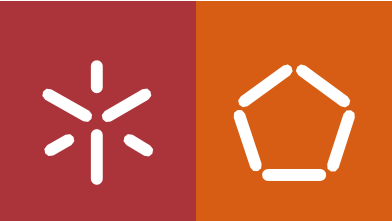




Tradução de Linguagem Gestual em Realidade
Mista

Manuel Ribeiro

UMinho | 2023



Universidade do Minho
Escola de Engenharia

Manuel Rebelo de Melo Ribeiro

**Tradução de Linguagem Gestual em
Realidade Mista**

Outubro de 2023



Universidade do Minho
Escola de Engenharia

Manuel Rebelo de Melo Ribeiro

Tradução de Linguagem Gestual em Realidade Mista

Relatório de Trabalho de Projeto de Mestrado
Mestrado Integrado em Engenharia e Gestão de Sistemas de
Informação

Trabalho efetuado sob a orientação do
Professor Doutor Luís Gonzaga Mendes de Magalhães

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



Atribuição-NãoComercial-Compartilhalgual

CC BY-NC-SA

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

AGRADECIMENTOS

A realização desta tese de mestrado não teria sido possível sem o apoio de várias pessoas.

Gostava de agradecer em primeiro lugar ao meu orientador, Professor Luís Gonzaga, pelo apoio valioso e pela orientação ao longo de todo o processo. O seu feedback, sugestões e disponibilidade foram fundamentais para o desenvolvimento do trabalho.

Gostava de agradecer ao Centro de Computação Gráfica, pela disponibilização da rede neuronal que permitiu criar a aplicação desenvolvimento neste trabalho.

Gostava de agradecer também ao João Oliveira, que me ajudou com o uso da rede neuronal fornecida e com a realização da ligação ao servidor, e ao José Rocha, que me ajudou com o funcionamento da aplicação no dispositivo *Hololens 2*. O auxílio prestado pelos dois foi essencial e facilitou em muito o desenvolvimento do projeto.

Por fim, mas não menos importante, gostava de agradecer à minha família pelo apoio e motivação ao longo desta caminhada, que sem dúvida foi essencial para que eu pudesse concluir este trabalho.

Agradeço ainda aos meus amigos, que sempre estiveram presentes para me apoiar e me motivar.

A todos, o meu sincero agradecimento.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio, nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Tradução de Linguagem Gestual em Realidade Mista

A necessidade de criar sistemas capazes de auxiliar indivíduos com dificuldades auditivas é cada vez mais premente, uma vez que estes permitem a inserção destes mesmos indivíduos na sociedade. Infelizmente, indivíduos que não são capazes de comunicar através de métodos convencionais, mais concretamente, através de línguas faladas, apresentam dificuldades acrescidas em vários setores da sociedade, nomeadamente no ensino e no mercado de trabalho, dificuldades estas que prejudicam gravemente indivíduos da comunidade surda. De maneira a combater estas dificuldades, propõe-se com este trabalho criar uma solução capaz de permitir realizar uma conversa entre um indivíduo sem dificuldades auditivas e um indivíduo com dificuldades auditivas que utilize uma Língua Gestual como principal meio de comunicação.

De maneira a permitir a criação desta solução foi primeiro analisado o conceito de Língua Gestual e foram avaliadas algumas soluções existentes criadas com o propósito de auxiliar indivíduos com dificuldades auditivas a melhor comunicarem com indivíduos sem dificuldades auditivas.

Com uma melhor compreensão das necessidades que indivíduos com dificuldades auditivas apresentam e com conhecimento das necessidades satisfeitas pelas soluções atualmente disponibilizadas, foi idealizada uma aplicação capaz de responder a algumas destas necessidades ainda não adereçadas.

Com a idealização da aplicação realizada, em seguida foi documentado o processo de desenvolvimento da aplicação. Este processo envolve a enumeração e instalação das diversas ferramentas utilizadas, o desenvolvimento inicial da aplicação, onde o processo de criação das interfaces iniciais é descrito, os problemas identificados durante o desenvolvimento inicial da aplicação tal como as alternativas adotadas para resolver esses mesmos problemas e por fim o desenvolvimento final da aplicação, já com as alterações adotadas implementadas.

O desenvolvimento deste trabalho levou à criação de uma nova solução capaz de auxiliar indivíduos que utilizam língua gestual como principal método de comunicação e à avaliação de algumas soluções atualmente utilizadas para este mesmo efeito.

Por fim, é realizada uma breve conclusão e possível trabalho futuro é apresentado.

Palavras-chave: Língua Gestual; LGP; Realidade Mista.

ABSTRACT

Sign Language Translation using Mixed Reality

The need to create systems capable of providing aid to individuals with hearing difficulties is increasingly pressing, as these systems allow the inclusion of these same individuals in our society. Unfortunately, individuals who are not able to communicate through conventional methods, more specifically, through spoken languages, face increased difficulties in various sectors of society, namely education and the job market, difficulties that seriously harm individuals from the deaf community. To combat these difficulties, this work proposes to create a solution capable of allowing a conversation to be carried out between an individual without hearing difficulties and an individual with hearing difficulties who uses Sign Language as the main means of communication.

To allow the creation of this solution, the concept of Sign Language was first analyzed and some existing solutions created with the purpose of helping individuals with hearing difficulties to better communicate with individuals without hearing difficulties were evaluated.

With a better understanding of the needs that individuals with hearing difficulties face and with knowledge of the needs satisfied by the solutions currently available, an application capable of responding to some of these needs that have not yet been addressed was idealized.

With the application idealized, the application development process was then documented. This process involves the enumeration and installation of the various tools used, the initial development of the application, where the process of creating the initial interfaces is described, the problems identified during the initial development of the application as well as the alternatives adopted to solve these same problems and finally the final development of the application, with the adopted changes already implemented.

The development of this work led to the creation of a new solution capable of helping individuals who use sign language as their main method of communication and the evaluation of some solutions currently used for this same purpose.

Finally, a brief conclusion is made and possible future work is presented.

Keywords: Sign Language; *PSL*; Mixed Reality.

ÍNDICE

Direitos de Autor e Condições de Utilização do Trabalho por Terceiros.....	ii
Agradecimentos.....	iii
Declaração de Integridade	iv
Resumo.....	v
Abstract.....	vi
Lista de Abreviaturas e Siglas	xiii
Lista de Figuras.....	xiv
Lista de Tabelas	xvi
1. Introdução.....	1
1.1 Enquadramento e Motivação.....	1
1.2 Objetivos e Resultados Esperados	2
1.3 Organização do Documento	2
2. Revisão do Estado da Arte.....	4
2.1 Língua Gestual.....	4
2.1.1 O que é?	4
2.1.2 Situação Atual	5
2.1.2.1 No Mundo.....	5
2.1.2.2 Em Portugal	7
2.1.3 Tecnologias de Auxílio	7
2.1.3.1 Smart Glasses.....	8
2.1.3.2 Smart Gloves.....	8
2.1.3.3 Digital Avatars	9
2.2 Tradução da Língua Gestual.....	9
2.2.1 Datasets.....	10

2.2.2	Soluções para LGP	11
2.2.3	Soluções analisadas	12
2.2.3.1	BrightSign Glove	13
2.2.3.2	SignAll Chat	14
2.2.3.3	Hand Talk	15
2.2.3.4	Holographic Sign Language Interpreters.....	15
2.2.4	Análise inicial das soluções.....	16
2.2.5	Análise detalhada das soluções	17
3.	Especificação da Solução	27
3.1	Contexto.....	27
3.2	Requisitos.....	28
3.2.1	Requisitos funcionais.....	28
3.2.2	Requisitos não funcionais	29
3.3	Conceptualização Inicial da Aplicação	30
4.	Desenvolvimento.....	34
4.1	Instalação e Uso das Ferramentas	34
4.1.1	<i>Unity</i> (2020.3.30f1)	35
4.1.2	<i>Visual Studio 2019</i> (16.11.30) e <i>Visual Studio 2022</i> (17.5.5)	35
4.1.3	<i>MRTK</i> (2.8.3)	36
4.1.4	<i>ML-Agents</i> (1.0.8)/ <i>Barracuda</i> (2.1.3).....	37
4.1.5	<i>MediaPipeUnityPlugin</i> (0.11.0)	38
4.1.6	<i>Microsoft Azure Cognitive Services (Speech Service)</i>	38
4.1.7	<i>PyCharm</i> (2022.3.1)	39
4.1.8	<i>Postman</i>	39
4.1.9	<i>HoloLens 2 Emulator</i> (10.0.20348.1535).....	40

4.1.10	<i>Connect</i>	41
4.1.11	<i>Windows Device Portal</i>	41
4.1.12	Dependencies (1.11).....	41
4.2	Aplicação Inicialmente Desenvolvida	42
4.2.1	<i>Objetos MRTK</i>	43
4.2.2	Interface Principal (Inicial)	45
4.2.3	Interface ICDA (Inicial).....	47
4.2.4	Interface ISDA (Inicial)	51
4.3	Complicações Iniciais e Alternativas Adotadas	54
4.3.1	Rede Neuronal - Complicação.....	54
4.3.2	Rede Neuronal - Alternativa.....	60
4.3.3	<i>MediaPipeUnityPlugin</i> – Complicação	60
4.3.4	<i>MediaPipeUnityPlugin</i> - Alternativa	61
4.3.5	<i>MRTK</i> – Complicação.....	61
4.3.6	<i>MRTK</i> – Alternativa.....	62
4.3.7	Emulador <i>Hololens</i> – Complicação	62
4.3.8	Emulador <i>Hololens</i> – Alternativa	63
4.4	Complicações Agravadas e Alternativas Adotadas.....	63
4.4.1	<i>MediaPipeUnityPlugin</i> – Complicação	63
4.4.2	<i>MediaPipeUnityPlugin</i> – Alternativa.....	65
4.4.3	Rede Neuronal – Complicação.....	66
4.4.4	Rede Neuronal – Alternativa	67
4.4.5	<i>Build</i> da aplicação – Complicação	68
4.4.6	<i>Build</i> da aplicação – Alternativa	68
4.4.7	<i>Hololens 2</i> – Complicação.....	70

4.4.8	<i>Hololens 2 - Alternativa</i>	70
4.5	Conclusão do desenvolvimento da Aplicação	71
4.5.1	Estrutura da cena final	71
4.5.2	Interface Partilhada (Final).....	72
4.5.3	Elementos ICDA (Final).....	73
4.5.4	Elementos ISDA (Final).....	74
4.6	<i>Build e Deployment em Hololens 2</i>	75
4.6.1	<i>Unity Editor (Build)</i>	75
4.6.2	<i>Visual Studio 2022 (Deployment)</i>	77
4.6.3	Aplicação no <i>Hololens 2</i>	78
5.	Solução Final.....	79
5.1	Fluxograma da solução	79
5.2	Componentes finais	79
5.2.1	Interface Partilhada	80
5.2.2	ICDA	80
5.2.3	ISDA	81
5.2.4	Menu de Ajuda	82
5.2.5	Menu de Saída	83
6.	Avaliação da Solução	84
6.1	Teste da Solução	84
6.1.1	Ligação de <i>Internet</i> Fraca	84
6.1.2	Luminosidade Elevada.....	85
6.1.3	Luminosidade Reduzida.....	85
6.1.4	Imagem com Duas Pessoas	86
6.1.5	<i>Speech-to-Text</i> com Duas Pessoas	86

6.1.6	Condições ideais	87
6.2	Comparação com Soluções já Existentes.....	88
7.	Conclusão e Trabalho Futuro	91
7.1	Conclusão	91
7.2	Trabalho futuro	92
	Referências	95

LISTA DE ABREVIATURAS E SIGLAS

<i>ASL</i>	<i>American Sign Language</i>
<i>ASR</i>	<i>Automatic speech recognition</i>
CCG	Centro de Computação Gráfica
CTILG	Serviços de Tradução e Interpretação de Língua Gestual
<i>GUI</i>	<i>Graphical User Interface</i>
ICDA	Indivíduo com dificuldades auditivas
<i>IDE</i>	<i>Integrated development environment</i>
ISDA	Indivíduo sem dificuldades auditivas
LG	Língua Gestual
LGP	Língua Gestual Portuguesa
Libras	Língua Brasileira de Sinais
<i>LTS</i>	<i>Long-Term Support</i>
<i>MRTK</i>	<i>Mixed Reality Toolkit</i>
<i>ONNX</i>	<i>Open Neural Network Exchange</i>
<i>SEE</i>	<i>Signed Exact English</i>
<i>VS</i>	<i>Visual Studio</i>

LISTA DE FIGURAS

Figura 1: BrightSign Glove (BrightSign, 2022)	13
Figura 2: SignAll Chat (SignAll, 2021)	14
Figura 3: HandTalk (Hand Talk App, 2023)	15
Figura 4: Holographic Sign Language Interpreters (ACM SIGGRAPH Blog, 2022).....	16
Figura 5: Hand Landmark Model (Mediapipe, 2020)	28
Figura 6: Esboço da interface idealizada	30
Figura 7: Esboço da “interface inicial”	33
Figura 8: Interação entre utilizadores	33
Figura 9: Módulos <i>Unity</i> essenciais	35
Figura 10: Configurador <i>MRTK</i>	37
Figura 11: Requisitos <i>Hyper-V</i>	40
Figura 12: Script para instalar <i>Hyper-V</i>	40
Figura 13: Diagrama inicial da solução.....	42
Figura 14: Estrutura fundamental das cenas criadas	44
Figura 15: Estrutura da Interface Inicial.....	46
Figura 16: Função “Speech” Interface Inicial	46
Figura 17: Interface Inicial criada	47
Figura 18: Estrutura da Interface ICDA.....	48
Figura 19: Componente “Dictation Handler”	49
Figura 20: Interface ICDA (inicial).....	51
Figura 21: Estrutura da Interface ISDA	52
Figura 22: Interface ISDA (inicial)	53
Figura 23: Criação do ficheiro <i>ONNX</i>	55
Figura 24: Ficheiro <i>ONNX</i> criado.....	56
Figura 25: Erro <i>multiple precompiled assemblies</i>	56
Figura 26: Versão do Google.Protobuf.dll.....	57
Figura 27: Ficheiro <i>ONNX</i> no <i>Unity</i>	58
Figura 28: Resultado do teste do ficheiro <i>ONNX</i>	59
Figura 29: Importação do ficheiro <i>ONNX</i> teste.....	59

Figura 30: Erros gerados pela importação do ficheiro <i>ONNX</i>	59
Figura 31: Erro “ArgumentOutOfRangeException”	64
Figura 32: Erro InvalidOperationException	64
Figura 33: Problema com o servidor	66
Figura 34: Erro de validação <i>Unity</i>	68
Figura 35: Erro <i>OpenXR Features</i>	69
Figura 36: Erro <i>OpenXR Features</i> supostamente resolvido.....	69
Figura 37: Secção <i>OpenXR</i> final.....	69
Figura 38: Algumas das dependências do <i>package</i>	70
Figura 39: Estrutura da Interface Final	72
Figura 40: <i>Build Settings</i> da aplicação	76
Figura 41: Conectar PC ao dispositivo <i>Hololens</i>	77
Figura 42: Diagrama final da solução.....	79
Figura 43: Menu de definições ICDA	81
Figura 44: Elementos <i>Mediapipe</i>	81
Figura 45: Funcionamento da aplicação.....	82
Figura 46: Menu de ajuda.....	82
Figura 47: Menu de saída	83
Figura 48: Erro de conexão.....	85
Figura 49: Texto quando o segundo utilizador está longe.....	87
Figura 50: Texto quando o segundo utilizador está perto	87

LISTA DE TABELAS

Tabela 1: Comparação de <i>datasets</i> (<i>alterada</i>) (Facundo Quiroga, 2022)	11
Tabela 2: Análise inicial das soluções.....	17
Tabela 3: Análise detalhada das soluções	19
Tabela 4: Comparação das quatro soluções	24
Tabela 5: Requisitos funcionais	28
Tabela 6: Requisitos não funcionais	29
Tabela 7: Análise detalhada da solução criada	88
Tabela 8: Comparação das cinco soluções.....	89

1. INTRODUÇÃO

Na primeira secção deste capítulo é realizado um enquadramento inicial e são apresentadas as motivações para a criação deste trabalho. Na segunda secção deste capítulo, os objetivos e resultados esperados para o trabalho são expostos. Por fim, na última secção é apresentada a estrutura do documento tal como uma pequena descrição de cada capítulo criado.

1.1 Enquadramento e Motivação

A criação de sistemas inclusivos tem vindo a ser uma preocupação cada vez mais presente. Quando falamos em sistemas em que a comunicação verbal é crucial, essa necessidade torna-se ainda mais premente. Com a evolução e constante criação de projetos e iniciativas focadas em integração social e o apoio a essa mesma integração (Lozano Diaz et al., 2018), é importante garantir que todos os elementos da sociedade dispõem, no mínimo, de alguma capacidade de comunicação universal ou algum tipo de comunicação que possa ser compreendida pela maioria.

Infelizmente, alguns indivíduos têm dificuldade em comunicar através de métodos convencionais, o que cria problemas na interação com outros indivíduos, problemas estes que influenciam áreas como o trabalho e respetivo ambiente (Haynes & Linden, 2012a) e a educação (Hirabayashi et al., 2019; Luo et al., 2012; Shao et al., 2020). De maneira a atenuar estas dificuldades, podem ser usadas ferramentas com o intuito de facilitar estas interações e de as tornar mais orgânicas.

Felizmente, recentemente têm surgido diversas soluções que permitem a tradução automática de texto para língua gestual (Yang et al., 2022) e algumas que fazem o inverso, ou seja traduzem língua gestual em texto ou voz (ACM & 2018, 2018). Este último problema torna-se mais complexo porque implica o reconhecimento de sinais realizados pelo utilizador, embora existam algumas soluções que tentam resolver este problema.

A inclusão social é um direito humano fundamental, sendo apenas através desta inclusão que indivíduos com dificuldades acrescidas realmente possuem os seus direitos sociais e culturais definidos na Declaração Universal de Direitos Humanos (Nations, 1948). É essencial promover a inclusão social, uma vez que sem esta é impossível criar uma sociedade realmente justa e igual para todos os seus membros. Para tornar realidade um mundo onde indivíduos com dificuldades acrescidas são realmente incluídos na sociedade é necessário continuar a desenvolver e disponibilizar diferentes ferramentas que permitam

a estes mesmos indivíduos com dificuldades acrescidas interagir com a sociedade da mesma maneira que a maioria dos membros da sociedade interage com esta.

1.2 Objetivos e Resultados Esperados

Esta dissertação tem como objetivo principal o desenvolvimento de um sistema de Realidade Mista (*MR*) capaz de traduzir língua gestual em texto, facilitando assim a comunicação entre indivíduos não capazes de comunicar verbalmente e a restante população. Com este trabalho, pretende-se também analisar as soluções existentes relativas ao reconhecimento de língua gestual, avaliar a sua utilidade e relevância para o contexto do trabalho e criar uma nova solução capaz de responder a algumas das necessidades que atualmente não são adereçadas. O sistema deverá fornecer, pelo menos, as seguintes funcionalidades:

- Reconhecimento dos sinais realizado pelo utilizador em língua gestual;
- Apresentação da respetiva legenda no *viewport* holográfico do dispositivo de *MR*;
- Aplicação de *MR*, de maneira a poder ser utilizado num dispositivo *Hololens*.

1.3 Organização do Documento

Este documento é composto por sete capítulos diferentes, sendo estes: 1. Introdução, 2. Revisão do Estado de Arte, 3. Especificação da Solução, 4. Desenvolvimento da Aplicação, 5. Aplicação final, 6. Avaliação da Solução e 7. Conclusão e Trabalho Futuro.

No primeiro capítulo, Introdução, ao qual esta secção faz parte, é realizado o enquadramento do projeto e são apresentados os objetivos que se pretendem cumprir com este projeto. De seguida, a estrutura do documento é apresentada.

No segundo capítulo, Revisão do estado de arte, este começa por apresentar o conceito de Língua Gestual e como estas línguas são abordadas à volta do mundo. Em seguida, a forma como ferramentas ou aplicações podem ser utilizadas para auxiliar indivíduos que utilizam estas LG são apresentadas. Com a análise inicial destas aplicações, foram analisadas e comparadas quatro soluções atualmente existentes que pretendem facilitar a comunicação entre indivíduos que utilizam LG e indivíduos que não utilizam LG.

No terceiro capítulo, Especificação da Solução, é definido, antes do processo de desenvolvimento, o sistema que deve ser desenvolvido no âmbito deste projeto, especificando aspetos importantes como as

ferramentas que devem ser utilizadas, o que deve ser capaz de fazer e por fim quais são os requisitos que esta aplicação deve apresentar.

No quarto capítulo, Desenvolvimento, é primeiro descrito a aplicação inicialmente idealizada e as diferentes ferramentas que deveriam ser utilizadas para criar as diferentes funcionalidades. Em seguida, é descrito o processo de desenvolvimento da aplicação criada, dando especial atenção às ferramentas que foram utilizadas para poder criar as funcionalidades referidas, às interfaces e funcionalidades criadas e que compõem a aplicação, aos diversos problemas detetados durante o desenvolvimento da aplicação. No quinto capítulo, Aplicação Final, é apresentada a aplicação criada juntamente com uma breve descrição de cada um dos elementos que a compõem.

No sexto capítulo, Avaliação da Solução, foram realizados cinco testes diferentes que pretendem testar o comportamento da aplicação caso esta seja utilizada em condições não ideais e foi realizada uma comparação entre a solução desenvolvida e as quatro soluções previamente analisadas, de maneira a averiguar os pontos positivos e negativos que esta nova aplicação apresenta comparativamente às soluções existentes.

Por fim, no sétimo capítulo, Conclusão e Trabalho Futuro, é apresentada a conclusão tomada com o desenvolvimento deste projeto e são apresentadas alternativas para o desenvolvimento futuro da aplicação criada.

2. REVISÃO DO ESTADO DA ARTE

Neste segundo capítulo é realizada uma revisão do estado da arte, sendo esta focada nas Línguas Gestuais e nas suas variadas características. Durante este capítulo, aspetos relacionados com a sociedade e os diferentes métodos adotados por esta para auxiliar indivíduos com dificuldades auditivas são abordados, nomeadamente o *SEE*, ou *Signed Exact English*, e diferentes tecnologias e ferramentas utilizadas na criação de soluções capazes de auxiliar estes mesmos indivíduos.

2.1 Língua Gestual

2.1.1 O que é?

A língua Gestual é uma língua que provém das comunidades de pessoas surdas ou com dificuldades auditivas e utiliza as mãos (classificadas através da forma, postura, localização e movimento destas), a cabeça, a postura corporal, as expressões faciais, o olhar e os padrões dos lábios (Von Agris et al., 2008). Esta língua apresenta todas as características fundamentais de uma língua natural (Yin et al., 2021), ou seja, esta é uma língua desenvolvida naturalmente pelo ser humano, de forma não premeditada, apresentando a sua própria gramática e léxico. Esta língua não é universal nem mutualmente inteligível, embora haja algumas semelhanças entre as diversas línguas gestuais.

Antigamente, acreditava-se que uma língua gestual não era nada mais do que gestos, gestos estes que podiam ser reconhecidos universalmente (Lucas, 2001). Atualmente, apesar de algumas línguas gestuais serem reconhecidas como línguas legítimas e de serem adotadas por vários países como uma das suas línguas oficiais, ainda existe falta de legislação significativa capaz de, por exemplo, garantir o acesso a serviços sociais numerosos e de qualidade por parte de indivíduos com dificuldades auditivas (WFD, 2010).

Esta língua, criada inicialmente pela comunidade surda, também é utilizada por pessoas que não conseguem falar fisicamente, pessoas com dificuldades acrescidas ou familiares de pessoas que utilizam esta língua.

O número das línguas gestuais existentes é desconhecido, devido sobretudo ao facto de depender da comunidade através da qual foi criada, mas considera-se que cada país (geralmente) apresenta a sua própria língua gestual nativa. A revista “Ethnologue”, publicada em 2022, reconhece a existência de 154

línguas gestuais (Ethnologue, 2022), sendo que a plataforma *SIGN-HUB*, criada e desenvolvida pelo projeto *SIGN-HUB* (2016-2020) e financiada pela iniciativa *Horizon 2020*, reconhece a existência de 214 línguas gestuais, dispostas num atlas de acordo com a posição geográfica de cada uma (SIGN-HUB, 2020).

É importante referir que a língua gestual não é o mesmo que linguagem corporal, sendo esta última uma forma de comunicação não verbal, embora a língua gestual apresente tanto sinais como gestos (Goldin-Meadow & Brentari, 2017), sendo estes gestos o movimento (manual) que tanto pessoas que usam línguas faladas como pessoas que usam línguas gestuais realizam quando comunicam.

Os sinais manuais em língua gestual são gerados e interpretados através do uso de três “blocos de construção”, sendo estes o formato da mão (*handshape*), o movimento (*motion*) e o local da articulação (*place of articulation*), sendo a combinação destes três que determina o significado do sinal manual (Ding & Martinez, 2009). Para além das mãos, a postura corporal e expressão facial também fazem parte de cada sinal realizado.

2.1.2 Situação Atual

De maneira a melhor compreender as necessidades vividas por pessoas com dificuldades auditivas que utilizam uma língua gestual como principal meio de comunicação é importante compreender melhor a situação atualmente vivida por estes indivíduos e as diferentes alternativas que a sociedade tem vindo a utilizar ou pretende utilizar para melhorar a qualidade de vida destas pessoas e promover a integração destas na sociedade. De maneira a melhor expor estas questões, é importante diferenciar entre a situação vivida atualmente no mundo e a situação vivida atualmente em Portugal, uma vez que as línguas gestuais e as comunidades que as criam deparam-se, como muitas outras, com desafios bastante diferentes consoante o país ou cultura com que interagem.

2.1.2.1 No Mundo

A língua gestual, e a sua legitimidade, são cada vez mais reconhecidas, embora a sua importância nem sempre seja equiparada ao estatuto legal que as línguas faladas desse país apresentam. Este reconhecimento é relativamente recente, sendo a Suécia, em 1981, o primeiro país a apresentar legislação relativamente a língua gestual. Desde então, mais 76 países apresentam algum nível de reconhecimento pelas suas Línguas Gestuais, alguns tão recente quanto 2022, sendo esta informação

gerida pela *World Federation of the Deaf* (WFD, n.d.), uma instituição global que luta para garantir os direitos de igualdade das mais de 70 milhões de pessoas surdas em todo o mundo.

Esforços para garantir que pessoas surdas ou com dificuldades auditivas são capazes de se inserir na sociedade é uma preocupação atual, tal como a inserção de outras minorias, mas infelizmente este processo tem vindo a demorar mais do que o previsto, sendo uma das questões mais prementes não o custo, mas sim a falta de tecnologia capaz de facilitar esta mesma inserção (Diaz et al., 2018). Este esforço tem resultado na criação de várias soluções capazes de facilitar a comunicação entre pessoas que não são capazes de comunicar verbalmente e pessoas que são, focando-se normalmente na tradução de texto corrido (ou voz) para língua gestual, embora a situação inversa, ou seja, tradução da língua gestual para texto (ou voz), também já exista, embora menos recorrente.

Nos dias de hoje, pessoas surdas apresentam dificuldades acrescidas no setor da educação, podendo esta dificuldade acrescida ser relacionada com o facto da leitura ser uma atividade mais difícil para quem aprendeu uma língua gestual como primeira língua (CITY- & 1999, 2019). A insistência no uso da leitura como método de ensino na educação e não o uso da língua gestual também pode ser visto como uma forma de Audismo, o que pode ser definido como “...*Tom Humphries (1975) ... Audism: (O ^ dizm) n. The notion that one is superior based on one's ability to hear or behave in the manner of one who hears.*” (Bauman, 2004), algo que é realçado por um estudo realizado em Zimbabwe com o intuito de repensar a forma como o ensino de crianças surdas acontece, discutindo algumas oportunidades que podem ser aproveitadas para melhorar este mesmo ensino (Musengi et al., 2013).

Dificuldades acrescidas no trabalho também é outro fator que influencia negativamente as pessoas surdas ou com dificuldades auditivas e a sua inserção na sociedade, normalmente relacionadas com a dificuldade de comunicação entre o indivíduo e o grupo de trabalho ou entre o indivíduo e a gerência.

De acordo com um questionário realizado (Haynes & Linden, 2012b), acomodações normalmente existentes no trabalho para auxiliar estas pessoas são assistências telefónicas (*telephone aids*), assistência dos colegas de trabalho (*co-worker helps*) e comunicação eletrónica (*eletronic communication*), mas as taxas de satisfação registadas rondam apenas os 50%, sendo que os problemas principais registados foram a falta de comunicação eficaz em grupo e a falta de apoio dos colegas de trabalho.

Outro aspeto relevante quando falamos do contexto do trabalho é a influência que o uso de um interpretador de língua gestual tem na equipa, uma vez que através deste não é possível conhecer verdadeiramente a pessoa com dificuldades auditivas com quem estamos a comunicar e, igualmente relevante, é a constante recordação que o ambiente do trabalho foi alterado para alguém e não para todos, reforçando a ideia de que a pessoa surda apresenta uma certa incapacidade, sendo esta

normalmente a incapacidade de comunicar sem ajustes razoáveis ao seu ambiente direto (Young et al., 2019).

Mundialmente, o Dia Mundial dos Surdos é celebrado no último domingo de setembro.

2.1.2.2 Em Portugal

Relativamente a Portugal e à assistência prestada às pessoas surdas ou com dificuldades auditivas no processo de inserção social, podemos considerar que este ainda apresenta falhas, existindo atualmente poucas ferramentas capazes de facilitar um dos pontos mais fundamentais desta inserção: a comunicação. Embora estas ferramentas sejam pouco utilizadas e de difícil acesso, podemos considerar que em termos de legislação, Portugal foi um dos primeiros países a integrar a língua gestual a nível nacional e adotá-la como língua oficial, tendo esta incorporação sido realizada em 1997.

Tal como em outros países, podemos considerar a existência de mais do que uma língua gestual, sendo estas línguas dependentes da comunidade surda à qual está associada e às variações existentes entre elas, embora exista a Língua Gestual Portuguesa (LGP), a única que é tomada em consideração em termos de legislação, embora esta também provenha e tenha sido desenvolvida pela comunidade surda, sendo a mais utilizada em Portugal.

Como maneira de comemorar a comunidade surda e as pessoas que a integram, dois dias nacionais foram criados, sendo estes o Dia Nacional do Intérprete da Língua Gestual Portuguesa (22 de Janeiro) e o Dia Nacional do Surdo (24 de Setembro).

Dentro de Portugal também existem várias associações vocacionadas para o apoio e prosperidade das comunidades surdas, como por exemplo a Associação Portuguesa de Surdos (APS), a Liga Portuguesa de Desporto para Surdos (LPDS), a Federação Portuguesa das Associações de Surdos (FPAS) e a Associação de Tradutores e Intérpretes de Língua Gestual Portuguesa (ATILGP).

2.1.3 Tecnologias de Auxílio

Com o maior esforço alocado na inserção de comunidades menos favorecidas na sociedade, o desenvolvimento de novas tecnologias e ferramentas é imperativo para garantir que este esforço é capaz de tornar esta inserção numa realidade e não apenas num sonho distante. Embora tenha começado apenas recentemente, o desenvolvimento destas novas ferramentas e tecnologias tem vindo a avançar muito rapidamente, embora ainda esteja longe de ser capaz de responder a todas as necessidades para as quais foram desenvolvidas e/ou não são capazes de ser integradas de uma maneira coletiva e

facilmente acessível, seja esta incapacidade relacionada com custo, dificuldade de distribuição, dificuldades de escalamento, resistência à mudança, falta de testes ou até mesmo falta de apoio pelas entidades responsáveis.

Dentro do grande leque de tecnologias e ferramentas em desenvolvimento ou já existentes no mercado e relacionadas com pessoas surdas ou com dificuldades auditivas, podemos destacar algumas, como os implantes cocleares e os serviços de retransmissão de vídeo (*video relay service*). Para além destas, podemos identificar algumas ferramentas e tecnologias que apenas recentemente começaram a abordar as necessidades das pessoas com dificuldades auditivas, muitas destas relacionadas com o uso de Inteligência Artificial (mais concretamente, *machine learning*). Este avanço tecnológico tem vindo a auxiliar diferentes soluções a melhor corresponder às necessidades destas mesmas pessoas, sendo este auxílio prestado através do uso de modelos (treinados) de *machine learning* para permitir a tradução de texto para língua gestual (e vice-versa) e através da criação de legendas para vídeos ou filmes. Algumas ferramentas criadas para auxiliar pessoas com dificuldades auditivas a melhor se integrarem na sociedade são apresentadas de seguida.

2.1.3.1 Smart Glasses

Óculos inteligentes, ou *Smart Glasses*, podem apresentar inúmeras características e serem utilizados para propósitos diferentes, mas apresentam, em regra, um pequeno computador capaz de oferecer algumas funcionalidades únicas ao seu utilizador. Uma dessas funcionalidades é a visualização de conteúdo na lente dos óculos. Esta capacidade permite ao utilizador visualizar dois ou mais conteúdos ao mesmo tempo, podendo esta capacidade provir ou de um *optical head-mounted display* (OHMD), ou de um *heads-up display* (HUD, implementado diretamente nos óculos), ou de um *augmented reality overlay* (AR overlay).

Várias soluções já existem, apresentando muitas vezes funcionalidades ou focos diferentes, embora algumas se foquem no auxílio a pessoas com dificuldades auditivas (Miller et al., 2017; Parker et al., 2022; Salvi et al., 2021).

2.1.3.2 Smart Gloves

Semelhante ao caso dos *Smart Glasses*, as *Smart Gloves*, ou luvas inteligentes, também são uma ferramenta utilizada no auxílio de pessoas com dificuldades auditivas e que utilizam uma língua gestual como principal meio de comunicação. Componentes normalmente utilizados são por exemplo *flex*

sensors, IMUs (Inertial Measurement Units), módulos de Bluetooth e Arduino, embora estes possam variar bastante, dependendo da solução estudada. Várias soluções relacionadas com a tradução língua gestual através destas mesmas luvas já foram criadas, muitas destas desenvolvidas em contexto académico, sendo alguns exemplos a luva desenvolvida por engenheiros da UCLA (Parker et al., 2022) ou a luva desenvolvida por estudantes de eletrónica da universidade de Tripoli (Abougarair, 2022) . Ao contrário dos *Smart Glasses*, estas luvas necessitam, em regra, de uma interface para permitir o uso das suas diferentes funcionalidades, sendo esta interface normalmente criada a partir de uma aplicação para dispositivos móveis. Uma das soluções comerciais já existente e mais desenvolvida será a *BrightSign Glove*(*BrightSign - Translate Any Sign into Any Language*, n.d.) , solução esta que será analisada mais à frente.

2.1.3.3 Digital Avatars

O uso de avatares digitais, ou *Digital Avatars*, tem vindo a aumentar na última década, apresentado, em regra, o intuito de representar, digitalmente, uma pessoa. O uso destes avatares é bastante variado, mas, no que toca ao seu uso no auxílio de pessoas com dificuldades auditivas, este costuma ser utilizado para representar artificialmente a tradução de palavras ou de voz para uma ou mais línguas gestuais, podendo este avatar ser apresentando em diferentes interfaces, como por exemplo os *Smart Glasses* e as aplicações provenientes das *Smart Gloves*. Estes avatares também podem ser integrados diretamente em *websites* ou aplicações móveis independentes, permitindo a tradução do texto presente numa dada página para uma língua gestual à escolha. Um bom exemplo destes avatares e do seu uso para o auxílio de pessoas com dificuldades auditivas é por exemplo o Avatar *VirtualSign*, avatar este desenvolvido por uma empresa portuguesa com o intuito de traduzir texto ou áudio para LGP (*BrightSign - Translate Any Sign into Any Language*, n.d.).

2.2 Tradução da Língua Gestual

Como já referido, um dos aspetos mais importantes no que toca à integração de indivíduos com dificuldades auditivas na sociedade é a capacidade (mútua) de comunicar livremente. De maneira a tornar esta necessidade uma realidade, é necessário desenvolver um novo sistema capaz de facilitar a comunicação entre indivíduos com dificuldades auditivas e os restantes indivíduos que compõem a sociedade.

De maneira a criar esta solução, vários caminhos podem ser tomados e várias ferramentas podem ser utilizadas, mas, normalmente, o uso de *machine learning*, e, consequentemente, o uso de *datasets*, serão sempre essenciais para permitir a tradução de uma dada língua gestual em texto e, posteriormente, voz e vice-versa.

Machine learning é um subcampo da Engenharia e da Ciência da Computação que pode ser definido como sendo um conjunto de métodos capazes de detetar automaticamente padrões em informação e posteriormente utilizar esses mesmos padrões para prever informação futura ou realizar outros tipos de decisões com base em incerteza (Murphy, 2012).

As aplicações deste subcampo são inúmeras, embora neste caso foquem-se sobretudo na identificação e reconhecimento de partes do corpo.

2.2.1 Datasets

Existem vários *datasets* criados com o propósito de possibilitar a tradução de uma dada língua gestual em texto ou voz, sendo importante avaliar aspetos como o número de palavras/termos que possuem e até que ponto podem ser utilizadas para criar soluções capazes de auxiliar indivíduos com dificuldades auditivas. De maneira a apresentar melhor os diferentes *datasets* existentes capazes de apoiar o desenvolvimento de soluções capazes de ajudar indivíduos com dificuldades auditivas, alguns destes são expostos de seguida de uma maneira mais detalhada, dando especial ênfase à composição (ou melhor, às diferenças de composição) de cada um dos *datasets* apresentados. A Tabela 1 representada de seguida, desenvolvida por um Investigador do Instituto de *Investigación en Informática LIDI*, permite comparar vários *datasets* existentes e passíveis de serem utilizados no desenvolvimento de soluções de auxílio à comunidade surda. É importante referir que a tabela apresentada apenas retrata quatro dos *datasets* analisados pelo investigador uma vez que a Tabela 1 é uma representação editada da tabela apresentada originalmente, de maneira a melhor apresentar alguns dos *datasets* existentes.

Tabela 1: Comparação de *datasets* (alterada) (Facundo Quiroga, 2022)

id	Name	Country	Classes	Subjects	Samples	Data	Language level	Type	Annotations	Availability
2	RWTH-PHOENIX-Weather	Germany	1200	9	45760	53gb	Sentence	Videos	Face, hand, end/start (unfinished)	Publicly Available
5	Boston ASL LVD	USA	3300+	6	9800		Word	Videos, multiple angles	hand, end/start	Publicly Available
14	DEVISIGN-L	China	2000	8	24000	?	Word	videos (rgb)		Contact author
20	LSFB-CONT	Belgium	6883	100	85000+	~2 TB	Word / Sentence	Videos	landmarks hand, end/start, french translation	Publicly available

Através da Tabela 1 é possível constatar as várias diferenças existentes entre quatro dos *datasets* existentes. A Língua Gestual a que se destina varia bastante, algo que infelizmente ainda não se repercute nas soluções atualmente existentes. O número de palavras ou frases contidas dentro de cada *dataset* varia consideravelmente, normalmente limitado pela extensão e número de vídeos gravados e utilizados para desenvolver esse mesmo *dataset*. Tanto pode ser considerado o uso de palavras como o de frases para a criação de *datasets*, aspeto este que em algumas situações tem em consideração tanto palavras como frases. Em regra, os *datasets* são compostos por uma coleção de vídeos, embora haja *datasets* que incorporam apenas imagens. A forma como as anotações são gravadas varia significativamente, mas apresenta obrigatoriamente o movimento da mão do início ao fim e pode incluir também o movimento da cabeça. Por fim, a disponibilidade destes *datasets* varia muito, podendo este ser de livre acesso, necessitar de contactar com o autor ou, em casos pouco frequentes, já não se encontrar disponível.

2.2.2 Soluções para LGP

Até à data de criação deste documento, existe apenas um único sistema capaz de auxiliar indivíduos que utilizem a LGP como principal e único método de comunicação. Este sistema chama-se *VirtualSign* e é capaz de traduzir português em texto para LGP (*Virtual Sign – Língua Gestual Portuguesa*, n.d.). De momento, este sistema é apenas composto por um avatar que pode ser utilizado em determinados *websites* para representar em LGP o texto detetado, encontrando-se em desenvolvimento um *plug-in* deste sistema para o *Microsoft Office* e uma aplicação móvel. As únicas aplicações práticas deste sistema podem ser encontradas no *website* do Município de Matosinhos, no *website* do Município de Guimarães e no *website* do Município de Albergaria-a-Velha (*WireAVATAR*, n.d.).

Para além deste sistema, uma outra aplicação móvel encontra-se em desenvolvimento, por parte da Fundação Altice. Esta aplicação, denominada LGP App, pretende realizar exatamente a mesma função que os avatares atualmente utilizados nos *websites* referidos, mas de momento existe apenas um único vídeo da mesma, sem qualquer data prevista de lançamento (*Programa Inclui / Fundação Altice*, n.d.). É de denotar que o avatar utilizado pelo sistema *SignAll* e o apresentado no vídeo da aplicação LGP são exatamente iguais, sendo apenas alterada a camisola do avatar para corresponder à organização a quem pertence a aplicação. Com este aspeto em conta, talvez esta aplicação LGP seja a referida como estando em desenvolvimento pelo *website* oficial do sistema *SignAll*.

2.2.3 Soluções analisadas

Com o auxílio de *machine learning* e dos *datasets* criados, já foram desenvolvidas várias soluções para tentar auxiliar indivíduos com dificuldades auditivas que utilizam uma língua gestual para comunicar. Embora o número de soluções existentes já seja elevado, infelizmente o número de soluções comercialmente disponíveis é bastante mais reduzido e o seu custo, em regra, é bastante alto, salvo casos em que o único dispositivo necessário é um telemóvel. Outro aspeto importante é que, embora existam soluções variadas, as vantagens e desvantagens que cada abordagem apresenta são claras, sendo então importante sempre ter em conta o contexto no qual a inserção de uma dada solução é necessário. Outro aspeto a ter em conta é que o desenvolvimento de várias soluções, infelizmente, não se traduz sempre em aspetos inovadores ou diferentes neste campo, sendo que muitas das soluções criadas apresentam as mesmas funcionalidades e ferramentas, embora possa-se focar (ou não) na tradução de uma língua gestual diferente. O fator inovador destas soluções também leva a que muitas destas sejam desenvolvidas em contexto académico, sendo o resultado final destes projetos de difícil acesso ou apresente pouca informação relativamente ao uso e funcionamento da solução em causa.

Infelizmente, o desenvolvimento de soluções que utilizem LGP ainda é extremamente reduzido, sendo apenas conhecido até à data do desenvolvimento deste documento duas aplicações ainda em desenvolvi, referidas anteriormente, que vão de alguma maneira auxiliar pessoas que utilizam esta língua gestual como principal método de comunicação, embora nenhuma destas esteja relacionada com o comunicação com outras pessoas.

2.2.3.1 BrightSign Glove

A BrightSign Glove é uma *Smart Glove* que, com o auxílio de uma aplicação móvel, é capaz de traduzir qualquer língua gestual para mais de 30 línguas faladas, apresentando o resultado da tradução de uma dada língua gestual através de voz (voz esta que pode ser customizada para 450 possibilidades diferentes) e é capaz de reconhecer qualquer língua falada e apresentá-la em texto.

Esta solução, ao contrário das seguidamente apresentadas, não depende diretamente de um *dataset* já criado, mas sim do *dataset* que será criado pelo utilizador desta mesma luva, o que favorece esta solução bastante no que toca à facilidade de comunicação com indivíduos que utilizam uma língua falada, mas, infelizmente, apresenta diversos desafios para o indivíduo que utiliza uma língua gestual para comunicar, uma vez que não só obriga a criação de um “dicionário” completo que, caso não exista, a luva não apresenta qualquer funcionalidade, como todo o processo de criação deste mesmo dicionário, tal como as restantes funcionalidades apresentadas na aplicação móvel, necessitam de conhecimento de uma dada língua falada, uma vez que todo o processo é guiado por texto escrito dentro da aplicação.

O custo desta solução, apresentado no *website* oficial da organização, é de 2325,95€, podendo este valor variar de acordo com o país no qual a compra é realizada. É importante referir que a aplicação móvel não apresenta nenhum custo acrescido. Na Figura 1 pode-se visualizar um exemplar da *Smart Glove* em questão.



Figura 1: BrightSign Glove (BrightSign, 2022)

2.2.3.2 SignAll Chat

A *SignAll Chat*, desenvolvida pela empresa *SignAll*, é a primeira solução desta empresa focada mais no processo de comunicação do que no de educação. Comparativamente às restantes soluções apresentadas, esta foi criada com o intuito de facilitar a integração de indivíduos com dificuldades auditivas que utilizam uma língua gestual para comunicar nas empresas, mais concretamente, nos setores que envolvem interação direta com o cliente. Esta solução, também em contraste com as outras analisadas, não foi concebida com o intuito de ser facilmente transportável, sendo em vez focada mais na criação de uma *workstation* capaz de captar, com facilidade, os sinais criados pelo colaborador e seguidamente apresentá-los através de texto. Infelizmente, à semelhança da *BrightSign Glove*, a resposta obtida do cliente que comunica através de língua falada, neste caso, necessariamente o Inglês, é apresentada em texto, o que obriga o colaborador a apresentar conhecimentos de uma língua falada. Esta solução baseia-se na solução principal desenvolvida pela *SignAll*, sendo esta a *SignAll Lab*, que no fundo é a *workstation* utilizada na solução *SignAll Chat*, embora esta última apresente algumas adições, como por exemplo a adição de um segundo monitor, para a visualização da conversa por parte do cliente. Tendo por base o custo da solução *SignAll Lab*, uma vez que esta serve como base para a solução apresentada, o custo da *SignAll Chat* foi considerado como sendo semelhante, não sendo assim inferior a 7000\$ por ano para um dado negócio, ou seja, cerca de 6530€. Na Figura 2 pode-se visualizar uma simulação do uso prático da *solução SignAll Chat* em contexto organizacional.



Figura 2: SignAll Chat (SignAll, 2021)

2.2.3.3 Hand Talk

A solução *Hand Talk* é uma aplicação móvel, gratuita, capaz de traduzir língua falada, mais concretamente, Inglês ou Português, para língua gestual, ASL ou Libras, respetivamente. A aplicação recebe texto ou voz e traduz este, através do avatar digital Hugo ou através do avatar digital Maya, para a língua gestual desejada, apresentando também a possibilidade de partilhar a animação criada. Em contraste com as outras soluções, esta não é utilizada diretamente por um indivíduo com dificuldades auditivas, mas sim por um indivíduo que utilize uma língua falada como principal método de comunicação, mas, tal como as restantes soluções, promove uma maior facilidade de comunicação e, consequentemente, uma maior integração destes mesmos indivíduos na sociedade. Embora esta seja das soluções mais desenvolvidas disponíveis atualmente, existem inúmeras soluções que realizam exatamente a mesma função, tendo uma destas sido comprada pela empresa que criou esta mesma solução (HandTalk, 2018). Como já referido, esta solução é gratuita e disponível diretamente no telemóvel. Na Figura 3 pode-se visualizar a página inicial da aplicação *HandTalk*.

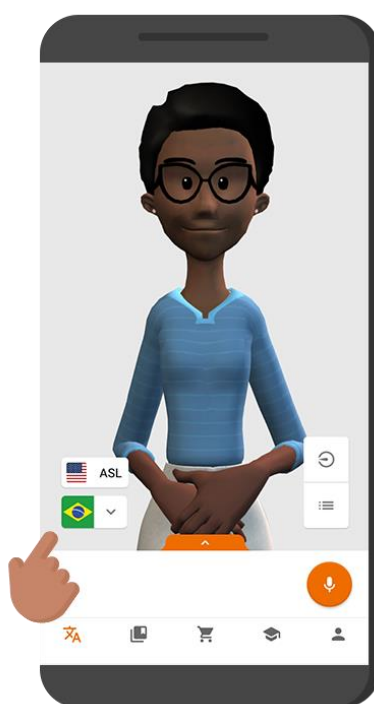


Figura 3: HandTalk (Hand Talk App, 2023)

2.2.3.4 Holographic Sign Language Interpreters

Esta solução, criada por uma equipa da Universidade de Purdue, localizada nos Estados Unidos, não apresenta um nome oficial, embora *Holographic Sign Language Interpreters* seja o nome mais

recorrentemente utilizado para referir a solução apresentada. Esta solução, através do uso dos *Smart Glasses* da *Microsoft*, *HoloLens*, representa holograficamente um avatar digital, avatar este que traduz, em tempo real, uma voz, mais concretamente, a de um professor universitário, e apresenta a tradução realizada por meio de animações pré criadas. Infelizmente, a solução apresentada poderá nunca ser disponibilizada, embora a sua inserção em contexto educativo, comparativamente às restantes apresentadas, apresentaria o maior valor. Na Figura 4 pode-se visualizar a interação que se deseja criar a partir da implementação desta nova solução, dando especial ênfase na representação, holográfica, do avatar digital.



Figura 4: Holographic Sign Language Interpreters (ACM SIGGRAPH Blog, 2022)

2.2.4 Análise inicial das soluções

Com o intuito de comparar inicialmente as soluções apresentadas, foi elaborada a Tabela 2. Com esta tabela, pretende-se avaliar os aspetos mais fundamentais das soluções apresentadas, sendo estes o tipo de língua gestual que é suportada pela solução apresentada, a forma como o utilizador dessa mesma solução pode interagir com a solução, mais concretamente, a interface ou modo de interação utilizado para interagir diretamente com a solução, o *input* necessário para permitir o funcionamento da solução, o *output* gerado após a utilização da solução e por fim a disponibilidade de uma dada solução, ou seja, se esta pode ser adquirida por um cliente interessado ou não.

Através da Tabela 2, conseguimos constatar que ASL foi a língua gestual mais utilizada, presente em todas as soluções estudadas, embora no caso da *BrightSign Glove*, o uso desta língua dependa completamente do utilizador e do conhecimento que este apresenta desta. A forma como o utilizador

interage com a solução varia para cada um dos casos, embora o uso de aplicações móveis esteja presente em dois dos casos, mas com funcionalidades diferentes (no caso da *BrightSign* como ferramenta de auxílio e no caso do *Hand Talk* como ferramenta principal). Relativamente ao input necessário, a *BrightSign Glove* e o *SignAll Chat* ambas necessitam de uma língua gestual, enquanto que as restantes necessitam de voz, embora a *SignAll Chat* utilize voz como *input* quando necessita de apresentar a resposta não do utilizador mas sim do indivíduo com quem o utilizador comunica. O *output* registado para cada uma das soluções, como esperado, é em regra o oposto do *input*, ou seja, em casos onde o input é texto ou voz o output registado é uma língua gestual e vice-versa, embora no caso do *SignAll Chat* todo o output gerado é na forma de texto. Por fim, Três das quatro soluções apresentam-se disponíveis diretamente ao consumidor, sendo a única exceção a *Holographic Sign Language Interpreters*, que ainda se encontra em desenvolvimento.

Tabela 2: Análise inicial das soluções

Solução	<i>BrightSign Glove</i>	<i>SignAll Chat</i>	<i>Holographic Sign Language Interpreters</i>	
			<i>Language</i>	<i>Hand Talk</i>
Língua gestual	Várias	ASL	SEE e ASL	ASL e Libras
Interface/modo interação	Luva/smartphone	Monitor/câmaras externas	<i>HoloLens</i> /avatar	Aplicação móvel/avatar (Hugo ou Maya)
Input	Língua gestual	Voz OU <i>ASL</i>	Voz	Texto ou voz
Output	Voz	Texto	<i>SEE</i> (avatar)	<i>ASL</i> ou Libras (avatar)
Disponibilidade	Disponível para venda	Disponível para venda	Solução em desenvolvimento	Disponível para venda

2.2.5 Análise detalhada das soluções

Após analisar os parâmetros básicos e característicos destas soluções, podemos em seguida analisar mais detalhadamente cada uma destas soluções apresentadas ao avaliarmos outros fatores relevantes e mais focados na possibilidade de adoção e facilidade de uso destas mesmas soluções como uma ferramenta útil. De maneira a avaliar estes fatores relevantes, foi criada a Tabela 3.

Fatores relevantes a serem avaliados são então a Portabilidade de uma dada solução, ou seja, até que ponto a solução criada pode ser transportada livremente e com que facilidade é que esse transporte

pode ser realizado. Este fator é importante para melhor compreender a facilidade que o utilizador terá a integrar com esta solução num novo meio ou para diferentes ocasiões dentro de um meio já definido. Quanto maior for a portabilidade de uma solução, mais fácil será o seu uso em atividades variadas, permitindo assim ao utilizador uma solução “para tudo”, embora soluções que apresentem uma maior portabilidade podem, embora não seja regra, apresentar menos funcionalidades para uma dada situação quando comparadas a uma solução apenas destinada a esse dado cenário.

Tanto o Custo como a Disponibilidade são fatores relevantes no que toca à capacidade de um dado utilizador de conseguir utilizar uma dada solução. Estes fatores estão diretamente relacionados, sendo que o Custo foca-se no valor que um indivíduo necessita pagar para adquirir ou usar uma dada solução e a Disponibilidade foca-se na capacidade que um indivíduo tem de conseguir adquirir determinada solução e aonde esta aquisição pode ser realizada. O custo, infelizmente elevado, pode ser considerado uma das principais razões pela falta de soluções facilmente acessíveis a pessoas com dificuldades auditivas.

As línguas que uma determinada solução consegue utilizar também é um fator relevante e que pode ser analisado mais extensivamente comparativamente à análise inicial, sendo importante distinguir soluções que apenas são capazes de facilitar a comunicação com uma língua falada ou com uma língua gestual de soluções que são capazes de comunicar com mais do que uma destes tipos de línguas.

Fatores como a Facilidade de Interação e a Comodidade proporcionada pela solução a indivíduos com dificuldades auditivas e a indivíduos sem dificuldades auditivas também são fatores extremamente importante e os quais não conseguimos analisar a partir da comparação inicialmente realizada, sendo especialmente estes fatores que permitem determinar até que ponto a integração de uma destas soluções como meio de auxílio na comunicação é um fator vantajoso ou apenas influência ligeiramente a qualidade de comunicação entre dois ou mais indivíduos.

Por fim, a Facilidade de uso, semelhantemente à Facilidade de Interação e à Comodidade dos interlocutores é um fator bastante importante para avaliar até que ponto uma solução auxilia o processo de comunicação entre dois indivíduos, avaliando neste caso mais concretamente o utilizador e a facilidade que o utilizador tem em usar uma dada solução.

Tabela 3: Análise detalhada das soluções

Solução	<i>BrightSign Glove</i>	<i>SignAll Chat</i>	<i>Holographic Sign Language Interpreters</i>	<i>Hand Talk</i>
Portabilidade	Facilmente transportável, uma vez que o pacote é apenas constituído por uma luva e a aplicação móvel, podendo haver a adição de uma coluna móvel	Muito difícil de transportar uma vez que se assemelha a uma estação de trabalho, necessitando de componentes como monitores e várias câmaras externas para realizar a interação.	Apenas necessita do uso de um telemóvel, apresentando assim uma portabilidade excelente.	Necessita apenas da utilização dos óculos HoloLens, sendo assim fácil realizar o transporte desta solução.
Custo	2365,95€	7000€/ano	Aplicação gratuita, embora apresente a possibilidade de gastar dinheiro para adquirir opções cosméticas para os avatares	Solução não comercial, embora os óculos custem entre 3500\$ e 5199\$ (3260-4840€)
Disponibilidade	Venda pública, disponível no site oficial ou em várias revendedoras online	Solução comercial para negócios, disponível através do website da empresa, mas sujeita ao preenchimento de um formulário e à apresentação de uma proposta por parte da empresa detentora do produto.	Aplicação gratuita disponível na <i>Google Play</i> e na <i>App Store</i>	Solução não disponibilizada ao público

Línguas	De acordo com os criadores, é capaz de traduzir qualquer língua gestual para texto ou voz, podendo esta tradução ser realizada para mais de 40 línguas faladas diferentes	Capaz de traduzir língua gestual americana (<i>ASL</i>) para Inglês e vice-versa	Capaz de traduzir tanto a língua inglesa como a língua portuguesa (do Brasil) para <i>ASL</i> ou Libras, respetivamente.	É captada a língua inglesa falada, que posteriormente pode ser traduzida tanto para <i>ASL</i> como para <i>SEE</i> .
Facilidade de interação	O utilizador (com a luva equipada) realiza o conjunto de sinais que previamente associou a texto e, através do uso da aplicação, o texto resultante aparece no ecrã e ao mesmo tempo é transmitido através das colunas do telemóvel ou da coluna móvel. Para receber uma resposta, o utilizador com dificuldades auditivas apenas tem de recorrer mais uma vez à aplicação, onde receberá a informação em formato de texto.	O colaborador (com dificuldades auditivas), utiliza <i>ASL</i> para comunicar, sendo esta captada através das câmaras e traduzida para texto, texto este que depois é apresentado no monitor ou ecrã utilizado. O cliente (sem dificuldades auditivas), responde através de comunicação verbal, que é traduzida em texto, resultando numa conversa (ou <i>chat</i>) entre as duas pessoas em texto corrido.	O utilizador (sem dificuldades auditivas) escreve o que pretende traduzir (ou vocaliza, sendo possível a aplicação captar som e traduzi-lo), sendo posteriormente apresentada a tradução em língua gestual do conteúdo pretendido.	O utilizador necessita apenas de se posicionar de maneira a captar a voz do professor, sendo todo o processo de tradução automático, embora o utilizador seja capaz de alterar algumas aspetos relativos ao avatar, como o tamanho deste.
Comodidade do indivíduo sem dificuldades auditivas	O indivíduo sem dificuldades auditivas não necessita de realizar um esforço extra, sendo a aplicação e a luva que se preocupa em fazer com que o utilizador “emita” o comportamento de pessoas	Não existe a opção de receber a mensagem do indivíduo com dificuldades auditivas através de voz, sendo obrigado assim a ler o texto apresentado no ecrã, embora o resto	Apenas necessita de instalar a aplicação e realizar uma operação extremamente simples, apresentando assim uma comodidade elevada	Não necessita de realizar nenhuma atividade de maneira a poder ser realizada a atividade de tradução

	sem dificuldades auditivas, sendo o conteúdo comunicado tanto através de texto como através de som.	do processo não apresente qualquer inconveniente para o indivíduo sem dificuldades auditivas (o cliente).		
Comodidade do indivíduo com dificuldades auditivas	O utilizador é obrigado a ser capaz de ler uma das 40 línguas disponíveis para poder realizar qualquer atividade dentro da aplicação e durante uma conversa com indivíduos sem dificuldades auditivas	O indivíduo com dificuldades auditivas é obrigado a saber comunicar através de uma língua falada, neste caso o inglês. Dado o contexto empresarial, a presença constata de câmaras também pode ser um problema para o indivíduo com dificuldades auditivas (neste caso, um colaborador dessa mesma empresa).	Necessita apenas de visualizar o conteúdo apresentado ou, através da partilha, do vídeo resultante do processo de tradução.	O utilizador não necessita de realizar nenhuma ação de maneira a obter a tradução do conteúdo falado em formato digital (neste caso, por holograma), mas necessita de estar a utilizar os óculos durante o processo de tradução
Facilidade de uso	A configuração do dispositivo pode ser morosa, mas após configurar é extremamente fácil de utilizar.	Após a instalação dos componentes, não é necessário realizar nenhuma atividade de maneira a permitir a realização da conversa, sendo assim de fácil uso.	Extremamente fácil de baixar e utilizar, não apresentando qualquer dificuldade ou ambiguidade.	Extremamente fácil de usar, requer apenas a utilização dos óculos.

De maneira a poder avaliar as soluções e o valor que cada uma destas traz para os indivíduos que usufruem destas mesmas soluções, foi criado um sistema de pontos capaz de melhor comparar as quatro soluções abordadas tendo em conta os sete fatores anteriormente analisados. É importante referir que o fator das línguas, de maneira a melhor avaliar as diferenças entre as quatro soluções, foi dividido em dois, apresentando agora uma componente referente às línguas gestuais utilizadas e outra componente referente às línguas faladas utilizadas. Ao contrário do fator Línguas, o fator Comodidade do indivíduo sem dificuldades auditivas e Comodidade do indivíduo com dificuldades auditivas foi agrupado num fator mais geral, denominado de Comodidade, sendo que cada um dos fatores anteriores passou a compor apenas um componente do fator mais abrangente. Por fim, o fator Custo foi avaliado tendo em conta o uso da solução por mais do que um ano, de maneira a melhor refletir o valor de cada solução. Estas mudanças foram realizadas com o intuito de garantir uma pontuação final capaz de melhor caracterizar as quatro soluções apresentadas.

O sistema de pontos criado tem em consideração uma variação de classificação de 1 a 5, sendo que quanto mais baixo for a pontuação, pior é a solução tendo em conta o fator avaliado. O sistema criado apresenta como métricas:

Portabilidade

- 1 - A solução não consegue de maneira nenhuma ser transportada para outro local;
- 2 - A solução necessita de um esforço considerável para ser transportada (equipa especializada e/ou equipamento especial);
- 3 - A solução pode ser transportada, embora necessite de ser desmontada (conhecimento básico e poucos componentes);
- 4 - A solução pode ser transportada facilmente, embora necessite de equipamento especializado;
- 5 - A solução necessita apenas de uma telemóvel ou outro dispositivo “essencial”.

Custo

- 1 – Mais de 10000€;
- 2 – Mais de 5000€;
- 3 – Mais de 2500€;
- 4 – Mais de 1000€;
- 5 – Gratuita.

Disponibilidade

- 1 – Inacessível;
- 2 – Acessível apenas em contexto não comercial ou de investigação;

- 3 – Acessível apenas a organizações;
- 4 – Acessível a qualquer pessoa, mas apresenta um custo (simbólico ou não) ;
- 5 – Acessível a qualquer pessoa e sem qualquer custo.

Línguas Gestuais do fator Línguas

- 1 – 1;
- 2 - 2;
- 3 - 3;
- 4 – 4;
- 5 – 5 ou mais.

Línguas Faladas do fator Línguas:

- 1 – 1;
- 2 - 2;
- 3 - 3;
- 4 – 4;
- 5 – 5 ou mais.

Facilidade de Interação

- 1 – A interação entre pessoa com dificuldade e pessoa sem é impossível através unicamente desta solução;
- 2 – Ambos os indivíduos apresentam dificuldades acrescidas na comunicação através deste produto;
- 3 – Ambos ou apenas um dos indivíduos necessita de se adaptar significativamente à forma como a interação é realizada (como por exemplo, apresentar conhecimentos de uma língua que não a sua principal);
- 4 – Ambos ou um dos indivíduos necessita de realizar alguma forma, ligeira, de adaptação de maneira a realizar/facilitar a interação;
- 5 – Nenhum dos indivíduos necessita de realizar qualquer sacrifício de maneira a facilitar a interação entre eles.

Comodidade do indivíduo sem dificuldades auditivas

- 1 – não é capaz de interagir com/utilizar a solução;
- 2 – necessita de alterar drasticamente a forma como comunica (inserção de novas ferramentas complicadas, por exemplo);
- 3 – necessita de realizar ajustes consideráveis na forma como comunica (melhor aprender uma dada língua, por exemplo) ;

4 – necessita de realizar pequenos ajustes na forma como comunica (utilização de funcionalidades básicas, por exemplo);

5 – não necessita de realizar nenhuma atividade para melhorar a comunicação.

Comodidade do indivíduo com dificuldades auditivas

1 – não é capaz de interagir com/utilizar a solução;

2 – necessita de alterar drasticamente a forma como comunica (inserção de novas ferramentas complicadas, por exemplo);

3 – necessita de realizar ajustes consideráveis na forma como comunica (melhor aprender uma dada língua, por exemplo) ;

4 – necessita de realizar pequenos ajustes na forma como comunica (utilização de funcionalidades básicas, por exemplo);

5 – não necessita de realizar nenhuma atividade para melhorar a comunicação.

Facilidade de Uso

1 – impossibilidade de uso sem a presença de um técnico especializado;

2 – necessita de conhecimento avançados, embora possa ser utilizado individualmente;

3 – necessita de alguns conhecimentos não avançados de maneira a poder ser utilizado;

4 – necessita de conhecimentos mínimos para poder ser utilizado;

5 – qualquer indivíduo é capaz de utilizar (considerando que apresenta capacidades mínimas de independência).

Na Tabela 4 podemos visualizar os resultado obtidos após a análise cuidada de cada uma das soluções apresentadas.

Tabela 4: Comparação das quatro soluções

Solução		BrightSign Glove	SignAll Chat	Hand Talk	Holographic Sign Language Interpreters
Portabilidade		4	2	5	4
Custo		4	1	5	3
Disponibilidade		4	3	5	1
Línguas	LG	5	1	2	2
	Faladas	5	1	2	1
Facilidade de interação		3	3	1	1

Comodidade	ISDA	5	4	4	5
	ICDA	2	3	1	5
Facilidade de uso		2	4	4	5
Total (35)		25.5	17.5	24.5	20.5
Total sem Custo e Disponibilidade (25)		17.5	13.5	14.5	16.5

Através da elaboração da Tabela 4, conseguimos constatar que a solução que apresenta a maior pontuação foi a *BrightSign Glove*, com uma pontuação de 25.5. Podemos considerar que este aspeto se deveu sobretudo ao facto de permitir o utilizador da luva comunicar utilizando qualquer língua gestual que deseja e para uma grande variedade de línguas faladas disponíveis. O custo também foi um fator importante, que, embora elevado, comparativamente às restantes soluções, não o é, e o facto de não apresentar qualquer constrangimento para um dos indivíduos aquando do momento da comunicação também foi um ponto positivo. Embora esta solução apresente a maior pontuação, apresenta defeitos graves da perspetiva de um indivíduo com dificuldades auditivas, sendo o mais relevante o facto de que o indivíduo com dificuldades necessita, obrigatoriamente, de conhecer uma língua falada para poder utilizar a aplicação da luva que, consequentemente, controla a luva em si. Sem conhecimentos de uma língua falada, o utilizador necessitará de ajuda para criar o dicionário de sinais que irá utilizar para comunicar através da luva. Este problema poderia ser resolvido através da criação de *datasets* padrão para alguma das línguas gestuais mais utilizadas, que em si não limita a possibilidade de o utilizador criar o seu próprio dicionário, e a adição de símbolos em vez de texto nos botões ou menus necessários para o uso da aplicação e das suas diversas ferramentas.

Em segundo lugar, com uma pontuação inferior em um ponto, com 24.5, ficou a solução *Hand Talk*. Comparativamente com as outras soluções, esta destaca devido sobretudo ao facto de ser acessível por qualquer indivíduo, não apresentar qualquer custo e não necessitar de qualquer tipo de equipamento para além do telemóvel para poder funcionar. A grande desvantagem desta solução, quando analisada o seu valor para o auxílio na comunicação, é o facto de esta não se focar diretamente no auxílio dos dois ou mais indivíduos numa conversa, mas sim focar-se no auxílio de apenas um dos indivíduos, neste caso, do indivíduo sem dificuldades auditivas. O facto de se focar apenas num dos utilizadores, neste caso, leva a que um indivíduo sem conhecimento de uma dada língua gestual seja apenas capaz de comunicar com um indivíduo que tenha conhecimento de uma dada língua gestual através da aplicação, mais concretamente, da partilha da animação do avatar da aplicação, o que em si obrigaria o indivíduo com conhecimentos de língua gestual a comunicar obrigatoriamente a partir de texto (língua falada).

Em terceiro lugar ficou a solução *Holographic Sign Language Interpreters*, com 20.5 pontos. Esta solução apresenta grandes vantagens no que toca à comodidade dos indivíduos que estão a comunicar e a facilidade de uso, mas não permite dois indivíduos comunicarem entre eles diretamente, uma vez que o sistema é apenas capaz de detetar voz e traduzi-la para língua gestual e não o contrário, e, talvez mais importante ainda, não se encontra disponível comercialmente, sendo impossível adotá-la de momento. Por fim, em quarto lugar, ficou a solução *SignAll Chat*. Esta solução não se distingue particularmente em nenhum dos fatores, não apresentando uma vantagem clara num destes, mas infelizmente apresenta a pior pontuação no fator de Portabilidade, uma vez que não foi concebida para ser transportada facilmente, visto ser baseada numa *workstation* e apresenta o maior custo de todas as soluções, obrigando também o utilizador (neste caso colaborador) a apresentar conhecimentos de uma língua falada (neste caso, de inglês).

Através da classificação obtida na Tabela 4, as soluções também podem ser avaliadas tendo apenas em conta o valor que trazem para a comunicação em si, sendo este valor obtido através da não contabilização dos dois fatores que não afetam (diretamente) a comunicação, sendo estes o Custo e a Disponibilidade. Ao não contabilizar estes dois fatores, a solução com maior pontuação continuaria a ser a *BrightSign Glove*, agora com uma diferença de apenas um ponto para o segundo classificado, neste caso o *Holographic Sign Language Interpreters*, que passou para segundo lugar, com 16.5 pontos. A terceira solução passou a ser a *Hand Talk*, com uma pontuação de 14.5. Por fim, a solução *SignAll Chat* ocuparia a quarta posição, apresentando a pontuação mais baixa, de 13.5. Com a remoção do fator Custo e do fator Disponibilidade conseguimos melhor avaliar uma solução pelo que ela proporciona ao utilizador, o que levou a uma mudança considerável nas posições iniciais das soluções.

3. ESPECIFICAÇÃO DA SOLUÇÃO

De maneira a contribuir para a inserção de indivíduos com dificuldades auditivas que utilizam como principal meio de comunicação a LGP, propõe-se, na primeira secção deste capítulo, a elaboração de uma aplicação, através da ferramenta *Unity*, capaz de traduzir língua gestual, neste caso, LGP, para português, em formato de texto, e de traduzir português, transmitido oralmente, para português, em formato de texto. A solução final deve ser capaz de funcionar nos *Smart Glasses HoloLens 2*, desenvolvidos pela *Microsoft*.

A segunda secção deste capítulo foca-se em apresentar os requisitos funcionais e não funcionais que a solução criada deve cumprir, tendo em conta as características e funcionalidades desejadas.

Por fim, na terceira secção deste capítulo, esta é referente à conceptualização inicialmente realizada da aplicação referida e que se deseja criar, onde são definidas as interfaces que devem compor a aplicação, a estrutura que estas interfaces devem apresentar e as funcionalidades que devem apresentar.

3.1 Contexto

A ideia fundamental desta nova solução é permitir a comunicação direta entre dois indivíduos, um que apresenta dificuldades auditivas e utiliza a LGP como principal meio de comunicação e outro indivíduo que não apresenta dificuldades auditivas e não apresenta conhecimentos suficientes de LGP para poder comunicar através desta língua.

De maneira a criar esta solução, será utilizada uma rede neuronal, já treinada, criada pelo Centro de Computação Gráfica (CCG) , capaz de reconhecer um certo conjunto de sinais de LGP e consequentemente atribuir um significado a esses mesmos sinais, e o *Mediapipe*, uma biblioteca criada pela *Google* capaz de detetar um determinado número de pontos (denominados de *landmarks*) a partir de imagens através do uso de *machine learning* (ML), pontos estes que podem ser utilizados para averiguar a posicionamento das mãos, cara, e postura de um dado indivíduo, entre outros.

A solução final apresentada será criada em *Unity* e será composta por duas interfaces, uma para cada utilizador, consoante as necessidades que este apresenta. Caso o utilizador em questão apresente dificuldades auditivas, a solução tem de ser capaz de detetar a posição das mãos do utilizador através da biblioteca *Mediapipe*, por meio da captação de imagem da câmara embutida nos óculos *HoloLens*. Após captar a posição das mãos, mais concretamente, dos vinte e um pontos para cada mão reconhecidos pela biblioteca *Mediapipe* e apresentados na Figura 5 (o nome atribuído ao modelo dos

vinte e um pontos é *Hand Landmark Model*), a solução tem de ser capaz de atribuir essa posição, através do uso da rede já treinada, importada para o *Unity* através do uso do *package ML-Agents*, à palavra ou frase correspondente em língua portuguesa.

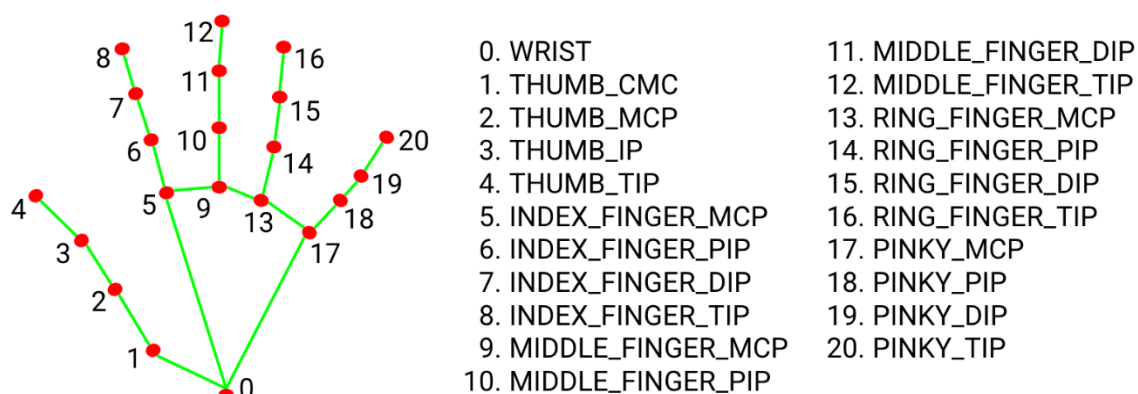


Figura 5: Hand Landmark Model (Mediapipe, 2020)

Após transformar o sinal criado pelo utilizador em texto da língua portuguesa, este texto tem de aparecer na interface criada para o segundo utilizador, neste caso o utilizador sem dificuldades auditivas. Caso ocorra o inverso, a solução tem de ser capaz de reconhecer a voz do utilizador sem dificuldades auditivas e apresentar o texto resultante desse reconhecimento nos *Smart Glasses* do utilizador com dificuldades auditivas.

3.2 Requisitos

De maneira a assegurar que a solução a criar corresponde às necessidades identificadas, foi necessário identificar os requisitos funcionais e não funcionais que devem constar da solução final.

3.2.1 Requisitos funcionais

Na Tabela 5 podem ser visualizados os cinco requisitos funcionais identificados.

Tabela 5: Requisitos funcionais

Requisitos	Descrição
Capaz de capturar os sinais	Para que a solução seja capaz de mediar a comunicação entre dois indivíduos, necessita de ser capaz de capturar os sinais realizado pelo indivíduo com dificuldades auditivas

Capaz de traduzir os sinais capturados	Após capturar os sinais, a solução tem de ser capaz de identificar o significado de cada um destes sinais.
Apresentar o texto resultante da tradução	Após identificar o significado de cada sinal, a solução tem de ser capaz de apresentar a expressão resultante na interface desejada.
Disponibilizar diferentes interfaces consoante o utilizador	A solução deve apresentar uma interface para cada um dos dois tipos de utilizador, sendo estes utilizadores que utilizam a LGP para comunicar e utilizadores que utilizam a língua portuguesa para comunicar.
Apresentar secção destinada à ajuda na utilização	A aplicação deve apresentar uma pequena secção onde explica as diferentes funcionalidades presentes na interface de cada um dos tipos de utilizadores

3.2.2 Requisitos não funcionais

Na Tabela 6 podem ser visualizados os cinco requisitos não funcionais identificados.

Tabela 6: Requisitos não funcionais

Requisitos	Descrição
Compatível com o dispositivo <i>HoloLens 2</i>	A aplicação tem de ser capaz de correr nos <i>Smart Glasses HoloLens 2</i> .
Sistema deve ser independente	A solução tem de ser capaz de funcionar sem influência externa, apresentando todos os componentes necessários no <i>HoloLens</i> utilizado.
Utilizar <i>Unity</i> como ferramenta principal	A ferramenta <i>Unity</i> deve ser utilizada como principal ferramenta para criar a solução idealizada.
Utilizar o <i>Mediapipe</i> como ferramenta de deteção de sinais	A biblioteca <i>Mediapipe</i> deve ser utilizada para identificar os sinais criados pelo utilizador com dificuldades auditivas.
Funcionamento sem falhas	A solução final deve conseguir funcionar sem apresentar falhas.

3.3 Conceptualização Inicial da Aplicação

Inicialmente, e em concordância com a especificação da solução, o desenvolvimento do projeto, em *Unity*, focou-se sobretudo na criação de duas interfaces, uma destinada para indivíduos com dificuldades auditivas (ICDA) e outra destinada para indivíduo sem dificuldades auditivas (ISDA). Para cada uma destas interfaces, as funcionalidades que se esperavam encontrar em cada uma já foram exploradas, embora seja importante clarificar estas mesmas funcionalidades e a forma como se espera que estas sejam incorporadas na solução final.

Para a interface destinada a ICDA, esta necessita ser capaz de converter, em tempo real, língua portuguesa transmitida oralmente (proveniente do indivíduo sem dificuldades auditivas) para língua portuguesa em formato de texto, de maneira a poder ser compreendida por parte do indivíduo com dificuldades auditivas.

Para a interface destinada a ISDA, esta necessita de ser capaz de converter, em tempo real, LGP (proveniente do indivíduo com dificuldades auditivas e obtidos através da câmara do dispositivo) para língua portuguesa em formato de texto.

Para desenvolver esta aplicação, foi inicialmente criado um esboço, esboço este que apenas pretende representar os elementos-chave que devem estar presentes aquando da implementação e desenvolvimento de cada uma das interfaces referidas. Este esboço pode ser visualizado na Figura 6.

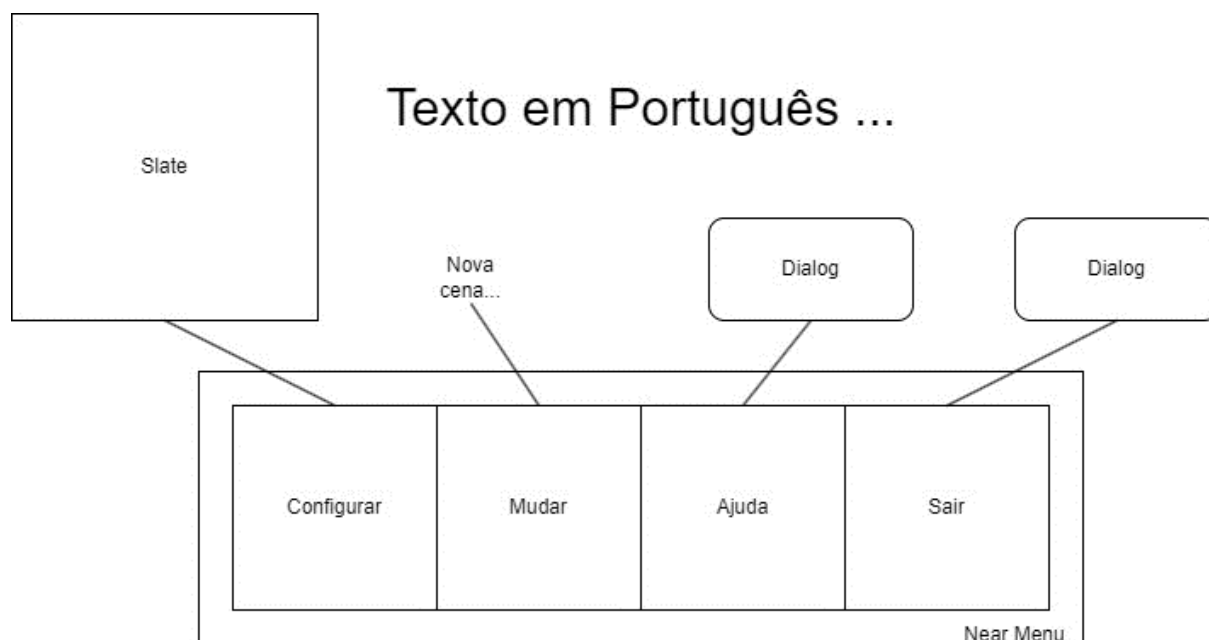


Figura 6: Esboço da interface idealizada

É relevante destacar o facto deste esboço ter sido criado com base na interface utilizada nos dispositivos *Hololens*. Foi apenas realizado um esboço uma vez que todos os elementos estruturais inicialmente pensados podiam ser aplicados de forma diferente em cada interface de maneira a corresponder às necessidades destas sem afetar as funcionalidades que se esperam encontrar em cada uma.

Após ponderar inicialmente a estrutura que cada interface deve apresentar com base no esboço criado, foi então necessário avaliar com mais detalhe as ferramentas que devem ser utilizadas para permitir criar estas mesmas interfaces e, sobretudo, as ferramentas que podem ser utilizadas para criar as funcionalidades esperadas destas mesmas interfaces.

Relativamente à criação estética da interface, optou-se pelo uso do *package Mixed Reality Toolkit*. Este package, desenvolvido pela *Microsoft*, fornece as ferramentas, *framework* e objetos necessário para criar aplicações para *Hololens 1* e *Hololens 2*, sendo a solução oficial para a criação de interfaces nos dispositivos referidos.

Relativamente às ferramentas utilizada para a criação das funcionalidades que se esperam estar presentes na aplicação, optou-se pelo uso da função “Dictation”, proveniente do *package MRTK*, para a deteção e transcrição de áudio em português para texto em português e optou-se pelo *package MediaPipeUnityPlugin* para permitir o uso do *MediaPipe* diretamente no Editor *Unity*. Também é importante referir o uso da rede neuronal previamente mencionada, proveniente do Centro de Computação Gráfico (CCG), capaz de traduzir/atribuir significado ao conjunto de *landmarks* identificados pelo *Mediapipe*. Esta rede será utilizada diretamente no Editor através do *package ML-Agents*.

Relativamente à aplicação como um todo, esta deve ser desenvolvida em *Unity*, ferramenta já previamente selecionada, sendo a escolha da versão a ser utilizada extremamente importante uma vez que esta influência não só o funcionamento de elementos relevantes como o *MediaPipe* e o *Mixed Reality Toolkit*, mas também pode criar problemas de incompatibilidade com outras ferramentas ou *packages* igualmente importantes para o desenvolvimento da aplicação. Relativamente ao *MRTK*, é importante evidenciar que este *package* apenas pode ser utilizado em *Unity 2020* ou *Unity 2021* (ambas *LTS*), enquanto o package *MediaPipeUnityPlugin*, o único plugin atual que permite o use do *MediaPipe* diretamente no *Unity*, é apresentado como sendo um *Native Plugin* do *Unity*, baseado na versão 2022.3.6f1 deste, embora não apresente, à partida, qualquer incompatibilidade com versões mais antigas do *Unity*. Tendo isto em conta, foi escolhida a versão do *Unity* 2020.3.30f1, sobretudo pelo facto de já apresentar alguma familiaridade com esta versão e ser compatível com a maioria das ferramentas/*packages* que podem ser futuramente utilizadas/adotados no projeto.

Tendo em conta os packages já referidos e as diferentes necessidades de cada interface, primeiro relativamente à interface destinada a ISDA, visto esta já se encontrar melhor idealizada, foi confirmado o uso da rede neuronal fornecida pelo CCG, rede esta que necessita obrigatoriamente do uso do *MediaPipeUnityPlugin*, visto ser o único package capaz de permitir o uso do *MediaPipe* diretamente em *Unity*. Para permitir o uso da rede diretamente em *Unity* também é necessário utilizar um *package* capaz de realizar essa mesma ação, sendo o único *package* fornecido oficialmente pelo *Unity* o *ML-Agents*, *package* este compatível com qualquer versão do *Unity* após a 2020.1. É importante realçar que o *package ML-Agents* utiliza *Barracuda*, uma biblioteca de inferência de redes neurais, também disponível como um *package* para diferentes versões do *Unity*, incluindo a utilizada atualmente. Para ambos os *packages*, a rede neuronal necessita de se encontrar em formato *ONNX*.

Relativamente agora à interface para ICDA, foi necessário analisar possíveis ferramentas capazes de converter língua portuguesa transmitida oralmente em texto. Felizmente, o próprio *Hololens* é capaz de detetar e transformar em texto o som captado pelo microfone do próprio dispositivo, sendo esta função denominado “Dictation” e proveniente diretamente do *MRTK* ou *Mixed Reality Toolkit*.

Com base na escolha das ferramentas a ser utilizadas e na construção de um esboço inicial das interfaces, conclui-se então que a aplicação final deverá ser composta por pelo menos duas interfaces distintas, semelhantes em termos de estrutura, mas diferentes em termos de funcionalidade e conteúdo. Para o desenvolvimento estético e comportamental das interfaces, deve ser utilizado o *Mixed Reality Toolkit*, de maneira a manter coerência com as restantes interfaces existente nos equipamentos *Hololens* e de maneira a facilitar o processo de desenvolvimento.

Outro aspeto que foi ponderado aquando da idealização das interfaces distintas foi a criação de um painel introdutório que permitiria selecionar a interface desejada, intitulado “interface inicial”. A presença deste painel não só serviria para garantir que nenhum dos dois tipos distintos de utilizadores necessitaria de realizar ações adicionais para poder chegar à interface desejada (caso existissem apenas duas interfaces, o utilizador, independente do tipo, seria forçado a começar numa das interfaces que poderá não ser a interface que deseja utilizar e poderá enfrentar dificuldades a mudar para a interface desejada caso não apresente conhecimento prévio da aplicação em questão) como também permitiria uma introdução inicial às duas interfaces existentes.

Devido à facilidade de implementação desta “interface inicial”, foi decidido adicioná-la, sendo um pequeno esboço apresentado na Figura 7.

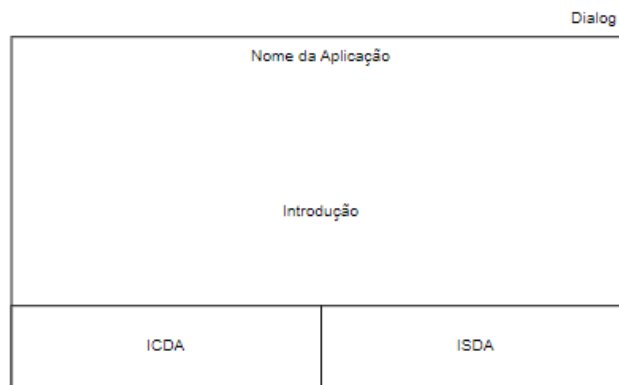


Figura 7: Esboço da “interface inicial”

É esperado que o resultado final apresentado simule a conversa ilustrada na Figura 8, permitindo, através do uso dos *Smart Glasses HoloLens 2*, duas pessoas com conhecimentos linguísticos diferentes comunicarem entre si.

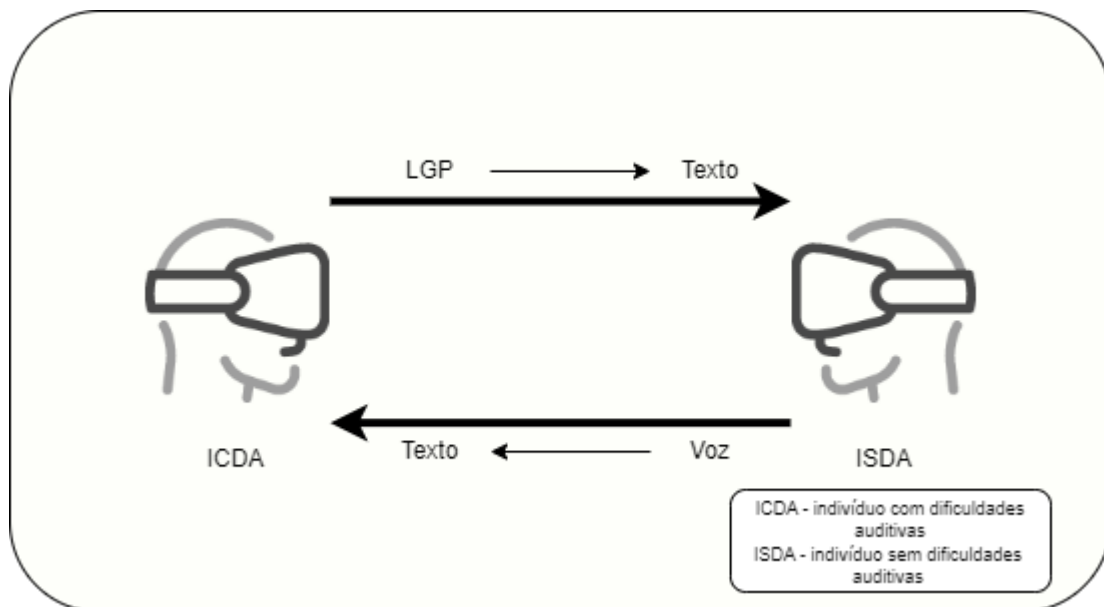


Figura 8: Interação entre utilizadores

4. DESENVOLVIMENTO

Este capítulo pretende descrever o processo de desenvolvimento da aplicação, os problemas identificados e apresentar os componentes relevantes utilizados durante o desenvolvimento da mesma.

Este capítulo é bastante extenso, sendo este composto por seis secções diferentes.

A primeira secção é referente às ferramentas utilizadas para criar e testar a aplicação, onde é apresentada uma pequena descrição da função de cada ferramenta tal como do processo de instalação/uso da mesma.

A segunda secção descreve o processo de desenvolvimento das três interfaces inicialmente idealizadas, incluindo as funcionalidades que se esperava que cada uma destas interfaces apresente.

A terceira e quarta secções ambas apresentam os problemas identificada aquando do processo de desenvolvimento da aplicação tal como as alternativas adotadas para resolver estes mesmos problemas.

A grande distinção entre estas duas secções é que a primeira é referente aos problemas identificados no processo inicial de desenvolvimento, realizado de acordo com a idealização inicial da aplicação, enquanto a segunda é referente aos problemas identificados após a realização das primeiras alterações até ao fim do desenvolvimento da aplicação criada.

A quinta secção descreve o desenvolvimento da aplicação final, tendo em consideração todos os problemas identificados e as alternativas adotadas. De maneira geral, a estrutura da cena criada é bastante semelhante à inicialmente idealizada, embora o funcionamento da aplicação seja completamente diferente.

Por fim, a sexta secção apresenta o processo de *deployment* da aplicação no dispositivo alvo.

4.1 Instalação e Uso das Ferramentas

Para permitir o desenvolvimento da aplicação em questão, constituída fundamentalmente por duas interfaces, cada uma destinada a um tipo diferente de utilizador, e por uma interface “inicial”, criada para permitir a escolha da interface desejada e introduzir as funcionalidades presentes em cada uma das interfaces previamente referidas, é necessário instalar um conjunto, algo extenso, de ferramentas, aplicações ou *packages* capazes de permitir ou auxiliar o bom funcionamento desta mesma aplicação, permitir a realização do processo de *build* e *deployment* desta mesma aplicação no dispositivo alvo ou testar a aplicação criada no dispositivo alvo.

É relevante referir que muitas das ferramentas utilizadas foram adicionadas durante o processo de desenvolvimento do projeto e devido às mudanças realizadas ao mesmo.

4.1.1 *Unity* (2020.3.30f1)

A primeira ferramenta, na qual é baseado todo o desenvolvimento da aplicação em questão, é o *Unity*, mais concretamente a versão 2020.3.30f1. Embora esta versão faça parte do *Unity 2020*, foi publicada em 2022 e é uma versão *LTS*, ou *Long-Term Support*, destinada à criação e desenvolvimento de projetos que se encontram em produção ou prestes a ser lançados. Para instalar esta ferramenta, é necessário primeiro instalar o *Unity Hub* (neste caso, a versão utilizada foi a mais recente, 3.5.2). O *Unity Hub* foi instalado diretamente a partir do *website* oficial do *Unity*.

Após instalar o *Unity Hub*, pode então ser realizado a instalação da versão desejada na secção “Installs”. Tendo em conta a aplicação a desenvolver, é essencial que durante a instalação do *Unity* este apresente como módulos adicionais “Universal Windows Platform Build Support” e “Windows Build Support (IL2CPP)” como apresentado na Figura 9, de maneira a permitir o desenvolvimento para dispositivos *Hololens*.

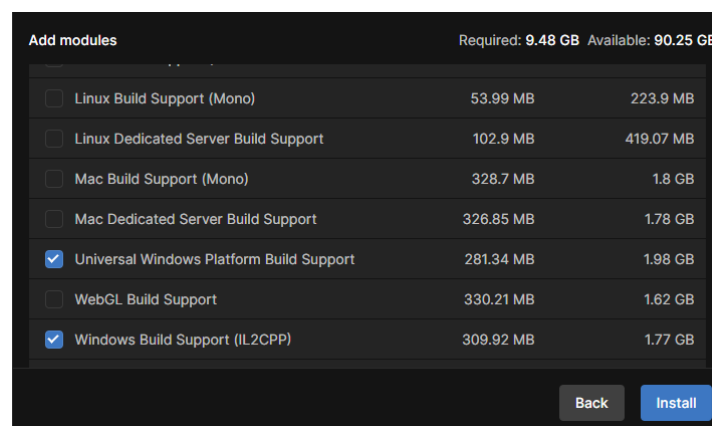


Figura 9: Módulos *Unity* essenciais

4.1.2 *Visual Studio 2019* (16.11.30) e *Visual Studio 2022* (17.5.5)

A segunda ferramenta em questão é essencial para permitir o desenvolvimento da aplicação em *Unity* é o *Visual Studio*, mais concretamente o *Visual Studio 2019* (versão 16.11.30) e o *Visual Studio 2022* (versão 17.5.5). Para o desenvolvimento de *scripts* em *Unity* é necessário utilizar um Editor Externo de *Scripts* (ou *External Script Editor*) capaz de criar ou modificar código em C#, a linguagem de programação usada no desenvolvimento de aplicações em *Unity*. No caso desta aplicação, embora estejam listadas duas versões distintas do *Visual Studio*, apenas o *Visual Studio 2019* foi utilizado para a elaboração de

scripts para a aplicação, enquanto o *Visual Studio 2022* foi utilizado para testar a aplicação em ambiente simulado e para realizar o *deployment* da aplicação no dispositivo alvo.

O processo de instalação do *Visual Studio 2019* e do *Visual Studio 2022* é igual, sendo apenas necessário aceder ao website oficial do *Visual Studio* e seleccionar a versão desejada.

Após instalar as versões desejadas, é necessário abrir a aplicação *Visual Studio Installer*, de maneira a instalar as “Cargas de Trabalho” e os “Componentes Individuais” necessários para permitir o funcionamento dos *packages* previamente seleccionados. As “Cargas de Trabalho” que o *VS 2019* necessita são “Desenvolvimento para Desktop em C++”, “Ferramentas de build da Plataforma Universal do Windows”, “Desenvolvimento de jogos com Unity” e “Desenvolvimento para Desktop com .NET” e os “Componentes individuais” que necessita são “SDK do Windows 10 (10.0.19041.0)”, “Conetividade do dispositivo USB” e “Suporte à Plataforma Universal do C++ para ferramentas de build v142 (ARM64)”. Relativamente ao *VS 2022*, este necessita das mesmas “Cargas de Trabalho” que o *VS 2019* e de apenas três dos “Componentes individuais”, sendo estes “SDK do Windows 10 (10.0.19041.0)”, “Conetividade do dispositivo USB” e “Suporte à Plataforma Universal do C++ para ferramentas de build v142 (ARM64)”.

Com a instalação de duas versões diferentes de *VS*, é importante garantir que a utilizada no *Unity* é a *VS 2019*. Para tal, basta aceder à secção “Preferences” do projeto criado e ver qual é o *External Script Editor* seleccionado.

4.1.3 *MRTK* (2.8.3)

Relativamente ao package *MRTK*, ou *Mixed Reality Toolkit*, apesar deste ser um dos *packages* mais importantes do projeto, a instalação do mesmo é relativamente simples. Primeiro, é necessário instalar a ferramenta “Mixed Reality Feature Tool”, disponibilizada no *website* oficial da *Microsoft*. Com esta ferramenta instalada, basta inserir a localização do projeto de *Unity* criado e seleccionar os *packages* do *MRTK* que necessitam estar presentes no projeto, sendo estas neste caso: “Mixed Reality Toolkit Foundation”, “Mixed Reality Toolkit Standard Assets”, “Mixed Reality Toolkit Tools” e “Mixed Reality Toolkit OpenXR Plugin”.

Após instaladas as *features*, é necessário abrir o projeto existente e configurar o package *MRTK*, tendo sido adotadas as definições padrão. A janela de configuração aparece mal os *packages* são adicionados e encontra-se representada na Figura 10.

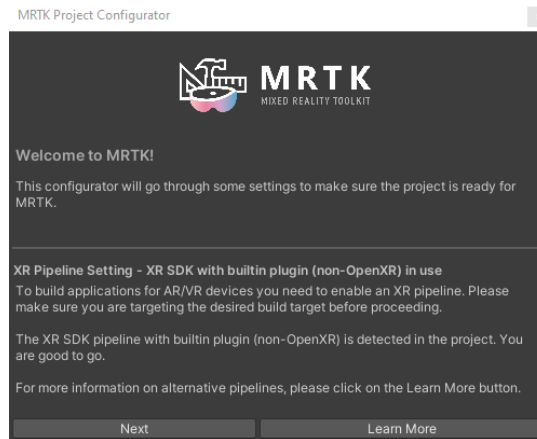


Figura 10: Configurador *MRTK*

No caso de ser necessário adicionar qualquer *feature* deste *package* no projeto criado para além das já descarregadas, é essencial garantir que nenhuma das *features* novas a instalar já existe no projeto, uma vez que a ferramenta de instalação não verifica a existência dos *packages* a instalar dentro do projeto.

4.1.4 *ML-Agents* (1.0.8)/ *Barracuda* (2.1.3)

Tanto o *package ML-Agents* como o *package Barracuda* são necessário para poder implementar a rede neuronal diretamente no Editor *Unity*, visto serem os únicos *packages* oficiais capazes de inferir a rede neuronal em questão.

Apesar de *ML-Agents* e de *Barracuda* serem *packages* oficiais e de terem de ser instalados separadamente, algo que em versões passadas não acontecia, o *package Barracuda* é uma dependência do *package ML-Agents* e, consequentemente, essencial para permitir ao *ML-Agents* realizar inferência, a principal função desejada para esta aplicação.

Para realizar a instalação do *ML-Agents* na versão do *Unity* utilizada, basta apenas procurar o mesmo no *Package Manager* e instalar a versão mais recente.

Para a instalação do *package Barracuda*, o processo já é mais complexo, devido sobretudo ao facto de este já não se encontrar mais no *Package Manager*. Esta alteração foi realizada justamente na versão do *Unity* na qual o projeto foi criado (*Unity* 2020.3), sendo agora necessário aceder diretamente ao ficheiro “manifest.json” do projeto, onde todos os *packages* utilizados no projeto em questão são referidos, juntamente com a respetiva versão. Após aceder ao ficheiro, é necessário adicionar “com.unity.barracuda”: “2.1.3” ao mesmo e voltar a abrir o projeto onde a aplicação está a ser criada.

4.1.5 *MediaPipeUnityPlugin* (0.11.0)

Um dos *packages* mais utilizados no projeto é essencial para permitir o funcionamento da rede neuronal no qual o projeto se baseia. Sem este *package* não seria possível criar uma aplicação que não necessite de qualquer elemento externo complementar para permitir o funcionamento da aplicação no dispositivo alvo. Outra razão que leva a este *package* apresentar um grau de importância maior do que os outros *packages* utilizados é o facto de, embora o *MediaPipe* não ser a única biblioteca que permite identificar *landmarks* do corpo humano, esta é das mais completas, apresentando um número consideravelmente maior de *landmarks* identificadas, e, mais importante ainda, foi a biblioteca utilizada aquando da elaboração da rede neuronal.

Relativamente ao processo de instalação do *package*, este foi realizado tendo em conta as características do dispositivo no qual seria utilizado, dando especial ênfase ao aspeto de desempenho da aplicação a criar. Considerando que se espera que a aplicação final seja capaz de correr no dispositivo *Hololens 2*, o *MediaPipeUnityPlugin* foi criado de maneira a ocupar o menor espaço possível e em conformidade com as instruções apresentadas no repositório *GitHub* para uso em *Windows 10* deste mesmo *plugin*.

Para construir o *plugin* em questão foi primeiro necessário instalar uma versão do *Python* maior ou igual a 3.9.0, uma versão do *Bazel* recente, uma versão do *NuGet* recente, a última versão do *MSYS2*, a versão 3.4.16 do *OpenCV*, por fim, a última versão do *NumPy*. Embora a versão de alguns dos componentes possa variar, é essencial utilizar esta versão específica de *OpenCV*, uma vez que caso não seja esta a versão utilizada, será necessário realizar várias alterações aos scripts utilizados para construir o *package*. Com todas as dependências necessárias instaladas, foi utilizado o comando “python build.py build –desktop cpu –opencv=cmake -v” para criar o *package* em questão. É essencial realçar a importância de garantir que as variáveis de ambiente do sistema para cada um dos componente que necessita destas se encontra presente, sendo recomendado a adição destas através do *GUI* do *Windows* e não através da linha de comandos.

4.1.6 *Microsoft Azure Cognitive Services* (*Speech Service*)

Devido às alterações sofridas aquando do desenvolvimento do projeto, foi necessário optar por uma alternativa à função de “Dictation” do *MRTK*. Para uso como serviço *Speech-to-Text* foi escolhido o *Cognitive Service* intitulado de “Speech Service”. Este serviço é capaz de reconhecer, traduzir e transcrever áudio para texto e suporta uma grande variedade de línguas, incluindo o português. Este serviço, ao contrário do *package MRTK*, suporta também a tradução do texto detetado para outra língua

desejada, sendo então possível traduzir outras línguas oralmente transmitidas (como Espanhol e Francês) para português em formato escrito.

O uso deste serviço é relativamente simples e o processo de instalação não é complicado, uma vez que envolve apenas o uso do *SpeechSDK* disponibilizado livremente, embora sejam necessário cumprir alguns requisitos para a instalação deste *unitypackage*.

O primeiro passo a tomar para utilizar este serviço é subscrever ao mesmo. Para tal, é necessário entrar na *Microsoft Azure* com uma conta *Microsoft* válida e criar uma nova instância deste serviço.

Após preencher todos os campos e criar a instância do serviço, o utilizador passa a ter acesso a duas chaves, uma das quais será utilizada na aplicação, sendo agora apenas necessário instalar o *SpeechSDK*.

O *SpeechSDK* pode ser instalado diretamente a partir do *website* oficial *Microsoft Learn*, sendo importante garantir que a versão instalada é a disponibilizada especificamente para *Unity*.

Com o ficheiro “SpeechSDK.unitypackage” resultante, basta importar este diretamente para o editor, após garantir que todos os ficheiros que o compõem estão seleccionados.

4.1.7 PyCharm (2022.3.1)

PyCharm é um *IDE (Integrated development environment)* de *Python* utilizado neste projeto para criar código em *python*. Devido à ocorrência de um problema aquando do desenvolvimento da aplicação, foi necessário utilizar esta ferramenta para criar um servidor *proxy* capaz de intermediar a conexão da aplicação com outro servidor.

A instalação deste *IDE* é realizada diretamente a partir do *website* oficial, sendo depois necessário reiniciar o dispositivo. É importante referir que a versão de *Python* utilizada num dado projeto é seleccionada aquando da criação desse mesmo projeto, sendo necessário existir uma versão de *Python* válida no dispositivo utilizado.

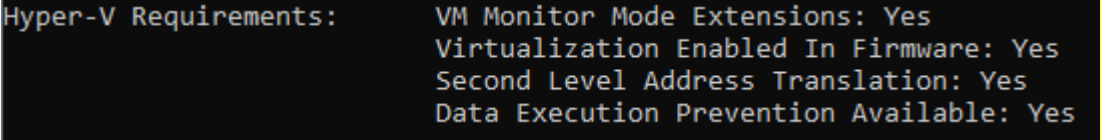
4.1.8 Postman

O *Postman* é uma plataforma de *software* para desenvolvimento de *APIs*, sendo utilizado neste projeto para testar o formato do pedido a enviar ao servidor adotado durante o desenvolvimento do mesmo, a resposta enviada pelo servidor já referido e mais importante ainda a conexão realizada com esse mesmo servidor, especialmente os protocolos e certificados utilizados. Para instalar esta ferramenta é apenas necessário aceder ao *website* oficial do *Postman* e realizar a instalação do mesmo através do *installer*.

4.1.9 HoloLens 2 Emulator (10.0.20348.1535)

De maneira a testar inicialmente a aplicação no dispositivo alvo, foi necessário utilizar um emulador. Para instalar este emulador é primeiro necessário garantir que o dispositivo utilizado apresenta as características desejadas, sendo estas: um CPU de 64 bits com quatro ou mais cores, 8 GB de RAM ou mais, suportar as funcionalidades “Hardware-assisted virtualization”, “Second Level Address Translation (SLAT)” e “Hardware-based Data Execution Prevention (DEP)” e apresentar uma placa gráfica com *DirectX 11* ou superior e *WDDM 2.5 graphics driver* (embora versões mais antigas do *graphics driver* possam funcionar, apresentando uma performance inferior ao esperado). Por último, é necessário o *CPU* suportar *Hyper-V*. Relativamente ao equipamento utilizado para emular o dispositivo, este cumpre quase todas as condições, estando em falta a versão do *graphics driver* recomendado e o suporte oficial do *Hyper-V* (embora seja possível adicioná-lo). O dispositivo atualmente não suporta *Hyper-V* devido à versão do *Windows* atualmente a ser utilizado (*Windows 10 Home*), sendo necessário uma versão de *Windows* mais completa ou, como realizado neste caso, a instalação desta funcionalidade não suportada oficialmente.

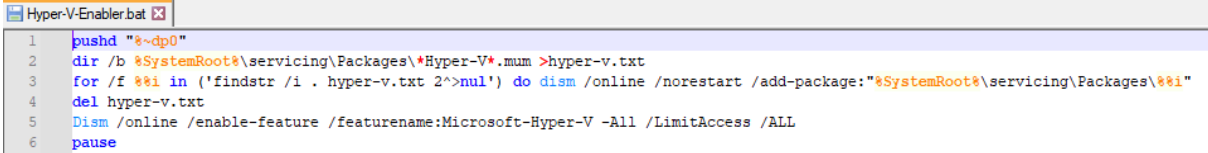
Para primeiro averiguar se é possível ou não instalar esta funcionalidade no dispositivo desejado é necessário, a partir da linha de comandos, utilizar o comando “systeminfo” para verificar se todos os quatro requisitos para *Hyper-V* estão presentes no dispositivo, como demonstrado pela Figura 11.



```
Hyper-V Requirements:      VM Monitor Mode Extensions: Yes
                           Virtualization Enabled In Firmware: Yes
                           Second Level Address Translation: Yes
                           Data Execution Prevention Available: Yes
```

Figura 11: Requisitos *Hyper-V*

Estando confirmada a possibilidade de utilizar a funcionalidade *Hyper-V* é então necessário criar um ficheiro *bash* (neste caso denominado de “Hyper-V-Enabler.bat”) com um pequeno *script*, *script* este demonstrado na Figura 12.



```
Hyper-V-Enabler.bat
1 pushd "%~dp0"
2 dir /b %SystemRoot%\servicing\Packages\*Hyper-V*.mum >hyper-v.txt
3 for /f %i in ('findstr /i . hyper-v.txt 2^>nul') do dism /online /norestart /add-package:"%SystemRoot%\servicing\Packages\%i"
4 del hyper-v.txt
5 Dism /online /enable-feature /featurename:Microsoft-Hyper-V -All /LimitAccess /ALL
6 pause
```

Figura 12: Script para instalar *Hyper-V*

Após criado o ficheiro, é necessário correr este como administrador e reiniciar o dispositivo de maneira a guardar as alterações. Após recomeçar o dispositivo, a aplicação *Hyper-V Manager* passa a estar disponível, sendo agora possível utilizar o *Hyper-V*.

Relativamente ao Emulador em si, este pode ser instalado diretamente a partir do *website* oficial.

4.1.10 *Connect*

Connect é uma aplicação criada para *Windows PC* e tem como propósito projetar o ecrã de outros dispositivos *Windows* no ecrã do *Windows PC* utilizado. Esta aplicação foi utilizada para permitir a partilha da aplicação criada para *Hololens 2* através do computador e para facilitar a captura de imagens do funcionamento desta mesma aplicação.

Para instalar a aplicação, é necessário aceder à secção “App”, dentro das “Settings” do computador, e seleccionar a opção “Optional features”. Após seleccionar esta última opção é necessário procurar por “Wireless Display” e instalar essa mesma *feature*. Após instalada, a aplicação *Connect* passa a estar presente no sistema.

4.1.11 *Windows Device Portal*

O *Windows Device Portal* é um servidor *web* criado por um dado dispositivo *Windows* (neste caso, pelo *Hololens 2*), ao qual outro dispositivo *Windows* se pode conectar (através do *browser*) para configurar ou gerir o dispositivo que criou o servidor. Este servidor foi utilizado com o intuito de poder analisar os ficheiros “UnityPlayer.log” criados pela aplicação quando esta corre no dispositivo *Hololens 2*.

Para utilizar esta ferramenta, é necessário ativar os modos “Developer Mode” e “Device Portal” do dispositivo a disponibilizar o servidor. Uma vez que o dispositivo *Hololens 2* não se encontra conectado por *USB* com o dispositivo que pretende aceder ao servidor criado, é necessário copiar o endereço *IPv4* do dispositivo *Hololens* e copiar esse mesmo endereço para o *browser* do dispositivo que pretende aceder ao servidor.

Na primeira vez que esta conexão é realizada é necessário inserir um *PIV* de seis dígitos apresentados no dispositivo *Hololens*, sendo em seguida necessário criar um “user name” e uma “password” para aceder ao servidor. Para usos futuros, estes dados serão necessários para entrar no *Device Portal*.

4.1.12 *Dependencies* (1.11)

Dependencies é uma aplicação criada com base no *software legacy* “Dependency Walker”. Esta aplicação utilizada no projeto foi criada a partir deste *software* com o intuito de realizar a mesma tarefa, sendo esta representar as dependências de um dado módulo (neste caso, “mediapipe_c.dll”). Esta aplicação foi utilizada no projeto com o intuito de analisar as dependências que o *MediaPipeUnityPlugin*

necessitaria para ser utilizado num dado dispositivo. Esta aplicação foi escolhida uma vez ser a recomendada pelo *package* em questão aquando da existência de problemas na deteção/reconhecimento de dependências do ficheiro em questão por parte do dispositivo utilizado. A instalação desta aplicação é realizada a partir do repositório *GitHub* com o mesmo nome, sendo este “Dependencies”. Para utilizar a aplicação é apenas necessário instalar a última versão disponibilizada e descomprimir o arquivo descarregado.

4.2 Aplicação Inicialmente Desenvolvida

Nesta secção o processo de desenvolvimento da aplicação inicialmente criada é documentado. Esta primeira aplicação, composta por três interfaces diferentes, foi criada com base na idealização inicialmente realizada e que adotava o uso da função “Dictation” para a funcionalidade *Speech-to-Text* de português transmitido oralmente para português escrito e adotava o uso da rede neuronal diretamente no *Unity* através de dois *packages* oficiais para a classificação dos sinais detetados.

A partir do diagrama representado na Figura 13 é possível visualizar as diferentes funções que se esperam encontrar em cada uma das interfaces a serem criadas, juntamente com o *package*/ferramenta que deve ser utilizado para criar as mesmas.

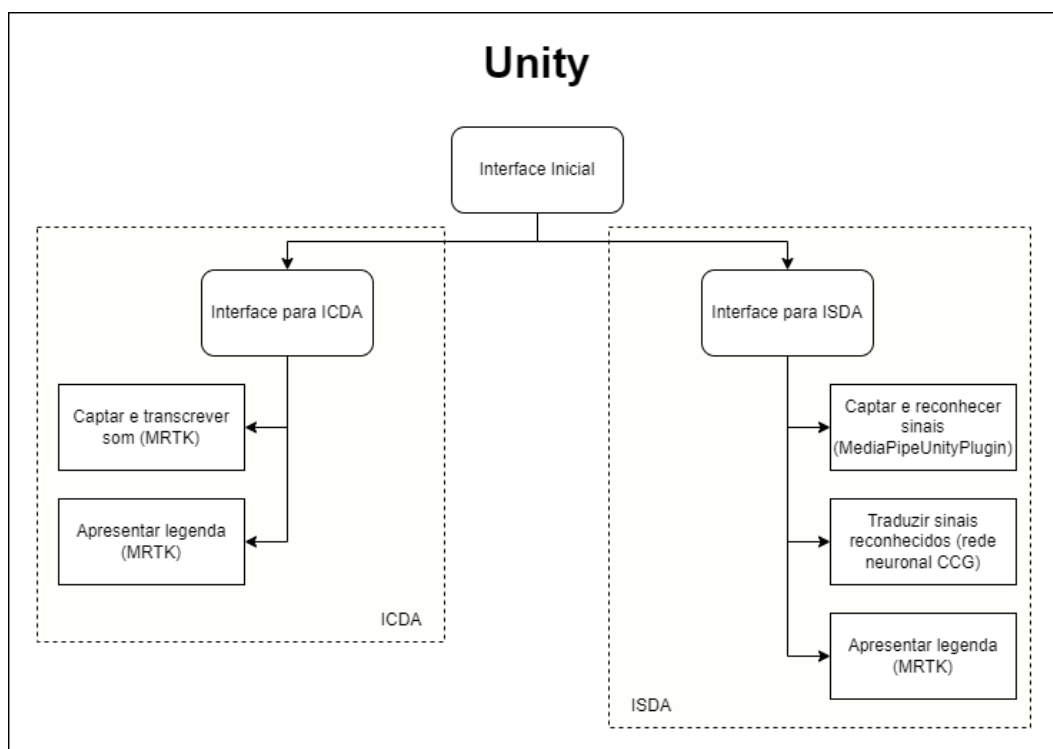


Figura 13: Diagrama inicial da solução

Infelizmente, por razões de incompatibilidade explicadas em detalhes nas secções seguintes deste capítulo, não foi possível adotar ambas as opções inicialmente escolhidas, ou seja, a função “Dictation” e a incorporação da rede diretamente no *Unity*, tendo sido necessário desenvolver uma nova aplicação com as mesmas funcionalidades, mas utilizando métodos e *packages* substancialmente diferentes.

Apesar desta aplicação inicialmente idealizada não corresponder exatamente à aplicação final desenvolvida, todos os objetos criados durante o desenvolvimento desta foram utilizados na aplicação final, à exceção dos desenvolvidos para a pequena interface introdutória, uma vez que esta secção deixou de existir, algo também explicado em secções futuras.

Por fim, a descrição do desenvolvimento desta aplicação é importante porque foi com base nesta que a maioria dos elementos visuais foram desenvolvidos e o desenvolvimento desta aplicação serve como um ponto inicial fulcral para melhor compreender as diversas dificuldades expostas na próxima secção que levaram a alterações consideráveis na forma como a aplicação funciona.

Antes de descrever o processo individual de cada interface, ou, mais concretamente, de cada cena criada para cada interface, é importante descrever os diferentes objetos adicionados em todas as cenas criadas ao longo do projeto, dando especial ênfase ao objeto relacionado com a adição e modificação de *MRTK Profiles*.

4.2.1 *Objetos MRTK*

Antes de descrever as cenas criadas para cada uma das interfaces idealizadas, é importante descrever o processo, algo peculiar, utilizado para incorporar o *package MRTK* nestas mesmas cenas criadas. O primeiro passo para incorporar este *package* nas cenas criadas passa por, durante o processo de configuração, instalar o *package* extra *TMP Essentials*. Este *package* é necessário porque fornece objetos que permitem apresentar o texto da aplicação em formato 3D, permitem escalar este mesmo texto de maneira a ser visível a maiores distâncias e são mais otimizados para desempenho do que os objetos padrão do *Unity*.

Após adicionar o *package* referido, foram utilizados os objetos presentes em duas das cenas exemplo fornecidas, “DefaultLightingScene” e “DefaultManagerScene”, para criar uma nova cena, “1.Start”. Esta nova cena foi criada para servir de base às três cenas inicialmente idealizadas. Ao criar esta nova cena, foi utilizado o objeto “Directional Light”, para adicionar uma fonte de luz à cena em questão, o objeto “MixedRealityToolkit”, para selecionar o “Profile” que o *package* deve adotar durante o correr da cena corrente, e o objeto “MixedRealityPlayspace”, para adicionar o objeto “Camera” responsável pelo *rendering* da cena. Para além destes objetos pertencentes às cenas exemplo, também foi adicionado um

objeto normalmente presente no desenvolvimento de aplicações com *MRTK*, “MixedRealitySceneContent”, utilizado para organizar todos os restantes objetos adicionadas à cena criada.

Todas as cenas criadas neste projeto *Unity* apresentam a mesma estrutura fundamental que a cena “1.Start”, sendo esta a representada na Figura 14.

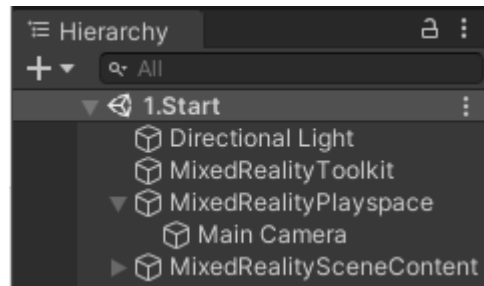


Figura 14: Estrutura fundamental das cenas criadas

O segundo objeto, “MixedRealityToolkit”, é de longo o mais importante, uma vez que este é responsável pela definição de todos os diferentes Perfis utilizados na cena onde este se encontra. O *package MRTK* utiliza Perfis para definir diferentes aspetos da cena onde este é inserido, como por exemplo a adição de ferramentas de diagnóstico ou alterar as configurações do objeto “Camera” utilizado na cena. De uma maneira geral, existe um Perfil principal, normalmente denominado de “Configuration Profile”, utilizado para definir aspetos mais básicos como a presença ou não de ferramentas de diagnóstico ou se o “Teleport System” se encontra ativo na cena atual, e existem outros Perfis, como por exemplo “Camera Profile” e “Speech Profile”, que apenas permitem editar aspetos mais específicos e detalhados. Para criar ou editar qualquer um dos Perfis do *package* utilizado é necessário primeiro clonar o mesmo e apenas nesse clone podem ser realizadas alterações.

Neste projeto, apenas dois “Configuration Profiles” foram criados, sendo o primeiro utilizado em cenas onde a função “Dictation” ou “Speech” sejam utilizadas e o segundo em cenas onde nenhuma destas funções sejam utilizadas.

Para o primeiro Perfil, quando a função “Dictation” é utilizada foi necessário criar um novo “Input Profile” com o *Data Provider* “Windows Dictation Input”, enquanto que quando a função “Speech” é utilizada foi necessário criar um novo “Input Profile” com o *Data Provider* “Windows Speech Input” e um novo “Speech Commands Profile” com os *Speech Commands* a serem detetados pela aplicação.

Relativamente ao objeto “MixedRealitySceneContent” criado, em todas as cenas onde este foi utilizado, foi necessário adicionar o componente com o mesmo nome, “MixedRealitySceneContent”. Este componente é essencial porque permite alinhar os objetos associados ao objeto

“MixedRealitySceneContent” com a altura da cabeça ou postura da cabeça, o que corresponde ao comportamento ideal no dispositivo alvo.

No que toca aos objetos associados ao objeto “MixedRealitySceneContent”, muitos destes foram criados com base em *prefabs* provenientes do *MRTK* para criar a interface da aplicação. Para poder encontrar e adicionar estes *prefabs*, foi instalada a ferramenta *MRTK Toolbox*. Infelizmente, esta ferramenta não apresenta todos os *prefabs* disponibilizados pelo *package*, sendo recomendado procurar diretamente na lista de ficheiros presente no projeto. É também importante referir que uma vez que não foi utilizada esta ferramenta para adicionar os *prefabs*, foi necessário selecionar a opção “Unpack Completely” para cada conjunto de objetos, de maneira a poderem ser editados livremente.

Por fim, cada botão proveniente do *package MRTK* apresenta dois componentes que podem ser utilizados para adicionar novos *event handlers* ao evento “OnClick”, sendo estes “Interactable” e “ButtonConfigHelper”. Independentemente do *event handler* criado, este pode ser adicionado a qualquer um dos componentes, nunca sendo necessário estar presente em ambos.

4.2.2 Interface Principal (Inicial)

Inicialmente, a estrutura idealizada da aplicação a ser criada era composta por três interfaces diferentes, sendo a primeira destas denominada de “Interface Principal”. Esta interface tinha como função apresentar ao utilizador, mal este abra a aplicação, as duas restantes interfaces da aplicação, sendo este processo realizado através de uma pequena descrição das respetivas funcionalidades presentes nestas interfaces. Juntamente com a descrição das interfaces, esta Interface Inicial também permitiria ao utilizador selecionar uma destas duas restantes interfaces e, consequentemente, abrir essa mesma interface.

Esta abordagem foi inicialmente idealizada desta maneira com o intuito de não só facilitar o processo de seleção da interface a ser usada pelo utilizador, mas de também garantir que nenhum dos dois tipos distintos de utilizador se depara com um maior grau de dificuldade para encontrar a interface que melhor o auxilia. O facto de ambas as interfaces apresentarem necessidades bastantes diferentes também foi um fator que levou a divisão das duas.

Inicialmente, foi então criada a Interface Inicial, a partir dos objetos *MRTK* referidos na subsecção anterior, sendo esta intitulada de “1.Start”. Esta cena, comparativamente às cenas futuramente descritas, é relativamente simples, sendo composta por apenas dois objetos para além dos presentes em todas as cenas criadas, “SpeechManager” e “MainMenu”. Na Figura 15 é possível observar a estrutura hierárquica da cena em questão.



Figura 15: Estrutura da Interface Inicial

O objeto “SpeechManager” é utilizado nesta cena com o intuito de adicionar o *script* “SpeechInputHandler” à cena em questão. Este script provém do *MRTK* e é utilizado para definir o comportamento que se espera que a aplicação adote caso uma dada palavra seja mencionada durante o correr da aplicação em questão. Esta funcionalidade é apenas possível através do uso da função “Speech” e foi adicionada com o intuito de testar a forma como esta mesma função poderia ser adotada para funcionar em objetos que não botões. Neste projeto, foi adotado este método para criar uma forma alternativa de selecionar a interface destinada a ISDA, sendo este método a referência da palavra “Select” durante o correr da aplicação, como pode ser visualizado na Figura 16.

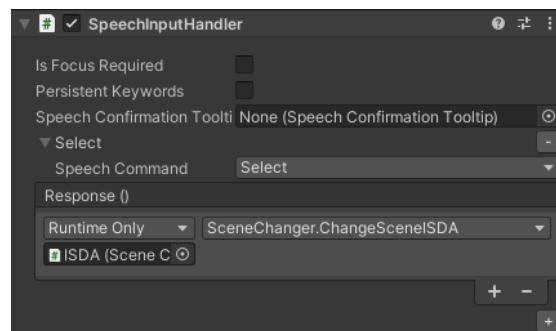


Figura 16: Função “Speech” Interface Inicial

Relativamente ao objeto “MainMenu”, este foi criado com base num *prefab* do *MRTK* e compõe a estrutura toda da Interface Principal desenvolvida. Esta interface é quase idêntica à inicialmente idealizada, sendo esta um painel retangular composto por um Título, por uma Descrição e por dois botões, um relativo à interface para ICDA e outro relativo à interface para ISDA.

No caso destes três elementos da interface, as alterações realizadas comparativamente ao *prefab* utilizado foram simples, tendo sido alterado o texto apresentado na aplicação, desativado o botão central e a legenda “SeeltSayltLabel” do botão “ICDA” e por fim foi adicionado o script “ChangeScene” como componente aos botões “ICDA” e “ISDA”, juntamente com um novo *event handler*.

Na Figura 17 é possível visualizar a última versão da Interface Inicial criada.

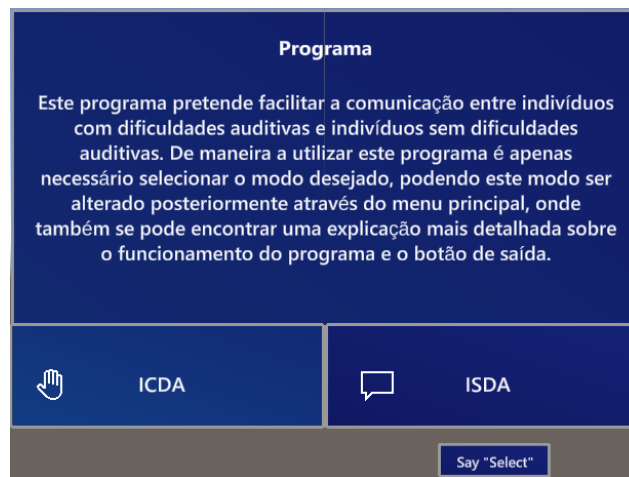


Figura 17: Interface Inicial criada

Para resumir, a Interface Inicial apresenta tanto a estética como as funcionalidades que se esperava que esta tivesse quando foi idealizada, ou seja, que fosse capaz de permitir ao utilizador selecionar a interface que deseja utilizar. Para além destas funcionalidades, também foi adicionada o uso da função “Speech”, algo que inicialmente seria apenas adicionado à interface ISDA.

4.2.3 Interface ICDA (Inicial)

Antes de começar a descrição da interface ICDA criada, é importante relembrar que o “Configuration Profile” utilizado nesta cena é diferente do utilizado nas outras duas e não apresenta nenhum “Speech Command” ativo. Uma vez que a função “Speech” não é utilizada nesta cena, devido à natureza da mesma. Esta é a maneira mais fácil de remover o conteúdo do campo “Voice Command” presente em todos os botões da cena sem ter de os remover um a um.

Inicialmente, a estrutura idealizada da aplicação a ser criada era composta por três interfaces diferentes, sendo a segunda destas denominada de “Interface ICDA”. Esta interface, destinada a indivíduos com dificuldades auditivas, tinha como função transcrever língua portuguesa transmitida oralmente, proveniente de um indivíduo sem dificuldades auditivas e que pretenda comunicar com o utilizador desta interface, para língua portuguesa em formato de texto. Inicialmente também foi definido que a deteção e transcrição da língua divulgada oralmente deve ser feita pela função “Dictation” proveniente do *package MRTK*.

Em concordância com a estrutura base de todas as cenas criadas neste projeto, foi criado a cena “2.ICDA FINAL”, que pretende tornar realidade a interface ICDA inicialmente idealizada. Tal como as restantes cenas do projeto, esta apresenta também os quatro objetos *standard* de todas as cenas criadas para esta aplicação. Relativamente ao conteúdo da cena em si, tanto estético como funcional, este é criado a

partir dos objetos associados apenas ao “MixedRealitySceneContent”, sendo apenas relevante falar destes mesmos.

Para o desenvolvimento estético e funcional desta interface, foi necessário adicionar a esta cena um objeto capaz de representar o texto resultante do uso da função “Dictation”, um menu capaz de permitir ao utilizador ativar a função “Dictation”, mudar de cena, abrir o menu de ajuda e sair da aplicação e dois menus, um de ajuda e um para verificar a vontade do utilizador de sair da aplicação.

Para apresentar o resultado em formato de texto foi adicionado o objeto “Canvas”, para criar o menu com as quatro funcionalidades referidas foi adicionado o objeto “NearMenu” e para os dois menus foram adicionados os objetos “Ajuda” e “Sair”.

A estrutura da cena criada pode ser visualizada na Figura 18, excluindo os objetos comuns, utilizados em todas as cenas.

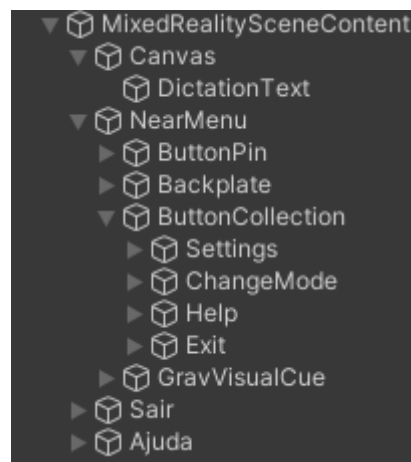


Figura 18: Estrutura da Interface ICDA

Comparativamente à Interface Principal, é importante reforçar que esta nova cena já não apresenta nenhum objeto denominado “SpeechManager”, uma vez que o recurso “Speech” já não é utilizado globalmente para aceder à Interface ISDA.

Relativamente ao primeiro objeto adicionado, “Canvas”, uma vez que a aplicação desenvolvida está a ser criada para o dispositivo *Hololens 2* através do *MRTK*, é essencial garantir que o conteúdo a apresentar nesta cena, sem contar com os menus, seja criado dentro de um objeto do tipo “MRTK Canvas”. Se o conteúdo a ser transcrito não se encontrar dentro de um *canvas* deste tipo, não vai apresentar o comportamento que se espera apresentar numa aplicação *Hololens*.

Para criar o objeto “Canvas” foi primeiro necessário adicionar um objeto *canvas* novo à cena e converter este para “MRTK Canvas”. Este processo é relativamente simples, sendo apenas necessário seleccionar o *canvas* adicionado e seleccionar a opção “Convert to MRTK Canvas”, sendo os quatros passos que compõem este mesmo processo de conversão realizados automaticamente.

No que toca à forma como este objeto adicionada o conteúdo detetado obtido pela função “Dictation”, este é realizado através do objeto “DictationText”. Este objeto, apesar de ser do tipo *Text-TMP*, é o elemento mais importante desta cena, uma vez que é aqui que o sistema “Dictation” é utilizado. Considerando que o “Configuration Profile” utilizado nesta cena apresenta o *Data Provider* do sistema “Dictation”, é apenas necessário adicionar o *script* “DictationHandler” ao objeto de texto referido. Para a função funcionar como inicialmente esperado, foi necessário remover a opção “Start Recording On Start”, de maneira que a função “Dictation” não comece mal a cena é aberta, e foi necessário adicionar quatro *event handlers* com o método “TextMeshProUGUI.text”, para mudar o texto apresentado no objeto “DictationText”. O componente adicionado pode ser visualizado na Figura 19.

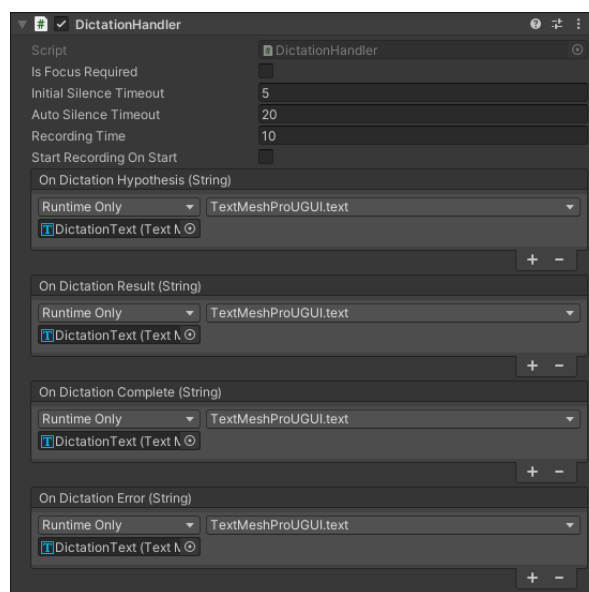


Figura 19: Componente “Dictation Handler”

Para começar a deteção do áudio a partir da função “Dictation”, uma vez que a opção “Start Recording on Start” não se encontra selecionada, foi utilizado um dos quatro botões para começar o processo de gravação. O botão utilizado, “Settings”, apresenta como única função iniciar a gravação áudio através do método “StartRecording”. Relativamente à forma como esta tradução é parada, o processo é ainda mais simples, sendo apenas necessário deixar de fornecer áudio à função durante um certo tempo. Caso nenhum áudio seja detetado nos primeiros cinco segundos (*initialSilenceTimeout* = 5f), a função termina. Caso passem vinte segundos sem receber áudio, considerando que a função foi pelo menos utilizada nos primeiros cinco segundos iniciais (*autoSilenceTimeout* = 20f), a função termina.

Com um dos três objeto adicionados referidos, passa-se para o segundo, “NearMenu”, onde podem se encontrados os quatro botões utilizados para interagir com esta mesma interface. Para cada um dos quatro botões, em termos estéticos, foram alterados os nomes apresentados na aplicação, foi adicionado

um ícone característico a cada um e foram removidos os objetos “SeeltSayltLabel”, por razões previamente apresentadas.

O primeiro botão, “Settings”, é responsável por começar o processo de gravação do serviço “Dictation”, tendo sido adicionado um novo *event handler* com o método “StartRecording”, como já referido.

O segundo botão, “ChangeMode”, é utilizado para mudar a cena em que o utilizador atualmente se encontra para a outra cena existente, ou, por outras palavras, mudar da interface atualmente visualizada para a outra interface existente. Neste caso, a mudança seria passar da interface para ICDA para a interface para ISDA. As alterações realizadas neste botão são muito semelhantes às realizadas nos botões pertencentes à Interface Principal inicialmente idealizada, uma vez que apresentam o mesmo propósito. Foi então necessário adicionar o componente “SceneChanger” e um novo *event handler* com as mesmas condições do *event handler* criado na Interface Principal.

O terceiro botão, “Help”, é utilizado para abrir o menu de ajuda. Relativamente ao objeto “Help” em si, é apenas necessário saber que, à semelhança do objeto “ChangeMode”, apenas foi adicionado o componente “HelpPop” e foi adicionado um novo *event handler* capaz de abrir o menu em questão e esconder o resto da interface.

O quarto e último botão, “Exit”, é utilizado para abrir o menu de confirmação de saída. As alterações realizadas ao objeto original são muito semelhantes às alterações realizadas nos dois últimos botões, tendo sido mais uma vez adicionado um novo componente, “Quit”, e tendo sido adicionado um novo *event handler* relativo ao componente adicionado. Tal como no caso do botão “Help”, o método utilizado no *event handler* apenas torna visível o menu de saída e esconde o resto da interface.

O próximo objeto encontrado, seguindo a ordem da hierarquia, denomina-se “Sair” e corresponde ao menu de saída. As alterações realizadas neste objeto foram extremamente semelhantes às realizadas ao objeto da Interface Principal, sem considerar o uso da função “Speech”.

Relativamente às alterações estéticas realizadas, foi alterado o título, a descrição e os nomes dos botões, para além da desativação dos componentes “SeeltSayltLabel” e do objeto “ButtonOne”.

Relativamente às alterações funcionais, foi adicionado o mesmo componente a ambos os botões, sendo este o script “Quit”, juntamente com um *event handler* diferente para cada botão. Para o botão “LeaveFinal” foi criado um *event handler* com o método de sair da aplicação e para o botão “Stay” foi criado um *event handler* com o método de fechar o menu de saída e reativar o resto da interface.

Para finalizar, o último objeto, “Ajuda”, corresponde ao menu de ajuda e tem como finalidade explicar ao utilizador o funcionamento da interface onde atualmente se encontra, neste caso, da Interface ICDA.

As alterações estéticas realizadas neste objeto, igual às alterações estéticas realizadas no objeto “Sair”, foram a mudança do título e da descrição, a remoção dos objetos “SeeltSayltLabel” e a desativação de dois dos três botões provenientes do *prefab*. Ao contrário da Interface Principal, neste caso foram desativados os dois botões mais pequenos e utilizado o maior.

Relativamente às alterações funcionais, foi adicionado um novo componente e foi adicionado um novo *event handler*. O método “ClosePop” utilizado no *event handler* desativa o painel de ajuda e ativa os restantes elementos da interface.

Uma vez que tanto o menu de saída como o menu de ajuda foram utilizados em todas as cenas referentes a um dos dois tipos de utilizador, não serão representando nesta secção. Contudo é relevante mostrar a interface que o utilizador consegue visualizar mal abre esta cena, sendo esta representada na Figura 20.

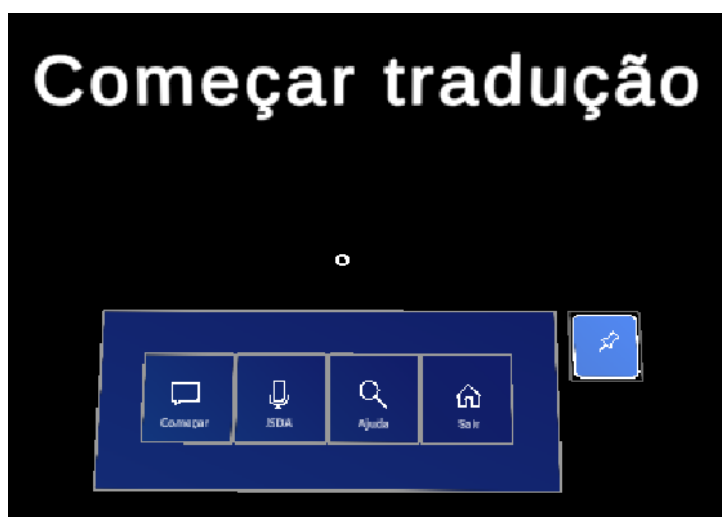


Figura 20: Interface ICDA (inicial)

4.2.4 Interface ISDA (Inicial)

Inicialmente, a estrutura idealizada da aplicação a ser criada era composta por três interfaces diferentes, sendo a terceira destas denominada de “Interface ISDA”. Esta interface, destinada a indivíduos sem dificuldades auditivas, tinha como função captar, detetar e classificar sinais provenientes do uso de LGP como meio de comunicação e apresentar o texto resultante dessa mesma classificação, em língua portuguesa. Inicialmente, foi definido que a deteção dos sinais e respetivos pontos deveria ser realizada pelo *package* não oficial *MediaPipeUnityPlugin* e que a classificação desses mesmos pontos deveria ser realizada pela rede neuronal, diretamente no *Unity*.

No que toca ao desenvolvimento desta interface, a cena criada para esta interface denomina-se “3.ISDA”. A estrutura desta cena é extremamente semelhante à estrutura da cena “2.ICDA”, correspondente à interface ICDA. De uma maneira geral, os objetos listados em cada uma são os mesmos, salvo duas

exceções. A primeira exceção é a presença do objeto “Screen”, utilizado para representar a imagem captura pela câmara do dispositivo. A segunda é a composição do objeto “NearMenu”, que passa a ter apenas três botões em vez de quatro. Esta mudança deve-se ao facto da deteção de *landmarks* nesta cena começar mal o utilizador entra na mesma, não havendo assim a necessidade de um botão capaz de começar ou parar o processo de deteção, como acontece na cena “2.ICDA”. Uma vez que um dos botões já não são precisos, foram utilizados apenas três botões, um para mudar de cena, “ChangeMode”, um para abrir o menu de ajuda, “Help”, e outro para abrir o menu de saída, “Exit”. Na Figura 21 é possível visualizar a estrutura desta interface.

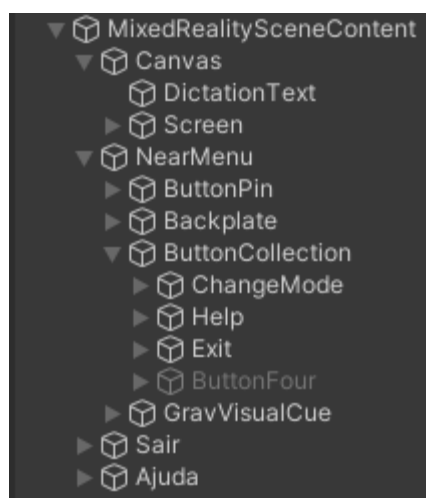


Figura 21: Estrutura da Interface ISDA

Esta interface, por razões apresentadas na próxima secção, nunca foi finalizada em termos funcionais, mas todo o conteúdo apresentado nesta secção foi utilizado no desenvolvimento da aplicação final, à exceção da função “Speech”, algo também referido na próxima secção.

Relativamente ao primeiro objeto adicionado, “Screen”, este foi adicionado com o intuito de representar as *landmarks* identificadas pelo *package MediaPipeUnityPlugin*. Para tal, foi primeiro preciso criar um *script*, “TesteHands”, capaz de detetar as mãos e respetivas *landmarks*, de desenhar essas mesmas mãos a partir das *landmarks* identificadas e, mais importante ainda, capaz de guardar as coordenadas X e Y de cada *landmark* detetada para estas poderem ser enviadas para a rede neuronal. A forma como este processo é realizado é algo extensa e, uma vez que este *script* sofre alterações significativas ao longo do desenvolvimento do projeto, uma descrição mais detalhada pode ser encontrada nas secções seguintes.

Relativamente à rede neuronal, o processo de importação e uso é extremamente simples, sendo apenas necessário adicionar o ficheiro em questão à pasta “Assets” e criar um *script* capaz de fornecer à rede neuronal o *input* que esta necessita para gerar um *output*, sendo esta neste caso “TesteHands”. Ao

correr a rede, o resultado final consistirá no *index* da palavra com o maior *confidence score*, ou seja, a palavra que a rede reconhece como sendo a mais provável. Com o *index* obtido, é apenas necessário percorrer o ficheiro de texto associado à rede e transcrever a palavra que se encontra nesse *index*.

Nesta cena também foi utilizada a função do *MRTK* denominada “Speech”. Esta função pode ser utilizada imediatamente por alguns objetos provenientes do mesmo *package* que esta, desde que este apresente o componente “Interactable” no Inspetor. Esta função permite ao utilizador realizar ações através de comandos de voz, não sendo necessário interagir manualmente com a aplicação. Para ativar a função “Speech” basta atribuir um dos “Speech Commands” criados no início do projeto ao campo “Voice Command” do componente referido e selecionar a opção “Requires Focus”, para ser necessário o utilizador olhar diretamente para o objeto com quem pretende interagir. Nesta aplicação também foram alteradas as legendas “SeeltSayItLabels” de maneira a corresponderem à palavra que deve ser pronunciada para uma dada ação ocorrer.

Para finalizar, no que toca às alterações estéticas adicionais realizadas nesta cena, foi realizada a mudança dos nomes e símbolos dos três botões principais, foi adicionada uma descrição algo extensa no menu de ajuda relativa à forma como a interface em questão pode ser utilizada e foram ativadas e editadas as legendas “SeeltSayItLabel” para representarem a função “Speech”. Relativamente às alterações funcionais, foi adicionado o ficheiro de texto intitulado “Output” correspondente ao output da rede neuronal em formato *.onnx*, foi adicionado o *script* “TesteHands” ao objeto “MixedRealitySceneContent”, foram adicionados os respetivos componentes e *event handlers* já referidos na cena anteriormente apresentada e substituídos os “Speech Commands” de acordo com o botão a qual estavam associados.

A cena criada para esta interface pode ser visualizado na Figura 22.



Figura 22: Interface ISDA (inicial)

4.3 Complicações Iniciais e Alternativas Adotadas

Infelizmente, foram inúmeros os problemas identificados imediatamente após a idealização da aplicação e definição das diferentes ferramentas a serem utilizadas aquando da criação das funcionalidades dessa mesma aplicação.

As seguintes complicações enumeradas em muito influenciaram o projeto e a forma como se esperava que este funcionasse, o que levou a mudanças drásticas sobretudo na forma como a rede neuronal fornecida pelo CCG foi implementada no projeto.

4.3.1 Rede Neuronal - Complicação

Anteriormente ao início do desenvolvimento da aplicação em questão, a rede neuronal fornecida pelo Centro de Computação Gráfica (CCG) ainda se encontrava em desenvolvimento, não sendo possível testar a forma como se esperava incorporar a mesma na aplicação a ser criada, embora o método para realizar tal feito teria sempre de passar pelo uso do *package ML-Agents*, mais concretamente do *package Barracuda*, que, no caso inicialmente analisado, seria utilizado pelo *package ML-Agents* para realizar inferência na rede neuronal e, consequentemente, permitir “traduzir”, a partir de um conjunto de imagens, os sinais apresentados em LGP.

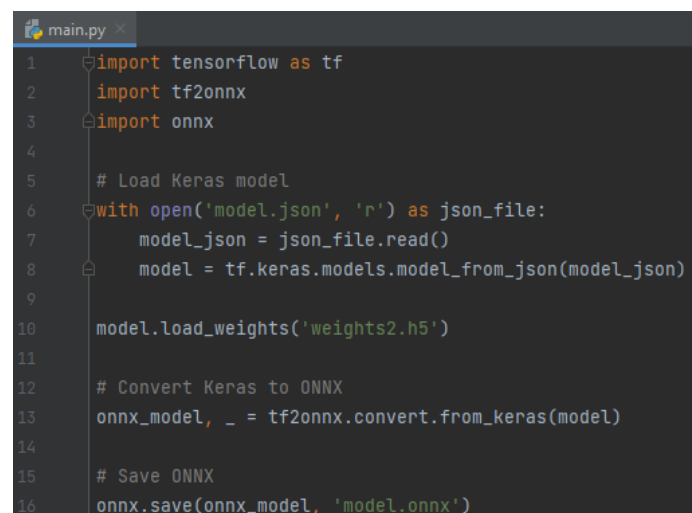
Durante o desenvolvimento da solução, foi recebida uma rede neuronal inicial que, embora não fosse o resultado final e ainda fosse alvo de mudanças, já apresentava a estrutura final da rede e alguns possíveis resultados, resultados estes que, regra geral, correspondiam a uma única palavra.

A rede neuronal recebida, apesar de poder ser alvo de alterações, já se encontrava no formato final, sendo este .h5, ou *Hierarchical Data Format version 5 (HDF5)*. Este formato é normalmente utilizado para guardar redes neuronais treinadas usando as *frameworks TensorFlow* ou *Keras*, sendo caracteristicamente capazes de armazenar aspetos relevantes como os *weights* e *biases* da rede. Infelizmente, o *Unity* não é capaz de utilizar redes neuronais no formato .h5. Na verdade, com apoio dos *packages* oficiais do *Unity*, este é apenas capaz de importar e utilizar redes neuronais no formato .onnx (*ONNX*).

Antes de testar o *package ML-Agents*, foi necessário converter a rede neuronal em formato .h5 para .onnx, de maneira a esta poder ser utilizada diretamente no *Unity*. Para tal, foi criado, através do uso do *IDE PyCharm*, um pequeno script em *python* capaz de converter a rede em questão para o formato .onnx. Infelizmente, o processo de conversão não foi bem sucedido, sendo necessário mais ficheiros para além do ficheiro .h5 final.

Uma vez que não foi possível converter a rede neuronal em formato *HDF5* diretamente em formato *ONNX*, foi necessário dividir a rede neuronal em três, sendo esta agora composta pelo ficheiro “config.json”, que serve a função de especificar a arquitetura da rede, pelo ficheiro “model.json”, que contém o conjunto de classes que o modelo pode prever e, por fim, pelo ficheiro “weights.127-0.02.h5” (ficheiro que acaba por ser renomeado para “weights2.h5”), que contém os pesos da rede neuronal, carregados após o carregamento do “config.json”. Com a rede agora dividida em três é possível tentar a conversão para o formato *ONNX*.

De maneira a realizar a conversão da rede dividida em três, foi criado mais um *script* em *python*, através do *PyCharm*, capaz de pegar nos ficheiros “model.json” e “weights2.h5” para criar o ficheiro “model.onnx”. Na Figura 23 é possível visualizar o script criado.



```
1 import tensorflow as tf
2 import tf2onnx
3 import onnx
4
5 # Load Keras model
6 with open('model.json', 'r') as json_file:
7     model_json = json_file.read()
8     model = tf.keras.models.model_from_json(model_json)
9
10 model.load_weights('weights2.h5')
11
12 # Convert Keras to ONNX
13 onnx_model, _ = tf2onnx.convert.from_keras(model)
14
15 # Save ONNX
16 onnx.save(onnx_model, 'model.onnx')
```

Figura 23: Criação do ficheiro *ONNX*

Como se pode constatar, este script utiliza três *packages* distintos, sendo o primeiro e mais reconhecível o *tensorflow*, utilizado neste script para carregar o modelo *Keras* a partir do *JSON* e do ficheiro dos pesos (weights.h5). Após carregado o modelo, este é convertido para *ONNX*, através do *package tf2onnx*. Após convertido, é utilizado o *package onnx* para guardar o ficheiro criado com o nome “model.onnx”. Na Figura 24 é possível visualizar a pasta do projeto *PyCharm*, onde o ficheiro criado se encontra, juntamente com os ficheiros que o criaram.

Utilizador > PycharmProjects > JSONtoONNX

Search JSONtoONNX

Name	Date modified	Type	Size
.idea	10/11/2023 1:56 AM	File folder	
venv	5/25/2023 6:16 PM	File folder	
config	5/15/2023 5:27 PM	JSON File	1 KB
main	10/11/2023 2:18 AM	Python File	1 KB
model	5/15/2023 5:27 PM	JSON File	3 KB
model.onnx	10/11/2023 2:22 AM	ONNX File	4,683 KB
model_cat	5/19/2023 2:49 PM	JSON File	74 KB
model_cat.onnx	5/25/2023 6:37 PM	ONNX File	81,605 KB
model_cat_classes	5/19/2023 2:49 PM	JSON File	1 KB
weights	5/19/2023 2:50 PM	H5 File	245,047 KB
weights2	5/15/2023 5:27 PM	H5 File	14,004 KB

Figura 24: Ficheiro *ONNX* criado

É relevante referir que a presença de ficheiros idênticos aos utilizados na criação do ficheiro *.onnx* nesta pasta de projeto, sendo estes *weights.h5* (semelhante a *weights2.h5*), *model_cat.json* (semelhante a *config.json*) e *model_cat_classes.json* (semelhante a *model.json*) deve-se ao facto destes ficheiros representarem uma rede neuronal semelhante à previamente descrita, tendo estes ficheiros sido utilizados inicialmente para testar o funcionamento do *script* e testar o método de importação do ficheiro *ONNX* criado para o *Unity*.

Durante o processo de conversão da rede neuronal, foi realizada também a instalação dos *packages ML-Agents* e *Barracuda*, de maneira a permitir ao Editor identificar ficheiros com o formato *ONNX*.

Infelizmente, imediatamente após recomençar o Editor depois da instalação do *Barracuda*, surgem dois erros, ambos relacionados com a presença de mais do que uma “precompiled assembly” com o mesmo nome, neste caso *Google.Protobuf.dll*, como pode ser observado na Figura 25.

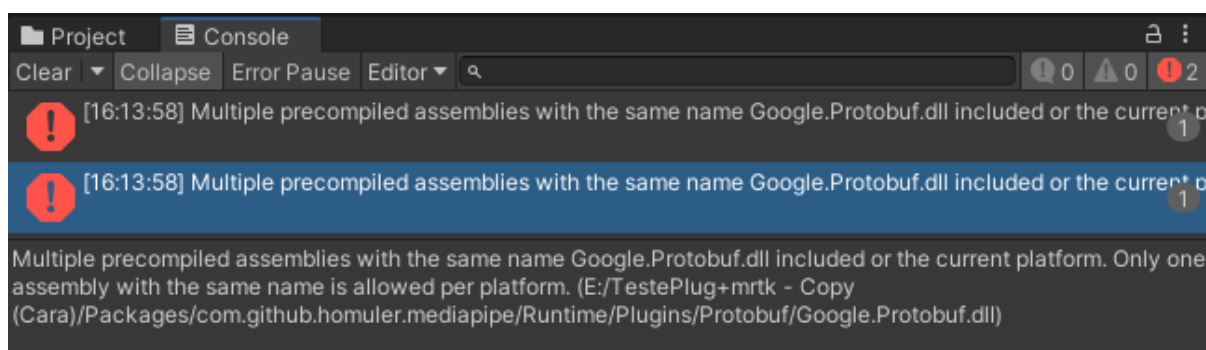


Figura 25: Erro *multiple precompiled assemblies*

O erro apresentado “Multiple precompiled assemblies with the same name Google.Protobuf.dll included or the current platform. Only one assembly with the same name is allowed per platform” é gerado devido ao facto de tanto o package *MediaPipeUnityPlugin* como o package *Barracuda* (mais concretamente, a

versão 2.1.3, utilizada pelo package *ML-Agents*) apresentarem uma “precompiled assembly” denominada de *Google.Protobuf.dll*. No caso de ambos os *packages* necessitarem deste ficheiro para funcionar corretamente, seria necessário encontrar uma versão de ambos capaz de correr a mesma versão do *Google.Protobuf.dll* e apagar uma das “assemblies” importadas de maneira a permitir que ambos os *packages* funcionem corretamente.

De maneira a averiguar se seria possível utilizar uma versão diferente de um dos *packages* em questão para tornar a sua utilização conjunta compatível, foram testadas várias versões do package *Barracuda* isoladamente, sem a presença do package *ML-Agents*, uma vez que este é relativamente recente e utilizada poucas versões distintas do package *Barracuda*. Este processo foi mais complicado devido sobretudo ao facto de não ser inicialmente conhecida a versão do *Google.protobuf.dll* utilizado no package *MediaPipeUnityPlugin* (no caso do *Barracuda* 2.1.3, este utiliza o *Protobuf* 3.10.0). Após uma análise mais profunda, foi possível encontrar a versão do *Google.Protobuf.dll* utilizado, sendo esta a 3.20.0, como apresentado na Figura 26.

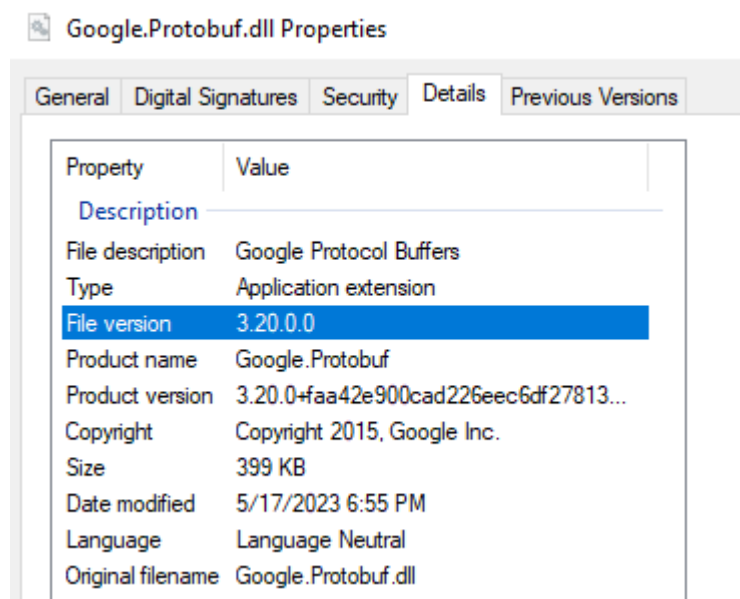


Figura 26: Versão do Google.Protobuf.dll

Ao contrário do esperado, não será possível utilizar um *Google.Portobuf.dll* comum, uma vez que o package *MediaPipeUnityPlugin* utiliza a versão 3.20.0, uma versão acima da versão utilizada pelo package *Barracuda* mais atual.

A única maneira restante de utilizar os dois *packages* no mesmo projeto passa por remover um dos *Google.protobuf.dll* de maneira a eliminar a incompatibilidade criada pela diferença entre versões. Neste caso, foi removido a “assembly” do package *Barracuda*, com a precondição de não ser possível utilizar qualquer função deste package relacionada com a “assembly” removida.

De maneira a testar se o *package ML-Agents* seria capaz de realizar as mesmas funcionalidades necessárias para o desenvolvimento da aplicação, foi importada e testada uma rede neuronal proveniente do repositório *GitHub* oficial.

O propósito desta rede é detetar objetos através da captura de vídeo, sendo o resultado final desejado a identificação correta desse objeto. O processo de importação é extremamente simples, sendo apenas necessário arrastar a rede neuronal em formato *ONNX* para a localização desejada dentro da pasta “Assets”. No caso desta apresentar um formato válido e for reconhecida pelo Editor, o símbolo que a representa tal como as opções que podem ser customizadas na secção “Inspector” serão as apresentas na Figura 27.

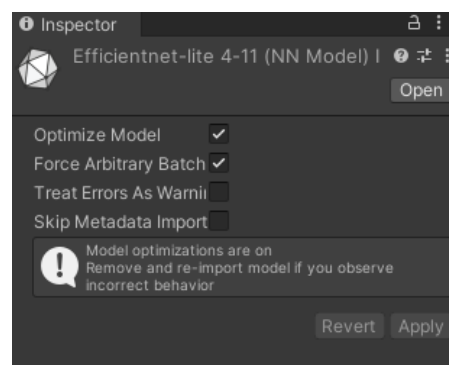


Figura 27: Ficheiro *ONNX* no *Unity*

Devido à natureza do formato *ONNX*, o ficheiro em questão apenas guarda a arquitetura dessa mesma rede, não guardando elementos essenciais como os “weights” e “parameters”. Para utilizar esta mesma rede é então necessário no mínimo um *script* capaz de criar o *input* necessário, neste caso as imagens capturadas a partir do vídeo gerado pela câmara e os ajustes necessários de maneira a estas imagens corresponderem ao formato de input desejado pelo modelo. Mais importante ainda, este script tem de ser capaz de utilizar o ficheiro *ONNX* para classificar essas mesmas imagens. Outro componente que deve estar presente é um ficheiro de texto (.txt) composto pelos *outputs* esperados do modelo, organizados de acordo com o número do *index* a que pertencem. Sem este ficheiro, não é possível atribuir o nome do objeto identificado uma vez que o modelo *ONNX* gera apenas o *index* da palavra que considera melhor descrever o objeto. Na Figura 28 é possível ver o resultado final gerado pela aplicação teste criada.



Figura 28: Resultado do teste do ficheiro *ONNX*

Tendo então testado o processo de importação e utilização de uma rede neuronal e com a conversão da rede neuronal para o único formato suportado pelos *packages* oficiais do *Unity*, é apenas necessário importar esta mesma rede diretamente para o Editor. É importante referir que após o processo de conversão do ficheiro de teste “weights.onnx” (complementado pelos ficheiros “model_cat.json” e “model_cat_classes.json”), este também foi importado para o Editor, onde não apresentou qualquer irregularidade, como constatado na Figura 29.

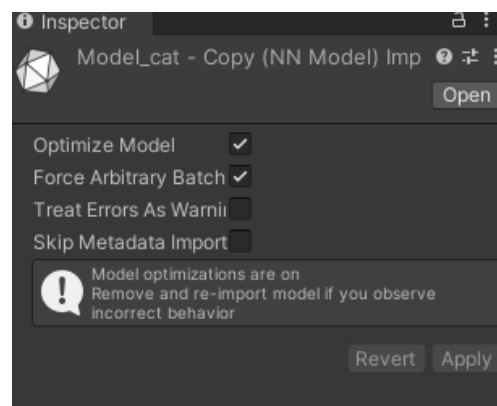


Figura 29: Importação do ficheiro *ONNX* teste

Apesar dos testes previamente realizados e da resolução do problema de compatibilidade entre os *packages* *MediaPipeUnityPlugin* e *ML-Agents*, a importação do modelo resultou em dois erros inesperados, ambos relacionados com a presença de um “Unknown type Loop”, como apresentado na Figura 30.

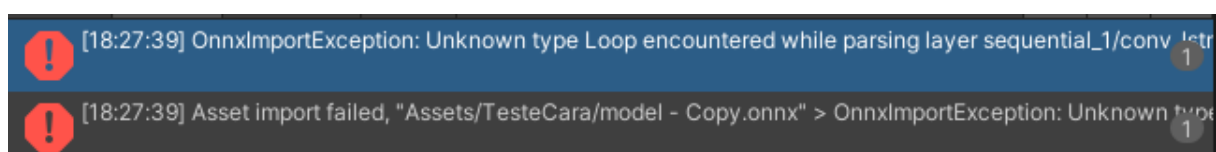


Figura 30: Erros gerados pela importação do ficheiro *ONNX*

A presença de *loops* no modelo é algo que impossibilita completamente o uso do único formato suportado pelos *packages* oficiais do *Unity*, *ONNX*, uma vez que este não permite qualquer tipo de comportamento dinâmico. Atualmente, não existe nenhuma forma de adaptar um modelo em *ONNX* para aceitar comportamento dinâmico, embora esteja em desenvolvimento uma “feature” que permitirá modelos *ONNX* serem executados com inputs dinâmicos, denominada de “dynamic shape inference”.

Com o facto de o modelo necessário apresentar conteúdo dinâmico, a única solução para incorporar diretamente a rede neuronal na aplicação passaria pela mudança da versão utilizada do *Unity* 2020 para a mais recente, *Unity* 2022.3, uma vez que esta já permite o use do *package ML-Agents* 2.0.0, que já não utiliza o *package Barracuda*, mas sim *Sentis*, uma nova biblioteca de inferência contruída em *TensorFlow*, não sendo mais necessário o uso de *ONNX*. Infelizmente, não é possível utilizar qualquer versão mais recente do que *Unity* 2021, uma vez que tal não é suportado pelo *package MRTK*, responsável pela interface, comportamento e *deployment* da aplicação para o dispositivo *HoloLens* escolhido.

4.3.2 Rede Neuronal - Alternativa

Uma vez que não será mais possível utilizar a rede neuronal diretamente no *Unity*, foi necessário adotar uma nova abordagem à forma como esta pode ser utilizada pela aplicação. A alternativa escolhida foi hospedar a rede neuronal, no seu formato original, num servidor ao qual a aplicação consegue aceder. Uma vez que a rede neuronal em questão já se encontrava hospedada por parte do CCG, foi decidido utilizar esse mesmo servidor para fornecer à aplicação o resultado obtido pela rede neuronal, que corresponde à palavra, em língua portuguesa, correspondente aos sinais de LGP captados.

4.3.3 *MediaPipeUnityPlugin* – Complicação

Inicialmente, a aplicação a criar necessitava apenas de ser capaz de detetar as duas mãos do utilizador e utilizar as *landmarks* detetadas dessas mãos para obter a tradução, em língua portuguesa, do sinal realizado. De uma forma muito resumida, para permitir a obtenção das *landmarks* necessárias, seria necessário criar um *script*, semelhante ao apresentado no único tutorial existente para este *package* (intitulado de “Official Solution” e focado apenas na deteção dos *landmarks* da cara), capaz de obter a *ImageFrame* criada e posteriormente utilizá-la juntamente com o *package* referido para obter os *landmarks* desejadas. Infelizmente, o processo de obtenção das *landmarks* das mãos, normalmente referido como *Hand Tracking*, apresenta necessidades acrescidas comparativamente ao caso da deteção

dos pontos da cara, ou *Face Tracking*. Estas necessidades, posteriormente detetadas, são a necessidade de carregar diferentes modelos de *machine learning* já treinados (importados para o *Unity* em formato .bytes), a necessidade de utilizar um *MediaPipe graph* diferente (denominado de “Hand_tracking_cpu” em vez de “Face_mesh_desktop_live”), a necessidade de criar e enviar *Side Packets* (necessário para o *MediaPipe graph* a usar) e a necessidade de obter não só as *landmarks*, mas também informação relativa a qual das mãos é que foi detetada (aspeto também relacionado com o *MediaPipe graph* mas proveniente de outro *calculator*).

Para além do facto de ter sido necessário despendar tempo considerável a averiguar os elementos que deviam ser alterados ou adicionados para permitir a obtenção das *landmarks* das mãos em vez das *landmarks* da cara, foi também adicionada a necessidade de não só detetar os pontos das mãos, mas também da cara, em simultâneo. O processo para obter estas *landmarks* da cara adicionais, embora se encontra bem documentado e seja relativamente simples, necessita de um *MediaPipe graph* diferente do utilizado. Caso ambos os *graphs* fossem utilizados ao mesmo tempo, alguns erros tinham tendência a ocorrer e o performance era extremamente afetado pela utilização do segundo *graph*.

4.3.4 *MediaPipeUnityPlugin* - Alternativa

De maneira a resolver este problema foi necessário juntar ambos os *MediaPipe graphs* já referidos, tendo especial atenção em elementos que possam levar a problemas de compatibilidade, mais concretamente variáveis que apresentam o mesmo nome. Com apenas um *MediaPipe graph* a realizar ambas as funções foi possível obter tanto as *landmarks* das mãos, a mão em questão e as *landmarks* da cara, apresentando sempre um nível de desempenho razoável.

4.3.5 *MRTK* – Complicação

Durante o uso do *package MRTK* foi detetado o facto da língua portuguesa, previamente definida como a língua falada e escrita a utilizar (devido ao facto de a língua gestual a ser utilizada é a LGP) não ser suportada pelo dispositivo *Hololens 2* e, subsequentemente, não ser suportada pelo *package MRTK* no que toca às funções de “Dictation” e “Speech”. As únicas línguas suportadas são Inglês (Estados Unidos, Reino Unido, Canadá, Índia e Austrália), Francês, Alemão, Japonês, Mandarim (Chinês simplificado e Chinês tradicional) e Espanhol.

4.3.6 MRTK – Alternativa

Devido a este problema, não é mais possível realizar a ação de detecção e escrita da voz detetada uma vez que português não é uma das línguas suportadas. De maneira a resolver este problema e de garantir a existência de todas as funcionalidades chaves na aplicação final, foi escolhido um novo método para reconhecimento de voz, sendo este o “Speech service”, um dos vários serviços fornecidos pelos *Azure Cognitive Services*. Embora o uso desta ferramenta não seja tão simples quanto o uso da função “Dictation” do *package MRTK*, o processo de incorporação deste serviço não é complicado e não leva a qualquer perda em termos de performance, existindo documentação extensa complementada com exemplos destinados à utilização deste mesmo serviço no Editor *Unity*.

4.3.7 Emulador *Hololens* – Complicação

Inicialmente, o uso do Emulador capaz de simular o comportamento da aplicação no dispositivo *Hololens* foi adotado com o intuito de permitir o teste desta mesma aplicação enquanto o dispositivo *Hololens* no qual a aplicação ia ser apresentada não se encontrava disponível. Infelizmente, como já referido na secção de instalação, o computador no qual está a ser desenvolvido a aplicação utiliza o *Windows 10 Home* como sistema operativo. O facto de utilizar a versão *Home* deste sistema operativo e não a versão *Pro*, *Enterprise* ou *Education*, faz com que este sistema operativo em particular não suporte nativamente *Hyper-V*, um dos componentes essenciais para correr o emulador, responsável por correr mais do que um sistema operativo (neste caso, o original e o do emulador *Hololens*) no mesmo *hardware*. Também como já referido na secção de instalação, é possível instalar e correr o *Hyper-V* mesmo usando *Windows 10 Home*, embora possa apresentar problemas de desempenho ou falhas no sistema.

Apesar do facto do *Hyper-V* não ser suportado pelo sistema operativo, foi possível começar o emulador (*HoloLens 2* versão 10.0.20348.1535) com sucesso no *Visual Studio 2022*. Apesar deste ter começado com sucesso, o desempenho da aplicação no dispositivo emulado foi extremamente fraco e nada semelhante ao desempenho registado no dispositivo alvo. A principal razão que terá causado esta diferença pode ser atribuída sobretudo ao *hardware* do computador utilizado para correr o emulador, especialmente a placa gráfica, que suporta até *WDDM 2.4 graphics driver*, enquanto a única versão recomendada é a *WDDM 2.5 graphics driver* para o emulador em questão.

4.3.8 Emulador *Hololens* – Alternativa

De maneira a resolver esta questão, duas alternativas foram identificadas, sendo esta: criação de uma *virtual machine* com as características mínimas necessárias para correr o emulador ou o uso do dispositivo *Hololens 2* em si para o teste da aplicação criada.

A criação da *virtual machine* (*VM*), com a extensa lista de características mínimas necessárias para correr o emulador, referidas na secção de instalação, apresenta um certo custo, pelo menos quando criada através do *Azure Virtual Machines*. Para além do custo, um dos aspetos a ter em consideração ao comparar as duas soluções é o facto de ser extremamente difícil avaliar aspetos importantes como a distância a que a interface do utilizador se encontra comparativamente ao braço do utilizador (forma principal como o utilizador interage com a interface em dispositivos *Hololens*).

A utilização do dispositivo em si para teste da aplicação não pode ser imediata, visto este encontrar-se em uso, mas o tempo necessário até este se encontrar disponível não seria muito. Uma das desvantagens desta alternativa seria a necessidade de utilizar ferramentas externas (num dispositivo à parte) para diagnosticar erros relevantes anotados nos *Logs* da aplicação.

Tendo em conta os aspetos referidos, optou-se pelo uso do dispositivo final como meio de teste e avaliação da aplicação em desenvolvimento, embora seja necessário recorrer a mais ferramentas para o diagnóstico de erros críticos aquando da execução da aplicação.

4.4 Complicações Agravadas e Alternativas Adotadas

Infelizmente, mesmo após resolver as diversas complicações revelados nos momentos iniciais do desenvolvimento da aplicação, várias complicações adicionais foram encontradas e alterações no projeto tiveram de ser realizadas de maneira a permitir o funcionamento adequados de todas as funcionalidades inicialmente previstas. As seguintes complicações enumeradas, muitas relacionadas com as alternativas adotadas na secção anterior, não são apenas relativas aos momentos finais de desenvolvimento, mas também estão relacionadas com a aplicação final no dispositivo alvo.

4.4.1 *MediaPipeUnityPlugin* – Complicação

Inicialmente, o projeto foi idealizado tendo como base três interfaces, duas destas relacionadas com os dois tipos distintos de utilizador e uma destinada à introdução e seleção de uma das duas interfaces referidas. Infelizmente, as três interfaces, cada uma pertencente a uma cena de *Unity* diferente, foram desenvolvidas até à sua conclusão separadamente, algo que na maioria dos projetos não seria alvo de

qualquer complicação, mas neste caso foi. Com a divisão das três interfaces por cenas e devido à forma como as *landmarks* são obtidas, dois erros críticos são apresentados. O primeiro, presente na Figura 31, é referente ao ciclo *While*, criado com o intuito de atualizar a *inputTexture* com os últimos pixéis da *webCamTexture* (por outras palavras, atualiza a imagem a enviar pela mais recentemente obtida).

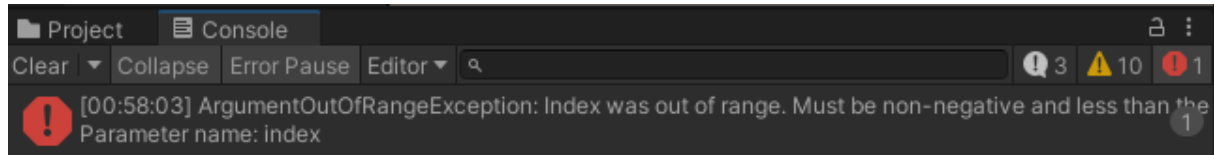


Figura 31: Erro “ArgumentOutOfRangeException”

Sem este ciclo, a imagem enviada seria sempre a mesma, o que poderia ser uma possibilidade caso a intenção da aplicação fosse a tradução de sinais a partir de imagens e não tradução em tempo real. É também importante referir que este erro ocorre sempre que é selecionada a interface destinada a ISDA, independentemente desta seleção ser realizada a partir da interface principal (Index 0 -> Index 1) ou a partir da interface destinada a ICDA (Index 1 -> Index 2).

De maneira a tentar evitar a ocorrência do primeiro erro, a interface inicial foi removida, começando a aplicação a correr com a interface destinada a ISDA, apresentando a opção de mudar para a interface destinada a ICDA. Infelizmente, esta abordagem leva ao segundo erro crítico, presente na Figura 32, sendo este relativo ao facto da classe “ResourceManager” ter sido iniciada mais do que uma vez, o que não é uma operação inválida.

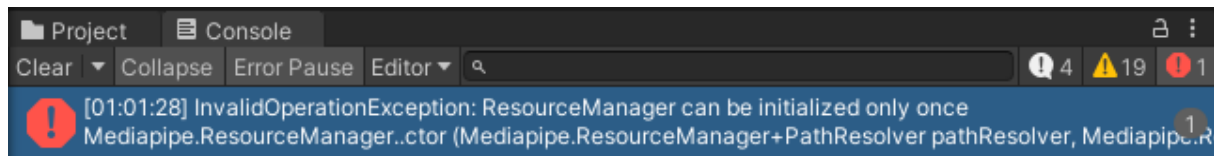


Figura 32: Erro InvalidOperationException

Este erro ocorre quando se pretende voltar à interface destinada a ISDA a partir da interface destinada a ICDA, começando como anteriormente referido na interface para ISDA (Index 2-> Index 1 -> Index 2). Tal como o primeiro erro, o *ResourceManager* é essencial para o funcionamento da aplicação, sendo necessário alterar a estrutura da mesma para permitir o seu correto funcionamento.

Uma vez que tanto o ciclo *While* como a classe *ResourceManager* são essenciais para o desenvolvimento desta solução é mais fácil alterar drasticamente a interface da aplicação para poder resolver estes dois erros.

Para além da primeira complicação, as *landmarks* que necessitam ser enviadas também foram alvo de alterações. Inicialmente, tinha sido apenas visionado a obtenção e envio das *landmarks* relativas às mãos

cara, sendo estas distinguidas entre direita e esquerda. Posteriormente, foram acrescentadas as *landmarks* da cara, sendo necessário alterar o *MediaPipe graph* de maneira a permitir a obtenção em simultâneo das *landmarks* necessárias. Com o desenvolvimento da aplicação, não será apenas necessário as *landmarks* das mãos e da cara, mas também da postura. Infelizmente, o processo atualmente utilizado para obter as *landmarks* das mãos e da cara, que passa pela conversão de dois *MediaPipe graphs* em um, não pode ser realizado novamente para incorporar os três *MediaPipe graphs* em um, uma vez que o *MediaPipe graph* relativo à postura apresenta variáveis que partilham o mesmo nome com um dos outros *graphs*. Outro aspeto, importantíssimo, é o facto de a performance ser bastante afetada na situação em que os *três graphs* são alterados e unidos num só, sendo esta falta de performance notada sobretudo no desenho da representação espacial dos pontos identificados.

4.4.2 *MediaPipeUnityPlugin* – Alternativa

Para resolver os dois erros críticos, seria sempre necessário alterar drasticamente a forma como as três interfaces interagem em si, sendo que a solução encontrada se foca na eliminação de duas das três cenas criadas (a relativa à interface principal e a relativa a uma das duas interfaces restantes), passando assim a haver apenas uma única cena, onde as duas interfaces diferentes foram agregadas de maneira a criar uma interface conjunta. Embora a interface principal tenha sido removida, parte do intuito principal que servia (o utilizador não ser forçado a começar na interface contrária à qual pretende utilizar) continua presente na aplicação, no aspeto em que nenhuma das duas interfaces são apresentadas antes de serem selecionadas, seleção esta realizada no *NearMenu* comum. A outra razão pelo uso da interface principal passa pela necessidade de explicar ao utilizador que interface é que deve escolher (tendo em conta o tipo de utilizador que é). Embora já não seja tão evidente, este aspeto continua a ser coberto pela aplicação após a alteração, sendo que a informação relevante relativa às duas diferentes interfaces e à razão para as escolher se encontra agora presente apenas na opção “Ajuda”, localizada no *NearMenu* comum.

Para resolver a segunda complicação, será necessário adotar como base o exemplo “Holistic”, ou Holístico, que trata cada uma das partes do corpo que pretendemos obter (cara, postura, mão esquerda e mão direita) no mesmo *graph* e de forma independente, não sendo mais necessário realizar a distinção das mão através de um “calculator” separado nem de realizar o desenho das *landmarks* em três “Annotation Layers” diferentes. Embora esta alternativa seja a única maneira de obter o conjunto de pontos das mãos, cara e postura sem afetar gravemente o desempenho da aplicação, a forma como esta obtém os pontos e desenha as anotações é consideravelmente diferente dos métodos e classes

usadas para obter os pontos das mãos e da cara. O facto de não ser possível utilizar os mesmos métodos para obter as *landmarks* e desenhar as anotações, juntamente com o facto de não haver qualquer tipo de documentação ou exemplo que retrate o funcionamento deste exemplo, especialmente relativo ao *graph* utilizado, levou a um grau significativo de dificuldade acrescida aquando do desenvolvimento da solução, não havendo qualquer outra alternativa capaz de resolver a complicação identificada.

4.4.3 Rede Neuronal – Complicação

Com a impossibilidade de utilizar a rede neuronal diretamente no Editor, optou-se pelo uso de um servidor, fornecido por parte do CCG, onde esta mesma rede já se encontrava hospedada. Embora esta mudança aparenta facilitar o uso da rede neuronal por parte da aplicação, uma vez que é apenas necessário enviar o conjunto de *landmarks* identificadas, a realidade infelizmente não foi a mesma. Foram três as complicações identificadas causadas pelo uso do servidor fornecido pelo CCG, sendo esta, por ordem de ocorrência, mudanças no formato a ser enviado para o servidor, impossibilidade de verificar o certificado *SSL/TLS* do servidor e problemas de compatibilidade entre o protocolo *TLS* utilizado e a versão do *Unity* utilizada.

Relativamente ao primeiro problema, referente à mudança do formato do pedido a ser enviado ao servidor, este deve-se sobretudo ao facto da rede estar em constante desenvolvimento, sendo necessário alterar o conteúdo a ser enviado para o servidor de maneira a corresponder às novas necessidades.

Relativamente ao segundo problema, este foi detetado através do uso da ferramenta *Postman*, utilizada com o intuito de tentar descobrir a razão pela qual o envio de pedidos para o servidor a partir do Editor gerava erros. O problema pode ser visualizado na Figura 33.

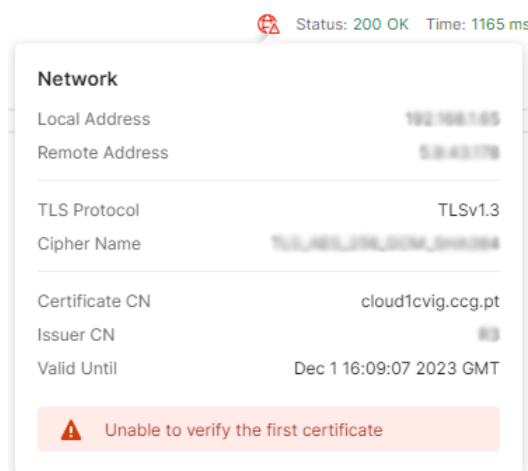


Figura 33: Problema com o servidor

Através desta ferramenta, é possível detetar que não é possível verificar a validade do Certificado fornecido pelo servidor.

Finalmente, relativamente ao terceiro e último problema, o mais crítico, a versão atualmente utilizada do Editor *Unity* não suporta a realização de pedidos *POST* para servidores que utilizam o *TLS* 1.3 como protocolo de segurança, sendo necessário avaliar uma forma alternativa para enviar e obter a informação do servidor.

4.4.4 Rede Neuronal – Alternativa

Para o primeiro problema, foi necessário alterar as *landmarks* registadas pelo script criado (processo relativamente simples, após já ter registado as *landmarks* anteriores) e, mais importante ainda, foi necessário analisar melhor a forma como o servidor recebe as *landmarks* em questão. O ficheiro *config* inicial foi descartado, uma vez que já não correspondia ao formato desejado, e foi realizado um *POST request* à diretoria do servidor responsável pelo ficheiro *config* do servidor (*/config/*). Apesar deste ser o ficheiro *config* desejado, a diretoria para qual o pedido é realizado para obter a palavra (*/mp_estimator/*) requer um formato diferente, mais concretamente um *JSON* composto apenas por oito listas, cada uma correspondente às 176 *landmarks* desejadas detetadas num único *frame*. Devido às limitações do *package* inicialmente utilizado para criar o ficheiro *JSON* a ser enviado (*JsonUtility*), foi necessário criar a estrutura do ficheiro a enviar sem o auxílio de qualquer *package*.

Relativamente ao segundo problema, a única maneira de o resolver passa pela não verificação do certificado do servidor aquando do envio do pedido *POST*. Este comportamento não é ideal uma vez que torna a ligação ao servidor pouco segura, mas uma vez que não é possível realizar alterações ao servidor em questão e a aplicação foi criada com o intuito de investigação e não de distribuição, não verificar o certificado do servidor não compromete o funcionamento da mesma.

Por fim, para resolver o último problema é necessário mudar drasticamente a forma como as *landmarks* são enviadas e a resposta do servidor é recebida. Foram identificados duas alternativas distintas para resolver este problema, sendo esta a mudança da versão utilizada pelo Editor ou a criação de um servidor proxy, sendo a única função deste receber o ficheiro *JSON* com as *landmarks* desejadas, a partir de um pedido *POST* por parte da aplicação, e enviá-las para o servidor final, retornando a resposta deste para a aplicação. Infelizmente, por questões de incompatibilidade, não é possível atualizar a versão do Editor para uma versão compatível com o protocolo *TLS* 1.3 (a versão mais antiga que suporta este protocolo é a *Unity* 2022.1.0f1), sendo então necessário criar um servidor *proxy* capaz de comunicar com o servidor desejado.

4.4.5 *Build* da aplicação – Complicação

Relativamente ao processo de *build* da aplicação, processo este que necessita ser efetuado para poder ser realizado o *deployment* desta mesma aplicação no dispositivo *Hololens* escolhido, é necessário garantir que a aplicação criada cumpre um certo conjunto de requisitos. Este requisitos podem ser encontrados na secção “Project Validation” (“Edit” -> “Project Settings” -> “XR Plug-in Management”). De maneira a garantir que todos os requisitos são cumpridos, o desenvolvedor tem a opção de “validar” todos os aspetos que não correspondem ao esperado, sendo este processo realizado automaticamente pelo Editor. O facto de todos os requisitos se encontrarem corretamente definidos é essencial e serve de confirmação para o desenvolvedor que a aplicação irá apresentar o comportamento desejado no dispositivo onde vai ser importado. No caso desta aplicação, apesar de todos os requisitos se encontrarem corretamente definidos aquando do *build* da aplicação, esta não apresentava o funcionamento desejado no dispositivo *Hololens* e, caso o projeto fosse fechado e reaberto no editor, um erro de validação aparecia sempre (sendo posteriormente dado como resolvido após “resolver” o projeto). O erro de validação pode ser visualizado na Figura 34.

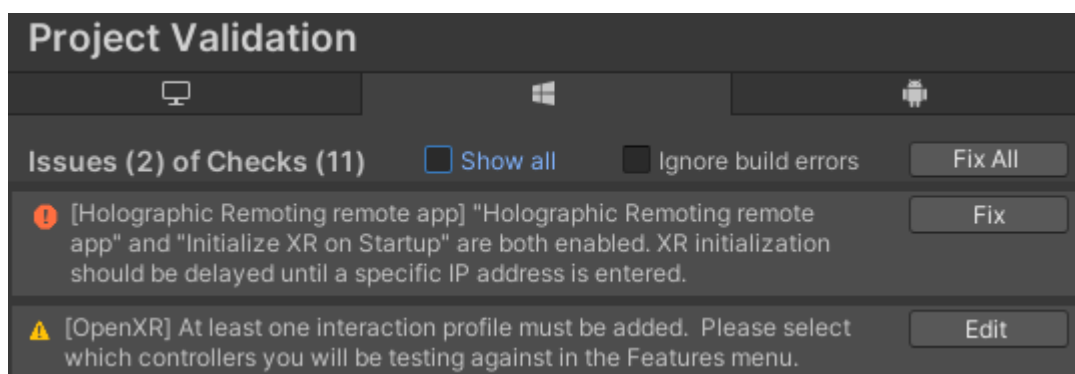


Figura 34: Erro de validação *Unity*

4.4.6 *Build* da aplicação – Alternativa

De maneira a resolver este problema foi necessário consultar tanto a documentação oficial como fóruns online, tendo chegado à conclusão de que se tratava de um *bug*, uma vez que ao carregar no botão “Fix” ou “Fix All”, nenhuma ação era realizada por parte do editor para resolver a ação, embora o aviso fosse removido. Este comportamento pode ser comprovado pela comparação das Figuras 35 e 36, sendo que nesta primeira imagem pode ser visualizado o erro inicialmente detetado e na segunda pode ser visualizada o mesmo erro, agora sem mensagem de erro.

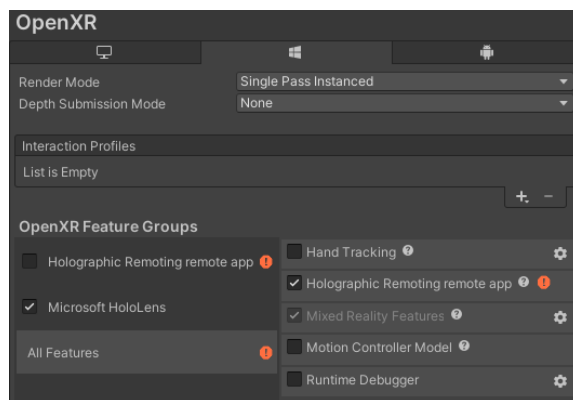


Figura 35: Erro *OpenXR Features*

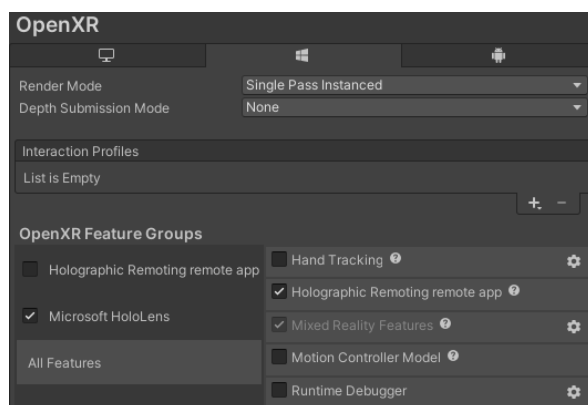


Figura 36: Erro *OpenXR Features* supostamente resolvido

Após analisar com mais detalhe a situação, foram realizadas alterações à secção *OpenXR* de maneira a permitir o funcionamento correto da aplicação no dispositivo escolhido. Esta secção pode ser visualizada na Figura 37.

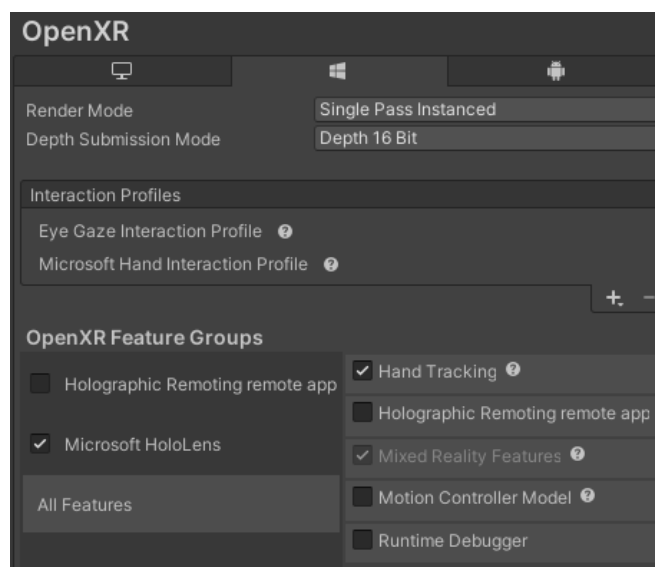


Figura 37: Secção *OpenXR* final

4.4.7 *Hololens 2* – Complicação

A aplicação criada depende de um conjunto alargado de *packages* para permitir criar as funcionalidades inicialmente definidas. Um desses *packages* é justamente o *MediaPipeUnityPlugin*, que é essencial e insubstituível para realizar a tarefa mais importante da aplicação, sendo esta a deteção das *landmarks* do corpo do indivíduo com quem o utilizador pretende comunicar. Este *package*, verificado no início do projeto, suporta *Windows 10*, embora a sua compatibilidade ainda seja considerado experimental. Infelizmente, apenas após realizar o processo de *deployment* da aplicação criada para o dispositivo *Hololens 2*, foi descoberta a incompatibilidade deste mesmo *package*, essencial para o funcionamento da aplicação, com o sistema operativo do *Windows* utilizado pelo dispositivo em questão. Esta incompatibilidade deve-se ao facto do dispositivo *Hololens 2* não utilizar o SO *Windows 10*, mas sim uma versão deste mesmo software denominada de *Windows Holographic OS*, otimizada apenas para o uso desta em dispositivos de *MR*, o que faz com que não apresente uma quantidade considerável de *features* ou ferramentas necessárias para o funcionamento deste mesmo *plugin*, algo que foi descoberto através do uso da aplicação *DependenciesGui*. Na Figura 38 é possível visualizar algumas das dependências deste *package*.

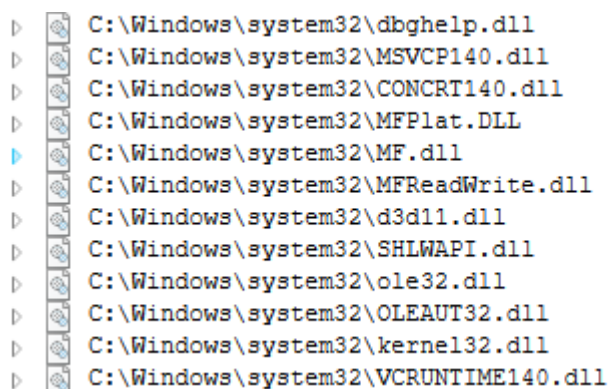


Figura 38: Algumas das dependências do *package*

4.4.8 *Hololens 2* - Alternativa

Com esta incompatibilidade completamente inesperada, foi necessário avaliar uma nova forma de apresentar a aplicação criada. Foi ponderado o uso de um dispositivo *Android*, uma vez que tanto o *package MRTK*, como o *package MediaPipeUnityPlugin* como o *package SpeechSDK* (utilizado pelo serviço *Speech-to-text* da *Microsoft Azure*) suportam este dispositivo, mas seria necessário realizar um número considerável de alterações à aplicação criada, especialmente aos scripts associado a cada uma da funcionalidades da aplicação. Para além de mudanças aos níveis dos *scripts*, que não seriam simples, o processo de teste da aplicação teria de ser realizado diretamente através do dispositivo, uma vez que

o *package MediaPipeUnityPlugin*, que passaria a correr através do *GPU* e não do *CPU*, não é compatível com *Windows* e, portanto, não seria possível fazer algo tão básico como correr a cena criada no Editor. Por fim, outra desvantagem do uso deste *smartphone* como dispositivo é o facto de este obrigar o indivíduo que utiliza língua gestual a ter de optar entre conseguir visualizar a resposta oral do indivíduo com quem está a falar ou responder a essa pessoa através de LGP, algo que não é possível se tiver uma das mãos ocupadas.

Com estes aspetos em conta, foi decidido manter a aplicação apenas no Editor *Unity*, adotando o dispositivo *android* como um possível trabalho futuro.

4.5 Conclusão do desenvolvimento da Aplicação

Nesta secção é apresentado o processo de desenvolvimento da aplicação final criada. Este processo foi dividido em quatro, sendo a primeira parte correspondente à estrutura adotada pela cena criada a partir da junção das interfaces ICDA e ISDA. Com a estrutura da cena evidenciada, a segunda parte é referente aos objetos dessa mesma cena que compõem a Interface Partilhada, ou seja, todos os objetos que são utilizados tanto para indivíduos ICDA ou ISDA. Por fim, as duas restantes partes ambas falam dos objetos utilizados para cada um dos tipos de utilizadores, sendo a primeira destas referente a ICDA e a segunda referente a ISDA.

4.5.1 Estrutura da cena final

Com base no desenvolvimento inicial da aplicação, onde foram criadas as três interfaces inicialmente idealizadas, e com base nos problemas e alternativas identificadas, foi então desenvolvido a aplicação final. Esta aplicação, em contraste com a aplicação inicialmente idealizada, apresenta apenas uma única cena, sendo esta dividida em duas interfaces. Para ICDA, é utilizado o *Azure Cognitive Service* “Speech Service” através do *SpeechSDK*. Para ISDA, são utilizados dois servidores diferentes, um para a rede e outro para permitir a aplicação *Unity* comunicar com este primeiro, e o *package MediaPipeUnityPlugin* para obter o *input* desejado para a rede.

A interface criada foi denominada de Interface Final e a cena criada para juntar as duas interfaces denominou-se de “ICDA + ISDA”. Em termos de estrutura, apesar de já não serem mais utilizadas três interfaces diferentes, esta continua idêntica às estruturas das interfaces previamente criadas, apresentando os objetos básicos de uma aplicação criado com *MRTK*, sendo estes “*Directional Light*”, “*MixedRealityToolkit*”, “*MixedRealityPlayspace*” e “*MixedRealitySceneContent*”. Para além dos objetos

básicos, apresenta também os menus de saída e ajuda, “Sair” e “Ajuda”, um *NearMenu* com quatro botões principais, “NearMenu” e um objeto *canvas* para representar o resultado de ambas as interfaces, “Canvas”.

No que toca aos objetos adicionados apenas nesta cena, estes foram o “Manager”, onde foram organizados os diferentes *scripts* utilizados nesta cena, e o “SlateUGUI”, que corresponde ao painel das definições da interface ICDA.

A estrutura da cena final pode ser visualizada na Figura 39.

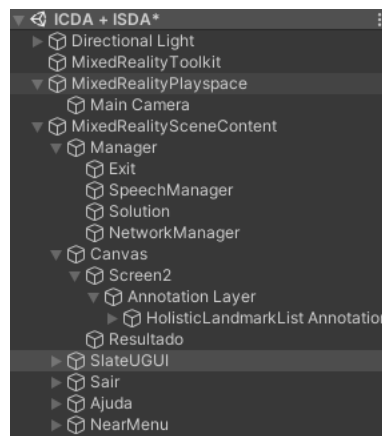


Figura 39: Estrutura da Interface Final

4.5.2 Interface Partilhada (Final)

Relativamente aos objetos da interface partilhados entre as duas interfaces, estes objetos são “Canvas”, “Sair”, “Ajuda” e “NearMenu”. Todos estes provêm das interfaces ICDA e ISDA inicialmente criadas e quase não foram alvo de alterações, dada a natureza de cada um destes objetos.

Relativamente ao “Canvas”, é a este que se encontra associado o objeto “Resultado”, objeto este que é partilhado por ambas as interfaces para demonstrar o resultado obtido em cada uma. Relativamente ao “Sair”, este representa o menu de saída para o utilizador previamente referido. Relativamente ao “Ajuda”, este representa um menu de ajuda para o utilizador, também este previamente referido. Por fim, “NearMenu”, como já referido, é um menu com quatro botões utilizados para realizar certas ações dentro da interface.

Devido à natureza destes mesmo elementos, apenas o objeto “Ajuda” e “NearMenu” sofreram alterações comparativamente ao seu primeiro momento de desenvolvimento. Para o objeto “Ajuda”, foi alterada a descrição deste menu, de maneira a descrever o funcionamento de ambas as interfaces e não de uma só. Para o objeto “NearMenu”, os dois primeiros botões foram alterados esteticamente e funcionalmente, sendo agora denominados “Cap. Voz” (Captura de Voz) e “Cap. Sinais” (Captura de Sinais) dentro da

aplicação. O primeiro botão, em vez de ativar o processo de gravação e de não alterar nenhum dos elementos da interface, agora abre o painel “SlateUGUI”, onde as definições do *Speech Service* podem ser customizadas, e esconde o *NearMenu* enquanto este painel está a ser utilizado. O segundo botão, em vez de realizar a mudança para a outra interface disponível, ativa os elementos gráficos da solução para ISDA e começa o processo de identificação das *landmarks* a partir da imagem captada.

4.5.3 Elementos ICDA (Final)

Esta subsecção é referente apenas aos elementos da aplicação, sejam estes objetos, *scripts*, ou outros, relacionados com a interface ICDA. Os objetos utilizados exclusivamente nesta cena denominam-se “SlateUGUI” e “SpeechManager” e representam o menu das definições do *Speech Service* e o *script* “SpeechRecognition”, respetivamente. Para além destes objetos, são utilizados os objetos “Resultado” e “NearMenu” para permitir o funcionamento desta interface.

Tal como referido na subsecção anterior, o processo de seleção da interface destinada a ICDA já não é realizada mal a aplicação é iniciada, mas sim a partir da escolha desta mesma interface a partir do *NearMenu* partilhado entre as duas interfaces. Após selecionada esta opção a partir do *NearMenu*, os objetos referentes a esta mesma interface tornam-se visíveis e os objetos referentes a outras interfaces tornam-se invisíveis (sejam estes elementos referentes à outra interface, “Ajuda” ou “Sair”). A partir da janela das definições, é possível mudar a língua oralmente transmitida que se pretende detetar, sendo o resultado final sempre apresentado em português. De momento apenas é possível escolher Português (transcrição do áudio captado), Francês ou Espanhol, embora qualquer língua, desde que suportada pelo *Cognitive Service Azure*, possa ser facilmente adicionada.

Para começar a tradução/transcrição, é apenas necessário selecionar o botão “Começar Tradução”, sendo possível depois fechar a janela das definições de maneira a tornar o texto traduzido/transcrito, apresentado no meio do ecrã, mais visível. Para voltar a abrir esta mesma janela, é apenas necessário voltar a selecionar a opção “ICDA” a partir do *NearMenu*. De maneira a parar a tradução, é necessário apenas carregar no botão “Parar Tradução”.

Para permitir o funcionamento do *Speech Service* como desejado foi criado um novo *script*, “SpeechRecognition”, baseado nos exemplos fornecidos pela *Azure Cognitive Services* no repositório *GitHub* oficial e que foi adicionado à cena através do objeto “SpeechManager”.

De uma forma muito resumida, este *script* permite à aplicação criada detetar, transcrever e traduzir áudio numa das três línguas que podem ser selecionadas na interface, apresentando como resultado texto em língua portuguesa. Para realizar esta função, é utilizado o *Microsoft Cognitive Services Speech*

SDK. Como já referido, para permitir o funcionamento desta função é primeiro necessário criar uma instância deste serviço no portal oficial *Azure*.

4.5.4 Elementos ISDA (Final)

Esta subsecção é referente apenas aos elementos da aplicação, sejam estes objetos, *scripts*, ou outros, relacionados com a interface ICDA. Os objetos utilizados exclusivamente nesta cena denominam-se “Screen2”, “Solution” e “NetworkManager”. O primeiro é utilizado para representar a imagem captada pela câmara do dispositivo juntamente com a representação das *landmarks* identificadas, o segundo adiciona o *script* “TesteHolistic” à cena, permitindo o uso do mesmo durante o correr da aplicação, e o terceiro apresenta como componente o *script* “NetworkManager” e permite a aplicação enviar pedidos *HTTP* para o servidor proxy criado. Para além destes objetos, são utilizados os objetos comuns “Resultado” e “NearMenu” para permitir o funcionamento desta interface.

Para além dos objetos adicionados e dos objetos reaproveitados, também foi utilizado um servidor *flask*, no *IDE PyCharm*, capaz de comunicar com o servidor fornecido pelo CCG.

Tal como referido na subsecção anterior, o processo de seleção da interface destinada a ISDA já não é realizada mal a aplicação é iniciada, mas sim a partir da escolha desta mesma interface a partir do *NearMenu* partilhado entre as duas interfaces. Após selecionada esta opção a partir do *NearMenu*, os objetos referentes a esta mesma interface tornam-se visíveis e os objetos referentes a outras interfaces tornam-se invisíveis (sejam estes elementos referentes à outra interface, “Ajuda” ou “Sair”).

Com a interface para ISDA selecionada, o processo de captação de imagem e de deteção das *landmarks* começa, sendo o resultado apresentado no “Screen2”. Para representar a imagem detetada pela câmara, esta aparece diretamente no “Screen2” enquanto a deteção das *landmarks* é realizada a partir do objeto “HolisticLandmarkList Annotation”, criado com base num *prefab* fornecido pelo *package MediaPipeUnityPlugin*.

Todas as funcionalidades presentes nesta interface são criadas e geridas a partir do *script* “TesteHolistic”. Este *script* foi inicialmente criado a partir do exemplo oficial fornecido pelo *package* utilizado, exemplo este apenas utilizado para a deteção das *landmarks* da cara. Uma vez que este *script* era apenas utilizado para a deteção dos pontos faciais e, com o desenvolvimento do projeto, gerou-se a necessidade de identificar as *landmarks* da cara, postura e mãos, foi necessário alterar grande parte deste *script* e utilizar o *MediaPipe graph* destinado à cena *Holistic* em vez do *MediaPipe graph* criado inicialmente.

Com o uso de um servidor externo em vez da rede neuronal em si, foi necessário criar mais um novo *script*, desta vez capaz de realizar a conexão com um dado servidor. Este *script* utiliza o *package* oficial *UnityWebRequest* para realizar pedidos *POST* (método de comunicação *HTTP*) e foi criado com a intenção de comunicar diretamente com o servidor do CCG disponibilizado. Infelizmente, devido ao protocolo de segurança utilizado por este servidor, não é possível comunicar diretamente com este, tendo sido necessário comunicar primeiro com um servidor *proxy* capaz de enviar o documento *JSON* para o servidor principal e depois reenviar a resposta recebida para a aplicação.

O script criado, “Network Manager”, realiza então um pedido *POST*, através do método *HTTP*, com o documento *JSON* a ser enviado ao servidor *proxy* criado. Mal receba o documento, o servidor *proxy* realiza um pedido *POST* com o documento *JSON* a ser enviado ao servidor do CCG. Quando o servidor *proxy* recebe a resposta do servidor do CCG, esta é enviada como resposta ao pedido *POST* realizado pela aplicação e posteriormente apresentada no ecrã do utilizador.

4.6 Build e Deployment em Hololens 2

Na secção seguinte, os processos de *Build* e *Deployment* da aplicação final criada são descritos. Apesar de terem sido utilizados os seguintes métodos para a realização do processo de *Build* e *Deployment*, tanto um como o outro podem ser abordados de maneiras diferentes, dada a natureza do *package MRTK* e a natureza do dispositivo *Hololens 2*. Tendo em conta este aspeto, foram apenas documentados os processos de *Build* e de *Deployment* utilizados no projeto. No caso do *Build*, este é realizado através do Editor *Unity*, enquanto o *Deployment* é realizado através do *Visual Studio 2022*.

4.6.1 Unity Editor (Build)

Antes de iniciar o processo de *Build* são necessário realizar alguns passos. O primeiro, e essencial, é verificar a presença das pastas “StreamingAssets”, “Plugins” e “WebPlayerTemplates” na pasta dedicada aos “Assets” e garantir que todo o conteúdo que se espera encontrar em cada uma dessas pastas está presente. Relativamente à pasta “Streaming Assets”, esta deve conter ficheiros que o *MediaPipeUnityPlugin* necessita aceder durante o uso da aplicação, sendo esta pasta criada durante o processo de importação do *package*. Relativamente à pasta “Plugins”, esta foi necessário criar e deve conter o *Mediapipe_c.dll*, essencial para o funcionamento do plugin ao qual é associado. Relativamente à pasta “WebPlayerTemplates”, esta foi criada para guardar qualquer cena, *script* ou qualquer outro

elemento que não deva fazer parte da aplicação a ser lançada, mas que se encontre na pasta “Assets” do Editor.

O segundo passo para se poder começar o processo de *Build* consiste na verificação de todas as funcionalidades presentes nas interfaces criadas de maneira a garantir que a aplicação funciona como esperado no dispositivo para o qual vai ser importado, uma vez que nenhuma alteração poderá ser realizada após a construção da aplicação.

O processo de *Build* da aplicação é relativamente simples e realizado inteiramente no Editor. A secção destinada a este processo denomina-se “Build Settings” e encontra-se dentro da secção “File”. Após abrir esta secção, o desenvolvedor necessita escolher as cenas que devem estar presentes na aplicação a construir (neste caso, apenas uma cena foi seleccionada), sendo em seguida necessário seleccionar a plataforma para a qual a aplicação foi criada e onde se pretende que seja utilizada juntamente com a seleção de aspetos relevantes como “Target Device”, “Architecture”, entre outros. A seleção das características da aplicação a ser contruída foi realizada tendo como base a documentação oficial existente.

Antes de concluir o processo de *Build* foi necessário alterar o nome da aplicação, juntamente com outros aspetos menos relevantes. Para realizar tais alterações, dentro da secção “Build Settings”, é necessário seleccionar a opção “Player Settings”. Dentro desta secção, vários aspetos relativos à forma como o utilizador final visualiza a aplicação podem ser alterados, nomeadamente o nome da mesma, os ícones e a versão a que a *build* atual representa. Para além da customização, também é importante garantir que a aplicação é capaz de aceder a todas as “Capabilites” do dispositivo em questão que necessita para poder funcionar. As “Capabilites” que esta aplicação necessita são: “InternetClient”, “InternetClientServer”, “PrivateNetworkClientServer”, “WebCam” e “Microphone”.

Para concluir o processo de *Build* é apenas necessário seleccionar a opção “Build”. Na Figura 40 é possível visualizar a cena escolhida, tal como a plataforma e as características da aplicação.

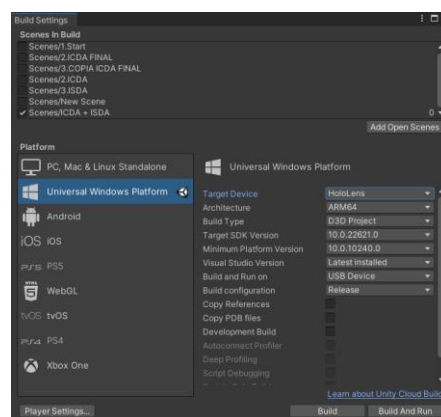


Figura 40: *Build Settings* da aplicação

Após selecionar a opção *Build*, o Editor cria uma pasta, neste caso “Build_Final”, onde podemos encontrar a aplicação construída. É importante referir que a ordem pela qual as cenas são carregadas na aplicação depende da ordem em que estas se encontram na secção “Build Settings”, sendo que nesta secção as cenas são organizados por ordem numérica e, em seguida, por ordem alfabética, embora seja possível organizar manualmente as cenas desejadas ao selecionar e arrastar as mesmas para a posição desejada.

4.6.2 *Visual Studio 2022* (Deployment)

Após a construção da aplicação é necessário realizar o *Deployment* da aplicação no dispositivo *Hololens 2*. Para tal, é necessário aceder à pasta criada através do processo de *Build* e abrir o ficheiro da solução criada, neste caso, “Tese.sln”, no *Visual Studio 2022*. Após abrir a solução criada no *VS 2022*, é necessário selecionar as opções “Release” (inicialmente “Debug”) e “ARM64” (inicialmente “x64”), localizadas no centro superior da aplicação. Uma vez que o dispositivo *Hololens 2* foi conectado ao PC através de um cabo *USB*, é também necessário selecionar a opção “Dispositivo” (inicialmente “Computador Local”). Por fim, para realizar o *Deployment*, foi selecionada a opção “Iniciar sem Depurar”, encontrada dentro da secção “Depurar”, localizada imediatamente acima da opção “ARM64” selecionada.

Caso seja a primeira vez que o dispositivo *Hololens 2* esteja a ser utilizado para instalar uma aplicação não nativa ou caso seja a primeira vez que o PC onde se está a correr o processo de *Deployment* se esteja a conectar com o dispositivo *Hololens 2* em questão, serão necessárias realizar algumas ações prévias neste mesmo dispositivo. Independente do caso, será necessário, dentro do dispositivo *Hololens*, ativar a opção “Developer mode”. Para o primeiro caso, nenhuma ação adicional é necessária. Para o segundo caso, será necessário selecionar a opção “Pair” localizada dentro da secção “For Developers”. Após selecionar esta opção, um código composto por seis números é gerado pelo dispositivo, sendo necessário inserir este mesmo *PIN* no *VS 2022* durante o processo de *Deployment*. Na Figura 41 é possível visualizar o processo em questão.

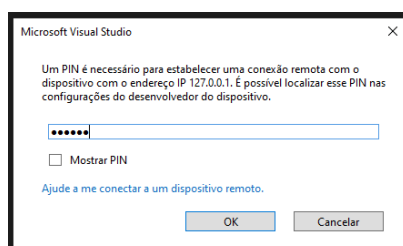


Figura 41: Conectar PC ao dispositivo *Hololens*

4.6.3 Aplicação no *Hololens 2*

Após acabar o processo de compilação, a aplicação abre automaticamente no dispositivo conectado, não sendo necessário realizar qualquer ação adicional. Caso o utilizador saia da aplicação ou aceda ao menu principal, este poderá abrir novamente a aplicação ao entrar na secção “All Apps” (localizada no menu principal), onde todas as aplicações instaladas estão presentes, e seleccionar a aplicação desejada.

5. SOLUÇÃO FINAL

Nesta Capitulo é apresentada a aplicação final criada para o dispositivo *Hololens 2*. Infelizmente, como já referido na secção do capítulo anterior referente às complicações agravadas, o *package MediaPipeUnityPlugin* não é compatível com este dispositivo, não sendo então possível utilizar a funcionalidade de deteção e classificação dos sinais de LGP através do mesmo. Apesar deste aspeto, a aplicação criada apresenta todas as funcionalidades esperadas quando utilizada no Editor onde foi criada, sendo a partir desta que a aplicação final é descrita nesta secção.

5.1 Fluxograma da solução

A solução final criada em muito varia da solução inicialmente pensado, apresentando uma estrutura completamente diferente da inicialmente descrita. Na Figura 42 é possível visualizar o diagrama da solução final, onde apenas uma interface é representada, ao contrário das três iniciais. Este diagrama também apresenta quatro componentes diferentes (Unity, Microsoft Azure, Servidor Proxy e Servidor teste CCG) e não apenas um (Unity) devido, como já referido, à impossibilidade de realizar todas as tarefas necessárias através do Unity.

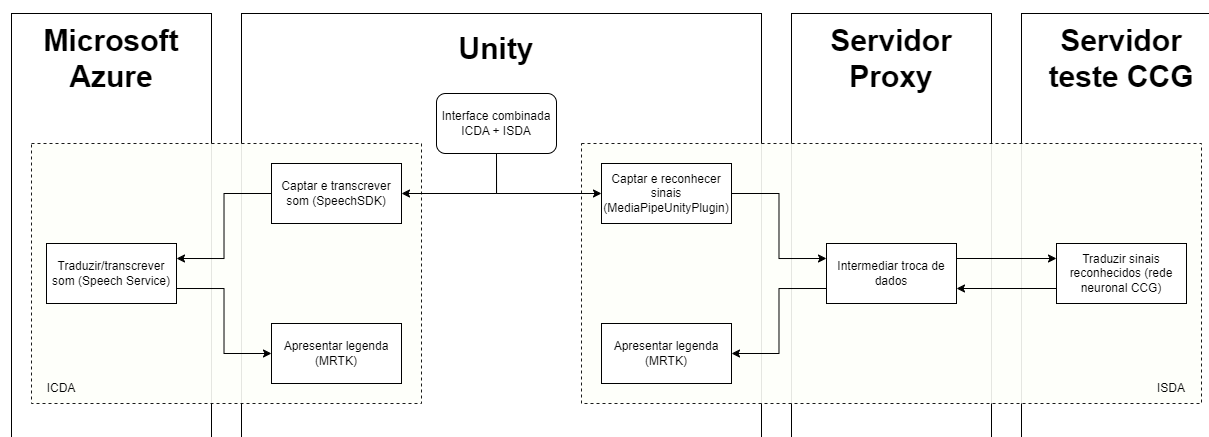


Figura 42: Diagrama final da solução

5.2 Componentes finais

Neste secção são apresentados todos os componentes que compõem a solução final criada. Esta apresentação é realizada através de uma breve descrição da função de cada um destes componentes e uma ou mais figuras, quando adequado.

5.2.1 Interface Partilhada

No que toca à Interface Partilhada, criada a partir da junção das Interfaces ICDA e ISDA iniciais, esta é composta por um pequeno menu de quatro botões e por um campo de texto e é a primeira coisa que o utilizador vê mal entra na aplicação. Cada um dos quatro botões referidos pode ser selecionado pelo utilizador para realizar uma dada ação.

O primeiro botão deste menu, intitulado “Cap. Voz”, representa a secção da interface ICDA inicialmente criado, sendo então responsável pela funcionalidade *Speech-To-Text* desta aplicação.

O segundo botão, intitulado “Cap. Sinais”, representa a secção da interface ISDA inicialmente criada, sendo então responsável pela funcionalidade de deteção e classificação de sinais desta aplicação.

O terceiro botão, intitulado “Ajuda”, representa o menu de ajuda criado, sendo então responsável por informar o utilizador das diferentes funcionalidades da aplicação tal como a forma como estas podem ser acedidas e utilizadas.

O quarto e último botão, intitulado “Sair”, representa o menu de saída inicialmente criado, sendo então responsável por confirmar a intenção do utilizador de sair da aplicação.

Para finalizar, o campo de texto, inicialmente preenchido com “Começar Tradução” é onde o texto resultante de cada uma das funcionalidades é apresentado.

5.2.2 ICDA

Relativamente ao primeiro botão, que apresenta como nome “Cap. Voz”, ou Captação de Voz, este é utilizado para abrir o painel das definições da funcionalidade *Speech-To-Text* implementada na aplicação. Este painel permite ao utilizador selecionar, a partir de um menu *dropdown*, a língua que este deseja detetar e permite começar ou terminar o processo de deteção de voz. Durante o uso deste painel, mais nenhum dos restantes três botões podem ser selecionados, sendo primeiro necessário fechar o painel em questão para se poder selecionar outro destes quatro botões.

O texto resultante desta funcionalidade tanto aparece no painel das definições em si como no próprio campo da Interface Partilhada destinada para este mesmo propósito. Na Figura 43 é possível visualizar o painel das definições criado.



Figura 43: Menu de definições ICDA

5.2.3 ISDA

Relativamente ao segundo botão, que apresenta como nome “Cap. Sinais”, ou Captação de Sinais, este é utilizado para tornar visível a captura de câmara do dispositivo em questão. Para além de tornar visível a captura de imagem, também começa o processo de identificação e classificação dos sinais detetados, mudando o conteúdo da caixa de texto principal consoante o sinal realizado para a câmara. Na Figura 44 é possível visualizar a Interface Partilhada com a representação das *landmarks* identificadas pelo *MediaPipeUnityPlugin* e na Figura 45 é possível visualizar o utilizador da aplicação e o texto gerado pelo processo de tradução de língua gestual.

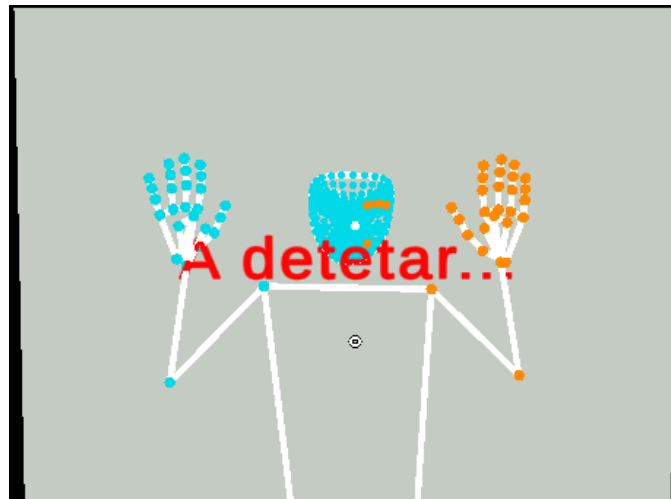


Figura 44: Elementos *Mediapipe*

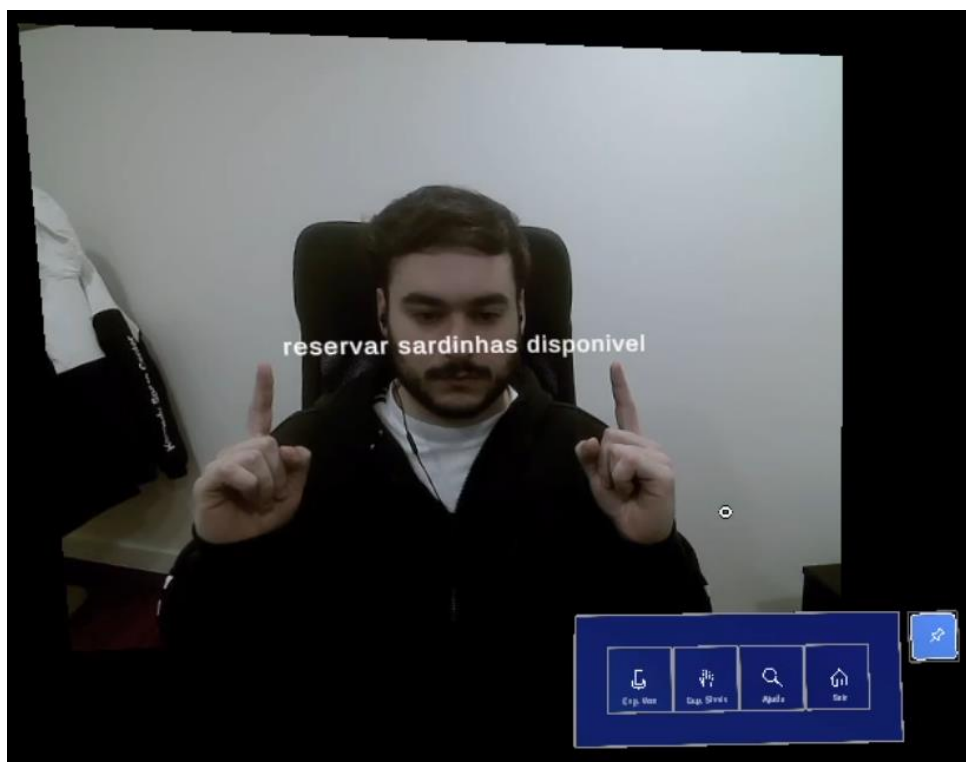


Figura 45: Funcionamento da aplicação

5.2.4 Menu de Ajuda

Relativamente ao terceiro botão, que apresenta como nome “Ajuda”, este é utilizado para abrir o menu de ajuda da aplicação. Este menu é dividido em três, apresentando para cada um dos outros botões uma secção de ajuda sucinta, onde é explicado o que cada um destes faz, como utilizar as janelas geradas ao seleccioná-los e o que se espera obter ao utilizar cada um. Na Figura 46 é possível visualizar o menu de ajuda criado.

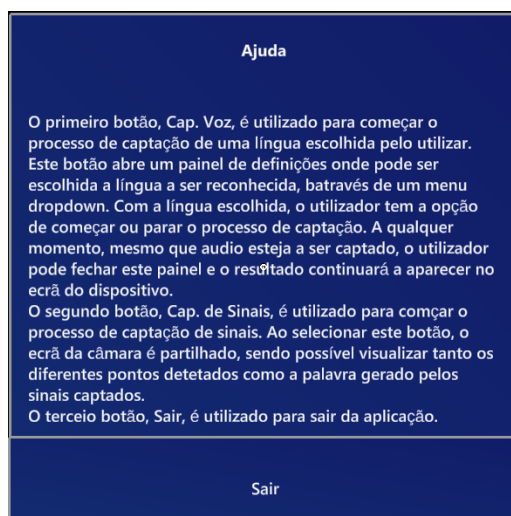


Figura 46: Menu de ajuda

5.2.5 Menu de Saída

Relativamente ao quarto e último botão, que apresenta como nome “Sair”, este é utilizado para abrir o menu de saída da aplicação. O menu de saída da aplicação foi criado com a intenção de garantir que o utilizador realmente deseja sair da aplicação e que não selecionou esta opção apenas por engano. Através deste menu, é possível continuar o processo de saída da aplicação ou voltar para a aplicação. Na Figura 47 é possível visualizar o menu de saída criado.

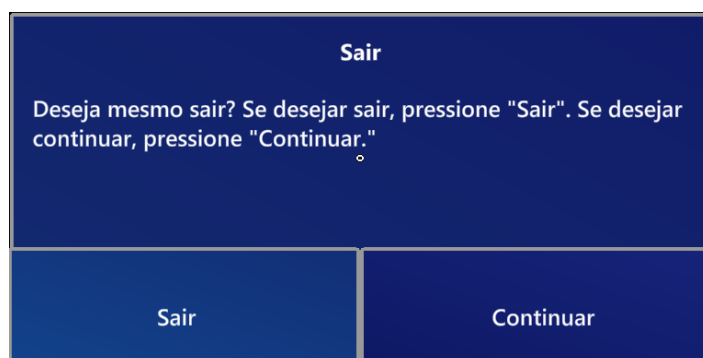


Figura 47: Menu de saída

6. AVALIAÇÃO DA SOLUÇÃO

Neste sexto capítulo do documento é realizada a avaliação da aplicação criada. Este capítulo é dividido em duas secções, Teste da Solução e Comparação com Soluções já Existentes.

Nesta primeira secção, Teste da Solução, são realizados seis testes para averiguar se a aplicação criada mantém o comportamento esperado no caso de se deparar com condições não ideais de funcionamento, sejam estas relativas à ligação de *Internet* ou ao espaço onde esta é utilizada.

Na segunda secção, Comparação com Soluções já Existentes, é comparada a aplicação criada com as soluções previamente analisadas e comparadas. Esta secção serve para melhor compreender as vantagens e desvantagens que esta aplicação apresenta quando se trata da prestação de auxílio a indivíduos com dificuldades auditivas relativamente às poucas soluções previamente analisadas.

6.1 Teste da Solução

Com o desenvolvimento da aplicação final concluído, foram realizados alguns testes de maneira a averiguar algumas das limitações que esta aplicação pode apresentar caso as condições ideais para o seu uso não se reúnam. Esta secção foi dividida em seis subsecções, cada uma referente a um teste diferente realizado à aplicação. Na primeira secção foi testada a aplicação numa situação em que a ligação de *Internet* estabelecida é fraca. De seguida, foi testada a aplicação num ambiente com elevada luminosidade e num ambiente com reduzida luminosidade. Na quarta subsecção foi testada a aplicação no caso de existir a presença de mais do que uma pessoa na imagem da câmara utilizada. Na quinta secção, foi testada a aplicação no caso de duas pessoas falarem ao mesmo tempo durante a funcionalidade *Speech-to-Text*. Por fim, foi testado o funcionamento da aplicação num ambiente ideal.

6.1.1 Ligação de *Internet* Fraca

Este primeiro teste pretende avaliar o comportamento da aplicação caso esta se depare com uma ligação à *internet* fraca. Para tal, foi posicionado o dispositivo que utiliza esta aplicação em três pontos diferentes, cada uma mais afastado da fonte de *Internet* do que o anterior.

No caso do primeiro ponto, relativamente próximo à fonte de *internet*, a aplicação apresentou o comportamento esperado.

No caso do segundo ponto, onde a ligação à fonte de *internet* já não poderia ser considerado forte, a aplicação apresentou surpreendentemente um comportamento quase idêntico ao identificado no primeiro ponto.

No caso do terceiro ponto, onde a ligação à fonte de internet era considerado fraca, a funcionalidade *Speech-to-Text* não era capaz de funcionar a uma velocidade aceitável, apresentando também falhas na deteção da voz. Neste caso, houve também a ocorrência de alguns erros durante o correr da aplicação, o que levavam ao não funcionamento da aplicação. Os erros identificados estão representando na Figura 48, sendo ambos relacionados com problemas na realização da conexão com o serviço “Speech Service”.

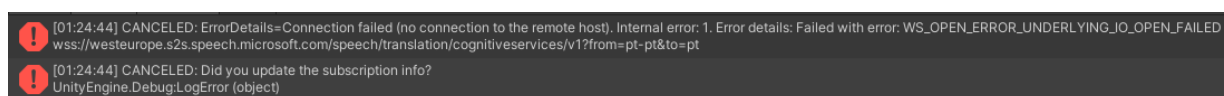


Figura 48: Erro de conexão

Com base neste teste, é importante manter uma ligação estável de *internet* para garantir o bom funcionamento desta aplicação, embora esta ligação não tenha de ser extremamente forte.

6.1.2 Luminosidade Elevada

Este segundo teste pretende avaliar a aplicação no caso desta ser utilizada num ambiente extremamente luminoso, o que pode dificultar a captação da imagem e reconhecimento das partes do corpo. Para recriar este ambiente, foram utilizadas duas lanternas distintas, ambas apontadas à câmara do dispositivo. Estas lanternas começaram inicialmente lado a lado com o utilizador, sendo posteriormente aproximadas ao ponto de quase ocuparem o ecrã da aplicação.

Durante todo o teste, a aplicação continua a ser capaz de detetar com precisão as partes do corpo e as diferentes *landmarks* identificadas em cada parte. A aplicação apenas apresenta dificuldade a identificar o utilizador caso as lanternas em si obstruam o utilizador e não a luz emitida por estas.

Com este teste, garante-se que a aplicação pode ser utilizada em ambientes extremamente luminosos.

6.1.3 Luminosidade Reduzida

Este terceiro teste pretende avaliar a aplicação no caso desta ser utilizada num ambiente muito pouco luminoso, o que pode dificultar a captação da imagem e reconhecimento das partes do corpo. Para recriar este ambiente, foi utilizada a aplicação durante a noite num espaço sem luminosidade.

Inicialmente a aplicação não é capaz de detetar qualquer parte do corpo. Com um pequeno aumento da luminosidade no espaço utilizado, a aplicação é capaz de detetar qualquer parte do corpo que deveria detetar.

Com este teste, confirma-se a necessidade por parte da aplicação de ser utilizada num ambiente com alguma fonte de luz, mesmo que esta fonte seja bastante fraca. Dependendo do dispositivo utilizado, este aspeto poderia não ser um problema com o uso deste dispositivo como fonte de luz.

6.1.4 Imagem com Duas Pessoas

Relativamente ao quarto teste, este é realizado para avaliar o comportamento da aplicação caso mais do que uma pessoa se encontre à frente da câmara.

No caso de duas ou mais pessoas se encontrarem em frente da câmara do dispositivo usado pela aplicação para captar vídeo, apenas as partes do corpo de uma das pessoas é identificada, normalmente a que se encontra mais próxima da câmara. Normalmente, a escolha da pessoa a ser identificada é realizada com base na facilidade de reconhecer as diferentes partes do corpo que a aplicação está à procura dessa mesma pessoa, sendo a proximidade à câmara um dos fatores que mais influencia a escolha da pessoa a ser identificada.

Caso ambos os utilizadores se encontrarem lado a lado, muito perto um do outro, a aplicação pode ter dificuldade a identificar os pontos da cara do indivíduo escolhido, alternando entre os dois indivíduos, embora as mãos e a postura não sofram alterações.

Caso o utilizador identificado não seja o desejado, este terá de sair do alcance da câmara por completo para o outro utilizador ser reconhecido.

Com este teste, confirma-se ser recomendado apenas captar com a câmara do dispositivo um único indivíduo no início da aplicação de maneira a garantir o bom funcionamento da aplicação, embora mais do que um indivíduo possa aparecer na câmara do dispositivo sem ocorrerem anomalias.

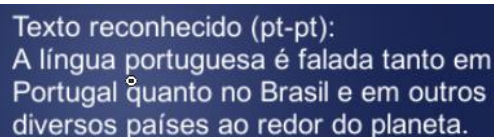
6.1.5 *Speech-to-Text* com Duas Pessoas

No que toca ao quinto teste, este foi realizado para verificar o comportamento da aplicação no caso de duas pessoas comunicarem verbalmente ao mesmo tempo durante a captação de voz. Neste teste, foram consideradas duas situações distintas. A primeira e talvez mais comum é a ocorrência de barulho de fundo durante o processo de captação de voz, onde a segunda fonte de voz se encontra algo afastada do dispositivo onde a aplicação está a ser utilizada. A segunda situação, menos comum, passa pela

tentativa de comunicar com dois indivíduos diferentes ao mesmo tempo, o que pode levar a respostas em simultâneo.

Para ambas as situações, durante o correr da aplicação cada indivíduo leu um pequeno excerto, sendo o utilizado pelo indivíduo com quem o utilizador pretende comunicar “A Língua Portuguesa é falada tanto em Portugal, quanto no Brasil e em outros diversos países ao redor do planeta” e o utilizado pelo indivíduo a imitar barulho de fundo “Quem se interessa por aprender a falar Português já pode contar com um ensino eficiente”.

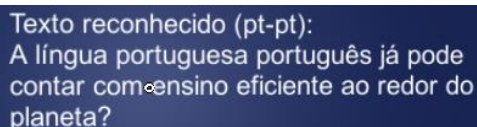
Na primeira situação, quando o segundo indivíduo se encontra longe do dispositivo utilizado, a aplicação costuma apresentar apenas o texto do indivíduo com quem o utilizador pretende comunicar. Este texto pode ser visualizado na Figura 49.



Texto reconhecido (pt-pt):
A língua portuguesa é falada tanto em
Portugal quanto no Brasil e em outros
diversos países ao redor do planeta.

Figura 49: Texto quando o segundo utilizador está longe

Na segunda situação, quando o segundo indivíduo está mais perto do dispositivo, a aplicação tem dificuldade a distinguir a voz do indivíduo com quem o utilizador quer comunicar e a voz do segundo indivíduo, como constatando na Figura 50.



Texto reconhecido (pt-pt):
A língua portuguesa português já pode
contar com ensino eficiente ao redor do
planeta?

Figura 50: Texto quando o segundo utilizador está perto

Com este teste, confirma-se a necessidade de a aplicação ser utilizada num espaço pouco ruidoso, onde o número mínimo de vozes possam ser ouvidas. Embora a voz de outro indivíduo possa não afetar a captação de voz do indivíduo desejado no caso deste primeiro se encontrar longe, caso este primeiro indivíduo se aproxime ou se falar mais alto a captação de voz pode ser significativamente alterada.

6.1.6 Condições ideais

No que toca ao último teste, este foi realizado para verificar o comportamento da aplicação no caso desta ser utilizada num ambiente ideal, ou seja, num ambiente com luminosidade moderada, sem ruído de fundo, com boa conexão à internet e com apenas uma única pessoa à frente da câmara.

Dentro destas condições, o comportamento registado pela aplicação é o esperado: não apresenta falhas nem quebras durante o funcionamento da mesma, tanto para a funcionalidade de reconhecimento de

sinais como para a funcionalidade de reconhecimento de voz. No capítulo 6, referente à demonstração da aplicação final, é possível visualizar o comportamento da aplicação num ambiente ideal.

6.2 Comparação com Soluções já Existentes

Com o desenvolvimento da aplicação concluído, é possível agora avaliar a solução com base nos fatores inicialmente utilizados para avaliar as quatro soluções analisadas. Os fatores em questão são a Portabilidade da solução, o Custo da solução, a Disponibilidade, o número de línguas que suporta (Gestuais ou Faladas), a Facilidade de interação, a Comodidade (tanto de ICDA como de ISDA) e por fim a Facilidade de Uso. Estes sete fatores, expostos inicialmente no capítulo referente à revisão de literatura, são utilizados neste projeto para tentar avaliar as vantagens e desvantagens de cada aplicação e para permitir compará-las. Apesar de serem comparadas, tanto as quatro soluções previamente analisadas como a solução desenvolvida apresentam vantagens distintas em diversas áreas, o que faz com que cada uma destas soluções seja mais ideal que as restantes numa dada situação. Uma vez que a situação em que o utilizador da solução se depara influencia muito a solução que este deve adotar, é necessário não excluir nenhuma das soluções devido apenas à classificação total que este registou nesta análise. Seguindo o método adotado no capítulo da revisão de literatura, a solução criada será primeiro analisada individualmente, através da criação da Tabela 7, onde é descrita com algum detalhe a aplicação no que toca aos sete diferentes fatores utilizados para a avaliar. A partir destas descrições, poderá ser então atribuído um valor entre um e cinco para cada um dos fatores, tendo em conta os critérios expostos no capítulo da revisão de literatura.

Tabela 7: Análise detalhada da solução criada

Solução	Aplicação desenvolvida
Portabilidade	Utiliza o dispositivo <i>Hololens 2</i> , sendo fácil o realizar o transporte da mesma.
Custo	Solução não comercial, embora os óculos custem entre 3260-4840€
Disponibilidade	Solução apenas disponível para investigação
Línguas (LG e faladas)	É captada a língua portuguesa, espanhola ou francesa e posteriormente traduzida para português ou é captada e traduzida a LGP para português
Facilidade de Interação	Tanto o utilizador com dificuldades auditivas como o utilizador sem dificuldades auditivas têm apenas de abrir a aplicação e selecionar um botão para começar o processo de tradução.
Comodidade ISDA	Apenas tem de descarregar a aplicação e realizar uma ação simples

Comodidade ICDA	Apenas tem de descarregar a aplicação e realizar uma ação simples
Facilidade de uso	Fácil de usar, sendo apenas necessário abrir a aplicação e seleccionar o botão da interface desejada para ambos os tipos de utilizador

Com a descrição de cada um dos fatores a serem analisados realizada, é então possível comparar as cinco soluções analisadas de maneira a averiguar os pontos fortes e fracos de cada uma e identificar a solução que melhor auxilia a comunicação entre indivíduos sem dificuldades auditivas e indivíduos com dificuldades auditivas. Na Tabela 8 é possível visualizar a comparação entre as cinco soluções distintas, sendo a cinzento representada a pontuação mais alta de cada um dos fatores e a verde a pontuação Total e a pontuação Total sem Custo de Disponibilidade mais elevadas.

Tabela 8: Comparação das cinco soluções

Solução	Aplicação criada	BrightSign Glove	SignAll Chat	Hand Talk	Holographic Sign Language Interpreters
Portabilidade	4	4	2	5	4
Custo	3	4	1	5	3
Disponibilidade	2	4	3	5	1
Línguas	LG	1	5	1	2
	Faladas	3	5	1	2
Facilidade de interação	4	3	3	1	1
Comodidade	ISDA	4	5	4	5
	ICDA	4	2	3	1
Facilidade de uso	5	2	4	4	5
Total (35)	24	25.5	17.5	24.5	20.5
Total sem Custo e Disponibilidade (25)	19	17.5	13.5	14.5	16.5

A partir da tabela criada, é possível ver que a solução desenvolvida apresenta uma classificação Total igual a 24, o que a coloca em terceiro, a apenas 1.5 pontos do primeiro. O facto desta solução não apresentar uma avaliação mais alta deve-se ao número mais reduzido de línguas que esta é capaz de

utilizar ou traduzir, ao custo algo elevado, considerando o custo do dispositivo *Hololens*, e ao facto desta solução não ser disponibilizada ao público para uso.

No caso de não serem tidos em conta o Custo do dispositivo utilizado e a Disponibilidade desta solução ao público, esta solução passa a apresentar a avaliação mais alta, sendo esta de 19 pontos, 1.5 pontos acima do segundo. Esta classificação mais elevada comparativamente às restantes soluções deve-se sobretudo ao facto desta solução ser capaz de acomodar tanto ISDA como ICDA, o que facilita o processo de comunicação entre estes dois indivíduos, exatamente o aspeto que se pretendia avaliar.

Com esta análise podemos concluir que caso o utilizador apresenta-se dificuldades auditivas e quisesse comunicar com um indivíduo sem dificuldades auditivas, ou vice-versa, a melhor solução a adotar seria provavelmente a desenvolvida, considerando claro que o utilizador é capaz de aceder a esta solução e utiliza a língua portuguesa ou LGP para comunicar.

7. CONCLUSÃO E TRABALHO FUTURO

Neste último capítulo do documento é apresentada a conclusão final relativa a todo o trabalho realizado durante a elaboração deste projeto. Para além de apresentar a conclusão final do trabalho, este capítulo, na secção Trabalho Futuro, apresenta diferentes possibilidades que podem ser adotadas futuramente para melhorar a aplicação desenvolvida, nomeadamente a adoção de uma plataforma diferente.

7.1 Conclusão

Com a criação deste trabalho, o objetivo fundamental foi sempre melhor compreender a situação que indivíduos com dificuldades auditivas acrescidas enfrentam na sociedade, as diferentes alternativas adotadas pela sociedade para inserir estes mesmos indivíduos na sociedade e a criação de uma ferramenta realmente capaz de facilitar a inserção destes mesmo indivíduos na sociedade. De uma maneira geral, considero que este objeto foi cumprido.

No que toca à compreensão dos indivíduos da sociedade que se deparam com problemas auditivos e da sua língua, comumente referida como Língua Gestual Portuguesa, considero que através deste trabalho é possível melhor compreender a necessidade de promover a inserção deste indivíduos e a necessidade de reconhecer estas mesmas Línguas Gestuais como métodos legítimos de comunicação, algo que infelizmente nem sempre é realizado dada a sua origem comunitária e ao processo pouco burocrático que é realizado para as criar.

No que toca às diferentes alternativas adotadas pela sociedade, considero que as alternativas tecnológicas expostas neste documento retratam a maioria se não a totalidade das diferentes soluções adotadas pela sociedade de hoje em dia para auxiliar a inserção social de indivíduos com dificuldades auditivas. A exposição detalhada de quatro das soluções atualmente mais avançadas ou inovadoras que pretendem auxiliar estes mesmo indivíduos com dificuldades auditivas no processo de comunicação, onde apresentam mais dificuldade, é essencial para averiguar as necessidades destes indivíduos que realmente estão a ser atendidas ou que futuramente serão atendidas, sendo ainda mais essencial para averiguar as necessidades que não são atendidas e no futuro próximo provavelmente também não serão atendidas.

No que toca à criação de uma ferramenta capaz de facilitar a inserção de indivíduos com dificuldades acrescidas na sociedade, considero que a aplicação criada neste projeto, caso esta seja incorporada num dispositivo real e distribuída livremente, seja capaz de cumprir este objetivo gratificante. A aplicação

desenvolvida neste projeto em muito facilitaria o processo de comunicação entre dois indivíduos, um que apresente dificuldades auditivas e outro que não, o que iria promover positivamente a inserção destes indivíduos com dificuldades, sendo a mais premente justamente a dificuldade de comunicação, caso esta fosse ligeiramente adaptada e disponibilizada livremente.

Com a Revisão de Literatura realizada neste projeto, considero que é possível compreender o maior grau de complexidade das línguas gestuais quando comparadas a outras línguas, causado pelo facto de serem criadas com base numa dada comunidade e de normalmente não existirem ou não serem adotadas línguas gestuais criadas pelo governo. Felizmente, Portugal reconhece a Língua Gestual Portuguesa como uma língua legítima de comunicação, sendo até considerada a Língua Gestual oficial do país.

Com o desenvolvimento da aplicação, embora ache que esta ainda necessite de algumas alterações para realmente ser viável utilizar livremente por indivíduos com dificuldades auditivas e indivíduos sem dificuldades auditivas, considero que esta demonstra exatamente o ponto que se esperava demonstrar quando foi inicialmente idealizada: ferramentas para o auxílio de indivíduos com dificuldades auditivas não precisam de mais avanços tecnológicos para serem viáveis, as ferramentas de desenvolvimento atualmente existentes mais que permitem a criação de diferentes ferramentas de auxílios para estes indivíduos, sendo possível, com o apoio de entidades competentes, desenvolver e distribuir estas mesmas ferramentas para os indivíduos que realmente precisam.

Por fim, com base neste trabalho, convido o leitor a apoiar não o desenvolvimento de ferramentas que facilitam a vida à maioria, mas sim o desenvolvimento de ferramentas que proporcionem uma vida digna à minoria.

O desenvolvimento de ferramentas de auxílio para os indivíduos que realmente necessitam é crucial e um dos passos que devemos seguir para tornar a sociedade atual realmente inclusiva.

7.2 Trabalho futuro

No que toca à Revisão de Literatura, dois aspetos poderiam ser adicionados no futuro de maneira a adicionar valor ao trabalho criado. O primeiro aspeto que poderia ser adicionado no futuro é uma análise das diferentes ações adotadas pela sociedade no futuro e que não foram documentadas neste trabalho. É essencial expor novas iniciativas e ferramentas que podem auxiliar pessoas com dificuldades acrescidas de maneira a estas serem mais facilmente adotadas por quem realmente precisa. O segundo aspeto que pode ser analisado no futuro são as ferramentas utilizadas pelo povo português para ajudar a inserção de indivíduos com dificuldades auditivas na sociedade. Embora esta análise tenha sido

realizada, chegou-se à conclusão que existe apenas um único sistema em português capaz de auxiliar estes indivíduos, sendo este utilizado em apenas três *websites* diferentes de Municípios, e que apenas uma outra aplicação se encontra em desenvolvimento, utilizando como base este mesmo sistema.

Relativamente agora à aplicação, dado os diversos contratempos enfrentados durante o desenvolvimento desta e as inúmeras alterações que tiveram de ser realizadas para permitir o desenvolvimento da aplicação criada, alguns aspetos desta aplicação não correspondem ao que se esperava obter inicialmente, não sendo cumpridos dois dos requisitos inicialmente estipulados, algo que pode ser melhorado no futuro.

O primeiro requisito não cumprido foi o sistema criado deve ser independente, ou seja, não necessita de serviços, aplicação ou ferramenta que não se encontre dentro da própria aplicação criada. Este requisito não pode ser cumprido devido a duas razões distintas. A primeira razão foi a necessidade de adotar um serviço externo para a funcionalidade *Text-to-Speech*, que inicialmente era realizada pelo próprio dispositivo, mas, devido ao facto de o serviço utilizado pelo dispositivo não suportar português, não foi possível utilizar o serviço inicialmente previsto. A segunda razão foi o facto da rede neuronal utilizada para classificar os dados não poder ser utilizada dentro do editor onde foi criada a aplicação, uma vez que esta apresenta comportamento dinâmico, algo não suportado pelo formato de ficheiro suportado (*ONNX*).

Mesmo no futuro, a incompatibilidade da funcionalidade para com português é algo que se espera que mantenha, não havendo outra forma de permitir a existência desta funcionalidade através de algo que não uma ferramenta externa.

Relativamente à segunda razão, para permitir o cumprimento deste requisito, seria necessário obrigatoriamente atualizar o projeto para uma versão de *Unity* igual ou superior à 2022.3, o que em si implica o uso do *package MRTK 3*, ainda em desenvolvimento, em vez do *package MRTK 2*. Com o uso desta versão *Unity*, é possível utilizar modelos não só do tipo *ONNX*, mas também do tipo *TensorFlow*, o que permitiria utilizar a rede neuronal diretamente no editor, sem necessitar de um servidor externo. É importante referir que com a mudança da versão do *package MRTK*, poderá ser necessário voltar a desenvolver toda a interface criada para a aplicação.

Relativamente ao segundo requisito que não pode ser cumprido, sendo este a criação de uma aplicação compatível com os óculos *Hololens*, este não pôde ser cumprido porque o *package MediaPipeUnityPlugin*, a única forma de utilizar o *MediaPipe* diretamente no editor *Unity*, tornando assim a aplicação não dependente de sistemas externos, é compatível com *Windows 10*, mas não com o sistema operativo criado com base no *Windows 10* e utilizado pelo dispositivo *Hololens*, *Windows*

Holographic. Devido a esta incompatibilidade de *software*, a não ser que o *package* em questão seja atualizado para suportar este sistema operativo específico, o que é extremamente improvável, seria necessário adotar outro dispositivo capaz de hospedar a soluções.

Durante a fase final de desenvolvimento, foi contemplado o uso de um dispositivo móvel *android* como substituto ao dispositivo *Hololens*, uma vez que estes já são compatíveis com o *package* em questão. O dispositivo *android* contemplado foi o *Oculus Quest* visto este utilizar *android* e ser também um dispositivo de realidade mista, embora algo diferente do dispositivo *Hololens*. A adoção deste pode ser uma solução futura para evitar este problema de compatibilidade, embora seja necessário, no mínimo, alterar a forma como o *package MediaPipeUnityPlugin* é utilizado. Neste *package*, seria necessário passar a utilizar o *GPU* em vez do *CPU*, algo que exige a adição de diferentes excertos de código ao *script* criado e, acima de tudo, faz com que a cena criada não possa ser corrida e testada em *Unity* porque o uso do *GPU* não é suportado para dispositivos *Windows*. Ao não ser possível correr a cena criada em *Unity*, será necessário exportar o projeto sempre que seja preciso testar o funcionamento da aplicação, sendo também necessário o uso de ferramentas externas para analisar este mesmo funcionamento. Outro aspeto extremamente importante a ter em conta é o facto do *package* atualmente utilizado ter sido criado para *Windows 10*, o que faz com que este não só não apresente elementos necessários para o *package* funcionar em *android*, como também não é possível criar o *package* neste sistema operativo de maneira a incluir esses mesmos elementos. A solução para este problema seria utilizar a versão completa do *package* disponibilizado pelo autor, sendo então necessário recriar a aplicação criada num novo projeto *Unity* onde este *package* completo estaria presente.

Um aspeto algo pequeno e de pouca relevância que poderia ser facilmente adicionado à aplicação seria a leitura do texto detetado a partir dos sinais, o que faria com que a conversa entre os dois indivíduos fosse mais parecida com qualquer outra conversa realizada entre dois indivíduos da mesma sociedade. Para finalizar, outro aspeto que poderia ser alterado no futuro para valorizar a aplicação criada seria a remoção da funcionalidade *Speech-to-Text* como um todo, sendo criado em vez o oposto do que foi realizado na funcionalidade de deteção e classificação de sinais, ou seja, em vez de captar som e apresentar texto, fazer com que a aplicação capte som e transforme esse som em sinais. Para tal, seria necessário, entre outras coisas, um repositório de sinais de LGP capaz de representar um número considerável de sinais.

Caso todas as alterações em questão fossem realizadas, seria possível adaptar a solução atual de maneira a esta não ter de depender de qualquer ferramenta externa e passaria a ser compatível com um dispositivo de realidade aumentada, embora não o inicialmente desejado.

REFERÊNCIAS

- Abougarair, A. J. (2022). Smart glove for sign language translation Modelling and different Controllers Design for Stabilizing Nonlinear System View project Adaptive Traffic Light Dynamic Control Based on Road Traffic Signal from Google Maps View project. *Article in International Journal of Robotics and Automation*. <https://doi.org/10.15406/iratj.2022.08.00253>
- ACM, K. K.-C. of the, & 2018, undefined. (2018). Technology for the deaf. *DI.Acm.Org*, 61(12), 16–18. <https://doi.org/10.1145/3283224>
- Assistive Technology Enters the Classroom With Holographic Sign Language Interpreters - ACM SIGGRAPH Blog*. (n.d.). Retrieved February 12, 2023, from <https://blog.siggraph.org/2022/07/assistive-technology-enters-the-classroom-with-holographic-sign-language-interpreters.html/>
- Bauman, H. D. L. (2004). Audism: Exploring the Metaphysics of Oppression. *The Journal of Deaf Studies and Deaf Education*, 9(2), 239–246. <https://doi.org/10.1093/DEAFED/ENH025>
- BrightSign - Translate any sign into any language*. (n.d.). Retrieved February 10, 2023, from <https://www.brightsignglove.com/>
- CITY-, S. B.-F. M.-K., & 1999, undefined. (2019). Clinical and cultural issues in caring for deaf people. *Researchgate.Net*. https://www.researchgate.net/profile/Steven-Barnett-4/publication/13301062_Clinical_and_cultural_issues_in_caring_for_Deaf_people/links/5d1f65c2458515c11c14fc4d/Clinical-and-cultural-issues-in-caring-for-Deaf-people.pdf
- Compramos a ProDeaf, nossa concorrente na tradução para Libras!* (n.d.). Retrieved February 12, 2023, from <https://www.handtalk.me/br/blog/handtalk-prodeaf/>
- Diaz, C., Arellano, M., Rosillo, V., Sci., A. O.-Res. Comput., & 2018, undefined. (2018). Augmented Reality System to Promote the Inclusion of Deaf People in Smart Cities. *Rcs.Cic.Ipn.Mx*, 147(2), 49–64. https://www.rcs.cic.ipn.mx/2018_147_2/Augmented%20Reality%20System%20to%20Promote%20the%20Inclusion%20of%20Deaf%20People%20in%20Smart%20Cities.pdf
- Ding, L., & Martinez, A. M. (2009). Modelling and recognition of the linguistic components in American Sign Language. *Image and Vision Computing*, 27(12), 1826–1844. <https://doi.org/10.1016/J.IMAVIS.2009.02.005>

- Eberhard, M., D., F. Simons, G., & D. Fennig, C. (n.d.). *Ethnologue: Languages of the World*. Retrieved February 12, 2023, from [https://www.google.com/search?q=Eberhard%2C+David+M.%2C+Gary+F.+Simons%2C+and+Charles+D.+Fennig+\(eds.\).+2022.+Ethnologue%3A+Languages+of+the+World.+Twenty-fifth+edition.+Dallas%2C+Texas%3A+SIL+International&rlz=1C1GCEA_enPT1012PT1012&oq=Eberhard%2C+David+M.%2C+Gary+F.+Simons%2C+and+Charles+D.+Fennig+\(eds.\).+2022.+Ethnologue%3A+Languages+of+the+World.+Twenty-fifth+edition.+Dallas%2C+Texas%3A+SIL+International&aqs=chrome..69i57j69i60l2.674j0j4&sourceid=chrome&ie=UTF-8](https://www.google.com/search?q=Eberhard%2C+David+M.%2C+Gary+F.+Simons%2C+and+Charles+D.+Fennig+(eds.).+2022.+Ethnologue%3A+Languages+of+the+World.+Twenty-fifth+edition.+Dallas%2C+Texas%3A+SIL+International&rlz=1C1GCEA_enPT1012PT1012&oq=Eberhard%2C+David+M.%2C+Gary+F.+Simons%2C+and+Charles+D.+Fennig+(eds.).+2022.+Ethnologue%3A+Languages+of+the+World.+Twenty-fifth+edition.+Dallas%2C+Texas%3A+SIL+International&aqs=chrome..69i57j69i60l2.674j0j4&sourceid=chrome&ie=UTF-8)
- Goldin-Meadow, S., & Brentari, D. (2017). Gesture, sign, and language: The coming of age of sign language and gesture studies. *Behavioral and Brain Sciences*, 40, e46. <https://doi.org/10.1017/S0140525X15001247>
- Hand Talk App*. (n.d.). Retrieved February 10, 2023, from <https://www.handtalk.me/en/app/>
- Hands / mediapipe*. (n.d.). Retrieved February 12, 2023, from <https://google.github.io/mediapipe/solutions/hands>
- Haynes, S., & Linden, M. (2012a). Workplace accommodations and unmet needs specific to individuals who are deaf or hard of hearing. *Http://Dx.Doi.Org/10.3109/17483107.2012.665977*, 7(5), 408–415. <https://doi.org/10.3109/17483107.2012.665977>
- Haynes, S., & Linden, M. (2012b). Workplace accommodations and unmet needs specific to individuals who are deaf or hard of hearing. *Http://Dx.Doi.Org/10.3109/17483107.2012.665977*, 7(5), 408–415. <https://doi.org/10.3109/17483107.2012.665977>
- Hirabayashi, N., Fujikawa, N., Yoshimura, R., & Fujisawa, Y. (2019). Development of learning support equipment for sign language and fingerspelling by mixed reality. *ACM International Conference Proceeding Series*. <https://doi.org/10.1145/3325291.3325373>
- Lozano Diaz, C., Maciel, R., Larios, M., & Ochoa, A. (2018). Augmented Reality System to Promote the Inclusion of Deaf People in Smart Cities. In *Research in Computing Science* (Vol. 147, Issue 2).
- Lucas, C. (2001). The Sociolinguistics of Sign Languages. In *Cambridge University Press*. Cambridge University Press. https://books.google.pt/books?hl=pt-PT&lr=&id=lxpuT_s8AkEC&oi=fnd&pg=PR9&dq=sign+language+community&ots=2sH4Mxllu

- S&sig=oG4BN7uidjxbv_9ik-eCPGCBF0&redir_esc=y#v=onepage&q=sign%20language%20community&f=false
- Luo, X., Han, M., Liu, T., Chen, W., & Bai, F. (2012). Assistive learning for hearing impaired college students using mixed reality: A pilot study. *Proceedings of the 2012 International Conference on Virtual Reality and Visualization, ICVRV 2012*, 74–81. <https://doi.org/10.1109/ICVRV.2012.20>
- Miller, A., Malasig, J., Castro, B., Hanson, V. L., Nicolau, H., & Brandão, A. (2017). The use of smart glasses for lecture comprehension by Deaf and hard of hearing students. *Conference on Human Factors in Computing Systems - Proceedings, Part F127655*, 1909–1915. <https://doi.org/10.1145/3027063.3053117>
- Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press. https://books.google.pt/books?hl=pt-PT&lr=&id=RC43AgAAQBAJ&oi=fnd&pg=PR7&dq=Machine+Learning:+A+Probabilistic+Perspective&ots=umoC9ALq56&sig=AbRK_PhAQubc2Eqpe5i0CcMh7PI&redir_esc=y#v=onepage&q=Machine%20Learning%3A%20A%20Probabilistic%20Perspective&f=true
- Musengi, M., Ndofirepi, A., & Shumba, A. (2013). Rethinking Education of Deaf Children in Zimbabwe: Challenges and Opportunities for Teacher Education. *The Journal of Deaf Studies and Deaf Education*, 18(1), 62–74. <https://doi.org/10.1093/DEAFED/ENS037>
- Nations, U. (1948). *Universal Declaration of Human Rights / United Nations*. <https://www.un.org/en/about-us/universal-declaration-of-human-rights>
- Parker, A., Switenky, C., Mathew, R., & Dannels, W. (2022). Mixed Reality Prototype Device Showcase: Using Smart Glasses to Enhance Language Access. *Frameless*, 4(1). <https://scholarworks.rit.edu/frameless/vol4/iss1/31>
- Programa Inclui / Fundação Altice*. (n.d.). Retrieved October 30, 2023, from <https://fundacao.altice.pt/tecnologia/programa-inclui>
- Salvi, S., Pahar, S., & Kadale, Y. (2021). Smart Glass Using IoT and Machine Learning Technologies to aid the blind, dumb and deaf. *Journal of Physics: Conference Series*, 1804(1), 012181. <https://doi.org/10.1088/1742-6596/1804/1/012181>
- Shao, Q., Sniffen, A., Blanchet, J., Hillis, M. E., Shi, X., Haris, T. K., Liu, J., Lamberton, J., Malzkahn, M., Quandt, L. C., Mahoney, J., Kraemer, D. J. M., Zhou, X., & Balkcom, D. (2020). Teaching American Sign Language in Mixed Reality. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(4), 27. <https://doi.org/10.1145/3432211>

- Sign language recognition datasets* · Facundo Quiroga. (n.d.). Retrieved February 13, 2023, from https://facundoq.github.io/guides/sign_language_datasets/slr
- SIGN-HUB*. (n.d.). Retrieved February 12, 2023, from <https://thesignhub.eu/>
- Virtual Sign – Língua Gestual Portuguesa*. (n.d.). Retrieved October 30, 2023, from <https://virtuallsign.com/>
- Von Agris, U., Knorr, M., & Kraiss, K. F. (2008). The significance of facial features for automatic sign language recognition. *2008 8th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2008*. <https://doi.org/10.1109/AFGR.2008.4813472>
- We translate sign language. Automatically / SignAll Chat*. (n.d.). Retrieved February 10, 2023, from <https://www.signall.us/chat>
- WFD. (n.d.). *The Legal Recognition of National Sign Languages - WFD*. Retrieved January 29, 2024, from <https://wfdeaf.org/news/the-legal-recognition-of-national-sign-languages/>
- WFD. (2010). *Submission of the World Federation of the Deaf (WFD) OHCHR thematic consultation on article 32 of the CRPD*.
- wireAVATAR*. (n.d.). Retrieved October 30, 2023, from <https://www.wiremaze.com/o-que-fazemos/cidadania/wireavatar>
- Yang, F. C., Mousas, C., & Adamo, N. (2022). Holographic sign language avatar interpreter: A user interaction study in a mixed reality classroom. *Computer Animation and Virtual Worlds*, 33(3–4), e2082. <https://doi.org/10.1002/CAV.2082>
- Yin, K., Moryossef, A., Hochgesang, J., Goldberg, Y., & Alikhani, M. (2021). Including Signed Languages in Natural Language Processing. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 7347–7360. <https://doi.org/10.18653/V1/2021.ACL-LONG.570>
- Young, A., Oram, R., & Napier, J. (2019). Hearing people perceiving deaf people through sign language interpreters at work: on the loss of self through interpreted communication. *https://doi.org/10.1080/00909882.2019.1574018*, 47(1), 90–110. <https://doi.org/10.1080/00909882.2019.1574018>