PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Investigando soluções Text-to-speech

Pilar Baptista Fernandez

RELATÓRIO PROJETO FINAL II

CENTRO TÉCNICO CIENTÍFICO – CTC

DEPARTAMENTO DE INFORMÁTICA

Curso de Graduação em Ciência da Computação

Orientador: Marcos Kalinowski

Rio de Janeiro, Junho de 2021

Pilar Baptista Fernandez

Investigando soluções Text-to-speech

Relatório de Projeto Final, apresentado ao programa de Graduação em Ciência da Computação da PUC-Rio como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Marcos Kalinowski

Rio de Janeiro,

Junho de 2021

Resumo

FERNANDEZ, Pilar. KALINOWSKI, Marcos. Investigando soluções Text-to-speech. Rio de Janeiro, 2021. 37 p. Relatório Final de Projeto Final II – Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

Sistemas de síntese de fala são geralmente avaliados de acordo com a sua facilidade de entender e qualidade de áudio. Para este trabalho, foram selecionados sistemas de quatro principais fornecedores e avaliados de dois pontos de vista diferentes. A primeira avaliação, busca comparar aspectos das soluções text-to-speech ao serem utilizadas em outros sistemas, por desenvolvedores. A segunda teve como foco o usuário final, coletando informações para classificar as soluções de acordo com sua clareza e qualidade. Todas as soluções estudadas possuem pontos positivos e negativos. Com esse estudo, pode-se entender as características de cada uma delas e chegar a uma conclusão de qual seria mais indicada de acordo com as necessidades do sistema.

Palavras-chave

Text-to-speech, síntese de fala, SSML, estudo experimental.

Abstract

FERNANDEZ, Pilar. KALINOWSKI, Marcos. Investigando soluções Text-to-speech. Rio de Janeiro, 2021. 37 p. Relatório Final de Projeto Final II – Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

Speech synthesis systems are typically evaluated regarding their ease of understanding and audio quality. In this paper, four system from main suppliers were selected and evaluated from two different points of view. The first evaluation seeks to compare aspects of the text-to-speech solutions when used in other systems, by developers. The second one focused on the final user, collecting information to classify the solutions according to their clarity and quality. All the studied solutions have positive and negative points. With this study, it is possible to understand the characteristics of each one of them and to reach a conclusion of which would be more indicated according to the needs of the system.

Keywords

Text-to-speech, speech synthesis, SSML, experimental study

Sumário

1. Introdução

1.1 Contexto e motivação

Existem diversas soluções e ferramentas que utilizam a tecnologia Text-to-speech para realizar a leitura de textos por máquinas. No entanto, escolher qual se adequa mais a seu sistema ou aplicação pode não ser tão claro, já que não existe uma maneira simples e direta de compará-las facilmente sem implementar de fato cada uma.

1.2 Objetivos

Para chegar a uma conclusão de qual solução será mais indicada para atender às necessidades de uma aplicação, é necessário avaliar alguns pontos sobre cada uma delas e compará-las. Dificuldade de implementação, qualidade de reprodução, naturalidade da voz durante a fala e preço são exemplos de fatores a serem considerados e observados para obter uma conclusão.

Com isso, esse estudo busca fornecer uma maneira fácil e prática de entender como funcionam algumas das soluções existentes, com suporte ao português do Brasil, e uma ferramenta para auxiliar os desenvolvedores na escolha e na implementação das tecnologias text-to-speech.

1.3 Organização da Monografia

Para realizar este estudo, serão feitos exemplos práticos de uso de quatro tecnologias text-to-speech, comparando o esforço e tempo empregado para que cada uma delas esteja funcionando corretamente. Este primeiro estudo será feito buscando auxiliar os desenvolvedores a decidir qual é a melhor ferramenta com foco na implementação e documentação de cada *API*.

No entanto, levando em consideração o usuário final, que irá de fato utilizar o sistema que possui a tecnologia, também deve ser considerado qual das ferramentas possui uma melhor aceitação. Para isso, será feito um estudo experimental, onde usuários irão escutar textos lidos por todas as soluções estudadas e opinar qual teve o melhor desempenho em cada um dos pontos que serão apresentados.

2. Fundamentação Teórica e Trabalhos Relacionados

2.1 Introdução

Neste capítulo, conceitos sobre as tecnologias de síntese de fala text-to-speech serão abordados. Tendo o objetivo de fornecer uma fundamentação teórica para o trabalho a fim de verificar a sua importância, serão descritos exemplos de soluções já existentes no mercado e explicações de como funcionam.

2.2 Text-to-speech

Um sintetizador text-to-speech é um sistema capaz de ler qualquer texto em voz alta. Como citou TAYLOR et al 2009, sistemas como estes tem uma gama muito vasta de aplicações possíveis. Seu primeiro uso foi em sistemas de leitura para pessoas com deficiência visual, onde eram lidos textos de livros e convertidos em voz. Outra aplicação interessante é no aprimoramento da interação humano-computador. Um sintetizador de fala pode ser uma ferramenta muito útil para auxiliar no aprendizado de novas línguas, sendo utilizado para melhor entendimento de pronúncias.

O text-to-speech é uma tecnologia em que se nota muito claramente a evolução que passou e que continua passando. Cada vez mais a reprodução da voz humana é feita com mais precisão e qualidade e se tornando mais confortável de escutar. Apesar disto, certamente ainda há muitas oportunidades de melhorias e aperfeiçoamentos na técnica.

Como explicado por SASIREKHA et al 2012, a fala feita pela máquina se baseia na concatenação de unidades de fala que, quando unidas, formam uma palavra ou uma frase. Além disso, outro fator responsável pela naturalidade da fala pela máquina é o ritmo empregado durante o discurso.

Um sistema text-to-speech é formado por cinco componentes (SARIREKHA et al 2012):

- Análise e detecção de texto, onde é feita uma análise do texto a ser lido de forma a organizá-lo em uma lista de palavras. Quando necessário, é feito a conversão de números e abreviações para palavras escritas.
- Normalização do texto, que é a transformação do texto para sua forma de pronunciação, com o objetivo de definir de forma precisa a pontuação, acentuação e pontos de parada em uma frase.
- Análise fonética, que vai de fato converter os símbolos ortográficos das letras e sílabas para seus fonemas utilizando um alfabeto fonético.
- Modelagem de entonação, onde é feita a variação do discurso enquanto é falado. A entonação e o ritmo da fala têm um papel muito importante de imprimir emoções e garantir que fique o mais próximo possível da maneira que um humano fala.

2.3 Text-to-phoneme

Existem sistemas text-to-speech simples que possuem um escopo pequeno o suficiente para permitir que a fala seja produzida a partir de palavras pré-determinadas guardadas em um dicionário limitado. Porém, essas soluções não são escaláveis. Se tratando de uma linguagem real, o custo computacional de guardar todas as palavras existentes tornaria a síntese de fala inviável. Também, caso o texto contenha alguma grafia diferente da prevista na língua, o sistema não teria ferramentas para ser capaz de fazer a sua conversão para fala.

Uma forma de mitigar esse problema e possibilitar que qualquer texto seja lido corretamente, independente de sua semântica, é realizando um mapeamento de texto para fonema, chamado de text-to-phoneme.

Em sua tese, BILCU, 2008 define o mapeamento text-to-phoneme como um passo preliminar na síntese text-to-speech que está diretamente ligado com a naturalidade e o bom entendimento da fala. Isso é feito com a conversão das palavras do texto em sua transcrição fonética. Após a conversão, esses fonemas irão juntamente gerar a fala sintética.

A maioria dos sistemas text-to-speech possuem uma cobertura máxima de palavras com qualidade de pronunciação. Em geral, a estratégia de converter o texto em fonema se faz necessária quando são encontradas palavras diferentes ou não nativas à linguagem. Com isso, mesmo a frase ou palavra não tendo um significado conhecido ou não pertencendo ao dicionário da língua utilizada, ainda pode ser lido de uma maneira próxima da fala de um ser humano.

Segundo BILCU, uma boa forma de realizar o mapeamento de texto para fonema é utilizando redes neurais devido a sua habilidade de auto adaptação.

Para treinar uma rede neural a fazer o mapeamento, são feitas iterações onde um par de fonema e letra é fornecido. Com isso, forma-se um dicionário com poucas palavras que é passado como *input* para a rede uma quantidade de vezes necessárias até alcançar a convergência dos pesos sinápticos.

2.4 Redes Neurais

Redes neurais é uma área de estudo de aprendizado de máquina que mapeia um modelo matemático buscando replicar a estrutura neural do cérebro humano. GALLANT,1993 define um modelo de rede neural como um conjunto de unidades computacionais (ou células) e um conjunto de

conexões de dados. Essas células possuem conexões entre si com pesos positivos ou negativos.

Os neurônios artificiais possuem receptores de entrada que são responsáveis por perceber um novo sinal chegando na rede. Esses *inputs* são examinados pelas células e é calculado um valor positivo ou negativo chamado de ativação. Essa nova ativação gerada é passada pelas conexões até outras células presentes na rede neural. As conexões irão determinar se a ativação irá influenciar a célula receptora a produzir um resultado similar ou mudar sua ativação de acordo com o sinal do seu peso.

GALLANT, 1993 também explica que esse peso presente nas conexões determina o quanto a chegada da ativação para a célula receptora influencia no seu resultado. Quanto maior seu valor absoluto, maior será seu efeito na célula receptora.

As redes neurais auxiliam na criação de sistemas de inteligência artificiais. Através dessa tecnologia, pode ser realizado o treinamento do modelo para uma base de dados fornecida inicialmente. Desta forma, o sistema se torna capaz de obter novas informações e ter autonomia e capacidade de lidar com novos dados que possam chegar até ele futuramente em sua execução.

2.5 Soluções Text-to-speech

Para fazer os estudos e comparações do projeto foram selecionadas quatro soluções conhecidas na área de text-to-speech.

Para fazer requisições às API's utilizando o Node Js é necessário instalar o pacote com o gerenciador de pacotes npm e importar dentro do arquivo onde irá será feita a requisição. Além disso, todos eles contam com um sistema de autenticação, onde é gerado um *token* de acesso que deve ser passado no código, para então utilizar a API.

2.5.1 Watson Text to Speech (TTS)

Watson é a plataforma de serviços cognitivos da IBM para negócios (IBM,2021). Entre muitas das soluções oferecidas, Watson tem o serviço de text-to-speech que dispõe de mais de 14 vozes para cada 14 idiomas e dialetos diferentes.

O modelo da IBM utiliza três redes neurais profundas para fazer a transformação do texto em fala, entre elas Previsão Métrica da Fala, Melhoria da Acústica e Sintetização da voz.

2.5.2 Google Cloud Text-to-Speech

A solução text-to-speech da Google Cloud conta com um grupo de mais de 220 vozes em mais de 40 idiomas e variantes (GOOGLE, 2021). Além disso, é possível criar uma voz exclusiva além das já existentes pré-definidas.

Na sua criação, a Google se baseou na experiência de síntese de voz do *DeepMind*, que é uma empresa de pesquisa e desenvolvimento de máquinas de inteligência artificial.

2.5.3 Microsoft Azure

A Microsoft Azure oferece uma plataforma de conversão de texto em fala com mais de 150 vozes e em mais de 50 idiomas e variantes (MICROSOFT, 2021). Além disso, também possibilita o usuário criar sua própria voz personalizada.

2.5.4 Amazon Polly

Amazon Polly é o serviço de conversão de texto em fala da Amazon, onde é feita a leitura de textos em mais de 30 idiomas diferentes, possibilitando a personalização da voz (AMAZON, 2021).

A Amazon utiliza uma tecnologia neural que faz com que a voz e o jeito de falar nas leituras sejam mais compatíveis com o aplicativo, dependendo de sua natureza e de suas necessidades.

2.6 SSML

Todas essas soluções buscam aproximar cada vez mais a fala da máquina à dos humanos e garantir que seja o mais natural possível. Além disso, outro ponto relevante na utilização de uma tecnologia text-to-speech é a personalização. Ao oferecer a possibilidade da mudança da voz e de detalhes do modo como é feita a leitura do texto, se tem uma maior aceitação e confiança de que o resultado do discurso será como esperado. Para isso, é utilizado uma linguagem de marcação chamada Speech Synthesis Markup Language, ou SSML.

Speech Synthesis Markup Language (SSML) é uma linguagem de marcação baseada em XML que tem como objetivo transformar o texto recebido em fala sintetizada. Essa linguagem faz com que o texto se torne mais próximo de como falamos, e possibilita uma grande variedade de personalizações e melhorias em cada parte, fornecendo à máquina as informações necessárias para que o discurso saia da maneira esperada pelo ouvinte.

Linguagens de marcação, em geral, são utilizadas para definir padrões dentro de um documento. Isso é feito para definir o modo de exibição ou execução de um conteúdo de maneira não ambígua. Para tal, são utilizados marcadores, ou *tags*, que são imbuídos de um significado próprio, que, quando for lido posteriormente será reconhecido, tendo a informação de como aquele conteúdo deve ser executado.

A linguagem SSML possui algumas marcações que têm significados específicos, entre elas:

- <speak> é o elemento raiz de um documento SSML;
- <break> tem como função controlar a pausa entre as palavras durante a leitura de uma frase;

- <say-as> indica informações especificas sobre um texto que deve ser levado em conta ao ser lido em voz alta, como uma data, por exemplo;
- <audio> é utilizado para inserir arquivos de áudios gravados e reproduzi-lo juntamente com a saída da fala que foi sintetizada;
- <sub> faz a substituição de um texto, por ser utilizado para decifrar siglas, por exemplo;
- <prosody> personaliza o tom, a taxa de fala e o volume dos elementos contidos no texto;
- <emphasys> adiciona ou remove ênfase em partes do texto lido;

Deste modo, utilizando as marcações que compõem a linguagem SSML, obtemos um texto com marcações que irá gerar uma síntese de fala definida de maneira clara e objetiva.

2.7 Estudo Experimental

O estudo ou pesquisa experimental tem como objetivo testar hipóteses levantadas. Isso é feito pela manipulação das variáveis relacionadas com o objeto de estudo, a fim de chegar a alguma conclusão ao compará-las. Desta forma, é possível realizar generalizações com o uso das técnicas de coleta de amostragem.

Em seu livro, WOHLIN et al 2012, explica que experimentos são realizados corretamente quando controlamos a situação e buscamos manipular comportamentos diretamente, precisamente e sistematicamente. Além disso, para obter resultados satisfatórios é importante envolver mais de um objeto de estudo para comparar seus resultados.

Os experimentos podem ser orientados ao humano, onde o que varia para cada caso é a inspeção humana, ou à tecnologia, onde diferentes ferramentas são aplicadas para diferentes objetos a serem estudados.

2.8 Trabalhos Relacionados

Tendo percebido uma necessidade de atualizar o método de avaliação formal de sistemas text-to-speech, CARDOSO et al, 2015 busca criar um método para avaliar o potencial que esses sistemas possuem. Para isso, foram levados em conta quão compreensível, natural, preciso e inteligível são esses sistemas, e testado se pontos específicos da linguística do idioma eram claros na leitura. Os resultados obtidos, comparando falas artificiais com falas humanas, mostraram que ao testar uma leitura livre de um texto, os seres humanos receberam pontuações maiores que os sistemas. Porém, ao fazer o segundo teste focando em aspectos pontuais de uma fala, os resultados foram similares entre os seres humanos e o sistema de síntese de fala.

Avaliar uma ferramenta text-to-speech levando em consideração frases simples e curtas pode não ser suficiente para obter uma percepção real de como é o seu desempenho. Com o objetivo de entender melhor como se dá essa diferença, no trabalho gerado por CLARK et al 2019 foram investigadas três formas de se avaliar a naturalidade de uma síntese text-to-speech de um texto extenso. Com a comparação de resultados de avaliações de frases isoladas, parágrafos inteiros e uma seção do texto foi possível perceber diferenças. Mostrando assim, que para avaliar o desempenho de um sistema de síntese de fala para textos grandes, avaliar apenas frases curtas nesse sistema não é o bastante para obter um resultado confiável.

Em seu trabalho, CAMBRE et al 2020 teve como objetivo avaliar o comportamento de soluções text-to-speech focado em textos de longa duração. A pergunta principal era se aquela leitura seria agradável de escutar por vários minutos seguidos, não apenas palavras soltas. Com esse estudo, foi concluído que as sínteses de fala estão próximas de se tornar tão parecidas como a forma que um ser humano fala. Também

observaram que as diferentes vozes obtiveram um nível de desempenho e aceitação parecidos entre si.

2.9 Considerações Parciais

Com isso, para esse projeto, será estudado como funcionam as tecnologias Text-to-speech. Será realizado um exemplo, por meio de código, para cada uma das soluções mencionadas, levando em consideração o esforço de desenvolvimento e aprendizagem empregado. Posteriormente, será realizado um estudo experimental, que permitirá comparar as soluções do ponto de vista do usuário final, e buscar entender qual delas é mais agradável e possui um melhor entendimento do texto a ler lido.

3. Explorando Soluções Text-to-Speech

3.1 Introdução

Este capítulo tem como objetivo comparar as soluções do ponto de vista de um Desenvolvedor de Software. Para isso, serão considerados aspectos de implementações, tais como: qualidade da documentação e quantidade de modificação necessária do código fornecido nas documentações para se adequar ao sistema sendo desenvolvido.

3.2 Implementação Exemplo

Com o intuito de aproximar ao máximo de como as soluções seriam utilizadas em sistemas reais, foi desenvolvido uma aplicação web para enviar um texto e selecionar uma das ferramentas para lê-lo. O código pode ser encontrado em: https://github.com/pilarfernandezz/text-to-speech-tester.

A tela inicial da aplicação pode ser observada na imagem 3.1 deste capítulo.

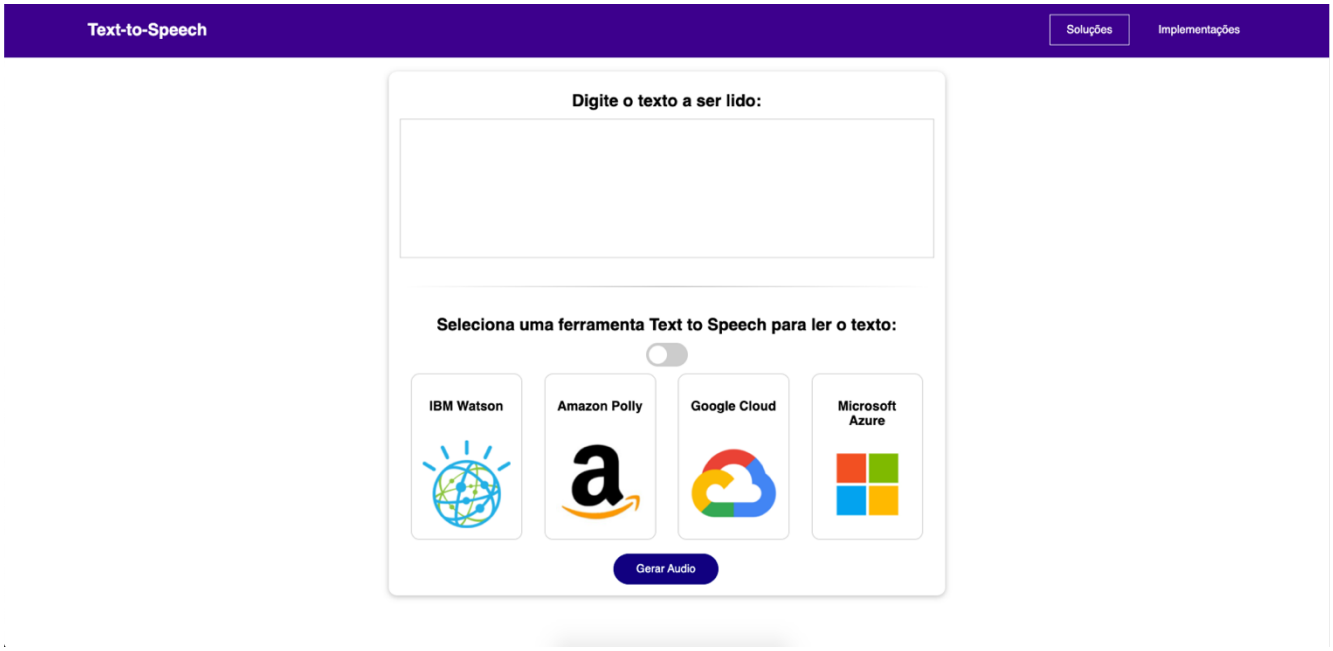

Figura 3.1

Ao digitar um texto e selecionar uma das aplicações disponíveis, o áudio é gerado e exibido abaixo, podendo ser executado, como mostra a imagem 3.2 a seguir.

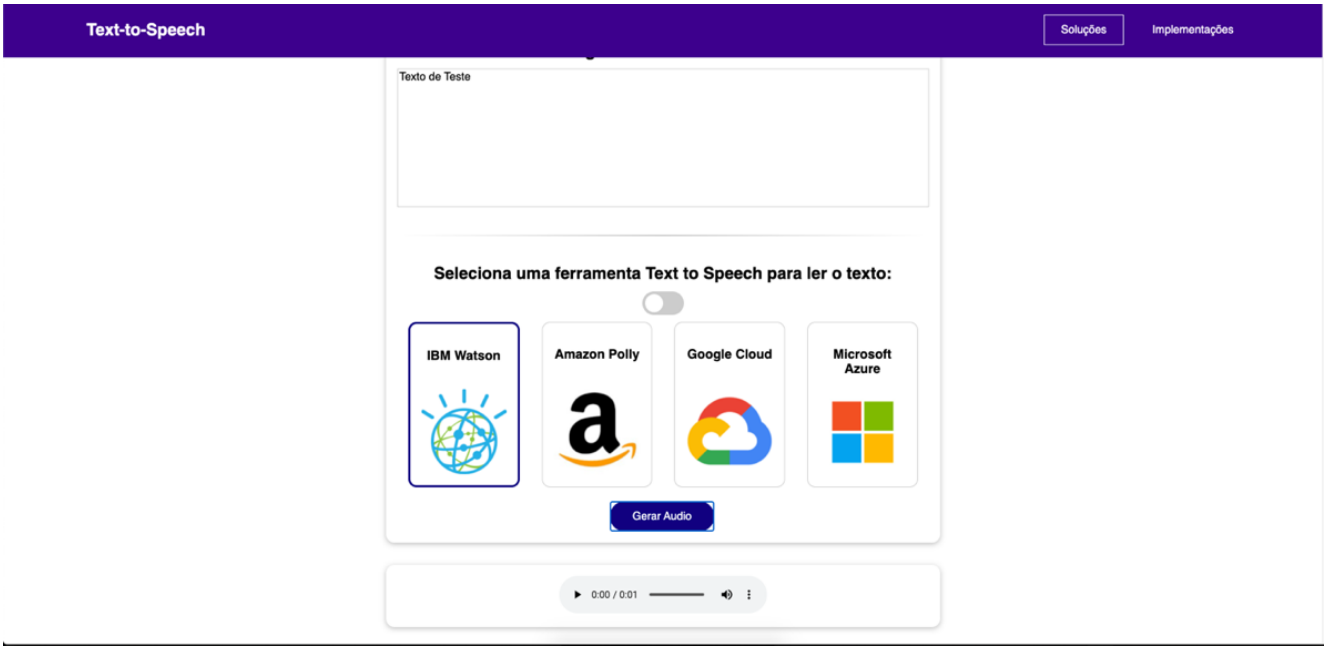

Figura 3.2

Para fornecer ao usuário uma forma rápida de visualizar e comparar os códigos utilizados para realizar a requisição a cada uma das API's, foi criada uma página (figura 3.3) exibindo as quatro soluções. Quando clicadas, é exibido o código fonte utilizado para fazer todo o processo de conversão de texto em fala naquela solução.

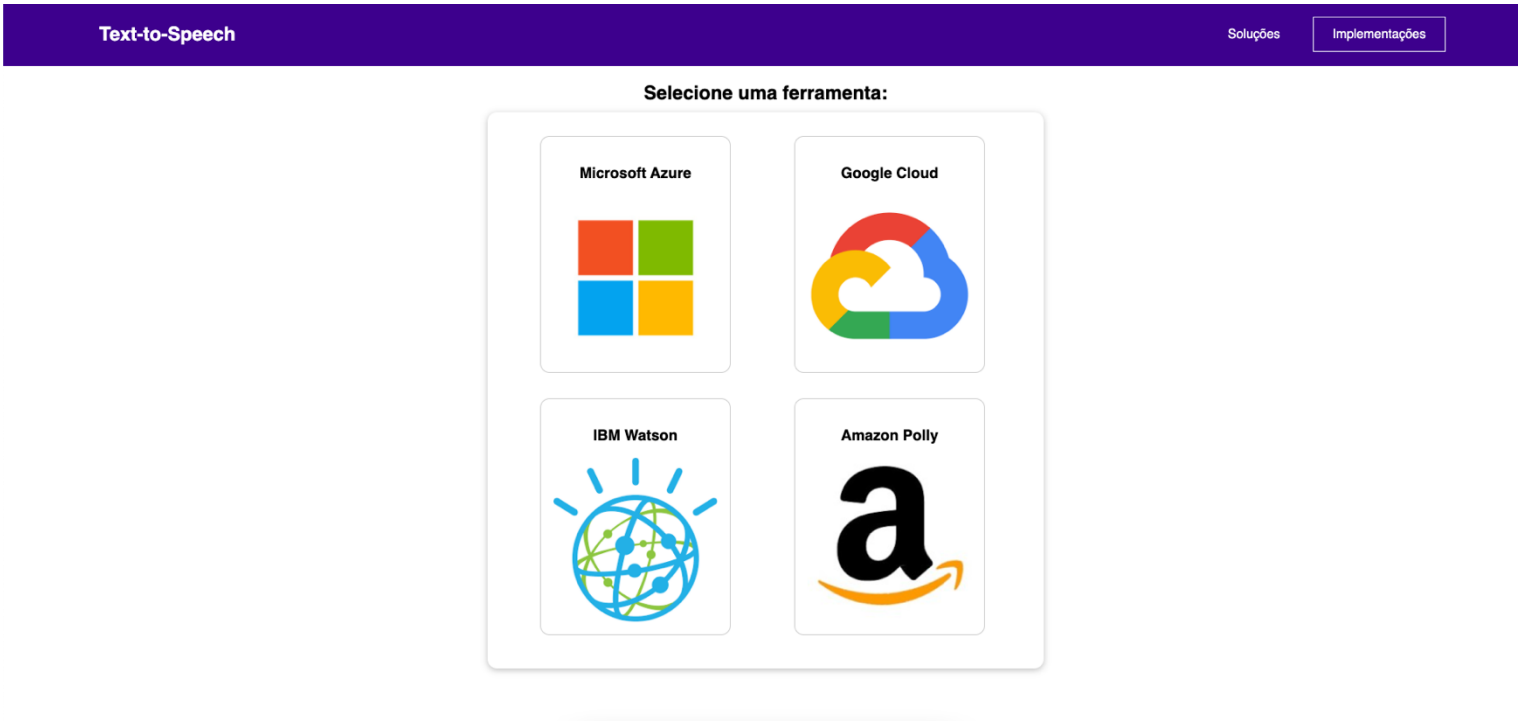

Figura 3.3

3.2.1 Arquitetura da Solução

A arquitetura utilizada para desenvolver a solução foi a de microserviços (figura 3.4).

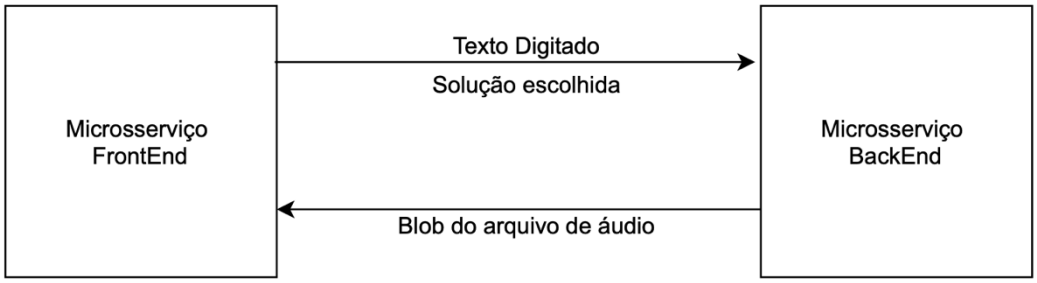

Figura 3.4

- Microserviço FrontEnd: Desenvolvido em Vue.js, utiliza cliente HTTP Axios para realizar requisições ao servidor.
- Microserviço BackEnd: Servidor para realizar as requisições às *API's* das soluções testadas. Recebe o requerimento do cliente (descrito acima) com o texto digitado e a solução escolhida e faz a chamada para a API adequada. Esse procedimento pode ser observado na figura 3.5. O servidor foi desenvolvido em Node.js e utiliza o framework Express para auxiliar o sistema de rotas e gerenciar as requisições HTTP. Seu código pode ser encontrado em: https://github.com/pilarfernandezz/text-to-speech-server. Como na parte visual da aplicação só pode ser exibido o arquivo após ter terminado de ser gravado, foi necessário garantir que todas as requisições às *API's* terceiras sejam assíncronas.

```
import { generateAudioIBM } from './toolsAPIcalls/IBM.js';
import { generateAudioMS } from './toolsAPIcalls/Microsoft.js';
import { generateAudioGoogle } from './toolsAPIcalls/Google.js';
import { generateAudioAWS } from './toolsAPIcalls/AWS.js';

const create = async (text, tool) => {
    var file;
    if(tool === 'IBM'){
        file = await generateAudioIBM(text);
    }
    else if(tool === 'Microsoft'){
        file = await generateAudioMS(text);
    }
    else if(tool === 'Google'){
        file = await generateAudioGoogle(text);
    }
    else if(tool === 'AWS'){
        file = await generateAudioAWS(text);
    }

    return file;
};

export default { create };
```

Figura 3.5

Para todas as soluções foram utilizadas suas documentações oficiais como guia para fazer as requisições.

3.2.2 Implementação IBM

Para realizar requisições à API Watson Text-to-Speech da IBM foi necessário criar uma conta na IBM Cloud e gerar uma chave e uma URL de serviço, que serão utilizadas como método de autenticação. Essa chave e URL são passadas no código para o método de autenticação *IamAuthenticator* (IBM,2021). Para garantir a segurança dessas informações, foi criado um arquivo *.env*, que guarda todas as variáveis de ambiente da aplicação.

Como o método *synthesize* (figura 3.6) já é assíncrono, apenas foi chamado dentro de uma função assíncrona com *await*, o que garantiu que só iria retornar o arquivo quando o método já estivesse terminado seu processamento.

A API já suporta textos SSML por default, logo não foi necessário fazer nenhuma mudança para adequar esse requerimento. Porém, o Watson Text-to-Speech não suporta totalmente alguns comandos na língua portuguesa, como a *tag "say-as"*, por exemplo.

```
import fs from 'fs';
import TextToSpeechV1 from 'ibm-watson/text-to-speech/v1.js';
import { IamAuthenticator } from 'ibm-watson/auth/index.js';

export const generateAudioIBM = async (text) => {
    var file = null;
    const textToSpeech = new TextToSpeechV1({
        authenticator: new IamAuthenticator({
            apikey: process.env.IBMAPIKEY,
        }),
        serviceUrl: process.env.IBMSERVICEURL,
    });

    const params = {
        text,
        voice: 'pt-BR_IsabelaV3Voice', // Optional voice
        accept: 'audio/wav'
    };

    await textToSpeech
        .synthesize(params)
        .then(response => {
            const audio = response.result;
            return textToSpeech.repairWavHeaderStream(audio);
        })
        .then(repairedFile => {
            fs.writeFileSync('./file/audio.wav', repairedFile);
            console.log('audio.wav written with a corrected wav header');
        })
        .catch(err => {
            console.log(err);
        });

    file = fs.readFileSync('./file/audio.wav');
    return file
}
```

Figura 3.6

3.2.3 Implementação Microsoft

Para utilizar a ferramenta de conversão de fala da Microsoft Azure é necessário ter uma conta Azure e o recurso do serviço criado nesta conta (MICROSOFT,2021). Com isso, é gerada uma chave que será utilizada como o método de autenticação da API. Assim como a chave utilizada na solução da IBM, esta também ficará no mesmo arquivo .env por questões de segurança.

O método *speakSsmlAsync* (figura 3.7) da forma como se encontra na documentação não é assíncrono. Portanto, para assegurar que o arquivo só será retornado para o cliente HTTP quando tiver concluído, foi necessário adicionar algumas ferramentas nativas do JavaScript como *Promises*.

```
1  import sdk from"microsoft-cognitiveservices-speech-sdk";
2
3  export const generateAudioMS = async (text) => {
4      var subscriptionKey = process.env.MICROSOFTAPIKEY;
5      var serviceRegion = process.env.MICROSOFTSERVICEREGION;
6
7      var filename = "audio.wav";
8
9      var audioConfig = sdk.AudioConfig.fromAudioFileOutput(filename);
10     var speechConfig = sdk.SpeechConfig.fromSubscription(subscriptionKey, serviceRegion);
11
12     var synthesizer =  new sdk.SpeechSynthesizer(speechConfig, audioConfig);
13     const ssml = ```<speak version="1.0" xmlns="https://www.w3.org/2001/10/synthesis"
14                     xml:lang="pt-BR"><voice name="pt-BR-FranciscaNeural">```
15                     + text +'</voice></speak>'
16
17     let writeText = await (() => {
18         return new Promise((resolve, reject) => {
19             synthesizer.speakSsmlAsync(
20                 ssml,
21                 result => {
22                     if (result.errorDetails) {
23                         reject(result.errorDetails);
24                     } else {
25                         resolve(result.privAudioData);
26                     }
27                     synthesizer.close();
28                 },
29                 error => {
30                     console.log('error',error)
31                     synthesizer.close();
32             });
33         });
34     })();
35     try{
36         return writeText
37     } catch(e) {
38         console.log(e)
39     }
40 };
```

Figura 3.7

Antes de passar o texto para o método que irá realizar a conversão para fala, é criada uma variável SSML que guarda o texto fornecido em uma formatação SSML, logo, não foi feita nenhuma modificação para permitir que seja passado um SSML para a API.

3.2.4 Implementação Google

A Google Cloud utiliza a mesma conta da Google, logo, caso já possua uma conta comum, não será necessário criar uma nova para fazer a utilização das *API's* (GOOGLE, 2021).

O método de autenticação da Google, diferentemente das soluções anteriores, é feito por variável de ambiente na máquina. Ao ativar a API Cloud Text-to-Speech é gerado um arquivo JSON que contém a chave da conta de serviço. Para fazer a autenticação é criada uma variável de ambiente GOOGLE_APPLICATION_CREDENTIALS que tem como valor o caminho do arquivo JSON criado anteriormente.

Não foram necessárias mudanças adicionais para garantir o assincronismo além de realizar a chamada ao método *synthesizeSpeech* (figura 3.8) e o de escrita do arquivo de áudio dentro de um método *async* com *await*.

Da maneira como a documentação fornece o código, a API não reconhece as marcações SSML. Para isso, foi necessário adicionar a opção *'ssml': text* à variável de configuração *request*.

```
import textToSpeech from '@google-cloud/text-to-speech';
import fs from 'fs';
import util from 'util';

export const generateAudioGoogle = async (text) => {
    const client = new textToSpeech.TextToSpeechClient();

    const request = {
        'input':{
            'ssml': text
          },        voice: { languageCode: 'pt-BR',
                'name': 'pt-BR-Standard-A',
                ssmlGender: 'NEUTRAL' },
        audioConfig: { audioEncoding: 'MP3' },
    };

    const [response] = await client.synthesizeSpeech(request);

    const writeFile = util.promisify(fs.writeFile);
    await writeFile('./file/output.mp3', response.audioContent, 'binary');
    console.log('Audio content written to file: output.mp3');

    const file = fs.readFileSync('./file/output.mp3');
    return file
}
```

Figura 3.8

3.2.5 Implementação AWS

Para utilizar os serviços oferecidos pela Amazon Polly, deve ser feito um cadastro no Amazon Web Services (AWS) e criar um *IAM user* (AMAZON, 2021).

Para obter acesso às chaves e segredos de autenticação, é necessário instalar e configurar o AWS Command Line Interface (AWS CLI), como é explicado na documentação. O método de autenticação é por variável de ambiente, que deve ser criada na máquina.

O método *writeText* (figura 3.9) não é assíncrono, portanto, foi preciso adicionar *Promise* para aguardar o término da gravação do arquivo.

```
import AWS from 'aws-sdk';

const Polly = new AWS.Polly({
    signatureVersion: 'v4',
    region: 'us-east-1'
})

export const generateAudioAWS = async (text) => {
    let params = {
    'TextType': 'ssml',
    'Text': text,
    'OutputFormat': 'mp3',
    'VoiceId': 'Vitoria'
    }

    let writeText = await (() => {
        return new Promise((resolve, reject) => {
            Polly.synthesizeSpeech(params, (err, data) => {
                if (err) {
                    reject(err);
                    console.log(err.code)
                } else if (data) {
                    resolve(data.AudioStream)
                }
            })
        });
    })();
    try{
        return writeText
    } catch(e) {
        console.log(e)
    }
}
```

Figura 3.9

Além disso, da forma em que se encontra na documentação, textos no formato SSML não são interpretados. Para fazer com que texto SSML seja lido de acordo, foi adicionada a opção *'TextType': 'ssml'* na variável de configuração da API.

3.3 Estrutura da aplicação FrontEnd

Como foi mencionado anteriormente, a aplicação possui duas páginas, uma para fazer a criação e reprodução do áudio e outra para exibir os códigos utilizados nas soluções.

Com isso, a estrutura do código em Vue.js ficou em duas pastas principais: *views* e *components* (figura 3.10). Cada página web corresponde a um *view* e cada *view* utiliza componentes criados na pasta *components*. A pasta de componentes foi subdivida de acordo com o local no qual os componentes foram utilizados na aplicação.

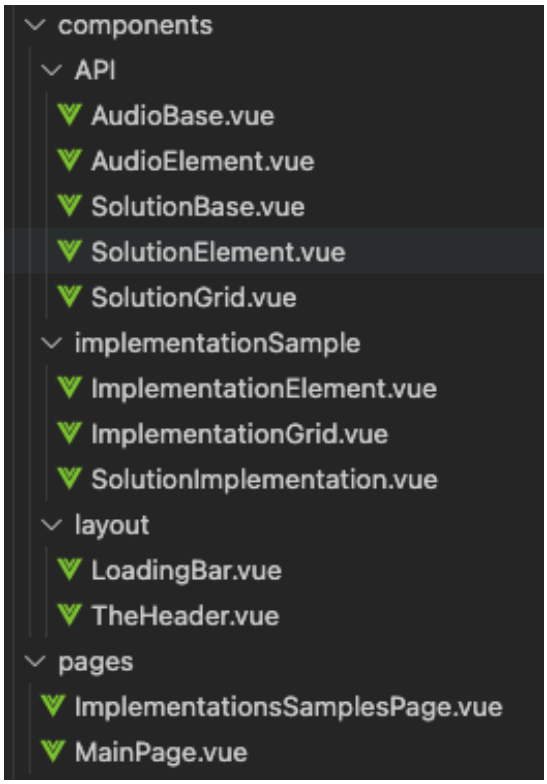

Figura 3.10

3.4 Comparação do ponto de vista do desenvolvedor

Todas as soluções possuem boas documentações e API funcionais. Porém, existem alguns pontos que as diferem.

O método de autenticação por variáveis de ambiente presentes nas soluções da AWS e da Google foi mais demorado de implementar. Além disso, é um método que pode gerar uma barreira ao tentar rodar a aplicação em uma máquina diferente ou ao fazer o *deploy*. Já nas soluções da Microsoft e da IBM, as variáveis de autenticação são guardadas dentro da aplicação, apenas requerendo métodos seguros para guardá-las e garantir que não serão divulgadas ou publicadas em repositórios públicos.

Ao se tratar de modificações necessárias no código para atender os requisitos de resposta HTTP, as soluções que necessitaram menos mudanças foram as da IBM e Google. Já nas chamadas às *API's* da Microsoft e AWS foi preciso modificar consideravelmente o código original fornecido pelas empresas.

Ao se levar em conta suporte a textos com a formatação SSML, as soluções que já eram capazes de receber e interpretar esses textos recebidos são as da IBM e da Microsoft.

Finalmente, com o volume de texto passado nos testes, não foi percebida uma diferença relevante de desempenho e tempo de resposta das soluções.

3.4 Considerações Parciais

Ao levar em conta todas os pontos vistos acima a respeito de cada uma das soluções, é possível notar que todas têm pontos negativos e positivos. Logo, estas informações têm o intuito de auxiliar o desenvolvedor no momento da escolha de qual API de conversão de texto utilizar de acordo com as necessidades e requisitos do sistema a ser desenvolvido.

4. Estudo Experimental investigando as soluções

4.1 Introdução

Esse capítulo tem como objetivo comparar as soluções em estudo do ponto de vista do usuário final, que estará potencialmente utilizando aplicações que possuem o recurso de síntese de fala. Para isso, foi feito um questionário com base em exemplos de conversão de texto em fala que procurou compreender os pontos fortes e fracos de cada uma das soluções do ponto de vista dos entrevistados selecionados.

4.2 Planejamento

4.2.1 Definição do objetivo

Para definir os objetivos do estudo com clareza, será utilizado o *template Goal Question Metric* (GQM) criado por CALDIERA et al 1994.

Analisar Ferramentas Text-to-speech
com o propósito de caracterizar
com respeito a qualidade, clareza
do ponto de vista de usuários de sistemas de software
no contexto do uso das ferramentas para geração de um arquivo de áudio a partir de um arquivo comum utilizando notação SSML.

4.2.2 Procedimentos de coleta

A escolha da seleção da amostra foi feita por amostragem por conveniência e foi escolhida de maneira a obter pessoas de diferentes idades, sexo e familiaridade com a tecnologia estudada.

Cada um dos participantes separadamente escutou o mesmo áudio criado pelas quatro soluções estudadas. Após a execução de cada áudio foram feitas as perguntas a seguir:

1- Em relação a qualidade, quão parecido é com um humano?
2- Quão fácil foi entender com clareza o texto?

A primeira pergunta foi inspirada no teste de Turing (TURING, 1950), que busca compreender se uma inteligência artificial consegue se passar por um ser humano. Mesmo considerando que as chances das tecnologias atuais passarem no teste sejam pequenas, a pergunta será feita com o objetivo de analisar o quão próximo deste feito cada solução estudada poderia estar.

As perguntas deverão ser respondidas em uma escala *Likert*, que irá representar um conjunto de opções de resposta para cada pergunta:

- Extremamente parecido/claro
- Muito parecido/claro
- Mais ou menos parecido/claro
- Um pouco parecido/claro
- Nem um pouco parecido/claro

Somente após as quatro soluções terem sido escutadas e as perguntas acima respondidas para cada uma delas, será pedido que o participante distribua cinco fichas entre as quatro soluções disponíveis.

Com isso, é obtido uma análise individual de cada uma das ferramentas de conversão de texto e, posteriormente, compará-las juntamente.

O texto a ser utilizado para gerar os quatro arquivos de áudio foi o seguinte:

“Era uma vez um jovem pastor que costumava levar o seu rebanho de ovelhas para a serra a pastar. Como estava sozinho durante todo o dia, aborrecia-se muito. Então, pensou numa maneira de ter companhia e de se divertir um pouco. Voltou-se na direção da aldeia e gritou: "Lobo! Lobo!". Os camponeses correram em seu auxílio. Não gostaram da graça, mas alguns deles acabaram por ficar junto do pastor por algum tempo. O rapaz ficou tão contente que repetiu várias vezes a façanha. Alguns dias depois,

um lobo saiu da floresta e atacou o rebanho. O pastorzinho pediu ajuda, gritando ainda mais alto do que costumava fazer: "Lobo! Lobo!". Como os camponeses já tinham sido enganados várias vezes, pensaram que era mais uma brincadeira e não o foram ajudar. O lobo pôde encher a barriga à vontade porque ninguém o impediu. Quando regressou à aldeia, o rapaz queixou-se amargamente, mas o homem mais velho e sábio da aldeia respondeu-lhe: "Na boca do mentiroso, o certo é duvidoso.""

A linguagem SSML ainda não é totalmente unificada, e possui variação entre as soluções da forma de utilizá-la. Os textos com as respectivas formatações SSML se encontram no Anexo I deste documento.

4.2.3 Procedimentos de análise

Com o estudo finalizado, será feita a análise dos dados obtidos.

Cada uma das opções de métrica da escala *Likert* utilizada irá contabilizar uma quantidade relativa de pontos para cada uma das soluções do estudo.

Com isso, será feita uma contagem de frequência simples e uma análise qualitativa comparando a quantidade de pontos que cada solução recebeu.

Também será contabilizada o número de fichas que cada solução receber na segunda parte do questionário e feita a comparação das quantidades obtidas.

4.3 Operação

O estudo foi feito com nove participantes. Com cada um dos participantes foi realizada uma chamada pelo aplicativo Zoom para possibilitar o compartilhamento da tela com a aplicação descrita no capítulo 3 deste trabalho. Cada sessão durou em média 20 minutos.

Foram executados os quatro áudios gerados por cada uma das soluções e feitas as perguntas planejadas anteriormente. De modo a obter resultados menos influenciados possível, os participantes não tinham o conhecimento de quais soluções estavam escutando.

Para guardar as respostas e possibilitar a análise dos resultados obtidos com os estudos, foi utilizado o Google Forms. O formulário foi preenchido juntamente com os participantes durante a duração da chamada à medida que as perguntas feitas foram respondidas.

4.4 Resultados

Na primeira pergunta, em relação a qualidade e semelhança com um humano foi o obtido o resultado que pode ser observado no gráfico presente na Figura 4.1.

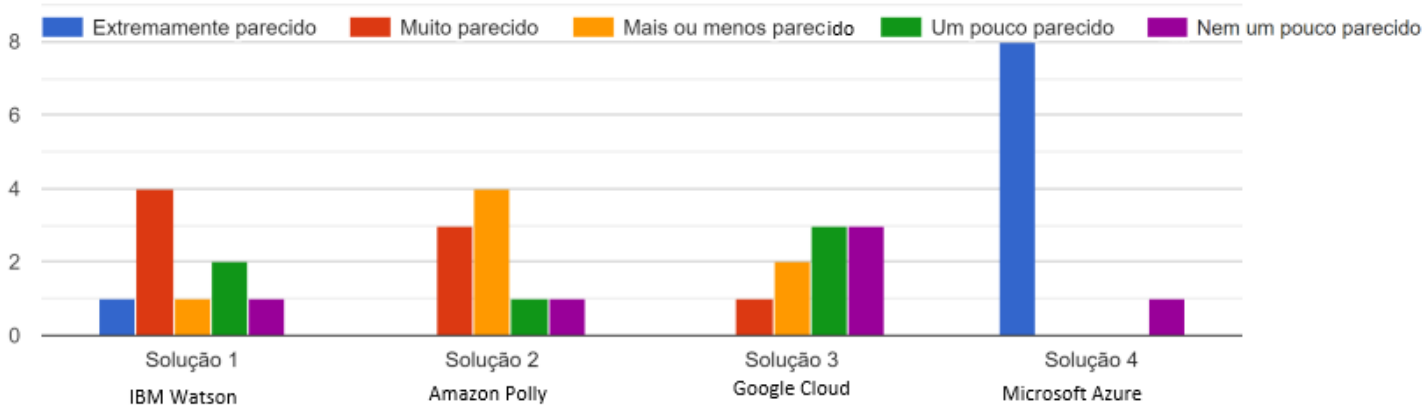

Figura 4.1 – Resultado da avaliação da qualidade

Foi gerado outro gráfico para análise da segunda pergunta que busca compreender a clareza do texto lido. Esse gráfico pode ser observado na figura 4.2 abaixo.

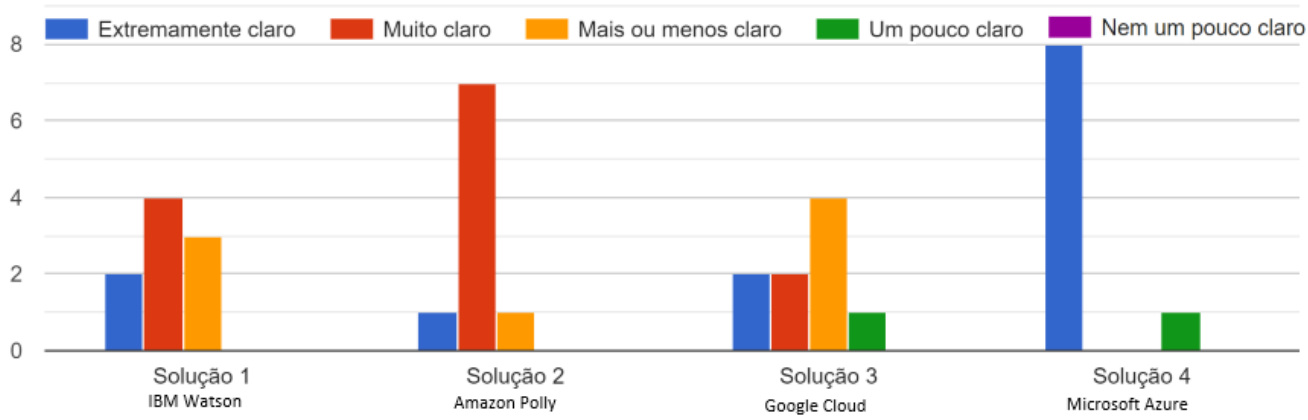

Figura 4.2 – Resultado da avaliação de clareza

Após a análise de ambos os gráficos, é possível perceber que a solução Text-to-Speech da Microsoft Azure obteve resultados melhores que as outras soluções, tanto no aspecto de clareza quanto de qualidade. Já as outras soluções obtiveram resultados comparáveis entre si.

Na segunda parte do estudo, foram distribuídas fichas para as soluções de acordo com a preferência de cada um dos participantes. Com isso, foi possível obter o seguinte resultado:

IBM Watson: 1 + 1 + 2 + 2 + 1 + 2 + 0 + 0 + 1 = 10 pontos
Amazon Polly: 1 + 0 + 0 + 0 + 1 + 1 + 1 + 1 + 1 = 6 pontos
Google Cloud: 0 + 0 + 0 + 3 + 0 + 0 + 1 + 1 + 0 = 5 pontos
Microsoft Azure: 3 + 4 + 3 + 0 + 3 + 2 + 3 + 3 + 3 = 24 pontos

Assim como nos testes anteriores, a solução oferecida pela Microsoft Azure obteve um resultado melhor, recebendo uma quantidade de pontos que corresponde a mais que a soma do total de pontos distribuídos pelos participantes do estudo para as demais soluções.

Para justificar os resultados obtidos pelo estudo e compreender os pontos positivos e negativos das soluções, foram levados em consideração alguns comentários realizados pelos participantes do estudo ao responder as perguntas.

Segundo os comentários, o motivo da solução da Microsoft Azure ter sido mais bem classificada e pontuada se deu pela boa fluência e entonação das pontuações na leitura do texto, que o tornou mais agradável de escutar do que as outras. Já a solução da Google Cloud, que obteve os piores resultados gerais, foi dita ser mais artificial e que falta clareza nas pontuações. No entanto, não foi uma opinião unânime. Todas as soluções tiveram pontos positivos e negativos levantados.

Em geral, as soluções da IBM Watson e da Amazon Polly receberam opiniões similares. Consideradas robóticas e com um aspecto sintético na leitura, mas que não são fatores que impossibilitam a compreensão do texto.

4.5 Considerações Parciais

Com o estudo realizado, foi possível observar que a solução da Microsoft Azure foi a que recebeu uma melhor aceitação entre os participantes do estudo. Porém, todas as soluções em geral foram bem recebidas e foram capazes de cumprir a função de realizar uma leitura de um texto de forma compreensível.

5. Considerações Finais

5.1 Limitações

Como as aplicações são precificadas de acordo com a utilização e o número de requisições realizadas, não foi possível realizar o *deploy* da aplicação desenvolvida e disponibilizá-la publicamente.

5.2 Contribuições

Com os estudos gerados a partir deste trabalho, foi possível obter uma compreensão de conceitos sobre a tecnologia de conversão de texto em fala Text-to-Speech. Após isto, foram estudadas quatro soluções para então ser capaz de realizar comparações do ponto de vista de um desenvolvedor e de um potencial usuário final.

Deste modo, são oferecidas informações úteis para um desenvolvedor que busca escolher uma tecnologia de conversão de fala para o seu próximo projeto de acordo com suas necessidades.

6. Referências bibliográficas

AMAZON. Amazon Polly. **What Is Amazon Polly?** Disponível em: https://docs.aws.amazon.com/polly/latest/dg/what-is.html. Acesso em: Maio de 2021.

BILCU, B. **Text-to-phoneme mapping using neural networks.** Tampere University of Technology, 2008.

CALDIERA, G., BASILI V.R., ROMBACH, H.D. The goal question metric approach. **Encyclopedia of software engineering**, p. 528-532, 1994.

CAMBRE, J. *et al.* **Choice of voices: A large-scale evaluation of text-to-speech voice quality for long-form content.** CHI Conference on Human Factors in Computing Systems, 2020.

CARDOSO, W. *et al.* Eva**luating text-to-speech synthesizers**. Research-publishing. net, 2015.

CLARK, R. *et al.* **Evaluating long-form text-to-speech: Comparing the ratings of sentences and paragraphs.** arXiv preprint arXiv:1909.03965, 2019.

GALLANT, Stephen I. **Neural network learning and expert systems.** MIT press, 1993.

GOOGLE. Google Cloud, 2021. **Text-to-Speech documentation.** Disponível em: https://cloud.google.com/text-to-speech/docs. Acesso em: Maio de 2021.

IBM. IBM Cloud, 2021. **IBM Cloud API Docs.** Disponível em: https://cloud.ibm.com/apidocs/text-to-speech. Acesso em: Maio de 2021.

MICROSOFT. Microsoft Azure, 2021. **Get started with text-to-speech.** Disponível em: https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/get-started-text-to-speech. Acesso em: Maio de 2021.

SASIREKHA, D., CHANDRA, E. **Text to speech: a simple tutorial**. International Journal of Soft Computing and Engineering (IJSCE). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.682.5362&rep=rep1&type=pdf, 2012.

TAYLOR, P. **Text-to-speech synthesis.** Cambridge University Press, 2009.

TURING, A. M. **Computing Machinery and Intelligence**. Mind, 1950.

WOHLIN, C. *et al.* **Experimentation in software engineering**. Springer Science & Business Media, 2012.

Anexo I – Textos com formatação SSML

1 Watson Text to Speech

<speak>
Era uma vez um jovem pastor que costumava levar o seu rebanho de ovelhas para a serra a pastar. Como estava sozinho durante todo o dia, aborrecia-se muito. Então, pensou numa maneira de ter companhia e de se divertir um pouco. Voltou-se na direção da aldeia e gritou: <prosody rate="fast">"Lobo! Lobo!" </prosody>. Os camponeses correram em seu auxílio. Não gostaram da graça, mas alguns deles acabaram por ficar junto do pastor por algum tempo. O rapaz ficou tão contente que repetiu várias vezes a façanha. <break time="1s"/>Alguns dias depois, um lobo saiu da floresta e atacou o rebanho. O pastorzinho pediu ajuda, gritando ainda mais alto do que costumava fazer: <prosody rate="fast">"Lobo! Lobo!" </prosody>. Como os camponeses já tinham sido enganados várias vezes, pensaram que era mais uma brincadeira e não o foram ajudar. O lobo pôde encher a barriga à vontade porque ninguém o impediu. Quando regressou à aldeia, o rapaz queixou-se amargamente, mas o homem mais velho e sábio da aldeia respondeu-lhe: "Na boca do mentiroso, o certo é duvidoso."
</speak>

2 Google Cloud Text-to-Speech

<speak>
Era uma vez um jovem pastor que costumava levar o seu rebanho de ovelhas para a serra a pastar. Como estava sozinho durante todo o dia, aborrecia-se muito. Então, pensou numa maneira de ter companhia e de se divertir um pouco. Voltou-se na direção da aldeia e gritou: <prosody volume="+20.00%" >"Lobo! Lobo!" </prosody>. Os camponeses correram em seu auxílio. Não gostaram da graça, mas alguns deles acabaram por ficar junto do pastor por algum tempo. O rapaz ficou tão contente que

repetiu várias vezes a façanha. <break time="1s"/> Alguns dias depois, um lobo saiu da floresta e atacou o rebanho. O pastorzinho pediu ajuda, gritando ainda mais alto do que costumava fazer: <prosody volume="+20.00%" >"Lobo! Lobo!" </prosody>. Como os camponeses já tinham sido enganados várias vezes, pensaram que era mais uma brincadeira e não o foram ajudar. O lobo pôde encher a barriga à vontade porque ninguém o impediu. Quando regressou à aldeia, o rapaz queixou-se amargamente, mas o homem mais velho e sábio da aldeia respondeu-lhe: "Na boca do mentiroso, o certo é duvidoso.” </speak>

3 Microsoft Azure

Era uma vez um jovem pastor que costumava levar o seu rebanho de ovelhas para a serra a pastar. Como estava sozinho durante todo o dia, aborrecia-se muito. Então, pensou numa maneira de ter companhia e de se divertir um pouco. Voltou-se na direção da aldeia e gritou: <prosody volume="+20.00%" >"Lobo! Lobo!" </prosody>. Os camponeses correram em seu auxílio. Não gostaram da graça, mas alguns deles acabaram por ficar junto do pastor por algum tempo. O rapaz ficou tão contente que repetiu várias vezes a façanha. <break time="1s"/>Alguns dias depois, um lobo saiu da floresta e atacou o rebanho. O pastorzinho pediu ajuda, gritando ainda mais alto do que costumava fazer: <prosody volume="+20.00%" >"Lobo! Lobo!" </prosody>. Como os camponeses já tinham sido enganados várias vezes, pensaram que era mais uma brincadeira e não o foram ajudar. O lobo pôde encher a barriga à vontade porque ninguém o impediu. Quando regressou à aldeia, o rapaz queixou-se amargamente, mas o homem mais velho e sábio da aldeia respondeu-lhe: "Na boca do mentiroso, o certo é duvidoso."

4 Amazon Polly

<speak>
Era uma vez um jovem pastor que costumava levar o seu rebanho de ovelhas para a serra a pastar. Como estava sozinho durante todo o dia, aborrecia-se muito. Então, pensou numa maneira de ter companhia e de se divertir um pouco. Voltou-se na direção da aldeia e gritou: <prosody volume="+6dB" >"Lobo! Lobo!" </prosody>. Os camponeses correram em seu auxílio. Não gostaram da graça, mas alguns deles acabaram por ficar junto do pastor por algum tempo. O rapaz ficou tão contente que repetiu várias vezes a façanha. <break time="1s"/>Alguns dias depois, um lobo saiu da floresta e atacou o rebanho. O pastorzinho pediu ajuda, gritando ainda mais alto do que costumava fazer: <prosody volume="+6dB" >"Lobo! Lobo!" </prosody>. Como os camponeses já tinham sido enganados várias vezes, pensaram que era mais uma brincadeira e não o foram ajudar. O lobo pôde encher a barriga à vontade porque ninguém o impediu. Quando regressou à aldeia, o rapaz queixou-se amargamente, mas o homem mais velho e sábio da aldeia respondeu-lhe: "Na boca do mentiroso, o certo é duvidoso."
</speak>