg = "⚠️ Alerta de segurança ativado!"
                        onSpeak(msg)
                        lastMessage = msg
                    },
                    modifier = Modifier.fillMaxWidth(0.9f)
                ) { Text("Alerta") }
            }

            item {
                Button(
                    onClick = {
                        val msg = "📩 Nova notificação!"
                        onSpeak(msg)
                        onSendNotification("Doma Assist", msg)
                        lastMessage = "Notificação enviada!"
                    },
                    modifier = Modifier.fillMaxWidth(0.9f)
                ) { Text("Notificação") }
            }

            item {
                Button(
                    onClick = onOpenBluetoothSettings,
                    modifier = Modifier.fillMaxWidth(0.9f)
                ) { Text("Config. Bluetooth") }
            }

            item {
                Button(
                    onClick = {
                        toggleBluetoothSimulated()
                        lastMessage = "BT simulado alterado"
                    },
                    modifier = Modifier.fillMaxWidth(0.9f)
                ) { Text("Simular BT") }
            }

            item {
                Text(text = lastMessage, fontSize = 12.sp, modifier =
Modifier.padding(top = 8.dp))
            }
        }
    }
  }
}
```

Ánalise e Conclusão

- Criar este App me ajudou a pensar em novas possibilidades de solução em casos onde faltam recursos,

- Pensamento lógico e tomada de decisão em curto prazo para entrega do app,

- Soluções auxiliares viáveis e funcionais, validando o projeto,

- Teste e validação continuo com uso do emulador o que permitiu testar não somente em um aparelho de emulador, mas em vários,

- Descobri a importante Utilidade do funcionamento de outros dispositivos utilizando bluetooth com sistema de áudio para atender pessoas com necessidades especiais.

Criando este pequeno app, aprendi a utilizar de forma proveitosa tecnica e profissional o Android Studio, afim de entregar um app que possa atender pessoas com necessidades especiais, pude compreender a necessidade de adaptações e novas soluções para que pudesse entregar o aplicativo de forma funcional e validada.