Aplicação para caracterização da voz

Pedro Branco Santos
(Licenciado em Engenharia Informática e Multimédia)

Trabalho de Projeto para obtenção do grau de Mestre em Engenharia Informática e Multimédia

Orientador:
Doutor Hugo Tito Cordeiro

Júri:
Presidente: Doutor Rui Manuel Feliciano de Jesus
Vogais:
Doutor Carlos Eduardo de Meneses Ribeiro
Doutor Hugo Tito Cordeiro

julho de 2025

Aplicação para caracterização da voz

Pedro Branco Santos
(Licenciado em Engenharia Informática e Multimédia)

Trabalho de Projeto para obtenção do grau de Mestre em Engenharia Informática e Multimédia

Orientador:
Doutor Hugo Tito Cordeiro, DEETC/ISEL

Júri:
Presidente: Doutor Rui Manuel Feliciano de Jesus, DEETC/ISEL
Vogais:
Doutor Carlos Eduardo de Meneses Ribeiro, DEETC/ISEL
Doutor Hugo Tito Cordeiro, DEETC/ISEL

julho de 2025

Declaração de integridade

Declaro que esta(e) dissertação / trabalho de projeto / relatório de estágio é o resultado da minha investigação pessoal e independente. O seu conteúdo é original e todas as fontes listadas nas referências bibliográficas foram consultadas e estão devidamente mencionadas no texto. Mais declaro que todas as referências científicas e técnicas relevantes para o desenvolvimento do trabalho estão devidamente citadas e constam das referências bibliográficas.

Pedro Branco Santos

Pedro Branco Santos
Lisboa, 15 de julho de 2025

Agradecimentos

O meu percurso académico teve os seus momentos memoráveis, mas esses momentos foram graças a certas pessoas que foram importantes quer para o desenvolver do mestrado quer para o projeto. Sendo assim, quero agradecer às pessoas que se seguem.

Quero agradecer ao meu coordenador e engenheiro Hugo Tito Cordeiro por me ter ajudado na realização do projeto, por ter discutido a ideia do projeto comigo e por ter disponibilizado todas as ferramentas necessárias na realização do projeto.

Quero agradecer a todos os meus colegas que me ajudaram no desenrolar do mestrado, pelos bons e maus momentos.

Quero agradecer ao ISEL por providenciar todo o conhecimento e por fornecer o espaço para desenvolver e começar a minha carreira.

Resumo

Este projeto propõe uma aplicação simples e intuitiva com a capacidade de se ligar a um servidor, com o foco a melhorar a experiência do utilizador através de dinamismo e modelação.

A funcionalidade principal da aplicação é apresentar resultados, numa interface, que caracterizam a fala através da comunicação com o servidor, mantendo sempre a troca de informação segura e que não haja perdas desta. Um dos objetivos principais do projeto foi garantir que a experiência do utilizador seja agradável, removendo qualquer complexidade na interface da aplicação. A arquitetura do servidor está desenhada de modo a garantir que a informação é sempre processada e que o utilizador obtenha sempre os resultados desejados.

Este projeto desenvolveu uma aplicação protótipo que permite a extração de parâmetros dos sinais de fala para rastreio de patologias da voz. A aplicação pretende ser dinâmica, nomeadamente através da inclusão de novos algoritmos. A interface de utilizador foi desenvolvida em Unity3D de modo a permitir o uso em múltiplas plataformas. Os algoritmos foram desenvolvidos em Python de modo a tirar partido das bibliotecas de processamento de fala e sinal existentes.

Palavras-chave: Aplicação modelar, Caracterização da voz, Extração de características do sinal de fala, simplicidade.

Abstract

This project proposes a simple and intuitive application that connects to a server, focusing on enhancing user experience through modeling and dynamic behavior of the application.

The main purpose of the application is to present data that is related to speech analysis. This data comes from a server that the application connects to, and the server has the responsibility of maintaining the connection and ensuring that the data is never lost and always received. One of the main purposes of the project was to ensure that the user experience was pleasant by removing any complexity on the interface of the application. The server's architecture was designed to ensure that the information sent from the application was always received and ensure that the user got the wanted results from the requests.

For this project, the technologies used were Unity to design and implement the application and Python to implement the server as well as all required Python functions for the server to run.

The project described here shows all the steps, thoughts and designs developed, including a guide for the user to navigate the application, always focusing on user experience and usability, modeling and dynamic behavior.

Keywords: modeling, dynamic, UI, simplicity, user experience

Índice

Índice de figuras

1.Introdução

1.1 Motivação

A motivação principal do projeto é criar um protótipo para a análise de ficheiros de fala de uma forma dinâmica consoante as necessidades do utilizador.

Atualmente no mercado não existe nenhuma aplicação que tenha a capacidade de modelação consoante as necessidades de cada utilizador, portanto é oportuno criar uma interface simples e dinâmica que permita mostrar os resultados das operações realizadas em Python.

A alternativa apresentada consiste em utilizar a plataforma Unity para criar uma aplicação de modo a fazer uma ligação HTTP com um servidor em linguagem Python para realizar diversas operações que sejam capazes de retirar determinadas características da fala.

A solução proposta é um conceito que é concebível, mas é dependente de versões entre diferentes tecnologias. Uma das motivações é criar uma aplicação onde o utilizador conseguia modelar e utilizar independente das versões para evitar a constante manutenção da aplicação.

1.2 Objetivos

O objetivo do projeto é criar uma interface Unity com a ligação a um servidor Python. Esta interface tem como objetivo ser intuitiva e dinâmica para o utilizador. O utilizador deverá conseguir perceber facilmente o objetivo da aplicação e modelá-la a seu gosto.

A interface deve ser intuitiva. O utilizador deve conseguir perceber facilmente a informação que lhe é pedida pela aplicação e a sua execução. O utilizador deve conseguir importar as funções escritas em Python para a aplicação Unity.

A importação das funções escritas em Python será guardada numa base de dados. Respeitando uma das palavras-chave do projeto, simplicidade, a base de dados é uma base de dados local onde serão guardadas todas as funções importadas pelo utilizador.

No servidor Python são implementados 3 métodos para a caracterização da fala; RPPC, *Relative Power of Periodic Component,* LBST, *Low Band Spectral Tilt,* e HBST *High Band Spectral Tilt*. Cada um destes métodos retorna os seus respetivos valores para o servidor enviar para a aplicação Unity. O servidor será local para manter a simplicidade na aplicação, mas a comunicação usa protocolos de mensagem para garantir que a informação é recebida sem perdas. A aplicação permite que seja atualizada dinamicamente através de funções

Python, onde podem ser analisados outros parâmetros do sinal de fala ou até a implementação classificadores.

Todo o fluxo da aplicação deve ser claro e intuitivo para evitar que o utilizador se sinta sobrecarregado com demasiada informação. O utilizador terá sempre a possibilidade de voltar atrás caso queira mudar dados ou fazer outro tipo de ação.

1.3 Organização do Documento

Este documento está organizado em seis capítulos. A Introdução descreve a motivação e os objetivos do projeto, enquanto o Estado da Arte apresenta um levantamento das tecnologias de processamento de fala utilizadas, o seu enquadramento na logística da fala e os principais parâmetros de fala abordados neste projeto. Segue-se a secção de Materiais e Métodos, onde são detalhadas as versões utilizadas, as bases de dados e os métodos implementados no servidor Python. A Implementação detalha toda a aplicação por página, o fluxo do servidor e a comunicação entre a aplicação Unity e o servidor Python, incluindo ainda um manual que apresenta cada página e as suas funcionalidades. A Conclusão e Trabalho Futuro faz um balanço e análise do projeto desenvolvido, e identifica as principais funcionalidades que seriam úteis implementar numa versão futura.

2. Estado da Arte

2.1 Aplicações do sinal de fala

Como referido anteriormente, este projeto tem como objetivo mostrar um protótipo que permite dinamismo e simplicidade para a análise de fala, mas para a criação da ideia do projeto foi necessário realizar um estudo de programas atuais para obter um conceito único. Também foi feito um estudo teórico dos conceitos e parâmetros do sinal de fala para desenvolver métodos que consigam caracterizar a fala.

2.1.1 DisVoice

O DisVoice [1] é uma *framework* que utiliza a linguagem Python para obter características no sinal de fala como pulso glotal, articulação, fonação, entre outros. O programa pode receber tanto uma vogal sustentada ou fala corrente para a extração de características. As características consistem em pulso glotal, fonação e articulação da fala

Estas características são importantes para reconhecer emoções ou para reconhecer diferentes patologias relacionadas com a voz do paciente como cancro ou nódulos.

Na figura 1 está representado um exemplo de execução do programa para obter as características do pulso glotal de um sinal.

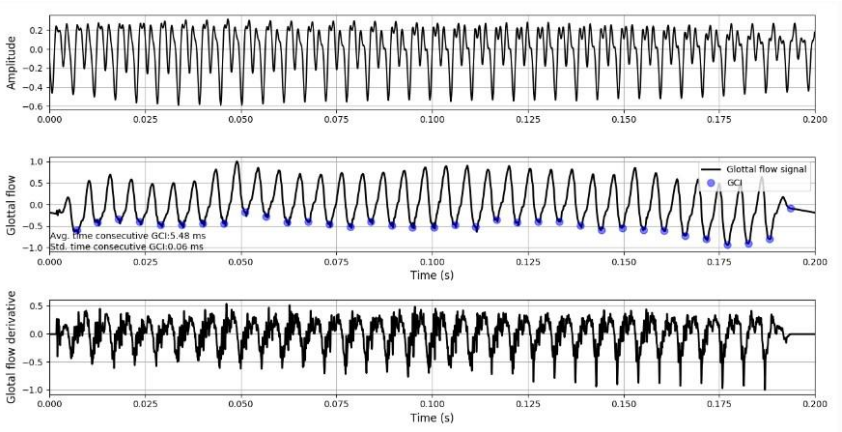

Figura 1: Exemplo DisVoice

2.1.2 Praat

O Praat [2], criado por Paul Boersma e David Weenink, é uma ferramenta de análise usada com o objetivo de produzir fonemas, mas também tem a capacidade de analisar, sintetizar e manipular um sinal de fala.

O Praat é dividido em cinco principais funcionalidades:

1. sintetização de fala, o programa cria um sinal de fala através da curvatura do *pitch* ou através do trato vocal do utilizador,

2. manipulação de fala, o programa tem a funcionalidade de alterar o *pitch*, intensidade ou duração da fala,

3. caracterização de fala, o programa usa AIF, Alfabeto Internacional de Fonemas, para repartir um sinal de fala em palavras ou fonemas,

4. modelação de redes neuronais, o programa utiliza redes neuronais pré-definidas como *Optimality-Theory* e *Harmonic-Grammar,* mas também possui a capacidade de criar um modelo de aprendizagem novo, e por fim

5. análise estatística, o programa possui diferentes técnicas de análise estatística como análise discriminante, escalamento multidimensional e análise de componentes principais.

A figura 2 é apresentado um exemplo de execução do Praat.

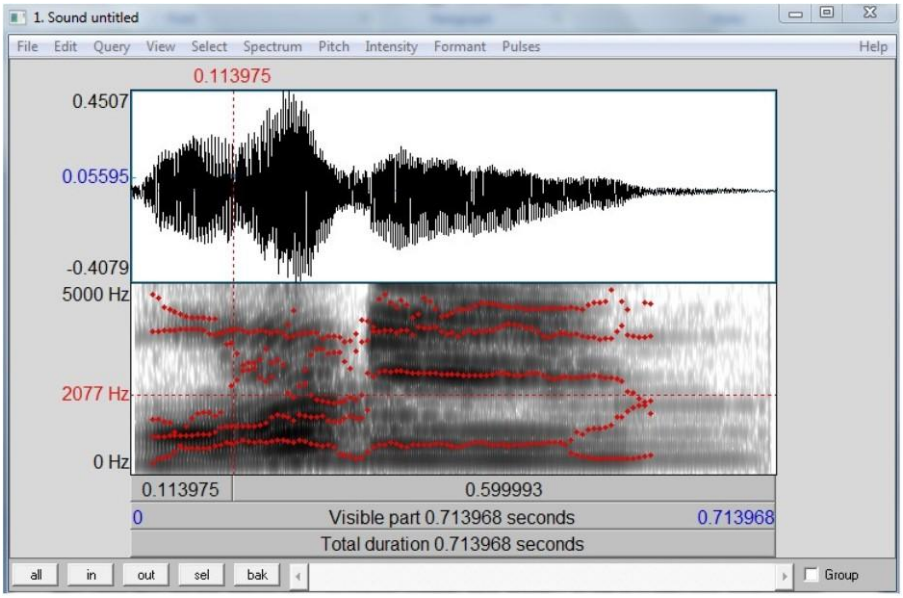

Figura 2: Exemplo Praat

2.1.3 WaveSurfer

O WaveSurfer [3] é uma biblioteca *open-source* utilizada para a customização e visualização de um sinal de fala. As aplicações típicas são a análise de fala/som e a anotação/transcrição de som. O WaveSurfer pode ser estendido por plug-ins e também incorporado noutras aplicações.

Em relação às anteriores é a ferramenta mais simples permitindo ao utilizador realizar operações básicas como copiar/colar um sinal de fala, normalização, remoção da componente DC, *Direct Current*, adição de reverberação no sinal, entre outras funcionalidades. A biblioteca

também tem a possibilidade de representar o sinal de fala em diversas formas como por exemplo espectrograma.

A figura 3 mostra um exemplo de execução do WaveSurfer.

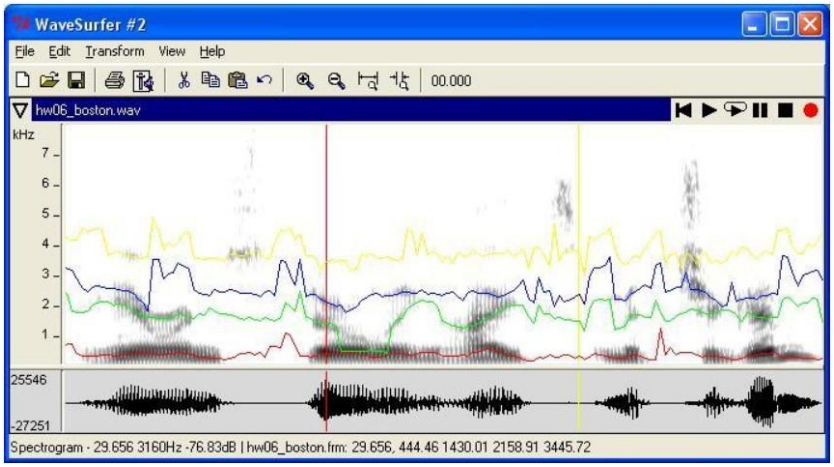

Figura 3: Exemplo WaveSurfer

2.1.4 DiagnoScope

O *DiagnoScope* [4], também conhecido como *DiagnoVoice*, é um programa que tem a capacidade de realizar análise de fala. O principal objetivo do *software* é ser simples e fácil de utilizar pois o seu público-alvo são iniciantes na prática da medicina. Este *software* possui uma lista de características como por exemplo, análise da vogal “a”, análise da voz de um orador, análise da frequência fundamental, criação de oscilogramas e espectrogramas, entre outras funcionalidades.

Este programa é bastante completo, pois permite guardar o historial médico de cada paciente e criar relatórios automáticos e detalhados.

Em termos funcionais a aplicação apresenta soluções sólidas no âmbito da saúde, no entanto a sua interface não é intuitiva. Na figura 4 mostra uma parte da documentação deste *software* de 2010.

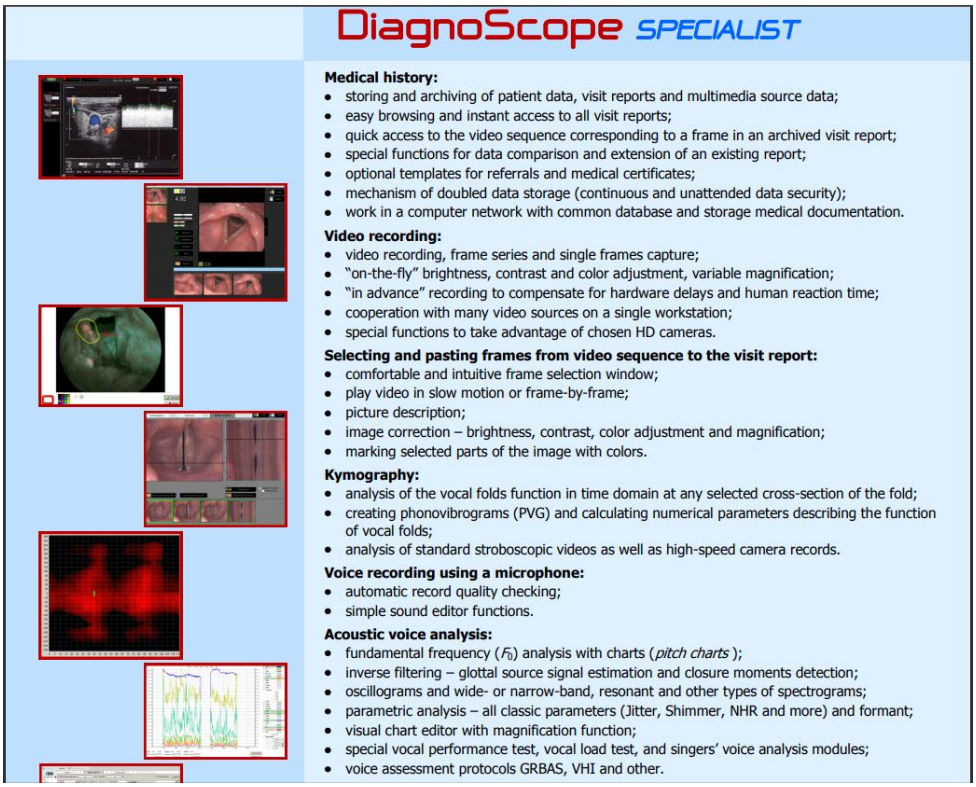

Figura 4: Documentação DiagnoScope

Com base na figura 4 conseguimos perceber o número de funcionalidades que o programa possui. Descrevendo as mais relevantes e começando pelo historial médico; onde são guardados os dados de cada paciente para facilitar a pesquisa, oferecendo um acesso rápido aos arquivos associados a cada paciente, gravação de vídeos; que permite uma análise profunda dos dados com a opção de editar o vídeo para ajustes de cor, vibração e contraste e a gravação de voz; com um editor de áudio para a análise do sinal de fala.

2.1.5 Aplicações de fala com protocolos de comunicação

Além das ferramentas previamente mencionadas, existem aplicações de fala que se destacam pela integração de protocolos de comunicação, permitindo a transmissão de dados em tempo real. Estas soluções são particularmente relevantes em contextos como telemedicina, assistentes virtuais e sistemas de monitorização remota.

2.1.5.1 Nuance Speech Server

O Nuance Speech Server [5] é uma plataforma avançada que oferece reconhecimento e síntese de fala, utilizando o protocolo MRCP [6] (Media Resource Control Protocol) para comunicação entre navegadores de voz e servidores de processamento de fala. O MRCP

permite que aplicações controlem remotamente recursos de fala, como reconhecimento automático de fala (ASR) [7] e conversão de texto em fala (TTS) [8], através de comandos padronizados.

A versão mais recente, MRCPv2, utiliza o SIP [9] (Session Initiation Protocol) para estabelecer sessões e o SDP [10] (Session Description Protocol) para configurar canais de comunicação. A transmissão de áudio pode ser realizada via RTP [11] (Real-time Transport Protocol) ou, alternativamente, através de mensagens SPEECHDATA sobre TCP [12], permitindo uma entrega mais eficiente e com menor latência, especialmente em ambientes com largura de banda limitada ou redes móveis instáveis.

Na figura 5 mostra a arquitetura que o Nuance Speech Server utiliza durante o seu algoritmo.

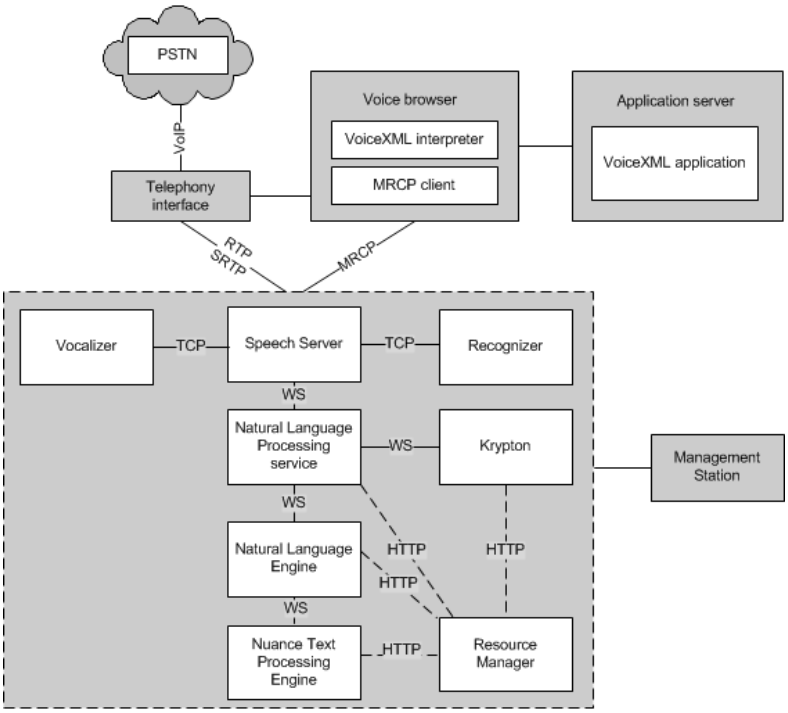

Figura 5: Arquitetura Nuance Speech Server

2.1.5.2 Assistive Context-Aware Toolkit (ACAT)

Desenvolvido pela Intel, o ACAT [13] (Assistive Context-Aware Toolkit) é um sistema de código aberto projetado para auxiliar pessoas com deficiências motoras severas a comunicarem-se através de movimentos mínimos, como contrações faciais. O ACAT integra sensores (como infravermelhos ou webcams) para detetar movimentos, uma interface de seleção de letras e palavras, e um sistema de previsão de texto que antecipa as intenções do usuário, convertendo-as em fala sintetizada.

O sistema foi originalmente criado para o físico Stephen Hawking e permite não apenas a comunicação verbal, mas também o controlo de aplicações como navegadores, editores de texto e leitores de PDF. A modularidade do ACAT permite a integração com diversos dispositivos e protocolos de comunicação, adaptando-se às necessidades individuais dos usuários.

2.1.5.3 Pictalk

O Pictalk [14] é uma aplicação voltada para a comunicação alternativa e aumentativa, especialmente útil para pessoas com distúrbios do neuro desenvolvimento. Através de uma interface baseada em pictogramas, o Pictalk permite que os usuários se comuniquem de forma eficaz, utilizando uma voz sintética para verbalizar as mensagens construídas. Além disso, oferece ferramentas como agendas visuais e criadores de recursos personalizados, facilitando a organização do dia a dia e promovendo a autonomia dos usuários.

3. Materiais e métodos

3.1 Materiais

Relativamente ao servidor, foi utilizado o Python com a versão 3.7, a biblioteca Librosa com a versão 0.10.1, a biblioteca Scipy com a versão 1.5.3, a biblioteca numpy com a versão 1.21.3, a biblioteca matplotlib com a versão 3.3.2 e as bibliotecas do protocolo HTTP e JSON incorporadas na instalação do Python.

Relativamente à aplicação, foi utilizado o Unity com a versão 2021.2.7f1 e um pacote de *ToolTip* chamado *SSTools*. Este pacote permite criar mensagens temporárias para o utilizador, com o objetivo de mostrar ao utilizador se a ação do utilizador é aceite ou se falta alguma parametrização na aplicação.

Para testes da aplicação foi utilizada a base de dados ALS com amostras de controlo e amostras patológicas e a base de dados MEEI com amostras de controlo e 3 tipos de amostras patológicas; edema, paralisia e nódulos das cordas vocais.

3.2 Métodos

As características dos sinais de fala, nomeadamente o RPPC, o LBST e o HBST, foram estimadas e implementadas no servidor Python.

3.2.1 RPPC

O RPPC [15] corresponde ao valor máximo extraído de uma autocorrelação normalizada. Para o seu cálculo, é crucial que a janela de análise contenha o mesmo número de períodos de pitch para todas as amostras, garantindo assim a comparabilidade entre elas. Para tal, são realizadas duas autocorrelações. A primeira é calculada com uma janela de 30 ms, com o objetivo de estimar o período de pitch. Em seguida, a segunda autocorrelação é realizada utilizando uma janela com duração igual a seis vezes o valor do período de pitch estimado. Após esta segunda autocorrelação, o sinal é normalizado pela autocorrelação da função de janela utilizada – neste caso, uma janela retangular. O valor final da segunda autocorrelação normalizada corresponde ao valor do RPPC.

A figura 6 mostra um exemplo de uma autocorrelação do algoritmo e a figura 7 mostra uma um exemplo de RPPC durante o algoritmo.

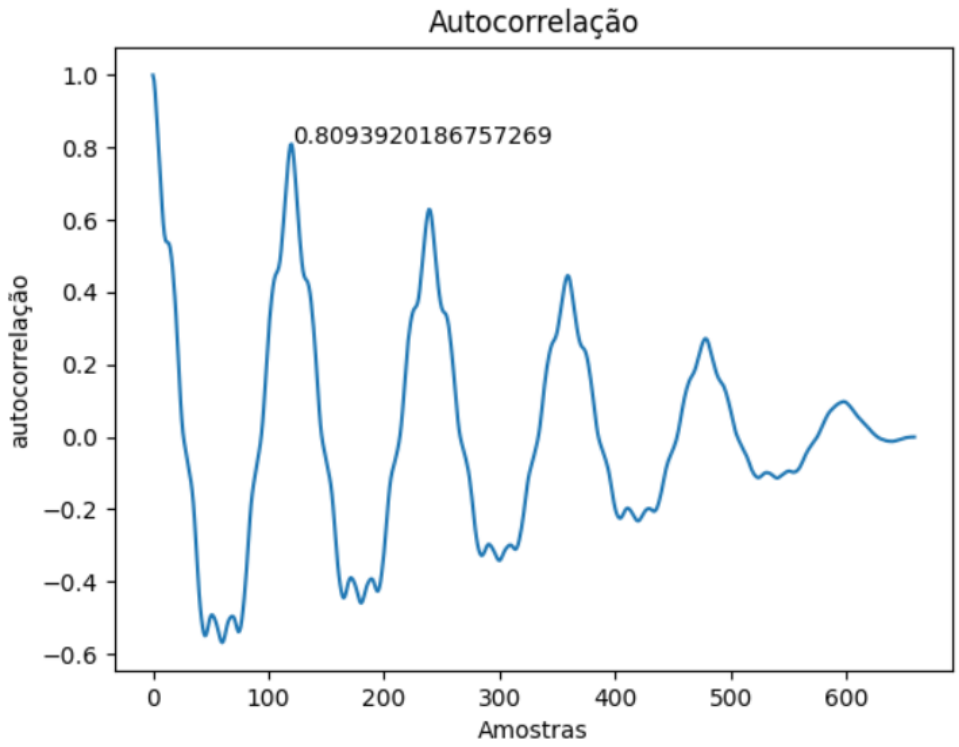

Figura 6: Exemplo autocorrelação

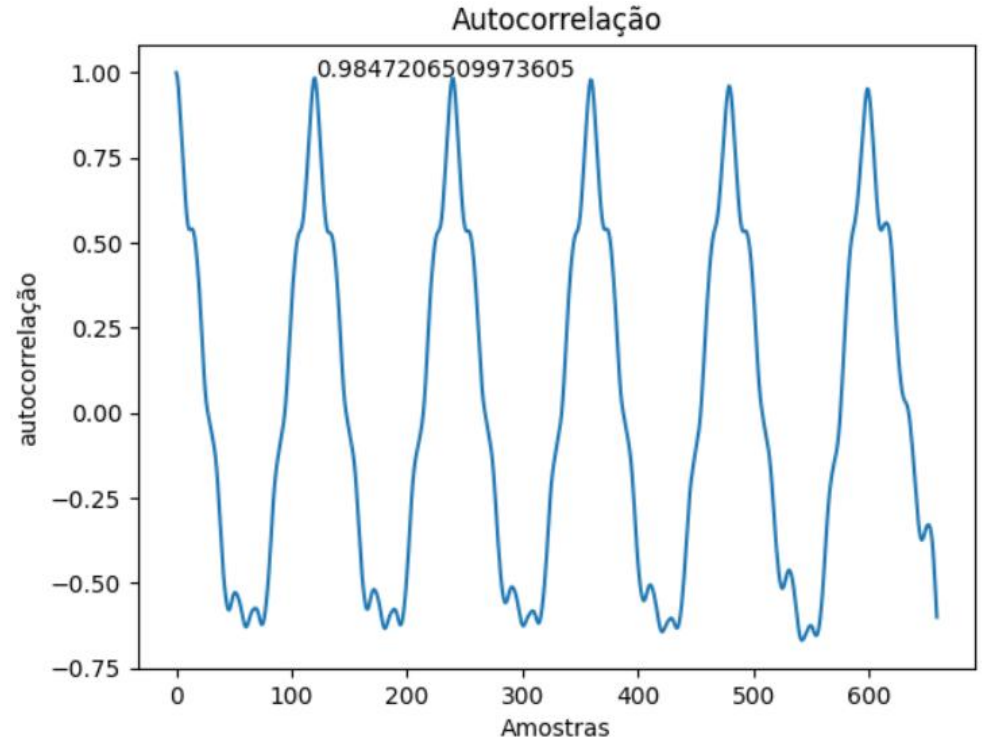

Figura 7: Exemplo RPPC

3.2.2 LBST e HBST

O LBST e o HBST [16] contem informação que permite identificar se o orador tem patologia ou se é saudável. Esta informação é obtida através de um declive, que é calculado entre a zona da primeira harmónica, e a zona das harmónicas do primeiro formante.

Em relação aos valores do LBST, existe uma grande probabilidade de o orador ser saudável caso o este valor seja positivo. Caso contrário é provável que o orador possua uma patologia da voz. Este facto deve-se à existência de patologia provocar um aumento da degradação vogal e á diminuição da riqueza espetral do sinal nas harmónicas do sinal de fala. O LBST é o primeiro declive calculado que corresponde aos dois máximos locais entre os valores [0, 420] Hz e]420, 1260] Hz, que correspondem aos intervalos *First Band Maximum Energy* (FBME) e *Second Band Maximum Energy* (SBME) respetivamente.

Em relação ao HBST, este valor ajuda a perceber a atenuação da reta formada pelos dois valores. Se a reta tiver menor declive, significa que o orador tem uma maior possibilidade de ser patológico. Caso contrário o orador tem uma maior possibilidade de ser saudável. O HBST é o segundo declive calculado com os máximos locais entre]420, 1260] Hz e [2100, 5800] Hz, que correspondem aos intervalos SBME e TBME, *Third Band Maximum Energy*.

O LBST é descrito em (3), o HBST é descrito em (4) e na figura 8 apresenta um exemplo dos valores LBST e HBST numa trama de um sinal de fala.

$$LBST = \frac{SBME - FBME}{SBMEfreq - FBMEfreq} \tag{3}$$

$$HBST = \frac{TBME - SBME}{TBMEfreq - SBMEfreq} \tag{4}$$

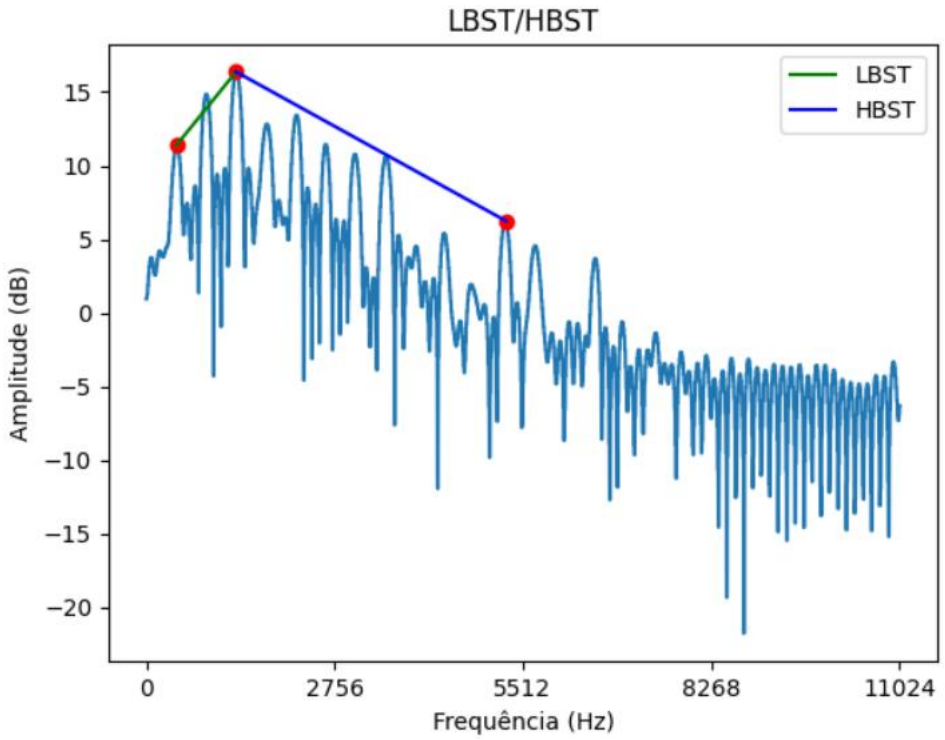

Figura 8: Exemplo LBST/HBST

3.3 Descrição das bases de dados

Para os testes da aplicação e do código realizado em Python, foram usadas duas bases de dados com características diferentes. Ambas as bases de dados utilizam o mesmo tipo de fala, vogal /a/ sustentada.

A primeira base de dados [17] possui 39 ficheiros de controlo e 15 ficheiros de patologia, onde 29 oradores são masculinos e 25 oradores são femininos, e têm uma frequência de amostragem de 22050 Hz. A base de dados é designada por ALS.

A segunda base de dados [18] possui 89 ficheiros de controlo e 128 ficheiros de patologia, que por sua vez estão divididos em três tipos de patologias, 42 ficheiros de edema, 19 ficheiros de nódulos e 67 ficheiros de paralisia. Os ficheiros da base de dados estão amostrados a 8000 e 44000 Hz e tem o nome de MEEI, onde foram apenas usadas amostras com patologia de paralisia nas cordas vocais.

3.4 Análise das bases de dados

De modo a verificar o funcionamento das funções desenvolvidas, foram desenhados diversos gráficos para as funções implementadas para as duas bases de dados previamente descritas.

Começando pelo RPPC foram desenhados dois gráficos de pontos que representam a média do valor RPPC por sinal nos grupos de controlo e de paralisia nas cordas vocais. Os gráficos estão representados nas figuras 9 e 10.

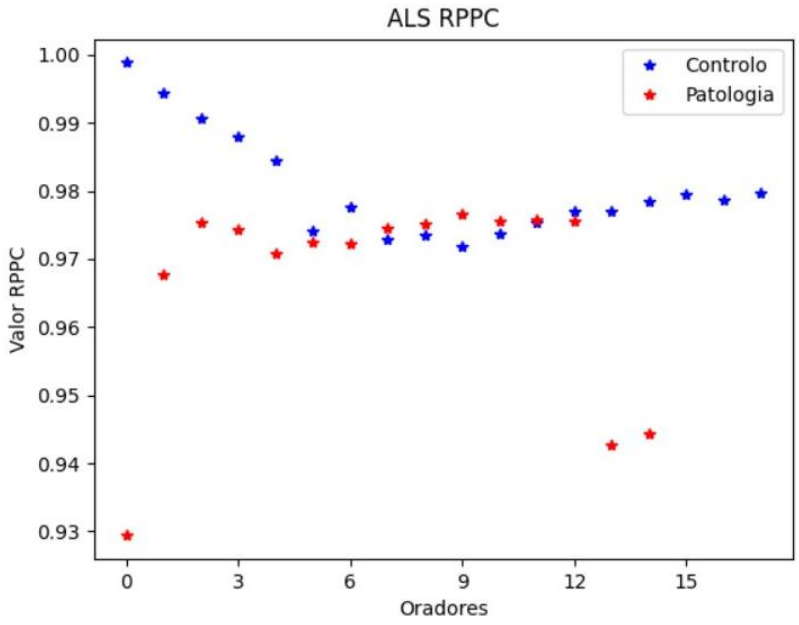

Figura 9: ALS RPPC

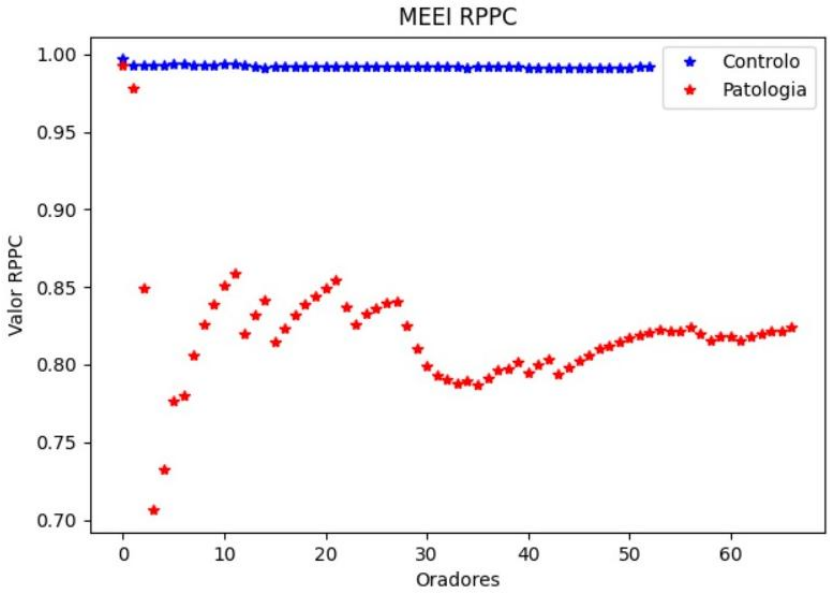

Figura 10: MEEI RPPC

Com o RPPC implementado e os gráficos desenhados, consegue-se observar que os gráficos apresentam resultados distribuição dos dados. Na figura 10 nota-se uma diferença no valor do RPPC entre os dados de controlo e de paralisia.

No gráfico referente à base de dados ALS, observa-se que os valores de RPPC dos oradores com patologia (em vermelho) são, na sua maioria, inferiores aos do grupo de controlo (em azul). Esta diferença é mais pronunciada nos primeiros oradores (ex. orador 0, com um valor em torno de 0.93), sendo que os restantes valores tendem a aproximar-se dos do grupo controlo, mas mantendo-se ligeiramente mais baixos.

No gráfico relativamente à base de dados MEEI o contraste entre os dois grupos é mais evidente. Os sujeitos com patologia apresentam valores de RPPC significativamente mais baixos, com uma média visivelmente inferior (entre ~0.72 e ~0.87), em comparação com o grupo de controlo, cujos valores se mantêm consistentemente muito próximos de 1.

Esta diferença clara indica que os indivíduos doentes apresentam uma degradação substancial na sua fala

De seguida foram desenhados os gráficos para as funções do LBST e HBST para ambas as bases de dados. Para a MEEI foi utilizada a patologia paralisia outra vez para manter a consistência na amostra e análise das bases de dados. As figuras 11, 12, 13 e 14 representam o LBST e HBST para ambas as bases de dados.

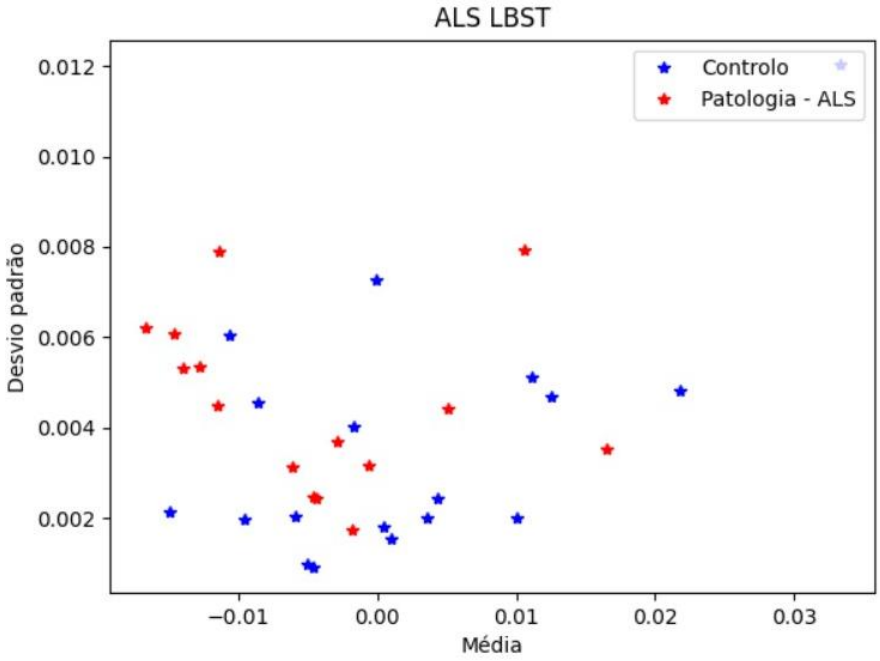

Figura 11 ALS LBST

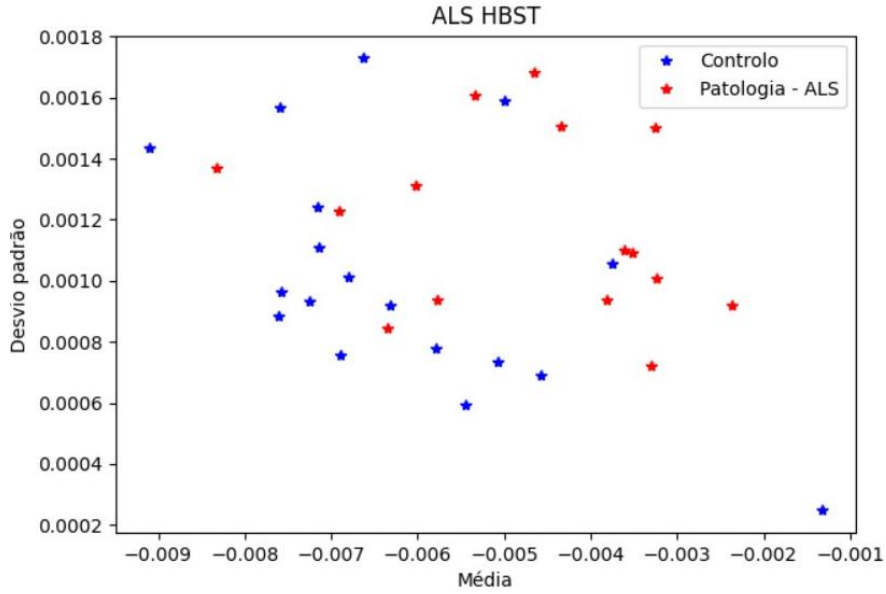

Figura 12: ALS HBST

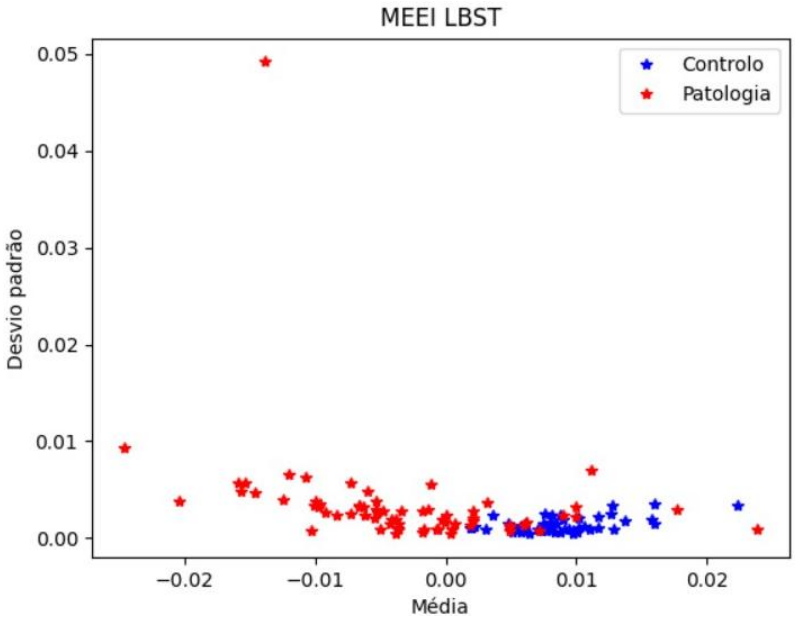

Figura 13: MEEI LBST

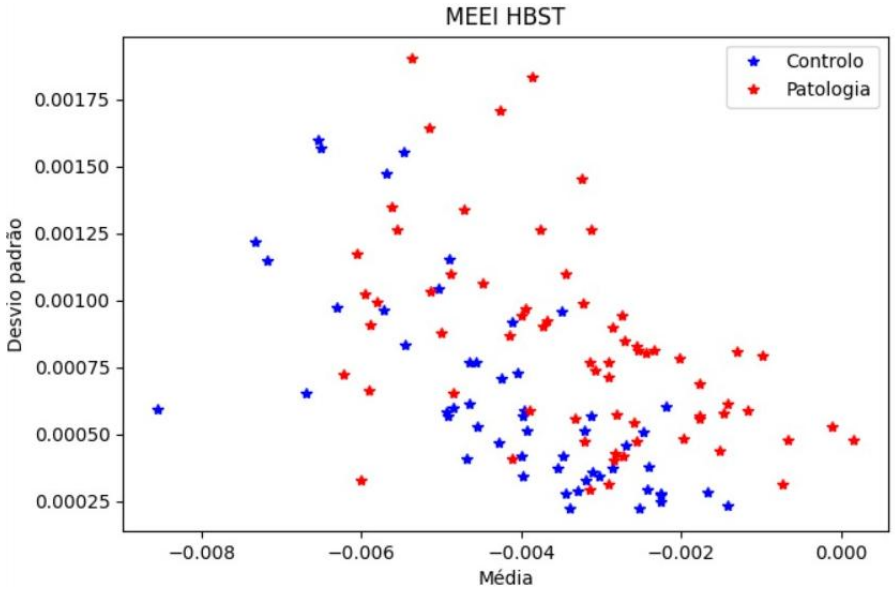

Figura 14: MEEI HBST

Os resultados de ambas as funções para as duas bases de dados apresentam resultados promissores. Esta conclusão é tirada porque nos 4 gráficos previamente desenhados observa-se informação relevante, nomeadamente a distribuição dos dados, permitindo assim classificar os dados como saudável ou patologia.

Nos gráficos relativos à base de dados ALS, os dados de controlo apresentam uma distribuição mais junta e concentrada numa área enquanto que nos dados de patologia os dados estão mais espalhados pelo espaço de análise.

Relativamente à base de dados MEEI a mesma distribuição de dados é observada. No gráfico da figura 13 os dados de patologia apresentam um valor inferior de média em relação

aos dados de controlo indicando que os sujeitos avaliados têm uma degradação significativa na fala. Na figura 14, HBST, também se observa o comportamento previamente descrito.

As figuras seguintes apresentam os gráficos de bigodes para o RPPC, LBST e HBST para uma melhor interpretação dos resultados. Estes gráficos demonstram melhor a distribuição de valores dos diferentes gráficos, permitindo uma melhor análise dos resultados obtidos. As figuras 15, 16, 17, 18, 19 e 20 representam os gráficos de bigodes para o RRPC, LBST e HBST para as duas bases de dados.

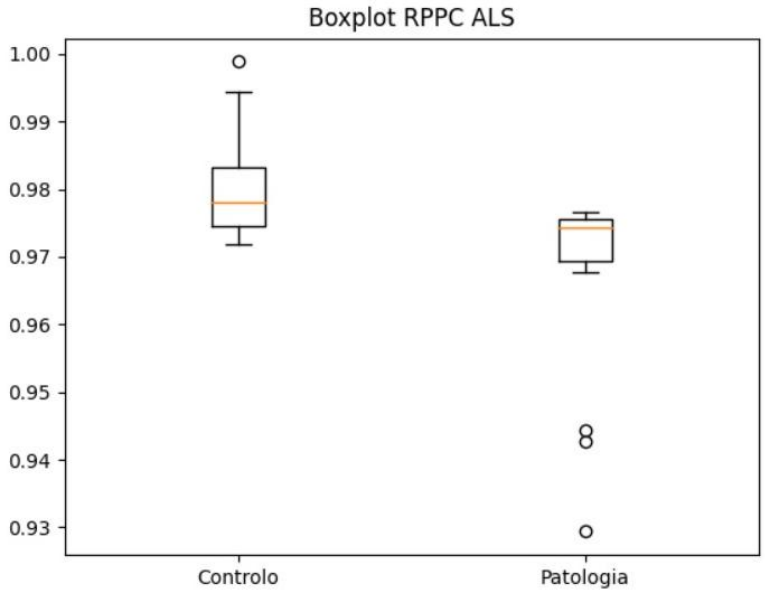

Figura 15: Boxplot RPPC ALS

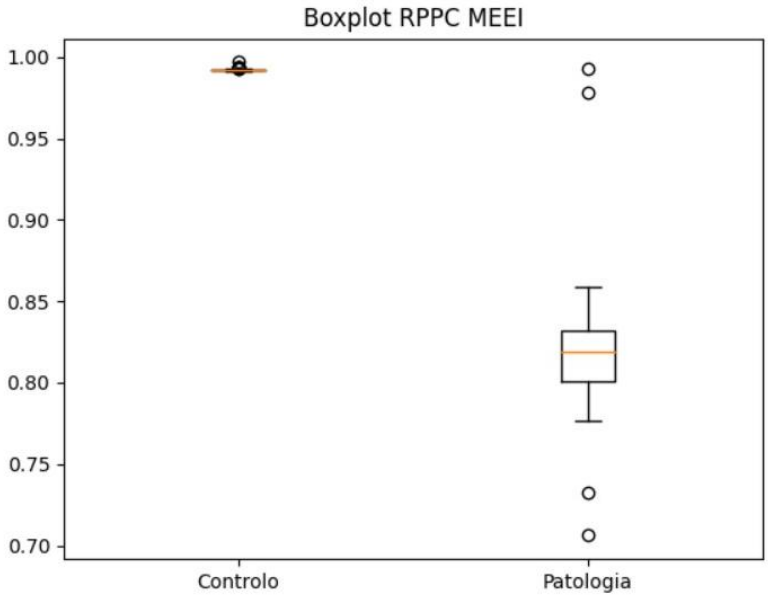

Figura 16: Boxplot RPPC MEEI

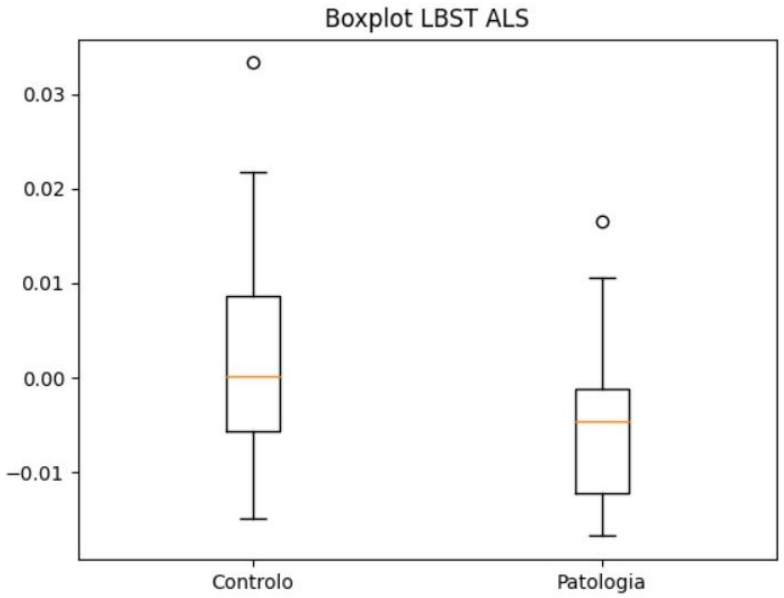

Figura 17: Boxplot LBST ALS

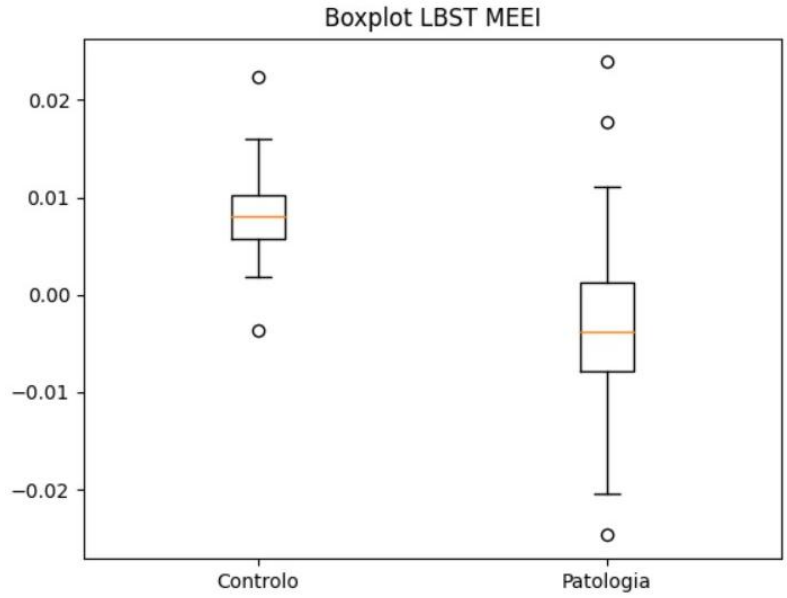

Figura 18: Boxplot LBST MEEI

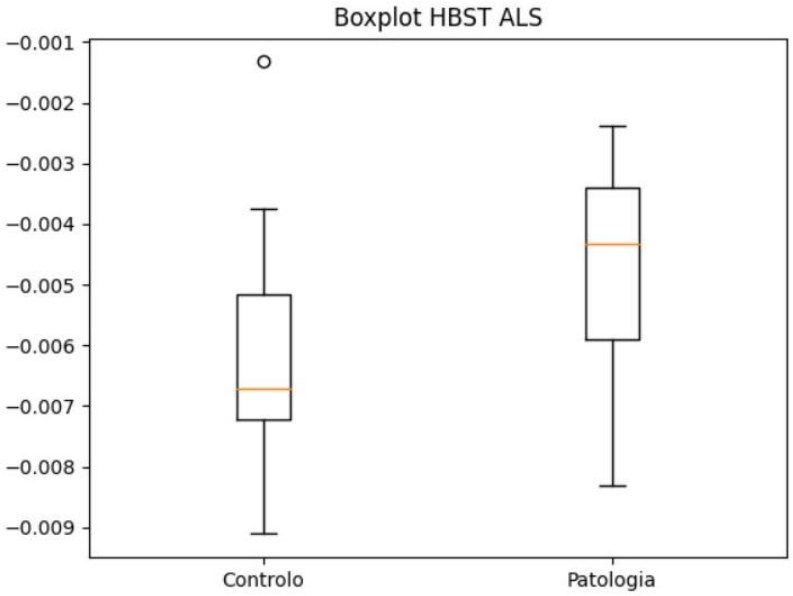

Figura 19: Boxplot HBST ALS

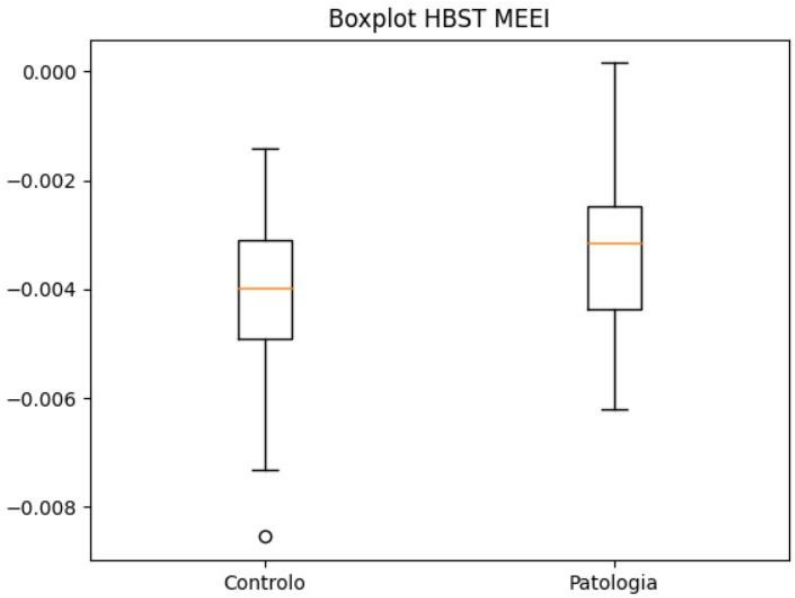

Figura 20: Boxplot HBST MEEI

Os boxplots apresentados nas figuras 15 e 16 fornecem uma comparação das distribuições dos valores de RPPC. No dataset ALS, a mediana dos valores de RPPC para o grupo Controlo é de aproximadamente 0,978, enquanto no grupo Patologia é ligeiramente inferior, cerca de 0,975. A dispersão dos dados é relativamente baixa em ambos os grupos, indicando uma consistência nos valores centrais de RPPC. No entanto, o grupo Patologia mostra alguns valores significativos abaixo do limite inferior (valores próximos de 0,93), sugerindo a presença de oradores com patologia, e o grupo Controlo apresenta um conjunto de valores acima do limite superior, com um valor próximo de 1,0, indicando oradores saudáveis. Por outro lado, no dataset MEEI, o grupo Controlo demonstra uma mediana elevada, próxima de 1,0, e uma

dispersão mínima, o que reflete num grupo de oradores saudáveis. Em contraste, o grupo Patologia mostra uma mediana consideravelmente mais baixa, aproximadamente 0,82, e uma maior dispersão dos dados, evidenciada por um intervalo interquartil (IQR ou mediana) mais amplo. Adicionalmente, o grupo Patologia inclui múltiplos valores abaixo do limite inferior, com valores próximos de 0,7, apontando para casos com um desempenho marcadamente reduzido.

Os boxplots apresentados nas figuras 17 e 18 fornecem uma comparação das distribuições dos valores de LBST. No dataset ALS, o grupo Controlo para LBST mostra uma mediana próxima de zero com um IQR moderado, o que reflete uma dispersão equilibrada. O grupo Patologia, por sua vez, apresenta uma mediana ligeiramente negativa, indicando uma tendência para valores inferiores, e uma dispersão similar à do grupo Controlo. Para o dataset MEEI, o grupo Controlo de LBST revela uma mediana ligeiramente positiva (cerca de 0,008) e uma dispersão relativamente pequena. Em contraste, o grupo Patologia apresenta uma mediana negativa (em torno de -0,004) e maior dispersão, com múltiplos valores abaixo do limite inferior. A diferença clara entre as medianas destes grupos sugere um impacto relevante da condição patológica nesta métrica.

Quanto ao HBST, figuras 19 e 20, no dataset ALS os valores são negativos em ambos os grupos. O grupo Controlo apresenta uma mediana mais baixa (aproximadamente -0,0068) do que o grupo Patologia (em torno de -0,0045). Embora a dispersão seja similar, os valores menos negativos no grupo Patologia podem sugerir um aumento relativo nesta métrica para indivíduos com patologia, e há um valor no grupo Controlo abaixo do limite inferior. Finalmente, no dataset MEEI, o grupo Controlo de HBST mostra uma mediana negativa (cerca de -0,004) com dispersão moderada e a presença de um valor baixo. Já o grupo Patologia tem uma mediana ligeiramente superior (perto de -0,003), indicando uma tendência a valores maiores nesta métrica, com uma dispersão ligeiramente menor e sem a observação de valores altos.

4.Implementação do modelo

Este capítulo apresenta o manual onde se descreve a aplicação do ponto de vista do utilizador, a comunicação entre o Unity e o Python e uma explicação das metodologias usadas e implementadas no projeto.

4.1 Manual

A aplicação foi desenvolvida no Unity3D e possui 8 páginas diferentes que serão descritos neste capítulo.

4.1.1 Página inicial

A primeira página é composta pelo ícone da aplicação e por um botão que dá início ao programa, ilustrado na figura 21.

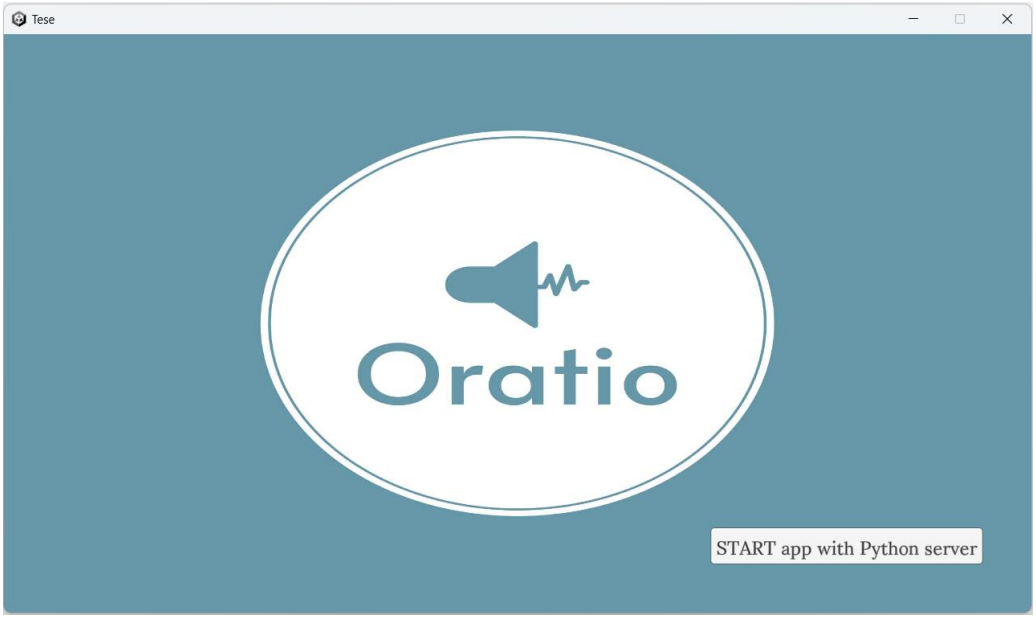

Figura 21: Página inicial

4.1.2 Menu

No menu inicial o utilizador pode realizar 3 operações diferentes, gravar um sinal de fala, importar uma função para análise de sinais de fala ou chamar as funções presentes na base de dados. Na figura 22 está representado este menu.

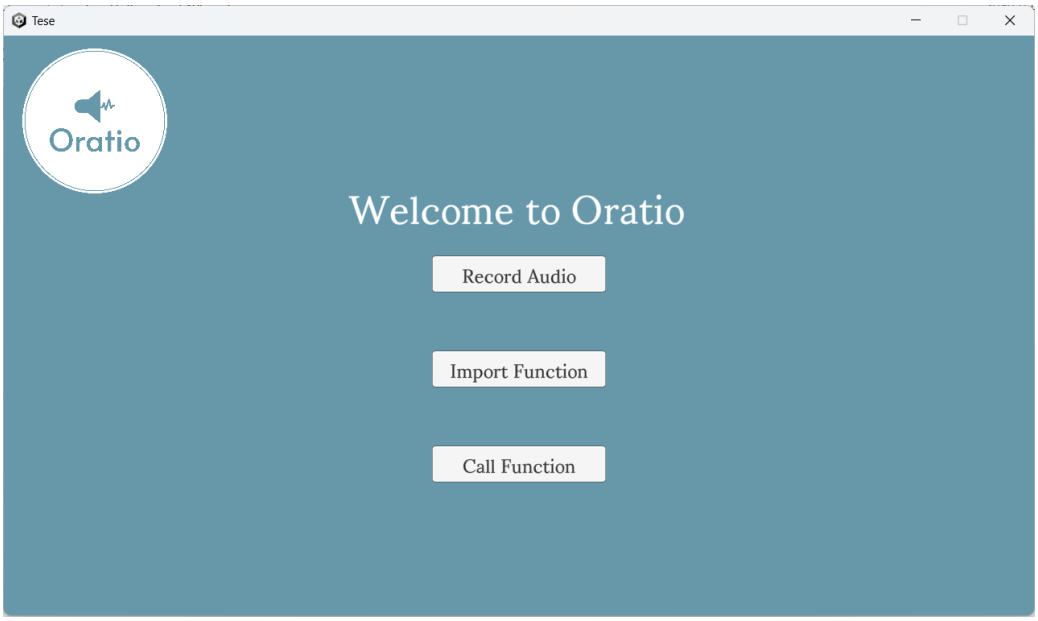

Figura 22: Menu

4.1.3 Gravar um sinal de fala

No menu da aquisição de sinais de fala existem botões que permitem iniciar e parar a aquisição do sinal de fala e uma caixa de texto onde o utilizador pode especificar o tempo máximo da gravação. Na figura 23 mostra esta página.

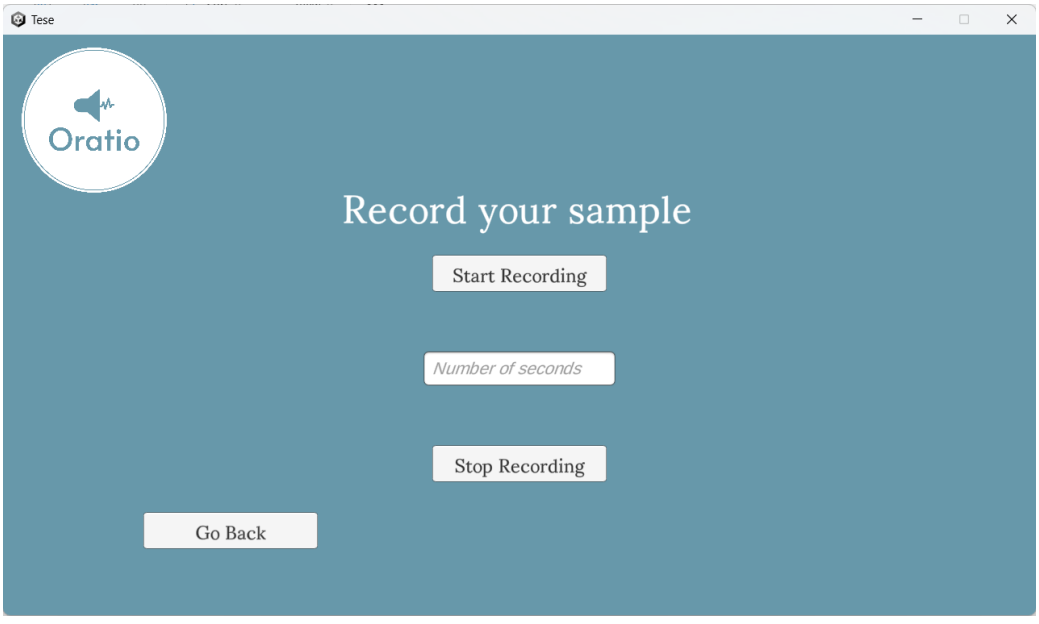

Figura 23: Página de gravação

4.1.4 Importar função

Outra das operações disponíveis é a importação de uma nova função de Python. Este importe é feito através de um ficheiro de texto com a extensão .txt. A função deverá ter o nome da função e os atributos da função encapsulados pelos caracteres "(" e ")". Cada atributo da função deverá ter um tipo atribuído como por exemplo "*String*" ou *"int"* e cada atributo pode ser documentado para o utilizador, mostrando essa documentação na aplicação. Nas figuras 24 e 25 estão representados um exemplo de um importe e a execução do botão, visto que não é necessária uma página.

```
LBST_HBST(string function=Função a escolher LBST/HBST, int n_fft=Janela da FFT, float hop_length=Andamento em segundos, float win_length=Tamanho da
janela em segundos, string window=Tipo de janela)
```

Figura 24: Exemplo de importe

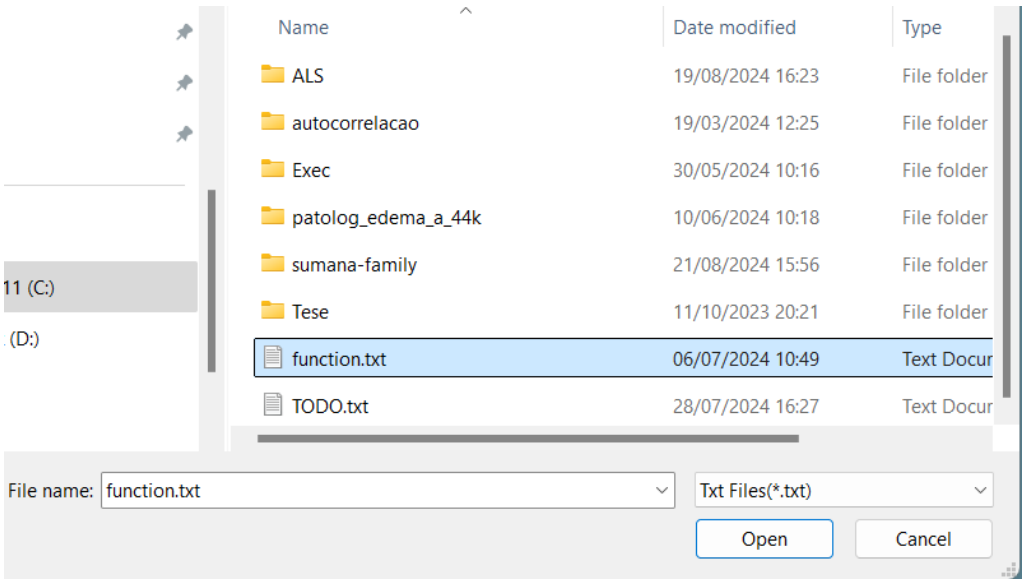

Figura 25: Seleção do ficheiro de texto

A base de dados das funções da aplicação é constituída por todos os métodos importados pelo utilizador, por ordem de inserção. O ficheiro da base de dados encontrado no caminho persistente, ou em inglês *persistentdataPath*. A diretoria da base de dados será algo semelhante a "C:\Users\<currentUser>\AppData\LocalLow\DefaultCompany\Tese".

Esta base de dados é um ficheiro de texto que é sempre atualizado quando o utilizador importa uma nova função. Esta solução evita que a aplicação precise de ser reiniciada para atualizar a informação da aplicação e evita processamento extra caso a base de dados fosse remota.

4.1.5 Seleção de função

Seguindo o fluxo da aplicação, é apresentado ao utilizador uma página onde é apresentado uma lista de funções presentes na base de dados para o utilizador escolher a operação que quer fazer. A lista é apresentada num elemento *dropdown* onde este vai buscar toda a informação dinamicamente à base de dados, o que significa que se o utilizador quer adicionar outra função sem fechar a aplicação, a nova função irá estar presenta no elemento *dropdown.* Na figura 26 mostra a execução da página.

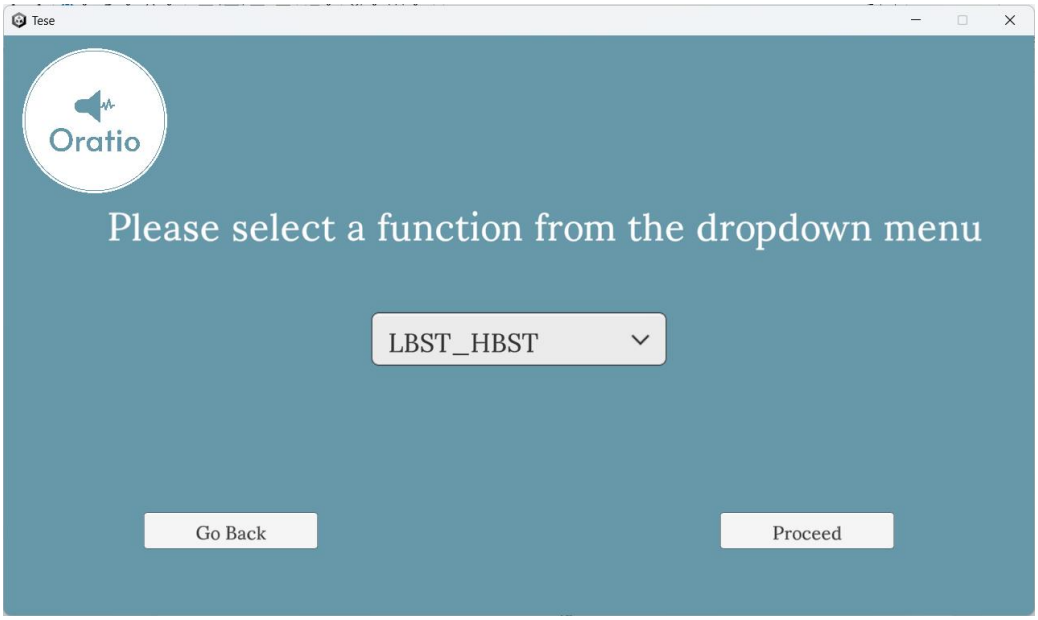

Figura 26: Página de funções

4.1.6 Seleção de ficheiro ou pasta

Com a função de análise selecionada, o utilizador tem agora a escolha de um ficheiro de fala ou uma pasta com vários ficheiros de fala. Esta seleção é feita através de um *file browser* implementado no Unity.

Na figura 27 é mostrado a página de seleção de dados.

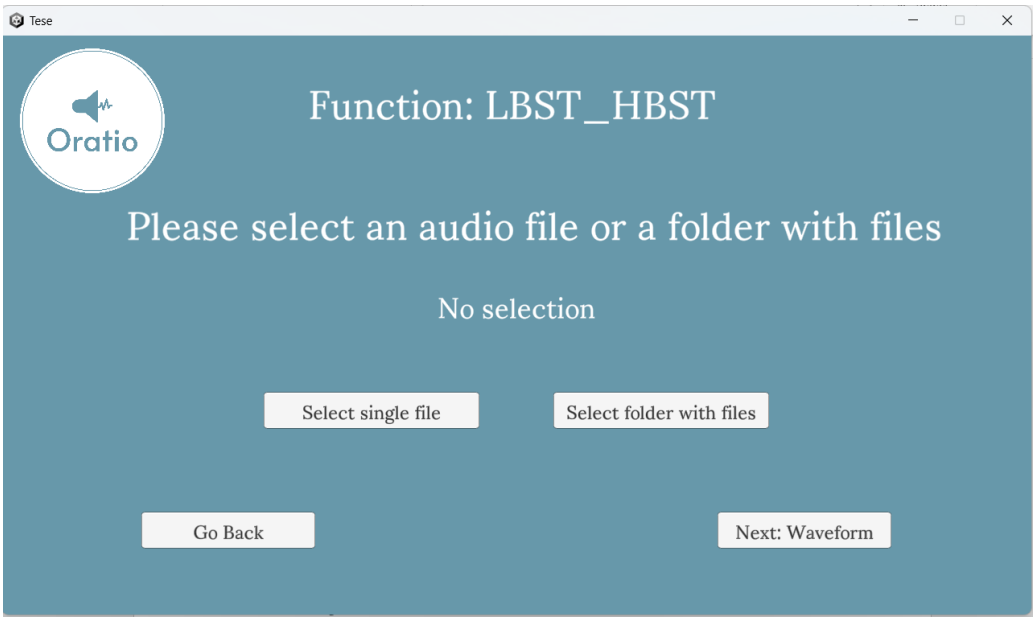

Figura 27: Página de seleção de dados

4.1.7 Seleção de amostra

Caso o utilizador tenha escolhido um ficheiro de fala, este tem a opção extra de selecionar um ponto no sinal desenhado, marcando assim o início da trama que quer calcular, ou selecionar o sinal por completo. Estas duas escolhas são feitas ao pressionar sobre o sinal ou clicando no botão do lado direito respetivamente.

A figura 28 mostra a página com uma amostra selecionada e a figura 29 mostra a página com a seleção por completa do sinal.

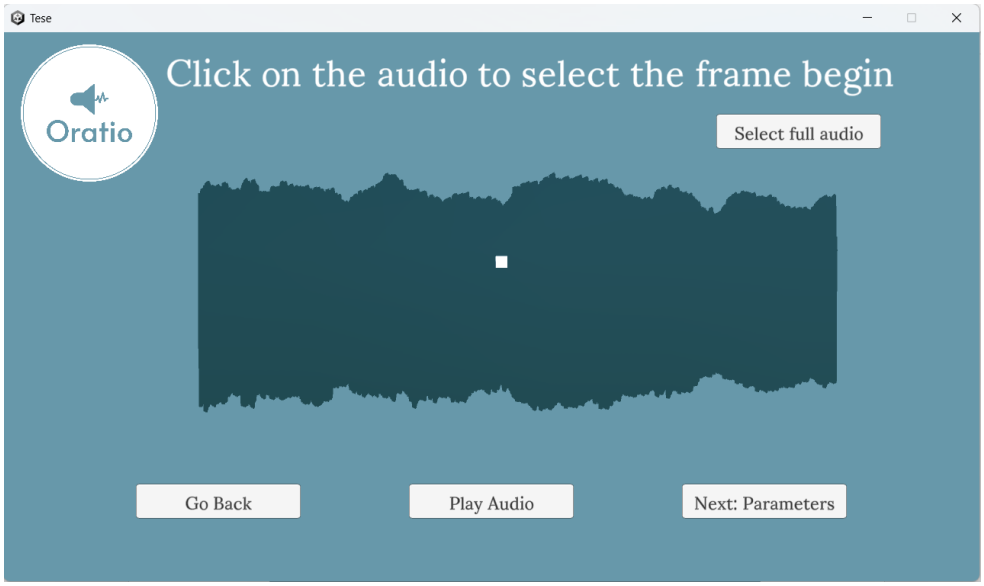

Figura 28: Seleção de um ponto no sinal

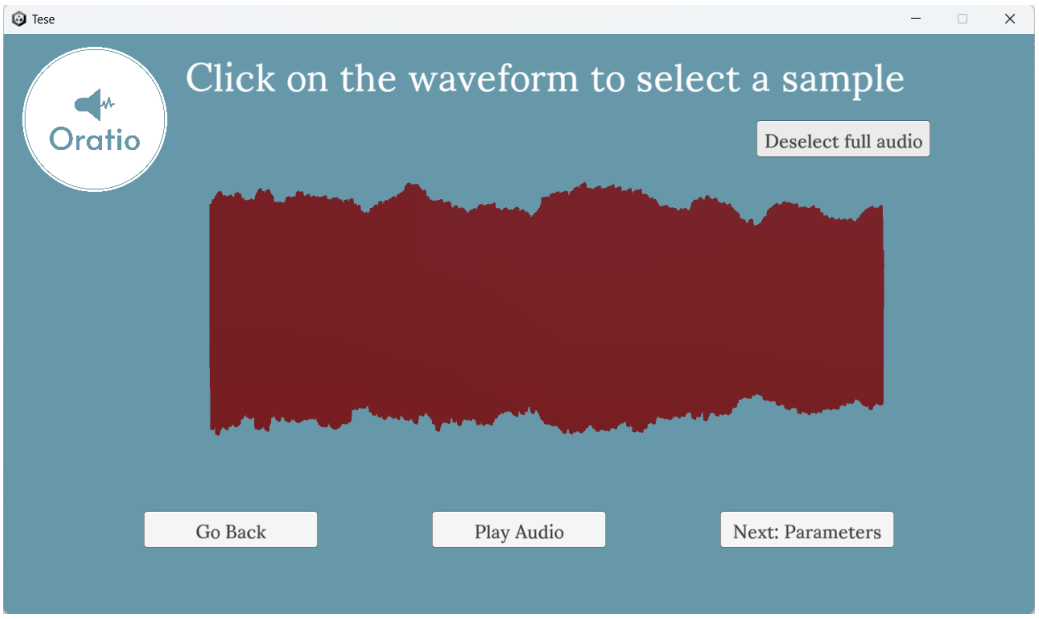

Figura 29: Seleção do sinal por completo

4.1.8 Atribuição de valores

Após a seleção do sinal de fala é apresentada uma janela com a informação dos parâmetros da função. Cada campo da função é constituído por um texto que representa o tipo e o nome da variável com um objeto associado que representa a documentação dessa mesma variável.

A figura 30 mostra a página de atribuição de valores.

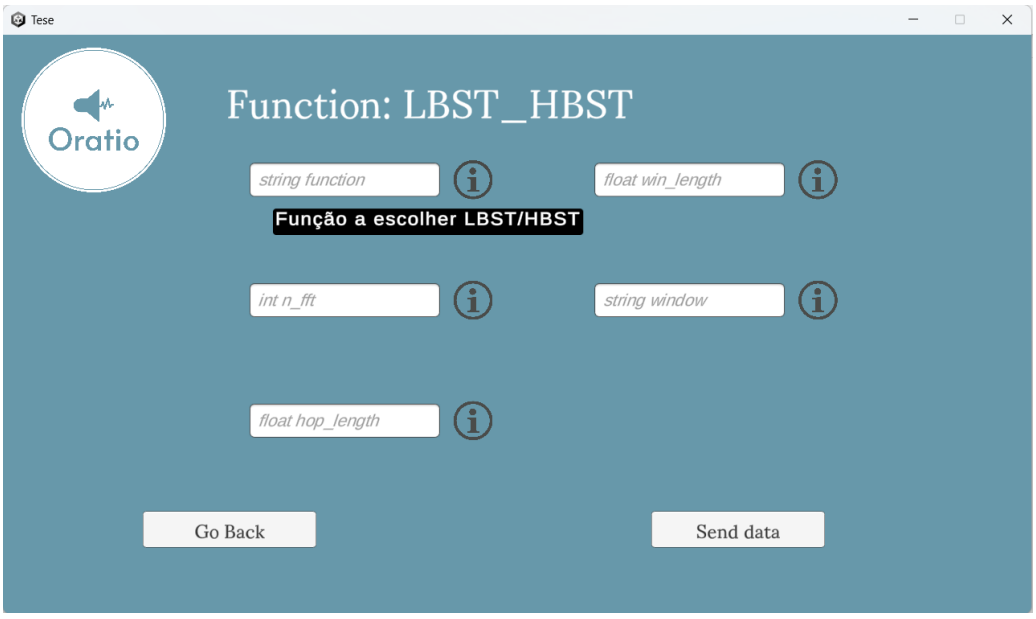

Figura 30: Atribuição de valores

4.1.9 Página de resultados

Por fim são mostrados os resultados da função e parâmetros escolhidos pelo utilizador numa página final. Na figura 31 é possível ver esta página com os resultados da função LBST/HBST com o ficheiro 002.wav para o sinal todo e com os parâmetros “Lbst”, 4096, 0.03, 0.015, “Boxcar”, respectivamente de acordo com a figura 30.

Figura 31: Resultados

4.2. Implementação

4.2.1 Modelação

A aplicação consiste em criar uma interface modular e dinâmica usando o Unity para fazer a comunicação com a plataforma Python, através de um servidor. Neste subcapítulo apenas são descritas as classes e o planeamento envolvido de cada classe e nos próximos subcapítulos é que são aprofundados os conceitos abordados na modelação do projeto.

As classes foram desenhadas utilizando a linguagem UML [19], *Unified Modeling Language* para a representação de cada classe utilizada no Unity.

As classes implementadas em Unity têm o objetivo de tratar e representar os dados escolhidos pelo utilizador e fornecer ao servidor Python estes mesmos dados. O servidor Python ao receber os dados do Unity, como a função selecionada e os parâmetros dessa mesma função, vai alimentar a aplicação Unity com o resultado da função escolhida.

A aplicação requer que o utilizador escolha um servidor Python ao qual a aplicação se deve ligar, logo foi desenhada uma classe para desenvolver esta ação. A classe possui 1 variável e 1 método. A variável é a definição da extensão do ficheiro que o utilizador deve escolher, com o objetivo de ser intuitivo de perceber o que a aplicação está a pedir, e o método que inicia o processo do servidor.

A figura 32 mostra o diagrama da classe do servidor Python.

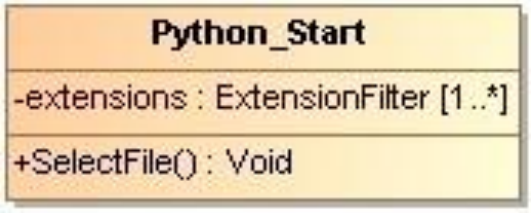

Figura 32: Classe Python_Start

Foi desenhada uma classe para tratar da execução principal da aplicação. Esta classe é extensiva e todos os métodos estão relacionados com as funcionalidades principais da aplicação, desde o tratamento da informação de seleção dos diferentes botões ao envio dos dados para o servidor Python. A classe possui 34 variáveis e 18 métodos para o controlo do fluxo da aplicação. A figura 33 mostra a classe principal da aplicação.

Figura 33: Classe Python_Connection

Foi desenvolvida uma classe capaz de desenhar o sinal de fala que o utilizador escolheu para este o conseguir observar e interagir com o sinal. A interação com o sinal consiste na seleção do início de uma *frame* para ser analisada ou se o utilizador quiser tem a possibilidade de selecionar o sinal por completo. Foi também desenvolvido a capacidade de reprodução do sinal para o utilizador conseguir confirmar a seleção do sinal de fala que pretende. A classe possui 16 variáveis e 9 métodos. A classe além de desenhar o gráfico, alimenta o fluxo principal da aplicação para a troca de informação sobre o tipo de dados que o utilizador quer analisar. A figura 34 mostra o diagrama da classe GraphPlot, responsável pelo desenho/representação do gráfico.

Figura 34: Classe GraphPlot

O utilizador deve conseguir gravar um sinal de fala de modo a perceber toda a interação necessária e o estado da gravação. A funcionalidade implementada tem o extra de proporcionar a duração do sinal, consoante as necessidades do utilizador. O utilizador também tem a opção de parar o sinal antes que o tempo segue ao fim caso o deseje. Durante o processo de gravação é importante que o utilizador consiga perceber quando é que o sinal está a ser gravado ou não, portanto foi atribuída a cor vermelha ao botão de gravação quando a gravação é iniciada e a cor é retirada quando a gravação acaba quer por tempo quer pela interrupção do utilizador. A classe possui 6 variáveis e 8 métodos. As variáveis e métodos estão diretamente relacionados com o tratamento de informação capturada pelo microfone do utilizador para depois ser gravado num ficheiro de áudio. A figura 35 mostra o diagrama da classe MicrophoneRecorder.

Figura 35: Classe MicrophoneRecorder

Como referido anteriormente, o objetivo é integrar as funções do servidor Python na interface Unity. Isto é feito através de um importe de um ficheiro de texto que possui a função descrita pelo utilizador. Esta função deverá seguir um protocolo de modo que toda a informação da função seja representada na interface. Foi desenhada uma lista de *Dropdown* que possui todas as funções que o utilizador importou para a aplicação. Foi escolhida a lista *Dropdown* porque possui um *design* simples e a sua utilização é intuitiva. Esta lista irá obter as funções importadas, acedendo a uma base de dados que é criada quando o utilizador importa a primeira função, e deverá ser atualizada sempre que o utilizador aceda à lista de funções, tornado assim este processo dinâmico e não requer o reinício da aplicação caso o utilizador quiser adicionar diversas funções na mesma execução. A classe possui 2 variáveis e 1 método. O método consiste na leitura da base de dados e a atualização de valores da lista *Dropdown*. A figura 36 mostra o desenho da classe DropdownClass.

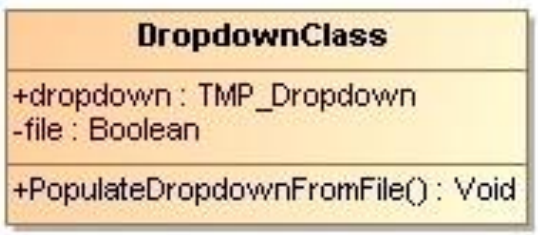

Figura 36: Classe DropdownClass

O melhor modo de integrar esta documentação é através de *Tooltips*. Uma *Tooltip* [20] é uma mensagem temporária que ocorre quando o utilizador passa o cursor do rato por cima de uma imagem, ícone ou qualquer elemento da interface gráfica. Este mecanismo foi desenhado através de 2 classes; uma que trata da mensagem, dados e informação corrente de cada objeto correspondente à posição do cursor e uma classe que trata dos eventos do cursor. A *Tooltip* tem como referência um ícone de informação, figura 37.

Figura 37: Ícone de informação

A classe que trata dos dados possui 6 variáveis e 7 métodos. A classe é responsável pela informação da mensagem, bem como a sua representação gráfica. Esta classe deve comunicar com a classe principal de modo a ir buscar a documentação relativa à variável corrente de análise. A mensagem e a posição desta deve ser dinâmica e simples de modo a não sobrecarregar a aplicação com informação.

A classe que trata dos eventos é uma simples classe com 1 variável e 2 métodos. A variável corresponde à mensagem que é apresentada ao utilizador e os métodos correspondem ao evento do cursor passar por cima do ícone e sair do ícone. Estes eventos são interfaces nativas do Unity, diminuindo assim a complexidade da classe. As figuras 38 e 39 mostram os diagramas de classe para a criação de uma *Tooltip*.

Figura 38: Class Tooltip

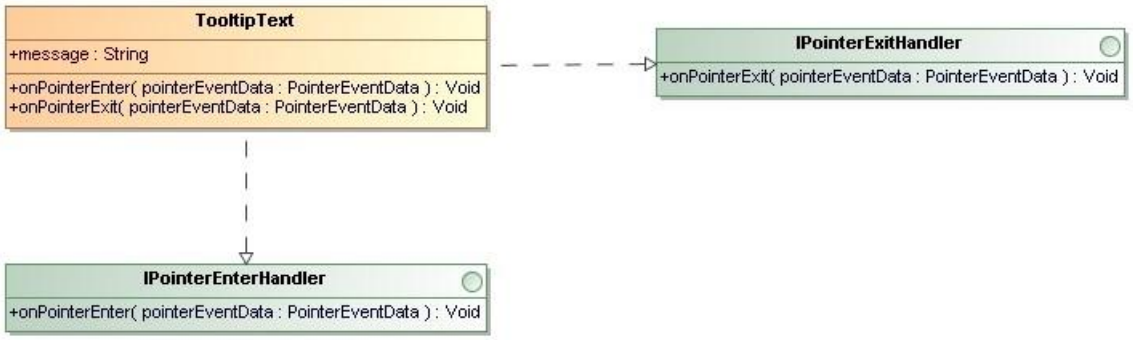

Figura 39: Classe ToolTipText

Com as classes todas desenhadas foram feitas as relações necessárias para o funcionamento da aplicação. O desenho final ajuda a perceber como é que a aplicação irá funcionar, diminui a complexidade da aplicação e consegue-se observar as relações entre as classes. A figura 40 representa a arquitetura por completa, pronta para ser desenvolvida.

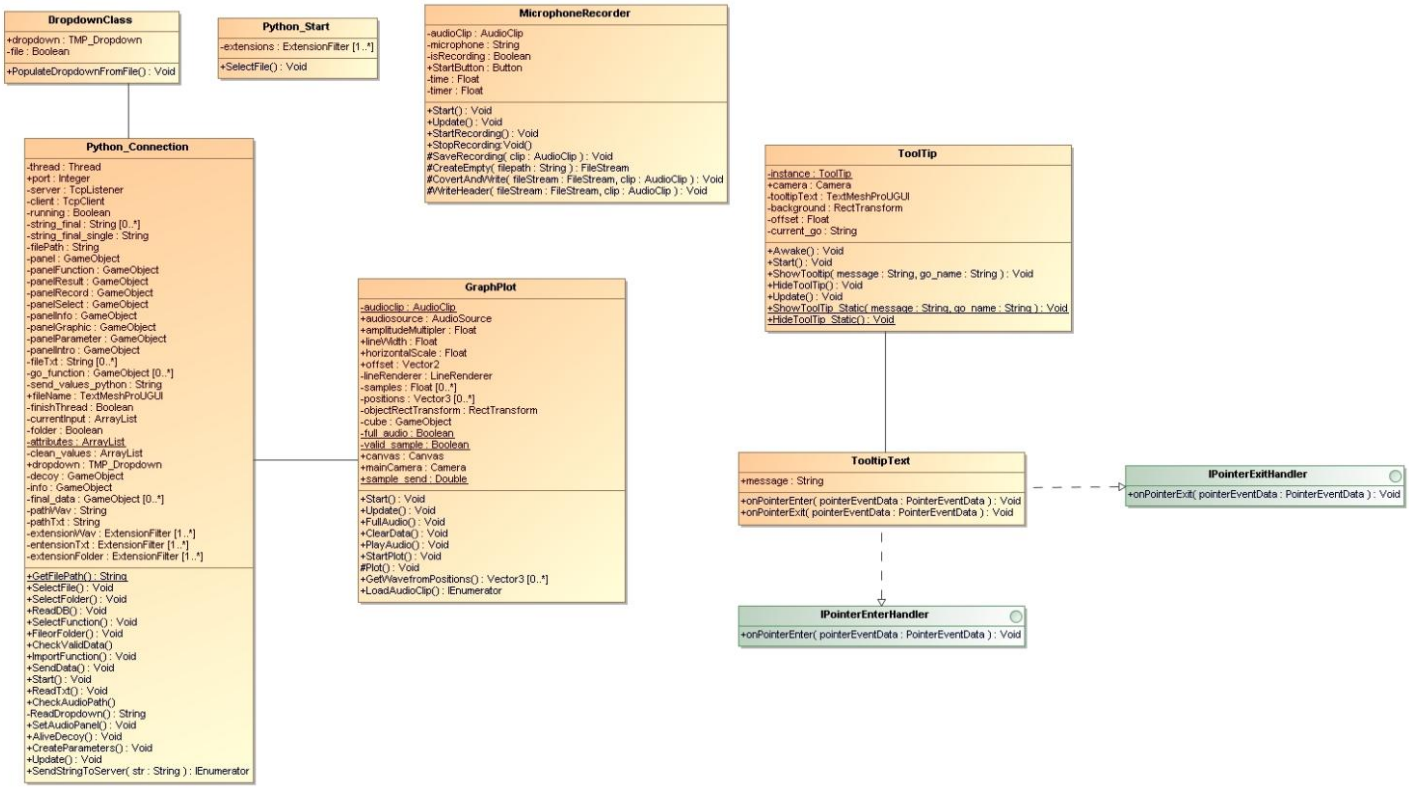

Figura 40: Arquitetura geral

4.2.2 Unity

No Unity é feita toda a componente que é apresentada ao utilizador. No entanto existe processamento de dados e ações do utilizador.

Começando pela primeira interação do utilizador na página inicial, o utilizador seleciona o ficheiro de Python que quer ligar ao servidor. Esta seleção é feita com o método *OpenFilePanel* da classe *StandaloneFileBrowser* que recebe um *array* de extensões para filtrar o tipo de dados que o utilizador pode selecionar, neste caso ficheiros com a extensão .py para o servidor. Com o ficheiro selecionado é iniciado o ficheiro de Python através da linha de comandos com o padrão *$"/k\"{nome_do_ficheiro}* \"". O parâmetro */k* [21] serve para lançar um processo e mantê-lo vivo.

De seguida é apresentado ao utilizador um menu com 3 opções, gravar áudio, importar função e chamada de função.

Começando pela opção gravar áudio, o utilizador necessita de colocar o número de segundos da gravação. Toda esta interação é simples e intuitiva porque o utilizador consegue perceber quando é que a gravação começou e quando esta acaba, caso o utilizador não queira terminar a gravação antes. Se o utilizador quer parar a gravação antes do tempo, tem a possibilidade de o fazer ao clicar no botão respetivo. Relativamente à gravação em si, é iniciado um *FileStream* onde irá ser escrito a informação capturada pelo microfone do computador em 2 canais diferentes, para ser reproduzido em colunas com o som do lado esquerdo e direito.

Outra opção disponível é o importe de uma função. Esta função é uma função presente no servidor Python e segue o padrão da figura 24. A função é escrita numa única linha e o importe é feito ao selecionar o ficheiro de texto onde se encontra esta função, semelhante ao início do servidor Python. É criada uma extensão específica para .txt para ajudar o utilizador a fazer o importe. Quando o importe é feito, a função escolhida é guardada numa base de dados local para utilização futura.

A última opção é chamar a função que é o fluxo principal da aplicação. Quando o utilizador escolhe esta opção, é-lhe apresentado uma lista de funções atualmente presentes na base de dados. A lista é atualizada dinamicamente o que significa que se o utilizador quiser voltar atrás e fazer um novo importe, quando entrar no menu de escolha de função a nova função irá estar presente para ser selecionada. A lista consiste num elemento *Dropdown* do Unity onde cada valor é lido dinamicamente da base de dados.

De seguida o utilizador irá escolher o tipo de dados na página de dados, único ficheiro ou uma pasta com ficheiros. Esta seleção é semelhante às anteriores do servidor e do importe com a exceção da seleção da pasta. Esta seleção usa a classe *StandaloneFileBrowser,* mas com o método *OpenFolderPanel* que em vez de receber um conjunto de extensões, recebe uma *string* com a localização inicial para a seleção da pasta. A partir desde ponto o fluxo da aplicação divide-se em 2 partes; o utilizador seleciona um único ficheiro e é lhe apresentado uma página para selecionar especificamente o que o utilizador pretende, se o início de uma *frame* ou o sinal por completo, que por sua vez irá passar para a página de atribuição de valores, ou o utilizador seleciona uma pasta de ficheiros e passa diretamente para a atribuição de valores.

Na seleção de um único ficheiro o utilizador consegue ver na página seguinte o gráfico do sinal escolhido. Este sinal é lido dinamicamente através de uma corrotina [22]. A corrotina retorna os valores do sinal de fala que irão ser convertidos para coordenadas do mundo Unity. O gráfico é desenhado através de um *LineRenderer*, após a conversão de coordenadas, para representar o gráfico. O utilizador tem duas opções nesta página; a seleção do início de uma *frame* ou a seleção do ficheiro por completo. A seleção da *frame* é feita ao clicar em cima do gráfico, selecionando assim o seu início. As coordenadas da aplicação são convertidas para

o mundo Unity para respeitar a posição que o utilizador quer escolher e assim é criado um cubo que marca a seleção. A seleção por completo é feita através de um botão e quando o utilizador clica no botão, o gráfico muda de cor e qualquer seleção feita antes é apagada. Um extra adicionado foi a reprodução do áudio para o utilizador conseguir ouvir o ficheiro de fala que escolheu.

Com os dados selecionados é apresentado ao utilizador a página de atributos onde o utilizador pode escolher os valores que desejar para cada variável da função escolhida. Cada campo tem uma caixa do texto onde o utilizador consegue perceber que atributo está a preencher e qual o seu tipo, bem como um ícone de informação ao lado de cada caixa com a documentação de cada atributo. Para a criação destes campos é utilizado um *decoy* para criar as caixas de texto para a atribuição de valores. É lida a função escolhida no *Dropdown* para ser feita uma pesquisa na base de dados local com o objetivo de ir buscar os dados da função. A posição destas caixas é dinâmica e varia consoante o número de atributos da função, de maneira que a informação esteja sempre presente e clara para o utilizador. O projeto suporta um máximo de 9 atributos para manter a claridade na página. Também é interpretado cada tipo de variável que a função tem para criar as caixas de texto correspondentes a cada tipo, isto é, uma variável do tipo *Integer* só aceita caracteres numéricos e uma variável do tipo *String* só aceita caracteres entre A e Z. Caso o utilizador decida voltar atrás e mudar a função escolhida ou o tipo de dados, estas caixas são dinamicamente ajustáveis e limpam os valores previamente selecionados pelo utilizador.

Por fim é feita a comunicação para o Python com os valores escolhidos e serão obtidos resultados consoante os dados do utilizador. De seguida é descrita a comunicação entre as duas plataformas.

4.2.3 Comunicação

A comunicação entre o Unity e o Python é feita através do protocolo HTTP com a utilização de *sockets*. Para este projeto esta comunicação é feita em *localhost*, ou seja, o IP do servidor será 127.0.0.1 e o porto utilizado para a comunicação é 25000.

Um *socket* é uma ligação ponto-a-ponto entre dois programas que estejam presentes na mesma rede. Visto que a ligação será de *localhost*, ambos os programas Unity e Python irão estar na mesma rede. Este *socket* utiliza o protocolo TCP/IP [23], *Transmission Control Protocol* e *Internet Protocol* respetivamente, onde os dados serão codificados para os seus caracteres ASCII para depois serem descodificados no servidor Python. Este protocolo é responsável pelo envio da mensagem e que cada ponto da comunicação receba a sua mensagem por ordem e síncrona. O protocolo começa por fazer um acordo com 3 passos

entre cada ponto. O primeiro passo é designado de *SYN*, que é o primeiro pedido para começar a ligação, de seguida é enviado uma resposta de reconhecimento, *SYN_ACK*, com o objetivo de reconhecer a ligação e formar a ligação entre os dois pontos. Por fim é enviado uma mensagem *ACK* onde esta já possui os dados que quer enviar.

Na figura 41 é ilustrado um exemplo de como a comunicação é realizada entre o Unity e o Python.

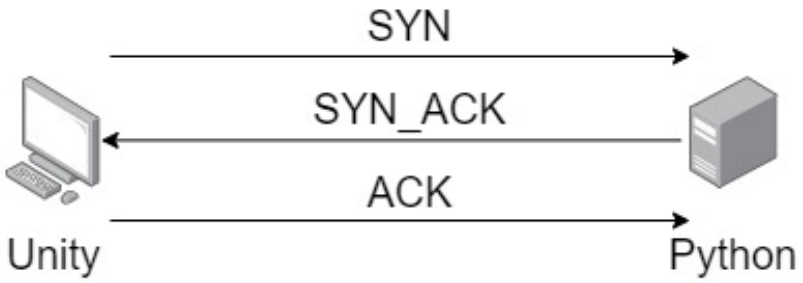

Figura 41: Protocolo TCP/IP

Os dados da mensagem possuem o seguinte formato; o primeiro atributo da mensagem corresponde ao nome da função, de seguida são os diversos parâmetros que a função possua, seguinte é enviado a diretoria quer do ficheiro de fala quer da pasta com diversos ficheiros, e por fim é enviado um atributo de controlo sobre o sinal para diferenciar se os dados a ser processados são por completo ou se existe uma trama para ser calculada.

Com a receção dos dados o servidor Python vai processar os dados e executar a função pedida pelo cliente Unity, onde no final o servidor vai enviar o resultado da função no mesmo *socket*, fechando o mesmo no final do envio. Esta mensagem é constituída por 4 valores, máximo, mínimo, média e desvio padrão.

O servidor fica outra vez disponível para a receção de outro *socket* para continuar a disponibilizar resultados ao utilizador.

A figura 42 mostra um esboço do protocolo e a comunicação realizada entre os dois pontos.

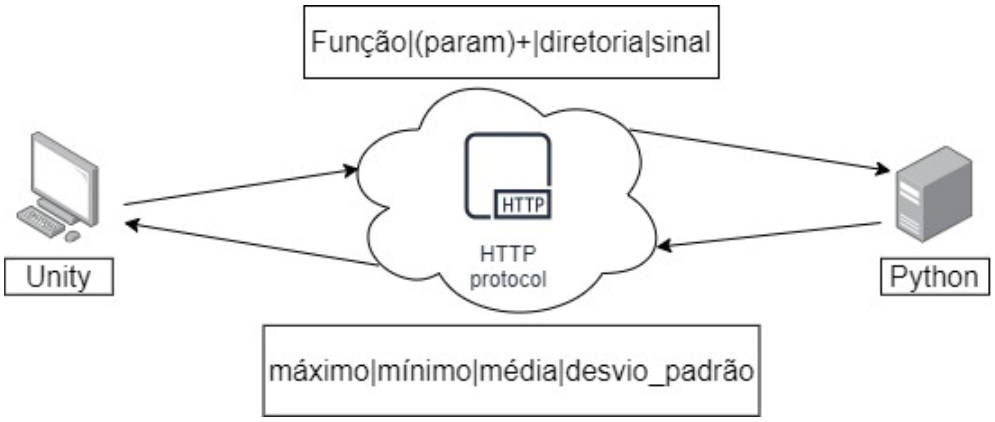

Figura 42: Protocolo e comunicação

4.2.4 Python

O algoritmo do servidor Python segue a seguinte lógica: a mensagem é recebida no servidor com o formato apresentado na Figura 42, sendo imediatamente repartida pelo carácter delimitador "|", definido no protocolo de comunicação. A primeira ação do algoritmo consiste em interpretar essa mensagem e identificar qual a função selecionada pelo utilizador na aplicação desenvolvida em Unity.

Consoante a função escolhida, o algoritmo prossegue para determinar o tipo de dados que o utilizador pretende processar:

- uma pasta com vários ficheiros de áudio,
- uma amostra completa de um sinal, ou
- o início de uma determinada trama.

Com base nestas informações, é chamada a função correspondente, que realiza o processamento necessário de acordo com os atributos fornecidos.

A mensagem de resposta enviada de volta para o Unity é construída com os dados produzidos pelo algoritmo, respeitando o mesmo protocolo de formatação definido para a comunicação. As Figuras 43, 44 e 45 ilustram exemplos das mensagens trocadas entre o Unity e o servidor Python para os três tipos de execução suportados: trama, amostra completa e pasta com múltiplas amostras, respetivamente.

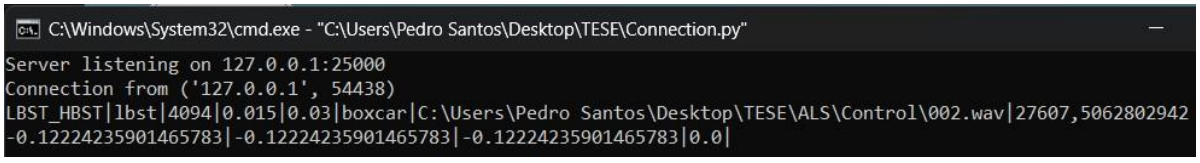

Figura 43: Protocolo para frame

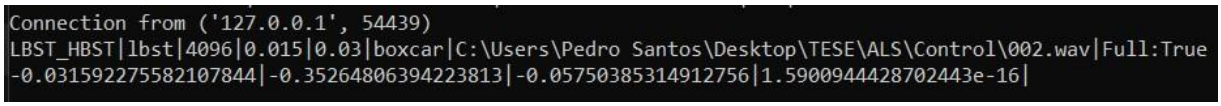

Figura 44: Protocolo para amostra

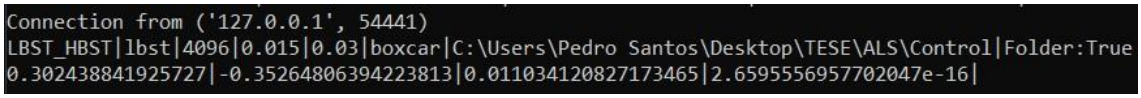

Figura 45: Protocolo para diversas amostras

A comunicação entre cliente e servidor baseia-se no protocolo TCP/IP, estabelecendo uma ligação direta entre o Unity e o servidor Python. A escolha deste protocolo deve-se à sua fiabilidade na entrega de mensagens, o que é essencial para aplicações interativas que exigem integridade nos dados trocados. A figura 46 ilustra a execução do servidor Python.

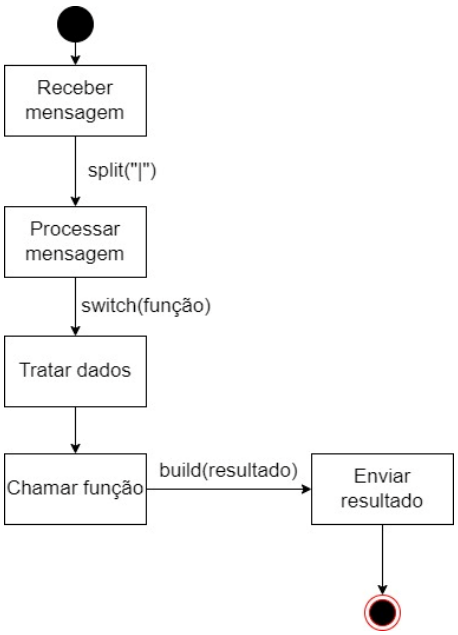

Figura 46: Algoritmo

Apesar de a implementação do servidor cumprir os objetivos propostos no contexto do protótipo, foram identificadas algumas limitações que poderão comprometer a sua utilização em cenários mais exigentes ou em ambientes de produção. Estas limitações encontram-se descritas de seguida:

- A implementação atual não inclui paralelização ou *threads*, o que pode limitar a escalabilidade caso se pretenda alargar a aplicação para mais utilizadores ou introduzir operações mais pesadas.
- Não existe ainda tratamento de segurança (ex. encriptação das mensagens ou autenticação entre cliente e servidor), o que limita a utilização da aplicação em redes públicas ou ambientes distribuídos.
- A gestão de erros, embora funcional, poderá ser melhorada com a definição de códigos de erro normalizados e com mensagens mais informativas para o utilizador.

4.3. UI

O UI apresenta uma cor azul, para representar um ambiente calmo, com um tom de cinzento, que representa inteligência e sabedoria. A conjugação das duas cores mostra um ambiente simples para o utilizador conseguir perceber que informação tem de escolher em cada página da aplicação. O código hexadecimal da cor é #6798AA.

O tipo de letra escolhido foi o *sumana regular*, que é um tipo de letra serifada, para ajudar a melhorar a leitura da informação da aplicação Unity.

O logo da aplicação é composto pelo nome e pelo ícone. O nome da aplicação é Oratio, inspirado no latim da palavra “fala”, e o ícone é composto por um altifalante a reproduzir som, para representar o objetivo da aplicação para a caracterização da voz.

Como descrito no subcapítulo Unity, cada página possui o ícone da aplicação no canto superior esquerdo e a informação principal está representada no meio da aplicação com o objetivo de ser intuitivo e simples para o utilizador navegar na aplicação.

5. Avaliação do UI e testes de usabilidade

Com o intuito de avaliar a eficácia da interface (UI) e da experiência do utilizador (UX) do protótipo desenvolvido, foi realizado um teste de usabilidade com um grupo de participantes. O principal objetivo consistiu em perceber se a navegação era intuitiva, se a disposição dos elementos facilitava o uso da aplicação e se as tarefas essenciais podiam ser concluídas sem dificuldades.

Foi convidado um grupo de 13 utilizadores para interagir com o protótipo. Cada participante foi instruído a realizar um conjunto de tarefas previamente definidas (como iniciar uma gravação, navegar entre ecrãs, aceder às definições, entre outras), enquanto verbalizava os seus pensamentos e dúvidas durante o teste.

Antes do início do teste, foi lida a seguinte introdução para contextualizar os participantes:

Olá, desde já agradeço a participação neste teste.

Vamos fazer um pequeno teste para perceber se a interface da aplicação está clara e fácil de usar. A aplicação que vai ver é um protótipo que serve para trabalhar com gravações e análise da fala.

O objetivo do teste é entender se o protótipo é intuitivo e claro e se existe alguma dificuldade a realizar alguma tarefa.

Enquanto usa a app, pode ir dizendo em voz alta o que está a pensar. Por exemplo, se alguma coisa for confusa, se não souber onde clicar ou se algo parecer estranho. Isso ajuda-me a melhorar.

Vou pedir-lhe que faça algumas tarefas simples, como encontrar opções no ecrã ou iniciar uma gravação.

Quando estiver pronto, podemos começar.

Após a realização das tarefas, foi solicitado aos participantes que avaliassem o protótipo com base em sete métricas principais, numa escala de 1 (muito fraco) a 5 (muito bom). Os resultados estão presentes na tabela 1.

Tabela 1: Resultado do teste

Métrica	Descrição	Valor
Clareza visual	Facilidade de entender os elementos visuais	4,4 / 5
Consistência visual	Manutenção de padrões visuais e estilos	4,4 / 5

Facilidade de navegação	Intuitividade na localização de menus e botões	4,1 / 5
Feedback visual	Qualidade das respostas visuais após ações do usuário	3.2 / 5
Legibilidade dos textos	Facilidade de leitura em diferentes dispositivos	2.7 / 5
Adaptação a diferentes telas	Avaliação da responsividade do layout	78% satisfeitos
Satisfação estética geral da interface	Avaliação visual e emocional geral da interface	3.9 / 5

De forma geral, o protótipo obteve resultados positivos, com destaque para a clareza e consistência visual, que foram as métricas mais valorizadas pelos utilizadores. A aplicação foi considerada intuitiva, e a maioria dos participantes conseguiu concluir todas as tarefas propostas sem dificuldades significativas.

As métricas com menor pontuação foram o **feedback visual** e a **legibilidade dos textos**. Foram identificadas situações em que os utilizadores não receberam confirmação clara após determinadas ações (como a localização onde os ficheiros de gravação são guardados), o que gerou alguma incerteza. No que diz respeito à legibilidade, alguns textos apresentavam tamanho ou contraste insuficientes.

Apesar destes pontos a melhorar, os utilizadores demonstraram compreender o propósito da aplicação e sentiram-se sempre contextualizados ao navegar pelas várias páginas do protótipo. Os resultados obtidos forneceram dados relevantes que irão orientar melhorias futuras.

6. Conclusão e trabalho futuro

O projeto apresenta um protótipo com o objetivo de ser modelável consoante as necessidades do utilizador, englobando uma interface Unity com o servidor Python. A implementação do projeto teve em mente os conceitos de simplicidade, dinamismo e modelação de modo a aplicação ser modelada consoante as necessidades do utilizador. A aplicação remove todas as complexidades do UI, tornando a aplicação fluída e intuitiva de modo a não sobrecarregar o utilizador com informação.

O projeto divide-se em quatro partes; modelação, onde foram desenhadas as classes do Python e do Unity, o desenvolvimento Unity, onde foram implementadas as classes desenhadas e foi realizado um estudo da palete de cores e o significado emocional de cada uma de modo a transmitir os conceitos pensados para a realização da aplicação, a comunicação entre o Unity e o Python, onde foi implementado um protocolo TCP/IP para garantir que não haja perdas na troca de mensagens, e o desenvolvimento Python, onde foi implementado um servidor local e os algoritmos para a análise de fala descritos ao longo do projeto.

Foram desenhadas todas as classes para a implementação da aplicação e cada classe representa o seu objetivo de forma clara e baixa complexidade, tornando assim a aplicação fluída e sem latência. Foram realizados ajustes ao longo do desenho e pensamento das classes consoante o *feedback* de funcionalidades que a aplicação deveria ter.

Foram implementados os métodos RPPC, LBST e HBST consoante as respetivas referências bibliográficas. Os métodos apresentam resultados semelhantes às referências bibliográficas.

A implementação do servidor foi feita em *localhost* para manter a simplicidade no desenvolvimento da aplicação. No entanto foi implementado o protocolo TCP/IP para garantir que cada mensagem seja entregue e não haja perda de informação na troca de mensagens. Isto garante que o utilizador tem sempre os resultados inseridos na interface.

Todas as páginas da aplicação estão intuitivas e claras de modo que o utilizador sabe sempre em que ponto está na aplicação. A aplicação tem toda a informação e o estado dos dados selecionados pelo utilizador. O utilizador consegue sempre voltar atrás em qualquer ponto da aplicação para trocar dados caso deseje. Os testes de usabilidade realizados permitiram validar que a aplicação apresenta uma base sólida no que respeita à experiência do utilizador. A estrutura, o aspeto visual e a clareza geral foram bem recebidos. As observações recolhidas apontam para pequenos ajustes a nível de feedback visual e legibilidade, que serão considerados em próximas iterações do design.

Foram implementadas as funcionalidades necessárias para mostrar o protótipo desenvolvido. Como tal existem outras funcionalidades que poderão ser adicionadas num trabalho futuro e algumas limitações.

Uma das funcionalidades é introduzir a escolha de caminhos de gravação do microfone. Esta funcionalidade torna a aplicação mais dinâmica e versátil para qualquer tipo de utilizador. Outra funcionalidade seria a introdução de servidores remotos. Esta funcionalidade aumenta a complexidade do projeto, mas aumenta a diversidade de funções que o utilizador pode aceder. Esta funcionalidade teria também de ter uma base de dados remota para o servidor conseguir aceder a todas as funções importadas por cada utilizador. A última futura funcionalidade seria a introdução de algoritmos de identificação de orador, ou seja, inteligência artificial que ao receber um sinal de fala conseguia identificar se o paciente correspondente ao sinal possui alguma patologia ou se era saudável.

Uma das principais limitações da aplicação desenvolvida está relacionada com o desenho do gráfico. Atualmente, a representação gráfica dos dados é bastante simples e não oferece funcionalidades interativas, como a possibilidade de fazer zoom ou deslocamento (pan). Esta limitação pode dificultar a análise detalhada dos dados, especialmente quando há um grande volume de informação representada no gráfico. Apesar desta limitação, a aplicação cumpre o seu objetivo principal de apresentar os dados de forma clara. No entanto, a introdução de funcionalidades interativas poderá representar um passo importante para otimizar a usabilidade e tornar a análise mais eficaz.

7.Bibliografia

[1] DisVoice. (2024). https://github.com/jcvasquezc/DisVoice (consultado em 09 de julho de 2024)

[2] Praat. (2024). https://github.com/praat/praat (consultado em 12 de julho de 2025)

[3] WaveSurfer. (2024). https://wavesurfer.xyz/ (consultado em 14 de julho de 2025)

[4] DiagoScope. (2024). https://diagnova.eu/pages/offer/voice_analysis.html (consultado em 18 de julho de 2025)

[5] DeVilliers, E. M. (1996). Implementing voice recognition and natural language processing in the NPSNET networked virtual environment. (pp. 31-39).

[6] Burke, D. (2007). *Speech Processing for IP Networks: Media Resource Control Protocol (MRCP).* Chichester, Inglaterra: John Wiley & Sons.

[7] Toledo, T. F. de. (2019). *Integrando Sistemas de Reconhecimento Automático de Fala em Aplicações Web*. Curitiba, Brasil: Appris Editora.

[8] Dutoit, T. (1997). *An Introduction to Text-to-Speech Synthesis*. Dordrecht, Países Baixos: Kluwer Academic Publishers.

[9] Johnston, A. B. (2015). *SIP: Understanding the Session Initiation Protocol* (4ª ed.). Norwood, Massachusetts, EUA: Artech House.

[10] Roy, R. R. (2016). *Handbook on Session Initiation Protocol: Networked Multimedia Communications for IP Telephony*. Boca Raton, EUA: CRC Press.

[11] Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (1996). *RTP: A Transport Protocol for Real-Time Applications* (RFC 1889). Fremont, Califórnia, EUA: Internet Engineering Task Force (IETF).

[12] Cloudfare. (2024). https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/

[13] Bouck, E. C. (2016). *Assistive Technology*. Thousand Oaks, Califórnia, EUA: SAGE Publications.

[14] File, P., & Elder, L. (1997). *Using NLP in the Design of a Conversation Aid for Non-Speaking Children*. Dundee, Escócia: University of Abertay Dundee.

[15] Cordeiro H., Fonseca. J. and Meneses C. (2014). Spectral Envelope and Periodic Component in Classification Trees. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, (pp. 4607-4610).

[16] H. T. Cordeiro, C. Meneses. (2020). Voice spectrum energy band and tilt analysis for Bulbar ALS. *Procedia Computer Science*, (pp. 207-214).

[17] M. Vashkevich, A. Petrovsky, Y. Rushkevich, Bulbar ALS Detection Based on Analysis of Voice Perturbation and Vibrato, in: Signal Process. - Algorithms, Archit. Arrange. Appl. Conf. Proceedings, SPA, Division of Signal Processing and Electronic Systems, Poznan University of Technology (DSPES PUT), 2019: pp. 267–272. doi:10.23919/SPA.2019.8936657.

[18] (1994). *Kay Elemetrics, Elemetrics Disordered Voice Database (Version 1.03).*

[19] IBM. (2024). https://developer.ibm.com/articles/an-introduction-to-uml/ (consultado em 29 de junho de 2024)

[20] Cambridge. (2024). https://dictionary.cambridge.org/dictionary/english/tooltip (consultado em 06 de maio de 2024)

[21] CMD. (2024). https://ss64.com/nt/cmd.html (consultado em 23 de março de 2024)

[22] Unity. (2024). https://docs.unity3d.com/Manual/Coroutines.html (consultado em 04 de janeiro de 2024)

[23] Cloudfare. (2024). https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/ (consultado em 12 de fevereiro de 2024)