Estácio

Trabalho Prático | Trabalho Prático | DGT2816 Interação com sensores de smartphones e wearebles

Sasha Cardoso Vicente – Matrícula 202310275163

Campus 1194 POLO CENTRO - FLORIANÓPOLIS - SC

Interação com sensores de smartphones e wearebles – Turma 9001 – Semestre 2025.4

Link do repositório GIT:

https://github.com/SashaCardoso/Trabalho-Pratico---DGT2816-Interacao-com-sensores-de-smartphones-e-wearebles

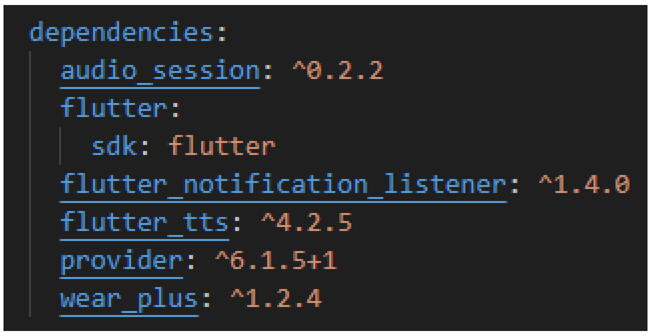
----

Objetivo da Prática

O objetivo desta prática é desenvolver um app capaz de ler em voz alta mensagens em texto e outras informações.

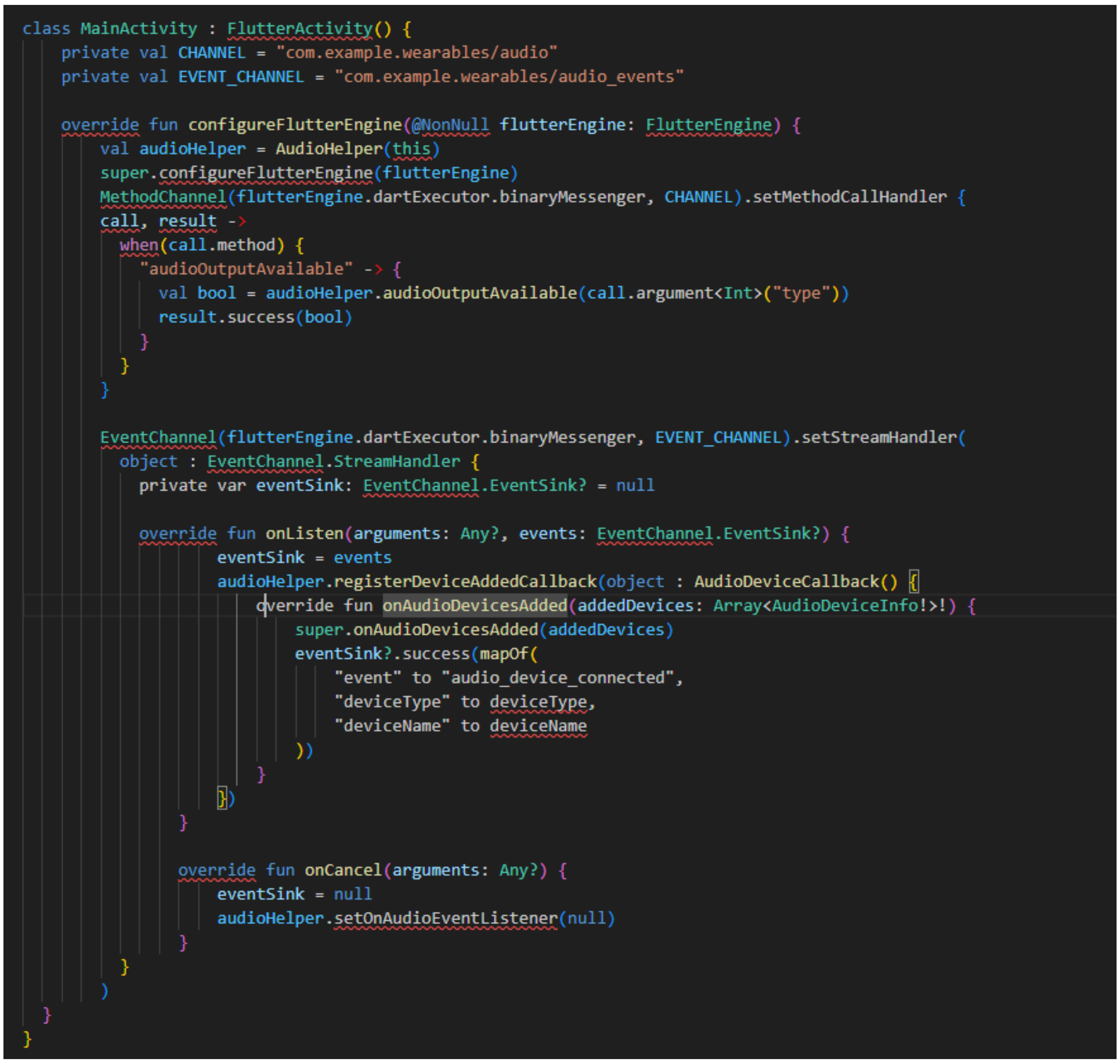
Análise

Durante o processo, usei estas dependencies:



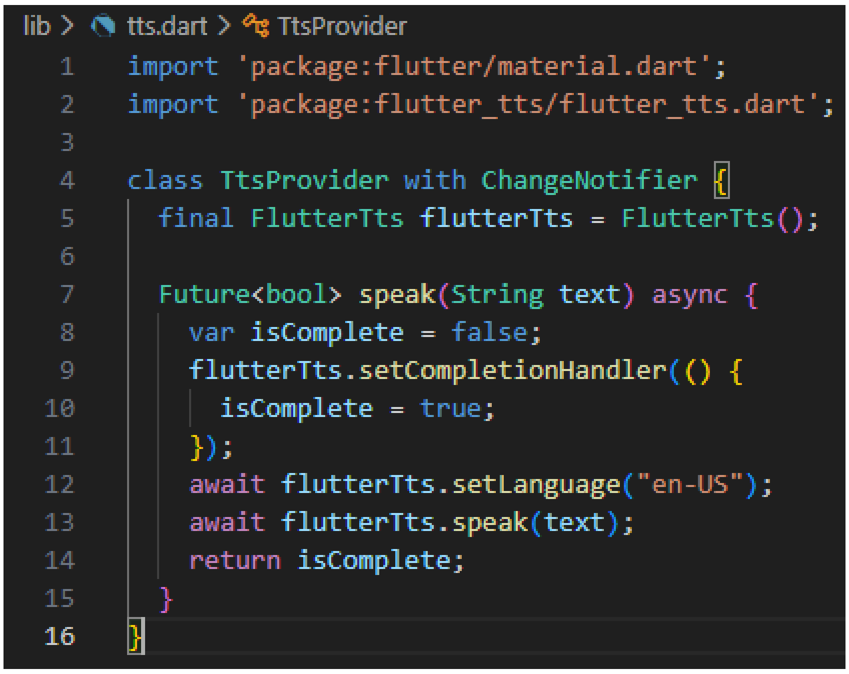
```
dependencies:
  audio_session: ^0.2.2
  flutter:
    sdk: flutter
  flutter_notification_listener: ^1.4.0
  flutter_tts: ^4.2.5
  provider: ^6.1.5+1
  wear_plus: ^1.2.4
```

Tentei seguir o código, que involve usar Kotlin, mas isso acabou complicando demais e não consegui fazer funcionar direito:

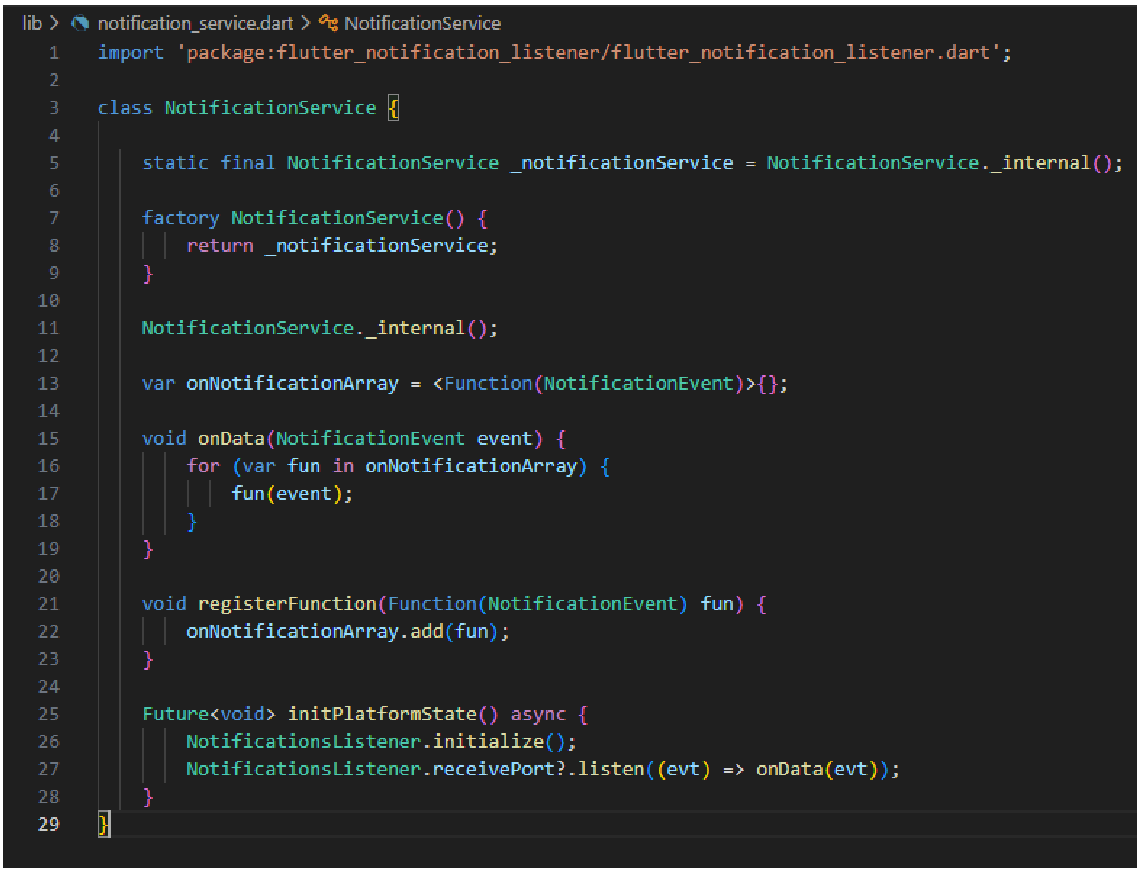


```kotlin
class MainActivity : FlutterActivity() {
    private val CHANNEL = "com.example.wearables/audio"
    private val EVENT_CHANNEL = "com.example.wearables/audio_events"

    override fun configureFlutterEngine(@NonNull flutterEngine: FlutterEngine) {
        val audioHelper = AudioHelper(this)
        super.configureFlutterEngine(flutterEngine)
        MethodChannel(flutterEngine.dartExecutor.binaryMessenger, CHANNEL).setMethodCallHandler {
        call, result ->
          when(call.method) {
            "audioOutputAvailable" -> {
              val bool = audioHelper.audioOutputAvailable(call.argument<Int>("type"))
              result.success(bool)
            }
          }
        }

        EventChannel(flutterEngine.dartExecutor.binaryMessenger, EVENT_CHANNEL).setStreamHandler(
          object : EventChannel.StreamHandler {
            private var eventSink: EventChannel.EventSink? = null

            override fun onListen(arguments: Any?, events: EventChannel.EventSink?) {
                    eventSink = events
                    audioHelper.registerDeviceAddedCallback(object : AudioDeviceCallback() {
                        override fun onAudioDevicesAdded(addedDevices: Array<AudioDeviceInfo!>!) {
                            super.onAudioDevicesAdded(addedDevices)
                            eventSink?.success(mapOf(
                                "event" to "audio_device_connected",
                                "deviceType" to deviceType,
                                "deviceName" to deviceName
                            ))
                        }
                    })
                }

                override fun onCancel(arguments: Any?) {
                    eventSink = null
                    audioHelper.setOnAudioEventListener(null)
                }
          }
        )
  }
}
```

Decidi ao invés ver se tem algo em Flutter que me permite alcançar o mesmo objetivo.


lib > tts.dart > TtsProvider

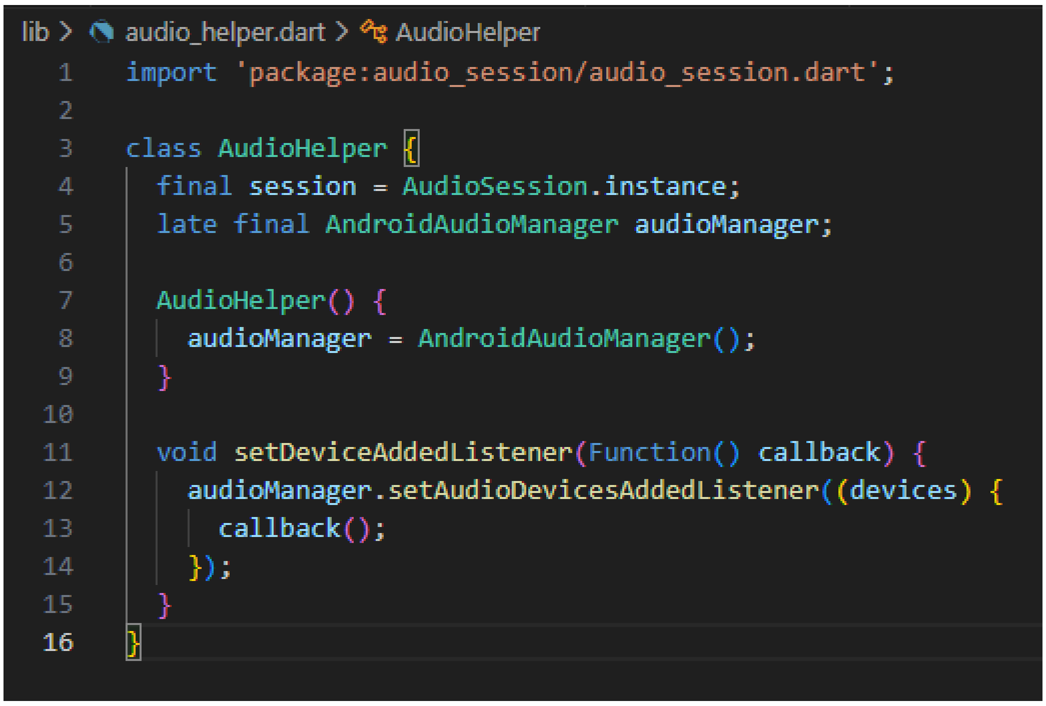
```dart
import 'package:flutter/material.dart';
import 'package:flutter_tts/flutter_tts.dart';

class TtsProvider with ChangeNotifier {
  final FlutterTts flutterTts = FlutterTts();

  Future<bool> speak(String text) async {
    var isComplete = false;
    flutterTts.setCompletionHandler(() {
      isComplete = true;
    });
    await flutterTts.setLanguage("en-US");
    await flutterTts.speak(text);
    return isComplete;
  }
}
```



lib > notification_service.dart > NotificationService

```dart
import 'package:flutter_notification_listener/flutter_notification_listener.dart';

class NotificationService {

  static final NotificationService _notificationService = NotificationService._internal();

  factory NotificationService() {
    return _notificationService;
  }

  NotificationService._internal();

  var onNotificationArray = <Function(NotificationEvent)>{};

  void onData(NotificationEvent event) {
    for (var fun in onNotificationArray) {
      fun(event);
    }
  }

  void registerFunction(Function(NotificationEvent) fun) {
    onNotificationArray.add(fun);
  }

  Future<void> initPlatformState() async {
    NotificationsListener.initialize();
    NotificationsListener.receivePort?.listen((evt) => onData(evt));
  }
}
```

Usei também um audio helper para registrar um callback quando um novo dispositivo de áudio for adicionado.



lib > audio_helper.dart > AudioHelper

```dart
import 'package:audio_session/audio_session.dart';

class AudioHelper {
  final session = AudioSession.instance;
  late final AndroidAudioManager audioManager;

  AudioHelper() {
    audioManager = AndroidAudioManager();
  }

  void setDeviceAddedListener(Function() callback) {
    audioManager.setAudioDevicesAddedListener((devices) {
      callback();
    });
  }
}
```

Estácio