AIVOZ: PLATAFORMA DE AUXÍLIO INTELIGENTE NO DESENVOLVIMENTO DE APLICAÇÕES *WEB* ORIENTADAS POR VOZ

AIVOZ: INTELLIGENT ASSISTANCE PLATFORM IN THE DEVELOPMENT OF VOICE-DRIVEN WEB APPLICATIONS

Lucas Marques Bandeira*
Henrique Viana Oliveira**

RESUMO

Com uma crescente taxa em inovação na área da Tecnologia da Informação, vários programadores estão sendo requisitados e atribuídos á funções. Logo, uma série de funções podem se tornar repetitivas, exaustivas e, consequentemente, acarretar em alguns problemas devido às posturas adotadas pelos trabalhadores. Um desses problemas é conhecido como Lesão por Esforço Repetitivo (LER). Desta forma, este trabalho tem como objetivo o desenvolvimento de uma solução que possibilita evitar ou prevenir possíveis futuras lesões durante o processo de desenvolvimento, assim criando a plataforma AIVOZ. A AIVOZ permite que os usuários possam utilizar a voz para criar estruturas e linhas de códigos em linguagens *HTML* e *CSS* em seus projetos *web*. Para a realização desta pesquisa foram estudadas as aplicações de modelos de Reconhecimento Automático de Voz. O projeto utiliza a *API Web Speech* para o reconhecimento e interpretação de voz, que foi projetada para análise e síntese de fala, e tem como base os Modelos Ocultos de Markov para a captação e conversão do sinal analógico para digital.

Palavras-chave: 1. Reconhecimento Automático de Voz 2. Desenvolvimento Web 3. API Web Speech.

ABSTRACT

With an increasing rate of innovation in the Information Technology field, several programs are being requested and assigned to various functions. Therefore, a series of functions can become repetitive, exhausting and, consequently, lead to several problems due to the postures adopted by workers. One problem is known as Repetitive Strain Injury (RSI). Thus, this work aims to develop a solution that makes it possible to avoid or prevent previous ones during the development process, thus creating an AIVOZ platform. AIVOZ allows users to use a voice to create structures and lines of code in the languages HTML and CSS in their web projects. To carry out this research, applications of Automatic Speech Recognition models were studied. The

* Instituto Federal de Educação, Ciência e Tecnologia do Ceará, lucas.marques.bandeira@gmail.com
** Instituto Federal de Educação, Ciência e Tecnologia do Ceará, henrique.viana@ifce.edu.br

project uses a Web Speech API for speech recognition and interpretation, which was designed for speech analysis and synthesis, and it is based on the Hidden Markov Models for capturing and converting the analog to digital signal.

Keywords: 1. Automatic Speech Recognition 2. Web Development 3. API Web Speech.

1 INTRODUÇÃO

O Reconhecimento Automático de Voz é uma área de pesquisa ativa há mais de cinco décadas (YU; DENG, 2016) e sempre foi considerada como uma ponte importante na promoção de uma melhor comunicação. Nos últimos anos, a tecnologia da fala começou a mudar a maneira como se vive e trabalha e se tornou um dos principais meios para os humanos interagirem com alguns dispositivos. Existem muitas aplicações nas quais a tecnologia de fala tem um papel importante. Elas podem ser classificadas como aplicações que ajudam a melhorar a Comunicação Humano-Humano e que ajudam a melhorar a Comunicação Humano-Máquina.

A tecnologia da fala pode remover barreiras na Comunicação Humano-Humano. No passado, pessoas que falavam línguas diferentes precisavam de um intérprete humano para ser capaz de falar um com o outro. Por exemplo, pessoas que não falam chinês costumam ter dificuldade em viajar sozinhos na China. Esta barreira, no entanto, pode ser aliviada por sistemas de tradução de voz por fala (CLAYTON, 2012). A tecnologia de fala também pode ajudar as interações Humano-Humano de outras maneiras. Um sistema unificado de mensagens pode ser usado para converter mensagens de voz em textos. O texto transcrito pode então ser facilmente enviado para o destinatário através de e-mails, mensagens instantâneas ou mensagens curtas.

As tecnologias de fala também podem melhorar muito a Comunicação Humano-Máquina. As aplicações mais populares nesta categoria incluem pesquisa por voz, assistente digital pessoal, jogos, sistemas de interação em sala de estar e sistemas de infoentretenimento em veículos (sistemas que permitem os usuários interagirem com eles por meio da fala para que os usuários possam tocar músicas, pedir informações ou controlar o sistema) (YU; DENG, 2016).

O trabalho em questão foca em desenvolvimento de Comunicação Humano-Máquina, mas especialmente em uma plataforma da categoria de pesquisa por voz. A pesquisa por voz é a tecnologia que fornece aos usuários as informações que eles solicitam uma consulta falada (WANG et al., 2008). A informação normalmente existe em um grande banco de dados, e a consulta deve ser comparada com um campo no banco de dados para obter informações relevantes (YU et al., 2007; ZWEIG et al., 2007). O conteúdo do campo de busca, como informações de negócios ou nomes de produtos, muitas vezes são textos não estruturados.

A pesquisa por voz é uma tecnologia que permite ao usuário utilizar um comando de voz para realizar uma pesquisa em *sites* ou aplicativos. Fruto dos avanços no reconhecimento de fala, esse recurso apareceu pela primeira vez nos *smartphones*, possibilitando a substituição da barra de pesquisa. Hoje, a pesquisa por voz está se tornando cada vez mais uma nova forma de fazer

consultas na *Internet*, evidenciado pelo surgimento de novos produtos como assistentes de voz (Alexa com *Amazon Echo*[1], *Google Voice Search*[2], Cortana[3], Siri[4], etc.).

Ainda em relação à *Internet*, ela segue em constante crescimento e expansão. As criações de *sites* foram evoluindo e sendo requisitadas a todo momento. De acordo com *Internet Live Stats*[5], em 1994 existiam menos de três mil sites na rede mundial. Já no ano de 2014, a estimativa de páginas eram mais de um bilhão em toda rede mundial, ou seja, mais de 33.000.000% (trinta e três milhões por cento) em apenas 20 anos. Este crescimento se deu por vários fatores, sendo alguns deles, o desenvolvimento tecnológico, a disponibilidade em diferentes plataformas, a acessibilidade, entre outros.

Com esta grande demanda, algumas novas tecnologias estão surgindo para facilitar e agilizar o desenvolvimento de aplicações *web*. Vários trabalhos científicos buscaram agregar ou atribuir a implementação do Reconhecimento Automático de Voz para melhorar ou solucionar determinados problemas em desenvolvimento *web* (PELÁEZ-MORENO; GALLARDO-ANTOLÍN; MARÍA, 2001; GOTO; OGATA; ETO, 2007; WRIGLEY; HAIN, 2011; BESACIER et al., 2014; MENDELS et al., 2015). Tendo isso em vista, este trabalho tem como objetivo o desenvolvimento de uma plataforma chamada AIVOZ, capaz de auxiliar no desenvolvimento de aplicações *web*, com foco nas linguagens *HyperText Markup Language* (*HTML*) e *Cascading Style Sheets* (*CSS*).

No desenvolvimento desta plataforma foram utilizadas várias tecnologias. Como trabalho inicial, a plataforma AIVOZ tem como objetivo reconhecer os comandos de *HTML* falados pelo usuário. Para chegar neste objetivo, a interface *Speech Recognition* da *API Web Speech*[6] foi utilizada como captura de voz. O *Codemirror*, um editor de texto implementado em JavaScript, foi a biblioteca responsável pela interação do usuário com código (através dele o usuário pode editar os códigos gerado pela fala). Por fim, o *framework Bootstrap* foi utilizado para a criação e estruturação da página do AIVOZ.

A solução proposta pelo AIVOZ também tem como objetivo evitar ou prevenir possíveis futuras lesões durante o processo de desenvolvimento, diminuindo o uso de teclado e mouse, e a permanência em posições que são inadequadas diante do computador. De acordo com (BORGES, 2018), o mercado brasileiro está se modernizando e resultando em uma taxa crescente no cenário de TI, mas especificamente na área do desenvolvimento de aplicações, entretanto este trabalho se torna repetitivo, exaustivo e proporcionam vários problemas devido as posturas adotadas pelos trabalhadores ocasionando, assim, problemas de lesões que prejudicam ou impossibilitam a continuidade do trabalho. Um desses problemas é conhecido como Lesão por Esforço Repetitivo (LER). O AIVOZ vem então como uma alternativa para diminuir esses problemas.

A fim de descrever a implementação do AIVOZ, o presente trabalho está estruturado

[1] Fonte: https://www.amazon.com.br/b?ie=UTF8&node=19877613011
[2] Fonte: https://voice.google.com/u/0/about
[3] Fonte: https://edu.gcfglobal.org/pt/tudo-sobre-o-windows-10/o-que-e-e-como-funciona-a-cortana/1/
[4] Fonte: https://www.apple.com/br/siri/
[5] Fonte: https://www.internetlivestats.com/total-number-of-websites/
[6] Fonte: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

da seguinte maneira: a Seção 2 revisa todos os conceitos chaves relacionados ao tema de Reconhecimento Automático de Voz. A Seção 3 apresenta alguns trabalhos relacionados à esta pesquisa. A Seção 4 descreve a metodologia utilizada para desenvolvimento da solução proposta. Na Seção 5 são apontados os resultados prévios. Finalmente, na Seção 6 estão contidas as conclusões e os trabalhos futuros.

2 REFERENCIAL TEÓRICO

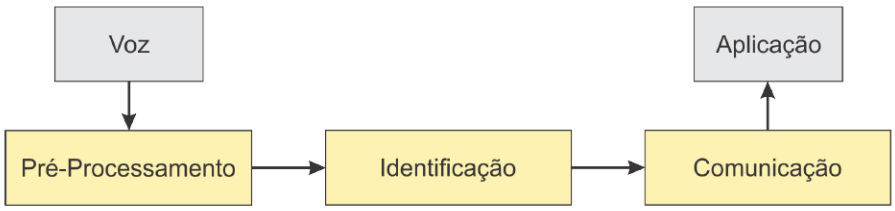
Nesta seção serão apresentados conceitos que se fazem necessários para compreender as definições do Reconhecimento Automático de Voz. Serão apresentados elementos importantes que compõem esse processo, bem como a estrutura básica de um modelo de reconhecimento de voz e o Modelo Oculto de Markov, uma técnica de Aprendizado de Máquina utilizada na área. Por fim, serão abordados conceitos das linguagens utilizadas em desenvolvimento *web*, nesse caso o *HTML*, *CSS* e *JavaScript*.

2.1 Reconhecimento Automático de voz

O Reconhecimento Automático de Voz (em inglês *Automatic Speech Recognition*) é a captura de sinais sonoros analógicos que são convertidos em dados digitais (SILVA, 2010). Os sistemas atuais de Reconhecimento Automático de Voz baseiam-se fundamentalmente em princípios de reconhecimento estatístico de padrões, onde os sinais acústicos são transformados em uma sequência de símbolos, analisados e estruturados, servindo de informação para uma máquina tomar decisões.

Em resumo, sua execução é composta de três tarefas fundamentais que são realizadas como mostrado na Figura 1 (ABREU, 2019).

Figura 1 – Modelo básico do processo Reconhecimento Automático de Voz.



Fonte: (ABREU, 2019).

Neste sistema sua execução ocorre em três etapas: Pré-processamento, Identificação e Comunicação.

1. **Pré-processamento:** Ocorre a conversão A/D (Analógico/digital), filtragem e extração dos parâmetros acústicos. O objetivo da etapa de pré-processamento do sinal é estabelecer um conjunto de parâmetros que contenha informações úteis, para serem usadas na etapa de identificação.

2. **Identificação:** É a etapa principal do processo onde é feito o reconhecimento da informação baseado em representações existentes dos padrões observados. Em seu atual uso, se tem utilizado diferentes técnicas como modelos estatísticos e Aprendizado de Máquina.

3. **Comunicação:** Os resultados são enviados para o ator externo que fará o uso do mesmo para tomar decisões.

Ao se trabalhar com o Aprendizado de Máquina, três paradigmas podem ser escolhidos para fazer a classificação: Aprendizado Supervisionado, Não-Supervisionado e por Reforço (LORENA; CARVALHO, 2003; BONACCORSO, 2017).

No Aprendizado Supervisionado um conjunto de exemplos de treinamento é fornecido nos quais o rótulo da classe designada é conhecido, isto é, as entradas e a saídas são conhecidas. Cada exemplo é composto por um vetor de valores de características ou atributos e o rótulo da classe designada. O algoritmo então tem por objetivo criar um classificador que determine corretamente a classe de novos exemplos que ainda não possuam o rótulo da classe (CARUANA; NICULESCU-MIZIL, 2006).

No Aprendizado Não-Supervisionado a saída não é conhecida. Os exemplos fornecidos são analisados e observados se alguns deles podem ser agrupados de alguma forma. Ocorrendo os agrupamentos, é realizado uma análise para determinar o que cada agrupamento representa no contexto do problema que está sendo analisado (MONARD; BARANAUSKAS, 2003). De uma forma geral, com aprendizado não supervisionado quer achar uma representação mais informativa dos dados que tem.

A terceira abordagem de Aprendizado de Máquina é a chamada aprendizagem por reforço, em que a máquina tenta aprender qual é a melhor ação a ser tomada, dependendo das circunstâncias na qual essa ação será executada (SUTTON; BARTO, 2018). O processo ocorre através de recompensas ou não ao classificador, dependendo do desempenho obtido na aproximação da saída desejada.

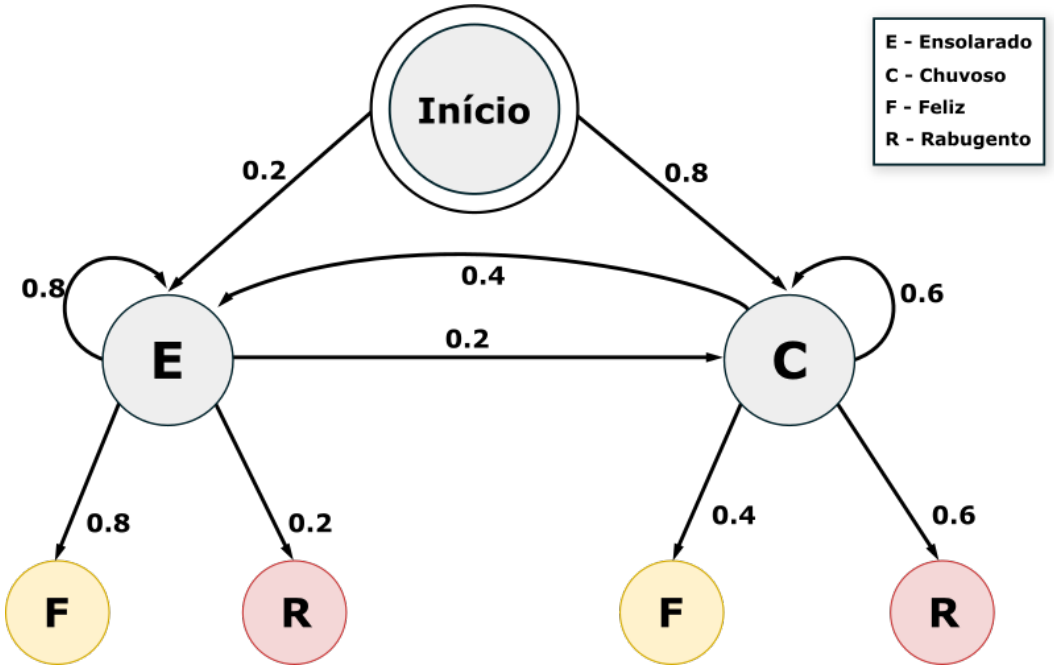
2.2 Modelo Oculto de Markov

No desenvolvimento desse trabalho foi escolhida para a classificação uma técnica de Aprendizado Não-Supervisionado chamada Modelo Oculto de Markov em inglês, Hidden Markov Model. Ele é um modelo estatístico que consiste em parâmetros desconhecidos, que tem como objetivo determiná-los a partir do que foi observado (MILLER; LEEK; SCHWARTZ, 1999; YNOGUTI et al., 1999). Os parâmetros observados são de certa forma independentes, ou seja, o parâmetro é definido pelo estado atual e não à sequência de eventos que ocorreram anteriormente. O Modelo Oculto de Markov inclui dois componentes básicos:

- Uma cadeia de Markov de estados finitos;
- Um conjunto finito de distribuições de probabilidade de saída.

Em geral, os Modelos Ocultos de Markov podem ser considerados como um conjunto de estados ligados por transições com probabilidades associadas a cada transição, como ilustra a Figura 2. O modelo começa com o estado inicial, e em cada passo de tempo, ocorre uma transição a um novo estado, e um símbolo de saída é gerado. A transição e o símbolo de saída são aleatórios, sendo regidos por distribuições de probabilidade.

Figura 2 – Modelo de estados.



Fonte: Elaborada pelos autores.

Pode se considerar, a título de ilustração, o exemplo da Figura 2, que elabora como uma pessoa se sente em climas diferentes. Como elementos dessa cadeia de Markov tem-se: Conjunto de Estados $S = \{Feliz, Rabugento\}$, Conjunto de estados ocultos $Q = \{Ensolarado, Chuvoso\}$, uma série de estados ao longo do tempo $z = \{z_1, z_2, \ldots, z_n\}$, onde $z_i \in S$ para $i \in \{1, \ldots, n\}$. Por exemplo, supondo que se tem uma série de estados observados por quatro dias $z = \{z_1 = Feliz, z_2 = Rabugento, z_3 = Rabugento, z_4 = Feliz\}$.

O sentimento que uma pessoa emite são chamados de observações. O clima que influencia os sentimentos de uma pessoa é chamado de estados ocultos, pois você não pode observá-los. Neste exemplo acima, os sentimentos (Feliz ou Rabugento) podem ser apenas observados. Pode se observar que uma pessoa tem 80% de chance de ser Feliz, visto que o clima no ponto específico de observação (ou melhor, dia neste caso) é ensolarado. Da mesma forma, a chance de 60% de uma pessoa ser rabugenta, visto que o clima é chuvoso. Aqui mencionados 80% e 60% são probabilidades de emissão, uma vez que lidam com observações.

Quando se considera os climas (estados ocultos) que influenciam as observações existem correlações entre dias consecutivos sendo ensolarados ou dias alternados sendo chuvosos. Há 80% para o clima ensolarado ser em dias sucessivos, enquanto 60% de chance para dias consecutivos serem chuvosos. As probabilidades que explicam a transição de/para estados ocultos são

probabilidades de transição.

Descrevendo matematicamente, se tem um conjunto de símbolos distintos observáveis $S = \{S_1, \ldots, S_n\}$ e um conjunto de estados ocultos $Q = \{Q_1, \ldots, Q_m\}$. Dado um tempo t, denota-se o estado no tempo t pelo símbolo q_t. Um Modelo Oculto de Markov básico é constituído de alguns conjuntos de parâmetros principais:

- As probabilidades de emissão de símbolo $B = \{b_{jk}\}$ – a probabilidade da observação S_k ser emitida pelo estado Q_j, i.e., $b_{jk} = P(S_k \text{ em } t \mid q_t = Q_j), 1 \leq j \leq m$ e $1 \leq k \leq n$.
- As probabilidades de transição de estado $A = \{a_{ij}\}$ – a probabilidade de estar no estado Q_j no instante de tempo subsequente dado que o estado atual é Q_i, i.e., $a_{ij} = P(q_{t+1} = Q_j \mid q_t = Q_i), 1 \leq i, j \leq m$.
- A distribuição de probabilidade a priori $\pi = \{\pi_i\}$ do sistema estar em um dado estado Q_i no instante inicial de tempo, i.e., $\pi_i = P(q_1 = Q_i), 1 \leq i \leq m$.

Logo, denota-se um modelo Modelo Oculto de Markov M por $M = (S, Q, A, B, \pi)$. Duas questões são principais dado um modelo M (LEITE, 2008). Para isso, considere uma sequência de observações $z = \{z_1, \ldots, z_l\}$, onde $z_i \in S$ e $i \in \{1, \ldots, l\}$.

1. Qual é a probabilidade de uma sequência observada? Isto é, considere uma sequência de observações $z = \{z_1, \ldots, z_n\}$. Dado o modelo M calcular a probabilidade de que o modelo gere a sequência de observações z (ou seja, $P(z \mid M)$).
2. Estimar os parâmetros de probabilidade de emissão (B) e transição de estados (A) do modelo M de modo que a probabilidade da sequência de observações $z = \{z_1, \ldots, z_n\}$ seja maximizada, i.e., $P(z \mid M)$ seja maximizada.

Em relação ao Reconhecimento Automático de Voz, os modelos ocultos de Markov modelam o processo de geração da voz através da máquina de estados estocástica. Os sons da voz, ou parâmetros derivados a partir desses sons, são gerados segundo as funções de distribuição de probabilidades correspondentes a cada estado. Este método fornece uma maneira natural e altamente confiável de reconhecer a voz em diversas aplicações práticas.

2.3 Desenvolvimento *Web*

O Desenvolvimento *web* é a criação e estruturação de páginas ou estruturas que são disponibilizadas em meio à *internet* (NORTHWOOD, 2018), onde seu intuito principal é de transmitir, informar e espalhar conteúdos de diversas áreas. As aplicações do desenvolvimento *web* são fundamentadas em tecnologias capazes de estruturar, estilizar e criar funções lógicas que são executadas dentro das páginas *web*. Os pilares do desenvolvimento *web* são: o *HTML* (linguagem de estruturação), *CSS* (linguagem de formatação) e Linguagens de Programação (como o *JavaScript*, *Python*, *Ruby*, *PHP*, etc.).

2.3.1 HTML

O *HTML* em inglês, *Hypertext Markup Language* é uma linguagem de marcação de hipertextos que auxilia na criação de páginas *web* (TEAM, 2011). A sua principal característica é a possibilidade de interligar um documento *web*, por meio de marcadores, que são conhecidos como *tags*, a outros documentos da *web* (SILVA, 2008). A partir dos marcadores podem ser criados atributos, agrupadores, identificadores ou classes de elementos, permitindo assim uma separação entre os conteúdos dentro da sua estrutura, bem como imagens, vídeos, textos, tabelas, entre outros. Todos esses componentes são responsáveis por uma vasta gama de ações e formatações em um documento *HTML*. A seguir serão apresentadas algumas informações básicas acerca desses componentes.

Os marcadores têm a função de interligar, por meio de estruturas formatadas as informações em uma página *web* seguindo algumas regras (SILVA, 2008). Eles sempre aparecem precedidos de "<" e sucedido de ">". Entre as delimitações existe o nome do marcador que caracteriza uma regra imposta. Por exemplo, o marcador para delimitar um parágrafo é o :

Texto a ser formatado em um parágrafo.

Considerando o exemplo acima, o é considerado um tipo de marcador que está condicionado a definir uma determinada regra ao Texto que está dentro dele (ROBBINS, 2012). Para fechamento da marcação, é utilizado o que finaliza até onde o marcador irá ser aplicado. Os marcadores podem ser abertos, como por exemplo o "
" (que realiza uma quebra de linha na página) e não necessitam estar em pares, ou fechados, como o que acontece em " Texto " (SCHMIDT, 2015). A Figura 3 apresenta uma lista dos principais marcadores HTML encontrados, que compreendem especialmente marcação e formatação de textos, imagens, listas, tabelas, links, formulários e vídeos.

2.3.2 Estrutura de uma página HTML

Geralmente uma estruturação *HTML* começa identificando que tipo de arquivo é, no caso isso é definido através do marcador "<!DOCTYPE html>". Logo em seguida vem a abertura da estrutura *HTML* definida por "<html>" e seu fechamento "</html>" na última linha do arquivo (ROBBINS, 2012).

O escopo da estrutura *HTML* é delimitado em duas partes: o cabeçalho e o corpo. O marcador "<head>" define o cabeçalho da página, onde serão importadas algumas bibliotecas, definição de identificação do local do arquivo CSS, título da página, ícone da página, entre outras informações que serão carregadas. Após o seu fechamento, tem-se o marcador "<body>" que é responsável pelo corpo da página, sendo o principal marcador da página. É nesta região que todo o conteúdo de uma página *web* será descrito, ou seja, onde todas as informações das páginas serão descritas, marcadas e formatadas pelas *tags*. A seguir um exemplo de um simmples código em *HTML*:

Figura 3 – Principais Marcadores *HTML*.

- <!-- -->
- <!DOCTYPE>
- <a>
- <applet>
- <area>
- <audio>
- <b>
- <base>
- <big>
- <blockquote>
- <body>
-
- <button>
-
- <code>
- <del>
-
-
- <embed>
- <font>
- <form>
- <frame>
- <frameset>
- <h1>
- <h2>
- <h3>
- <h4>
- <h5>
- <h6>
- <head>
- <hr>
- <html>
- <i>
- <iframe>
- <img>
- <ins>
-
- <map>
- <meta>
- <noframes>
- <noscript>
- <object>
-
- <p>
- <param>
- <pre>
- <q>
- <s>
- <samp>
- <script>
- <small>
- <video>
- <title>
-
- <strike>
-
- <style>
- <sub>
- <sup>
- <table>
- <tbody>
- <tfoot>
- <td>
- <thead>
- <th>
- <tr>
- <u>
- <var>
- <wbr>

Fonte: (SCHMIDT, 2015).

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Exemplo de título</title>
5      </head>
6      <body>
7          <p> Olá Mundo do HTML! </p>
8          <img src="imagem.jpg" alt="Minha Figura">
9      </body>
10 </html>
```

Para a customização dos conteúdos presentes em uma página *HTML*, existem três formas de estilizações presente, são elas: *Inline Style*, *Internal Style Sheet* e *External Style Sheet*.

- ***Inline Style***: é a forma mais simples de se adicionar uma propriedade à uma *tag*. Ela se baseia no fato de adicionar a propriedade na própria estrutura *HTML* conforme o exemplo abaixo (que está mudando a cor do texto do parágrafo para verde), entretanto não se é recomendado devido a difícil manutenção.

```
1 <p style="color: green;">
2     Olá Mundo!
3 </p>
```

- ***Internal Style Sheet***: O conteúdo da estilização é criado a partir de um marcador chamado <style> dentro de um arquivo *HTML*, na região do cabeçalho do arquivo. A seguir um exemplo.

```
<head>
    <style>
        <p style="color:green;">
             Olá Mundo!
        </p>
    </style>
</head>
```

- ***External Style Sheet***: a estilização é adicionada a um arquivo externo através de um arquivo *CSS* e ser importado conforme o exemplo abaixo. Essa é a forma mais recomendada de estilização, por conta da manutenção e da organização.

```
<head>
    <title> Exemplo de Site </title>
    <link rel="stylesheet" type="text/css "href="style.css">
</head>
```

2.4 *CSS*

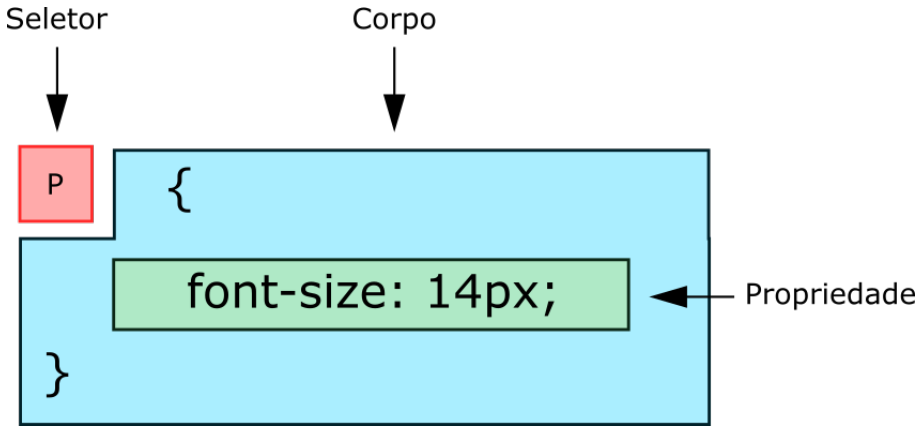
O *Cascading Style Sheets* (ou Folhas de Estilo em Cascata), também conhecido por CSS, é o padrão para definir a apresentação de documentos escritos em *HTML* (ROBBINS, 2012). A apresentação ou estilização, se refere à forma de como o documento é entregue ao usuário, mostrando-se em um navegador através de um computador ou *Smartphone*. De forma prática, ela funciona como uma camada de personalização ao conteúdo descrito pela página *HTML*.

O *CSS* é um código em que você pode fazer alterações rápidas de *layout*, como definição de cores e fontes, por exemplo. Essa camada proporciona não apenas a facilidade de personalização, como também ajuda a diminuir a repetição de conteúdo na estrutura do código *HTML*. Algumas das propriedades que ele permite são:

- manter um padrão de formatação em diferentes navegadores;
- controle de *layout* em apenas uma folha;
- criação de formatações com designs mais responsivos, pensando em usabilidade e experiência do usuário.

A sintaxe do *CSS* é composta de dois elementos principais, o seletor e o corpo da declaração. O seletor é a parte onde se define a qual/quais elementos (tags definidas no arquivo *HTML*) serão aplicados o estilo criado. O corpo da declaração é a parte onde se informa como os elementos referenciados pelo seletor devem ser modificados (suas propriedades). A Figura 4

Figura 4 – Estrutura do *CSS*.



Fonte: Elaborada pelos autores.

exemplifica a estruturação de uma estilização do marcador <p>, onde o parágrafo está sendo modificado com um tamanho de fonte de 40 *pixels*.

As propriedades dos seletores em *CSS* abrangem uma série de configurações, como alterações em fonte, cores, espaçamentos, bordas, alinhamento, etc. Uma lista das propriedades mais conhecidas, que podem ser aplicadas em diferentes seletores, pode ser vista na Figura 5.

Figura 5 – Exemplos de Atributos do *CSS*.

- background
- border-bottom-color
- border-left
- border-right
- border-style
- border-width
- font-family
- font-weight
- list-style-type
- text-indent
- background-color
- border-bottom-style
- border-left-color
- border-right-color
- border
- border-bottom-width
- border-left-style
- border-right-style
- border-top
- color
- font-size
- height
- table-layout
- text-transform
- border-top-color
- display
- font-style
- letter-spacing
- text-align
- vertical-align
- border-bottom
- border-color
- border-left-width
- border-right-width
- border-top-width
- font
- font-variant
- line-height
- text-decoration
- width

Fonte: (SCHMIDT, 2015).

3 TRABALHOS RELACIONADOS

Nesta seção serão apresentadas algumas aplicações que têm como principal foco o Reconhecimento Automático de Voz e que colaboraram com o estudo deste trabalho.

3.1 *Serenade*

O *Serenade* [7] é uma *API* (*Application Programming Interface*) poderosa que através do Reconhecimento Automático de Voz pode gerar determinadas funcionalidades nos navegadores ou em editores de texto. Ele tem um mecanismo de voz para texto desenvolvido especificamente para código, ao contrário da *API* de voz para texto do Google, que é projetada para conversação.

O *Serenade* atua um pouco como um assistente digital, permitindo que você descreva os comandos que está codificando, sem exigir que você necessariamente dite cada instrução palavra por palavra. Depois que um usuário fala o código, os modelos de aprendizado de máquina, em sua camada de processamento de linguagem natural, são treinados para identificar e traduzir construções de programação comuns em código sintaticamente válido.

Esta *API* é capaz de se incorporar em diferentes ferramentas dentro do sistema *Windows*. Por exemplo, ele pode ser usado em *plugins* como o *Atom*, *Vs Code*, *JetBrains*, *Google Chrome* e *Microsoft Edge*; em aplicações e *websites* mesmo sem *plugins* dedicados, como no *Jupyter*, *Slack*, *Discord*, *GitHub*, *JIRA*, *GitLab*, *Gmail* e *Linear*; e em linguagens de programação como *Python*, *JavaScript*, *HTML*, *Java*, C/C++ e *TypeScript*. Atualmente, a aplicação está voltada somente para sistemas *Windows* e necessita da instalação para utilizar todas as suas funcionalidades.

3.2 *Talon*

O *Talon* [8] visa trazer proficiência em programação e utilização de computadores *desktop* para pessoas que possuem algum tipo de dificuldade motora, tentando assim, melhorar a produtividade e possibilita qualquer pessoa a poder usar um computador. Ele permite que as pessoas usem um computador com a voz como método de entrada. Além da voz, ele suporta controles usando expressões faciais e rastreamento ocular.

Os recursos do *Talon* são personalizáveis e extensíveis por meio de uma *API* de *script* que usa *Python* em conjunto com uma linguagem de *script* mais simples e específica para o domínio chamada TalonScript (BERGE, 2021). Por meio desta *API*, os usuários podem definir ações personalizadas que respondem a determinados comandos de voz ou para onde o usuário está olhando. Essas ações podem ser muito simples, como emular uma sequência de teclas, mas também pode ser funções *Python* arbitrárias que podem enviar comandos para o funcionamento do sistema ou fazer solicitações de rede.

3.3 *VoiceCode*

VoiceCode [9] é uma plataforma de controle de voz avançada que utiliza o seu computador em tempo real. Ela faz desde escrever qualquer tipo de texto em aplicativos, *e-mails* e códigos, alternar aplicativos ou navegar em programas como o *Photoshop*, ou seja, o *VoiceCode* faz o trabalho mais rápido e fácil. Ele se destaca por seus comandos, onde podem ser encadeados e

[7] Fonte: https://serenade.ai/
[8] Fonte: https://talonvoice.com/
[9] Fonte: https://www.voicecode.io/

alinhados em qualquer combinação, permitindo que ações complexas sejam realizadas por uma única frase falada. Tirando proveito da aptidão natural de seu cérebro para a linguagem, você pode controlar seu computador de maneira mais eficiente e natural (DéSILETS; FOX; NORTON, 2006).

3.4 Comparativo entre os Trabalhos Relacionados e o AIVOZ

Estes projetos nasceram principalmente devido às lesões ocasionadas por uso excessivo do teclado, também conhecidas por Lesões de Esforço Repetitivo ou LER. Apesar da motivação comum dos projetos eles possuem algumas particularidades. A Tabela 1 mostra as principais características e comparações das plataformas que colaboraram para o desenvolvimento deste projeto, junto com as características presente no AIVOZ.

Tabela 1 – Tabela Comparativa entre as Aplicações.

Plataformas	Idiomas	Tecnologias	Sistemas	Disponibilidade
Serenade	Inglês	*JavaScript*, *TypeScript*, *Java*, C/C++,*Dart*	Windows	Híbrido
Talon	Inglês	*Python*	Windows, Linux e MacOS	Gratuito
VoiceCode	Inglês	*JavaScript*	Windows, Linux e MacOS	Gratuito
AIVOZ	Português	HTML e CSS	*Web*	Gratuito

Fonte: Elaborada pelos autores.

Apesar dos trabalhos terem suas grandes contribuições e uma vasta linguagem explorada, ainda há uma carência de soluções que facilitem no auxilio no desenvolvimento *web* utilizando a linguagem Português brasileiro (PT-BR). O trabalho proposto, última linha da tabela 1, oferece uma maneira mais abragente que utiliza da linguagem Português brasileiro (PT-BR) e de forma totalmente *online*, sem necessidade de instalações ou de pagamentos para usufruir dos benefícios propostos.

4 PROPOSTA

Nesta seção será apresentada a proposta deste trabalho, que consiste no desenvolvimento do AIVOZ, um sistema que auxilia na construção e desenvolvimento de aplicações *web* por meio da voz de forma totalmente *online*. A metodologia adotada para o desenvolvimento da proposta é apresentada em quatro etapas. Todas as etapas descrevem e ilustram toda a construção da plataforma, que consistem em Motivação, Definição das Tecnologias, AIVOZ e Resultados.

4.1 Motivação

Como dito anteriormente, A LER é uma síndrome que afeta músculos, nervos e tendões dos membros superiores principalmente (BARBOSA; SANTOS; TREZZA, 2007). Os principais

sintomas de LER são dor nos membros superiores e nos dedos, dificuldade para movimentá-los, formigamento, fadiga muscular, alteração da temperatura e da sensibilidade, redução na amplitude do movimento e inflamação. Na maioria das vezes, esses sintomas estão relacionados com uma atividade inadequada não só dos membros superiores, mas de todo o corpo. Por exemplo, se a pessoa ficar sentada diante do computador digitando no teclado por horas seguidas.

A partir desta problemática, foi pensada uma solução que possibilita evitar ou prevenir possíveis futuras lesões durante o processo de desenvolvimento, dando origem ao AIVOZ. Ele se trata de uma plataforma que proporciona ao desenvolvedor um intermédio de auxilio que, por meio da voz, o ajude no desenvolvimento de aplicações *web*, diminuindo o uso de teclados e mouse, e consequentemente a permanência em posições que são inadequadas diante do computador.

4.2 Definição das Tecnologias

Nesta etapa serão apresentadas as tecnologias que levaram ao desenvolvimento de toda estrutura do AIVOZ.

4.2.1 Bootstrap

O projeto proposto utilizou o *Bootstrap* [10] como *Front-End* pela sua facilidade e tecnologias de domínio do criador. O design de *website* é um dos o principais fatores que devem ser planejados adequadamente para que o *site* funcione bem e que possa ser acessado por vários navegadores, bem como ser executado em várias plataformas.

O *Bootstrap* é um *framework* para criar aplicações da *web* de forma rápida, fácil e gratuita. *Bootstrap* consiste da linguagens *HTML*, *CSS* e *JavaScript* para gerar *Grids*, *Layout*, Tipografias, Tabelas, Formulários, Navegação e outros. Com a ajuda do *Bootstrap*, pode-se criar sites responsivos com rapidez e facilidade e que podem funcionar perfeitamente em navegadores populares como *Google Chrome*, *Mozila Firefox*, *Safari*, *Opera* e *Internet Explorer*. Atualmente o *Bootstrap* se encontra na sua versão estável v5.1.3.

4.2.2 Web Speech API

A *Web Speech API* [11] foi projetada para análise e síntese de fala. Ele permite que os usuários enviem entradas de voz para aplicações na *web*. Em seguida, as aplicações da *web* utilizarão a *Web Speech API* para transformar a fala em texto. Existem dois componentes principais para esta *API*, que abrem novas possibilidades interessantes de acessibilidade e mecanismos de controle:

- O Reconhecimento Automático de Voz é acessado por meio da interface *SpeechRecognition*, que fornece a capacidade de reconhecer o contexto de voz de uma entrada de áudio

[10] Fonte: https://getbootstrap.com//
[11] Fonte: https://developer.mozilla.org/enUS/docs/Web/API/Web_Speech_API

(normalmente por meio do serviço de reconhecimento de voz padrão do dispositivo) e responder adequadamente. A interface *SpeechGrammar* representa um contêiner para um determinado conjunto de gramáticas que o aplicativo deverá reconhecer. Por exemplo, no AIVOZ as gramáticas se tratam de palavras-chave que correspondem aos comandos de voz e aos elementos das linguagens *HTML* e *CSS*.

- A síntese de voz é acessada por meio da interface *SpeechSynthesis*, um componente que permite que os programas leiam seu conteúdo de texto (normalmente por meio do sintetizador de voz padrão do dispositivo).

4.2.3 CodeMirror

CodeMirror [12] é um editor de texto versátil implementado em *JavaScript* para o navegador. Ele é especializado na edição de código e vem com vários modos de linguagem e complementos que implementam funcionalidades de edições mais avançadas. O *CodeMirror* é um projeto de código aberto compartilhado sob uma licença do Instituto de Tecnologia de Massachusetts (*MIT*). É o editor usado nas ferramentas de desenvolvimento para *Firefox*, *Chrome* e *Safari*, no *Light Table*, *Adobe Brackets*, *Bitbucket* e muitos outros projetos.

4.3 AIVOZ

O AIVOZ [13] é uma plataforma *web* que propõe praticidade, acessibilidade e eficiência na vida dos desenvolvedores. Ela permite que os usuários possam utilizar a voz para criar estruturas *HTML* e *CSS* em linhas de códigos em seus projetos.

A plataforma é composta por três elementos principais, sendo eles: Usuário, Sistema e Funções Analisadoras (Figura 6). O Usuário interage com o Sistema através da voz e escrita. O Sistema por sua vez está implementado no *framework Bootstrap* e está divido em dois componentes: um componente para captar voz e editar o código-fonte. As Funções Analisadoras são compostas pela *Web Speech API* e o *CodeMirror* e são responsáveis tanto pelo tratamento dos comandos de voz e da edição de texto pelo usuário. A *Web Speech API* possui uma série de funções que tem como objetivo treinar, analisar e reconhecer as gramáticas criadas. A partir dessas funções serão gerados os códigos-fonte em *HTML* e *CSS*. Por fim, o *CodeMirror* é responsável por retornar para o usuário de forma textual os códigos-fonte representados pelos comandos ditos pelo usuário e também permitir que o mesmo possa editá-lo.

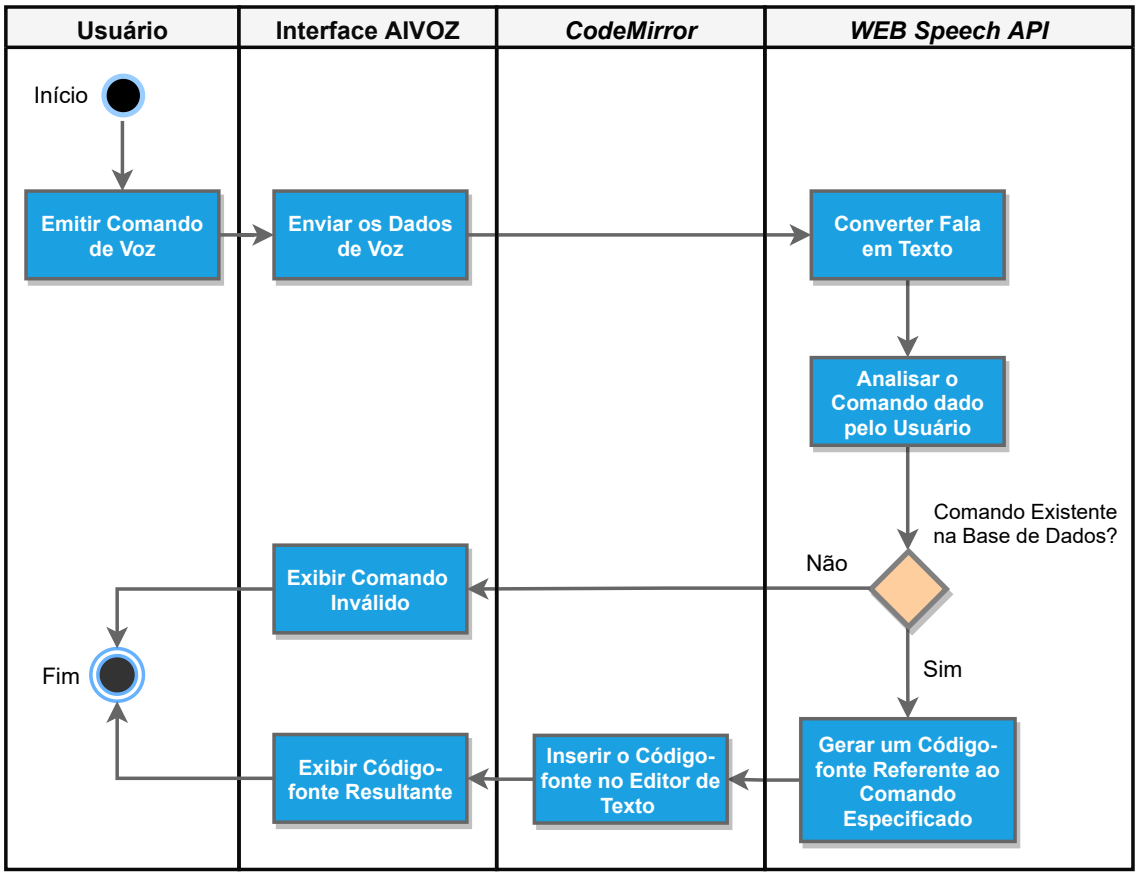
Na Figura 7 é mostrada a estrutura principal de funcionamento da plataforma AIVOZ. O usuário interage com o sistema emitindo um comando de voz. Após o comando ser efetuado, a *Web Speech API* é responsável pelo reconhecimento da fala do usuário. Caso o comando falado seja uma das funções implementadas no AIVOZ, a plataforma retornará a função chamada em linhas de código na interface da aplicação, mas especificamente no editor de texto do *CodeMirror*. Então o usuário poderá visualizá-la, podendo editá-la ou substituí-la.

[12] Fonte: https://codemirror.net/
[13] Fonte: https://lucasmarques2020.github.io/AIVOZTESTE/index.html

Figura 6 – Estrutura de Funcionamento do AIVOZ.

Fonte: Elaborada pelos autores.

Figura 7 – Fluxo do Geração de Código atrávez do AIVOZ.



Fonte: Elaborada pelos autores.

Como dito, a plataforma está divida em dois componentes: o primeiro componente está disposto na área esquerda da tela e é onde se encontra o editor (Figura 8), em que o usuário pode interagir por voz, mouse e teclado. Nela também se encontram quatro botões importantes. O primeiro é botão "*play*" executa o código ou atualiza o mesmo. O segundo botão, logo em seguida, é o botão de *download*, onde o usuário pode baixar o código em sua máquina já que o AIVOZ ainda não conta com um sistema de armazenamento em nuvem. O terceiro botão corresponde ao botão de fala, onde o usuário interage com o AIVOZ, assim, adicionando códigos por comandos de voz. O quarto e último botão, logo abaixo, é o botão que exibe o histórico de todos os comandos emitidos pelo usuário.

Figura 8 – Tela de Codificação do AIVOZ.

Fonte: Elaborada pelos autores.

O segundo componente se trata da área de demonstração *web* resultante aos códigos executados pelo desenvolvedor. Nela o desenvolvedor pode visualizar em tempo real os resultados obtidos pelos códigos falados ou digitados. Caso o usuário esqueça quaisquer comandos, basta ir no botão logo acima do visualizador e clicar, assim sendo encaminhado para a Documentação.

A *Web Speech API* desempenha o trabalho mais importante de toda plataforma. Ela é responsável por identificar o idioma e a palavra falada pelo usuário, onde é convertida e analisada pelas funções presentes na plataforma. Esse processo é apresentado no próprio site da plataforma. A função do código responsável usado nesse trabalho está a seguir:

```
(function() {
let speakBtn = document.querySelector('#speakbt');
let resultSpeaker = document.querySelector('#resultSpeak');

if (window.SpeechRecognition || window.webkitSpeechRecognition) {
   let SpeechRecognition = SpeechRecognition || webkitSpeechRecognition;
   let myRecognition = new SpeechRecognition();
    myRecognition.lang = 'pt-BR';

      speakBtn.addEventListener('click', function() {
        try {
            myRecognition.start();
            resultSpeaker.innerHTML = "Estou te ouvindo...";

        } catch (erro) {
                alert('erro: ' + erro.message);
        }

    }, false);
```

São criadas duas variáveis referentes ao clique do botão (*speakBtn*) e ao que é dito (*resultSpeaker*). Então é feita a verificação se a biblioteca da *Web Speech API* está presente. Então, as mesmas são instanciadas e podendo atribuir algumas especificações como idioma, tempo de execução e duração do tempo entre as palavras. Neste caso, só foi adicionado o idioma PT-BR. Logo em seguida são criados dois tratamentos referente ao botão, um para o evento clique e outro para um evento de erro ou exceção. O evento clique é executado quando o usuário clica no botão de voz e o sistema concede permissão através do evento *Listener*, onde então o usuário poderá interagir por voz. Já o evento de erro acontece quando ocorre algum problema, podendo ser enquadrado como navegador não compatível; bloqueio de microfone por padrão;

alguma extensão conflitante; mais de um clique executado pelo usuário; ou modificação no código-fonte.

Após o usuário emitir o comando de voz, é executada a função de reconhecimento. O código a seguir ilustra como é feito o tratamento e é condicionado para a criação das linhas de comandos e retornado para o usuário. O exemplo mostra a criação de uma *tag* do *HTML*.

```
1  function recur(resultSpeak,flag) {
2      let Vetor = resultSpeak.toLowerCase().split(' ');
3      let r = '';
4      let aux = '';
5      let auxValor = [];
6      let auxAtr = [];
7      let cont = 0;
8      if(Vetor[0] == 'criar' || Vetor[0] == 'Criar'|| Vetor[0] == 'cria'||
           Vetor[0] == 'criai'){
9      for (let i = Vetor.length-1; i >= 1; i--) {
10         //comando tags
11         if (Vetor[i] == 'div' || Vetor[i] == 'divi') {
12             r = '<div' + aux + '>\n\t' + r + '\n</div>';
13             aux = '';
14         }
15         }
```

A função *recur* é onde está toda a lógica da construção do código. Primeiro a frase é falada pelo usuário (no caso para inserir a *tag* , o usuário deverá falar "criar div") e é dividida em um *array* de palavras, então é feito um laço de repetição percorrendo esse *array*. Cada palavra analisada gera um bloco de código *HTML* de acordo com a *tag* ou atributo que a palavra representa. No fim é retornado a *string* com todo o bloco de código *HTML* que foi representado na fala. Pode ser criado de forma recursiva, dando um espaço de tempo pequeno entre as palavras.

Como já dito, o AIVOZ utiliza a Linguagem PT-BR como gramática para acionar as funções. Entretanto, a linguagem adotada dispõe de alguns problemas que podem atrapalhar no momento da captura da voz, como por exemplo vícios de linguagens, regionalismo, microfone com baixo desempenho, comandos extensos, homônimos, etc. Para evitar esses tipos de problemas, foram testadas e treinadas previamente dentro do *Web Speech API*, que tipos de palavras poderiam ser adicionadas nos comandos que não levem aos problemas citados acima.

Por exemplo, ao emitir o comando "criar div" o *Web Speech API* interpreta de duas maneiras, uma sendo "criar divi" e "criar div". De qualquer maneira o código é feito para tratar esses casos específicos. Logo, todos os comandos criados foram pensados de modo que não gerem o menor tipo de inconsistência/interpretações possíveis. Toda a gramática do AIVOZ está disponibilizada na documentação da plataforma, onde alerta ao usuário de todos os possíveis problemas que podem sofrer durante a execução.

Além das *tags*, também existe o tratamento para adicionar um atributo a uma *tag*. Para acionar o atributo, o usuário deve falar "com" e logo em seguida especificar qual tipo de atributo

vai ser destinado aquela *tag*. Por exemplo, é possível adicionar uma tag div com atributo de classe através do comando falado “criar div com classe ’container’. O código a seguir ilustra a lógica para esse tratamento.

```
//comando auxiliar
        else if (Vetor[i] == 'com') {
            for (let j = 0; j < cont; j++) {
                aux += ' ' + auxAtr[j] + '="' + auxValor[j] + '"';
            }
            cont = 0;
            auxValor = [];
            auxAtr = [];
        }
        //comando atributo
        else if (Vetor[i] == 'classe') {
            auxAtr[cont] = 'class';
            cont++;
        }
```

Após o Usuário emitir os comandos de voz e o *Web Speech API* tratá-los, é hora de retornar para o *CodeMirror*, as linhas de códigos-fonte referentes aos comandos pronunciados. A função responsável para retornar se encontra a seguir:

```
    default:
            editor.replaceSelection(recur(resultSpeak,true));
            break;
```

Para que o *CodeMirror* atenda às necessidades do projeto, é necessário a importação das bibliotecas referentes às linguagens *HTML* e *CSS*. Isso é importante pois faz com que o editor de texto exiba os códigos-fonte dessas linguagens de forma funcional para o Usuário. Foram feitas as seguintes importações após baixar a biblioteca:

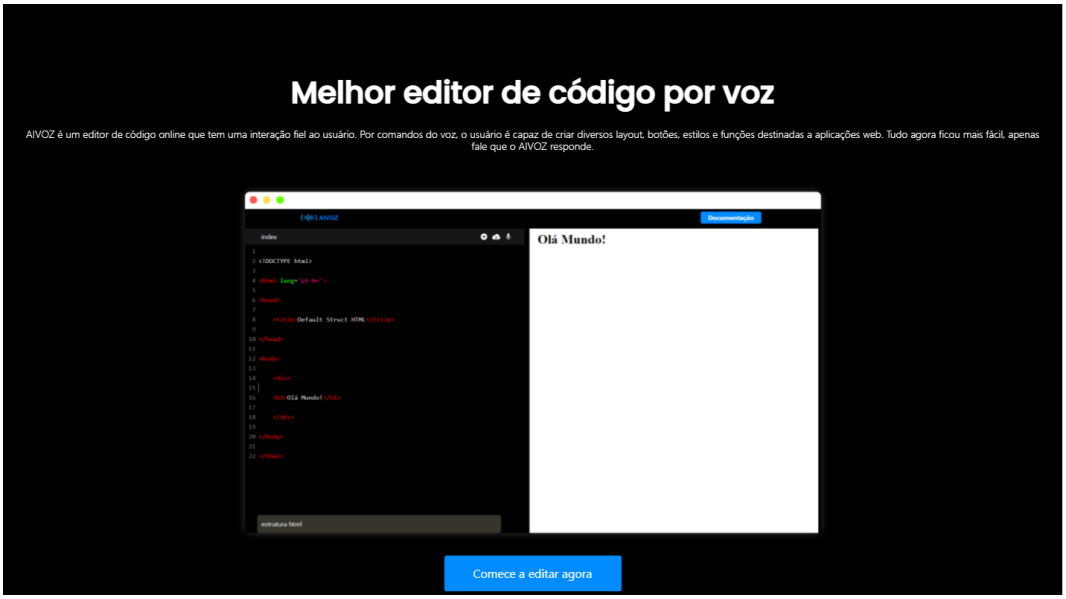
```
    <script src="codemirror/mode/javascript/javascript.js"></script>
    <script src="codemirror/mode/css/css.js"></script>
    <script src="codemirror/addon/hint/xml-hint.js"></script>
    <script src="codemirror/mode/xml/xml.js"></script>
    <script src="codemirror/mode/htmlmixed/htmlmixed.js"></script>
    <script src="codemirror/addon/hint/html-hint.js"></script>
    <script src="codemirror/addon/hint/show-hint.js"></script>
    <script src="codemirror/addon/hint/css-hint.js"></script>
```

5 RESULTADOS

Em sua tela inicial, o AIVOZ foi pensado em um *Layout* mais simples a fim de facilitar a vida do desenvolver, como mostrado na Figura 9. No canto superior direito da tela contém *Menu* principal. No lado esquerdo, é possível acessar a documentação do AIVOZ, onde são encontrados todas as informações pertinentes a plataforma, bem como: Recomendações, Motivação,

Linguagens Suportadas, Palavras-Chave, *Links* das Bibliotecas e Descrição do Autor. Na parte central são exibidas algumas informações promocionais da plataforma e o botão que direciona para a área de codificação.

Figura 9 – Tela Inicial AIVOZ.



Fonte: Elaborada pelos autores.

A Figura 10 contém a documentação do AIVOZ. Como já dito, ela contém todas as informações pertinentes as bibliotecas usadas, comandos e suas pronúncias, sobre o autor e demonstrações visual. No lado esquerdo, encontra-se o *Menu* navegável, onde o usuário poderá clicar na informação que deseja acessar. No lado direito encontram-se as informações referentes ao *menu*.

Figura 10 – Tela Documentação AIVOZ.

Fonte: Elaborada pelos autores.

Foram realizados estudo com a finalidade de avaliar o desempenho da plataforma entre estudantes e desenvolvedores. A duração desta avaliação foi de uma semana, começando dia 15 de novembro à 21 de novembro de 2021. Os critérios para as avaliações foram duas: A primeira foi disponibilizar a plataforma para teste dos usuários e a segunda foi avaliar através de um formulário criado no *Google Forms*. As questões aplicadas podem ser visualizados a seguir:

Questionário de Avaliação do AIVOZ

Q1. Acho que gostaria de usar este sistema com frequência.

Q2. Achei o sistema desnecessariamente complexo.

Q3. Achei o sistema fácil de usar.

Q4. Achei que seria necessário o apoio de um técnico para poder usar este sistema.

Q5. As funções deste sistema estavam bem integradas.

Q6. Achei este sistema muito inconsistente.

Q7. Imagino que a maioria das pessoas aprenderiam a usar este sistema rapidamente.

Q8. Achei o sistema muito complicado de usar.

Q9. Eu me senti muito confiante com o sistema.

Q10. Eu preciso aprender um monte de coisas antes de continuar usando este sistema.

Q11. Eu me senti confortável com este sistema.

Q12. Foi fácil encontrar a informação que eu precisava.

Q13. Eu gostei de usar a interface do sistema.

Q14. A interface do sistema é agradável.

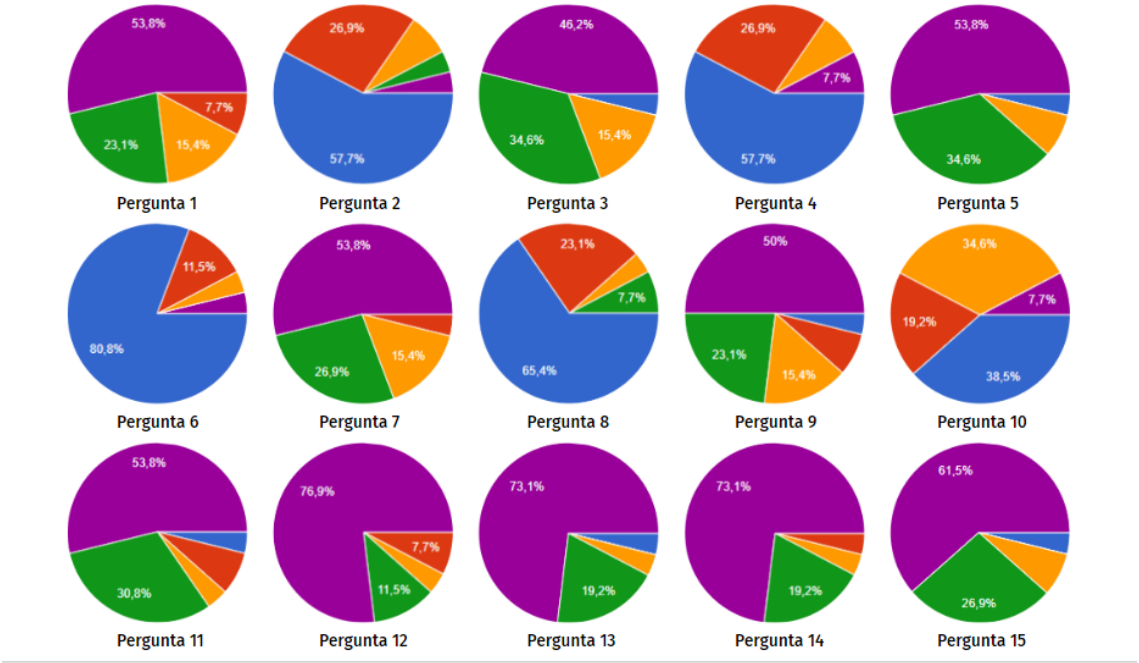
Q15. A organização de informações na tela do sistema é clara.

A plataforma e o formulário foram compartilhadas em grupo oficiais e grupos de desenvolvedores, incluindo uma mensagem de convite para testar e avaliar a plataforma. A imagem 11 mostra os resultados com a usabilidade da plataforma aos usuários que testaram.

A imagem 11 mostra os resultados obtidos de 26 usuários que avaliaram o sistema AIVOZ. O questionário foi utilizado para verificar os pontos fortes da plataforma. Dentre as questões avaliadas as que mais se destacaram de forma satisfatória para todos que avaliaram foram: Q1. Acho que gostaria de usar este sistema com frequência; Q3. Achei o sistema fácil de usar; Q5. As funções deste sistema estavam bem integradas; Q12. Foi fácil encontrar a informação que eu precisava; A interface do sistema é agradável. Portanto, nota-se que a plataforma conseguiu

atingir de forma satisfatória todas as necessidades dos usuários assim também como do trabalho proposto.

Figura 11 – Gráficos Específicos de Cada Questão.



Fonte: Elaborada pelos autores.

6 CONCLUSÕES

Este trabalho apresentou o AIVOZ, um sistema *web* que visa auxiliar no desenvolvimento por intermédio da voz. Seu objetivo é buscar evitar lesões que atingem muitas pessoas por conta de esforço repetitivo durante o trabalho ou no dia a dia. Espera-se que o sistema possa auxiliar diversos programadores que compartilham ou queiram evitar lesões ou, simplesmente, priorizem por acessibilidade. A plataforma está totalmente desenvolvido na linguagem PT-BR e os comandos também são destinados a este idioma visando facilitar mais ainda a vida dos desenvolvedores.

Os resultados apresentados neste trabalho são promissores. O sistema já está em execução e sendo utilizado por alunos e ex-alunos do IFCE, Campus Aracati. Dessa forma, espera-se auxiliar no desenvolvimento de sistemas *web* trazendo mais facilidade por intermédio da voz. A plataforma AIVOZ é totalmente *online* e gratuita, podendo ser acessada por todos. Vale salientar também que com os resultados obtidos no estudo de caso, o AIVOZ se mostrou um sistema capaz de cumprir os objetivos propostos.

Uma das dificuldades encontradas durante o desenvolvimento da aplicação é que a gramática PT-BR é muito vasta, e podem ocorrer muitos problemas linguísticos que ocasionam em problemas de compreensão de fala ao se comunicar com a plataforma. Por não conter um *Back-End*, o AIVOZ não consegue tratar todas as palavras. Por isso é difícil tratar todos os comandos de voz. Logo, a aplicação considerou apenas comandos mais simples e curtos. É

vislumbrado futuramente que o usuário possa utilizar tanto comandos em PT-BR ou inglês para referenciar alguns marcadores, seletores, funções ou valores de propriedades do *HTML* e *CSS*.

Para trabalhos futuros a plataforma pretende tratar a quantidade de comandos de forma mais eficiente. Além disso, pelo escopo e simplicidade do projeto, só foram tratadas as linguagens *HTML* e *CSS*, com apenas uma parte de suas sintaxes. O *CSS* possui uma diversidade de construções variando de seletores, valores e propriedades. É pretendido então fazer um tratamento completo para as linguagens *HTML*, *CSS* e, consequentemente, *Javascript*. Uma forma de realizar essa tarefa é permitir que o usuário insira comandos mais complexos, por exemplo, inserindo ao mesmo tempo uma *tag HTML* com seu nome, atributos e valores (ex: <img src='imagem.jpg' height=720px width=1280px>).

Por fim, esse trabalho serve como um passo inicial no estudo de tecnologias de reconhecimento de voz. É possível pensar em aplicar os conhecimentos aprendidos em outras tecnologias de reconhecimento de voz e outras linguagens mais utilizadas atualmente no mercado. Também é possível iniciar um estudo teórico aprofundando em Reconhecimento Automático de Voz, considerando seus fundamentos e algoritmos utilizados. Esse tipo de estudo pode contribuir com a criação de novos algoritmos em reconhecimento de voz e assim criar novos tipos de tecnologias de reconhecimento da fala.

REFERÊNCIAS

ABREU, M. A. L. D. **Desempenho de Técnicas de Aprendizado de Máquina em Reconhecimento de Voz**. Trabalho de Conclusão de Curso. São José, SC: IFSC, 2019.

BARBOSA, M. d. S. A.; SANTOS, R. M. d.; TREZZA, M. C. S. F. A vida do trabalhador antes e após a Lesão por Esforço Repetitivo (LER) e Doença Osteomuscular Relacionada ao Trabalho (DORT). **Revista Brasileira de Enfermagem**, SciELO Brasil, v. 60, p. 491–496, 2007.

BERGE, G. A. **Leveraging Language Tooling for Better Voice Coding: Implementing program awareness and structural editing for Talon**. Dissertação (Mestrado), 2021.

BESACIER, L. et al. Automatic speech recognition for under-resourced languages: A survey. **Speech communication**, Elsevier, v. 56, p. 85–100, 2014.

BONACCORSO, G. **Machine learning algorithms**. UK: Packt Publishing Ltd, 2017.

CARUANA, R.; NICULESCU-MIZIL, A. An empirical comparison of supervised learning algorithms. In: **Proceedings of the 23rd international conference on Machine learning**. New York, NY, United States: Association for Computing Machinery, 2006. p. 161–168.

CLAYTON, S. Microsoft Research shows a promising new breakthrough in speech translation technology. 2012. Acessado em: 02 nov. 2021. Disponível em: <https://blogs.microsoft.com/ai/microsoft-research-shows-a-promising-new-breakthrough-in-speech-translation-technology/>.

DéSILETS, A.; FOX, D. C.; NORTON, S. VoiceCode: An Innovative Speech Interface for Programming-by-Voice. In: **CHI '06 Extended Abstracts on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2006. (CHI EA '06), p. 239–242. ISBN 1595932984. Acessado em: 02 nov. 2021. Disponível em: <https://doi.org/10.1145/1125451.1125502>.

GOTO, M.; OGATA, J.; ETO, K. Podcastle: A web 2.0 approach to speech recognition research. In: **Eighth Annual Conference of the International Speech Communication Association**. NY: International Speech Communication Association (ISCA), 2007.

LEITE, P. B. C. **Identificação de Tipos de Culturas Agrícolas a partir de Seqüências de Imagens Multitemporais Utilizando Modelos de Markov Ocultos**. Tese (Doutorado) — PUC-Rio, 2008.

LORENA, A. C.; CARVALHO, A. C. de. Introdução às máquinas de vetores suporte. **Relatório Técnico do Instituto de Ciências Matemáticas e de Computaçao (USP/Sao Carlos)**, v. 192, p. 11, 2003.

MENDELS, G. et al. Improving speech recognition and keyword search for low resource languages using web data. In: INTERNATIONAL SPEECH COMMUNICATION ASSOCIATION (ISCA). **INTERSPEECH 2015: 16th Annual Conference of the International Speech Communication Association**. Germany, 2015. p. 829–833.

MILLER, D. R.; LEEK, T.; SCHWARTZ, R. M. A hidden markov model information retrieval system. In: **Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval**. NY: Association for Computing Machinery, 1999. p. 214–221.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, Manole Ltda, v. 1, n. 1, p. 32, 2003.

NORTHWOOD, C. **The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer**. Manchester, UK: Springer, 2018.

PELÁEZ-MORENO, C.; GALLARDO-ANTOLÍN, A.; MARÍA, F. Díaz-de. Recognizing voice over ip: A robust front-end for speech recognition on the world wide web. **IEEE Transactions on Multimedia**, IEEE, v. 3, n. 2, p. 209–218, 2001.

ROBBINS, J. N. **Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics**. CA: "O'Reilly Media, Inc.", 2012.

SCHMIDT, J. L. Apostila html basico. 2015.

SILVA, C. P. A. da. **Um software de reconhecimento de voz para português brasileiro**. Tese (Doutorado) — Dissertação de mestrado, Universidade Federal do Pará, Belém, Brasil, 2010.

SILVA, M. S. **Criando sites com HTML: sites de alta qualidade com HTML e CSS**. São Paulo, SP, Brasil: Novatec Editora, 2008.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. Cambridge, Massachusetts London, England: MIT press, 2018.

TEAM, A. C. **Web Design with HTML and CSS Digital Classroom**. 1st. ed. Chichester, United Kingdom: Wiley Publishing, 2011. ISBN 0470583606.

WANG, Y.-Y. et al. An introduction to voice search. **IEEE Signal Processing Magazine**, IEEE, v. 25, n. 3, p. 28–38, 2008.

WRIGLEY, S. N.; HAIN, T. Web-based automatic speech recognition service-webasr. In: **Twelfth Annual Conference of the International Speech Communication Association**. UK: Department of Computer Science, University of Sheffield Regent Court, 2011.

YNOGUTI, C. A. et al. **Reconhecimento de fala contínua usando modelos ocultos de Markov**. Tese (Doutorado) — Universidade Estadual de Campinas, Campinas, 1999.

YU, D.; DENG, L. **Automatic Speech Recognition.** London: Springer, 2016.

YU, D. et al. Automated directory assistance system-from theory to practice. In: CITESEER. **Interspeech**. USA, 2007. p. 2709–2712.

ZWEIG, G. et al. Voice-rate: A dialog system for consumer ratings. In: **NAACL/HLT (Demonstration Program)**. Association for Computational Linguistics, 2007. p. 31–32. Acessado em: 02 nov. 2021. Disponível em: <https://www.microsoft.com/en-us/research/publication/voice-rate-a-dialog-system-for-consumer-ratings/>.