FACULDADE ESTÁCIO DE SÁ

CURSO: DESENVOLVIMENTO FULL STACK

4º SEMESTRE – MATRÍCULA 202302595341

Repositório GitHub - alaimalmeida/doma (github.com)

ALAIM ALMEIDA DE OLIVEIRA

Lidando com sensores em dispositivos móveis

Salvador – BA

2024

1. Introdução

A evolução tecnológica e o avanço na utilização de dispositivos móveis e wearables, como os relógios inteligentes, têm impulsionado o desenvolvimento de aplicativos que utilizam sensores para proporcionar uma experiência mais imersiva e personalizada ao usuário. Os wearables oferecem uma oportunidade única para empresas integrarem soluções que aumentam a eficiência de seus colaboradores, especialmente em áreas como acessibilidade, comunicação e segurança.

A empresa "Doma", reconhecendo a importância de incluir seus funcionários com necessidades especiais em suas operações cotidianas, pretende desenvolver um aplicativo para dispositivos Wear OS. Este aplicativo visa aprimorar a comunicação, oferecer suporte em tempo real e fornecer alertas e notificações que podem ser acionados por áudio. Este relatório explora o processo de desenvolvimento desse aplicativo, abordando a utilização de sensores em dispositivos móveis e discutindo a importância do áudio no contexto de acessibilidade e segurança.

2. Objetivo

O objetivo deste relatório é descrever o processo de criação de um aplicativo para dispositivos Wear OS, focado na assistência a funcionários com necessidades especiais. O aplicativo será projetado para fornecer notificações em tempo real, respostas a comandos de voz, e alertas de segurança usando recursos de áudio. Este relatório também explorará como os sensores de dispositivos wearables podem ser utilizados para melhorar a interação dos usuários com o sistema, principalmente aqueles com deficiências visuais.

Desenvolvimento do Aplicativo

a) **Planejamento Inicial**

Antes de iniciar o desenvolvimento, é fundamental compreender as necessidades específicas dos usuários e como o aplicativo pode contribuir para a acessibilidade e a eficiência no ambiente de trabalho. No caso da empresa Doma, o público-alvo inclui funcionários com deficiência visual, de modo que o aplicativo deverá ser intuitivo, com foco no feedback auditivo e interação por comandos de voz.

b) **Utilização de Sensores Móveis**

Os dispositivos Wear OS possuem uma série de sensores que podem ser aproveitados para criar interações mais sofisticadas:

- **Sensor de movimento (acelerômetro e giroscópio)**: Pode ser utilizado para detectar o movimento do usuário, permitindo ações automáticas, como ativar notificações ou ajustar o áudio com base na posição do dispositivo.

- **Microfone**: Fundamental para capturar comandos de voz. Esse sensor permitirá que os usuários interajam com o aplicativo sem a necessidade de tocar na tela.
- **Sensor de luz ambiente**: Pode ser utilizado para ajustar automaticamente o volume do áudio com base no ambiente em que o usuário se encontra.

c) **Desenvolvimento da Interface do Usuário (UI) e Experiência do Usuário (UX)**

A interface do aplicativo deve ser minimalista e acessível, com uma ênfase no feedback auditivo, permitindo que os usuários naveguem e interajam com o sistema por voz. Como o público inclui pessoas com deficiência visual, a UI visual deve ser simples, com ícones grandes e sons indicativos para guiar o usuário.

Integração de Áudio e Comandos de Voz

O aplicativo oferecerá suporte a leitura de mensagens e notificações em voz alta, além de possibilitar que o usuário responda por comandos de voz. Por exemplo, ao receber uma notificação de uma mensagem, o sistema pode ler o conteúdo e solicitar ao usuário se ele deseja responder por voz, tudo sem a necessidade de interação manual.

d) **Alertas e Notificações de Segurança**

Um aspecto crucial do aplicativo é a capacidade de fornecer alertas de segurança. Com a integração de APIs de serviços de emergência, o aplicativo poderá emitir notificações de perigo, como tempestades ou outros eventos críticos, garantindo que o usuário esteja sempre informado e seguro. Esses alertas podem ser acompanhados por feedback vibratório, para garantir que o usuário perceba a notificação, mesmo em ambientes barulhentos.

3. Estrutura

Ao criar o aplicativo, temos uma determinada estrutura de pastas, onde ficam as partes importantes do projeto e do sistema android. Logo após a criação do app, criamos alguns arquivos que vão ajudar a compor o mesmo, como os sensores de áudio que vamos usar no projeto.

Logo abaixo, temos uma imagem com toda a estrutura das pastas criadas:

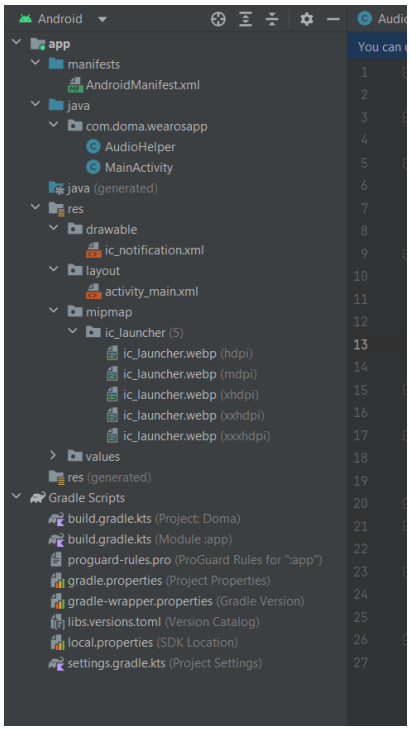


Imagem 01 – Estrutura de pastas

Dentro da pasta manifests, temos o arquivo chamado AndroidManifest.xml, onde são inseridas as chamadas e as importações das bibliotecas dos sensores que vão ser usados na criação do app, como sensor de bluetooth, notificações, gravação de áudio e outros.

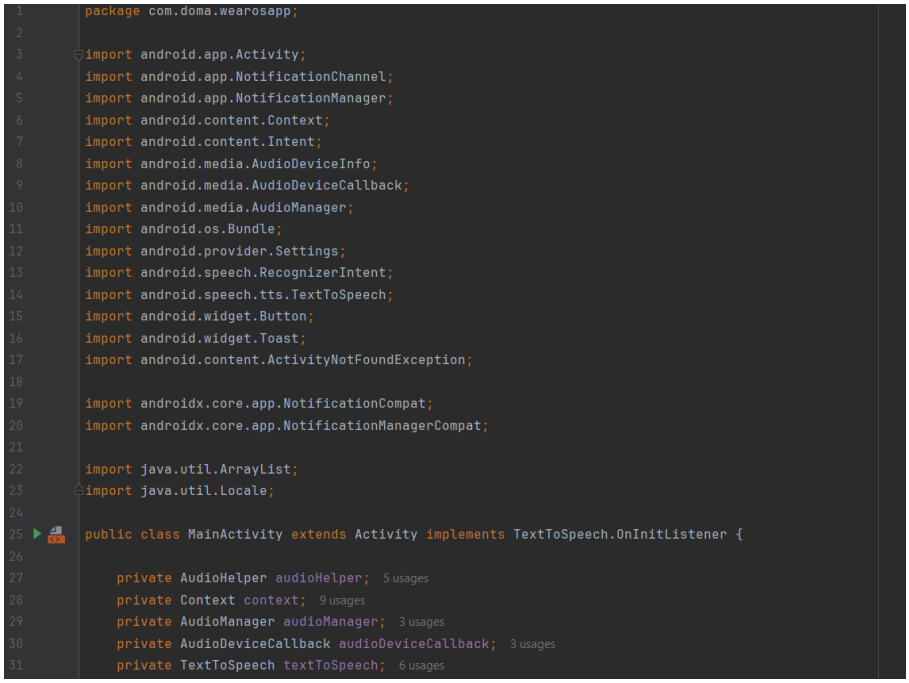
Segue abaixo o código desta página:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.doma.wearosapp">

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
    <uses-permission android:name="android.permission.ACCESS_NOTIFICATION_POLICY" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />

    <uses-feature
        android:name="android.hardware.type.watch"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Doma Wear OS App"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <meta-data
            android:name="com.google.android.wearable.standalone"
            android:value="true" />
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:taskAffinity="">
```

Imagem 02 – arquivo AndroidManifest

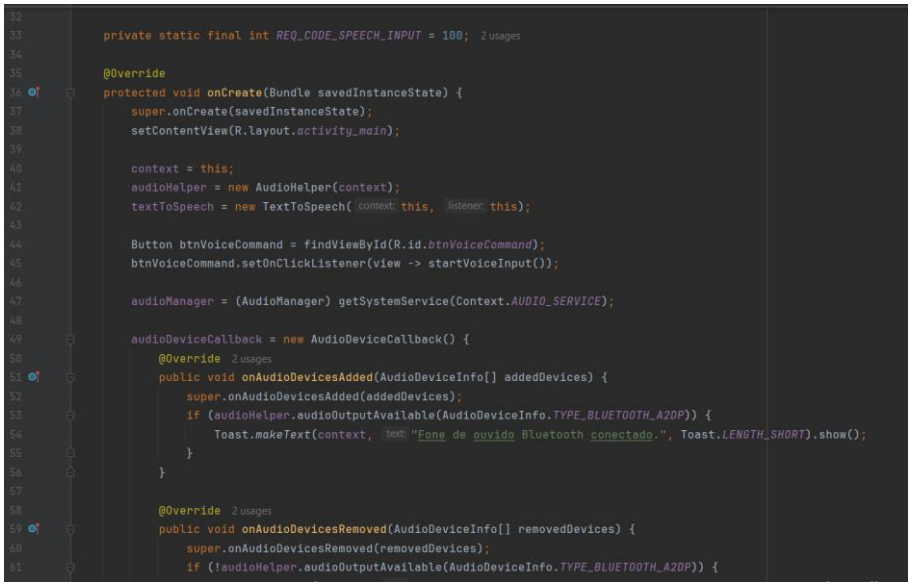
Logo em seguida, temos o arquivo chamado MainActivity onde no desenvolvimento de aplicações Android, incluindo aplicativos para Wear OS, o arquivo **MainActivity** desempenha um papel fundamental como ponto de entrada principal da aplicação. Ele é responsável por gerenciar a interface do usuário e interagir com os componentes do sistema Android para fornecer a funcionalidade desejada.

Abaixo está o código do arquivo:



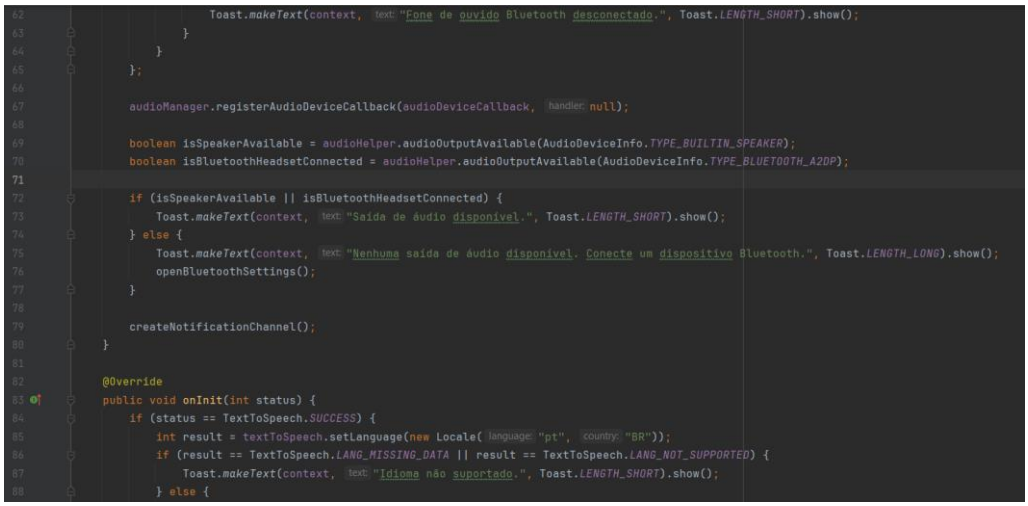
```
package com.doma.wearosapp;

import android.app.Activity;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Context;
import android.content.Intent;
import android.media.AudioDeviceInfo;
import android.media.AudioDeviceCallback;
import android.media.AudioManager;
import android.os.Bundle;
import android.provider.Settings;
import android.speech.RecognizerIntent;
import android.speech.tts.TextToSpeech;
import android.widget.Button;
import android.widget.Toast;
import android.content.ActivityNotFoundException;

import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;

import java.util.ArrayList;
import java.util.Locale;

public class MainActivity extends Activity implements TextToSpeech.OnInitListener {

    private AudioHelper audioHelper; 5 usages
    private Context context; 9 usages
    private AudioManager audioManager; 3 usages
    private AudioDeviceCallback audioDeviceCallback; 3 usages
    private TextToSpeech textToSpeech; 6 usages
```

Imagem 03 – Arquivo MainActivity



```
    private static final int REQ_CODE_SPEECH_INPUT = 100; 2 usages

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        context = this;
        audioHelper = new AudioHelper(context);
        textToSpeech = new TextToSpeech( context: this, listener: this);

        Button btnVoiceCommand = findViewById(R.id.btnVoiceCommand);
        btnVoiceCommand.setOnClickListener(view -> startVoiceInput());

        audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

        audioDeviceCallback = new AudioDeviceCallback() {
            @Override 2 usages
            public void onAudioDevicesAdded(AudioDeviceInfo[] addedDevices) {
                super.onAudioDevicesAdded(addedDevices);
                if (audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_A2DP)) {
                    Toast.makeText(context, text: "Fone de ouvido Bluetooth conectado.", Toast.LENGTH_SHORT).show();
                }
            }

            @Override 2 usages
            public void onAudioDevicesRemoved(AudioDeviceInfo[] removedDevices) {
                super.onAudioDevicesRemoved(removedDevices);
                if (!audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_A2DP)) {
```

Imagem 04 – Arquivo MainActivity

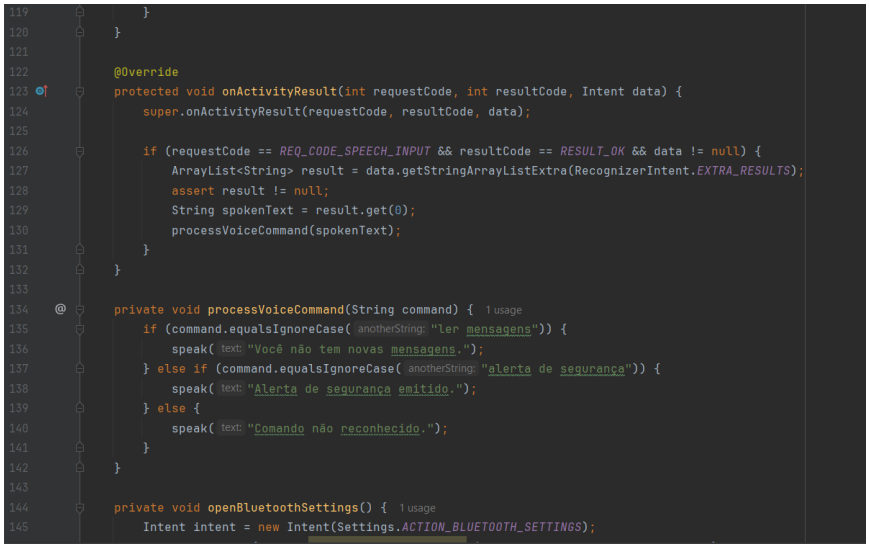


```
                Toast.makeText(context, text: "Fone de ouvido Bluetooth desconectado.", Toast.LENGTH_SHORT).show();
            }
        }
    };

    audioManager.registerAudioDeviceCallback(audioDeviceCallback, handler: null);

    boolean isSpeakerAvailable = audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BUILTIN_SPEAKER);
    boolean isBluetoothHeadsetConnected = audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_A2DP);

    if (isSpeakerAvailable || isBluetoothHeadsetConnected) {
        Toast.makeText(context, text: "Saida de áudio disponivel.", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(context, text: "Nenhuma saida de áudio disponivel. Conecte um dispositivo Bluetooth.", Toast.LENGTH_LONG).show();
        openBluetoothSettings();
    }

    createNotificationChannel();
}

@Override
public void onInit(int status) {
    if (status == TextToSpeech.SUCCESS) {
        int result = textToSpeech.setLanguage(new Locale( language: "pt", country: "BR"));
        if (result == TextToSpeech.LANG_MISSING_DATA || result == TextToSpeech.LANG_NOT_SUPPORTED) {
            Toast.makeText(context, text: "Idioma não suportado.", Toast.LENGTH_SHORT).show();
        } else {
```

Imagem 05 – Arquivo MainActivity

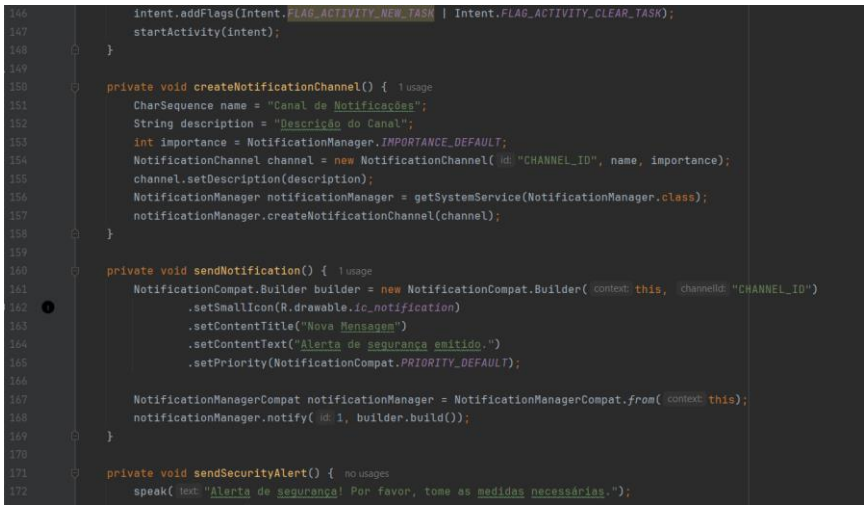
```
            }
        } else {
            Toast.makeText(context, text: "Inicialização do TextToSpeech falhou.", Toast.LENGTH_SHORT).show();
        }
    }

    private void speak(String text) {  5 usages
        textToSpeech.speak(text, TextToSpeech.QUEUE_FLUSH, params: null, utteranceId: "MessageID");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (textToSpeech != null) {
            textToSpeech.stop();
            textToSpeech.shutdown();
        }
        audioManager.unregisterAudioDeviceCallback(audioDeviceCallback);
    }

    private void startVoiceInput() {  1 usage
        Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
        intent.putExtra(RecognizerIntent.EXTRA_PROMPT,  value: "Diga algo...");
        try {
            startActivityForResult(intent, REQ_CODE_SPEECH_INPUT);
        } catch (ActivityNotFoundException a) {
            Toast.makeText(context, text: "Reconhecimento de voz não suportado.", Toast.LENGTH_SHORT).show();
```

Imagem 06 – Arquivo MainActivity



```
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == REQ_CODE_SPEECH_INPUT && resultCode == RESULT_OK && data != null) {
            ArrayList<String> result = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
            assert result != null;
            String spokenText = result.get(0);
            processVoiceCommand(spokenText);
        }
    }

    private void processVoiceCommand(String command) {  1 usage
        if (command.equalsIgnoreCase( anotherString: "ler mensagens")) {
            speak( text: "Você não tem novas mensagens.");
        } else if (command.equalsIgnoreCase( anotherString: "alerta de segurança")) {
            speak( text: "Alerta de segurança emitido.");
        } else {
            speak( text: "Comando não reconhecido.");
        }
    }

    private void openBluetoothSettings() {  1 usage
        Intent intent = new Intent(Settings.ACTION_BLUETOOTH_SETTINGS);
```

Imagem 07 – Arquivo MainActivity



```
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intent);
    }

    private void createNotificationChannel() { 1 usage
        CharSequence name = "Canal de Notificações";
        String description = "Descrição do Canal";
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel( id: "CHANNEL_ID", name, importance);
        channel.setDescription(description);
        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }

    private void sendNotification() { 1 usage
        NotificationCompat.Builder builder = new NotificationCompat.Builder( context: this,  channelId: "CHANNEL_ID")
                .setSmallIcon(R.drawable.ic_notification)
                .setContentTitle("Nova Mensagem")
                .setContentText("Alerta de segurança emitido.")
                .setPriority(NotificationCompat.PRIORITY_DEFAULT);

        NotificationManagerCompat notificationManager = NotificationManagerCompat.from( context: this);
        notificationManager.notify( id: 1, builder.build());
    }

    private void sendSecurityAlert() { no usages
        speak( text: "Alerta de segurança! Por favor, tome as medidas necessárias.");
```

Imagem 08 – Arquivo MainActivity

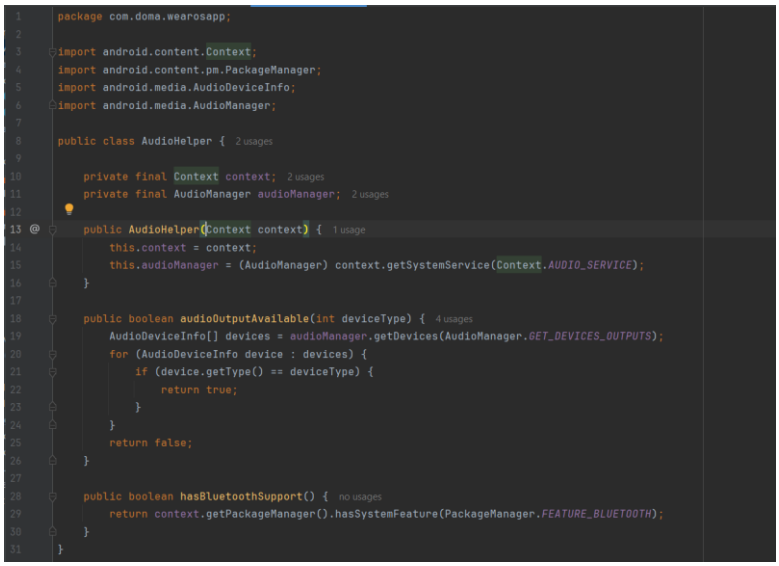
```
    private void sendSecurityAlert() { no usages
        speak( text: "Alerta de segurança! Por favor, tome as medidas necessárias.");
        sendNotification();
    }
}
```

Imagem 09 – Arquivo MainActivity

Depois de configurar o MainActivity, vamos em seguida configurar o arquivo AudioHelper.

O arquivo AudioHelper geralmente tem a função de gerenciar e facilitar o uso de áudio em um aplicativo Android ou Wear OS. Esse tipo de classe é comum em aplicativos que trabalham com sons, como comandos de voz, feedback auditivo, reprodução de música ou alertas sonoros.

Segue abaixo o código do arquivo:



```
package com.doma.wearosapp;

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

public class AudioHelper { 2 usages

    private final Context context; 2 usages
    private final AudioManager audioManager; 2 usages

    public AudioHelper(Context context) { 1 usage
        this.context = context;
        this.audioManager = (AudioManager) context.getSystemService(Context.AUDIO_SERVICE);
    }

    public boolean audioOutputAvailable(int deviceType) { 4 usages
        AudioDeviceInfo[] devices = audioManager.getDevices(AudioManager.GET_DEVICES_OUTPUTS);
        for (AudioDeviceInfo device : devices) {
            if (device.getType() == deviceType) {
                return true;
            }
        }
        return false;
    }

    public boolean hasBluetoothSupport() { no usages
        return context.getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH);
    }
}
```

Imagem 10 – Arquivo AudioHelper

4. Conclusão

O desenvolvimento de aplicativos para dispositivos Wear OS oferece inúmeras possibilidades, especialmente no que diz respeito à acessibilidade e à segurança. O aplicativo da empresa Doma representa um avanço no uso de tecnologia para beneficiar funcionários com necessidades especiais, proporcionando-lhes um meio de comunicação eficiente e seguro. Ao integrar áudio, sensores móveis e feedback em tempo real, a solução desenvolvida não apenas facilita a interação dos usuários com o sistema, mas também oferece um ambiente de trabalho mais inclusivo e colaborativo.

Esse projeto exemplifica como os wearables, quando utilizados de forma criativa e estratégica, podem impactar positivamente a vida das pessoas, promovendo inclusão e acessibilidade em diversas esferas profissionais e sociais.