CURSO DE SISTEMAS DE INFORMAÇÃO

BOT DE ACESSIBILIDADE PARA DEFICIENTES AUDITIVOS E/OU VISUAIS ATRAVÉS DOS APLICATIVOS DE MENSAGENS WHATSAPP E TELEGRAM

DAVID BOTELHO MARIANO

Palmas - TO

2022

CURSO DE SISTEMAS DE INFORMAÇÃO

BOT DE ACESSIBILIDADE PARA DEFICIENTES AUDITIVOS E/OU VISUAIS ATRAVÉS DOS APLICATIVOS DE MENSAGENS WHATSAPP E TELEGRAM

DAVID BOTELHO MARIANO

Trabalho apresentado ao Curso de Sistemas de Informação da Universidade Estadual do Tocantins - UNITINS, como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação, sob a orientação do Prof. Esp. Jocivan Suassone Alves.

Palmas - TO

2022

CURSO DE SISTEMAS DE INFORMAÇÃO

BOT DE ACESSIBILIDADE PARA DEFICIENTES AUDITIVOS E/OU VISUAIS ATRAVÉS DOS APLICATIVOS DE MENSAGENS WHATSAPP E TELEGRAM

DAVID BOTELHO MARIANO

Documento assinado digitalmente
gov.br JOCIVAN SUASSONE ALVES
Data: 10/02/2023 16:17:15-0300
Verifique em https://verificador.iti.br

Prof. Esp. Jocivan Suassone Alves
Orientador

Documento assinado digitalmente
gov.br JOSE ITAMAR MENDES DE SOUZA JUNIOR
Data: 10/02/2023 15:20:47-0300
Verifique em https://verificador.iti.br

José Itamar Mendes de Souza Júnior
Examinador

Documento assinado digitalmente
gov.br GLEISON BATISTA DE SOUSA
Data: 10/02/2023 15:01:40-0300
Verifique em https://verificador.iti.br

Gleison Batista de Sousa
Examinador

Palmas - TO
2022

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da Universidade Estadual do Tocantins

M333b

MARIANO, David Botelho
Bot de acessibilidade para deficientes auditivos e/ou visuais através dos aplicativos de mensagens Whatsapp e Telegram.. David Botelho Mariano. - Palmas, TO, 2022

Monografia Graduação - Universidade Estadual do Tocantins – Câmpus Universitário de Palmas - Curso de Sistemas de Informação, 2022.

Orientador: Jocivan Suassone Alves

1. Acessibilidade.. 2. Transcrição de voz.. 3. Reconhecimento de imagem.. 4. Inteligência artificial..

CDD 610.7

Elaborado pelo sistema de geração automatica de ficha catalográfica da UNITINS com os dados fornecidos pelo(a) autor(a).

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE SISTEMAS DE INFORMAÇÃO DA FUNDAÇÃO UNIVERSIDADE ESTADUAL DO TOCANTINS - UNITINS

Aos **16** dias do mês de **Dezembro** de **2022**, reuniu-se de forma remota, via plataforma do google meet, às **10:30 horas**, sob a Coordenação do Professor **Jocivan Suassone Alves**, a banca examinadora de Trabalho de Conclusão de Curso em Sistemas de Informação, composta pelos examinadores Professor **Jocivan Suassone Alves** (Orientador), Professor **José Itamar Mendes de Souza Júnior** e Professor **Gleison Batista de Sousa**, para avaliação da defesa do trabalho intitulado **"Bot de Acessibilidade Para Deficientes Auditivos e/ou Visuais Através dos Aplicativos de Mensagens WhatsApp e Telegram"** do acadêmico **David Botelho Mariano** como requisito para aprovação na disciplina Trabalho de Conclusão de Curso (TCC). Após exposição do trabalho realizado pelo acadêmico e arguição pelos Examinadores da banca, em conformidade com o disposto no Regulamento de Trabalho de Conclusão de Curso em Sistemas de Informação, a banca atribuiu a pontuação: **9,0**.

Sendo, portanto, o Acadêmico: (X) Aprovado ( ) Reprovado

Assinam esta Ata:

gov.br Documento assinado digitalmente
JOCIVAN SUASSONE ALVES
Data: 10/02/2023 16:15:44-0300
Verifique em https://verificador.iti.br

Professor Orientador: Jocivan Suassone Alves

Examinador: José Itamar Mendes de Souza Júnior

gov.br Documento assinado digitalmente
JOSE ITAMAR MENDES DE SOUZA JUNIOR
Data: 10/02/2023 15:22:25-0300
Verifique em https://verificador.iti.br

Examinador: Gleison Batista de Sousa

gov.br Documento assinado digitalmente
GLEISON BATISTA DE SOUSA
Data: 10/02/2023 15:04:32-0300
Verifique em https://verificador.iti.br

Prof. Jocivan Suassone Alves
Presidente da Banca Examinadora

Este trabalho é dedicado a Deus, Nosso Senhor Jesus Cristo.

Agradecimentos

Agradeço primeiramente a Deus pelo dom da vida e por ter me proporcionado chegar até aqui.

Aos meus pais, Denilson Mariano de Brito e Sandra Maria Botelho Mariano que foram meu apoio emocional durante todos os momentos de adversidades.

Ao meu irmão Daniel Botelho Mariano, que incentivou a realização deste projeto.

Ao meu orientador Jocivan Suassone Alves, por aceitar, por confiar, e auxiliar no projeto desde a elaboração até o final do processo.

Aos professores, pelas correções e ensinamentos que me permitiram apresentar um melhor desempenho no meu processo de formação profissional ao longo do curso.

Aos meus colegas de curso, pelo companheirismo e trabalho em equipe nos momentos mais difíceis.

Escreva algo que valha a pena ler
ou faça algo que valha a pena escrever.
(Benjamin Franklin)

Resumo

Existe uma crescente utilização de smartphones entre os indivíduos ao redor do mundo com o advento da internet, possibilitando a conexão entre eles. Entretanto, as pessoas com dificuldade auditiva e/ou visual padecem ao empenhar-se na inclusão nesse ambiente digital, uma vez em que os recursos de acessibilidades são uma minoria nos aplicativos existentes, desta forma, se tornam excluídos neste contexto. Esse trabalho visa o desenvolvimento de uma ferramenta, a partir das linguagens de programação Python e Go, para expandir o acesso a este grupo através do uso nos principais aplicativos de mensagens (WhatsApp e Telegram), por meio de técnicas de inteligência artificial, nas funcionalidades de transcrição de voz e na interpretação de imagens. Com a finalidade de validação do objetivo, foi realizado testes de carga em prol de analisar a performance alcançada ao decorrer na elaboração da aplicação.

Palavras-chaves: acessibilidade, inteligência artificial, transcrição de voz, reconhecimento de imagem.

Abstract

There is a growing use of smartphones among individuals around the world with the advent of the internet, enabling the connection between them. However, people with hearing and/or visual difficulties suffer when striving for inclusion in this digital environment, since accessibility resources are a minority in existing applications, thus, they become excluded in this context. This work aims to develop a tool, based on the Python and Go programming languages, to expand access to this group through its use in the main messaging applications (WhatsApp and Telegram), through artificial intelligence techniques, in the voice transcription and image interpretation functionalities. In order to validate the objective, load tests were carried out in order to analyze the performance achieved during the development of the application.

Keywords: accessibility, artificial intelligence, voice transcription, image recognition.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AGI - *Artificial General Intelligence.*

API - *Application Programmin Interface.*

CNN - *Convolutional Neural Networks.*

IA - Inteligência Artificial.

ML - *Machine Learning.*

RNN - *Recurrent Neural Network.*

VPS - *Virtual Private Server.*

OCR - *Optical Character Recognition.*

Sumário

1 Introdução

Após a revolução tecnológica surgida pela fabricação dos primeiros celulares digitais, a principal funcionalidade na sua origem, a comunicação por ligação, transferiu esta posição para outras finalidades contidas nos aparelhos, como câmera e acesso a internet. Diante deste cenário, o manuseio dos aplicativos de mensagens, como WhatsApp[1] e Telegram[2], aumentaram de forma exponencial (COUTINHO, 2014).

Segundo Campelo et al (2011), uma parcela da população se encontra numa exceção nessa fase eletrônica devido a alta quantidade de programas para celulares que não proporcionam acessibilidade, sustentando a segregação destes indivíduos. De acordo com o resultado obtido do Censo 2010, 46 milhões de brasileiros, aproximadamente 24% dos habitantes, afirmaram possuir alguma deficiência (IBGE, 2010).

Em 6 de julho de 2015 foi sancionada a Lei nº 13.146, que trouxe significativa mudança no que se refere à inclusão social. O normativo citado preceitua que é dever do Estado, da sociedade e da família assegurar à pessoa com deficiência, a efetivação dos direitos referentes à acessibilidade, conforme o artigo 8º (BRASIL, 2015).

Tendo essa lei como referência, esse trabalho se justifica sobre auxiliar os integrantes desses grupos que possuem alguma dificuldade para impulsionar a inclusão social, tornando-os capacitados de terem uma maior participação no convívio perante a sociedade digital. Nessa perspectiva, a proposta visa o desenvolvimento de uma ferramenta, incluída nos aplicativos de mensagens, para transcrição de áudios e interpretação de imagens, uma vez em que essas pessoas apresentam deficiência auditiva não conseguem compreender áudios em suas conversas e pessoas com deficiência visual não são capazes de interpretar as imagens recebidas.

Para concluir os objetivos de acordo com a problemática exposta, este trabalho está composto da seguinte configuração: no prosseguimento deste Capítulo 1, se faz exposto o objetivo geral e os específicos. Em sequência, no Capítulo 2 estão presentes os referenciais teóricos citados. No Capítulo 3 é exibido a metodologia baseada para construção da aplicação. Por conseguinte, no Capítulo 4 estão os resultados obtidos durante este estudo e, por último, no Capítulo 5 aborda o desfecho e considerações finais.

[1] https://www.whatsapp.com
[2] https://www.telegram.org

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho propõe-se desenvolver uma ferramenta, incorporada nos aplicativos de mensagens WhatsApp e Telegram, para transcrever áudios e interpretar imagens, através de técnicas de inteligência artificial.

1.1.2 Objetivos Específicos

- Elaborar uma revisão de literatura sobre as temáticas abordadas neste trabalho.
- Identificar as principais aplicações externas (APIs) que fornecem recursos necessários para auxiliar na construção deste trabalho.
- Desenvolver *bots* em que realizam a intersecção entre as APIs e com funcionalidades de conversa para acessibilidade de transcrição e interpretação nos aplicativos de mensagens.
- Produzir testes de performance com a finalidade de analisar o desempenho dos algoritmos desenvolvidos.

2 Referencial Teórico

Este capítulo expressa o referencial teórico orientado aos temas: a inteligência artificial, o reconhecimento de voz, a visão computacional, os aplicativos de mensagem Telegram e WhatsApp, as linguagens de programação que foram empregadas e os trabalhos relacionados a este.

2.1 Inteligência Artificial

A inteligência artificial pode ser definida como os aspectos de aprendizagem ou outros atributos que possam ser demonstrados com exatidão em que uma máquina possa replicá-la (DICK, 2019).

Conforme Russel e Norvig (2004), a primeira realização reconhecida nesta área foi constituída por Warrem Macculloch e Walter Pitts em 1943 ao propor uma arquitetura de neurônios artificiais. Entretanto, o projeto no qual angariou maior notoriedade naquela época é de Alan Turing, com o Teste de Turing, onde provocou o questionamento fundamentado na inviabilidade de diferenciar os resultados entre um ser humano e um computador.

A programação de uma IA se concentra em três habilidades cognitivas: aprendizado, raciocínio e autocorreção. Esse aspecto do desenvolvimento de inteligência artificial se concentra na aquisição de dados e na criação de regras sobre como transformar os dados em informações acionáveis. Estas normas, que são chamadas de algoritmos, fornecem aos dispositivos de computação instruções sobre como concluir uma tarefa específica (WAGMAN, 2003).

A inteligência artificial pode receber dois tipos baseados na técnica em que foi utilizada: fraca e forte. A primeira é caracterizada como um sistema projetado e treinado para concluir uma tarefa específica, como por exemplo: robôs industriais e assistentes pessoais virtuais, como a Siri da Apple (WEI, 2019). Já a denominada forte, também conhecida como inteligência geral artificial (AGI), descreve a programação que pode replicar as habilidades cognitivas do cérebro humano. Em outros termos, consegue usar a lógica difusa para aplicar o conhecimento de um domínio a outro e encontrar uma solução de forma autônoma caso é apresentado uma tarefa desconhecida (NG; LEUNG, 2020).

No que concerne ao nível de desenvolvimento de uma IA, é possível classificá-la em quatro categorias: máquinas reativas, memória limitada, teoria da mente e autoconsciência. O *Machine Learning* (ML), subconjunto da inteligência artificial, é um campo da ciência da computação que usa algoritmos para processar grandes quantidades de dados e aprender

com eles. Ao contrário da programação tradicional baseada em regras, os modelos aprendem com os dados de entrada para fazer previsões ou identificar padrões significativos sem serem explicitamente programados para tal função (DU et al., 2017).

Conforme Nguyen (2021), existem diferentes classes de modelos de ML, dependendo da função e estrutura pretendidas, nos quais se rotulam em: aprendizado de máquina supervisionado, não supervisionado, por reforço ou *Deep Learning*. A seguir, suas características são especificadas de forma detalhada.

- **Aprendizado de Máquina Supervisionado:** modelo é treinado com dados de entrada rotulados que se correlacionam com uma saída especificada e continuamente refinado para fornecer uma saída mais precisa à medida que dados de treinamento adicionais se tornam disponíveis. Após o aprendizado, o modelo está habilitado em analisar dados adicionais para produzir a saída desejada. Torna-se bem-sucedido quando o modelo pode produzir consistentemente previsões precisas quando fornecido com novos conjuntos de dados (ZIMMER et al., 2019).
- **Aprendizado de Máquina Não Supervisionado:** os dados de entrada não são rotulados nem a saída especificada. Em vez disso, os modelos são alimentados com grandes quantidades de dados brutos e os algoritmos são projetados para identificar quaisquer padrões significativos subjacentes. Logo, os resultados de modelos de aprendizado de máquina não supervisionados são interpretados por humanos para determinar se são significativos e relevantes (BARYANNIS et al., 2019).
- **Aprendizado por Reforço:** o modelo aprende dinamicamente em prol de alcançar a saída desejada por meio de tentativa e erro. Se o algoritmo do modelo for executado corretamente e atingir a saída pretendida, ele será recompensado, caso contrário, se não produzir a saída desejada, é penalizado. Desta forma, o modelo aprende ao longo do tempo a atuar de forma a maximizar a recompensa líquida (HAYDARI; YILMAZ, 2020).
- **Deep Learning:** conhecido também como aprendizado profundo, é um modelo construído em uma rede neural artificial, no qual os algoritmos processam grandes quantidades de dados não rotulados ou não estruturados por meio de várias camadas de aprendizado de maneira inspirada em como as redes neurais funcionam no cérebro humano (CHEN; LIN, 2014).

Na atualidade, a inteligência artificial é um campo que encontra-se prestigiada e que está sendo difundida nos aplicações existentes na rotina dos indivíduos, tal como processamento de linguagem natural, sistemas visuais, robótica e entre outros, como por exemplo suas técnicas estarem contidas em veículos aéreos não tripulados e biometria para reconhecimento de voz e de imagens (GOMES, 2010).

2.2 Reconhecimento de Voz

A comunicação verbal é um modelo de transmissão e recebimento de informações, nesse contexto, uma inovação recente é a tecnologia de reconhecimento de voz em que se torna possível a conversão de mensagens de voz em textos, contribuindo em aspectos como traduções e acessibilidade para pessoas com deficiência (TRIVEDI et al., 2018).

Os sistemas de reconhecimento por voz são capazes de entender por parte da máquina o que fora pronunciado. Este processo consegue ser realizado através de duas formas opostas: dependente do locutor ou independente do locutor. Em síntese, no primeiro modo o sistema é treinado baseado em somente reconhecer a voz de uma pessoa específica enquanto o segundo consegue com vários locutores (BRESOLIN, 2008).

2.3 Visão Computacional

A área de visão computacional está relacionada ao manuseio e procedimentos com imagens, no qual visa o progresso de sistemas em que sua finalidade seja analisar e interpretá-las, pretendendo alcançar componentes significativos, por meio de capturas por câmeras, *scanners*, sensores, e outros dispositivos (PERELMUTER et al., 1995).

As imagens fotográficas capturadas por diversos dispositivos são avaliadas semelhantes a sinais bidimensionais, portanto, no formato digital conseguem ser determinadas como funções bidimensionais confeccionadas por um conjunto de elementos finitos, os *pixels*, que dispõe de posição e valor específico (COUTO, 2012). Neste âmbito, as imagens fotográficas, análogas à visão humana, sintéticas e térmicas são as predominantes nesse interesse.

De acordo com Milano e Honorato (2014), as aplicações de visão computacional são organizadas por: aquisição da imagem, pré-processamento, extração de características, segmentação e reconhecimento.

2.4 Tecnologias

2.4.1 Python

O Python[1] é uma linguagem de programação interativa e interpretada, provendo uma estrutura de dados de elevado grau como dicionários e listas, além de possuir tipagem dinâmica, gerência automática de memória e exceções. Desenvolvida por Guido van Rossum, dispõe de ser $open - source$[2], além de deter uma sintaxe simples, enfatizando

[1] https://www.python.org

[2] Aplicações em que o código-fonte é compartilhado para que a comunidade possa utilizar, modificar e aperfeiçoar

uma legibilidade no código (SANNER et al., 1999).

Sustenta diversos paradigmas de programação como a orientada a objetos, estrutural e funcional assim como estar disponível para muitos sistemas operacionais. O Python desfruta de bastantes bibliotecas nativas, em que auxiliam no desenvolvimento, suporta a conexão dos principais bancos de dados existentes e domina várias áreas de programação como processamento de imagens, robótica, análise de dados e aprendizado de máquina (SRINATH, 2017).

2.4.2 Go

A linguagem Go[3], desenvolvida pelos engenheiros do Google (Robert Griesemer, Rob Pike e Ken Thompson), surgiu em 2007 porém se tornou um projeto de código aberto apenas em 2009. O Go foi planejado para solucionar dificuldades que a empresa possuia, no qual utilizava-se de outras linguagens (Java, C++, etc.), como reduzir o tempo de compilação e possíveis erros pelo ponteiro, necessidade de declarar o tipo da variável podendo ser inferida e questões de infraestrutura tal qual o início da utilização de processadores multinúcleos e sistemas distribuídos nas aplicações previamente desenvolvidas (SCHMAGER, 2010).

Provendo uma sintaxe simples, orientada a objetos, compilada, contendo coletor de lixo, múltiplos retornos e fortemente tipada, o Go encontra-se focado nos quesitos de concorrência e no paradigma de multiprogramação. A linguagem está estruturada por pacotes, gerenciamento de dependências e, embora não possui herança, dispõe de incorporação, no qual uma interface engloba outras interfaces ou assinaturas de métodos. Corporações como Amazon, SoundCloud, Canonical, Adobe, Oracle e Uber usufruem dos recursos da linguagem Go (WESTRUP; PETTERSSON, 2014).

2.5 Aplicativos de Mensagens

2.5.1 WhatsApp

O aplicativo de mensagens WhatsApp, possui mais de 1 bilhão de usuários, e foi adquirido pelo Facebook em 2014, o mensageiro provê serviços, de forma gratuita, mensagens de texto e áudio, ligações e chamadas de vídeo, envio de arquivos de diversas extensões com limitação no tamanho (SUTIKNO et al., 2016).

Ademais, detém uma criptografia de ponta-a-ponta, isto é, um sistema de comunicação que permite apenas os participantes das mensagens tenham acesso ao conteúdo enviado, garantindo integridade e segurança aos usuários (RASTOGI; HENDLER, 2017).

[3] https://go.dev

2.5.2 Telegram

Ao contrário do WhatsApp, o Telegram é um aplicativo baseado em plataforma de código aberto criado por Pavel Durov em 2013. Possuindo semelhanças a nível da funcionalidade de conversa, o aplicativo diferencia-se por criar uma identificação única para o usuário (uma espécie de codinome) permitindo que consiga conversar com a outra parte, através de uma busca, sem a necessidade de salvar o número de telefone (SUTIKNO et al., 2016).

Por ser adepto ao $open - source$, o Telegram possui uma documentação completa para o consumo de suas APIs, favorecendo o desenvolvimento de *bots*[4] e enriquecendo o aplicativo com mais funcionalidades. Além disso, suporta até 200 mil usuários em grupos de mensagens e transferência de arquivos ilimitados (FARAMARZI; TABRIZI; CHALAK, 2019).

2.6 Trabalhos Relacionados

No decorrer da investigação na bibliografia e repositórios, foram descobertos variados projetos relacionados aos temas de visão computacional e reconhecimento por voz. O levantamento consistiu em analisar os principais portais de artigos científicos, dentre eles, Google Acadêmico[5], Scielo[6], science.gov[7] e CAPES[8], bem como à apreciação de repositórios, produzido na principal plataforma de armazenamento de $softwares$, o GitHub[9], que de acordo com o seu diretor de tecnologia Jason Warner (2018), contempla mais de 100 milhões de sistemas em seu domínio.

Durante a pesquisa por conteúdos relacionados, não foi possível encontrar artigos de acordo com a necessidade desse trabalho, no entanto, na plataforma GitHub existem *bots* desenvolvidos para o aplicativo de mensagens Telegram, como por exemplo, remover usuários que propagam discurso de ódio em um grupo, verificar o clima em determinada cidade e até mesmo jogos simples são encontrados. No entanto, quando o tema é acessibilidade, a plataforma possui poucos *bots* para auxiliar na inclusão de pessoas com deficiência.

Foi encontrado o OCR Bot[10], ferramenta criada por Sumanjay que detecta textos em uma imagem e transcreve para o usuário através da biblioteca Tesseract[11]. Outra ferramenta fora descoberta com a finalidade de transcrição de voz em texto, denominada

[4] Aplicações automatizadas que são executadas com um determinado objetivo
[5] https://scholar.google.com.br
[6] https://www.scielo.br
[7] https://www.science.gov
[8] https://www-periodicos-capes-gov-br.ezl.periodicos.capes.gov.br
[9] https://github.com
[10] https://github.com/cyberboysumanjay/ocrbot
[11] https://tesseract-ocr.github.io

Voicos[12] desenvolvida por Nikita Karpukhin porém, o idioma base de conversão é o russo.

Apesar das APIs disponibilizadas pelas empresas do ramo tecnológico, referências em inteligência artificial, com o uso de reconhecimento de voz e imagem, não foram encontrados projetos na literatura e também não em repositórios de código-fonte, com a ênfase na acessibilidade nos aplicativos de mensagens para pessoas com deficiência visual e/ou auditiva.

É importante ressaltar que, até a data da publicação deste trabalho, o aplicativo de mensagem WhatsApp não provê de um algoritmo que execute os objetivos relacionados a acessibilidade apresentados anteriormente.

[12] https://github.com/grayuk/voicos

3 Metodologia

Este capítulo refere-se a descrição sobre a metodologia empenhada neste trabalho. No primeiro momento, está indicado as características de pesquisas que estão incluídas no estudo, na sequência aborda as definições das APIs externas e biblioteca de comunicação, as subáreas da inteligência artificial contidas nesse documento, o protótipo elucidado, os materiais empregados e o cronograma do planejamento.

3.1 Caracterização da Metodologia

A metodologia encontra-se orientada na fidelidade do controle das informações que serão expostas a fim de alcançar o resultado esperado, associando-os aos procedimentos, abordagens, natureza e objetivos (MARCONI; LAKATOS, 2010). A concepção desta ferramenta possui a abordagem qualitativa, tendo por objetivo em caráter exploratório, com o procedimento bibliográfico e de natureza aplicada.

3.2 Definição de Tecnologias

3.2.1 API de Reconhecimento por Voz

O levantamento bibliográfico sobre inteligência artificial aplicada para reconhecimento de voz abrangeu a análise de *softwares* de empresas referência na área, são elas: Google com o Tensorflow[1] e Google Voice Recognition, e a Mozilla em parceria com a Baidu para a desenvolvimento do DeepSpeech[2].

Os três projetos citados utilizam da técnica de *deep learning*, que é ensinar a máquina para aprender baseado na teoria de redes neurais. De forma equivalente ao sistema biológico humano, essa técnica consiste em "neurônios", responsáveis por atribuir valores para os dados de entrada. Como por exemplo, uma foto que ao passar nos "neurônios da rede" recebe valores que caracterizam aquela imagem em específico. Nesse caso, quanto maior a semelhança da foto inserida com os valores que a IA possui armazenada em memória, significa maior a probabilidade de correspondência entre os dois dados (ALPAYDIN, 2020).

O Tensorflow é uma biblioteca de código aberto, desenvolvida para criar e treinar modelos para sistemas inteligentes. Essa biblioteca possui APIs bem documentadas, opção de *debug*[3] nativo, além de suportar o uso de placa de vídeo para acelerar o processo de

[1] https://www.tensorflow.org
[2] https://deepspeech.readthedocs.io
[3] Conhecido por depuração, consiste na atividade de identificar problemas no código de uma aplicação

treinamento da inteligência artificial, e por fim, dispõe da capacidade de exibir de forma gráfica o desempenho na fase de treino da IA através do módulo TensorBoard[4]. O reconhecimento de voz por meio da ferramenta Tensorflow utiliza a técnica de *Convolutional Neural Networks* (CNN), em que a rede neural aprende os filtros baseado no agrupamento de semelhanças nos chamados "neurônios da rede" (SAINATH; PARADA, 2015). A explicação do processo de reconhecimento de voz usando o Tensorflow começa com a conversão do sinal de áudio em uma imagem. A frequência desse som é convertida em um array bidimensional (uma variável com múltiplos valores), dessa forma, pode-se gerar um espectrograma (imagem que reflete o áudio) a partir de um som. O motivo pelo qual se converte áudio em imagem, é que para uma máquina seja mais fácil analisar um gráfico em relação a um arquivo de áudio puro.

O DeepSpeech usa a base do Tensorflow, porém implementa técnicas de aprendizado não supervisionado, pois aprende diretamente com o dataset, que é o conjunto de dados inseridos para treinar o modelo de ML (HANNUN et al., 2014). O diferencial do DeepSpeech é utilizar uma rede neural recorrente (RNN), uma classe de redes neurais em que as conexões entre os neurônios formam um gráfico direcionado ao longo de uma sequência temporal (DU; SWAMY, 2013). O Google desenvolveu também sua plataforma online de reconhecimento de voz, chamado Google Voice Recognition, que permite a conversão de áudio em texto em mais de 120 idiomas. Possui suporte à tradução em tempo real e aceita arquivos de áudio ou diretamente do microfone. Também possui o recurso de "diarização de locutor" que é habilidade de identificar múltiplas pessoas conversando em um áudio, tal como, uma entrevista na rádio.

Para este trabalho, optou-se pela API de reconhecimento de voz da Google, uma vez em que, Kepuska e Bohouta (2017) realizaram através de pesquisa científica a comparação de APIs de reconhecimento de voz, e constataram que possui a API do Google possui assertividade de 91% na transcrição de mais de 600 áudios gravados.

3.2.2 API de Reconhecimento de Imagem

Em relação ao estudo das APIs de inteligência artificial direcionada para a interpretação de imagens, foram escolhidas as empresas líderes nesse mercado, são elas: Google, IBM, Microsoft e Amazon. Com a percepção de que a técnica de *deep learning* é predominante nesses sistemas e o diferencial entre estas aplicações são suas especialidades que cada uma possui.

A empresa IBM dispõe a ferramenta de uso gratuito para fins não comerciais, chamada de Watson Visual Recogniton, que faz uso da técnica de Deep Learning para ser capaz de reconhecer cenas, objetos, alimentos. Expõe a função extra da criação

[4] https://www.tensorflow.org/tensorboard

de um classificador com as configurações específicas para as necessidades da empresa (COLLINASZY; BUNDZEL; ZOLOTOVA, 2017).

No sistema Vision AI[5], construído pela Google, é uma ferramenta destinada ao reconhecimento de imagens de uso comercial, por meio deste é possível identificar produtos em fotos e realizar uma busca automática por preços e disponibilidade do mesmo, interpretação de textos manuscritos ou impressos, além da possibilidade da análise e remoção automatizada de conteúdos inapropriados.

Enquanto a Amazon oferece o serviço comercial Amazon Rekognition[6], que faz uso da técnica de *machine learning*, é especializado na análise facial com capacidade de detectar, analisar e comparar rostos para a validação da identidade do usuário e possui a habilidade de segmentar vídeos para buscar por imagens específicas (INDLA, 2021).

De forma semelhante, a Azure com o sistema Computational Visual Search[7] é capaz de extrair textos de imagem e de analisar um ambiente para detectar as pessoas em um espaço físico, por exemplo, para controlar o distanciamento social. Além do processamento de vídeos da câmeras de segurança para rastrear o caminho percorrido por clientes no interior da empresa, analise do tempo de fila e tempo de permanência em frente a determinadas vitrine, e até mesmo detectar o uso da máscara facial para previr a proliferação de doenças transmissíveis pelo ar.

A API selecionada para o processo de interpretação das imagens sucedeu-se da IBM, que segundo os pesquisadores Nilsson e Jonsson (2019), realizaram um estudo comparativo entre as APIs para esse fim, o resultado foi que a API da IBM teve assertividade aproximadamente de 65% para reconhecer animais e em média geral com outras categorias de imagem, 42% de acerto. Nos testes com mais de 15.000 imagens com diversos objetos e cenários, a porcentagem de acerto é considerada alta visto que a média geral com todas as outras concorrentes é de 42%. A quantidade de horas de treinamento não se aplica para as APIs de interpretação de imagens e áudios, visto que estes produtos são para o uso final e não com intuito no processo de treinamento de IA.

3.2.3 Biblioteca de Integração com WhatsApp

No quesito de comunicação para com o WhatsApp, é necessário utilizar uma biblioteca a fim de que o *bot* construído possa realizar a integração entre as APIs de inteligência artificial e o aplicativo de mensagem. Ao decorrer da investigação nos repositórios de projetos *open-source* existentes, constatou-se três opções apropriadas a atingir o objetivo:

[5] https://cloud.google.com/vision
[6] https://aws.amazon.com/pt/rekognition
[7] https://azure.microsoft.com/pt-br/services/cognitive-services/computer-vision

Baileys[8], *wa-automate*[9] e *whatsmeow*[10].

A primeira opção, o *Baileys*, é um *websocket* elaborado em TypeScript[11] que disponibiliza uma API com o intuito de apreciar suas funcionalidades. Além de permitir economizar na quantidade do processamento na memória, oferece suporte a múltiplos dispositivos e as versões do WhatsApp Web. Porém, se apresenta oscilante após um período de funcionamento.

Em contrapartida, a biblioteca *wa-automate* é produzida com NodeJS[12] e possibilita, além da API, uma versão estruturada para o seu uso com Docker[13]. Entretanto, utiliza-se de uma ferramenta baseada em navegador chamada Selenium[14] em que manifesta um desempenho inferior e impossibilitando escalonar com múltiplos *bots*.

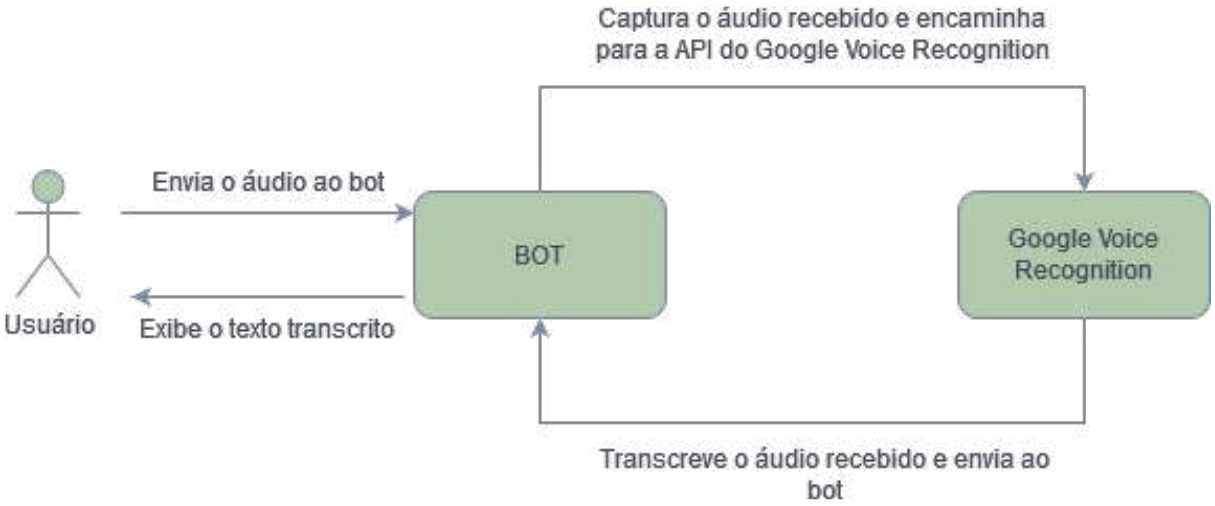
Por fim, o *whatsmeow* é uma ferramenta desenvolvida em Go aplicando a tecnologia de *websocket*, viabiliza a execução de forma leve, apresentando uma documentação eficiente e demonstrando ser extremamente estável e consistente. Por consequência, se deu a escolha dessa biblioteca em prol do propósito.

3.3 Transcrição de Fala

Com a finalidade do reconhecimento de voz, foi selecionado o *Google Voice Recognition*[15], devido a sua tecnologia que soluciona os principais problemas de um sistema de reconhecimento de voz principiante tais como a baixa precisão em frases com sotaque, a dificuldade na interpretação de áudio em ambientes acompanhada de elevada reverberação do som, cenários em que as palavras possuem fonética semelhantes, diferença de velocidade e entonação de voz, capacidade de pontuar frases e filtrar palavras inapropriadas (FORSBERG, 2003).

[8] https://github.com/adiwajshing/Baileys
[9] https://github.com/open-wa/wa-automate-nodejs
[10] https://github.com/tulir/whatsmeow
[11] https://www.typescriptlang.org
[12] https://nodejs.org
[13] https://www.docker.com
[14] https://www.selenium.dev
[15] https://cloud.google.com/speech-to-text

Figura 1 – Arquitetura da Transcrição de Fala



Fonte: Autoria Própria (2022)

A Figura 1 representa a arquitetura da transcrição de voz através da ferramenta proposta neste trabalho. O início é a partir da voz captada quando o usuário envia um áudio para uma conversa com o *bot*. Em seguida, a API de reconhecimento de fala do Google é requisitada, que por sua vez, retorna o texto transcrito para o *bot* que encaminha ao usuário.

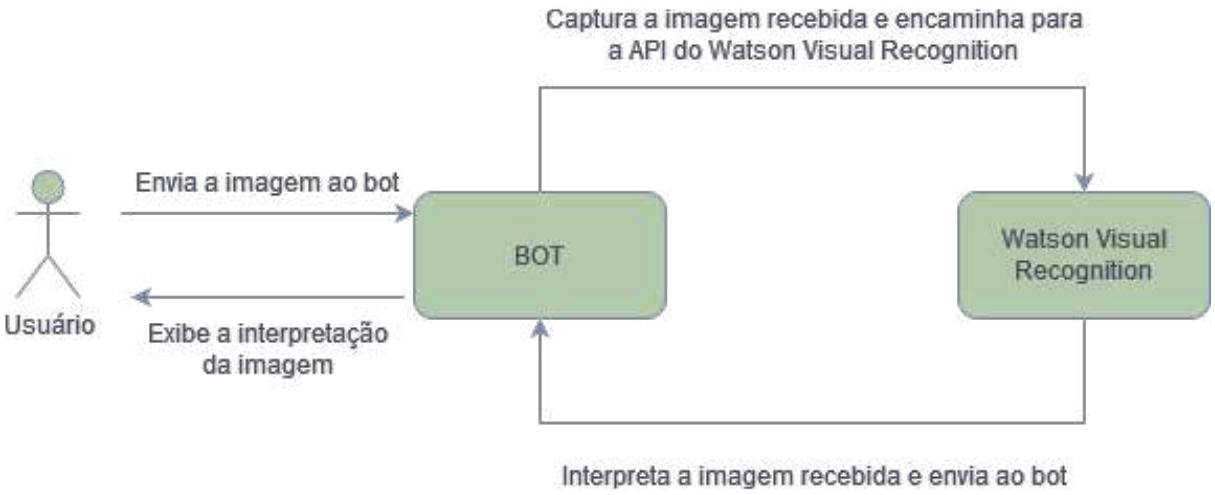
3.4 Reconhecimento de Imagem

O mecanismo de reconhecimento de imagens resultou pela API da IBM denominada de Watson Visual Recognition[16]. A preferência por esta aplicação se deu em razão dela permitir maior flexibilidade ao possibilitar a criação de um sistema próprio de classificação de imagens (PESCE et al., 2022) e ser gratuita para uso não comercial, essa restrição que proíbe o uso comercial não é um empecilho, visto que este trabalho não será comercializado por ser filantrópico.

A seguinte Figura demonstra o processo para a interpretação de uma imagem pela ferramenta. Inicia-se no momento em que o usuário envia uma imagem para uma conversa com o *bot* pelo aplicativo de mensagem. Logo após, é consumida a API de reconhecimento de imagem da IBM, responsável por interpretar o conteúdo existente.

[16] https://www.ibm.com/br-pt/cloud/watson-visual-recognition

Figura 2 – Arquitetura do Reconhecimento de Imagem



Fonte: Autoria Própria (2022)

3.5 Prototipação

Na Figura 3 está representado o protótipo baseado no comportamento relacionado a funcionalidade de interpretação de imagens, no qual o usuário encaminha o arquivo a ser examinado e a ferramenta retorna com a caracterização das propriedades envolvidas na imagem solicitada.

Figura 3 – Protótipo da Ferramenta

Fonte: Autoria Própria (2022)

3.6 Materiais

O projeto está sendo construído nas dependências da própria universidade, utilizando-se do laboratório de informática para elaboração e discussões sobre a ferramenta. Com o intuito do desenvolvimento, estão sendo utilizados as seguintes configurações de *hardware* e *software*, conforme detalhados na Tabela 1:

Tabela 1 – Hardware e Software

DESCRIÇÃO	FABRICANTE	VERSÃO/MODELO
Notebook	HP	EliteBook 840 G5
SO Windows	Microsoft	Windows 10 Pro 64x
Processador	Intel	Core i7-7500U 2.90 GHz
Memória RAM	Corsair	8 Gb DDR4
SSD	WD Green	512 Gb, m.2
Visual Studio Code	Microsoft	1.62.2

Fonte: Autoria Própria (2022)

3.7 Cronograma

Para alcançar os objetivos e metas deste projeto, o seguinte cronograma foi adotado:

Tabela 2 – Cronograma das Atividades

MÊS DE REFERÊNCIA	AÇÃO DESEMPENHADA
Janeiro/22	Definição do problema a ser resolvido
Fevereiro/22	Levantamento e estudo dos referenciais teóricos
Março/22	Estudo das APIs disponíveis para o reconhecimento de voz utilizando inteligência artificial
Abril/22	Estudo das APIs disponíveis para o reconhecimento de imagens utilizando inteligência artificial
Maio/22	Escrita do projeto de conclusão de curso
Junho/22	Apresentação do projeto de conclusão de curso
Julho/22	Correções sugeridas pela banca examinadora
Agosto/22	Implementação da funcionalidade de transcrição de voz na ferramenta
Setembro/22	Implementação da funcionalidade de reconhecimento de imagens na ferramenta
Outubro/22	Desenvolvimento da conexão entre a ferramenta e os aplicativos de mensagens
Novembro/22	Atualização na escrita do trabalho de conclusão de curso
Dezembro/22	Apresentação do trabalho de conclusão de curso

Fonte: Autoria Própria (2022)

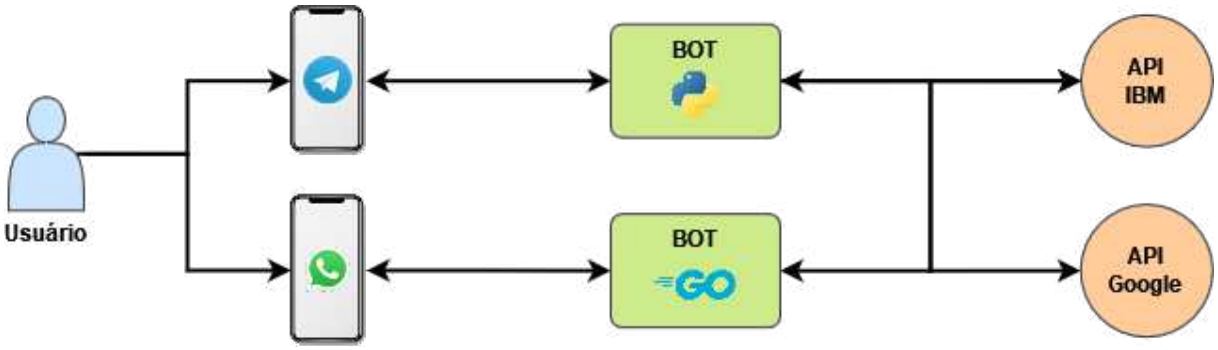
4 Resultados

Neste capítulo encontram-se expressados os resultados obtidos durante a confecção deste trabalho, como a construção da arquitetura do algoritmo, elaboração dos diagramas, implementação da solução nos aplicativos de mensagens e análises provenientes da execução dos testes desempenhados.

4.1 Arquitetura do Algoritmo

Nessa divisão, em seguida a etapa do estudo referente a definição das tecnologias que foram empregues neste trabalho, consolidou-se na elaboradação da seguinte arquitetura, apresentado na Figura 4 abaixo, no qual é expressado a integração entre os componentes presentes.

Figura 4 – Arquitetura do Trabalho



Fonte: Autoria Própria (2022)

No primeiro momento, o usuário deve encaminhar o arquivo, seja imagem ou áudio, em direção ao *bot* através do aplicativo de mensagem escolhido, podendo optar entre o WhatsApp ou Telegram. Por sua vez, o algoritmo recebe o arquivo enviado, realiza o processamento para identificar o formato e endereça a API designada, da IBM para interpretação da imagem ou do Google para transcrição do áudio. Por fim, após recuperar as informações requisitadas nas aplicações externas, a ferramenta efetua tratamentos com a intenção de melhorar a legibilidade dos dados e devolve para o usuário.

4.2 Diagramas

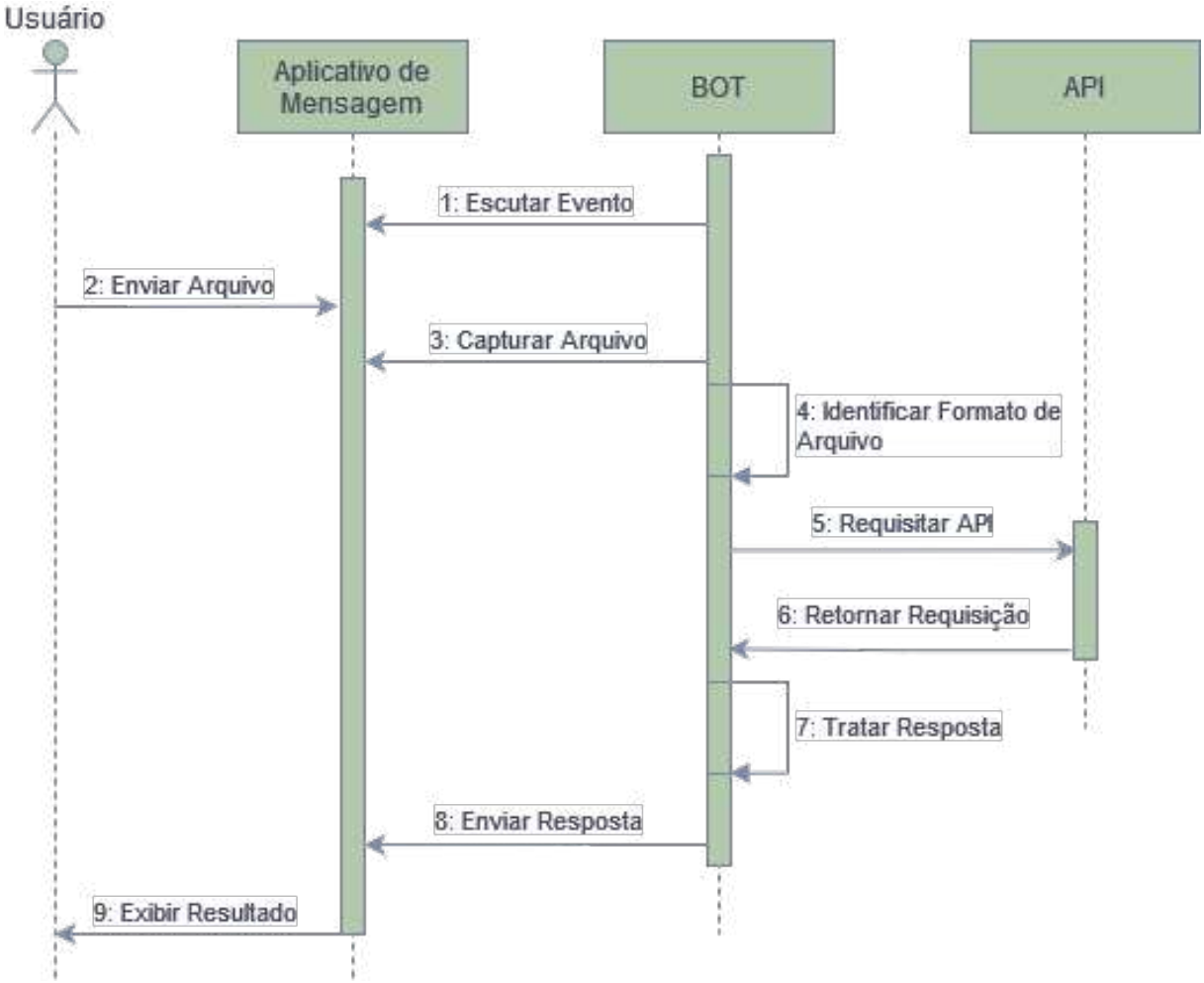
A existência dos diagramas num processo de desenvolvimento de software é de suma importância devido a sua finalidade de propiciar múltiplas perspectivas da aplicação

a ser construída, desta forma, concedendo aproximar-se da dificuldade da modelagem (GUEDES, 2009). Para tal, se deu a elaboração dos diagramas de sequência e atividade com o intuito de auxiliar na produção e documentação desta ferramenta.

4.2.1 Diagrama de Sequência

Na Figura 5 abaixo apresenta o diagrama de sequência modelado para esta aplicação. Nele, está contido a ordenação temporal dos eventos bem como as trocas entre os objetos que sucedem para o funcionamento dos objetivos propostos neste trabalho.

Figura 5 – Diagrama de Sequência

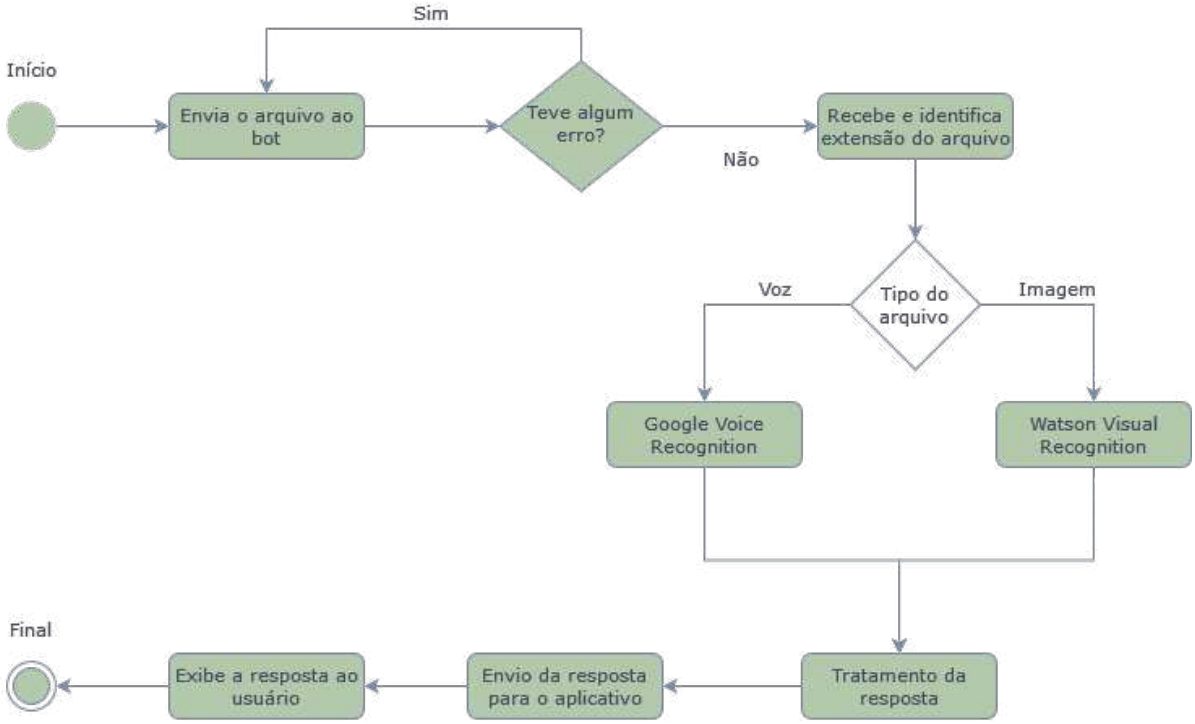


Fonte: Autoria Própria (2022)

4.2.2 Diagrama de Atividade

O diagrama de atividade descreve as etapas a serem reproduzidas para o término da execução baseado no fluxo de controle. Na Figura 6 logo abaixo, é exibida as fases para construção do algoritmo.

Figura 6 – Diagrama de Atividade



Fonte: Autoria Própria (2022)

4.3 Desenvolvimento

4.3.1 Telegram Bot

Torna-se relevante enfatizar que, pelo fato do Telegram ser de código aberto, o mesmo disponibiliza de API a fim de que facilita a formação de *bots*, logo não necessitando de uma biblioteca mediadora para performar tal função. Portanto, a seguir estão contidos os esclarecimentos relacionados a etapa de desenvolvimento deste aplicativo de mensagens.

Com o intuito de usar a API oferecida, é requerido o emprego de um token próprio a execução das requisições, no qual é possível obter através do BotFather[1], *bot* ofertado pelo Telegram com a finalidade de criar e gerenciar novos *bots*. Para acessá-lo, é exigido que o desenvolvedor tenha uma conta previamente cadastrada no aplicativo. Durante esta etapa, é solicitado o nome e o codinome (*username*) do *bot*, e ao final do processo, o token é exibido e está apto para iniciar sua atividade.

Baseando no diagrama de sequência elaborado na Figura 5, as etapas primárias para a construção do código são de escutar os eventos a fim de que o usuário possa enviar o arquivo requerido. Portanto, realizou-se a inicialização da instância do *bot* no qual será utilizado por todo o algoritmo desenvolvido, através do token gerado pelo *BotFather* (omitido por questões de segurança), como é demostrado no Algoritmo 1.

[1] https://t.me/botfather

```
bot_token = '<TOKEN_GERADO_PELO_BOTFATHER>'
bot = telegram.Bot(bot_token)
update_id = bot.get_updates()[0].update_id
for update in bot.get_updates(offset=update_id, timeout=10):
    update.message
```

Algoritmo 1 – Inicialização do Bot (Telegram)

Caso o arquivo enviado seja no formato de áudio, o *bot* irá efetuar duas requisições: identificar o caminho de armazenamento e realizar o *download* do mesmo. Na sequência, sofre uma conversão no formato de .oga para .wav e, posteriormente, encaminhado para a API do Google no qual é retornado a transcrição e texto sendo direcionado ao usuário. No Algoritmo 2 está relatando os processos citados.

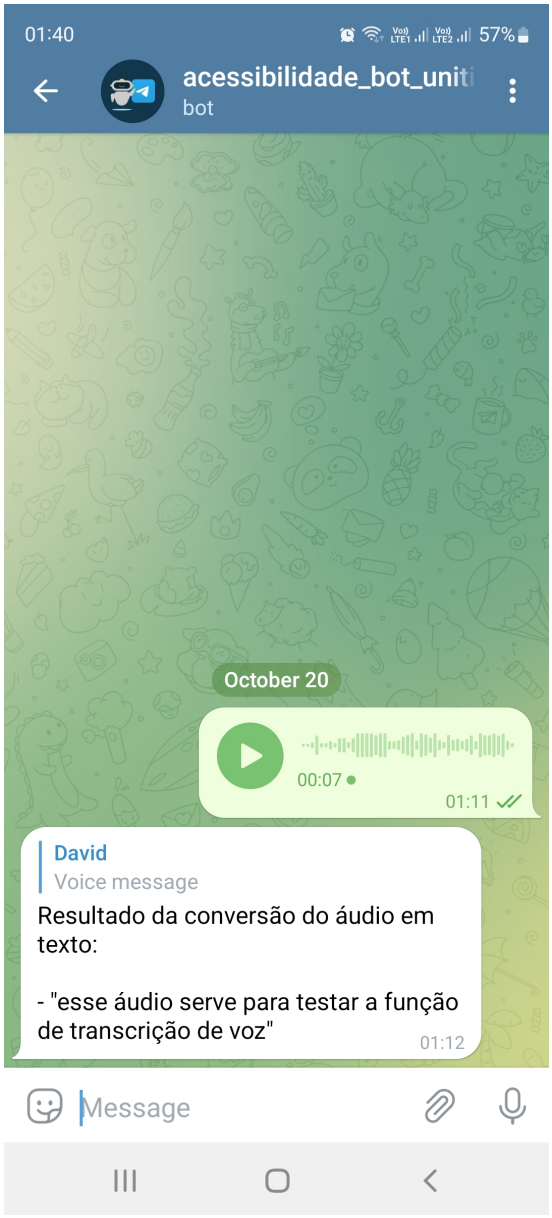
```
def regex_string(variavel, inicio, fim):
  frase_array_primario = variavel.split(inicio)
  frase_array_secundario = frase_array_primario[1].split(fim)
  resultado = frase_array_secundario[0]
  return resultado

def reconhecer_voz(audio_baixado_path):
  sound = AudioSegment.from_file(audio_baixado_path)
  sound.export("audio-convertido.wav", format="wav")
  r = sr.Recognizer()
  with sr.AudioFile('audio-convertido.wav') as source:
    audio = r.record(source)
  text_from_audio = r.recognize_google(audio, language='pt-BR')
  return text_from_audio

def baixar_audio(id_arquivo):
  requisicao_1 = requests.get('https://api.telegram.org/bot' + bot_token
    + '/getFile?file_id=' + id_arquivo)
  caminho_arquivo = regex_string(requisicao_1.text, '"file_path":"', '"'
   )
  requisicao_2 = requests.get('https://api.telegram.org/file/bot' +
   bot_token + '/' + caminho_arquivo)
  open('audio-capturado.oga', 'wb').write(requisicao_2.content)
  return reconhecer_voz('audio-capturado.oga')

if ", 'voice': {'" in str(update.message):
  id_arquivo = regex_string(str(update.message), "'file_id': '", "',")
  voz_transcrita = baixar_audio(id_arquivo)
  voz_transcrita = voz_transcrita.lower()
  update.message.reply_text("transcricao do audio: " + voz_transcrita)
```

Algoritmo 2 – Transcrição de Fala (Telegram)

A Figura 7 a seguir, retrata a tela referente ao *bot* no Telegram com a funcionalidade

de transcrever um áudio, por meio da API externa, a partir do envio de um arquivo pelo usuário para o aplicativo de mensagem.

Figura 7 – Tela de Transcrição do Áudio no Telegram



Fonte: Autoria Própria (2022)

Na condição do arquivo ser no formato de imagem, de natureza igual a situação anterior, o *bot* averigua o local onde está contido para que seja executado o descarregamento do mesmo. Após este procedimento, a imagem é endereçada a API da IBM tendo a finalidade de interpretar o arquivo solicitado e a informação regressa a origem na direção de ser repassada ao usuário do aplicativo. Esta API solicita um *token* de acesso, portanto se fez necessário o cadastro para utilização deste recurso a fim de que o algoritmo estivesse qualificado a esta funcionalidade. O detalhamento deste percurso encontra-se no Algoritmo 3 logo abaixo.

```
def baixar_foto(id_arquivo):
  requisicao_1 = requests.get('https://api.telegram.org/bot' + bot_token
    + '/getFile?file_id=' + id_arquivo)
  caminho_arquivo = regex_string(requisicao_1.text, '"file_path":"', '"'
   )
  full_caminho_arquivo = 'https://api.telegram.org/file/bot' + bot_token
    + '/' + caminho_arquivo

  headers = {'Accept-Language': 'pt-br'}
  requisicao_2 = requests.get('https://api.us-south.visual-recognition.
   watson.cloud.ibm.com/v3/classify?url=' + full_caminho_arquivo + '&
   version=2018-03-19', auth=('apikey', '<TOKEN_ACESSO_IBM>'), headers=
   headers)
  requisicao_2_json = requisicao_2.json()

  output = ""
  for x in range(len(requisicao_2_json['images'][0]['classifiers'][0]['
   classes'])):
    classe = requisicao_2_json['images'][0]['classifiers'][0]['classes'
   ][x]['class']
    precisao = requisicao_2_json['images'][0]['classifiers'][0]['classes
   '][x]['score']
    output += classe + ' (precisao de ' + str(precisao * 100) + '%), '

  return str(output.encode('utf-8'))

if "}, 'photo': [{'" in str(update.message):
  id_arquivo = update.message["photo"][2]["file_id"]
  predicao = baixar_foto(id_arquivo)
  update.message.reply_text("interpretacao da imagem: " + predicao)
```

Algoritmo 3 – Interpretação da Imagem (Telegram)

Equitativamente a função anterior, na Figura 8 exibe a tela no que se refere ao algoritmo executando o método de interpretação da imagem no Telegram, apresentando os conteúdos inseridos no arquivo encaminhado.

Figura 8 – Tela de Interpretação da Imagem no Telegram

Fonte: Autoria Própria (2022)

4.3.2 WhatsApp Bot

No contexto do desenvolvimento para o aplicativo de mensagens WhatsApp, se deu através da utilização da biblioteca *whatsmeow*, havendo as suas considerações anteriormente evidenciadas no capítulo relacionado a metodologia deste trabalho. Deste modo, na sequência desta seção, se encontram os estágios da construção do algoritmo na linguagem Go.

Semelhantemente a produção relacionada ao Telegram, também é fundamentado no diagrama de sequência exibido na Figura 5. Inicialmente, se fez necessário estruturar o vínculo do *bot* pela biblioteca supracitada com o aplicativo por intermédio do acesso via

QRCode[2]. No Algoritmo 4 logo abaixo, é exibido este processo composto pela configuração dos *logs*, os ajustes de sincronização entre os dispositivos conectados, a concepção do código para leitura e, por fim, a conexão com o WhatsApp.

```
func main() {
  waBinary.IndentXML = true
  if *debugLogs {
    logLevel = "DEBUG"
  }
  if *requestFullSync {
    store.DeviceProps.RequireFullSync = proto.Bool(true)
  }
  log = waLog.Stdout("Main", logLevel, true)
  dbLog := waLog.Stdout("Database", logLevel, true)
  storeContainer, err := sqlstore.New(*dbDialect, *dbAddress, dbLog)
  device, err := storeContainer.GetFirstDevice()
  cli = whatsmeow.NewClient(device, waLog.Stdout("Client", logLevel,
   true))
  ch, err := cli.GetQRChannel(context.Background())
  if err != nil {
    if !errors.Is(err, whatsmeow.ErrQRStoreContainsID) {
      log.Errorf("Failed to get QR channel: %v", err)
    }
  } else {
    go func() {
      for evt := range ch {
        if evt.Event == "code" {
            qrterminal.GenerateHalfBlock(evt.Code, qrterminal.L, os.
   Stdout)
        } else {
          log.Infof("Resultado do QR code: %s", evt.Event)
          }
    }()
  }
  cli.AddEventHandler(handler)
  err = cli.Connect()
}
```

Algoritmo 4 – Instância da Conexão (WhatsApp)

Após a biblioteca se conectar com o aplicativo de mensagens, o algoritmo se torna capaz de executar as funcionalidades definidas. No Algoritmo 5 demonstra as fases dos processamentos relacionadas a transcrição de áudio no WhatsApp, partindo da escuta dos eventos de mensagens recebidas, *download* e escrita do arquivo localmente, conversão no

[2] O *Quick Response Code* é um código de barras na versão bidimensional com a competência de disponibilizar informações

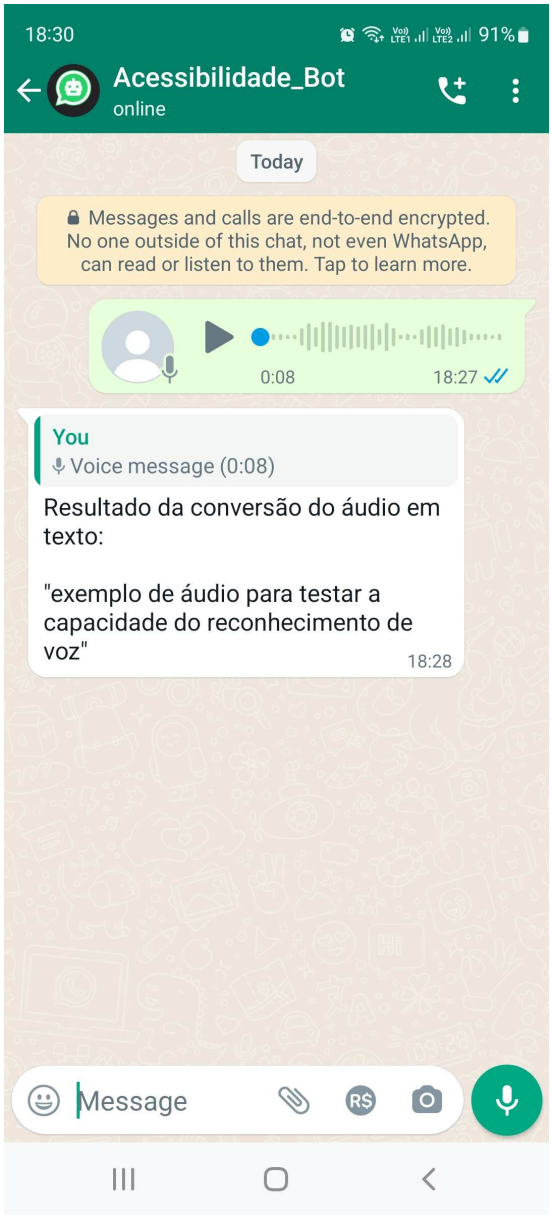
formato base64, requisição para API do Google e responder o usuário com a informação obtida.

```
case *events.Message:
audio := evt.Message.GetAudioMessage()
if audio != nil {
  data, err := cli.Download(audio)
  exts, _ := mime.ExtensionsByType(audio.GetMimetype())
  path := fmt.Sprintf("%s%s", evt.Info.ID, exts[0])
  err = os.WriteFile(path, data, 0600)
  log.Infof("Audio salvo em %s", path)
  idioma := "pt-BR"
  audioBase64, err := exec.Command("base64", path).Output()
  cmd, err := exec.Command("curl",
    "https://speech.googleapis.com/v1/speech:recognize",
    "--header",
    "Authorization: <TOKEN_OCULTADO>",
    "--header",
    "Content-Type: application/json",
    "--data",
    "{\"config\":{\"encoding\":\"OGG_OPUS\",\"sampleRateHertz\":16000,\"
   languageCode\":\"" + idioma + "\"},\"audio\":{\"content\":\"" +
   string(audioBase64) +"\"}}").Output()
  fmt.Println("resultado do curl: ", string(cmd))
  recipient, ok := parseJID(evt.Info.SourceString())
  msg := &waProto.Message{Conversation: proto.String(string(cmd))}
  resp, err := cli.SendMessage(context.Background(), recipient, "", msg)
  if err != nil {
    log.Errorf("Erro ao enviar mensagem: %v", err)
  } else {
    log.Infof("Mensagem enviada (timestamp do servidor: %s)", resp.
   Timestamp)
  }
}
```

Algoritmo 5 – Transcrição do Áudio (WhatsApp)

Na sequência, a Figura 9 expõe a atividade especificada acima no cenário fidedigno, no qual o usuário envia o áudio em direção ao *bot* que, por sua vez, realiza os procedimentos necessários e responde o remetente com a conversão do arquivo no formato de texto.

Figura 9 – Tela de Transcrição do Áudio no WhatsApp



Fonte: Autoria Própria (2022)

Relacionada a interpretação da imagem no WhatsApp, o Algoritmo 6 baixa o arquivo e identifica a extensão do mesmo. Por conseguinte, efetua o pedido com destino a API da IBM e, após a informação ser devolvida, retorna para o usuário com os elementos presentes.

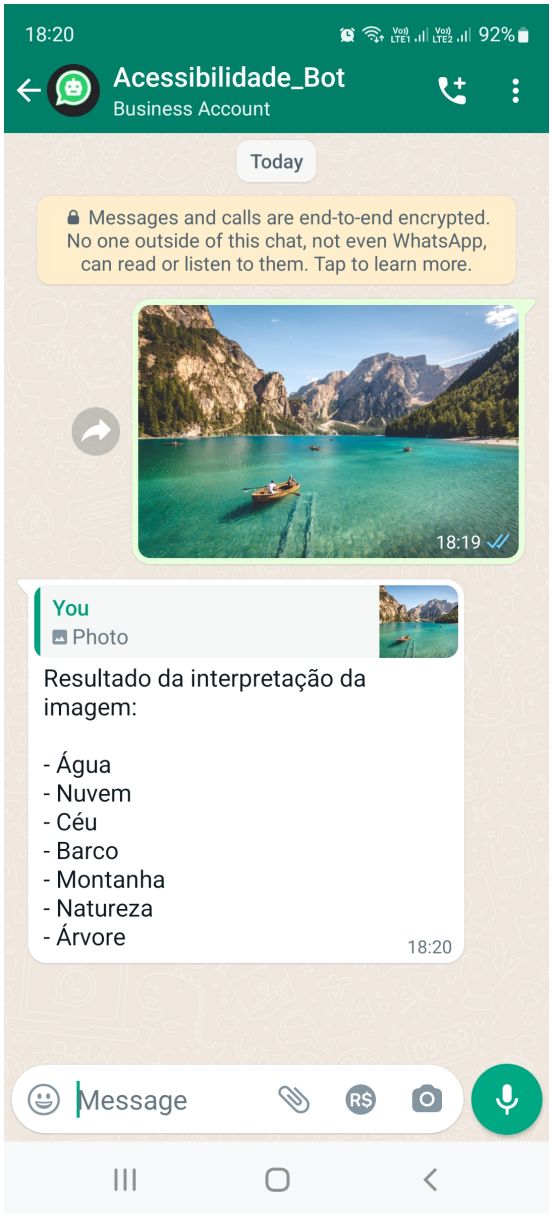
```
case *events.Message:
img := evt.Message.GetImageMessage()
if img != nil {
  data, err := cli.Download(img)
  exts, _ := mime.ExtensionsByType(img.GetMimetype())
  path := fmt.Sprintf("%s%s", evt.Info.ID, exts[0])
  err = os.WriteFile(path, data, 0600)
```

```
  log.Infof("Imagem salva em %s", path)
  cmd, err := exec.Command("curl",
    "https://api.us-south.visual-recognition.watson.cloud.ibm.com/v3/
   classify&version=2018-03-19",
    "--header",
    "Accept-Language: pt-br",
    "--user",
    "apikey:<TOKEN_OCULTADO>",
    "--form",
    "imagesFile=@" + path).Output()
  fmt.Println("resultado do curl: ", string(cmd))
  recipient, ok := parseJID(evt.Info.SourceString())
  msg := &waProto.Message{Conversation: proto.String(string(cmd))}
  resp, err := cli.SendMessage(context.Background(), recipient, "", msg)
  if err != nil {
    log.Errorf("Erro ao enviar mensagem: %v", err)
  } else {
    log.Infof("Mensagem enviada (timestamp do servidor: %s)", resp.
   Timestamp)
  }
}
```

Algoritmo 6 – Interpretação da Imagem (WhatsApp)

Em prol de testificar a prática mencionada anteriormente, a Figura 10 revela o funcionamento de interpretação da imagem no aplicativo de mensagens WhatsApp depois que o usuário solicita ao *bot* que o atende enviando as informações colhidas com os métodos executados.

Figura 10 – Tela de Interpretação da Imagem no WhatsApp



Fonte: Autoria Própria (2022)

4.3.3 Testes

Com a finalidade de validação do objetivo geral apontado neste trabalho, no quesito relacionado ao desempenho do algoritmo em harmonia com os aplicativos de mensagem, se deu a produção de testes de performance. De acordo com Pressman (2011), o teste de perfomance é arquitetado com o propósito de testar o desempenho no período de atividade da aplicação inserido na condição de sistema.

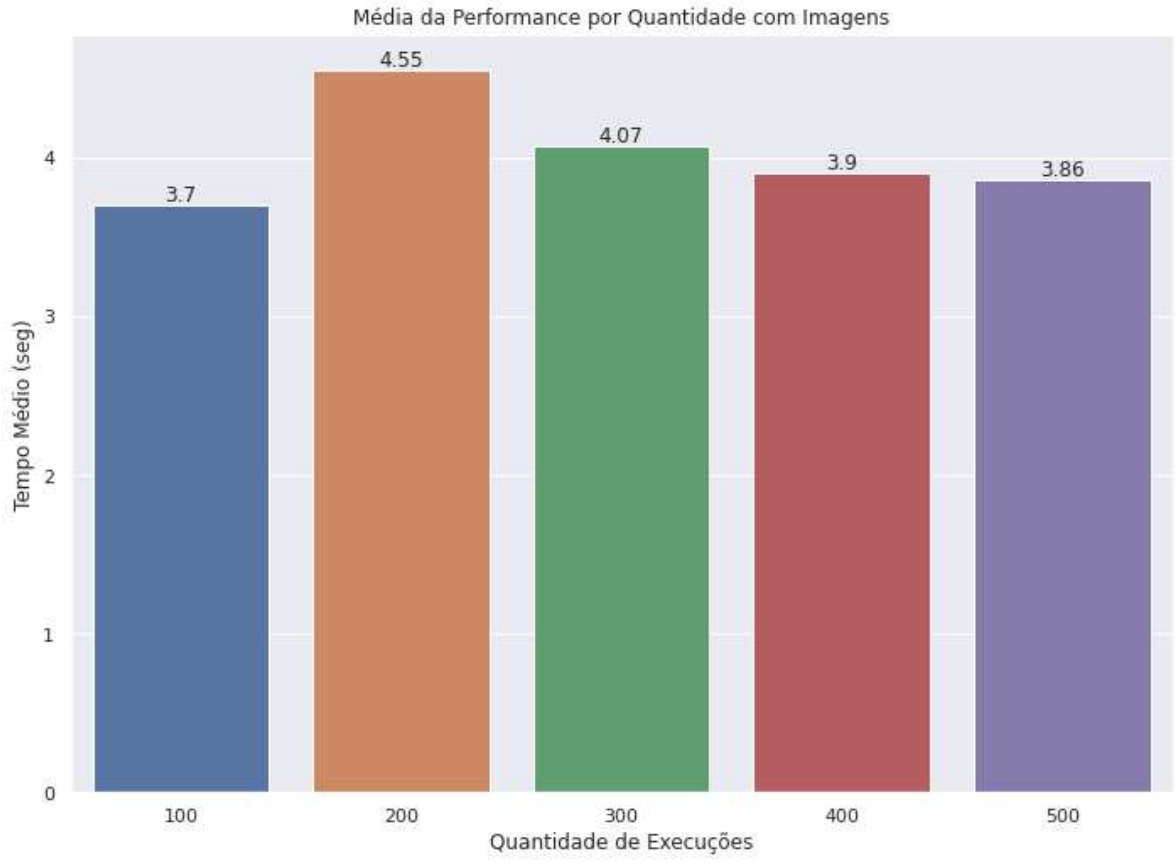
Nesse contexto, os testes de desempenho destinaram-se em duas categorias baseadas nos formatos dos arquivos em que engloba o algoritmo, imagem e áudio, e realizados através da mesma biblioteca utilizada no *bot* para o WhatsApp, o *whatsmeow*. Foram executados

no servidor contendo o sistema operacional Ubuntu 22.04 e com 8GB de memória no processamento.

Em ambas as categorias, relacionadas aos formatos dos arquivos, se deu a divisão dos testes em cinco etapas, no qual iniciou-se com cem requisições, incrementando com mais uma centena de solicitações em cada término de fase até alcançar o número de quinhetas requisições.

O teste de desempenho focado em imagens ocorreu-se da mensuração desde a remessa do arquivo enviado pelo usuário, processamento pelo *bot* desenvolvido, requisição e retorno da API da IBM, tratamento dos dados e devolução ao aplicativo. Na Figura 11 é possível observar que a duração dessa execução oscilou entre 3.70 e 4.55 segundos, mantendo uma média de 4.01 segundos.

Figura 11 – Gráfico do Teste de Performance - Imagem

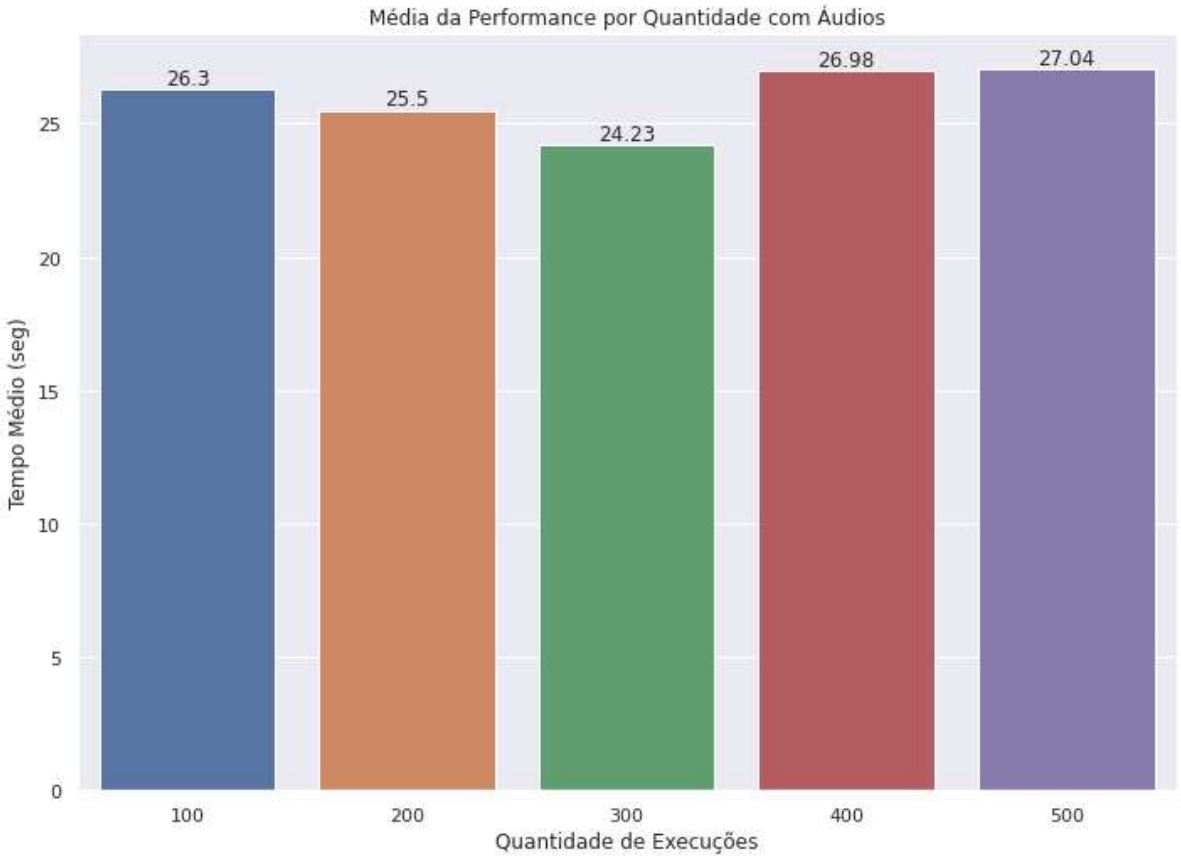


Fonte: Autoria Própria (2022)

Já o teste de performance direcionado aos áudios, possuiu como embasamento a metodologia desenvolvida na avaliação executada com imagens, entretanto alterou-se a requisição externa para API do Google, com o objetivo de transcrever os arquivos. Na Figura 12 verificou-se um intervalo de 24.23 a 27.04 segundos, com uma média de 26.01 segundos dessas ações. Além disso, notou-se um pequeno acréscimo no tempo médio

a partir do estágio de 400 requisições, devido a restrição em que o WhatsApp possui quando ultrapassa 1000 mensagens e as organiza como uma fila, porém a API externa e o processamento do algoritmo permaneceram-se estáveis.

Figura 12 – Gráfico do Teste de Performance - Áudio



Fonte: Autoria Própria (2022)

5 Conclusão

Tendo o propósito de integrar as pessoas com deficiência visual e/ou auditiva na era digital, o objetivo determinante deste trabalho se deu na implementação de uma solução para auxiliar e capacitar estes indivíduos. Como resultado, algoritmos relacionados a interpretação de imagens e transcrição de áudios nas principais plataformas de mensagens foram propostos.

Construídos direcionados aos aplicativos Telegram, desenvolvido em Python por meio da própria interface pública fornecida para efetuar a conexão, e WhatsApp, confeccionado na linguagem Go com emprego da biblioteca *whatsmeow* como ligação. Além disso, APIs de identificação das empresas IBM, referente as imagens, e Google, associada aos áudios, sucederam o uso nos *bots*.

Com a finalidade de validar a performance do artefato desenvolvido, se deu a construção de testes de carga por meio da quantidade de requisições enviadas ao algoritmo relacionada a duração dos processamentos existentes incluindo a devolução da resposta ao usuário. Estes testes foram segmentados em duas classes, conforme os formatos suportados sendo eles imagem e áudio, e constatou um tempo médio de 4.01 e 26.01 segundos, respectivamente, no qual demonstrou que o *bot* manteve seu processamento constante.

Perante aos resultados alcançados, aliados aos testes de performance desenvolvidos, pode-se concluir que a ferramenta apresentou de modo eficiente o seu propósito, beneficiando a inclusão destas pessoas atráves da acessibilidade na tecnologia, e cumprindo com a problemática exposta nos objetivos neste trabalho.

5.1 Trabalhos Futuros

Em seguimento, estão descritas algumas proposições relacionadas a trabalhos futuros.

- **Cache dos Arquivos Recebidos:** implementar um banco em memória, conhecido como cache, em que os arquivos (imagens e áudios) convertidos para o formato base64 e os retornos das requisições relacionadas a interpretação e transcrição seriam armazenados com o objetivo de diminuir a quantidade das solicitações externas e o tempo de resposta ao usuário.
- **Algoritmo de Recuperação da Qualidade da Imagem:** produzir um estudo relacionado a qualidade da imagem enviada pelo usuário, em que o emprego do algoritmo de redes neurais pode-se recuperar a resolução do arquivo a ser interpretado.

- **Acelerar o processamento de áudios longos:** dividir áudios longos e atribuí-los em threads diferentes, a fim de diminuir o tempo necessário para realizar a transcrição.
- **Transcrever áudio de vídeo:** extrair áudio de vídeo para que seja possível a transcrição, com a finalidade de possibilitar que pessoas com deficiência auditiva possam discutir sobre um conteúdo falado em um vídeo.

Referências

ALPAYDIN, E. *Introduction to machine learning*. Cambridge, Massachusetts: MIT Press, 2020.

BARYANNIS, G. et al. Supply chain risk management and artificial intelligence: state of the art and future research directions. *International Journal of Production Research*, Taylor Francis, v. 57, n. 7, p. 2179–2202, 2019.

BRASIL. Lei nº 13.146. *Diário Oficial da República Federativa do Brasil*, Brasília, 2015.

BRESOLIN, A. d. A. *Reconhecimento de voz através de unidades menores do que a palavra, utilizando Wavelet Packet e SVM, em uma nova estrutura hierárquica de decisão*. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, Natal, 2008.

CAMPELO, R. A. et al. Inclusão digital de deficientes visuais: o uso da tecnologia assistiva em redes sociais online e celulares. *Anais do Computer on the Beach*, p. 109–118, 2011.

CHEN, X.-W.; LIN, X. Big data deep learning: challenges and perspectives. *IEEE access*, Ieee, v. 2, p. 514–525, 2014.

COLLINASZY, J. et al. Implementation of intelligent software using ibm watson and bluemix. *Acta Electrotechnica et Informatica*, v. 17, n. 1, p. 58–63, 2017.

COUTINHO, G. L. *A era dos smartphones: Um estudo exploratório sobre o uso dos smartphones no Brasil*. 67 p. Monografia (Graduação) — Faculdade de Comunicação, Universidade de Brasília, Brasília, 2014.

COUTO, L. N. *Sistema para localização robótica de veículos autônomos baseado em visão computacional por pontos de referência*. Tese (Doutorado) — Universidade de São Paulo, 2012.

DICK, S. Artificial Intelligence. *Harvard Data Science Review*, v. 1, n. 1, jul 1 2019.

DU, K.-L.; SWAMY, M. N. *Neural networks and statistical learning*. Londres: Springer Science & Business Media, 2013.

DU, M. et al. Supervised training and contextually guided salient object detection. *Digital Signal Processing*, v. 63, p. 44–55, 2017.

FARAMARZI, S. et al. Telegram: An instant messaging application to assist distance language learning. *Teaching English with Technology*, IATEFL Poland Computer Special Interest Group & University of Nicosia, v. 19, n. 1, p. 132–147, 2019.

FORSBERG, M. Why is speech recognition difficult. *Chalmers University of Technology*, Citeseer, 2003.

GOMES, D. d. S. Inteligência artificial: conceitos e aplicações. *Olhar Científico. v1*, n. 2, p. 234–246, 2010.

GUEDES, G. T. *UML 2: Uma Abordagem Prática*. São Paulo: Novatec, 2009.

HANNUN, A. et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

HAYDARI, A.; YILMAZ, Y. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, 2020.

IBGE, I. B. de Geografia e E. *Censo demográfico 2010. Características gerais da população, religião e pessoas com deficiência*. Rio de Janeiro: IBGE, 2010.

INDLA, R. K. *An overview on amazon rekognition technology*. Dissertação (Mestrado) — California State University, San Bernardino, 2021.

KËPUSKA, V.; BOHOUTA, G. Comparing speech recognition systems (microsoft api, google api and cmu sphinx). *Int. J. Eng. Res. Appl*, v. 7, n. 03, p. 20–24, 2017.

MARCONI, M. d. A.; LAKATOS, E. Fundamentos da metodologia cientifica. 7ª edição-são paulo: Atlas. 2010.

MILANO, D. d.; HONORATO, L. B. Visão computacional. UNICAMP - Universidade Estadual de Campinas, 2014.

NG, G. W.; LEUNG, W. C. Strong artificial intelligence and consciousness. *Journal of Artificial Intelligence and Consciousness*, World Scientific, v. 7, n. 01, p. 63–72, 2020.

NGUYEN, A. et al. An analysis of state-of-the-art activation functions for supervised deep neural network. In: IEEE. *2021 International Conference on System Science and Engineering (ICSSE)*. Ho Chi Minh City, Vietnam, 2021. p. 215–220.

NILSSON, K.; JÖNSSON, H.-E. *A comparison of image and object level annotation performance of image recognition cloud services and custom Convolutional Neural Network models*. 2019.

PERELMUTER, G. et al. Reconhecimento de imagens bidimensionais utilizando redes neurais artificiais. *Anais do VIII Sibgrapi*, p. 197–203, 1995.

PESCE, F. et al. Identification of glomerulosclerosis using ibm watson and shallow neural networks. *Journal of nephrology*, Springer, p. 1–8, 2022.

PRESSMAN, R. *Engenharia de Software*. Rio de Janeiro: McGraw Hill, 2011.

RASTOGI, N.; HENDLER, J. Whatsapp security and role of metadata in preserving privacy. *ArXiv*, 2017.

RUSSELL, S. J.; NORVIG, P. *Inteligência artificial*. [S.l.]: Elsevier, 2004.

SAINATH, T.; PARADA, C. Convolutional neural networks for small-footprint keyword spotting. 2015.

SANNER, M. F. et al. Python: a programming language for software integration and development. *J Mol Graph Model*, v. 17, n. 1, p. 57–61, 1999.

SCHMAGER, F. *Evaluating the Go programming language with design patterns*. Dissertação (Mestrado) — Victoria University of Wellington, 2010.

SRINATH, K. Python - the fastest growing programming language. *International Research Journal of Engineering and Technology (IRJET)*, v. 4, n. 12, p. 354–357, 2017.

SUTIKNO, T. et al. Whatsapp, viber and telegram: Which is the best for instant messaging? *International Journal of Electrical & Computer Engineering (2088-8708)*, v. 6, n. 3, 2016.

TRIVEDI, A. et al. Speech to text and text to speech recognition systems-areview. *IOSR J. Comput. Eng*, v. 20, n. 2, p. 36–43, 2018.

WAGMAN, M. *Reasoning processes in humans and computers: Theory and research in psychology and artificial intelligence*. Londres: Praeger Publishers, 2003.

WARNER, J. *Thank you for 100 million repositories*. 2018.

WEI, L. Legal risk and criminal imputation of weak artificial intelligence. In: IOP PUBLISHING. *IOP Conference Series: Materials Science and Engineering*. [S.l.], 2019. v. 490, n. 6.

WESTRUP, E.; PETTERSSON, F. *Using the go programming language in practice*. Dissertação (Mestrado) — Lund University, 2014.

ZIMMER, R. et al. *Technical report: supervised training of convolutional spiking neural networks with PyTorch*. Toulouse, França: arXiv, 2019.