Missão Prática – Mundo 04 – Nível 03

Gilvan Pereira de Oliveira – 2023.01.53256-6

1197 – POLO CENTRO – SÃO LOURENÇO DA MATA - PE

RPG0025 - Lidando com sensores em dispositivos móveis – 9001 – 2024.2

Repositório GitHub:

Contextualização

Para uma melhoria na eficiência e na comunicação interna, a empresa “Doma” quer desenvolver um aplicativo Wear OS para assistência aos funcionários que têm necessidades especiais, uma forma de solidificar a interação entre os mesmos. Assim, com os aplicativos wearables podem usar áudio para fornecer informações em tempo real, como leitura de mensagens de texto, notificações, lembretes e respostas a comandos de voz. Isso pode ser especialmente útil para pessoas com deficiência visual. Além de serem úteis para treinamento e educação. Aplicativos podem usar áudio para fornecer instruções, dicas e feedbacks durante o aprendizado ou a prática de novas habilidades. Outra funcionalidade que a empresa quer adotar, é um aplicativo wearable que pode usar o áudio para fornecer alertas de segurança, como notificações de emergência, alertas de tempestades, notícias importantes ou informações críticas.

Requisitos Funcionais

1. Configuração do Ambiente:
   - Certifique-se de ter seu ambiente configurado.
   - Prepare um ambiente de simulação para Wear OS ou conecte um dispositivo wearable real.
2. Implementação de Saídas de áudio:
   - AudioDeviceInfo.TYPE_BUILTIN_SPEAKER, em dispositivos com um alto-falante integrado.
   - AudioDeviceInfo.TYPE_BLUETOOTH_A2DP quando um fone de ouvido Bluetooth estiver pareado e conectado.
   - Utilize o método getDevices() com o valor de FEATURE_AUDIO_OUTPUT para enumerar todas as saídas de áudio

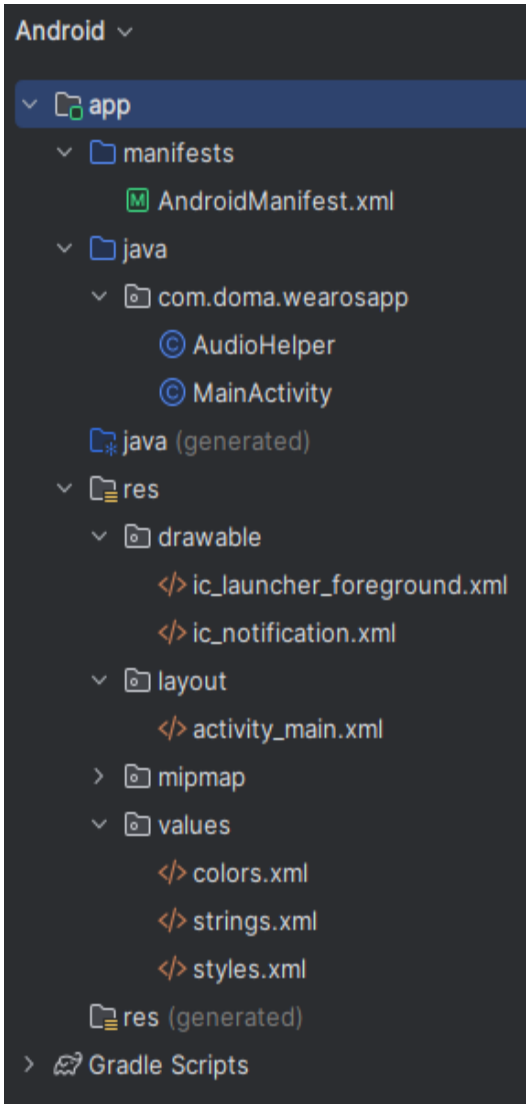
3. Detecção Dinâmica de Dispositivos de Áudio:
    - Seu app pode registrar um callback para detectar quando isso acontece usando registerAudioDeviceCallback
4. Facilitando a Conexão Bluetooth:
    - Se o app exigir que um fone de ouvido seja conectado para continuar, em vez de mostrar uma mensagem de erro, ofereça a opção de direcionar o usuário diretamente às configurações do Bluetooth para facilitar a conexão. Para isso, envie uma intent com ACTION_BLUETOOTH_SETTINGS
5. Reprodução de Áudio:
    - Depois de detectar uma saída de áudio adequada, o processo para tocar áudio no Wear OS é o mesmo usado em dispositivos móveis ou outros dispositivos.
6. Uso de Alto-falantes em Dispositivos Wear OS:
    - Para dispositivos Wear OS que incluem alto-falantes, incorpore funcionalidades de áudio para enriquecer a experiência do usuário.
    - Exemplos de uso incluem alarmes de relógio com notificações sonoras, apps de fitness com instruções de voz para exercícios, e apps educativos com feedback auditivo.

Desenvolvendo seu aplicativo - Resultados esperados

O aplicativo deverá ser capaz de ler mensagens e notificações em voz alta, responder a comandos de voz e fornecer alertas de segurança e instruções através de áudio. Este aplicativo não apenas melhora a eficiência e a comunicação interna na empresa "Doma", mas também demonstra a aplicação prática de tecnologias wearables para criar soluções acessíveis e inclusivas no local de trabalho.

RPG0025 - Lidando com sensores em dispositivos móveis

O projeto foi desenvolvido em Java. Optei por utilizar o editor de código AndroidStudio com o sdk 34, para realizar todo o processo de criação e implementação do aplicativo solicitado na missão prática do nível 03 do mundo 04. No qual, precisei criar um emulador utilizando o SDK do Android Studio.

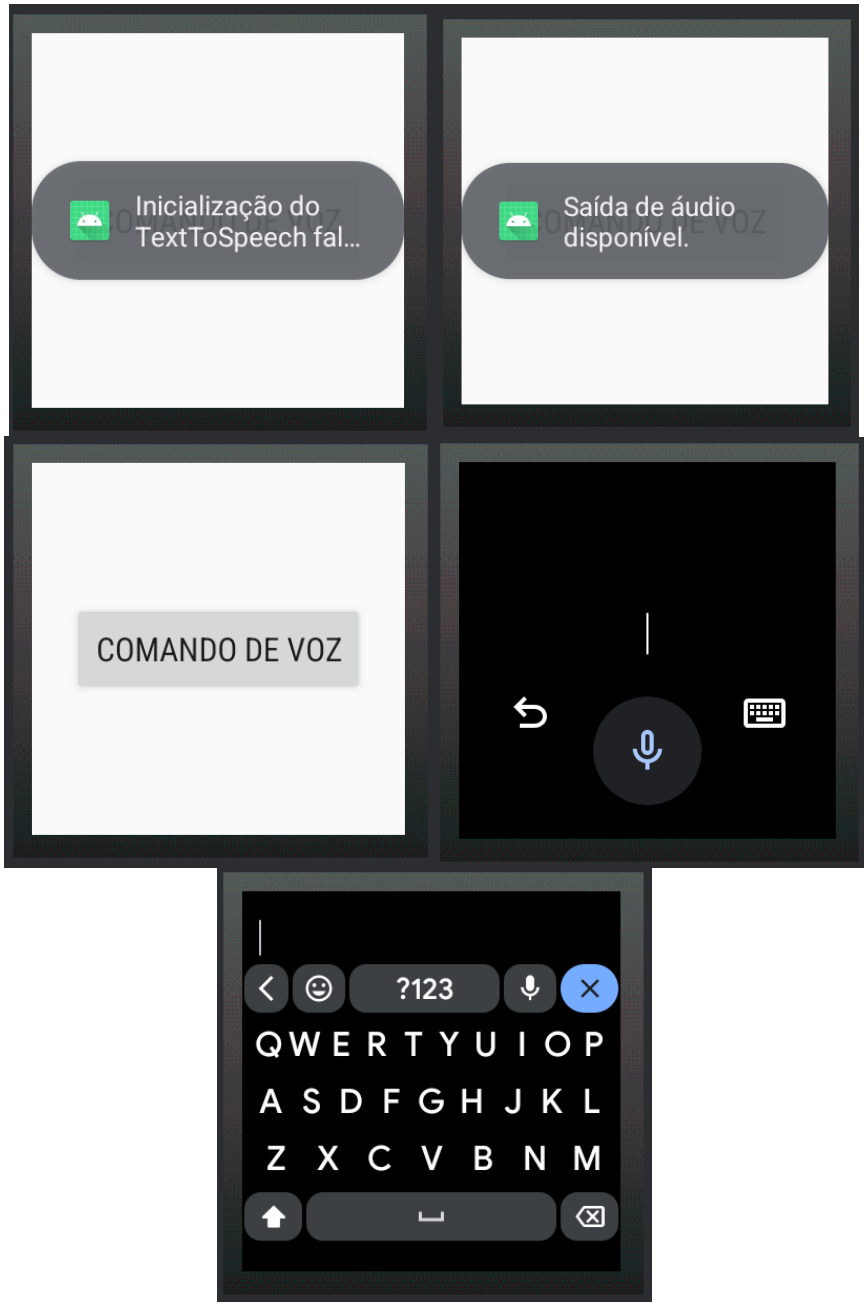


Foi idealizado a seguinte estrutura demonstrada ao lado:

Doma

```
└── app
    ├── manifests
    │   └── AndroidManifest.xml
    │       └── fotos utilizadas....
    ├── java
    │   └── com.doma.wearosapp
    │       ├── AudioHelper.java
    │       └── MainActivity.java
    └── res
    │   └── drawable
    │       ├── ic_launcher_foreground
    │       └── ic_notification.xml
    │   └── layout
    │       └── activity_main.xml
    │   └── mipmap
    │   └── values
    │       └── colors.xml
    │       └── strings.xml
    │       └── styles.xml
└── Gradle Scripts
```

Onde o aplicativo se inicia com uma tela solicitando pressionar o botão para inserir um comando de voz, e caso não seja compreendido, pode-se utilizar o teclado. Conexão com o bluetooth, leitura de notificações e conexão com fones ou caixa de som. Abaixo seguem imagens do aplicativo elaborado:



Abaixo seguem os códigos das págnas:

AndroidManifest.xml:

```
<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="@string/app_name"
        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="">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

AudioHelper.java:

```
package com.doma.wearosapp;

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

public class AudioHelper {

    private final Context context;
    private final AudioManager audioManager;

    public AudioHelper(Context context) {
        this.context = context;
        this.audioManager = (AudioManager)
```

```
context.getSystemService(Context.AUDIO_SERVICE);
    }

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

    public boolean hasBluetoothSupport() {
        return
context.getPackageManager().hasSystemFeature(PackageManager.FEATURE
_BLUETOOTH);
    }
}
```

MainActivity.java:

```
package com.doma.wearosapp;

import android.annotation.SuppressLint;
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;
    private Context context;
    private AudioManager audioManager;
    private AudioDeviceCallback audioDeviceCallback;
    private TextToSpeech textToSpeech;

    private static final int REQ_CODE_SPEECH_INPUT = 100;

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

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

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

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

        audioDeviceCallback = new AudioDeviceCallback() {
            @Override
            public void onAudioDevicesAdded(AudioDeviceInfo[] addedDevices) {
                super.onAudioDevicesAdded(addedDevices);
                if
(audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_A2D
P)) {
                    Toast.makeText(context, "Fone de ouvido Bluetooth conectado.",
Toast.LENGTH_SHORT).show();
                }
            }

            @Override
            public void onAudioDevicesRemoved(AudioDeviceInfo[]
removedDevices) {
                super.onAudioDevicesRemoved(removedDevices);
                if
(!audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_A2
DP)) {
                    Toast.makeText(context, "Fone de ouvido Bluetooth
desconectado.", Toast.LENGTH_SHORT).show();
                }
            }
        };
```

```
        audioManager.registerAudioDeviceCallback(audioDeviceCallback, null);

        boolean isSpeakerAvailable =
audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BUILTIN_SPEAKER
);
        boolean isBluetoothHeadsetConnected =
audioHelper.audioOutputAvailable(AudioDeviceInfo.TYPE_BLUETOOTH_A2D
P);

        if (isSpeakerAvailable || isBluetoothHeadsetConnected) {
            Toast.makeText(context, "Saída de áudio disponível.",
Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(context, "Nenhuma saída de áudio disponível. 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("pt", "BR"));
            if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
                Toast.makeText(context, "Idioma não suportado.",
Toast.LENGTH_SHORT).show();
            } else {
                speak("Bem-vindo ao aplicativo da Doma.");
            }
        } else {
            Toast.makeText(context, "Inicialização do TextToSpeech falhou.",
Toast.LENGTH_SHORT).show();
        }
    }

    private void speak(String text) {
        textToSpeech.speak(text, TextToSpeech.QUEUE_FLUSH, null,
"MessageID");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (textToSpeech != null) {
            textToSpeech.stop();
```

```
            textToSpeech.shutdown();
        }
        audioManager.unregisterAudioDeviceCallback(audioDeviceCallback);
    }

    private void startVoiceInput() {
        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, "Diga algo...");
        try {
            startActivityForResult(intent, REQ_CODE_SPEECH_INPUT);
        } catch (ActivityNotFoundException a) {
            Toast.makeText(context, "Reconhecimento de voz não suportado.",
Toast.LENGTH_SHORT).show();
        }
    }

    @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) {
        if (command.equalsIgnoreCase("ler mensagens")) {
            speak("Você não tem novas mensagens.");
        } else if (command.equalsIgnoreCase("alerta de segurança")) {
            sendSecurityAlert();
        } else if (command.equalsIgnoreCase("ler notificações")) {
            speak("Você tem novas notificações.");
        } else if (command.equalsIgnoreCase("instrução de segurança")) {
            speak("Em caso de emergência, siga as saídas de emergência mais
próximas.");
        } else {
            speak("Comando não reconhecido.");
        }
    }
```

```
    @SuppressLint("WearRecents")
    private void openBluetoothSettings() {
        Intent intent = new Intent(Settings.ACTION_BLUETOOTH_SETTINGS);
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intent);
    }

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

    private void sendNotification() {
        NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
"CHANNEL_ID")
                .setSmallIcon(R.drawable.ic_launcher_foreground)
                .setContentTitle("Nova notificação")
                .setContentText("Alerta de segurança ativado!")
                .setPriority(NotificationCompat.PRIORITY_DEFAULT);

        NotificationManagerCompat notificationManager =
NotificationManagerCompat.from(this);
        notificationManager.notify(1001, builder.build());

        // Leitura da notificação em voz alta
        speak("Alerta de segurança ativado!");
    }

    private void sendSecurityAlert() {
        sendNotification();
        speak("Atenção! Alerta de segurança ativado. Proceda com cautela.");
    }
}
```

ic_launcher_foreground.xml:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="48dp"
    android:height="48dp"
    android:viewportWidth="24"
    android:viewportHeight="24">
```

```
    <path
        android:fillColor="#FF0000"
        android:pathData="M12,2L2,22h20L12,2zM12,6l1.76,4.24L18,11l-
3.76,2.76L15.52,18l-3.52-2.24L8.48,18L10,13.76L6.24,11L10.24,10.24L12,6z"
/>
</vector>
```

ic_notification.xml:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24">
    <path
        android:fillColor="#FF000000"
        android:pathData="M12,2A10,10 0,1 1,2 12A10,10 0,0 1,12
2M13,9H11V17H13V9M13,7H11V5H13V7Z" />
</vector>
```

activity_main.xml:

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

    <Button
        android:id="@+id/btnVoiceCommand"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Comando de Voz"
        android:layout_centerInParent="true"/>

</RelativeLayout>
```

colors.xml:

```
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>
</resources>
```

strings.xml:

```
<resources>
    <string name="app_name">Doma Wear OS App</string>
</resources>
```

styles.xml:

```
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

build.gradle.kts (Module :app):

```
plugins {
    id("com.android.application")
}

android {
    namespace = "com.doma.wearosapp"
    compileSdk = 33

    defaultConfig {
        applicationId = "com.doma.wearosapp"
        minSdk = 26
        targetSdk = 33
        versionCode = 1
        versionName = "1.0"
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
}

dependencies {
    implementation("androidx.core:core-ktx:1.6.0")
    implementation("androidx.appcompat:appcompat:1.3.1")
}
```