Inclusão da Modalidade de Interação por Voz no EdiMM - Editor Multissemiótico e Multimodal

Flávio A. da Silva[1], André C. da Silva[1,2], Flávia L. Arantes[2]

[1] Grupo de Pesquisa Mobilidade e Novas Tecnologias de Interação
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP)
Campus Hortolândia

[2] Núcleo de Informática Aplicada à Educação - NIED
Universidade Estadual de Campinas - UNICAMP

flavioantsilva70@gmail.com, andre.constantino@ifsp.edu.br,
flalin@unicamp.br

Abstract*. The Multisemiotic and Multimodal Text Editor (EdiMM) is a tool to build multisemiotic texts, that is, texts involving images, sounds, videos, written and typed words, in a single document. EdiMM has interaction by the input input modalities keyboard, touch, mouse, and pen to trigger commands. Since EdiMM is developed in an evolutionary way with a new modality added by each iteration, the objective of this work was the inclusion of trigger commands by voice. Analyzing three solutions for speech recognition (IBM Watson, Microsoft Cortana and the API Web Kit Speech Recognition), we chose to use the third solution, available on Google Chrome, to activate menu functions and change basic colors in the drawing by voice.*

Resumo. *O Editor Multimodal e Semiótico (EdiMM) é um editor de texto que envolve imagens, sons, vídeos, palavras escritas e digitadas em um único documento. Possui interação pelas modalidades de entrada teclado, mouse, toque e caneta para acionamento de comandos. Sabendo que o EdiMM é desenvolvido de forma evolutiva com uma nova modalidade adicionada por iteração, o objetivo desse trabalho foi a inclusão do acionamento de comandos por voz. Analisando três soluções para reconhecimento de voz (IBM Watson, Microsoft Cortana e a API Web Kit Speech Recognition), optou-se por utilizar a terceira solução, disponível no Google Chrome, para acionar funções do menu e a mudança de cores básicas no traçado por meio de voz.*

1. Introdução

A IHC (Interação Humano-Computador) é uma área preocupada com a implementação de sistemas interativos, aprimorando experiências, com a preocupação e desenvolvimento de sistemas mais usáveis e úteis para o usuário final. Para Padovani (2002), "Interação Humano-Computador é um campo de estudo interdisciplinar que tem como objetivo geral entender como e porque as pessoas utilizam (ou não utilizam) a tecnologia da informação".

Hoje há uma preocupação de como o usuário interage com o conteúdo recuperado em termos cognitivos; o que adiciona mais componentes além do uso dos tradicionais periféricos como teclado, mouse e monitor. Hamilton, Kerne e Robbins (2012)

descrevem que as modalidades de interação baseada em caneta, multi toque, e mouse + teclado são fundamentalmente diferentes. Logo, comparações diretas de performance são difíceis de serem realizadas, e poucos estudos quantitativos foram feitos (Silva, 2014, pág. 52).

Kugler (2008), com base em relatórios do grupo Garther[1], afirma que interfaces naturais e não táteis e a tradução automática da fala seriam os desafios para os próximos 25 anos para a área de Tecnologia da Informação. Mas, esse desafio se complica quando agrega-se às diferentes tendências tecnológicas emergentes (reconhecimento facial, utilização de movimentos e gestos para comandos ao sistema, reconhecimento de voz, computação cognitiva entre outros). Shneiderman (2006) expõe a complexidade de dispositivos de interação existentes:

> "As novas tecnologias da informática incluiriam displays do tamanho de uma parede, aplicados para palmtops, minis sensores médicos parecidos com jóias e computadores sensíveis ao tato que mudam experiências sensoriais e formas de pensar." (SHNEIDERMAN, 2006, p.21)

É para esta complexidade de dispositivos, que suportam diferentes modalidades de interação, que a ferramenta EdiMM está sendo desenvolvida. O EdiMM (Freire et al., 2015; Arantes et al., 2017) - Editor de Texto Multissemiótico e Multimodal[2] - é uma ferramenta computacional em desenvolvimento pelo grupo de pesquisa Mobilidade e Novas Tecnologias de Interação do IFSP campus Hortolândia e o NIED/UNICAMP. O EdiMM visa auxiliar a construir textos multissemióticos, ou seja, textos que envolvam imagens, sons, vídeos, palavras escritas e digitadas, em um único documento. Segundo os autores, citando Braga e Ricarte (2005), o texto multissemiótico, produto da era digital/tecnológica, afeta não só as práticas sociais que envolvem a comunicação, mas também promove mudanças na própria linguagem.

O EdiMM também é uma ferramenta de interação multimodal, ou seja, uma ferramenta na qual o usuário pode interagir usando diferentes modalidades de entrada (como teclado, voz, toque, mouse, caneta, gestos) e, portanto, explorando o contexto de multidispositivos. O EdiMM está em desenvolvimento desde 2013 por meio de diversos projetos de iniciação tecnológica e Trabalhos de Conclusão de Curso. Em relação às modalidades de entrada, trabalhos relacionados a tela sensível ao toque e à caneta foram realizados, mas nenhum trabalho foi desenvolvido considerando a interação por voz. Assim, o objetivo deste trabalho foi evoluir o EdiMM de modo que seja possível utilizar a voz para acionar suas funcionalidades.

Na Seção 2 apresentamos os conceitos modalidade, multimodalidade e sistemas multimodais, além da ferramenta EdiMM, que utilizamos neste trabalho. Na Seção 3 apresentamos a metodologia utilizada para desenvolver o nosso trabalho de interação com comando de voz pelo usuário do EdiMM, estudo da aplicação da API *Speech Recognition do Web Kit* do Google Chrome. Na Seção 4 detalhamos o desenvolvimento deste trabalho modificando a aplicação, considerando recursos da linguagem JavaScript, para a entrada de comandos por voz do usuário, destacando trechos relevantes do código. Finalmente, na Seção 5, concluímos o trabalho com a certeza do poder dessa API, que até o término deste artigo, somente o Google Chrome fornece suporte a tal tecnologia.

---

[1] O grupo Gartner oferece consultoria e realiza pesquisa na área de tecnologia da informação, oferecendo informações sobre tendências e aconselhamento a clientes desde 1979. Mais informações em http://www.gartner.com .

[2] O EdiMM é acessível pelo link: https://zephyrus.unicamp.br/edimm/

2. Fundamentação Teórica

2.1. O EdiMM

Para o desenvolvimento do EdiMM, foram utilizadas as tecnologias HTML, JavaScript e CSS para implementar a interface de usuário (componente cliente da arquitetura adotada) em conjunto com técnicas de Design Responsivo (Marcotte, 2011) de modo que o editor possa ser utilizado em diferentes dispositivos e rodar em um servidor Apache, com o código da aplicação escrito na linguagem PHP. A Figura 1 apresenta a interface do EdiMM acessada em um computador de mesa, enquanto que a Figura 2 apresenta a interface acessada por um dispositivo móvel.

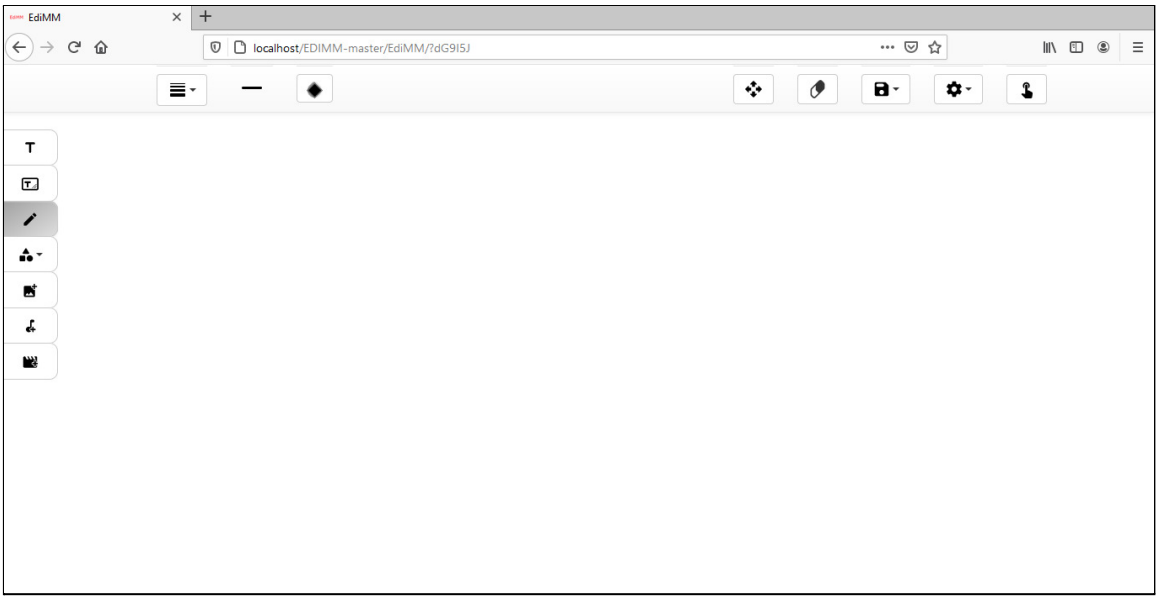

Figura 1. Imagem da interface de usuário do EdiMM acessado em um computador desktop.

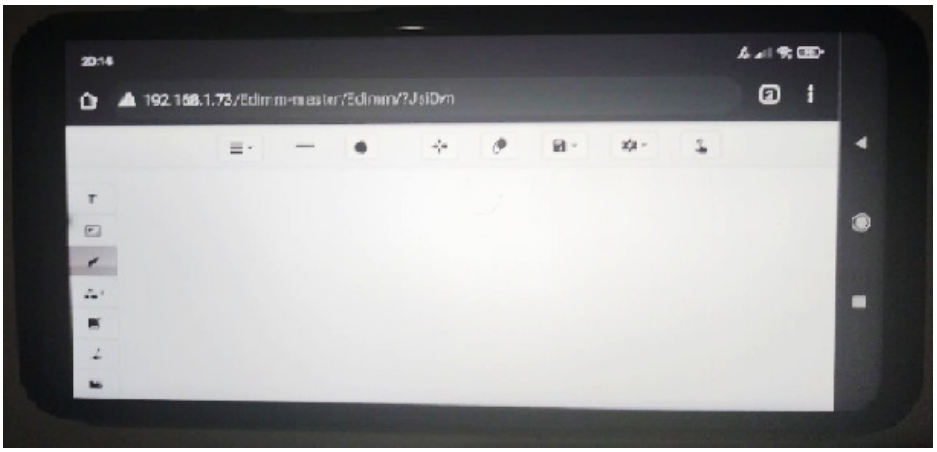

Figura 2. Imagem da interface de usuário do EdiMM acessado em um dispositivo móvel.

A interface do EdiMM se caracteriza por um grande espaço de trabalho, onde o usuário pode inserir as semioses (texto, imagem, vídeos, áudios, etc). No lado esquerdo apresenta-se uma barra de ferramentas com as opções para inserir uma semiose (de cima para baixo, inserir texto digitado, inserir uma caixa de texto, desenhar, inserir polígonos, inserir imagem, inserir áudio e inserir vídeo). No topo ficam dispostos elementos modificadores da semiose (como, por exemplo, cor, espessura, tamanho de fonte), bem como as ferramentas mover e apagar elementos. No topo também estão dispostos um botão para salvar, configurar a página e informar o uso de toque para interagir.

2.2. Modalidade de interação, Multimodalidade e Sistemas de interação multimodais

Para realização deste trabalho, investigamos sobre o termo modalidade, multimodalidade e sistemas de interação multimodal. Apresentamos a seguir um trecho da tese de Silva (2014) sobre modalidade:

> "O termo Modalidade será usado para definir o modo como uma entrada do usuário ou uma saída do sistema são expressas. Nigay e Coutaz (1995) definem modalidade como um método de interação que um agente pode usar para alcançar uma meta ressaltando que uma modalidade pode ser especificada em termos gerais como "usando fala" ou em termos mais específicos como "usando microfone". Por exemplo, pode-se usar a modalidade de entrada à caneta, que envolve os periféricos de entrada caneta, tela sensível à caneta, e a mão do usuário utilizada para segurar a caneta." (SILVA, 2014, pág. 6)

Assim, segundo Nigay e Coutaz, podemos citar como exemplos de modalidade: o teclado, a tela sensível ao toque, o mouse, a caneta e a fala. O teclado é um periférico de entrada de dados e comandos, constituído por um combinado de teclas representando letras, números, símbolos e algumas funções. O mouse consiste em um periférico utilizado para movimentar um ponteiro em uma determinada interface, possuindo manipulações como movimento e clique, bem como combinações como clique-duplo e clique-segura-arrasta-solta. O periférico denominado caneta (usualmente chamada de *stylus* em inglês) pode ser usado para mover um ponteiro, selecionar elementos da interface ou arrastá-los, e para simples escrita. O toque é uma modalidade tátil, que utiliza-se de sensores, para identificar uma ou mais regiões que os dedos entram em contato com uma superfície. A interatividade por voz, que necessita de um microfone para entrada, necessita de um processamento dos dados da entrada de modo a identificar comandos para o sistema.

Sistemas de interação monomodal (ou tradicional) utilizam uma modalidade de interação para a entrada, por exemplo, teclado ou mouse, e uma modalidade de saída, como monitor de vídeo, caixas de som ou impressoras. Em sistemas de interação multimodal (Figura 3), o usuário interage com o sistema de informação utilizando diferentes modalidades como, por exemplo, uso de recurso de voz, toque na superfície (*touchscreen*), aplicação de gestos entre outros, e as saídas de informações por meio de interfaces gráficas de vídeo, respostas sonoras ou hápticas, interface tátil em que um sistema fornece respostas ao usuário (por exemplo os controles *dualshock* do Playstation), entre outras. As modalidades podem apresentar dados de modo que se complementam ou são redundantes.

Figura 3. Esquema de entrada, processamento e saída na interação multimodal.

Devido ao propósito deste trabalho envolver a interação por voz, focaremos nossos estudos nessa modalidade de interação. Assim, a seguir, detalhamos alguns frameworks que auxiliam o programador a desenvolver aplicações que consideram a voz como modalidade de entrada de dados.

2.3. Soluções de reconhecimento de voz

Algumas soluções de reconhecimento de voz estão disponíveis atualmente, a exemplo, o IBM Watson, a Cortana da Microsoft e a API *Web Kit Speech Recognition*.

A IBM foi a primeira a apresentar sua IA (Inteligência Artificial) ou, como eles gostam de chamar, Computação Cognitiva, com o Watson. Estreando o Watson em 2011 no *Jeopardy* (programa de televisão da CBS Television Distribution, de perguntas e respostas variando entre História, Literatura, Cultura e Ciência).

Com o serviço de *Speech Recognition* do Watson é possível converter áudio em texto em mais de 15 idiomas. Esse serviço é fornecido por meio de uma API (*Application Programming Interface*), que também possui um funcionalidades para *Text-to-Speech*, que consiste em transformar texto em fala, ou seja, sintetização de voz, possuindo mais de 14 vozes diferentes com grande semelhança com a voz humana.

O preço do Watson nesse serviço varia de acordo com o *Data Center* escolhido; é sugerido aos brasileiros usar o Data Center de Washington-DC, EUA, que disponibiliza o uso do Watson por meio de três planos[3]:

1. Plano *Lite*: Grátis até 10 mil caracteres/mês.
2. Plano *Standard*: U$ 0,02/mil caracteres.
3. Plano *Premium*, valor varia conforme as seguintes características:
   - Os dados de uso e de treinamento são privados e armazenados em um ambiente isolado;
   - Alta disponibilidade e garantia de tempo de atividade no nível de serviço;
   - Ambiente de locatário único;

[3] Sugestão de preços em 2020

- Autenticação Mútua;
- HIPAA (Segurança e Regra de Privacidade) - Washington, DC.

A *Web Speech* API é uma tecnologia experimental (conforme nota de aviso no início de sua documentação), gratuita que pode executar a função de reconhecimento de voz e retornar em texto. Essa API é utilizada no editor de texto do Google Docs, disponível dentro do Google Drive e possui suporte para vários idiomas. A interface da W*eb Speech* API*, Speech Recognition* é poderosíssima, mesmo sendo uma tecnologia experimental, podendo ser utilizada em vários ambientes de desenvolvimento, embora o reconhecimento de voz da *Web Speech* API é limitado ao Chrome para Desktop e ao Android, mas com a necessidade de incluir interfaces pré-fixadas, como *Web Kit Speech Recognition*. No navegador Chrome, o uso do reconhecimento de voz em uma página Web envolve um mecanismo de reconhecimento baseado em servidor: o áudio é enviado para um serviço Web para processamento, por isso não funciona *offline*.

A assistente pessoal digital do Windows, a Cortana, tem como promessa manter o usuário bem informado, permitindo a realização de atividades diversas, por meio de dispositivos e plataformas distintas, com a realização de atividades através do comando de voz. Recurso que aprende com o usuário para melhor servi-lo. Essa é uma API da Microsoft, em um serviço Web, assim é necessário estar *on-line* também.

O reconhecimento de voz se baseia, simplesmente, em captar o áudio através de um periférico (microfone), conectado a um dispositivo de processamento (computador), e enviar a um servidor Web (no caso de tecnologias que oferecem o reconhecimento de voz como um serviço Web), o qual utiliza-se de uma lista gramatical, ou seja, um vocabulário desejado, onde é retornado uma lista de *Strings* (textos). Esse retorno, um conjunto de palavras identificadas, deve ser tratado por função em JavaScript com o objetivo de retirar os ruídos, ou seja, palavras que não são relevantes ao sistema. Por meio de outra função, cujo objetivo é filtrar, por comparação de *Strings*, os comandos necessários para a operação do sistema.

Sumarizando, apresentamos na Tabela 1 um comparativo entre essas diferentes tecnologias de reconhecimento de voz apresentadas.

Tabela 1. Comparação entre as tecnologias IBM Watson, Microsoft Cortana e a API *Web Kit Speech Recognition*.

Framework	**Integração com ferramentas Web**	**Localização da máquina de reconhecimento de voz**	**Suporte para dispositivos móveis**
IBM Watson	sim	Washington, DC	sim
Cortana	sim	Atualmente, o Reconhecimento do Locutor está disponível apenas no Oeste dos EUA	sim
API Web Kit Speech Recognition	sim	Servidor da Google	sim

Këpuska e Bohouta (2017) comparam três sistemas de reconhecimento de voz: Microsoft API, Google API (*API Web Speech Recognition*) e CMU Sphinx. Këpuska e Bohouta criaram uma ferramenta para fazer a comparação entre os sistemas considerando a qualidade do acústico, modelos de linguagem, modelos de palavras e erros de palavras. Os autores concluem que a solução do Google é superior em relação a modelo de linguagem e modelagem acústica (acústico de ambiente).

A Tabela 2 apresenta compatibilidade dos navegadores mais utilizados em relação ao framework API *Web Kit Speech Recognition*. Para a utilização da *API Web Kit Speech Recognition*, o navegador necessita dar suporte a alguns métodos e atributos da API, dispostos na primeira coluna da tabela: o construtor do objeto que irá cuidar do ciclo do processo de recebimento dos dados da voz, envio dos dados para o reconhecimento no servidor e recebimento dos resultados; o evento *start*, disparado quando o processo da coleta dos dados por um microfone começa; o evento *audiostart*, que o navegador deve chamar quando for iniciada a coleta do áudio; e o atributo *lang*, para definir o idioma a ser considerado no processo de reconhecimento. As demais colunas indicam se há suporte para um determinado navegador e a qual a partir de qual versão que o suporte é oferecido.

Tabela 2. Compatibilidade dos navegadores mais utilizados em computadores desktop e dispositivos móveis com o framework API *Web Kit Speech Recognition*.

Fonte: https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition

	DESKTOP						MOBILE					
Metódos e atributos do Speech Recognition	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Firefox for Android	Opera for Android	Safari on iOS	Samsung Internet
SpeechRecognition	33	<= 79	No	No	No	No	4.4.3	33	No	No	No	2.0
SpeechRecognition() constructor	33	<=79	No	No	No	No	37	Yes	No	No	No	Yes
start	33	<=79	No	No	No	No	Yes	Yes	No	No	No	Yes
audiostart event	33	79	No	No	No	No	Yes	Yes	No	No	No	Yes
lang	33	<=79	No	No	No	No	Yes	Yes	No	No	No	Yes

Também com a *Web Speech* API vem a *SpeechSynthesis*, com o propósito de síntese de voz, e consiste em conversão de texto em voz. Uma função que referencia essa API recebe o texto desejado e retorna os bytes do áudio a serem reproduzidos na máquina do usuário. Essa funcionalidade depende de um periférico de saída (alto-falantes) conectado a um dispositivo de processamento onde roda a aplicação. Conforme o site do Mozilla (2020) "O suporte para a síntese de voz da API *Web Speech* (*Speech Synthesi ou text to speech)* ainda está chegando entre os navegadores principais e atualmente está limitado aos seguintes:

- O Firefox desktop e móvel é compatível com Gecko 42+ (Windows) / 44 +, sem prefixos, e pode ser ligado virando a media.webspeech.synth.enabledbandeira para true dentro:config.
- Firefox OS 2.5+ oferece suporte, por padrão, e sem a necessidade de nenhuma permissão.
- O Chrome para Desktop e Android tem suporte desde a versão 33, sem prefixos."

3. Metodologia

Para a possibilitar a interação por voz pelo usuário no EdiMM utilizou-se a *Web Speech* API[4], da Google Cloud. Nossa escolha é pautada pelo trabalho de Këpuska e Bohouta (2017) e, por ser uma solução Web, visto que o EdiMM é uma ferramenta para a Web. Há outras formas e APIs para reconhecimento de voz que requerem exclusividade como Cortana do Windows, Watson da IBM, *Assistent* do Google, etc. Entretanto, preferimos o uso da *Web Speech* API por ser uma solução mais aberta.

Por se tratar de um trabalho de inovação, optamos por realizar modificações no EdiMM para incluir o *framework*, criando assim um protótipo, portanto caracterizamos este trabalho como exploratório, com as seguintes atividades:

1. Estudo da API *Web Kit Speech Recognition* e do EdiMM;
2. Integração do EdiMM com a API - implementação da funcionalidade de escolha de cor via voz;
3. Realização de testes e aprimoramento da solução;
4. Implementação do acionamento das demais ferramentas do menu por voz.

4. Desenvolvimento

A seguir detalharemos cada uma das atividades definidas na Seção Metodologia, com exceção da atividade 1, pois foi abordada na Seção 2.

4.1. Integração do EdiMM com a API - Implementação da funcionalidade de escolha de cor via voz

Para analisar a viabilidade de integração entre a *Web Speech* API e o EdiMM, selecionamos como estudo de caso a mudança, por meio da voz, da cor do traçado ao usar a ferramenta escrita (ícone "lápis" na barra de ferramentas da Figura 1). Definiu-se que, para acionar o mecanismo de reconhecimento de voz, dispor um novo botão com o ícone de microfone. Ou seja, ao clicar ou tocar no botão microfone no menu de ferramentas do EdiMM, ou clicando com o botão direito do mouse, a API será acionada e ao dizer uma cor (limitamos entre as cores mais utilizadas da paleta) será realizado o reconhecimento das palavras, retornando as palavras identificadas, onde esse retorno será filtrado por nosso código e, através de uma palavra-chave, será acionada a funcionalidade de mudança para a cor selecionada automaticamente e o EdiMM deverá dizer, por sintetização de voz, "Foi selecionada a cor" e o nome da cor reconhecida.

É importante relembrar que o *Web Kit Speech Recognition* do Google trabalha com o modelo cliente-servidor, por isso precisa estar conectado à internet e, por essa API ainda ser *beta* (experimental), muitos navegadores não suportam a mesma por isso, no início do código, utilizamos uma verificação se o navegador utilizado pelo usuário oferece suporte para essa tecnologia (linha 6 na Figura 4). Caso o navegador utilizado pelo usuário não possua suporte a esta tecnologia, é emitida uma mensagem ao usuário (linhas 16 a 18 da Figura 4) e o botão de acionamento da funcionalidade não é exibido ao usuário.

[4] Documentação:
https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API

```
1  window.addEventListener('DOMContentLoaded', function(e) {
2      var speakBtn = document.querySelector('#speakBtn');
3      var So = navigator.appVersion.indexOf("Linux; Android");
4      //alert(So);
5      // console.log(So);
6      if (window.SpeechRecognition || window.webkitSpeechRecognition && So == -1) {
7          var spech_api = window.SpeechRecognition || window.webkitSpeechRecognition;
8          var recognition = new spech_api(); //caniuse.com
9          recognition.lang = 'pt-Br';
10         speakBtn.addEventListener('click', function(e) {
11             recognition.start();
12
13         }, false);
14
15     } else {
16         alert("Navegador ou Sistema Operacional não suportam comando de voz");
17         console.log("Navegador não suporta");
18         $("#speakBtn").css("display", "none");
19     }
20 }, false);
```

Figura 4. Parte do código inserido no EdiMM para iniciar o uso da *Web Kit Speech Recognition*.

O reconhecimento de voz é acessado por meio da interface do *Speech Recognition* utilizando um construtor (linha 8 na Figura 4) para iniciar a interface, depois atribuímos a propriedade (.lang) para selecionar o idioma português (linha 9 da Figura 4) e utilizando o método start() (linha 11 da Figura 4) para iniciar a captura do áudio utilizando o microfone do dispositivo. Como decidiu-se pela criação de um botão disponível no menu para acionar o processo de captura e reconhecimento da fala, atribui-se um tratamento de evento neste botão (linhas 10 a 13 da Figura 4). O usuário, ao clicar no botão, irá acionar a função declarada no *listener* da linha 10, que solicita ao mecanismo de reconhecimento o início do processo de captura do que o usuário irá dizer. Após detectar uma pausa na fala do usuário, os dados são enviados para o servidor Web para reconhecimento.

4.2. Tratamento do retorno da API

O retorno da API se dá através da variável *results* (linha 2 da Figura 5), que nos retorna um vetor de *Strings*, por sua vez necessita ser tratado através de uma função JavaScript que modifica as palavras do vetor que somente tenham letras minúsculas, pois para o EdiMM não desejamos essa sensibilidade (linha 4 da Figura 5) e, em seguida, separa o texto reconhecido em palavras (linha 6 da Figura 5).

```
1  recognition.addEventListener('result', function(e) {
2      var result = e.results[0][0].transcript;
3      //passa o novo vetor em caixa baixa
4      var vetCaixaBaixa = result.toLowerCase();
5      //separa o vetor em index
6      var novo1 = vetCaixaBaixa.split(" ");
7
8      var novo = novo1;
9      var res = (novo.find(filt));
```

Figura 5. Parte do código inserido no EdiMM para tratamento dos dados retornados pela *Web Kit Speech Recognition*.

Observe que, na linha 9 da Figura 5, há uma chamada à função de nome *filt,* pois a função *find()* só aceita como parâmetro outra função. Nesta função é determinado um vetor de palavras-chave (linhas 3 a 5 da Figura 6), e se faz a comparação das palavras retornadas e tratadas em nossa variável (*var novo*), verificando a existência desse elemento dentro do vetor de palavras-chave (linhas 8 a 13 da Figura 6). Como, para o caso de acionamento de funções, não importa diferenciar maiúsculas de minúsculas, converte-se todos os caracteres das palavras identificadas para minúsculo (linha 7 da Figura 6).

```
1   function filt(element) {
2       var reservada = [];
3       reservada = ["apagar", "lápis", "vermelha", "escrever", "preto",
4        "preta", "vermelho", "vermelha", "amarelo", "amarela", "azul",
5         "paleta"];
6       var qtde = reservada.length;
7       var elemento = element.toLowerCase();
8       for (let i = 0; i < qtde; i++) {
9           if (reservada[i] == elemento) {
10              var res = elemento;
11              return res.toLowerCase();
12          }
13      }
14  }
```

Figura 6. Parte do código inserido no EdiMM para encontrar palavras-chave na String retornada pela *Web Kit Speech Recognition*.

4.3. Aplicação do comando pela determinação através da palavra-chave

Após a identificação de uma palavra-chave, esta é retornada pela função (linha 9 da Figura 6). Por meio de uma estrutura de seleção *switch* (linha 1 da Figura 7), damos tratamento à palavra reconhecida realizando a ação desejada. No caso de identificação da palavra 'vermelho', será executada a linha 6 da Figura 7, que aciona a função de mudança de cor chamada *color()*. Determinamos que em nosso trabalho iremos mudar algumas cores e alguns itens do menu, por isso foi criada a função *color*(), para a mudança de cor, e a função *action*(), para acionar o menu. Todas as ações através do comando de voz possui um retorno ao usuário, um feedback, utilizando os periféricos de alto falantes com respostas padrões pré-selecionadas.

4.4. Explicação sobre as funções de ação no sistema

A chamada para a função *color* é realizada conforme a palavra identificada (linhas 3, 6, 9 e 21 da Figura 7), passando como parâmetro a cor reconhecida por código hexadecimal e uma *string*, que irá produzir um feedback visual, mudando a cor do elemento de interface no formato de "gota" como mostra a Figura 8, e um feedback sonoro. A Figura 9 mostra o código para exibir o feedback para a mudança de cor na "gota" de nossa interface (linha 17 na Figura 9), e retorno sonoro, através da função *speak()* (linha 14 Figura 9), por meio de uma mensagem sintetizada (construída na linha 10 da Figura 9).

```
switch (res) {
    case 'preto':
        color('#000000', 'preto');
        break;
    case 'vermelho':
        color('#ff0000', 'vermelho');
        break;
    case 'amarelo':
        color('#ffff00', 'amarelo');
        break;
    case 'apagar':
        action("#delete", "apagar");
        break;
    case 'escrever':
        action("#keyboard", "escrever");
        break;
    case 'lápis':
        action("#draw", "lápis");
        break;
    case 'vermelha':
        color('#ff0000', 'vermelho');
        break;
    default:
        speak("Pode repetir não foi possivel entender");
        break;
}
```

Figura 7. Parte do código inserido no EdiMM para acionar funções a partir de palavras específicas retornada pela *Web Kit Speech Recognition*.

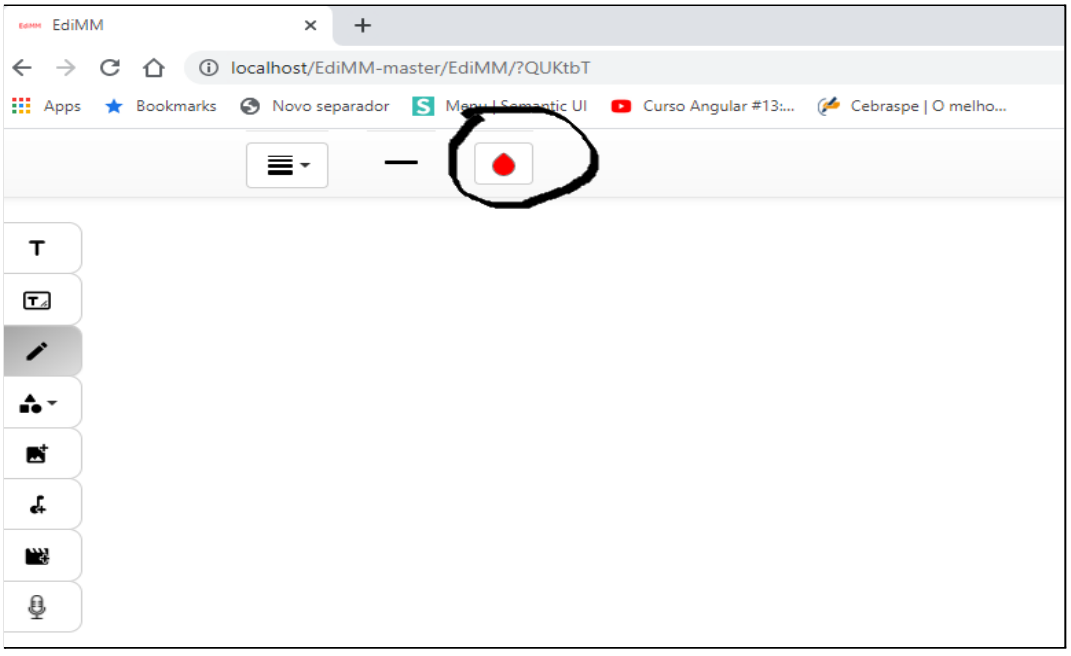

Figura 8. Feedback visual da mudança para a cor escolhida

```
1  function action(idHtml, busca) {
2      var id = idHtml;
3      $(id)[0].click();
4
5      speak("Foi escolhida a opção" + busca)
6  }
7  //---------------------------------
8  function color(color, name) {
9      var test = document.getElementsByClassName("color-picker")[0]['value'];
10     var u = "Foi escolhida a cor" + name;
11     var color = String(color);
12     var returnSetcolor = "this.value";
13     var loko = $(".color-picker").attr("onchange");
14     speak(u);
15     if (loko == "setColor(this.value);") {
16
17         setColor(color);
18
19     } else {
20         $(".color-picker").attr({ 'onchange': 'setColor(' + returnSetcolor + ')' });
21
22     }
23
24 }
```

Figura 9. Parte do código inserido no EdiMM para realizar a mudança de cor selecionada internamente e na interface de usuário.

Para o acionamento de outras funcionalidades, como escrever (com teclado), usar o lápis e apagar, as palavras escolhidas a serem ditas pelo usuário foram incluídas no conjunto de palavras-chave (linha 3 a 5 da Figura 6) que, quando encontradas no texto retornado pelo servidor de reconhecimento de voz (linha 9 da Figura 6), acionará uma ação no EdiMM por meio da chamada da função *action* (linhas 12, 15 e 18 da Figura 7). A função *action* (linhas 1 a 6 da Figura 9) simula um clique de mouse no elemento da interface de usuário, acionando assim a respectiva funcionalidade.

4.5. *Speech Synthesis*: recurso *text-to-speech*

A sintetização de voz é realizada por meio da *Speech Synthesis,* recurso *text-to-speech*, como é mais conhecida, consiste em transformar texto em voz, utilizando os alto-falantes do dispositivo em que o sistema está sendo executado, para reproduzir o áudio.

Criamos uma função *speak*() (chamada realizada nas linhas 5 e 14 da Figura 9), que está presente em todos os comandos de voz na aplicação, retornando um feedback da opção desejada, com frases padrão da aplicação. A função *speak* cria uma instância do sistema de sintetização de voz (linha 4 da Figura 10), altera a propriedade .lang com o idioma desejado (linha 6 da Figura 10), solicitando que a criação e reprodução do áudio referente ao texto recebido como parâmetro (linha 9 da Figura 10), a velocidade com que o enunciado ou a mensagem pré-determinada será falada é determinada pela propriedade *.rate*. Sendo que a velocidade normal é 1, sendo 2 o dobro de velocidade normal, podendo utilizar um float para controlar (linha 7 da Figura 10). Se a propriedade *.rate* não for setada o sistema de sintetização assume o valor 1, velocidade de fala normal.

Optamos por criar as funções em nossa aplicação para tornar a dependência da API o mínimo possível pois, como se trata de uma arquitetura cliente-servidor, a comunicação com o servidor da Google pode fazer nossa aplicação perder desempenho. Como trabalho futuro sugerimos a integração com outros reconhecedores de voz afim de verificar a dependência entre o EdiMM e os reconhecedores de voz.

```
1   //funcao de speak resposta----------------------------
2   function speak(params) {
3      //api de speak fala o que esta na variavel u.text
4      var u = new SpeechSynthesisUtterance();
5      u.text = params;
6      u.lang = 'pt-Br';
7      u.rate = 1.2;
8
9      return speechSynthesis.speak(u);
10  }
11
```

Figura 10. Parte do código inserido no EdiMM para realizar um feedback audível ao usuário.

5. Conclusão

O EdiMM, ferramenta multimodal para a elaboração de textos multissemióticos, está em desenvolvimento desde 2013 por meio de diversos projetos de iniciação tecnológica e Trabalhos de Conclusão de Curso. Em relação às modalidades de entrada, trabalhos relacionados a tela sensível ao toque e à caneta foram realizados, mas nenhum trabalho foi desenvolvido considerando a interação por voz. Assim, o objetivo deste trabalho foi evoluir o EdiMM de modo que seja possível utilizar a voz para acionar suas funcionalidades.

Após o estudo de três soluções para reconhecimento de voz (IBM Watson, Cortana da Microsoft e a *API Web Kit Speech Recognition*), optou-se pela adoção da *API Web Kit Speech Recognition*. Ao aplicar essa API no EdiMM conseguimos acionar funcionalidades da aplicação com comandos de voz, inicialmente por meio da mudança de cor e, em seguida, as demais funcionalidades da aplicação como “apagar”, "lápis" e “escrever”.

Observamos o quão poderosa a API *Web Kit Speech Recognition* pode ser através das diversas possibilidades de manuseio ao sistema. Acreditamos que, com o reconhecimento de voz, há uma melhoria de acessibilidade da ferramenta, e uma inclusão de pessoas e nova maneira de acesso ao aplicativo. Apontamos a verificação da acessibilidade como trabalhos futuros.

É necessário frisar que a solução de reconhecimento de voz adotada demora um pouco para interpretar (3 a 4 segundos) e está em teste até o momento da observância nesse artigo e, infelizmente, é suportada apenas pelo Chrome. Por ser uma API do Google, com arquitetura cliente-servidor tem a necessidade de uma conexão com a internet, portanto só funcionará *on-line*.

Também não foram implementadas as funcionalidades de formatação de texto digitado. O comando de voz através da referente API poderá desenvolver outras funções futuramente como a escrita de textos, sublinhar partes do referido texto, entre outras. Ao conhecermos o poder dessa API a criatividade cabe conforme a necessidade.

Para a realização deste trabalho foram utilizados conhecimentos das matérias de Interação Humano-Computador, Desenvolvimento WEB (principalmente JavaScript), Algoritmos e Lógica de Programação, Engenharia de Software, Sistemas Operacionais, Arquitetura de Software, entre outras.

Ressaltamos que este trabalho foi apresentado na Semana Nacional de Ciência e Tecnologia do campus Hortolândia em 2019, na Mostra de Trabalhos Técnicos, e foi premiado como melhor trabalho na modalidade protótipo.

Referências

ARANTES, F. L.; FREIRE, F. M. P.; BREUER, J.; SILVA, A. C. DA; OLIVEIRA, R. C. A. DE; VASCON, L. E. Towards a Multisemiotic and Multimodal Editor, Journal of Computer Science and Technology, v. 17, n. 02, e14., 2017. https://doi.org/10.24215/16666038.17.e14

BRAGA, D. B.; RICARTE, I. Letramento e Tecnologia. Campinas, UNICAMP: Cefiel & MEC, Secretaria de Ensino Fundamental, 2005.

EdiMM. Editor Multimodal e Multissemiótico. Disponível em: < https://zephyrus.unicamp.br/edimm > . Acessado em: agosto de 2020.

FREIRE, F. M. P.; ARANTES, F. L.; SILVA, A. C.; VASCON, L. E. L. Estudo de viabilidade de um Editor Multimodal: o que pensam os alunos?. In: XX Congreso Internacional de Informática Educativa (TISE 2015), 2015, Santiago, Chile. Proceedings of TISE – Nuevas Ideas en Informática Educativa, v. 11, 2015. p. 109-119.

HAMILTON, W.; KERNE, A.; ROBBINS, T. High-Performance Pen + Touch Modality Interactions: A real-time strategy game eSports context. In: Annual ACM Symposium on User interface software and technology (UIST 2012), 25., 2012, Cambridge, MA, EUA. *Proceedings of...* Nova Iorque, EUA: ACM, 2012. p. 309-318, doi: 10.1145/1866029.1866036.

KËPUSKA, V.; BOHOUTA, G. Comparing speech recognition systems (Microsoft API, Google API and CMU Sphinx), **International Journal of Engineering Research and Application**, v. 7, n. 3 (part-2), 2017. p. 20-24. Disponível em: < https://www.researchgate.net/profile/Gamal-Bohouta-2/publication/314938892_Comparing_Speech_Recognition_Systems_Microsoft_API_Google_API_And_CMU_Sphinx/links/58c75a2292851cd9c1509b2d/Comparing-Speech-Recognition-Systems-Microsoft-API-Google-API-And-CMU-Sphinx.pdf > . Acessado em: agosto de 2020.

KUGLER, L. Goodbye, **Computer Mouse. Communications of the ACM**, Nova Iorque, EUA: ACM, v. 51, n. 9, set. 2008. pp. 56.

MARCOTTE, E. Responsive Web Design. A Book Apart, 2011. 150 p.

Mozilla MDN Web Docs. Using the Web Speech API. https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API . Acessado em: agosto de 2020.

NIGAY, L.; COUTAZ, J. A Generic Platform for Addressing the Multimodal Challenge. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 13., 1995, Colorado, EUA. *Proceedings of...* Nova Iorque, EUA: ACM Press / Addison-Wesley Publishing Co., 1995. p. 98-105.

PADOVANI, S. **Avaliação Ergonômica de Sistemas de Navegação em Hipertextos Fechados**. In: MORAES, Anamaria de. Design e Avaliação de Interface. Rio de Janeiro, iUsEr, 2002. p. 27-58.

SHNEIDERMAN, B. O Laptop de Leonardo: Shneiderman, 2006. 228 p. Tradução de Vera Whaterly, Rio de Janeiro: Nova Fronteira. 2006.

SILVA, A. C. DA. **Interação multimodal em ambientes de EaD: proposta de arquitetura e impactos.** 2014. Tese (doutorado em Ciência da Computação) - Universidade Estadual de Campinas, Instituto de Computação, Campinas, SP. Disponível em: < http://www.repositorio.unicamp.br/handle/REPOSIP/275578 >. Acesso em: 28 ago. 2018.

Documento Digitalizado Público

TCC - Artigo - versão final

Assunto: TCC - Artigo - versão final
Assinado por: Andre Constantino
Tipo do Documento: Relatorio
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Andre Constantino da Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 19/08/2021 15:46:14.

Este documento foi armazenado no SUAP em 19/08/2021. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 748257
Código de Autenticação: ad0fd107ad