

Universidade Municipal de São Caetano do Sul

Escola Politécnica da USCS

Curso Superior de Graduação em Ciência da Computação

Everton de Sousa Monteiro

Filipe Noberto Justino

José Montoro Neto

Pedro Paulo de Araújo Euphrausino

Silas Bertholdo Ferreira

Victor Hugo Nascimento

LibrasConnect:

Reconhecimento Gestual na Tradução de Libras

São Caetano do Sul – SP

2024



Universidade Municipal de São Caetano do Sul
Escola Politécnica da USCS
Curso Superior de Graduação em Ciência da Computação

LibrasConnect:
Reconhecimento Gestual na Tradução de Libras

Everton de Sousa Monteiro	RA: 8163782
Filipe Noberto Justino	RA: 8162953
José Montoro Neto	RA: 8162332
Pedro Paulo de Araújo Euphrausino	RA: 8162016
Silas Bertholdo Ferreira	RA: 8161845
Victor Hugo Nascimento de Souza	RA: 8161848

São Caetano do Sul – SP

2024

Universidade Municipal de São Caetano do Sul
Escola Politécnica da USCS
Curso Superior de Graduação em Ciência da Computação

Everton de Sousa Monteiro
Filipe Noberto Justino
José Montoro Neto
Pedro Paulo de Araújo Euphrausino
Silas Bertholdo Ferreira
Victor Hugo Nascimento

LibrasConnect:
Reconhecimento Gestual na Tradução de Libras

Trabalho de conclusão de curso apresentado
à Universidade Municipal de São Caetano
do Sul, como requisito parcial para a
obtenção do Título de Bacharel em Ciência
da Computação.

Orientadora: Profa. Dra. Claudia Bianchi
Progetti

São Caetano do Sul – SP
2024

LibrasConnect:

Reconhecimento Gestual na Tradução de Libras

/ Everton de Sousa Monteiro

Filipe Noberto Justino

José Montoro Neto

Pedro Paulo de Araújo Euphrausino

Silas Bertholdo Ferreira

Victor Hugo Nascimento. – São Caetano do Sul – SP, 2024.

83f. : il. ; 30 cm.

Orientadora Profa. Dra. Claudia Bianchi Progetti

TCC (Graduação) – Universidade Municipal de São Caetano do Sul

Escola Politécnica da USCS

Curso Superior de Graduação em Ciência da Computação, 2024.

1. Libras. 2. Tecnologias Assistivas. 3. Visão Computacional. 4. Tradução. 5. Acessibilidade Digital.

I. Profa. Dra. Claudia Bianchi Progetti. II. Universidade Municipal de São Caetano do Sul. III. Ciência da Computação. IV. LibrasConnect: Reconhecimento Gestual na Tradução de Libras

REITOR DA UNIVERSIDADE MUNICIPAL DE SÃO CAETANO DO SUL

Prof. Dr. Leandro Campi Prearo

PRO-REITOR DE GRADUAÇÃO

Prof. Me. Silton Marcelli Romboli

GESTORA DOS CURSOS DE COMPUTAÇÃO

Profa. Ma. Cilene Aparecida Mainente

Universidade Municipal de São Caetano do Sul

**LibrasConnect:
Reconhecimento Gestual na Tradução de Libras**

Trabalho de conclusão de curso apresentado à Universidade Municipal de São Caetano do Sul, como requisito parcial para a obtenção do Título de Bacharel em Ciência da Computação, composta pela banca examinadora:

Profa. Dra. Claudia Bianchi Progetti
Orientadora

Professor 1
Universidade Municipal de São Caetano do Sul (USCS)

Professor 2
Universidade Municipal de São Caetano do Sul (USCS)

São Caetano do Sul – SP, 01 de janeiro de 20XX

"Inclusão é um direito daqueles que precisam, e incluir é um dever de todos."
(Letícia Butterfield)

Resumo

A comunicação é vital para o ser humano. Porém, para a população surda, integrar-se em uma sociedade predominantemente auditiva é desafiador. Há uma escassez de pessoas fluentes em Libras, e a tradução de línguas de sinais como a Libras, carece de soluções eficazes. Enquanto na população ouvinte, ferramentas como tradutores, gramáticas e dicionários são amplamente utilizadas para facilitar a comunicação, entre a população surda, tais tecnologias são escassas e de difícil acesso. O presente projeto, intitulado LibrasConnect, busca atender a essas demandas por meio do desenvolvimento de um software online para consultar, aprender e traduzir expressões entre português e Libras. A pesquisa incluiu uma revisão bibliográfica sobre a realidade do surdo no Brasil, o impacto da Libras, tecnologias assistivas, software de tradução automática e suas correlações. Também foi analisada a legislação que garante os direitos dos surdos, estabelecendo medidas de inclusão e acessibilidade. Muitos dos direitos reconhecidos pela legislação brasileira são definidos pela ONU como universais a todo ser humano, porém, as pessoas com deficiência ainda enfrentam obstáculos em usufruir de direitos básicos como educação, renda, e acesso à informação. Nesse contexto, o LibrasConnect não se apresenta apenas como uma ferramenta técnica, mas um projeto social. Além de facilitar a comunicação entre usuários falantes e surdos, visa contribuir para a acessibilidade digital e fomentar o aprendizado de Língua de Sinais.

Palavras-chaves: Libras. Tecnologias Assistivas. Visão Computacional. Tradução. Acessibilidade Digital.

Abstract

Communication is vital for human beings. However, for the deaf population, integrating into a predominantly auditory society is challenging. There is a shortage of people fluent in Libras, and translation between different sign languages such as Libras (Brazilian Sign Language), lacks effective solutions. While tools like translators, grammars, and dictionaries are widely used to facilitate communication among the hearing population, such technologies are scarce and difficult to access among the deaf population. The present project, titled LibrasConnect, aims to address these demands by developing an online software to consult, learn, and translate expressions between Portuguese and Libras. The research included a literature review on the reality of the deaf in Brazil, the impact of Libras, assistive technologies, automatic translation software, and their correlations. The legislation that guarantees the rights of the deaf, and establishes measures of inclusion and accessibility, was also analyzed. Many of the rights recognized by Brazilian legislation are defined by the UN as universal to all humans; however, people with disabilities still face obstacles in benefit from basic rights such as education, income, and access to information. In this context, LibrasConnect is not only presented as a technical tool but as a social project. In addition to facilitating communication among Libras users, it aims to contribute to digital accessibility and promote sign language learning.

Keywords: Libras. Assistive Technologies. Computer Vision. Translation. Digital Accessibility.

Lista de Figuras

Figura 1 – Esboço de interface do tradutor	19
Figura 2 – Esboço de interface do dicionário	20
Figura 3 – Diagrama de Casos de Uso	45
Figura 4 – Variável com chamada de função	46
Figura 5 – Configuração dos pontos(mp holistic) e traços(mp drawing)	46
Figura 6 – Configuração dos pontos(mp holistic) e traços(mp drawing)	47
Figura 7 – Aplicação do modelo mediapipe na câmera	47
Figura 8 – Aplicação do modelo mediapipe na câmera	48
Figura 9 – Resultado do reconhecimento dos pontos	48
Figura 10 – Configurações do ambiente de armazenamento de dados	49
Figura 11 – Criação das pastas	49
Figura 12 – Armazenamento dos landmarks	50
Figura 13 – Funções necessárias	50
Figura 14 – Definição e execução do modelo	51
Figura 15 – Resultado do treinamento	52
Figura 16 – Configuração da probabilidade do gesto executado	52
Figura 17 – Execução do script utilizando o modelo treinado	53
Figura 18 – Resultado do reconhecimento gestual	54
Figura 19 – Primeiro nível	55
Figura 20 – Segundo nível	56
Figura 21 – Terceiro nível	57
Figura 22 – Quarto nível	58
Figura 23 – Estrutura do banco de dados do sistema web	59
Figura 24 – Início	60
Figura 25 – Cadastro	61
Figura 26 – Tradução	62
Figura 27 – Aprendizagem	63
Figura 28 – Reconhecimento de Gestos	63
Figura 29 – Formulário exige senha com 8 caracteres	68
Figura 30 – Formulário exige letra maiúscula e caractere especial	69
Figura 31 – Formulário exige letra maiúscula	70
Figura 32 – Caractere Especial	71
Figura 33 – Número na senha	72
Figura 34 – Senhas iguais	73
Figura 35 – Senhas iguais com 8 caracteres	74
Figura 36 – Senha e os requisitos cumpridos	75
Figura 37 – Exibir senha desativado	76

Figura 38 – Exibir senha ativado	77
Figura 39 – Teste de Acurácia	78
Figura 40 – Teste de Loss (Perda)	79
Figura 41 – Teste em tempo real	79

Lista de Tabelas

Tabela 1 – Proporção de pessoas que sabem Libras - Brasil - 2019 16

Lista de Quadros

Quadro 1 - Cronograma	35
Quadro 2 - Requisitos Funcionais	37
Quadro 3 - Requisitos Não Funcionais - Segurança	38
Quadro 4 - Requisitos Não Funcionais - Usabilidade	39
Quadro 5 - Requisitos Não Funcionais - Desempenho	40
Quadro 6 - Requisitos Não Funcionais - Confiabilidade	41
Quadro 7 - Requisitos Não Funcionais - Portabilidade	41
Quadro 8 - Requisitos Não Funcionais - Manutenibilidade	43
Quadro 9 - Requisitos Não Funcionais - Compatibilidade	44

Lista de Gráficos

Grafico 1 - Ranking de frameworks mais utilizados 28

Lista de abreviaturas e siglas

ASL	Língua Americana de Sinais
CORS	<i>Cross Origin Resource Sharing</i>
CNN	<i>Convolutional Neural Networks</i>
CSS	<i>Cascading Style Sheets</i>
dB	Decibéis
GNU	<i>GNU is Not Unix</i>
HTML	<i>Hypertext Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IBGE	Instituto Brasileiro de Geografia e Estatística
IBM	<i>International Business Machines Corporation</i>
INES	Instituto Nacional de Educação de Surdos
JS	JavaScript
LIBRAS	Língua Brasileira de Sinais
ML	<i>Machine Learning</i>
OMS	Organização Mundial da Saúde
ONU	Organização das Nações Unidas
PLS	Projeto de Lei do Senado
PNS	Pesquisa Nacional de Saúde
RF	Requisito Funcional
RN	Regra de Negócios
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	<i>Structured Query Language</i>
SUS	Sistema Único de Saúde
SVG	<i>Scalable Vector Graphics</i>

TA	Tecnologia Assistiva
USCS	Universidade Municipal de São Caetano do Sul
W3C	<i>World Wide Web Consortium</i>
WSGI	<i>Web Server Gateway Interface</i>
XHTML	<i>Extensible Hypertext Markup Language</i>
XML	<i>Extensible Markup Language</i>

Sumário

1	INTRODUÇÃO	15
1.1	Objetivos	16
1.2	Justificativa	17
1.3	Delimitação do Estudo	18
1.3.1	Tradutor de Gestos	18
1.3.2	Dicionário de Sinais	19
2	REFERENCIAL TEÓRICO	21
2.1	Deficiência Auditiva/Surdez	21
2.2	Tradução via Software	22
2.3	Libras	22
2.4	Surdez e a Exclusão Social	23
2.5	Acessibilidade Digital e Tecnologias Assistivas	24
2.6	Ferramentas de Desenvolvimento	26
2.6.1	Machine Learning	26
2.6.2	Flask	27
2.6.3	Ferramentas e Bibliotecas de Processamento	29
2.6.4	Python	30
2.6.5	MySQL	30
2.7	Integração e Operação do Sistema	31
3	MÉTODO DE PESQUISA	32
4	DESENVOLVIMENTO	33
4.1	Planejamento do Projeto	33
4.2	Requisitos do Sistema	36
4.2.1	Requisitos Funcionais	36
4.2.2	Requisitos Não Funcionais	37
4.2.3	Diagrama de Casos de Uso	44
4.2.4	Reconhecimento Gestual	45
4.3	Desenho da Arquitetura do Software	54
4.4	Desenho do Projeto de Banco de Dados	58
4.4.1	Banco de dados do sistema web	58
4.4.2	Armazenamento de dados gestuais	58
4.5	Prototipação	60
4.6	Implementação	64
4.6.1	Evidências da Implementação do <i>Front-End</i>	64

4.6.2	Evidências da Implementação do <i>Back-End</i>	64
5	TESTES	66
5.1	Plano de Testes	66
5.2	Cenários de Testes	67
5.2.1	Sistema de <i>Login</i>	68
5.2.2	Reconhecimento de Gestos	78
6	CONCLUSÃO	80
	REFERÊNCIAS	81

1 Introdução

A comunicação é uma necessidade humana fundamental que, ao longo da história, tem sido facilitada pela tradução, desempenhando um papel essencial na superação de barreiras linguísticas e na promoção da interação entre diferentes culturas. Tradutores humanos e softwares de tradução automática têm sido amplamente utilizados em diversas áreas, como viagens, trabalho, educação, pesquisa, redes sociais e entretenimento. No entanto, há uma lacuna significativa na disponibilidade de ferramentas de tradução eficazes para as línguas de sinais.

Dados do Instituto Brasileiro de Geografia e Estatística (IBGE), divulgados em 2021, revelam que aproximadamente 2,3 milhões de pessoas no Brasil têm deficiência auditiva, correspondendo a cerca de 5% da população brasileira, e estima-se que 10,7 milhões possuam algum grau de surdez, conforme revelado pela Pesquisa Nacional de Saúde (PNS) de 2019 (IBGE, 2021). Globalmente, estima-se que cerca de 1,5 bilhão de pessoas sejam afetadas por perda auditiva, incluindo casos de perda parcial ou total. Projeções da Organização Mundial da Saúde (OMS) indicam que, até 2050, 2,5 bilhões de pessoas poderão desenvolver algum grau de surdez, sendo 1,1 bilhão de jovens em potencial risco de surdez permanente (GANDRA, 2021). Esse cenário é em grande parte atribuído ao hábito de expor os ouvidos a volumes sonoros excessivos, como ouvir música com fones de ouvido em níveis de volume que podem causar danos auditivos em apenas alguns minutos.

É pertinente observar que, no Brasil, onde mais de 10 milhões de pessoas têm algum grau de deficiência auditiva, apenas 9,2% afirmaram usar a Libras (Língua Brasileira de Sinais) como meio de comunicação. Em 2021, pela primeira vez, o IBGE questionou a proficiência na Libras de pessoas de 5 a 40 anos de sua idade, independentemente de sua condição de deficiência. Esses dados estão representados na Tabela 1 (IBGE, 2021). Quantificar os usuários de Libras pode direcionar políticas públicas, especialmente na educação. Embora nem todos os surdos se comuniquem, aqueles que o fazem requerem abordagens educacionais especializadas.

Tabela 1 – Proporção de pessoas que sabem Libras - Brasil - 2019

Grau de dificuldade para ouvir	Total	Proporção de pessoas de 5 a 40 anos de idade ou mais de idade, que referiram dificuldade permanente para ouvir, por conhecimento de Libras (%)					
		Sabe usar Libras				Não	
		Sim		Intervalo de confiança de 95%		Não	
		Pro- por- ção		Limite inferior	Limite superior	Pro- por- ção	Intervalo de confiança de 95%
Total	100,0	9,2	7,3	11,6	90,8	88,4	92,7
Alguma dificuldade	100,0	5,6	4,1	7,6	94,4	92,4	95,9
Muita dificuldade	100,0	12,9	8,6	19,0	87,1	81,0	91,4
Não consegue de modo algum	100,0	61,3	42,7	77,1	38,7	22,9	57,3
Deficiência auditiva (1)	100,0	22,4	16,5	29,6	77,6	70,4	83,5

Fonte: IBGE, Diretoria de Pesquisas, Coordenação de Trabalho e Rendimento, Pesquisa Nacional de Saúde 2019. (1)*
 Muita dificuldade de ouvir ou não consegue de modo algum ouvir.

Na pesquisa foi constatado que aproximadamente 1,7 milhão de indivíduos relataram enfrentar alguma forma de dificuldade auditiva. Destes, cerca de 153 mil indivíduos afirmaram possuir habilidades na utilização da Libras, o que representa aproximadamente 9,2% deste contingente populacional. Entre aqueles classificados como deficientes auditivos, caracterizados por enfrentarem graves dificuldades ou incapacidade total de audição, observou-se que 22,4% possuíam proficiência em Libras. Destaca-se a importância do domínio da Libras para os indivíduos que declararam incapacidade total de audição, uma vez que 61,3% deste grupo, aproximadamente 43 mil pessoas, demonstraram habilidades na referida língua (IBGE, 2021).

1.1 Objetivos

O objetivo desta pesquisa é investigar e desenvolver uma solução que contribua para o reconhecimento automático de sinais em Libras, facilitando a comunicação entre ouvintes e surdos. O projeto buscou explorar possibilidades de tecnologias assistivas, e optou-se pela criação de um sistema de reconhecimento de sinais em Libras e de um dicionário para consulta de sinais em Libras. O sistema de reconhecimento de gestos foi implementado utilizando técnicas de visão computacional e aprendizado de máquina, capazes de identificar e mapear os movimentos manuais característicos dos sinais em Libras. A captura de imagens é realizada por dispositivos com câmeras, enquanto um algoritmo disponibiliza a tradução em português. O sistema oferece exemplos visuais

e vídeos demonstrativos, facilitando o aprendizado dos sinais e garantindo uma melhor compreensão de seu contexto e significado. A intenção foi demonstrar como uma máquina pode ser treinada para reconhecer gestos e ilustrar a possibilidade de aplicação de um software tanto para fins educacionais quanto para melhorar a comunicação no cotidiano.

1.2 Justificativa

Considerando que a maioria da população surda carece de acesso à educação e à informação (GANDRA, 2021), e reconhecendo que o acesso à educação, a livre expressão e a participação na comunidade são direitos universais do ser humano (ONU, 1948), a presente iniciativa busca promover a inclusão social por meio do desenvolvimento de uma ferramenta que facilite o aprendizado de Libras, além de proporcionar uma interação mais eficaz entre pessoas surdas e ouvintes, por meio da tradução em tempo real.

Ademais, a iniciativa almeja contribuir para a pesquisa em tradução de linguagens de sinais, desenvolvendo metodologias que possibilitem a tradução entre Libras e línguas orais, visando avanços na área de comunicação acessível e incentivando novas pesquisas e aprimoramentos em tecnologias assistivas.

Considerando a legislação vigente, é estipulado que a União, Estados, Distrito Federal e Municípios devem adotar medidas necessárias para garantir a acessibilidade de conteúdo para pessoas com deficiência, conforme disposto na Lei nº 12.527/2011 (BRASIL, 2011). Além disso, a Lei nº 10.098/2000 estabelece que o Poder Público deve fomentar programas destinados à promoção de pesquisas científicas para o tratamento e prevenção de deficiências, ao desenvolvimento tecnológico de ajudas técnicas para pessoas com deficiência e à especialização de recursos humanos em acessibilidade (BRASIL, 2000).

O Decreto nº 5.626 de 2005 assegura o direito ao ensino bilíngue para surdos e determina que repartições públicas, como unidades do Sistema Único de Saúde (SUS), escolas públicas e outras autarquias devem oferecer atendimento em Língua Brasileira de Sinais (BRASIL, 2005).

O Projeto de Lei do Senado (PLS) nº 155 de 2017, atualmente em tramitação, cobra que repartições públicas, empresas concessionárias de serviços públicos e instituições financeiras ofereçam atendimento especializado para pessoas que se comunicam por Libras. Esta proposta é válida por garantir atendimento inclusivo à pessoa com deficiência auditiva, assegurando que sua língua não será um obstáculo para o exercício pleno da cidadania (REDAÇÃO, 2019).

Considerando a necessidade de atendimento em Libras reconhecida pelas legislações vigentes, espera-se que este projeto contribua para que o atendimento ao público, tanto em instituições públicas quanto privadas, tenha acesso ao aprendizado de Libras ou utilize

o software como ferramenta de consulta. Assim, será possível atender, ensinar, orientar e prestar suporte à população de forma inclusiva, assegurando que a população surda também usufrua de pleno acesso aos seus direitos fundamentais.

1.3 Delimitação do Estudo

O projeto LibrasConnect foi desenvolvido com o objetivo de facilitar o aprendizado e a interação em Libras, com dois módulos principais: o dicionário de sinais e o tradutor de gestos. Embora tenha sido idealizada uma plataforma de cursos ministrados por professores de Libras, essa ideia não foi implementada devido à limitação de recursos e tempo, sendo esboçada apenas em protótipos. O escopo do estudo foi delimitado aos módulos de reconhecimento de gestos e ao dicionário de sinais, que se tornaram o foco central do projeto.

O tradutor de gestos, desenvolvido em *Python*, utiliza técnicas de visão computacional para identificar sinais de Libras através da câmera do computador. O sistema captura e processa, em tempo real, imagens das mãos e rosto do usuário, reconhecendo os gestos e exibindo a tradução em texto na tela. Em complemento ao tradutor, foi elaborado um dicionário de sinais que permite consultas de gestos em Libras e ASL (American Sign Language).

O projeto teve início em 2023 na Universidade Metodista de São Paulo, com o desenvolvimento do módulo de dicionário, e foi aprimorado em 2024, após a transferência dos alunos para a USCS. Nessa fase, identificou-se a necessidade de melhorar o atendimento às demandas da comunidade surda, especialmente no que diz respeito à tradução de gestos para a língua portuguesa, mudando o foco do projeto para o reconhecimento de gestos com visão computacional. O trabalho envolveu revisão bibliográfica sobre Libras e tecnologias assistivas, além da pesquisa de ferramentas tecnológicas relacionadas a *Machine Learning*.

1.3.1 Tradutor de Gestos

O tradutor de gestos, desenvolvido em *Python*, utiliza técnicas de visão computacional para o reconhecimento de sinais em Libras por meio da câmera do computador. O software capta imagens em tempo real das mãos e do rosto do usuário, processando-as com base em um modelo treinado pela equipe de desenvolvimento para identificar os gestos com precisão.

O treinamento do modelo foi realizado com dados gerados pela equipe executando os gestos em frente à câmera, permitindo a captura de várias amostras sob diferentes ângulos e condições de iluminação. Para isso, foram utilizados *frameworks* como *TensorFlow* e *PyTorch*, com Redes Neurais Convolucionais ou Convolutional Neural Networks (CNNs),

possibilitando que o sistema aprendesse a reconhecer padrões de gestos associados a cada sinal em Libras.

O software permite que o usuário interaja diretamente com a câmera do dispositivo, realizando gestos que são capturados e processados em tempo real. As imagens são analisadas por um algoritmo de visão computacional pré-treinado, que identifica e interpreta os gestos. A tradução dos sinais é exibida em formato de texto na tela. A Figura 1 ilustra um esboço da interface e da usabilidade, destacando a simplicidade e funcionalidade alcançadas com o desenvolvimento do projeto.

Figura 1 – Esboço de interface do tradutor



Fonte: Autores.

1.3.2 Dicionário de Sinais

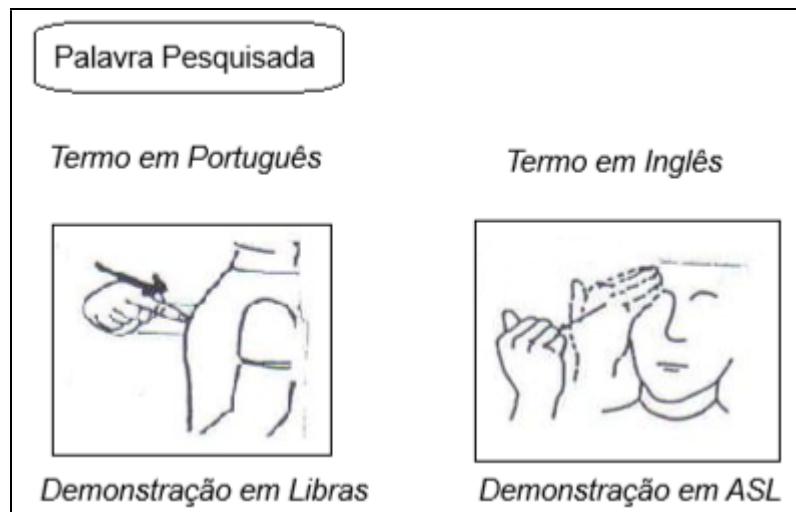
O desenvolvimento do módulo de consulta de expressões em Libras teve como objetivo criar uma ferramenta funcional e acessível, equivalente a um dicionário tradicional de gramática. Esse software inclui entradas lexicais e expressões comumente utilizadas na língua de sinais, oferecendo uma experiência de consulta prática e eficiente para os usuários.

A estrutura do módulo foi planejada para consultas rápidas, seguindo o modelo de um dicionário de gramática voltado para línguas faladas. Cada entrada lexical ou expressão é acompanhada por vídeos e/ou imagens que exemplificam a correta execução dos sinais. Tal abordagem facilita o entendimento e o aprendizado de Libras, consolidando o dicionário como uma ferramenta de referência prática para os seus usuários. A Figura 2 ilustra um esboço preliminar da interface e do funcionamento desse recurso.

De maneira semelhante a um dicionário de gramática que apresenta informações sobre a estrutura e uso das palavras em línguas faladas, este instrumento de consulta de sinais orienta o usuário quanto à estrutura e uso dos sinais em Libras. Os recursos visuais

disponíveis permitem que os usuários compreendam tanto o significado dos sinais quanto sua execução gestual e aplicação prática.

Figura 2 – Esboço de interface do dicionário



Fonte: Autores.

O recurso visa atender àqueles interessados em se comunicar com pessoas surdas, oferecendo uma ferramenta para consulta de gestos no cotidiano ou para o aprendizado do idioma.

2 Referencial Teórico

Este capítulo tem como objetivo apresentar os principais conceitos que fundamentam este trabalho. Será abordada a evolução histórica da tradução, com ênfase na tradução via software. Além disso, serão contextualizados o surdo, a surdez e as línguas de sinais no mundo e no Brasil, além de discutir a exclusão social enfrentada pela comunidade surda e a importância da acessibilidade digital em seu cotidiano, reconhecendo-a como um direito intrínseco. Também serão destacadas as inovações tecnológicas que têm possibilitado a inclusão das comunidades com deficiência auditiva nas relações interpessoais do século XXI. Por fim, serão apresentadas as ferramentas e linguagens de programação que serão utilizadas para o desenvolvimento do presente software.

2.1 Deficiência Auditiva/Surdez

A surdez é uma deficiência que compromete o sistema auditivo do indivíduo, resultando na redução da capacidade de ouvir e, consequentemente, em dificuldades para perceber e compreender a fala e outros sons do ambiente. Isso interfere no desenvolvimento da língua oral e inibe o envolvimento do indivíduo em processos de integração pessoais, educacionais e profissionais. O termo correto para tratar de pessoas com perda auditiva é “surdo ou surda”. Segundo o Decreto nº 5626 de 22 de dezembro de 2005, “considera-se pessoa surda aquela que, por ter perda auditiva, comprehende e interage com o mundo por meio de experiências visuais, manifestando sua cultura principalmente pelo uso da Libras” (BRASIL, 2005).

As perdas auditivas são classificadas em relação ao grau de comprometimento na obtenção de limiares auditivos correspondentes à menor intensidade de som que o indivíduo consegue ouvir. Tal classificação é medida na escala de decibéis (dB). A audição normal é caracterizada pela habilidade de detecção de sons até 25 dB. Para a deficiência auditiva leve, o limiar situa-se entre 25 e 40 dB. Para a moderada, o limiar está no intervalo de 45 a 70 dB. Já para a deficiência auditiva severa, a faixa é de 75 a 90 dB. Caso a habilidade de detecção de sons seja superior a 90 dB, é considerada deficiência profunda (LOUREIRO, 2004). A autora salienta que, a perda auditiva é uma questão de saúde física, podendo os fatores que a causam surgir durante os períodos pré-natal, peri-natal ou pós-natal e os fatores ocorridos nesses períodos podem ser categorizados em fatores hereditários, como síndromes familiares, e em fatores não hereditários, que englobam alterações endócrinas, bacterianas e uma variedade de outros motivos que podem surgir em diferentes estágios do desenvolvimento.

2.2 Tradução via Software

A tradução não é um fenômeno recente. Evidências sugerem que tradutores já atuavam no Antigo Egito e no Império Romano. Registros como a Pedra de Roseta, escrita em hieróglifos egípcios, em demótico e em grego, demonstram que a prática de tradução existe desde que o homem adotou a língua falada (ROMÃO, 1998). No século XV, a invenção da imprensa impulsionou a produção e circulação de textos escritos, aumentando a demanda por traduções. A tradução de textos religiosos para línguas vernáculas teve um impacto significativo na Reforma Protestante, com a tradução inovadora da Bíblia empreendida por Martinho Lutero (ROMÃO, 1998).

Por milênios, a tradução dependia de profissionais bilíngues, mas a era da informação trouxe a tradução automática como uma área de pesquisa ativa. Entre 1951 e 1954, a IBM e a Georgetown University realizaram o primeiro experimento de tradução automática entre russo e inglês (IBM, 1954). As pesquisas continuaram nas décadas de 1950 a 1980, porém, o desenvolvimento de softwares de tradução comercialmente viáveis só se tornou realidade após 1988.

Atualmente, o software de tradução mais utilizado é o Google Translate lançado em 2012, que padronizou o acesso gratuito a esse tipo de serviço mas permanece incompatível com línguas de sinais.

2.3 Libras

Segundo Cristiano (2017), a Libras é reconhecida como uma língua natural, espacial e visual, composta por gestos, expressões faciais e corporais. É usualmente utilizada pela comunidade surda brasileira para se comunicar, transmitir informações e expressar ideias. Essa linguagem possui sua própria gramática, sintaxe e vocabulário, distintos do português oral e escrito (LIBRAS, 2020).

A história da Libras tem início em 1857, quando Eduard Huet, um surdo francês, foi convidado por D. Pedro II para fundar a primeira escola para surdos no Brasil, então denominada Imperial Instituto de Surdos Mudos, atualmente conhecido como Instituto Nacional de Educação de Surdos (INES). A Libras foi desenvolvida pela instituição combinando elementos da Língua Francesa de Sinais com sinais já utilizados pela comunidade surda brasileira (BRASIL, 2021).

No entanto, a oficialização da Libras como idioma só ocorreu em 2002, com a Lei nº 10.436, que reconheceu Libras como um meio legal de comunicação e expressão em território nacional, um marco histórico para a comunidade surda brasileira (BRASIL, 2002). A Libras possui uma estrutura linguística complexa, composta por diversos elementos,

onde eles se combinam para formar frases e expressões. Estes elementos incluem: Sinais manuais, expressões faciais, movimento corporal, parâmetros linguísticos etc.

2.4 Surdez e a Exclusão Social

Cerca de 56 milhões de surdos, correspondendo a 80% dos 70 milhões de surdos no mundo, não têm acesso à educação, especialmente mulheres, meninas e pessoas em países em desenvolvimento, o que viola o direito universal à educação (OMS, 2019).

No Brasil, a taxa de analfabetismo entre pessoas com deficiência é de 19,5%, em contraste com 4,1% entre aqueles sem deficiência. Apenas 25,6% das pessoas com deficiência completaram o Ensino Médio, comparado a 57,3% das pessoas sem deficiência (IBGE, 2023).

Ao abordar a ineficácia na educação de surdos, Skliar (1997) aponta que

“O fracasso na educação dos surdos, com seus múltiplos e variados sintomas, constituiu e constitui ainda hoje motivo para dois tipos de justificativas igualmente inapropriados: por um lado, que os surdos são os responsáveis diretos por esse fracasso — fracasso, pois, da surdez, dos dons biológicos naturais —; por outro, que se trata de uma dificuldade metodológica, o que fortalece a necessidade de purificar e sistematizar ainda mais os métodos. [...] Na educação dos surdos, os surdos não fracassaram; fracassaram os ouvintes que nela trabalham.” (SKLIAR, 1997. p. 4)

Além da educação, a disparidade é evidente também no mercado de trabalho. Apenas 26,6% das pessoas com deficiência estavam empregadas em 2022, menos da metade do percentual para aqueles sem deficiência (60,7%). Mesmo entre pessoas com nível superior, a diferença persiste, com uma taxa de 54,7% para pessoas com deficiência empregadas, em comparação com 84,2% para aqueles sem deficiência (IBGE, 2023).

Ainda de acordo com IBGE (2023), os homens com deficiência receberam em média R\$ 2.157, cerca de 27% a menos do que os homens sem deficiência, que receberam em média R\$ 2.941. A disparidade foi ainda maior entre as mulheres com deficiência, cuja média salarial foi de R\$ 1.553, representando aproximadamente 34% a menos do que as mulheres sem deficiência, que receberam em média R\$ 2.347.

Assim como o índice de pobreza é elevado entre as pessoas com deficiência, o índice de deficiências entre populações pobres também tende a ser maior. A OMS (2011) reporta que 80% das pessoas com deficiência vivem em países em desenvolvimento e que a prevalência de pessoas com limitações físicas, intelectuais e motoras é maior em regiões mais pobres. Um estudo da OMS realizado em 56 países em desenvolvimento revelou que os mais pobres apresentam um quadro de saúde pior do que os mais ricos, além de um maior número de pessoas com deficiência. A pobreza pode levar ao surgimento de problemas de

saúde associados a diversas deficiências, incluindo baixo peso ao nascimento, desnutrição, falta de água potável ou saneamento adequado, condições inseguras de trabalho e de vida, lesões crônicas e menor acesso a vacinas. Além disso, a pobreza pode aumentar a probabilidade de que um problema de saúde já existente se torne uma deficiência, devido a um ambiente sem acessibilidade ou à falta de acesso aos serviços de saúde e reabilitação necessários OMS (2011). Dessa forma, estabelece-se um ciclo vicioso entre baixa renda e deficiência, no qual a deficiência pode ser tanto uma causa quanto uma consequência da pobreza.

Conforme estabelecido na Declaração Universal dos Direitos Humanos , todos têm o direito ao acesso à educação, saúde, mercado de trabalho e renda, sem qualquer forma de discriminação ONU (1948). No entanto, como previamente evidenciado, tais direitos são obstaculizados para as pessoas com deficiência devido ao preconceito e à inacessibilidade pública.

Skliar (1997) aborda a percepção da sociedade em relação aos surdos, afirmando que “os surdos, como tantos outros grupos humanos, são definidos apenas a partir de supostos traços negativos e percebidos como desvio da normalidade” (SKLIAR, 1997). De acordo com Campos (2011) complementa essa visão, destacando que:

“A modernidade [...] denominou e inventou modos de componentes negativos, tais como, marginal, louco, deficiente, drogado, homossexual, etc. No caso do surdo, [...] ele é apresentado como deficiente e não diferente; funciona como o depositário de todos os males, como o portador das “falhas” sociais.” (CAMPOS, 2011. p. 37)

2.5 Acessibilidade Digital e Tecnologias Assistivas

No Estatuto da Pessoa com Deficiência (Lei nº 13.146/2015) (BRASIL, 2015), acessibilidade é definida como:

“possibilidade e condição de alcance para utilização, com segurança e autonomia, de espaços, mobiliários, equipamentos urbanos, edificações, transportes, informação e comunicação, inclusive seus sistemas e tecnologias, bem como de outros serviços e instalações abertos ao público, de uso público ou privados de uso coletivo, tanto na zona urbana como na rural, por pessoa com deficiência ou com mobilidade reduzida.” (BRASIL. Lei nº 13.146/2015, Art. 3º)

Nas últimas décadas, diversas tecnologias para pessoas com deficiência auditiva foram desenvolvidas. Isso inclui o uso de aparelhos auditivos, implantes cocleares, terapia auditiva, entre outros. Além disso, medidas de acessibilidade digital, como ferramentas de tradução, legendas em vídeos e intérpretes de língua de sinais podem facilitar a participação plena dessas pessoas em diversos aspectos da vida.

O desenvolvimento tecnológico tem impulsionado transformações na sociedade,

refletindo também no setor educacional. Tais recursos têm contribuído diretamente para a implantação de plataformas de ensino online por instituições de ensino. De acordo com Salton, Agnol e Turcatti (2017), “[...] acessibilidade é oferecer possibilidades de transpor as barreiras que existem na sociedade, garantindo que todas as pessoas possam participar dos diversos âmbitos sociais”. Desse modo, a acessibilidade digital é a forma ou direito das pessoas de acessar a informação e a comunicação em ambientes digitais, de forma igualitária e sem barreiras (SALTON; AGNOL; TURCATTI, 2017, p. 11).

Pretto e Bonilla (2001), destacam a importância do acesso à informação para o processo de inclusão, afirmando que "para que a cidadania seja plena, precisamos investir na autonomia do cidadão e na democratização da informação, o que implica potencializar processos horizontais de organização, produção e aprendizagem coletiva que se constroem com o acesso às informações" (PRETTO; BONILLA, 2001).

Sendo assim, a acessibilidade digital se faz essencial para garantir à população com algum tipo de deficiência o devido acesso à educação, ao mercado de trabalho e ao pleno exercício de seus direitos como cidadãos. As barreiras digitais incluem a falta de conteúdos adaptados para pessoas com cegueira ou baixa visão, bem como a ausência de conteúdo com legendas ou intérpretes de Libras que facilitem a compreensão dos surdos.

Tecnologia Assistiva (TA) é o conjunto de artefatos disponibilizados as pessoas com necessidades especiais, as quais proporcionam uma vida mais independente, com mais qualidade e possibilidade de inclusão social descreve Sonza (2008). A autora enfatiza a importância dos recursos tecnológicos na promoção da autonomia e inclusão de pessoas com deficiência.

Além disso, o Estatuto da Pessoa com Deficiência (Lei nº 13.146/2015) também aborda a Tecnologia Assistiva, definindo-a como:

“produtos, equipamentos, dispositivos, recursos, metodologias, estratégias, práticas e serviços que objetivem promover a funcionalidade, relacionada à atividade e à participação da pessoa com deficiência ou com mobilidade reduzida, visando à sua autonomia, independência, qualidade de vida e inclusão social” (BRASIL. Lei nº 13.146/2015, Art. 3º).

Essa definição legal amplia o escopo da Tecnologia Assistiva, reconhecendo a relevância do acesso à informação e da comunicação na promoção de inclusão social das pessoas com deficiência.

A Lei nº 10.098/2000 (BRASIL, 2000) determina o fomento de pesquisas na área de tecnologia assistiva como uma obrigação da União, Estados, Distrito Federal e Municípios:

"Art. 21. O Poder Público, por meio dos organismos de apoio à pesquisa e das agências de financiamento, fomentará programas destinados: I – à promoção de pesquisas científicas voltadas ao

tratamento e prevenção de deficiências; II – ao desenvolvimento tecnológico orientado à produção de ajudas técnicas para as pessoas portadoras de deficiência; III – à especialização de recursos humanos em acessibilidade."

Atualmente, mesmo com a legislação em vigor e a crescente demanda por tecnologias assistivas para surdos, as opções de ferramentas para a aprendizagem da Libras ou ASL, bem como para a conversão entre essas linguagens, são limitadas. Portanto, a implementação do software apresentado neste trabalho busca contribuir com o direito universal de acesso à informação.

2.6 Ferramentas de Desenvolvimento

Nesta seção, serão apresentadas as ferramentas tecnológicas empregadas no desenvolvimento do projeto. As linguagens de marcação HyperText Markup Language (*HTML*) e Cascading Style Sheets (*CSS*) foram usadas para estilizar a parte inicial do sistema web, junto com *Javascript*. A linguagem de programação *Python* foi utilizada para a construção do código-fonte do reconhecimento de gestos, complementada por ferramentas de *machine learning* e *frameworks* como *Flask*, *TensorFlow* e *PyTorch*, além de bibliotecas *Python* como *Mediapipe*, *NumPy* e *Scikit-learn*.

Considerando que o projeto é sem fins lucrativos e levando em conta a relação entre deficiência e baixa renda discutida na seção 2.5, é fundamental que o tradutor seja acessível gratuitamente, caso venha a ser disponibilizado ao público. Para garantir essa acessibilidade, o desafio foi utilizar apenas recursos de código aberto (*open source*) e ferramentas gratuitas, garantindo a sustentabilidade do projeto sem comprometer o acesso das pessoas que mais precisam da solução.

2.6.1 Machine Learning

Machine Learning (ML) é uma subárea da Inteligência Artificial (IA) que se concentra em capacitar sistemas a aprenderem a partir de dados. Diferentemente da programação tradicional, em que o comportamento de um sistema é descrito explicitamente por meio de código, no ML os algoritmos são treinados com exemplos de comportamentos desejados. Esse processo de treinamento permite que o modelo ajuste seus parâmetros para realizar tarefas de forma autônoma, sem a necessidade de ser programado especificamente para cada caso. O resultado final é um modelo treinado, cujos parâmetros são ajustados com base nos padrões observados durante o treinamento (GOOGLE, 2023).

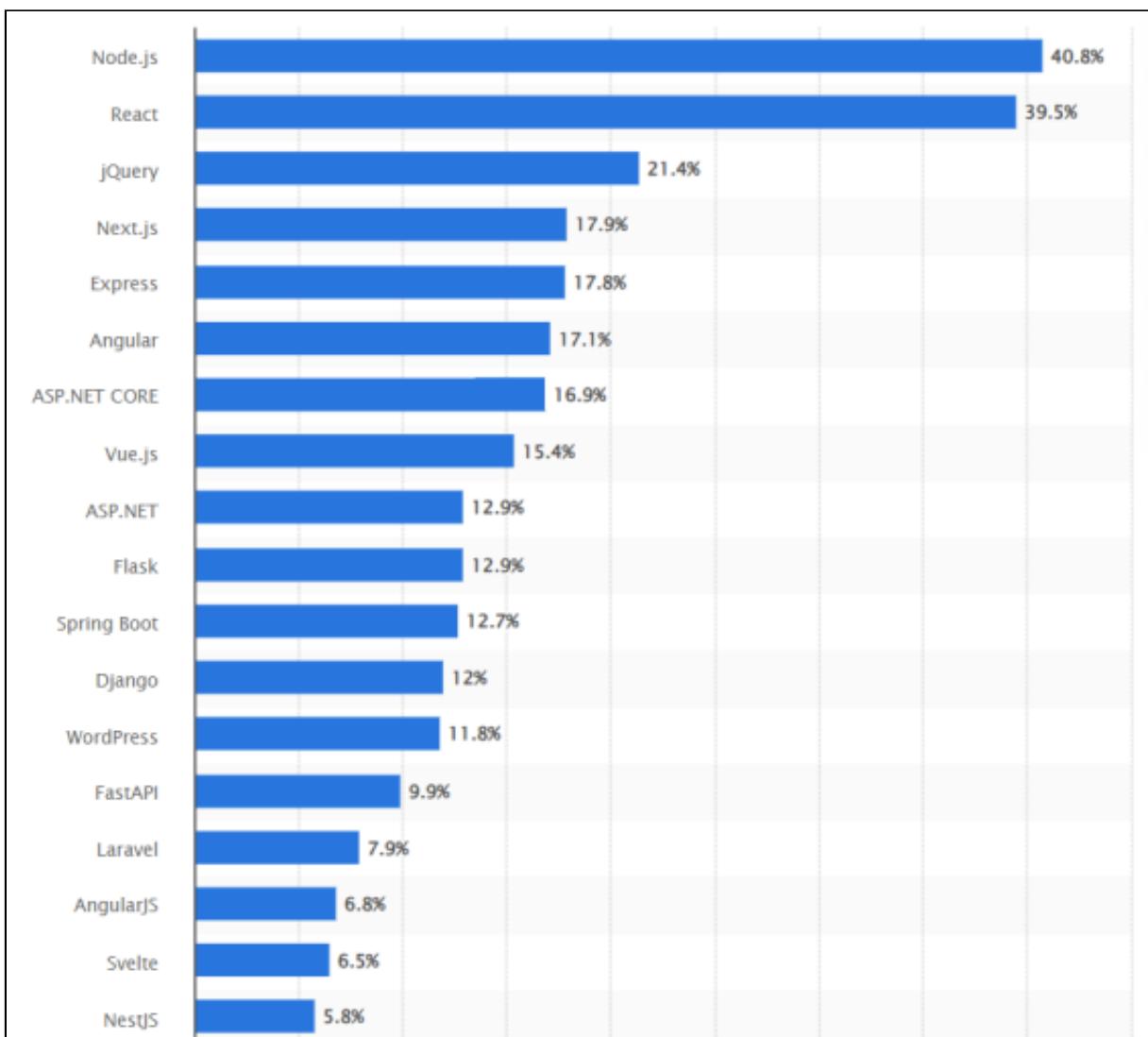
Neste estudo, o aprendizado de máquina foi aplicado ao reconhecimento de gestos em Libras. Um conjunto de dados foi gerado pela equipe de desenvolvimento, com base

em gestos realizados em frente a uma câmera. Para o treinamento do modelo, *frameworks* de aprendizado profundo, como *TensorFlow* e *PyTorch*, foram utilizados, pois oferecem suporte as (CNNs) redes neurais convolucionais. As CNNs são amplamente utilizadas em aplicações de visão computacional por sua eficácia no reconhecimento de padrões visuais, tornando-as particularmente adequadas para o reconhecimento de gestos.

2.6.2 Flask

Para a implementação da interface web entre o usuário e o sistema de reconhecimento de gestos, foi utilizado o *framework* *Flask*. Trata-se de um *framework* Web Server Gateway Interface (WSGI) leve, adequado para a rápida inicialização de projetos web e que oferece flexibilidade na escalabilidade da aplicação conforme as demandas do sistema. No contexto deste projeto, o *Flask* desempenha o papel de intermediar a comunicação entre o cliente (*frontend*) e o servidor (*backend*), processando as requisições enviadas pelo usuário e retornando os resultados do processamento dos gestos capturados pela câmera.

A simplicidade, flexibilidade, escalabilidade, confiabilidade e capacidade de processar grandes volumes de tráfego estão entre os principais fatores que justificam a ampla adoção deste *framework* por empresas. Atualmente, o *Flask* está entre os *frameworks web* mais utilizados globalmente, como ilustrado no Gráfico 1 e de acordo com dados divulgados pelo (VAILSHERY, 2024).

Grafico 1 – Ranking de frameworks mais utilizados

Fonte: STATISTA (2024)

Um recurso relevante do *Flask* utilizado no desenvolvimento foi o *Cross-Origin Resource Sharing* (CORS), implementado por meio do módulo *flask.cors*, permitindo que a aplicação receba requisições de diferentes origens. Essa funcionalidade é essencial para aplicações distribuídas, como a desenvolvida neste projeto, acessível por múltiplos dispositivos em diversos locais. O CORS garante o funcionamento eficiente em ambientes heterogêneos, respeitando as permissões de segurança necessárias para o compartilhamento de recursos entre origens distintas.

Diversas funções do *Flask* foram essenciais na comunicação entre *frontend* e *backend*, incluindo *request*, *jsonify*, *subprocess* e *logging*. A função *request* permite que o sistema acesse e processe os dados enviados pelo cliente, como parâmetros de formulários ou arquivos, por meio de métodos *Hypertext Transfer Protocol* (HTTP), como POST ou GET. No contexto do reconhecimento de gestos, *request* captura as imagens enviadas pela

câmera do usuário e transmite esses dados ao *backend* para processamento.

A função *jsonify* converte os dados processados em um formato *JavaScript Object Notation* (JSON), adequado para a transmissão entre o servidor e o cliente de maneira estruturada e compatível com a web. No projeto, essa função foi utilizada para retornar os resultados do reconhecimento de gestos ao *frontend*, permitindo a exibição dos dados na interface do usuário de forma eficiente. Já o *subprocess* possibilita a execução de comandos do sistema operacional a partir do código *Python*, utilizado, por exemplo, para invocar *scripts* externos ou processos auxiliares durante o desenvolvimento. A função *logging* foi empregada para gerar e armazenar registros detalhados de eventos e atividades do sistema, o que facilita o diagnóstico de erros e a auditoria do fluxo de dados.

A comunicação em tempo real entre cliente e servidor foi implementada com o uso de *sockets*, possibilitando o envio e recebimento bidirecional de dados, como as imagens da câmera. Esse mecanismo assegura que os gestos capturados sejam processados e traduzidos sem atrasos perceptíveis, proporcionando uma experiência de uso fluida para os usuários.

2.6.3 Ferramentas e Bibliotecas de Processamento

Diversas ferramentas e bibliotecas desempenham papéis fundamentais no sistema de reconhecimento de gestos, contribuindo para o desempenho eficiente e em tempo real.

Essas ferramentas e bibliotecas foram integradas de forma a construir um sistema robusto, capaz de realizar o reconhecimento de gestos com eficiência e em tempo real.

Mediapipe é um *framework* de visão computacional desenvolvido pelo *Google*, utilizado para a detecção de mãos e rostos. O *Mediapipe* fornece modelos pré-treinados para rastreamento de objetos, o que acelerou o desenvolvimento ao permitir que o foco principal fosse o treinamento específico para Libras (GOOGLE, 2024).

NumPy é uma biblioteca para manipulação numérica que facilita o processamento de grandes volumes de dados visuais. No contexto deste projeto, a *NumPy* é utilizada para manipular e preparar os dados de imagem antes de serem processados pelo modelo de aprendizado de máquina (NUMPY, 2024).

Scikit-learn (sklearn) é uma biblioteca de aprendizado de máquina que auxilia na criação de algoritmos de classificação. Ela desempenha um papel essencial no processo de treinamento e avaliação do modelo de reconhecimento de gestos (SCIKIT-LEARN, s.d.).

TensorFlow Keras é um *framework* de aprendizado profundo utilizado para o treinamento de redes neurais. Através do *Keras*, foi possível construir e treinar redes neurais convolucionais que conseguem extrair características visuais dos gestos e associá-los aos sinais correspondentes em Libras (TENSORFLOW, 2024) (KERAS, 2024).

Matplotlib é uma biblioteca de visualização de dados, utilizada para criar gráficos e representações visuais que facilitam a análise dos dados gerados durante o treinamento do modelo. No projeto, a *Matplotlib* é utilizada para plotar curvas de aprendizado, permitindo o acompanhamento do desempenho do modelo ao longo do tempo, como a taxa de acertos e o erro de classificação (MATPLOTLIB, 2024).

2.6.4 Python

Python é uma linguagem de programação de propósito geral e alto nível, caracterizada pela sua versatilidade e suporte a diferentes paradigmas de programação, incluindo procedural, orientado a objetos e funcional. Desenvolvida por Guido van Rossum no final da década de 1980 como sucessora da linguagem ABC, *Python* é amplamente adotada em diversos domínios, abrangendo desde o desenvolvimento web até o uso como linguagem de *script* em várias aplicações e na construção de sistemas robustos (MOZILLA, 2023).

A escolha do *Python* 3.12 foi motivada por suas características de simplicidade, confiabilidade e eficiência. Como uma linguagem de alto nível, *Python* facilita o desenvolvimento ágil e seguro, proporcionando uma base robusta e flexível que atende às exigências tecnológicas do projeto, assegurando tanto eficiência quanto escalabilidade. Além disso, *Python* é compatível com as principais ferramentas de ML, incluindo bibliotecas essenciais para este projeto, o que reforça sua adequação ao contexto de aplicações avançadas em aprendizado de máquina.

2.6.5 MySQL

O *MySQL* é um banco de dados de código aberto amplamente conhecido e um dos mais utilizados entre desenvolvedores, especialmente em áreas como *Ecommerce*, Plataformas de redes sociais e Gerenciamento de conteúdo, incluindo aplicações populares como *Facebook*, *Twitter*, *Netflix* e *Uber*. Sendo de código aberto, o *MySQL* permite que qualquer pessoa o utilize e modifique livremente de acordo com suas necessidades, graças à Licença Pública Geral GNU (GPL) (ORACLE, 2024).

Como um banco de dados relacional, o *MySQL* organiza informações em tabelas separadas e armazena os dados em arquivos físicos otimizados para desempenho. Esse modelo relacional oferece flexibilidade ao desenvolvedor, permitindo estabelecer regras que controlam as relações entre os dados, garantindo consistência e integridade. A linguagem *Structured Query Language* (SQL) é utilizada para acessar e manipular os dados, podendo ser executada diretamente, integrada a outras linguagens ou utilizada por meio de APIs específicas (ORACLE, 2024).

No projeto, o *MySQL* foi inicialmente escolhido para gerenciar os dados. Porém, em

uma etapa posterior, foi temporariamente substituído por arquivos JSON. Entretanto, com a necessidade de armazenar dados mais complexos, como os gestos capturados no módulo de reconhecimento, informações dos usuários no sistema de login, além de vídeos e textos do módulo de dicionário de expressões, tornou-se necessário retornar ao uso do MySQL para garantir uma estrutura mais robusta e escalável.

2.7 Integração e Operação do Sistema

Durante a operação, o software capture imagens em tempo real das mãos e do rosto do usuário, processando essas imagens com visão computacional para identificar gestos. O modelo treinado utilizando redes neurais convolucionais, analisa padrões de movimento e expressão. Em seguida, o sistema exibe a tradução dos gestos em formato de texto na interface web. A integração entre o *Flask* e o modelo de aprendizado de máquina garante que a comunicação entre usuário e servidor seja fluida e responsiva, proporcionando uma experiência interativa.

Para facilitar a análise e depuração dos dados processados, a biblioteca *matplotlib* foi utilizada, permitindo a visualização dos resultados intermediários do processo de reconhecimento. Além disso, ferramentas como *subprocess* e *logging* desempenham papéis essenciais na execução de tarefas externas e no monitoramento de eventos, assegurando que o fluxo de execução da aplicação seja controlado de maneira eficiente.

A combinação dessas tecnologias assegura que o sistema de reconhecimento de gestos seja acessível, escalável e pronto para evoluções futuras.

3 Método de Pesquisa

A pesquisa, de caráter exploratório e bibliográfico, envolveu o levantamento e análise de materiais previamente publicados, como artigos científicos, livros, teses e publicações em periódicos. O objetivo foi estabelecer uma base teórica que fundamentasse o desenvolvimento do projeto e situasse as soluções propostas no contexto das práticas e tecnologias existentes.

Na primeira etapa, a investigação focou na compreensão do problema por meio da consulta a fontes sobre Libras, além de dados estatísticos e informações específicas sobre a população surda. Estudos sobre a evolução da Libras no Brasil e suas particularidades foram revisados, buscando identificar as demandas da comunidade surda em comunicação e como a tecnologia poderia ser ajustada para atender essas necessidades. Também foram analisados autores que abordam a tradução e seus processos. Esse levantamento inicial foi essencial para reunir ideias, problemas e possíveis soluções.

Após a consolidação desse embasamento teórico, o foco foi direcionado à identificação e síntese das principais contribuições no campo das tecnologias assistivas, com ênfase em ferramentas de reconhecimento e tradução de sinais. Essa etapa incluiu a análise de projetos acadêmicos anteriores sobre visão computacional e aprendizado de máquina.

Durante o desenvolvimento prático da solução, fez-se necessário revisar bibliografias técnicas relacionadas a cada desafio enfrentado, especialmente no que se refere as linguagens de programação, documentação de bibliotecas e *frameworks* utilizados.

4 Desenvolvimento

Neste capítulo, serão detalhadas as etapas do desenvolvimento do projeto, com ênfase nas atividades de planejamento, definição de requisitos, cronograma de trabalho e o processo de implementação do reconhecimento gestual. Também serão abordados o desenho da arquitetura de software, os diagramas de caso de uso, a concepção do projeto de banco de dados, além dos processos de prototipação e implementação das funcionalidades previstas.

O planejamento foi fundamental para organizar e acompanhar as atividades do projeto, estruturado a partir de um cronograma detalhado que estabeleceu a meta das etapas subsequentes. Os requisitos do sistema foram identificados e divididos em funcionais, que descrevem as funcionalidades principais, e não funcionais, relacionados a aspectos como desempenho, segurança e usabilidade. Diagramas de caso de uso foram elaborados para representar os principais fluxos de interação entre o sistema e os usuários.

Também são descritos os desafios e as soluções encontradas durante o período de implementação do reconhecimento gestual, utilizando técnicas de visão computacional e aprendizado de máquina para garantir o funcionamento da interpretação dos sinais de Libras.

Por fim, serão apresentadas as fases de prototipação e implementação, onde as ideias propostas foram concretizadas em um sistema funcional, com ajustes e refinamentos contínuos ao longo do desenvolvimento.

4.1 Planejamento do Projeto

A fase de Planejamento foi fundamental para organizar e direcionar o desenvolvimento do projeto, estabelecendo os objetivos e metas das etapas subsequentes. Durante essa fase, foi realizada a organização estratégica das ideias, a escolha do tema e a definição clara do problema a ser abordado, como demonstra o Quadro 1.

O primeiro passo envolveu a seleção do tema, focado na criação de uma tecnologia voltada para Libras e acessibilidade. Essa escolha foi fundamentada pela crescente demanda por soluções tecnológicas que promovam acessibilidade, um tópico de relevância social e tecnológica discutido amplamente nos últimos anos.

Após a definição do tema, o problema identificado foi a dificuldade de comunicação fluente entre surdos e ouvintes em situações cotidianas. A justificativa do projeto se baseou na falta de soluções acessíveis que realizem a tradução de Libras em tempo real, além da defasagem no ensino da língua, apesar de ser um idioma oficial do Brasil. A falta de conhecimento da população ouvinte sobre Libras limita o acesso de pessoas surdas a

diversos ambientes.

Durante o planejamento, várias rodadas de discussão foram realizadas entre os membros do grupo para gerar ideias, explorar possibilidades tecnológicas e elaborar o escopo do projeto. Nessas reuniões, foram identificados os principais recursos necessários, bem como as limitações potenciais e os recursos disponíveis para o desenvolvimento. Rascunhos e esboços foram elaborados para auxiliar na visualização das abordagens discutidas, permitindo a formulação de uma solução viável para o problema.

Além disso, a equipe produziu um vídeo introdutório, conforme solicitado pela universidade, com o objetivo de apresentar o projeto de forma concisa. O material foi apresentado em março de 2024.

Outros aspectos relevantes dessa fase incluíram a elaboração do cronograma do projeto, a definição do professor orientador designado pela universidade e o estabelecimento de reuniões semanais de orientação para acompanhar e ajustar o progresso do projeto.

Quadro 1 – Cronograma

MESES	CRONOGRAMA											
	FEV	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV	DEZ	
Fase 1 - Formação do grupo e orientador												
Escolha do Grupo												
Escolha do Orientador												
Fase 2 - Planejamento e Definição												
Escolha do tema												
Definição do Problema e Justificativa												
Ideação e rodadas de discussão												
Preparar vídeo												
Fase 3 - Pesquisa e Análise												
Pesquisa do Problema e levantamento de dados												
Pesquisa de softwares semelhantes												
Análise de Design												
Fase 4 - Estruturação do projeto												
Elaborar Canvas												
Formalizar escolha e EAP												
Definir cronograma e reuniões												
Redigir monografia												
Fase 5 - Desenvolvimento do Software												
Definir ferramentas e padrões da interface												
Desenvolver protótipo e interface												
Arquitetura da solução e componentes do sistema												
Codificação e definição do repositório												
Fase 6 - Controle de Qualidade												
Validar componentes e concluir testes												
Validar com usuário final												
Fase 7 - Apresentação												
Apresentação de resultados												
Defesa na banca												

Fonte: Autores.

4.2 Requisitos do Sistema

Antigamente, os requisitos eram frequentemente considerados equivalentes às funcionalidades, representando tudo o que o software precisava realizar de forma funcional. No entanto, atualmente se reconhece que os requisitos de software vão além das funções. Eles abrangem, além das funcionalidades, metas, características e limitações que o sistema deve apresentar para atender a contratos, normas ou especificações conforme os usuários. De forma mais ampla, um requisito é uma condição essencial para atingir um determinado objetivo (HIGOR, 2013).

Nesse contexto, os requisitos são classificados em dois tipos: Requisitos Funcionais (RF), que descrevem o que o sistema deve fazer, e Requisitos Não-Funcionais (RNF), que definem as qualidades e limitações do sistema, como desempenho, segurança e usabilidade.

4.2.1 Requisitos Funcionais

Os requisitos funcionais dizem respeito às funcionalidades do sistema, ou seja, o que ele deve realizar e as informações que deve processar (HIGOR, 2013).

No Quadro 2, são apresentados os requisitos funcionais do LibrasConnect, que descreve as funcionalidades essenciais que o sistema deve oferecer para atender aos objetivos do projeto. Esses requisitos foram definidos com base nas necessidades dos usuários e nas especificações do sistema, garantindo que o LibrasConnect proporcione uma experiência eficaz e intuitiva no sistema.

Quadro 2 – Requisitos Funcionais

REQUISITOS FUNCIONAIS			
RF	NOME DO RF	ATOR	DESCRIÇÃO
01	GERENCIAMENTO DE USUÁRIOS	ADM e Usuário	O sistema deve permitir o cadastro e autenticação de usuários.
		Usuário	O usuário deve poder acessar o sistema por meio de login e senha.
		Usuário	O sistema deve fornecer funcionalidade de recuperação de senha, através do e-mail
02	REPOSITÓRIO DE VÍDEOS	ADM e Usuário	O sistema deve disponibilizar um repositório de vídeos com gestos em Libras e ASL
		ADM e Usuário	O sistema deve permitir a busca por gestos específicos através de palavras-chave.
03	RECONHECIMENTO DE GESTOS	Usuário	O sistema deve capturar os gestos feitos pelo usuário por meio da câmera e reconhecê-los em tempo real
		Usuário	O software deve identificar o gesto realizado e mostrar sua tradução correspondente.
		Usuário	O sistema deve oferecer feedback visual ou textual sobre o gesto reconhecido
04	TRADUÇÃO DE GESTOS	Usuário	O sistema deve permitir que os usuários traduzem gestos de Libras para ASL e vice-versa
05	ATUALIZAÇÃO DE CONTEÚDO	ADM	O sistema deve permitir a adição de novos vídeos e gestos
		ADM	O sistema deve permitir a moderação e revisão do conteúdo adicionado

Fonte: Autores.

4.2.2 Requisitos Não Funcionais

A qualidade de um software é uma meta fundamental no processo de desenvolvimento. Ao desenvolver um produto digital, é indispensável definir antecipadamente as características de qualidade que se deseja alcançar para orientar o projeto adequadamente. Nesse cenário, a norma ISO/IEC 25010, introduzida em 2011 como sucessora da ISO/IEC 9126, apresenta um modelo amplamente aceito para a qualidade de software (SILVA, 2024).

Este modelo especifica um conjunto de atributos e categorias de qualidade, bem como as relações entre eles. Sua estrutura serve como base para a definição dos requisitos

de qualidade e para a avaliação do software.

Entre esses requisitos, os requisitos não funcionais são de extrema importância, pois determinam os critérios que qualificam os requisitos funcionais. Esses critérios abrangem aspectos de qualidade do software, como segurança, usabilidade, desempenho, confiabilidade, portabilidade, manutenibilidade e compatibilidade. Esses elementos, estabelecidos no modelo de qualidade da ISO/IEC 25010, são essenciais para garantir que o software atenda às expectativas de desempenho, experiência do usuário e normas de robustez e eficiência do sistema (SILVA, 2024).

No Quadro 3, estão especificados os requisitos não funcionais de Segurança do LibrasConnect.

Quadro 3 – Requisitos Não Funcionais - Segurança

REQUISITOS NÃO FUNCIONAIS			
RNF	CARACTERÍSTICA	SUBCARACTERÍSTICA	DESCRIÇÃO
01	SEGURANÇA	Integridade	Devem ser realizadas cópias (backups) de todos os dados do sistema a cada 2 meses.
		Confiabilidade	O sistema deve ser protegido contra ataques DDoS para evitar a sobrecarga do servidor
		Confidencialidade	As comunicações entre servidor de dados e usuários devem ser criptografadas (SSL TLS)
			O sistema deve utilizar um arquivo .env para armazenar de forma segura as variáveis de ambiente, garantindo que informações sensíveis não sejam expostas.

Fonte: Autores.

No Quadro 4, estão especificados os requisitos não funcionais de Usabilidade do LibrasConnect, que são essenciais para garantir que o sistema não apenas funcione corretamente, mas também ofereça uma experiência de uso intuitiva e agradável para os usuários

Quadro 4 – Requisitos Não Funcionais - Usabilidade

REQUISITOS NÃO FUNCIONAIS			
RNF	CARACTERÍSTICA	SUBCARACTERÍSTICA	DESCRIÇÃO
02	USABILIDADE	Aprendizagem	A interface deve ser simples e intuitiva, facilitando o acesso rápido aos vídeos e à funcionalidade de reconhecimento de gestos
		Adequação	O site deve ser responsivo e adaptável a diferentes dispositivos, como smartphones, tablets e computadores.
		Operacionalidade	Deve fornecer suporte visual claro e feedback imediato após a execução de ações, como a busca ou o reconhecimento de gestos.

Fonte: Autores.

No Quadro 5, estão especificados os requisitos não funcionais de Desempenho do LibrasConnect, que são fundamentais para assegurar que o sistema opere de maneira eficiente e responsiva em diferentes condições de uso. Esses requisitos abordam diversos aspectos, como o tempo de resposta do sistema, a capacidade de processamento de dados, a utilização de recursos computacionais e a escalabilidade da aplicação.

Quadro 5 – Requisitos Não Funcionais - Desempenho

REQUISITOS NÃO FUNCIONAIS			
RNF	CARACTERÍSTICA	SUBCARACTERÍSTICA	DESCRIÇÃO
03	DESEMPENHO	Capacidade	O site deve suportar até 1000 usuários simultâneos sem degradação de desempenho.
		Comportamento Temporal	O tempo de carregamento de vídeos e reconhecimento de gestos deve ser rápido, com latência inferior a 200ms.
			O sistema deverá ter pelo menos 5 segundos de atualizações de tela (refresh).
		Utilização de Recursos	Os vídeos devem ser otimizados para streaming, evitando travamentos ou delays.

Fonte: Autores.

No Quadro 6, estão especificados os requisitos não funcionais de Confiabilidade do LibrasConnect, que são cruciais para garantir a estabilidade e a segurança do sistema durante sua operação. Esses requisitos incluem aspectos como a disponibilidade do serviço, a tolerância a falhas, a recuperação em caso de erros e a consistência dos dados processados, ressaltando a importância de implementar mecanismos que assegurem a continuidade das operações, mesmo diante de eventuais problemas técnicos.

Quadro 6 – Requisitos Não Funcionais - Confiabilidade

REQUISITOS NÃO FUNCIONAIS			
RNF	CARACTERÍSTICA	SUBCARACTERÍSTICA	DESCRIÇÃO
04	CONFIABILIDADE	Disponibilidade	O site deve ter alta disponibilidade, 24 horas e 7 dias de semana, evitando quedas frequentes ou interrupções no serviço
		Maturidade	O reconhecimento de gestos deve ser consistente, mantendo um nível elevado de precisão..
		Recuperabilidade	O tempo para reinicialização total da plataforma após a ocorrência de um erro terminal é de no máximo 30 segundos.

Fonte: Autores.

No Quadro 7, estão especificados os requisitos não funcionais de Portabilidade do LibrasConnect, que são essenciais para garantir que o sistema possa ser facilmente adaptado e utilizado em diferentes ambientes e plataformas. Esses requisitos abrangem a capacidade do software de ser transferido entre diversos sistemas operacionais, dispositivos e configurações de hardware, sem a necessidade de reconfiguração significativa.

Quadro 7 – Requisitos Não Funcionais - Portabilidade

REQUISITOS NÃO FUNCIONAIS			
RNF	CARACTERÍSTICA	SUBCARACTERÍSTICA	DESCRIÇÃO
05	PORTABILIDADE	Adaptabilidade	O site deve ser compatível com os navegadores modernos Chrome, Firefox, Safari, Edge, Opera e OperaGX.
		Capacidade de ser instalado	Deve funcionar em dispositivos com diferentes sistemas operacionais (Windows, Android, iOS).
		Capacidade para substituir	O código deve ser modular e flexível para facilitar futuras migrações para novas tecnologias ou ambientes.

Fonte: Autores.

No Quadro 8, estão especificados os requisitos não funcionais de Manutenibilidade

do LibrasConnect, que são fundamentais para garantir que o sistema possa ser facilmente mantido, atualizado e aprimorado ao longo do tempo. Esses requisitos incluem a capacidade de realizar modificações e correções de maneira eficiente, minimizando o tempo e os recursos necessários para a manutenção do software. A figura detalha aspectos como a modularidade do código, a clareza da documentação e a simplicidade das interfaces, que são cruciais para facilitar o trabalho dos desenvolvedores e administradores do sistema e impactando diretamente nas novas funcionalidades.

Quadro 8 – Requisitos Não Funcionais - Manutenibilidade

REQUISITOS NÃO FUNCIONAIS			
RNF	CARACTERÍSTICA	SUBCARACTERÍSTICA	DESCRIÇÃO
06	MANUTENIBILIDADE	Analisabilidade	O código deve ser bem documentado e seguir padrões de desenvolvimento, como Clean Code e SOLID, para facilitar atualizações e correções de bugs
			O código deve incluir uma funcionalidade de log, implementada com a biblioteca logging do Python, para auxiliar na identificação e correção de problemas, facilitando a manutenção.
		Modificabilidade	O sistema deve permitir a adição de novos vídeos e gestos de forma simples, sem a necessidade de modificações complexas
			O sistema deve usar o Git para controle de versão, permitindo o rastreamento eficiente de mudanças no código e facilitando o trabalho colaborativo e a manutenção
		Testabilidade	Deve haver testes automatizados, como testes unitários, de integração e de regressão, para garantir a integridade das funcionalidades após mudanças.
		Analisabilidade e Modularidade	O código deve seguir o padrão PEP8, garantindo consistência, legibilidade e organização, o que facilita a manutenção e expansão do código.

Fonte: Autores.

No Quadro 9, estão especificados os requisitos não funcionais de Compatibilidade do LibrasConnect, que são essenciais para assegurar que o sistema funcione de maneira eficiente e integrada em diferentes ambientes tecnológicos.

Quadro 9 – Requisitos Não Funcionais - Compatibilidade

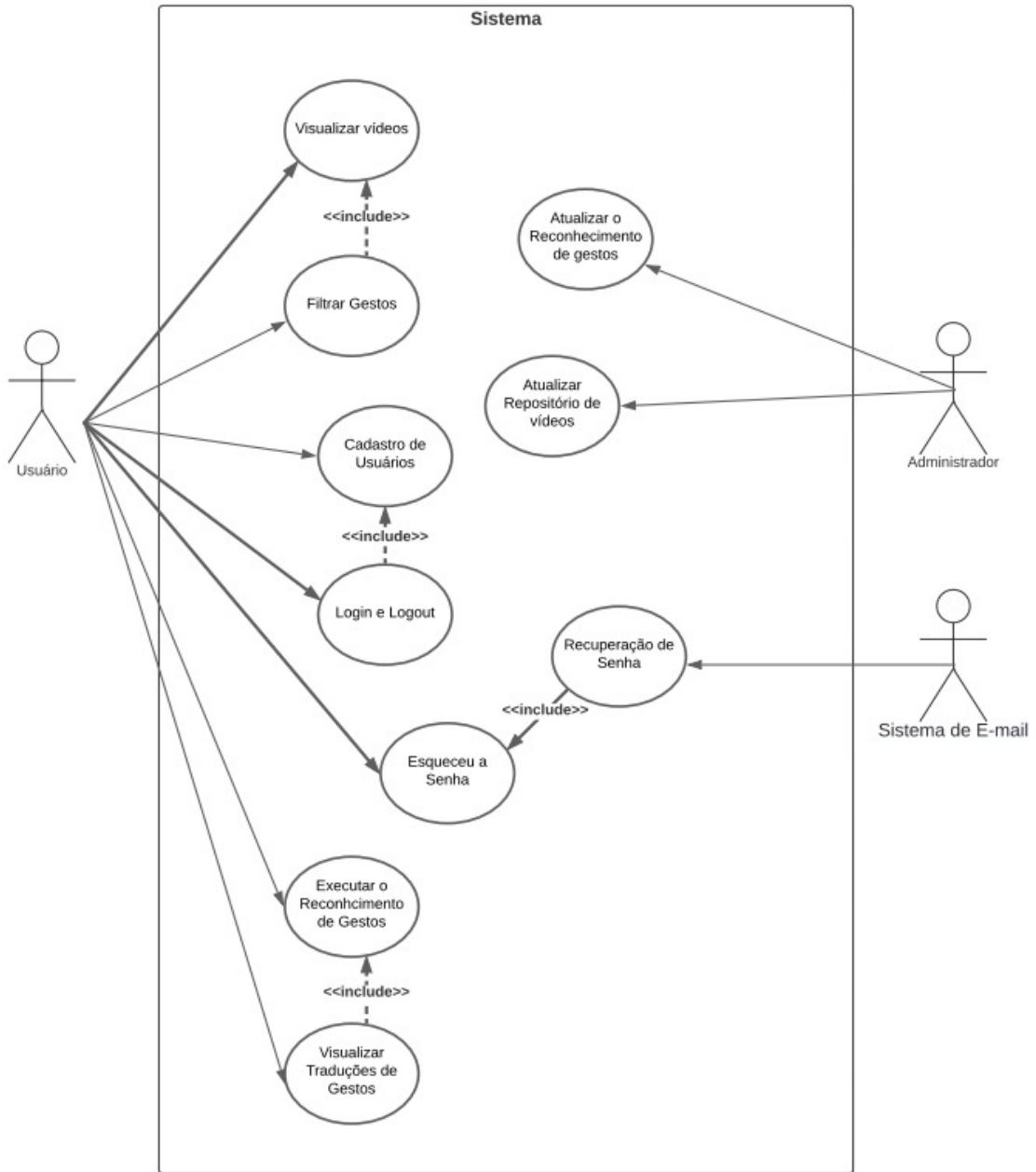
REQUISITOS NÃO FUNCIONAIS			
RNF	CARACTERÍSTICA	SUBCARACTERÍSTICA	DESCRIÇÃO
07	COMPATIBILIDADE	Coexistência	O sistema deve ser capaz de funcionar simultaneamente com outros aplicativos ou sistemas em execução no dispositivo do usuário, sem interferir no desempenho ou causar conflitos.
		Interoperabilidade	O sistema deve ser capaz de interagir com diferentes sistemas e plataformas, especialmente no que diz respeito à integração com o software de reconhecimento de gestos. Isso inclui garantir que o front-end do site seja compatível com diferentes navegadores e sistemas operacionais.

Fonte: Autores.

4.2.3 Diagrama de Casos de Uso

Os diagramas de casos de uso são empregados para modelar o comportamento de um sistema e auxiliar na captura de requisitos, abrangendo tanto os funcionais quanto os não funcionais. Esses diagramas também são eficazes para identificar as interações entre o sistema e os atores envolvidos. Por meio dos casos de uso e dos atores representados, é possível descrever as funcionalidades oferecidas pelo sistema e de que forma os atores interagem com ele, sem, no entanto, detalhar o funcionamento interno do sistema (IBM, 2021). Na figura 3, é apresentado o diagrama de casos de uso do LibrasConnect

Figura 3 – Diagrama de Casos de Uso



Fonte: Autores.

4.2.4 Reconhecimento Gestual

Para a execução do código que permitiu o reconhecimento gestual, utilizou-se como base o código fornecido pelo cientista de dados Nicholas Renotte, que apresenta um portfólio com uma aplicação básica de reconhecimento gestual utilizando LSTM e tf.keras (RENOTTE, 2021). Com base no código publicado por Renotte, foram aperfeiçoadas as

validações do modelo e desenvolvido um modelo próprio, mais complexo e estruturado, do início ao fim, incluindo a formulação da base de dados, uma das etapas essenciais do sistema de reconhecimento gestual. Diversas bibliotecas do *Python* foram empregadas para configurar o ambiente e tratar os dados, abrangendo também os testes de aferição da qualidade do algoritmo. O processo foi dividido em três partes principais: entrada de dados, treinamento do algoritmo e validação/execução final.

Primeiramente, realizou-se o armazenamento dos *landmarks*. Na parte inicial foi o momento que foram designados os pontos e traços utilizados para o reconhecimento dos gestos, como apresentado na Figura 4. Para a persistência de variáveis, foi necessário o uso da biblioteca *Mediapipe*, referida como “mp”.

Figura 4 – Variável com chamada de função

```
mp_holistic = mp.solutions.holistic
mp_drawing = mp.solutions.drawing_utils
```

Fonte: Autores.

Em seguida, a função apresentada foi utilizada para configurar o estilo dos pontos e traços que foram representados no reconhecimento do gesto. Esta função foi necessária para identificar visualmente onde os traços estavam sendo determinados e as conexões entre os pontos, como mostrado na Figura 5.

Figura 5 – Configuração dos pontos(mp holistic) e traços(mp drawing)

```
def draw_styled_landmarks(image, results):
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_TESSELATION,
                             mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                             mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)
                            )
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                             mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                             mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                            )
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                             mp_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),
                             mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
                            )
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                             mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                             mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                            )
```

Fonte: Autores.

No momento seguinte, a função *media pipe detection*, mostrada na Figura 6, realizou a detecção de *landmarks* em uma imagem, utilizando um modelo da biblioteca *Mediapipe*. Ela recebeu uma imagem e um modelo como parâmetros de entrada, sendo responsável por preparar a imagem para o processamento e retornou tanto a imagem processada quanto os resultados da detecção.

Figura 6 – Configuração dos pontos(mp holistic) e traços(mp drawing)

```
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image.flags.writeable = False
    results = model.process(image)
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    return image, results
```

Fonte: Autores.

Na continuação, a Figura 7 ilustra a primeira parte e a 8 ilustra a continuação. As figuras demonstram que o processamento dos *frames* foi realizado em um *loop*, onde a imagem foi capturada, convertida para o formato adequado para o *Mediapipe*, e submetida à detecção de *landmarks*. Em referência ao *loop*, ele pode ser interrompido ao pressionar a tecla 'q'. Em seguida, os resultados dessa detecção foram desenhados na imagem e exibidos ao usuário em tempo real, e por fim foi utilizado uma função da biblioteca *matplotlib*, onde viabilizou a visualização do resultado.

Figura 7 – Aplicação do modelo mediapipe na câmera

```
cap = cv2.VideoCapture(0)
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:

    for action in actions:
        for sequence in range(start_folder, start_folder+no_sequences):
            for frame_num in range(sequence_length):

                ret, frame = cap.read()
                image, results = mediapipe_detection(frame, holistic)
                draw_styled_landmarks(image, results)

                if frame_num == 0:
                    cv2.putText(image, 'COMEÇANDO COLETA', (120,200),
                               cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255, 0), 4, cv2.LINE_AA)
                    cv2.putText(image, 'COLETANDO frames PARA {} Vídeo Num {}'.format(action.upper(), sequence), (15,12),
                               cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                cv2.imshow('OpenCV Feed', image)
                cv2.waitKey(2000)
```

Fonte: Autores.

Figura 8 – Aplicação do modelo mediapipe na câmera

```

else:
    cv2.putText(image, 'COLETANDO frames PARA {} Video Num {}'.format(action, sequence), (15,12),
               cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
    cv2.imshow('OpenCV Feed', image)

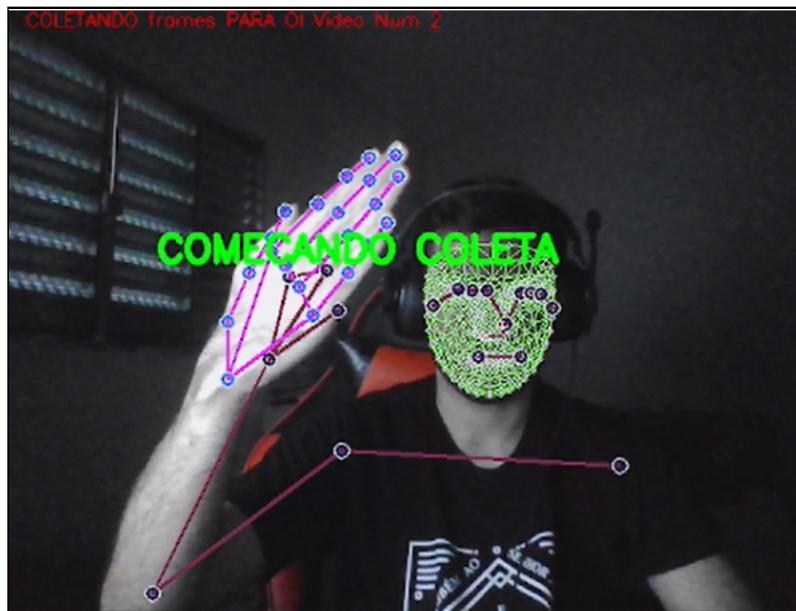
    keypoints = extract_keypoints(results)
    npy_path = os.path.join(DATA_PATH, action, str(sequence), str(frame_num))
    np.save(npy_path, keypoints)
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()

```

Fonte: Autores.

Nesse momento, a Figura 9 ilustrou a execução da abertura da câmera, com os pontos e traços sendo reconhecidos pela função *mediapipe detection*. Cada ponto da imagem representa uma coordenada, e esses valores foram armazenados para o treinamento do algoritmo. A coleta dos dados representados grava o gesto 30 vezes sequencialmente, onde cada repetição precisa ter algo de diferente dos treinos anteriores para que seja possível reconhecer o gesto em qualquer posição da janela de reconhecimento.

Figura 9 – Resultado do reconhecimento dos pontos



Fonte: Autores.

Na fase de armazenamento, ilustrada na Figura 10, foi determinado o número adequado de ocorrências de dados para posterior treinamento. Utilizou-se as bibliotecas *os* e *numpy* (referida como *np*), onde foi determinado o local para as pastas criadas, e o *numpy* definiu quais gestos foram treinados. Tanto a quantidade de gestos quanto a criação de pastas e arquivos foram determinadas em variáveis que posteriormente foram executadas de acordo com o número de repetições dos treinos.

Figura 10 – Configurações do ambiente de armazenamento de dados

```
DATA_PATH = os.path.join('MP_Data')
actions = np.array(['oi', 'obrigado', 'euteamo'])
no_sequences = 30
sequence_length = 30

start_folder = 0
```

Fonte: Autores.

Em seguida, a Figura 11 apresenta a quantidade de pastas criadas. Para cada gesto diferente, o *script* criou 30 pastas, conforme ilustrado na Figura 10

Figura 11 – Criação das pastas

```
for action in actions:
    action_dir = os.path.join(DATA_PATH, action)

    if os.path.exists(action_dir):
        dirmax = np.max(np.array(os.listdir(action_dir)).astype(int))
        for sequence in range(1, no_sequences+1):
            try:
                os.makedirs(os.path.join(action_dir, str(dirmax+sequence)))
            except:
                pass
    else:
        print(f"Directory {action_dir} does not exist.")
```

Fonte: Autores.

No momento seguinte, a Figura 12 exibiu o *loop* responsável por salvar os *landmarks* nos arquivos *.npy*. Dentro de cada pasta, são gerados 30 arquivos *.npy*, sendo que cada um deles contém as coordenadas do gesto treinado. Os dados armazenados nestes arquivos correspondem aos *landmarks* gravados sequencialmente durante cada sessão de treinamento.

Figura 12 – Armazenamento dos landmarks

```

sequences, labels = [], []
for action in actions:
    for sequence in np.array(os.listdir(os.path.join(DATA_PATH, action))).astype(int):
        window = []
        for frame_num in range(sequence_length):
            res = np.load(os.path.join(DATA_PATH, action, str(sequence), "{}.npy".format(frame_num)))
            window.append(res)
        sequences.append(window)
        labels.append(label_map[action])

```

Fonte: Autores.

Entrando na fase do treinamento do algoritmo, essa etapa utilizou-se funções da biblioteca *tensorflow.keras*, que permite o treinamento de redes neurais, semelhante ao funcionamento dos *landmarks*. Todas as configurações do modelo, incluindo o relatório de análise da eficiência, foram baseados nesta biblioteca.

Primeiramente foi efetuarmos a chamada das funções conforme é demonstrado na Figura 13.

Figura 13 – Funções necessárias

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Input, Dropout
from tensorflow.keras.callbacks import TensorBoard
from sklearn.model_selection import KFold
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
import os

```

Fonte: Autores.

No momento seguinte, ilustrado na Figura 14 foi realizar uma validação cruzada *K-Fold* em um modelo de rede neural adaptado para lidar com os *landmarks* utilizando a biblioteca *Keras*. Ele gera um modelo sequencial composto por várias camadas *LSTM* e densas, otimizadas para a tarefa de classificação. Em cada *fold*, os dados são divididos em conjuntos de treino e validação, onde o modelo é treinado e avaliado, e a precisão é registrada. Ao final, a precisão média de todos os *folds* é calculada e exibida, proporcionando uma avaliação robusta do desempenho do modelo.

Figura 14 – Definição e execução do modelo

```

model = Sequential()

kf = KFold(n_splits=10, shuffle=True, random_state=42)
fold_no = 1
accuracies = []

for train_index, test_index in kf.split(X):
    print(f'Training fold {fold_no}...')

    X_train, X_val = X[train_index], X[test_index]
    y_train, y_val = y[train_index], y[test_index]
    print(X_train.shape)

    model = Sequential()
    model.add(Input(shape=(30, 1662)))
    model.add(LSTM(64, return_sequences=True, activation='relu'))
    model.add(LSTM(128, return_sequences=True, activation='relu'))
    model.add(LSTM(64, return_sequences=False, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(len(actions), activation='softmax'))
    model.add(Dropout(0.3))

    model.add(Dense(64, activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(len(actions), activation='softmax'))
    model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])

    model.fit(X_train, y_train, epochs=3, batch_size=64, callbacks=[tb_callback])

    y_pred = (model.predict(X_val) > 0.30).astype(int)
    accuracy = accuracy_score(y_val, y_pred)
    print(f'Fold {fold_no} - Accuracy: {accuracy}')

    accuracies.append(accuracy)
    fold_no += 1
    del model
print(np.mean(accuracies))

```

Fonte: Autores.

Na Figura 15, representa o resultado obtido com as configurações determinadas nas camadas e densidades da rede neural. Cada alteração efetuada altera o resultado final. O treinamento do algoritmo foi feito através de alterações nas funções do *model.add* apresentado na figura 14.

Figura 15 – Resultado do treinamento

```

Epoch 23/30
2/2 0s 65ms/step - categorical_accuracy: 0.3667 - loss: 1.0972
Epoch 24/30
2/2 0s 65ms/step - categorical_accuracy: 0.3771 - loss: 1.0906
Epoch 25/30
2/2 0s 71ms/step - categorical_accuracy: 0.3719 - loss: 1.0921
Epoch 26/30
2/2 0s 62ms/step - categorical_accuracy: 0.3823 - loss: 1.0944
Epoch 27/30
2/2 0s 64ms/step - categorical_accuracy: 0.3719 - loss: 1.0960
Epoch 28/30
2/2 0s 67ms/step - categorical_accuracy: 0.3563 - loss: 1.0913
Epoch 29/30
2/2 0s 64ms/step - categorical_accuracy: 0.3719 - loss: 1.0945
Epoch 30/30
2/2 0s 64ms/step - categorical_accuracy: 0.3667 - loss: 1.0920
1/1 1s 552ms/step
Fold 10 - Accuracy: 0.0
0.0

```

Fonte: Autores.

Entrando na fase final, foi a fase de validar o modelo treinado. Nessa etapa foi necessário que os dados coletados e armazenados obtivessem uma boa acurácia para permitir a avaliação final.

O primeiro passo, ilustrado na Figura 16 destaca a função *prob viz*, que foi utilizada para visualizar as probabilidades de diferentes ações em um *frame* de vídeo. Ela desenhou barras coloridas no *frame*, com a largura de cada barra proporcional à probabilidade da ação correspondente. Cada barra foi colorida conforme uma lista predefinida de cores e acompanhada pelo nome da ação. A função retornou o *frame* modificado, que pode ser exibido ou salvo em outras partes do código

Figura 16 – Configuração da probabilidade do gesto executado

```

#11. Test in Real Time
from scipy import stats

colors = [(245,117,16), (117,245,16), (16,117,245)]
def prob_viz(res, actions, input_frame, colors):
    output_frame = input_frame.copy()
    for num, prob in enumerate(res):
        cv2.rectangle(output_frame, (0,60+num*40), (int(prob*100), 90+num*40), colors[num], -1)
        cv2.putText(output_frame, actions[num], (0, 85+num*40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2, cv2.LINE_AA)

    return output_frame

```

Fonte: Autores.

Em seguida, como ilustrado na Figura 17 demonstra a etapa que realizou a captura de vídeo em tempo real pela *webcam* e utilizou o *Mediapipe* para detectar *landmarks* corporais. Esses pontos-chave foram usados como entrada em um modelo de aprendizado profundo, que fez previsões sobre ações humanas. A cada 30 *frames*, o modelo gerou uma nova previsão, e se a probabilidade ultrapassasse um valor limite, a ação correspondente era

exibida no vídeo como texto, acompanhada de uma visualização gráfica das probabilidades. Além disso, o código inclui uma lógica para mostrar as ações mais recentes em sequência e encerrar a execução de maneira adequada quando solicitado.

Figura 17 – Execução do script utilizando o modelo treinado

```

sequence = []
sentence = []
predictions = []
threshold = 0.5

cap = cv2.VideoCapture(0)
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        ret, frame = cap.read()
        image, results = mediapipe_detection(frame, holistic)
        draw_styled_landmarks(image, results)

        keypoints = extract_keypoints(results)
        sequence.append(keypoints)
        sequence = sequence[-30:]

        if len(sequence) == 30:
            res = model.predict(np.expand_dims(sequence, axis=0))[0]
            print(actions[np.argmax(res)])
            predictions.append(np.argmax(res))

            if np.unique(predictions[-10:])[0]==np.argmax(res):
                if res[np.argmax(res)] > threshold:

                    if len(sentence) > 0:
                        if actions[np.argmax(res)] != sentence[-1]:
                            sentence.append(actions[np.argmax(res)])
                    else:
                        sentence.append(actions[np.argmax(res)])

                    if len(sentence) > 5:
                        sentence = sentence[-5:]

                    image = prob_viz(res, actions, image, colors)

                    cv2.rectangle(image, (0,0), (640, 40), (245, 117, 16), -1)
                    cv2.putText(image, ' '.join(sentence), (3,30),
                               cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

                    cv2.imshow('OpenCV Feed', image)

                    if cv2.waitKey(10) & 0xFF == ord('q'):
                        break
                cap.release()
                cv2.destroyAllWindows()

```

Fonte: Autores.

Por fim, essa abordagem permitiu uma análise contínua e em tempo real dos movimentos capturados, garantindo que o sistema fosse capaz de identificar e exibir ações de maneira precisa e eficiente, como ilustrado na Figura 18.

Figura 18 – Resultado do reconhecimento gestual



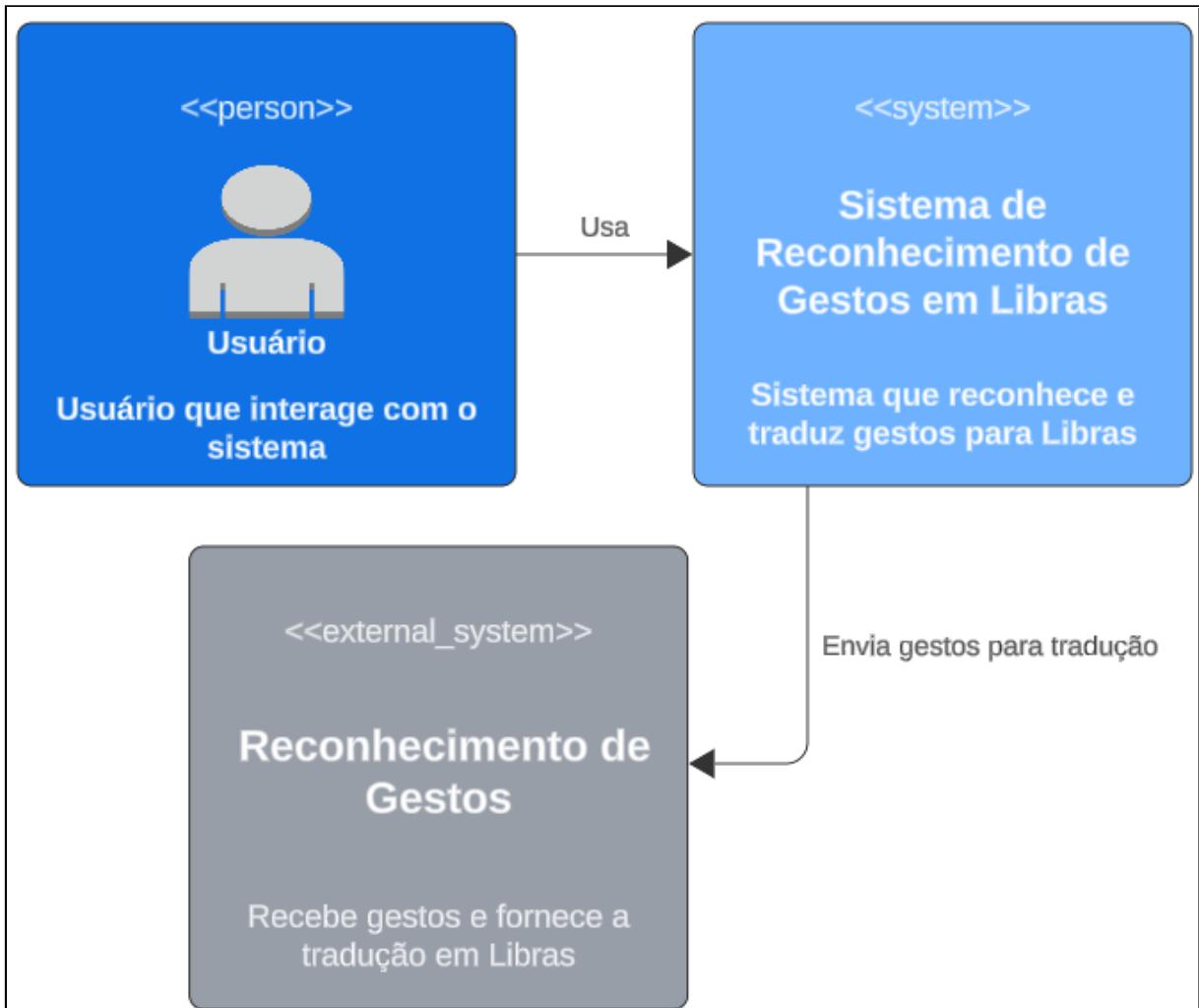
Fonte: Autores.

4.3 Desenho da Arquitetura do Software

O modelo C4 é uma metodologia de representação da arquitetura de software que divide a descrição de um sistema em quatro níveis distintos (CIPRIANO, 2021).

No primeiro nível da arquitetura, representado na Figura 19, é apresentada uma visão geral do sistema dentro do seu ambiente operacional, onde são identificados os principais atores envolvidos, incluindo tanto os usuários quanto os sistemas externos que interagem diretamente com o sistema em análise. Essa visão fornece uma perspectiva clara das principais interações entre o sistema e seu entorno, facilitando a compreensão das dependências e interfaces críticas. O Usuário, identificado como a entidade que utiliza diretamente o Sistema de Reconhecimento de Gestos em Libras, é responsável por acionar o sistema para realizar o processo de tradução, permitindo uma comunicação mais inclusiva para a comunidade surda. O sistema central, por sua vez, é encarregado de reconhecer e traduzir os gestos para Libras, atuando como o núcleo do processo de tradução. Além disso, o diagrama também destaca a presença de um Sistema Externo de Reconhecimento de Gestos, que recebe os gestos do usuário e os encaminha ao sistema central. Esse componente externo desempenha um papel crucial ao garantir que o processo de captura e envio de gestos seja eficiente, permitindo que o sistema central realize a conversão dos gestos e forneça a tradução de forma precisa e integrada.

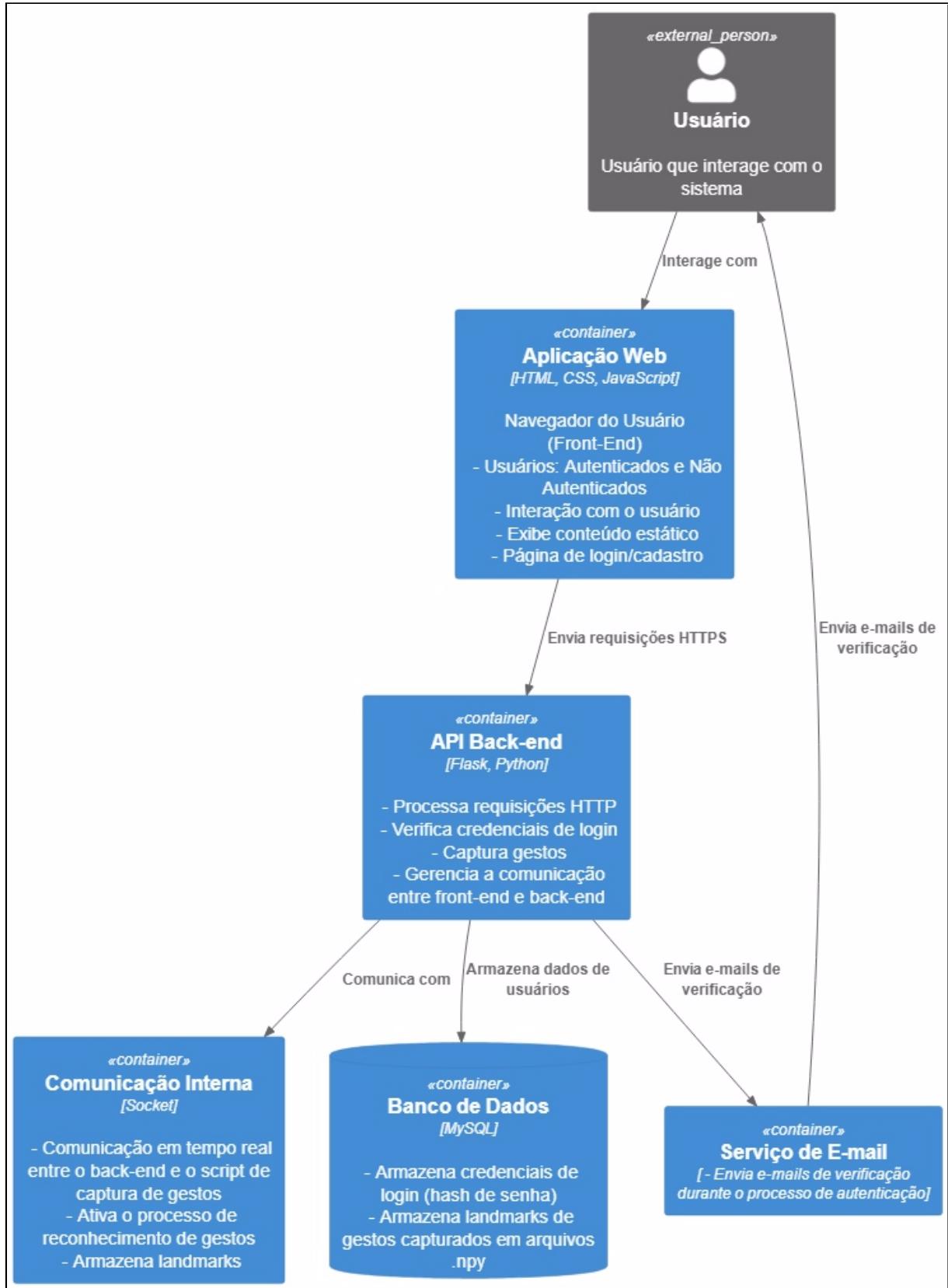
Figura 19 – Primeiro nível



Fonte: Autores.

No segundo ambiente da arquitetura, como ilustrado na Figura 20, o foco está nos principais contêineres que compõem o sistema e nas interações entre eles, proporcionando uma visão mais detalhada das funcionalidades e comunicação entre os módulos. O sistema é dividido em contêineres que representam as principais aplicações e serviços, como o *front-end*, o *back-end*, comunicação interna para captura de gestos, armazenamento de dados (Banco de Dados) e serviço de e-mail para verificação de usuários.

Figura 20 – Segundo nível

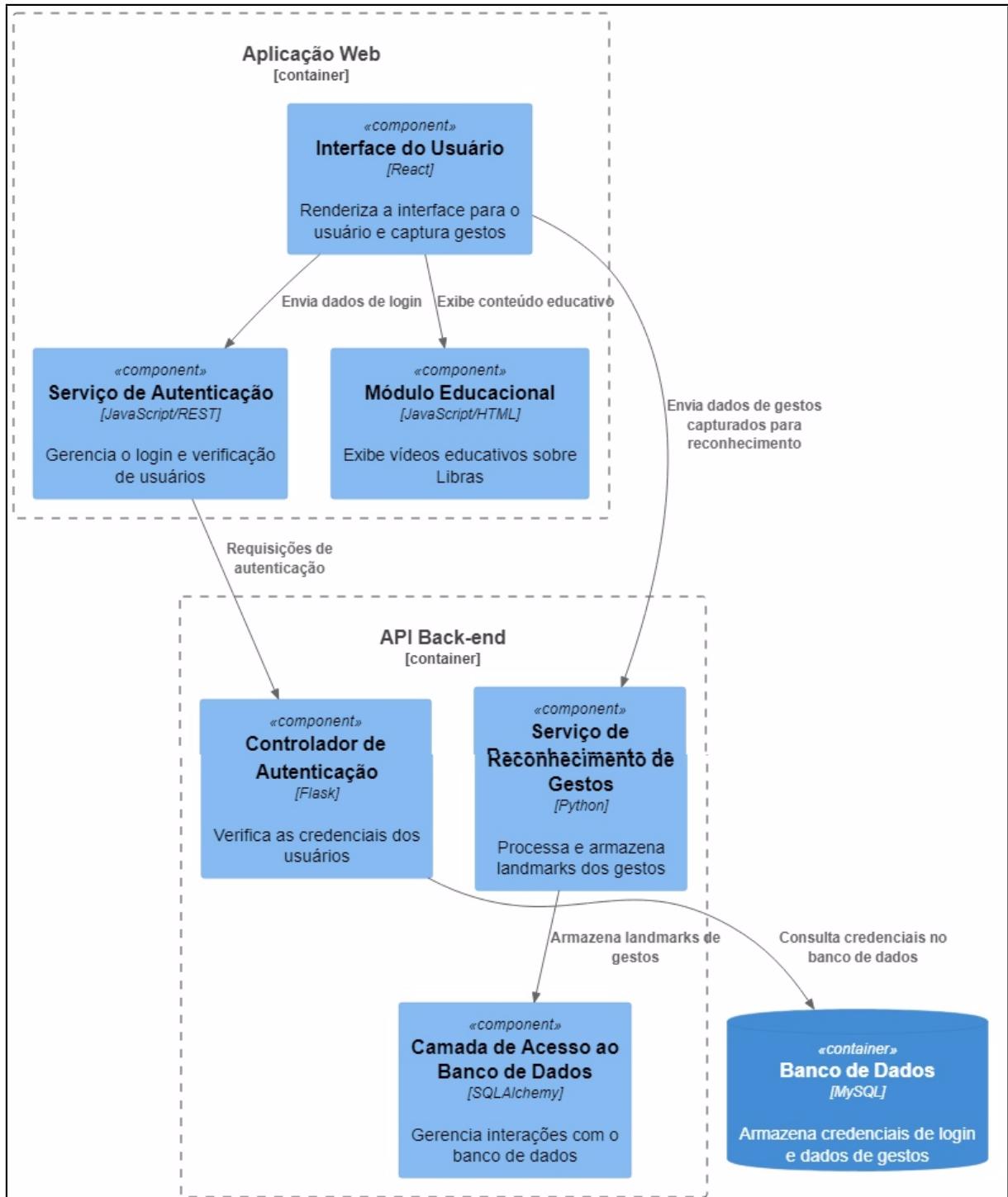


Fonte: Autores.

No terceiro ambiente da arquitetura, como ilustrado na Figura 21, o foco está nos

componentes específicos de cada contêiner e nas interações detalhadas entre eles. Esse nível descreve como os módulos internos de cada contêiner colaboram para realizar as funcionalidades do sistema, oferecendo uma visão granular dos serviços e fluxos de dados.

Figura 21 – Terceiro nível

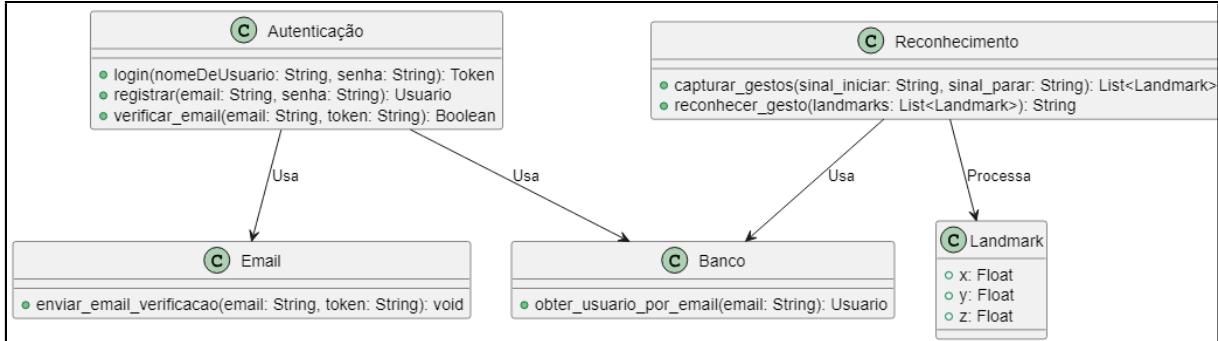


Fonte: Autores.

No quarto e último ambiente da arquitetura, como ilustrado na Figura 22, o foco é nas classes específicas e nos métodos detalhados de cada módulo. Esse nível proporciona

uma visão precisa das operações executadas por cada classe e como elas interagem para realizar funcionalidades específicas do sistema de reconhecimento de gestos.

Figura 22 – Quarto nível



Fonte: Autores.

4.4 Desenho do Projeto de Banco de Dados

Nesta seção, são apresentados os principais elementos do projeto de banco de dados para os sistemas propostos, incluindo tanto o reconhecimento de gestos quanto o sistema web.

4.4.1 Banco de dados do sistema web

O banco de dados utilizado no sistema web foi implementado em *MySQL* e tem como função principal o armazenamento dos dados de login dos usuários, bem como o conteúdo de mídia relacionado às explicações dos gestos do dicionário de sinais. No entanto, no módulo de reconhecimento de gestos, o banco de dados não foi necessário, pois os *landmarks* dos gestos são salvos em arquivos no formato *.npy*, adequados para o processamento eficiente de dados de reconhecimento de imagem. Na Figura 23 é possível visualizar os campos da tabela usuários: ID, Nome, Sobrenome, Email, Senha, Termos Aceitos, Data de Nascimento, Created In e Last Update.

4.4.2 Armazenamento de dados gestuais

Os dados gerados pelo reconhecimento gestual são estruturados em rótulos (*labels*) e *landmarks*. Para manipular esses dados, foi utilizada a biblioteca *NumPy*. Devido à grande quantidade de *landmarks* gerada por cada gesto, o armazenamento dessas informações em um banco de dados convencional foi considerado inviável. Como alternativa, adotou-se o uso de arquivos no formato *.npy*, nativos do *Python* e especificamente projetados para armazenar *arrays* ou matrizes de forma eficiente. Essa solução permite o salvamento e

Figura 23 – Estrutura do banco de dados do sistema web

usuarios	
!	id INT(11)
◆	nome VARCHAR(255)
◆	sobrenome VARCHAR(255)
◆	email VARCHAR(255)
◆	senha VARCHAR(255)
◆	termos_aceitos TINYINT(1)
◆	data_nascimento DATE
◆	created_in DATETIME
◆	last_update DATETIME
Indexes	

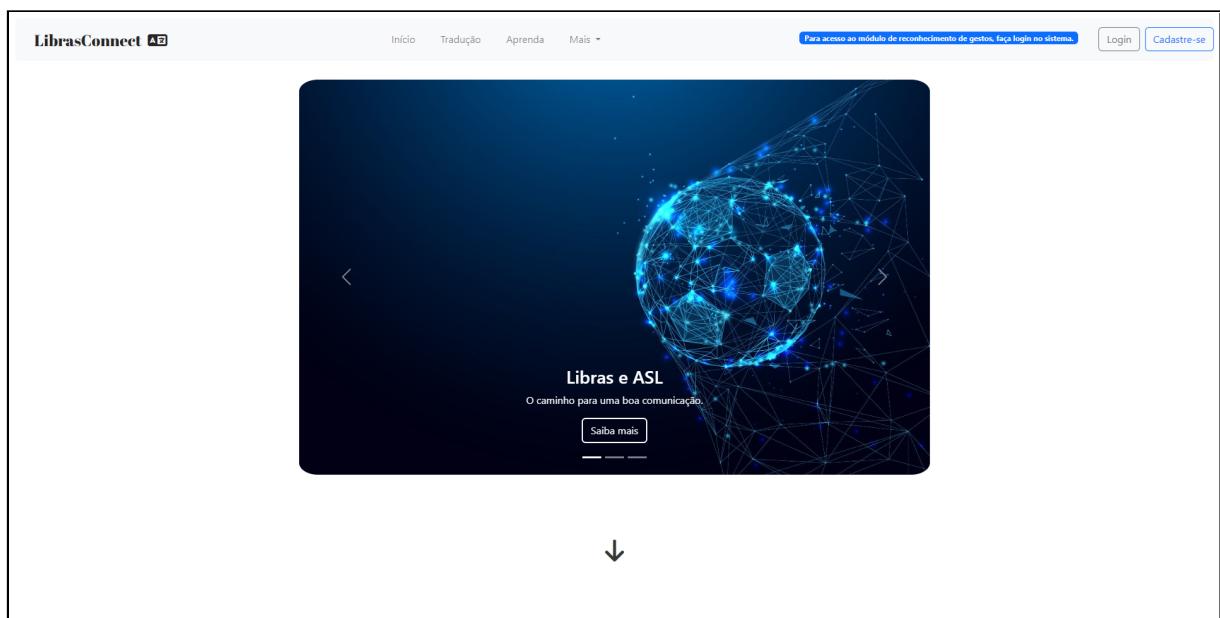
Fonte: Autores.

a recuperação dos *arrays* de maneira otimizada, proporcionando alto desempenho no processamento de dados (HARRIS et al., 2020).

4.5 Prototipação

A prototipação foi aplicada em mais de uma etapa do desenvolvimento. O primeiro protótipo consistiu em um site simples que demonstrava o funcionamento do módulo de dicionário para a pesquisa de gestos em Libras como ilustrado na Figura 24. A prototipação teve um papel crucial ao permitir a validação de ideias e requisitos antes da implementação final. Através de versões simplificadas da interface, foi possível testar a navegação no repositório de vídeos em Libras, possibilitando ajustes no design e na usabilidade, além de fornecer uma demonstração prática do uso previsto para a ferramenta.

Figura 24 – Início



Fonte: Autores.

O website incorpora uma tela de cadastro e login, demonstrado na Figura 25, que regista dados básicos do usuário. O cadastramento se faz necessário para permitir que o usuário salve verbetes favoritos do dicionário e acesse um histórico de gestos reconhecidos, proporcionando uma experiência mais personalizada. Além de atuar como uma barreira de segurança, protegendo contra exposição indevida, e evita o uso por robôs e algoritmos automatizados. Adicionalmente, o registro de usuários viabiliza o monitoramento da quantidade de acessos, permitindo escalar o sistema conforme a demanda, garantindo que ele opere de forma eficiente à medida que a base de usuários cresce.

Figura 25 – Cadastro

The screenshot shows a registration form titled "Cadastro". At the top center is a icon of three stylized human figures. Below the title, there are two input fields: "Nome" (Name) and "Sobrenome" (Last Name), both with placeholder text "Digite seu nome" (Type your name). Underneath these is a "Data de Nascimento" (Birth Date) field with a placeholder "dd/mm/aaaa" and a calendar icon. Next is an "E-mail" (Email) field with placeholder "Digite seu e-mail" (Type your email). Below that is a "Senha" (Password) field with placeholder "Digite sua senha" (Type your password) and a visibility toggle icon. Following is a "Confirme a Senha" (Confirm Password) field with placeholder "Confirme sua senha" (Confirm your password) and another visibility toggle icon. At the bottom of the form is a large blue "Cadastrar" (Register) button. Below the button, a link reads "Já possui cadastro? Faça login aqui" (Already have an account? Log in here).

Fonte: Autores.

A Figura 26 apresenta a tela do dicionário de gestos, anteriormente chamada de "Tela de Tradução". Durante o desenvolvimento, o projeto incorporou o reconhecimento de gestos, levando à divisão da tradução em dois módulos distintos: um para a pesquisa de gestos para falantes de português, e outro para o reconhecimento de gestos de um falante de Libras, com tradução para o português. Esse protótipo foi essencial para identificar que o projeto inicialmente contemplava apenas a tradução do português para Libras, sem abordar o sentido inverso. Como resultado, o foco foi ampliado para incluir a tradução de Libras para o português, utilizando reconhecimento de gestos e técnicas de *machine learning*. No entanto, o protótipo de tradução do português para Libras continua relevante para esse lado da tradução.

Figura 26 – Tradução

Tradução de Libras e ASL

Selecione uma frase em português (↑↓)
Select an english phrase option below (↑↓)

LIBRAS	ASL
Bem-vindo Bom dia Boa tarde Obrigado Bicicleta Bonito Cachorro	Welcome Good morning Good afternoon Thank you Bicycle Beautiful Dog

LIBRAS

ASL

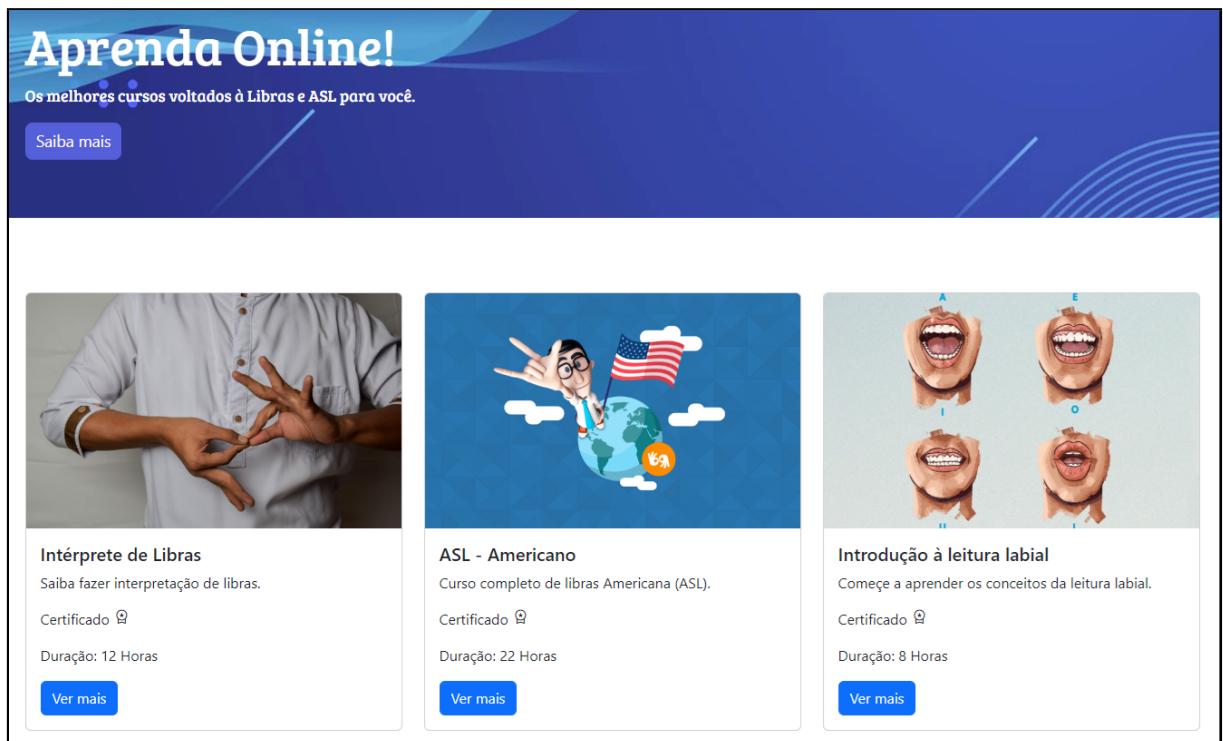
LIBRAS, BEM-VINDO (a)

You're welcome - Asl

Fonte: Autores.

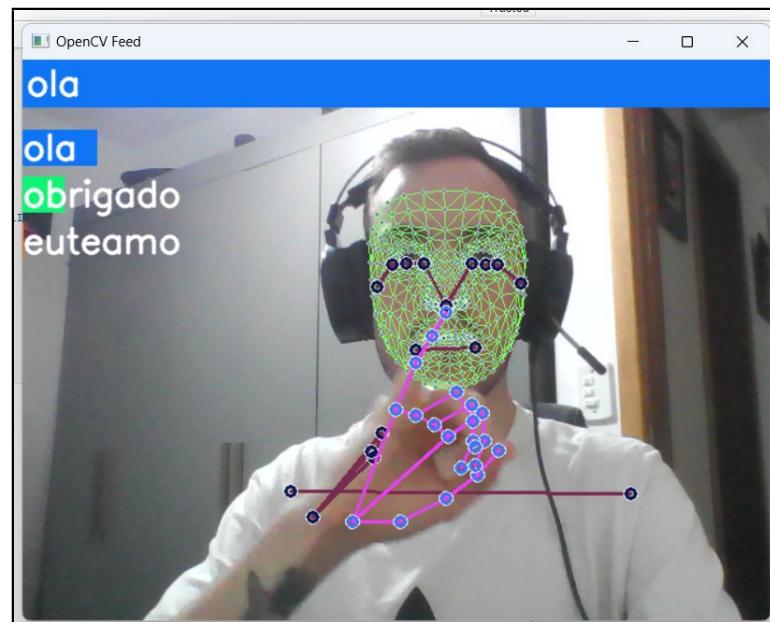
Para a prototipação, foram utilizados vídeos do *YouTube* incorporados através de um *iframe* em HTML. A funcionalidade demonstrava dois vídeos dispostos lado a lado, ensinando a execução da palavra tanto em Libras quanto em ASL.

A Figura 27 apresenta a tela de aprendizagem do site, cujo objetivo é fornecer uma plataforma para que professores de Libras publiquem seus cursos e para que alunos interessados localizem esses professores. O intuito é promover a conexão entre educadores e estudantes de Libras. Apesar de essa funcionalidade ter sido considerada nas prototipações, o *backend* necessário para seu funcionamento não foi implementado no projeto, servindo apenas como um exemplo que demonstra como seria a implementação com professores já cadastrados.

Figura 27 – Aprendizagem

Fonte: Autores.

A Figura 28 ilustra o funcionamento do sistema de reconhecimento de gestos em ação. Nesta interface, o usuário realiza um gesto em Libras, e a saída correspondente é exibida em tempo real na parte superior esquerda da tela, apresentando sua tradução.

Figura 28 – Reconhecimento de Gestos

Fonte: Autores.

4.6 Implementação

A arquitetura do sistema é dividida em dois componentes principais: o *front-end*, responsável pela interação com os usuários, e o *back-end*, que lida com o processamento e execução das funcionalidades do sistema.

4.6.1 Evidências da Implementação do *Front-End*

No desenvolvimento *front-end*, o sistema foi projetado para atender dois tipos de usuários: os autenticados e os não autenticados. Os usuários autenticados possuem acesso completo a todas as funcionalidades, incluindo a importante funcionalidade de captura de gestos, que é fundamental para o treinamento do sistema de reconhecimento gestual. Esses usuários, ao efetuarem login, têm suas credenciais validadas por meio de um processo que inclui a verificação do e-mail e a autenticação da senha, que está protegida por criptografia de hash. Caso as credenciais estejam corretas, o acesso é concedido; do contrário, uma mensagem de erro é exibida.

Por outro lado, os usuários não autenticados podem navegar pelas páginas e assistir a vídeos educativos sobre gestos em Libras e ASL, porém não têm acesso às funcionalidades de reconhecimento de gestos. Páginas como as de cadastro e login são reservadas apenas para aqueles que desejam contribuir com o treinamento do sistema.

O sistema oferece uma interface intuitiva com um menu de navegação que direciona o usuário para diferentes módulos. A página inicial apresenta painéis informativos sobre o sistema e a questão da surdez e perda auditiva. Em uma segunda página, os usuários têm acesso a uma área interativa para o aprendizado de frases específicas em Libras e ASL. A terceira página fornece guias e dicas para o desenvolvimento da comunicação em Libras, enquanto a quarta página está atualmente em desenvolvimento. Já a quinta página é exclusiva para usuários autenticados e permite a utilização da funcionalidade de reconhecimento de gestos.

4.6.2 Evidências da Implementação do *Back-End*

No desenvolvimento back-end, o sistema utiliza o *framework Flask*, escrito em *Python*, para facilitar a comunicação com as aplicações web. O *Flask* desempenha um papel essencial, criando rotas de comunicação que permitem o envio de dados entre o front-end e o back-end. Além disso, o sistema integra a biblioteca *Socket* para uma rota de comunicação interna, que realiza a transferência de dados entre o *Flask* e outros *scripts* do sistema.

O fluxo de dados ocorre da seguinte forma: quando um usuário interage com o sistema, o *framework Flask* envia um sinal que é processado pela rota do *Socket*. A partir desse ponto, o *script* de reconhecimento de gestos é executado. Durante o processo de reconhecimento, os *frames* capturados pela câmera são armazenados como *landmarks* em um arquivo .npy, que serve como classificador do gesto representado. Esses *landmarks* são essenciais para treinar o sistema, de modo que, ao reconhecer um gesto previamente registrado, o sistema possa identificar e converter o gesto em texto, exibindo-o na interface do usuário.

Para que o reconhecimento funcione corretamente, o sistema adota uma lógica específica: os *landmarks* de um *frame* só são gravados quando um sinal particular é recebido, como um intervalo de tempo ou o pressionamento de uma tecla. No caso atual, o gesto começa a ser capturado quando a tecla "S"(de "Start") é pressionada. A partir daí, todos os *landmarks* são armazenados no arquivo .npy, e o processo é repetido várias vezes em diferentes ângulos e posições da tela. Quando o treinamento do algoritmo é concluído, a tecla "Q"pode ser pressionada para encerrar o processo. Assim, o sistema se torna capaz de reconhecer gestos em qualquer posição da *webcam*.

5 Testes

A etapa de testes foi necessária para validar a eficácia e a confiabilidade do sistema desenvolvido, garantindo que todos os requisitos funcionais e não funcionais fossem atendidos conforme o especificado. Os testes realizados tiveram como objetivo principal identificar possíveis falhas e inconsistências, assegurando a robustez e a precisão das funcionalidades implementadas.

Os testes foram planejados para cobrir cenários que poderiam ser encontrados no uso cotidiano do sistema, assim como situações específicas que poderiam gerar problemas de funcionalidade. As seções a seguir descrevem em detalhes os planos de testes e os cenários utilizados, bem como os resultados obtidos após a execução dos testes.

5.1 Plano de Testes

O plano de testes foi estruturado para abranger duas partes essenciais do sistema: o módulo de Login e o módulo de reconhecimento de gestos. A abordagem adotada visou garantir a robustez e a eficiência de cada componente, verificando tanto a funcionalidade quanto o desempenho do sistema em diferentes cenários.

Para o sistema de Login, foram desenvolvidos testes de validação de senha, com o objetivo de assegurar que o mecanismo de autenticação fosse capaz de responder de maneira precisa a diferentes tipos de entradas fornecidas pelos usuários. Esses testes incluíram a verificação de credenciais corretas e incorretas, casos de uso de senhas fracas e outros tipos de validações.

No que tange ao módulo de reconhecimento de gestos, foram realizados testes específicos para medir a acurácia e a taxa de perda do algoritmo de reconhecimento de Libras. Esses testes tinham como objetivo avaliar a capacidade do sistema de identificar corretamente os gestos em diversos contextos, bem como sua eficiência em lidar com variabilidade nos padrões de movimento e ruído no ambiente. A taxa de acurácia foi analisada por meio de um conjunto de dados abrangente, que incluía diferentes gestos realizados por múltiplos usuários, permitindo uma avaliação rigorosa da precisão do sistema. Adicionalmente, a taxa de perda foi monitorada para identificar possíveis falhas na detecção e processamento dos gestos, fornecendo *insights* sobre a necessidade de ajustes no modelo de reconhecimento.

A abordagem combinada de testes funcionais e de desempenho permitiu uma validação abrangente dos principais componentes do sistema, contribuindo para a identificação de potenciais melhorias e para o aprimoramento da experiência do usuário final.

5.2 Cenários de Testes

Os cenários de testes foram definidos para avaliar de forma detalhada e precisa o comportamento de duas funcionalidades principais do sistema: o módulo de autenticação e o módulo de reconhecimento de gestos em Libras. No que se refere ao sistema de login, os cenários foram projetados para simular situações específicas que um usuário poderia encontrar ao interagir com o sistema, assegurando que todos os critérios de segurança fossem implementados corretamente, como verificação de complexidade e correspondência de senhas.

Além disso, para o módulo de reconhecimento de gestos em Libras, foram realizados testes de acurácia e taxa de erro, visando medir a precisão do sistema na tradução dos gestos para texto. Esses testes foram cruciais para avaliar a eficácia do algoritmo de reconhecimento, identificando a proporção de gestos traduzidos corretamente e as possíveis inconsistências que poderiam impactar a experiência do usuário.

5.2.1 Sistema de *Login*

O primeiro cenário no sistema de login foi verificar se o sistema aceita apenas senhas que contenham mais de 8 caracteres. O resultado obtido na Figura 29, ilustrou que o sistema funcionou conforme o esperado, aceitando apenas senhas com mais de 8 caracteres.

Figura 29 – Formulário exige senha com 8 caracteres

The screenshot shows a user registration form titled "Cadastro". The form includes fields for Name (Nome) and Surname (Sobrenome), both filled with "Silas" and "Ferreira" respectively. It also includes fields for Date of Birth (Data de Nascimento) set to "28/04/2000", Email (E-mail) set to "Testemail1@gmail.com", and Password (Senha) set to "teste12". A "Confirm Password" (Confirme a Senha) field contains "Confirme sua senha". A large blue "Cadastrar" (Register) button is at the bottom. Below the "Senha" field, a red error message states: "Senha deve conter no mínimo 8 caracteres." (Password must contain at least 8 characters.) At the bottom right, there is a link: "Já possui cadastro? Faça login aqui" (Already have an account? Log in here).

Fonte: Autores.

O segundo cenário no sistema de *login* foi validar se o sistema exige que as senhas contenham uma letra maiúscula e um caractere especial. O resultado obtido na Figura 30, ilustrou que o sistema funcionou conforme o esperado, aceitando apenas senhas com pelo menos uma letra maiúscula e um caractere especial.

Figura 30 – Formulário exige letra maiúscula e caractere especial

The screenshot shows a registration form titled "Cadastro". At the top center is a user icon consisting of three stylized human figures. Below the title, there are two input fields: "Nome" (Nome) containing "Silas" and "Sobrenome" (Sobrenome) containing "Ferreira". The next section is "Data de Nascimento" (Birth Date) with the value "28/04/2000" and a calendar icon. The "E-mail" field contains "Testemail1@gmail.com". The "Senha" (Password) field contains "teste123" and includes a visibility icon. The "Confirme a Senha" (Confirm Password) field contains "....." and includes a visibility icon. A large blue button at the bottom is labeled "Cadastrar" (Register). Below the button, two error messages are displayed in red: "Senha deve conter pelo menos uma letra maiúscula." and "Senha deve conter pelo menos um caractere especial.". At the bottom right, there is a link "Já possui cadastro? Faça login aqui" (Already have an account? Log in here).

Fonte: Autores.

O terceiro cenário no sistema de *login* foi assegurar que a senha contenha pelo menos uma letra maiúscula. O resultado obtido na Figura 31, ilustrou que o sistema funcionou conforme o esperado, aceitando apenas senhas com pelo menos uma letra maiúscula.

Figura 31 – Formulário exige letra maiúscula

The screenshot shows a user registration form titled "Cadastro". The form includes fields for Name (Nome and Sobrenome), Birth Date (Data de Nascimento), Email (E-mail), and Password (Senha). The password field contains "teste123!" and has a visibility toggle icon. The confirmation password field is empty. A large blue button labeled "Cadastrar" is at the bottom. Below the button, a red error message states "Senha deve conter pelo menos uma letra maiúscula." (The password must contain at least one uppercase letter.) At the bottom right, there is a link "Já possui cadastro? Faça login aqui".

Nome	Sobrenome
Silas	Ferreira

Data de Nascimento: 28/04/2000

E-mail: Testemail1@gmail.com

Senha: teste123!

Confirme a Senha:

Cadastrar

Senha deve conter pelo menos uma letra maiúscula.

Já possui cadastro? [Faça login aqui](#)

Fonte: Autores.

O quarto cenário no sistema de login foi garantir que o sistema exigisse ao menos um caractere especial na senha. O resultado obtido na Figura 32, ilustrou que o sistema funcionou conforme o esperado, aceitando apenas senhas com pelo menos um caractere especial.

Figura 32 – Caractere Especial

The screenshot shows a user registration form titled "Cadastro". The form includes fields for Name (Nome and Sobrenome), Birth Date (Data de Nascimento), Email (E-mail), and Password (Senha). The "Senha" field contains "teste123A" and has a visibility icon. The "Confirme a Senha" field contains "Confirme sua senha" and also has a visibility icon. A large blue button labeled "Cadastrar" is at the bottom. Below the "Cadastrar" button, a red error message states: "Senha deve conter pelo menos um caractere especial." At the bottom right, there is a link: "Já possui cadastro? Faça login aqui".

Fonte: Autores.

O quinto cenário no sistema de login foi garantir que o sistema exigisse ao menos um número na senha. O resultado obtido na Figura 33, ilustrou que o sistema funcionou conforme o esperado, aceitando apenas senhas com pelo menos uma número.

Figura 33 – Número na senha

The screenshot shows a user registration form titled "Cadastro". The form includes fields for Name (Nome and Sobrenome), Date of Birth (Data de Nascimento), Email (E-mail), and Password (Senha and Confirmar Senha). A blue "Cadastrar" button is at the bottom. A red error message "Senha deve conter pelo menos um número." is displayed below the password fields. A link "Já possui cadastro? Faça login aqui" is at the bottom right.

Nome	Sobrenome
Silas	Ferreira

Data de Nascimento: 28/04/2000

E-mail: Testemail1@gmail.com

Senha: teste!Abc

Confirmar Senha: Confirme sua senha

Cadastrar

Senha deve conter pelo menos um número.

Já possui cadastro? [Faça login aqui](#)

Fonte: Autores.

O sexto cenário no sistema de login foi verificar se o sistema permitia o registro quando as senhas inseridas nos campos (senha) e (confirmar senha) fossem iguais. O resultado obtido na Figura 34, ilustrou que o sistema funcionou conforme o esperado, aceitando apenas senhas iguais nos dois campos.

Figura 34 – Senhas iguais

The screenshot shows a user registration form titled "Cadastro". The form includes fields for Name (Nome and Sobrenome), Birth Date (Data de Nascimento), Email (E-mail), and Password (Senha and Confirmar Senha). Both password fields contain the same value: "teste!A1". A red error message at the bottom states: "A confirmação da senha não corresponde." (The password confirmation does not match).

Nome	Sobrenome
Silas	Ferreira

Data de Nascimento: 28/04/2000

E-mail: Testemail1@gmail.com

Senha: teste!A1

Confirmar Senha: teste!A2

Cadastrar

A confirmação da senha não corresponde.

Já possui cadastro? [Faça login aqui](#)

Fonte: Autores.

O sétimo cenário no sistema de login foi garantir que mesmo quando as senhas fossem iguais, ainda atendesse ao critério de ter mais de 8 caracteres. O resultado obtido na Figura 35, ilustrou que o sistema funcionou conforme o esperado, aceitando apenas senhas iguais nos dois campos e contendo 8 caracteres.

Figura 35 – Senhas iguais com 8 caracteres

The screenshot shows a user registration form titled "Cadastro" (Registration) with a user icon at the top. The form fields include:

- Nome (Name): Silas
- Sobrenome (Last Name): Ferreira
- Data de Nascimento (Date of Birth): 28/04/2000
- E-mail (Email): Testemail1@gmail.com
- Senha (Password): 1tes!@A
- Confirme a Senha (Confirm Password): 1tes!@A

A blue button labeled "Cadastrar" (Register) is at the bottom. A red error message "Senha deve conter no mínimo 8 caracteres." (Password must contain at least 8 characters) is displayed above the confirm password field. Below the button, there is a link "Já possui cadastro? Faça login aqui" (Already have an account? Log in here).

Fonte: Autores.

O oitavo cenário no sistema de login foi assegurar que o sistema permitisse o registro apenas quando as senhas inseridas fossem iguais e cumprisse todos os requisitos (comprimento, maiúsculas, números e caracteres especiais). O resultado obtido na Figura 36, ilustrou que o sistema funcionou conforme o esperado, aceitando apenas senhas iguais nos dois campos e cumprindo todos os requisitos.

Figura 36 – Senha e os requisitos cumpridos

The screenshot shows a user registration form titled "Cadastro". At the top center is a icon of three people. The form fields include:

- Nome:** Silas
- Sobrenome:** Ferreira
- Data de Nascimento:** 28/04/2000
- E-mail:** Testemail1@gmail.com
- Senha:** teste!@A2
- Confirme a Senha:** teste!@A2

A large green button at the bottom is labeled "Cadastrar". Below the button, a link says "Já possui cadastro? Faça login aqui".

Fonte: Autores.

E o último cenário no sistema de login foi testar se o botão de exibir/ocultar senha funcionasse corretamente. O resultado obtido na Figura 37 mostra o botão desativado, e na Figura 38 mostra ele ativado, ilustrando que o sistema funcionou conforme o esperado, mostrando que o botão funcionou corretamente.

Figura 37 – Exibir senha desativado

The screenshot shows a user registration form titled "Cadastro" (Registration) with a profile icon at the top. The form fields include:

- Nome**: Silas
- Sobrenome**: Ferreira
- Data de Nascimento**: 28/04/2000
- E-mail**: Testemail1@gmail.com
- Senha**: teste!@A2 (with a visible password icon)
- Confirme a Senha**: (with a visible password icon)

A large green button labeled "Cadastrar" (Register) is at the bottom. Below it, a link says "Já possui cadastro? Faça login aqui" (Already have an account? Log in here).

Fonte: Autores.

Figura 38 – Exibir senha ativado

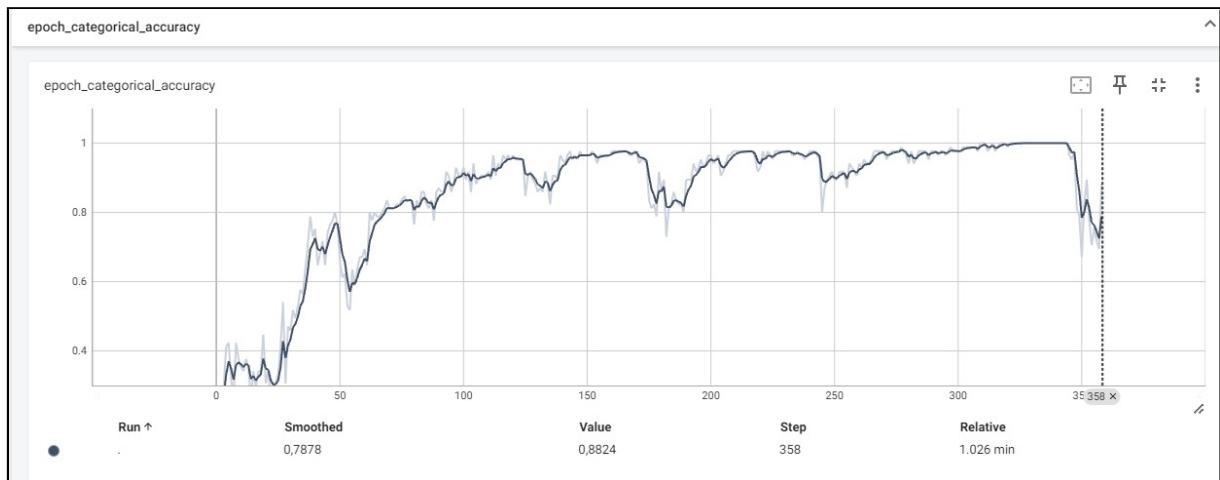
The screenshot shows a user registration form titled "Cadastro". At the top center is a icon of three stylized human figures. Below the title, there are two input fields: "Nome" (Name) containing "Silas" and "Sobrenome" (Last Name) containing "Ferreira". The next field is "Data de Nascimento" (Date of Birth) with the value "28/04/2000". Following that is an "E-mail" field with the value "Testemail1@gmail.com". Below these is a "Senha" (Password) field containing "....." and a "Confirme a Senha" (Confirm Password) field also containing ".....". Both password fields have eye icon buttons to toggle visibility. A large green button at the bottom is labeled "Cadastrar" (Register). At the bottom of the form, there is a link "Já possui cadastro? Faça login aqui" (Already have an account? Log in here).

Fonte: Autores.

5.2.2 Reconhecimento de Gestos

O primeiro cenário, durante o treinamento do algoritmo de reconhecimento de gestos em Libras, foi utilizado a ferramenta *TensorBoard* para monitorar e analisar o desempenho do modelo em tempo real. A Figura 39 apresenta o gráfico que ilustra a evolução da acurácia categórica ao longo das épocas de treinamento. Inicialmente, a acurácia oscilou em níveis mais baixos devido ao processo de ajuste dos parâmetros do modelo, mas à medida que o treinamento avançou, observou-se uma estabilização em torno de 0,88. Pequenas flutuações no desempenho são normais durante o aprendizado, mas a tendência geral indica que o modelo está convergindo para uma maior precisão, especialmente após a época 100. Esse monitoramento contínuo foi essencial para garantir que o modelo estivesse aprendendo de maneira eficiente e que os ajustes necessários pudessem ser feitos de forma dinâmica.

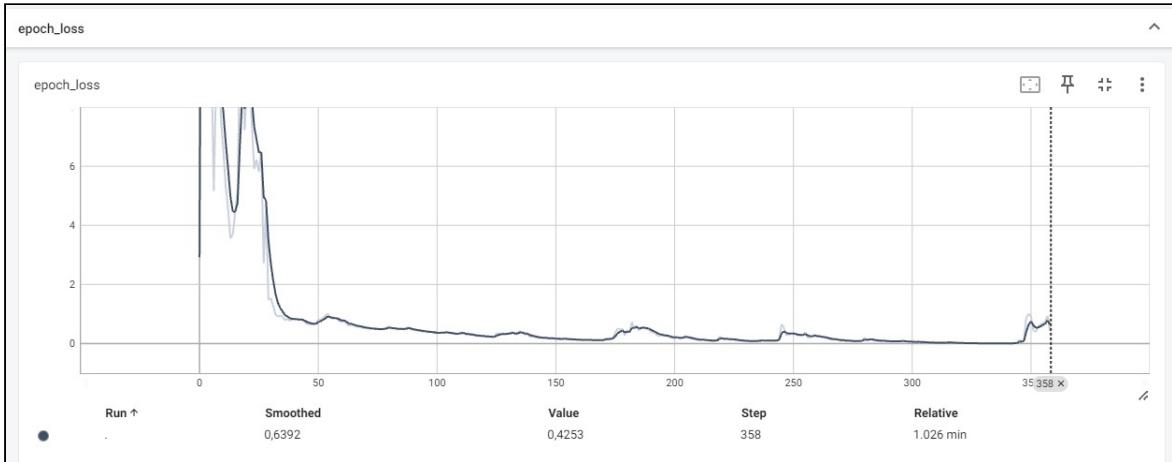
Figura 39 – Teste de Acurácia



Fonte: Autores.

No segundo cenário, foi utilizado a métrica de perda *loss*, que foi monitorada ao longo do treinamento por meio da ferramenta *TensorBoard*. A Figura 40 apresenta o gráfico que ilustra a evolução da função de perda durante as épocas de treinamento do modelo de reconhecimento de gestos em Libras. No início do processo, o *loss* apresentou valores elevados e instáveis, o que era esperado durante as primeiras iterações, pois o modelo ainda estava ajustando os pesos e aprendendo padrões nos dados. Conforme o treinamento avançou, observou-se uma queda significativa no *loss*, com estabilização a partir da época 50, atingindo valores em torno de 0,42. Esse comportamento indicou que o modelo estava minimizando os erros ao fazer previsões, sugerindo um aprendizado efetivo. No final do gráfico, pequenas flutuações no *loss* podem ser atribuídas a ajustes finos que o modelo continuou a fazer. Entretanto, o valor geral permaneceu baixo, indicando uma boa convergência.

Figura 40 – Teste de Loss (Perda)



Fonte: Autores.

No terceiro cenário, foi realizado um teste em tempo real utilizando a câmera do notebook. O modelo foi alimentado com dados de vídeo ao vivo, nos quais foi executado o gesto correspondente à saudação "Olá" em Libras. Conforme demonstrado na Figura 41, o algoritmo identificou corretamente o gesto e exibiu o *output* "Olá" na tela. O uso de *landmarks* (pontos de referência) faciais e de mãos, gerados pela malha de detecção do modelo, permitiu o rastreamento em tempo real dos movimentos, comprovando a eficácia do treinamento. Esse teste foi fundamental para validar o desempenho do modelo em um ambiente dinâmico e aproximar a aplicação do cenário final de uso, no qual os usuários poderão realizar gestos em Libras para reconhecimento imediato. Além do gesto de "Olá", foi realizado outro teste com o gesto de (Obrigado) em Libras onde o modelo também foi capaz de identificar corretamente este gesto, reforçando a precisão do modelo.

Figura 41 – Teste em tempo real



Fonte: Autores.

6 Conclusão

O projeto resultou no desenvolvimento de um sistema eficaz para o reconhecimento e tradução de gestos de Libras para o português, com capacidade de processar e interpretar sinais em tempo real. O módulo de dicionário de sinais complementou essa funcionalidade, permitindo a tradução inversa, de português para Libras, oferecendo uma ferramenta útil tanto para consulta quanto para o aprendizado de iniciantes na língua de sinais. As soluções desenvolvidas demonstraram viabilidade técnica e eficácia na mediação da comunicação entre surdos e ouvintes.

Apesar das limitações impostas pelo tempo e recursos disponíveis, o sistema atendeu aos objetivos inicialmente traçados, apresentando-se como uma ferramenta funcional e de potencial relevância para futuras pesquisas e implementações.

Além disso, o desenvolvimento do projeto permitiu a aplicação prática de técnicas de *Machine Learning* e o uso de ferramentas amplamente empregadas por programadores de IA. A experiência foi enriquecedora para os envolvidos, favorecendo o aprimoramento de habilidades técnicas e acadêmicas, além de proporcionar uma compreensão mais aprofundada das metodologias de IA aplicadas ao reconhecimento de gestos.

O impacto social de tecnologias como esta é imenso, ao promover a inclusão e viabilizar às pessoas surdas o exercício de seus direitos como cidadãos. Espera-se que este trabalho contribua para futuras iniciativas e pesquisas focadas na melhoria da qualidade de vida das pessoas com deficiência e no aperfeiçoamento da comunicação entre surdos e ouvintes, reafirmando o papel da tecnologia como instrumento de inclusão social.

Referências

BRASIL. *Lei nº 10.098, de 19 de dezembro de 2000 - estabelece normas de acessibilidade.* 2000. Disponível em: <https://www.planalto.gov.br/ccivil_03/LEIS/L10098.htm#art17>. Acesso em: 09/06/2024.

BRASIL. *Lei nº 10.436, de 24 de abril de 2002: Dispõe sobre a Língua Brasileira de Sinais - Libras e dá outras providências.* 2002. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/2002/l10436.htm>. Acesso em: 08/06/2024.

BRASIL. *Decreto nº 5.626, de 22 de dezembro de 2005.* 2005. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2004-2006/2005/decreto/d5626.htm>. Acesso em: 08/06/2024.

BRASIL. *Lei nº 12.527, de 18 de novembro de 2011 - dispõe sobre o acesso a informação.* 2011. Disponível em: <<https://www.jusbrasil.com.br/legislacao/1029987/lei-12527-11>>. Acesso em: 09/06/2024.

BRASIL. *Estatuto da Pessoa com Deficiência (Lei nº 13.146/2015).* 2015. Disponível em: <<https://www2.senado.leg.br/bdsf/bitstream/handle/id/513623/001042393.pdf>>. Acesso em: 09/06/2024.

BRASIL. *Conheça o INES.* 2021. Disponível em: <<https://www.gov.br/ines/pt-br/acesso-a-informacao-1/institucional/conheca-o-ines>>. Acesso em: 08/06/2024.

CAMPOS, M. de L. I. L. *Educação inclusiva para surdos e as políticas vigentes Em:Coleção UAB- UFSCar, Língua Brasileira de Sinais-Libras: uma introdução.* 2011. Disponível em: <http://livresaber.sead.ufscar.br:8080/jspui/bitstream/123456789/690/1/PE_LinguabrasileiradesinaisLibrasumaintroducao.pdf>. Acesso em: 07/06/2024.

CIPRIANO, L. *C4 Model.* 2021. Disponível em: <<https://medium.com/pravaler-digital-team/c4-model-9b6e56705496>>. Acesso em: 11/10/2024.

GANDRA, A. *OMS estima 2,5 bilhões de pessoas com problemas auditivos em 2050.* 2021. Disponível em: <<https://agenciabrasil.ebc.com.br/saudes/noticia/2021-03/oms-estima-25-bilhoes-de-pessoas-com-problemas-auditivos-em-2050>>. Acesso em: 02/09/2024.

GOOGLE. *Fluxo de trabalho de machine learning.* 2023. Disponível em: <<https://cloud.google.com/ai-platform/docs/ml-solutions-overview?hl=pt-br>>. Acesso em: 22/09/2024.

GOOGLE. *Guia de soluções do MediaPipe.* 2024. Disponível em: <<https://ai.google.dev/edge/mediapipe/solutions/guide?hl=pt-br>>. Acesso em: 22/09/2024.

HARRIS, C. R.; MILLMAN, K. J.; WALT, S. J. van der; GOMMERS, R.; VIRTANEN, P.; COURNAPEAU, D.; WIESER, E.; TAYLOR, J.; BERG, S.; SMITH, N. J.; KERN, R.; PICUS, M.; HOYER, S.; KERKWIJK, M. H. van; BRETT, M.; HALDANE, A.; RÍO, J. F. del; WIEBE, M.; PETERSON, P.; GÉRARD-MARCHANT, P.; SHEPPARD, K.; REDDY, T.; WECKESSER, W.; ABBASI, H.; GOHLKE, C.; OLIPHANT, T. E. *Array programming with NumPy.* 2020. Disponível em: <<https://www.nature.com/articles/s41586-020-2649-2>>. Acesso em: 24/09/2024.

HIGOR. *Introdução a Requisitos de Software*. 2013. Disponível em: <<https://www.devmedia.com.br/introducao-a-requisitos-de-software/29580>>. Acesso em: 09/06/2024.

IBGE. *PNS 2019: país tem 17,3 milhões de pessoas com algum tipo de deficiência*. 2021. Disponível em: <<https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/31445-pns-2019-pais-tem-17-3-milhoes-de-pessoas-com-algum-tipo-de-deficiencia>>. Acesso em: 08/06/2024.

IBGE. *Pessoas com deficiência têm menor acesso à educação, ao trabalho e à renda*. 2023. Disponível em: <<https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/37317-pessoas-com-deficiencia-tem-menor-acesso-a-educacao-ao-trabalho-e-a-renda>>. Acesso em: 09/06/2024.

IBM. *Press release, January 8, 1954. IBM-Georgetown Machine Translation Experiment*. 1954. Disponível em: <<https://aclanthology.org/www.mt-archive.info/IBM-1954.pdf>>. Acesso em: 08/06/2024.

IBM. *Modelos e Diagramas UML*. 2021. Disponível em: <<https://www.ibm.com/docs/pt-br/ras/7.5.0?topic=models-uml-diagrams>>. Acesso em: 11/10/2024.

KERAS. *Keras 3 API documentation*. 2024. Disponível em: <<https://keras.io/api/>>. Acesso em: 22/09/2024.

LIBRAS. *Leis, Decretos, Portarias, Resoluções*. 2020. Disponível em: <<https://www.libras.com.br/lei-de-libras>>. Acesso em: 09/06/2024.

LOUREIRO, C. de B. C. *Processo de apropriação da escrita da língua de sinais e escrita da língua portuguesa informática na educação de surdos*. 2004. Disponível em: <<https://lume.ufrgs.br/handle/10183/6447>>. Acesso em: 08/06/2024.

MATPLOTLIB. *API Documentation*. 2024. Disponível em: <<https://matplotlib.org>>. Acesso em: 22/09/2024.

MOZILLA. *MDN Web Docs - Python*. 2023. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Glossary/Python>>. Acesso em: 09/06/2024.

NUMPY. *NumPy documentation*. 2024. Disponível em: <https://numpy.org/devdocs/user/absolute_beginners.html>. Acesso em: 22/09/2024.

ONU. *Universal Declaration of Human Rights*. 1948. Disponível em: <<https://www.un.org/en/about-us/universal-declaration-of-human-rights>>. Acesso em: 08/06/2024.

ORACLE. *O que é o MySQL?* 2024. Disponível em: <<https://www.oracle.com/br/mysql/what-is-mysql>>. Acesso em: 08/06/2024.

PRETTO, N. D. L.; BONILLA, M. H. *Sociedade da Informação: democratizar o quê?* 2001. Disponível em: <https://www.academia.edu/250455/Sociedade_da_Informação_democratizar_o_quê>. Acesso em: 10/07/2024.

REDAÇÃO. *Serviço público precisa ter intérprete em Libras, determina projeto aprovado na CDH*. 2019. Disponível em: <<https://www12.senado.leg.br/noticias/materias/2019/09/26/servico-publico-precisa-ter-interprete-em-libras-diz-projeto-aprovado-na-cdh>>. Acesso em: 14/06/2024.

RENOTTE, N. *A practical implementation of sign language estimation using an LSTM NN built on TF Keras*. 2021. Disponível em: <<https://github.com/nicknochnack/ActionDetectionoforSignLanguage>>. Acesso em: 24/09/2024.

ROMÃO, T. L. C. *Aspectos Históricos e Práticos de Interpretação*. 1998. Disponível em: <<https://educacao.sme.prefeitura.sp.gov.br/wp-content/uploads/Portals/1/Files/19992.pdf>>. Acesso em: 13/06/2024.

SALTON, B. P.; AGNOL, A. D.; TURCATTI, A. *Manual de Acessibilidade em Documentos Digitais*. 2017. Disponível em: <<https://sites.riogrande.ifrs.edu.br/arquivos/1486518/manual-de-acessibilidade-em-documentos-digitais.pdf>>. Acesso em: 13/06/2024.

SCIKIT-LEARN. *API Reference*. s.d. Disponível em: <<https://scikit-learn.org/stable/api/index.html>>. Acesso em: 22/09/2024.

SILVA, R. *A ISO/IEC 25010 e sua importância para a qualidade de software*. 2024. Disponível em: <<https://blog.onedaytesting.com.br/iso-iec-25010/>>. Acesso em: 08/06/2024.

SKLIAR, C. *Bilingüismo e biculturalismo: Uma análise sobre as narrativas tradicionais na educação dos surdos*. 1997. Disponível em: <<http://projetoredes.org/wp/wp-content/uploads/Carlos-Skliar-1998.pdf>>. Acesso em: 10/06/2024.

SONZA, A. P. *Ambientes virtuais acessíveis sob a perspectiva de usuários com limitação visual*. 2008. Disponível em: <<https://lume.ufrgs.br/handle/10183/14661>>. Acesso em: 06/06/2024.

TENSORFLOW. *API Documentation*. 2024. Disponível em: <https://www.tensorflow.org/api_docs>. Acesso em: 22/09/2024.

VAILSHERY, L. S. *Most used web frameworks among developers worldwide, as of 2024*. 2024. Disponível em: <<https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/>>. Acesso em: 20/09/2024.