Implementação de API para Reconhecimento e Sintetização de Voz em um Aplicativo Móvel

Alternative Title: Implementing API for Speech Recognition and Synthesizing in a Mobile Application

Lucas Debatin
Centro Universitário de Brusque - UNIFEBE
Rua Dorval Luz, 123 - Santa Terezinha
Brusque – SC
lucasdebatin@hotmail.com

Aluizio Haendchen Filho
Centro Universitário de Brusque - UNIFEBE
Rua Dorval Luz, 123 - Santa Terezinha
Brusque – SC
aluizioh@terra.com.br

Jonathan Nau
Centro Universitário de Brusque - UNIFEBE
Rua Dorval Luz, 123 - Santa Terezinha
Brusque – SC
jonathan_nau@live.com

Pedro Sidnei Zanchett
Centro Universitário de Brusque - UNIFEBE
Rua Dorval Luz, 123 - Santa Terezinha
Brusque – SC
pedrozanchett@gmail.com

Wagner Correia
Centro Universitário de Brusque - UNIFEBE
Rua Dorval Luz, 123 - Santa Terezinha
Brusque – SC
wagnercorreia@hotmail.com.br

RESUMO

O uso de interfaces inteligentes, recursos de usabilidade e tecnologias de voz estão possibilitando que as aplicações se tornem cada vez mais ricas, em especial para auxiliar usuários inexperientes ou com necessidades especiais. Com isso muitas empresas desenvolvedoras de softwares estão buscando maneiras de implementar as tecnologias de voz em seus produtos, e uma das formas mais utilizadas é através do uso de *Application Programming Interface* (API's). As tecnologias de voz são divididas em duas categorias: reconhecimento de voz, que é utilizado em comandos por voz (converte a voz em texto), e sintetizador de voz, que é utilizado para melhorar a acessibilidade nos dispositivos (converte o texto em fala). Essas tecnologias fazem uso do Processamento de Linguagem Natural, subárea da Inteligência Artificial, para processar e manipular a linguagem humana em diversos níveis. Este artigo apresenta um levantamento das principais API's de reconhecimento e sintetização da voz, descrevendo as suas características e funcionalidades. Além disso, um estudo de caso mostra qual API foi escolhida dentre as que foram pesquisadas, e como a mesma foi implementada no aplicativo Alerta Brusque.

Palavras-Chave

Experiência de Usuário, Usabilidade, Reconhecimento de Voz, Sintetização de Voz.

ABSTRACT

The use of intelligent interfaces, usability features and voice technologies are enabling applications to become increasingly rich, especially to assist inexperienced users or those with special needs. Therewith, many software developers are looking for ways to implement voice technologies in their products, and one of the most commonly used forms is the Application Programming Interface (API). Voice technologies are divided into two categories: voice recognition, which is widely used in voice commands (converts voice to text), and speech synthesizer, which is widely used to improve accessibility in devices (convert text to speech). These voice technologies use Natural Language Processing techniques, subarea of Artificial Intelligence, in order to process and manipulate human language at several levels. This article presents an analysis of the main voice recognition and synthesizing APIs, describing their characteristics and functionalities. In addition, as a case study, it shows which API was chosen, among those that were researched, and how it was implemented in the Alert Brusque application.

CCS Concepts

• Computing methodologies → Natural language processing • Computing methodologies → Speech recognition • Human-centered computing → Natural language interfaces

SBSI 2017, Jun 5–8, 2017, Lavras, Minas Gerais, Brazil.

Keywords
User Experience, Usability, Voice Recognition, Voice Synthesizing.

1. INTRODUÇÃO
Atualmente, muitos softwares não atendem às diretrizes de acessibilidade da lei 5.296. Segundo a Casa Civil [1], essa lei nos diz que todos os dispositivos devem oferecer recursos que exigem menor esforço físico e mental dos usuários minimizando sua desorientação ou sobrecarga cognitiva.

Com base nessa lei, se faz importante pesquisar para implantar nos softwares, respostas rápidas por meio de recursos tecnológicos e técnicas de inteligência artificial, ao invés de utilizar como principal forma de entrada somente o *touch-screen* e o teclado.

Devido à evolução das tecnologias e a computação ubíqua da voz, houve a possibilidade de criar aplicações extremamente ricas, especialmente para pessoas com pouca experiência ou com necessidades especiais.

Projetar aplicativos que executam tarefas com mãos e olhos livres mostram-se extremamente vantajosos independentemente do tipo de usuário, ele será capaz de executar o sistema com apenas alguns segundos de orientação das atividades através de comandos sonoros e confirmação das solicitações verbalmente [2].

Baseado nisso, as tecnologias de voz estão sendo cada vez mais implementadas em softwares, como consequência, surgem no mercado diversas API's que realizam as tecnologias de voz, reconhecimento e sintetização.

Este artigo será embasado em uma fundamentação teórica sobre os temas: (i) processamento de linguagem natural (PLN); (ii) tecnologias de voz (reconhecimento e sintetização); e (iii) um levantamento bibliográfico das API's existentes no mercado. O artigo apresenta um estudo de caso, onde será abordado como foi a implementação da API escolhida no aplicativo Alerta Brusque.

O objetivo deste artigo é contribuir para que pesquisadores e desenvolvedores possam se beneficiar do estudo para projetos de pesquisa aplicada e desenvolvimento de aplicativos que utilizam API's de reconhecimento e sintetização de voz.

2. FUNDAMENTAÇÃO TEÓRICA
O reconhecimento e sintetização de voz visa facilitar o uso das entradas e saídas de informação. Por ser uma área de muito interesse, ela possui diversas aplicações, métodos e conteúdos. Nas próximas seções serão apresentados conceitos sobre as tecnologias de voz.

2.1 Processamento de Linguagem Natural
A linguagem natural é a mesma que, nós seres humanos, utilizamos diariamente para comunicar uns com os outros. Para compreender essa linguagem, o sistema computacional precisa ser capaz de processá-la e manipulá-la em diversos níveis, esse é o objetivo do processamento de linguagem natural, que é uma subárea da inteligência artificial. Conforme Coppin [3], inteligência artificial envolve utilizar métodos baseados no comportamento inteligente de humanos e outros animais para solucionar problemas complexos.

O processamento de linguagem natural está voltado, essencialmente, a três aspectos da comunicação em língua natural: som, estrutura e significado. Os cinco níveis de processamento e manipulação da linguagem, segundo Coppin [3], são: (i) fonologia; (ii) morfologia; (iii) sintaxe; (iv) semântica; e (v) pragmática.

Além dos níveis citados acima, o processamento de linguagem natural deveria aplicar algum conhecimento de mundo, ou seja, conhecer assuntos do mundo real, pois segundo Coppin [3], o objetivo final do processamento de linguagem natural seria ter um sistema com conhecimento de mundo o suficiente para ser capaz de se envolver em uma discussão com humanos sobre qualquer assunto.

2.2 Tecnologias de Voz
O Processamento da Linguagem Natural auxilia na manipulação que é necessária para o uso das tecnologias de voz. Essas tecnologias são utilizadas para melhorar a interação humana com computadores. Conforme Vilarinho et al. [4], os objetivos da Interação Humano–Computador (IHC) são produzir sistemas usáveis, seguros e funcionais.

Essas tecnologias fazem com que o usuário não dependa somente das mãos para alcançar o objetivo no dispositivo, ele poderá usufruir da sua voz, e com isso, tende a melhorar a acessibilidade para usuários com incapacidade motora, deficientes visuais, entre outros.

Segundo T4W Soluções [5], não é de hoje que se pode interagir com máquinas através da fala. Houve, porém, uma grande evolução e a tecnologia sofreu refinamentos que permitem uma reprodução mais próxima a nossa linguagem, possibilitando assim desenvolver aplicações mais avançadas.

As tecnologias de voz são divididas em duas categorias, o reconhecimento de voz, que é utilizado em comandos por voz, e o sintetizador de voz, utilizado para melhorar a acessibilidade dos deficientes visuais nos dispositivos.

2.2.1 Reconhecimento de voz
Reconhecimento de voz tem como objetivo converter a língua falada em texto, ou seja, reconhece a linguagem natural dos usuários. Segundo Pereira [6], "reconhecimento de voz habilita o computador a ouvir uma linguagem falada e determinar o que foi dito, ou seja, é o processo de converter em texto um som contendo palavras".

Os avanços nas técnicas de reconhecimento de voz indicam viabilidade da tecnologia em diversas aplicações, sobretudo em dispositivos móveis. Segundo Hearst [7], "alguns fatores contribuem para a demanda de interfaces com reconhecimento de voz nesses dispositivos: (1) por se tratar de um caminho natural ao uso da fala e (2) as interfaces de toque dificultam a escrita de textos longos".

Existem tecnologias de reconhecimento de voz maduras que não necessitam de treinamento como Apple Siri, Microsoft Cortana, Google *Now*. Três fatores contribuíram para essa maturidade: (i) o aumento da capacidade computacional; (ii) a quantidade de dados disponíveis para treinamento; e (iii) a evolução dos algoritmos de aprendizado de máquina [8]. Sendo possível identificar padrões complexos de voz com baixo tempo de resposta utilizando dispositivos portáteis e navegadores *web*. Isso sugere que o uso da voz deve aumentar significativamente a medida que o tempo de resposta e a precisão melhoram a ponto de atender as necessidades reais dos usuários [7].

Segundo Marangoni e Precipito [9], as etapas principais de um identificador típico de voz são: (i) projeto da gramática: as gramáticas definem as palavras que podem ser faladas por um usuário e pelos testes padrões em que podem ser faladas. Uma gramática deve ser criada e ativada para que um identificador saiba o que deve aguardar até escutar o áudio de entrada; (ii) processador de sinal: analisa as características do espectro (frequência) do áudio de entrada; (iii) reconhecimento do fonema: comparam os testes padrões do espectro aos testes padrões dos fonemas da língua que está sendo reconhecida; (iv) reconhecimento de palavras: comparam a sequência de fonemas prováveis de encontro às palavras e aos testes padrões das palavras especificadas pelas gramáticas ativas; e (v) geração de resultado: fornece a aplicação com a informação sobre as palavras que o identificador detectou no áudio de entrada. A informação do resultado será fornecida sempre uma vez que o reconhecimento de uma única sentença está completo, mas pode também ser fornecida durante o processo do reconhecimento. O resultado indica sempre a melhor suposição para o identificador de que o usuário tenha dito, mas pode também indicar suposições alternativas.

2.2.2 Sintetizador de voz

O objetivo de um sintetizador de voz, é converter o texto escrito em linguagem falada, ou seja, irá transformar o conteúdo do texto em áudio. O sintetizador recebe o texto na forma digital e transforma-o em ondas sonoras. Segundo Guilhoto e Rosa [10], "um programa de síntese de voz é útil nas situações em que o utilizador não pode desviar a atenção para ler algo ou não tem acesso ao texto escrito, seja porque a informação está distante ou porque o utilizador tem alguma deficiência visual".

Existem algumas limitações ao sintetizar a voz, segundo Marangoni e Precipito [9], "o *speech* sintetizadores (síntese da fala) pode cometer erros. As orelhas humanas são bem ajustadas para detectar estes erros, assim o trabalho cuidadoso de programadores pode minimizar erros e melhorar a qualidade da saída da fala".

Uma dessas limitações é referente ao som das pronúncias das palavras no idioma do usuário, e além disso, muitas API's possuem idiomas limitados. Outra limitação é na questão de sincronismo, da melodia, da pausa em vírgulas, isso deverá ser bem desenvolvido para não ficar um áudio muito artificial.

Segundo Melo e Pupo [11], entre os dispositivos de voz sintetizada, estão os programas que convertem texto em fala (ex.: DeltaTalk) e os leitores de tela com síntese de voz (ex.: Jaws *for* Windows, NVDA, Orca, *Virtual Vision*). Com o DeltaTalk, desenvolvido para sistema Windows, o usuário seleciona um texto e aciona a tecla virtual <F9> para que este seja "falado". Já os leitores de tela, além de converterem texto em fala, captam as informações textuais exibidas na tela do computador e as apresentam utilizando voz sintetizada.

3. LEVANTAMENTO BIBLIOGRÁFICO DE API'S

API é a sigla para *Application Programming Interface* ou, em português, Interface de Programação de Aplicativos. Segundo Ciriaco [12], "esta interface é o conjunto de padrões de programação que permite a construção de aplicativos e a sua utilização de maneira não tão evidente para os usuários". Na maioria das vezes a API se comunica com diversos outros códigos interligando diversas funções em um aplicativo.

Para a implementação das funcionalidades de comando por voz e resposta das informações em áudio, deverá ser localizada uma API que realiza essas funções, para integrar no aplicativo Alerta Brusque. Abaixo serão listadas algumas API's que poderão ser utilizadas para o desenvolvimento dessas funcionalidades.

3.1 A API Java *Speech*

O Java *Speech* é uma API que suporta as duas tecnologias de voz, reconhecimento e o sintetizador. As potencialidades da plataforma Java são um atrativo para o desenvolvimento em larga escala de aplicações. Segundo Marangoni e Precipito [9], para programadores desse tipo de aplicações de voz, a plataforma Java oferece uma alternativa atrativa com três características: (i) portabilidade, que possibilita que API's e as máquinas virtuais estejam disponíveis para uma larga variedade de plataformas de hardware e de sistemas operacionais e são suportados pela maioria dos navegadores; (ii) ambiente poderoso e compacto uma vez que a plataforma Java fornece aos programadores uma poderosa orientação a objeto, utilizando "*garbage collection*" que permite um desenvolvimento rápido e a confiabilidade melhorada; e (iii) rede segura que fornece uma rede de segurança robusta.

Segundo Pereira [6], a Sun definiu uma especificação que representa uma interface genérica para um motor de aplicação, a Java *Speech* API (JSAPI). A JSAPI não programa o motor em si, funcionando apenas como uma camada entre seus programas e os motores desenvolvidos por terceiros. Esses motores podem ser soluções em hardware ou software e podem estar na máquina local ou em um servidor, ou seja, são eles que capturam o som das falas ou sintetizam as palavras.

Ao contrário do que acontece com outras API's Java, não é fornecida uma programação de referência da Java *Speech* API. Em vez disso, no site oficial da JSAPI fornece uma lista de produtos e empresas que implementam a API, conforme a tabela 1 [6].

Tabela 1. Algumas implementações da JSAPI

Produto	Descrição
FreeTTS	Sintetizador de voz de código aberto totalmente escrito em Java.
IBM *Speech for* Java	Implementação da JSAPI baseada no IBM ViaVoice.
The Cloud Garden	Implementação desenvolvida para funcionar com engines baseados na SAPI da Microsoft. Funciona com os produtos IBM ViaVoice, *Dragon NaturallySpeaking* e outros.
Conversa *Web* 3.0	Um browser habilitado para funcionar com comandos de voz.

3.2 A API Google *Cloud Speech*

Google *Cloud Speech* API permite aos desenvolvedores converter áudio para texto através da aplicação de modelos de redes neurais poderosos para facilitar o uso da API. Esta API somente reconhece áudio, ou seja, a funcionalidade de sintetizar a voz não poderá ser realizada. Segundo a Google [13], *Speech* API reconhece mais de 80 línguas e variantes para apoiar a base de usuários global. Também pode-se filtrar conteúdo impróprio no resultado de texto. Por enquanto a Google está fornecendo acesso limitado para a pré-visualização do *Cloud Speech* API através do seu site de desenvolvedor. Os desenvolvedores podem tirar

proveito da API de graça, possivelmente a Google irá começar a cobrar pelo acesso em algum momento.

O *Cloud Speech* API acessa o mesmo conjunto de ferramentas que o Google usa para o reconhecimento de voz e ferramentas de comando de voz no Google *Search*, Google *Now*, e no teclado Google. Qualquer um que tenha usado a pesquisa de voz do Google sabe como o seu desempenho é rápido e preciso. Os desenvolvedores podem tirar proveito desta API, não só para capturar palavras faladas como texto, mas adicionar suporte para comandos por voz [14].

Conforme o Google [13], a API possui as seguintes características: (i) reconhecimento automático de fala, desenvolvido por redes de aprendizagem neural profunda para alimentar as suas aplicações como busca por voz ou transcrição da fala; (ii) vocabulário global, reconhece mais de 80 línguas e variantes com um extenso vocabulário; (iii) reconhecimento de streaming, retorna os resultados de reconhecimento parciais de imediato, assim que estiverem disponíveis; (iv) filtragem de conteúdo impróprio, filtrar conteúdo impróprio no resultado de texto; (v) em tempo real ou suporte *buffered* áudio, a entrada de áudio pode ser capturada pelo microfone de um aplicativo ou enviadas a partir de um arquivo de áudio pré-gravado; (vi) manuseamento de áudio com ruído, não há necessidade de processamento de sinal avançado ou amenizar o ruído do ambiente antes de enviar o áudio. O serviço pode conseguir lidar com áudio barulhento a partir de uma variedade de ambientes; e (vii) API integrada, os arquivos de áudio podem ser carregados na aplicação e, em versões futuras, integrado com o Google *Cloud Storag*e.

3.3 A API Bing *Speech*

A API da Microsoft converte áudio para texto, entende a intenção e converte o texto de volta como uma resposta natural. Essa API pode reconhecer o áudio vindo do microfone em tempo real, reconhecer o áudio vindo de uma fonte de áudio diferente em tempo real ou reconhecer o áudio a partir de um arquivo. Durante o processamento, como o áudio está sendo enviado para o servidor, os resultados de reconhecimento parciais poderão ser devolvidos.

Além disso, essa API também converte o texto para áudio. Segundo a Microsoft [15], quando os aplicativos precisam 'falar' de volta para os seus utilizadores, esta API pode ser usada para converter o texto que é gerado pelo aplicativo em áudio que pode ser jogado de volta para o usuário.

3.4 A API IBM Watson *Developer Cloud*

A IBM possui um supercomputador, cujo o nome é Watson. Esse supercomputador é utilizado, atualmente, para diagnósticos clínicos, e possui uma interação por voz. Com isso, a IBM liberou gratuitamente a API que realiza, no Watson, o reconhecimento e o sintetizador de voz.

Segundo a IBM [16], o serviço de reconhecimento de voz usa a inteligência artificial para combinar informações sobre gramática e estrutura da linguagem com o conhecimento da composição de um sinal de áudio para gerar uma transcrição exata. Ele usa as capacidades de reconhecimento de voz da IBM para converter a voz de vários idiomas em texto. Esse reconhecimento de voz tem suporte ao idioma português brasileiro, e inclui, a capacidade de detectar uma ou mais palavras-chave no áudio. O serviço pode ser acessado via conexão *WebSocket* ou API REST. Já o sintetizador de voz também possui suporte ao nosso idioma, e também é gratuito. Um diferencial é a possibilidade de controlar a pronúncia de palavras específicas.

3.5 A API AT&T *Speech*

Essa API foi lançada em 2012, pela empresa AT&T, e permitiu aos desenvolvedores adicionar funcionalidades de reconhecimento de voz para aplicações *web* e móveis. Conforme Wagner [17], a API desenvolvida pela AT&T é alimentada pelo motor de AT&T Watson de fala (nenhuma relação com a IBM Watson), um reconhecimento de voz e plataforma de compreensão da linguagem natural. A API utiliza processamento de linguagem natural para a compreensão de linguagem natural, reconhecimento de voz, a transcrição da fala, entre outros.

A AT&T *Speech* API consiste em três funções: voz para texto, voz para texto personalizado e, em texto para voz. Além disso, a AT&T oferece um site do desenvolvedor com uma documentação bem organizada, com aplicativos de demonstração, SDK's, *plug-ins*, fóruns, e muito mais.

3.6 A API W3C *Web Speech*

Web Speech API foi introduzida em 2012 pela comunidade W3C, organização que regulamenta a *web*. O principal objetivo é fazer com que os navegadores modernos reconheçam e compreendam a fala. Desde julho de 2015, o Chrome é o único navegador que implementou essa especificação, utilizando motores de reconhecimento de voz da própria Google.

A API abriu um novo mundo de oportunidades para novos aplicativos e novas funcionalidades de interação em aplicativos já existentes. Além disso, desde que o Google contribuiu com o seu próprio motor de reconhecimento de voz para apoiar essa API, tornou-se possível incorporar o melhor reconhecimento de voz existente no mercado de forma gratuita para o Chrome, mas não há garantia de que vai continuar a ser assim. Segundo a W3C [18], o *Web Speech* API visa permitir que desenvolvedores *web* forneçam, em um navegador *web*, reconhecimento de voz e funções de sintetizador de voz, que normalmente não estão disponíveis quando se utiliza reconhecimento de voz padrão ou software leitor de tela.

O *Web Speech* API é baseado em eventos, codificada com JavaScript. Chamadas para a API são manipuladas pelo usuário agente que por sua vez se encarrega de toda a comunicação com um serviço de reconhecimento de voz baseado na *web*. A arquitetura baseada em evento permite que os programas possam processar de forma assíncrona. Os eventos são também usados para relatar os resultados de reconhecimento de fala intermediária.

A API permite aos usuários gravar áudio a partir do microfone, que é então enviado através do protocolo HTTPS via solicitação POST para o serviço *web* de reconhecimento de fala. De acordo com Adorf [19] a *Web Speech* API é capaz de servir os usuários de todo o mundo, a API suporta diferentes idiomas. É possível definir o idioma para o reconhecimento de voz. De predefinição, o idioma é definido pelas configurações de localidade do dispositivo do usuário. O idioma tem de ser informado na configuração do aplicativo para o reconhecimento de voz. Por isso, deve ser conhecido antecipadamente qual idioma é esperado. Isto é, não é possível misturar livremente idiomas.

4. ESTUDO DE CASO: APLICATIVO ALERTA BRUSQUE

O levantamento bibliográfico das API's foi utilizado para localizar uma API que realize as funcionalidades de

reconhecimento e sintetização de voz, para integrar ao aplicativo Alerta Brusque. Este aplicativo não atende às diretrizes de acessibilidade da lei 5.296, e também, em situações de emergência o aplicativo deverá ser de fácil interação dos usuários. Essa implementação, das funcionalidades de voz, no aplicativo é um projeto de pesquisa de iniciação científica.

A região do Vale do Itajaí tem em seu histórico fenômenos naturais como enchentes, alagamentos e deslizamentos. O aplicativo Alerta Brusque, foi desenvolvido pensando em evitar grandes perdas para a sociedade, em parceria com a Defesa Civil de Brusque em apoio da prefeitura municipal de Brusque.

O aplicativo Alerta Brusque oferece a população acesso às informações geradas pelas estações automatizadas em tempo real, para os locais mais críticos da cidade de Brusque e cidades vizinhas. Com este aplicativo, disponível para as plataformas móveis Android e IOS, é possível visualizar o nível do rio Itajaí Mirim, bem como de pequenos rios da bacia, e até mesmo verificar a quantidade de chuva em diversos bairros. Os dados chegam ao sistema automaticamente sem a ação de pessoas. Cada estação de telemetria da cidade, realiza a transmissão de dados em intervalos regulares de 10 minutos. Atualmente as fontes de dados utilizadas pelo Alerta Brusque são: Defesa Civil de Brusque, CEOPS (Centro de Operação do Sistema de Alerta - FURB) e ANA (Agência Nacional das Águas).

Uma informação muito útil, presente no aplicativo, são as cotas das ruas. Esta informação, disponível sem a necessidade de acesso à *internet*, permite que os cidadãos tenham conhecimento sobre a situação de sua casa ou estabelecimentos comerciais em relação às cheias do rio. O aplicativo possui a cota de 1878 ruas. Para saber a cota da sua rua, o usuário deve digitar o nome da rua no campo de pesquisa do aplicativo.

Além das informações sobre rios e chuvas, o aplicativo permite o acesso ao número dos telefones úteis, em tempos de emergência. Os cidadãos podem fazer chamadas telefônicas diretamente do aplicativo e consultar sites de utilidade pública e notícias publicadas pela Defesa Civil do município de Brusque. Durante o estado de emergência, é possível consultar a localização de abrigos, e descobrir mais sobre casas em perigo devido a um deslizamento de terra, inundações, entre outros.

Este aplicativo tem acesso, em especial, a um total de 332 mil 433 habitantes atendendo num total de 4 municípios, com o monitoramento de 177 Km do rio Itajaí Mirim. Municípios a que são atendidos: Vidal Ramos (6 293 habitantes), Botuverá (4 864 habitantes), Brusque (119 719 habitantes) e Itajaí (201 557 habitantes).

O aplicativo foi desenvolvido no *framework* Cordova, sendo a linguagem de programação JavaScript a principal para o desenvolvimento das funcionalidades. Foi utilizada a tabela 2 para escolher qual API é mais adequada para este projeto.

Tabela 2. Resumo das funcionalidades das API's

API	Preço	Reconhecimento e Sintetização	Linguagem Suportada
Java *Speech*	Gratuito	Sim	Java
Google *Cloud Speech*	Gratuito	Não, apenas reconhecimento	Python, Node.js, C# e Java
Bing *Speech*	Parcialmente gratuito	Sim	C#
IBM Watson	Gratuito	Sim	Java
AT&T *Speech*	Gratuito	Sim	PHP, Ruby, Java e C#
W3C *Web Speech*	Gratuito	Sim	JavaScript

Entre essas API's mostradas na tabela acima, a escolhida foi a *Web Speech*. A escolha desta API é porque ela é de fácil integração por conta da compatibilidade da linguagem, ou seja, o aplicativo e a API são em JavaScript. Isso é muito importante, pois essa compatibilidade é essencial para uma boa e correta integração.

Além da fácil integração, essa API é de fácil instalação. Como o aplicativo é em Cordova, basta executar os comandos, dentro da pasta do projeto do aplicativo, para baixar os *plug-ins* da *Web Speech* API: "cordova plugin add cordova-plugin-tts" e "cordova *plugin add* https://github.com/macdonst/SpeechRecognitionPlugin".

O primeiro *plug-in*, realiza a sintetização de voz, e utiliza o AVSpeechSynthesizer no iOS e no Android utiliza o pacote android.speech.tts.TextToSpeech, ou seja, utiliza as melhores funções nativas para sintetização de voz. O segundo *plug-in* realiza o reconhecimento de voz, no Android este *plug-in* usa o pacote nativo android.speech.SpeechRecognizer, já no iOS é um pouco mais difícil de alcançar esta tarefa de modo que os desenvolvedores utilizaram o SDK *iSpeech* que pode ser testado gratuitamente por enquanto.

Após o *download* dos *plug-ins* da API, foi criada uma *view* (código fonte na Figura 1). Este código é o que será visto pelo usuário. Na tela irá possuir apenas um botão, que ao receber o clique, irá chamar a função "reconhecer()" do *controller*.

```
1  <div class="navbar-title">
2      <a href="#" ng-click="reconhecer()" style="margin-right: 20px; font-size: 33px;">
3          <span class="icon ion-ios-mic"></span>
4      </a>
5      Nível do Rio
6  </div>
```

Figura 1. Código fonte da View.

Após a criação da *view*, será necessário desenvolver o *controller* (código fonte na Figura 2). A função "reconhecer()" irá ouvir o que o usuário falar. Nesta função é criada uma instância para o objeto *SpeechRecognition*, e a partir disso é chamada a função "onresult()", que realiza a identificação do que foi dito, e passa por parâmetro para o serviço "dicionarioService".

```
$scope.reconhecer = function () {
    var recognition = new SpeechRecognition();
    recognition.lang = 'pt-BR';
    recognition.onresult = function (event) {
        if (event.results.length > 0) {
            $scope.recognizedText = event.results[0][0].transcript;
            $scope.$apply();
            matcher.compare($scope.recognizedText);
        }
        dicionarioService.dicionario($scope.recognizedText);
    };
    recognition.start().then($scope.reconhecer());
};
```

Figura 2 – Código fonte do *Controller*.

Dentro deste serviço possui a função "falar()". Essa função foi implementada conforme a Figura 3, e tem como objetivo, responder para o usuário o que foi recebido por parâmetro. É utilizado o plug-in Cordova chamado TTS.speak(), e é passado um objeto com algumas propriedades, e neste exemplo, é passado o texto (*text*), a linguagem do áudio (*locale*) e uma taxa que define o quão rápido o texto é falado (*rate*).

```
falar = function (termo) {
    TTS.speak({
        text: termo,
        locale: 'pt-BR',
        rate: 1.1
    }, function () {
        // Do Something after success
    }, function (reason) {
        // Handle the error case
    });
};
```

Figura 3 – Código fonte do Serviço (função falar).

Além disso, dentro do serviço, existe uma função chamada "dicionário()". Nesta função, primeiramente, será montado todo o dicionário (código fonte na Figura 4) de palavras possíveis e qual a função que será executada.

```
var dicionario_menu = {
        home: "index.html", início: "index.html", index: "index.html",
        telefones: "telefones.html", telefone: "telefones.html",
        abrigos: "abrigos.html", abrigo: "abrigos.html", abriu: "abrigos.html",
        links: "links.html", link: "links.html", likes: "links.html",
        notícia: "noticias.html", notícias: "noticias.html", news: "noticias.html",
        ruas: "ruas.html", rua: "ruas.html",
        informações: "sobre.html", sobre: "sobre.html", informação: "sobre.html",
        ajuda: "comandos.html", comandos: "comandos.html", comando: "comandos.html",
    },
    dicionario_cidade = {brusque: "Ponte Estaíada (Centro)", botuverá: "Botuverá"},
    dicionario_funcao = {chuva: "mmtel", rio: "riotel"},
```

Figura 4 – Código fonte do Serviço (função dicionário).

Após definir o dicionário é necessário ver se aquilo que o usuário falou está dentro desse dicionário (código fonte na Figura 5). Por exemplo, se o usuário falou "Eu quero ver os telefones", a função irá dividir a frase em palavras, e para cada palavra será verificado se faz parte de algum dicionário.

```
array = "", i = 0, cidade = "", funcao = "", menu = "";
array = texto.split(" ");
for(i = 0; i < array.length; i ++) {
    if (dicionario_cidade.hasOwnProperty(array[i])) {
        cidade = dicionario_cidade[array[i]];
    }
    if (dicionario_funcao.hasOwnProperty(array[i])) {
        funcao = dicionario_funcao[array[i]];
    }
    if (dicionario_menu.hasOwnProperty(array[i])) {
        menu = dicionario_menu[array[i]];
    }
}
```

Figura 5 – Código fonte do Serviço (função dicionário).

Quando a palavra fizer parte de algum dicionário, será realizado o que foi configurado para o mesmo fazer. Se não fizer parte de algum dicionário, o aplicativo irá responder por meio da função "falar()" que não foi encontrado o comando.

A Figura 6 representa como foi aplicado na tela, através da *view*, o botão de reconhecimento de voz. Esse botão é representado através de uma imagem de um microfone, e está localizado ao lado do menu.

Figura 6 – Tela do aplicativo Alerta Brusque.

O aplicativo Alerta Brusque será muito utilizado, pela sociedade, em situações de emergência, onde o rápido acesso às informações é fundamental. O reconhecimento e sintetização de voz, atuará para tornar esse acesso rápido, pois com alguns comandos de voz, o usuário será redirecionado para as principais telas do sistema.

5. DISCUSSÃO

Com o uso massivo de dispositivos móveis e tecnologias vestíveis, muitos softwares estão buscando novas formas de acessibilidade. As tecnologias de voz são uma excelente forma de acessibilidade para os usuários inexperientes, com incapacidade motora, deficientes visuais, entre outros, pois fazem com que o usuário não dependa somente das mãos para alcançar o objetivo no dispositivo, ele poderá usufruir da sua voz e audição. As

tecnologias de voz são um mercado em forte expansão, com muitas oportunidades, e ainda pouco explorado.

Para que uma API reconheça e sintetize a voz humana, a mesma necessita possuir um bom processamento de linguagem natural, ou seja, seguir os cinco níveis de processamento e manipulação da linguagem. Além do processamento, é necessário que a API disponibilize mecanismos para facilitar a sua utilização e integração com aplicações.

Nos testes e experimentos realizados, todas as API's descritas apresentaram boa performance. Entretanto, as que utilizam mecanismos da Google, além de reconhecer e sintetizar uma grande diversidade de idiomas, se destacaram pela aplicabilidade, pois apresentaram facilidade de integração.

Conforme demonstrado no estudo de caso, para integrar e configurar a API *Web Speech* no aplicativo, foi necessário apenas inserir duas linhas de código via Cordova. Além disso, a arquitetura baseada em eventos da API *Web Speech* é conveniente, pois permite que o programa que executa o reconhecimento possa ser interrompido a qualquer momento, aliviando o desenvolvedor de trabalho extra nas rotinas de manipulador de eventos [19].

6. CONCLUSÃO

Existem poucos trabalhos na literatura contemplando a utilização de APIs de reconhecimento e sintetização de voz em aplicativos móveis. Esta pesquisa pode ser considerada uma contribuição para o estado da arte, uma vez que pesquisadores e desenvolvedores poderão se beneficiar do estudo para projetos de pesquisa aplicada e desenvolvimento de aplicativos que utilizam API's de reconhecimento e sintetização de voz.

Nossas pesquisas em andamento estão direcionadas para o projeto e implementação de um dicionário *off-line* para o reconhecimento de voz, visto que todas as API's existentes necessitam de *internet* para poderem funcionar.

7. REFERÊNCIAS

[1] Casa Civil. 2004. *Decreto Nº 5.296 de 2 de dezembro de 2004*.

[2] Bernardes, W. and Kamimura, C. 2017. *Tecnologia de voz ganha espaço no setor de logística.* Disponível em: http://www.cgimoveis.com.br/logistica/tecnologia-de-voz-ganha-espaco-no-setor-de-logistica Acessado em: 24 jan. 2017.

[3] Coppin, B. 2010. *Inteligência artificial*. Rio de Janeiro: LTC.

[4] Vilarinho, L., Junior, A., Azevedo, L. A. and Matos, T. A. A. 2012. *Interface Homem-Máquina*. Disponível em: http://www.devmedia.com.br/interface-homem-maquina-revista-engenharia-de-software-magazine-47/24013. Acesso em: 25 jan. 2017.

[5] T4W Soluções. 2008. *Tecnologia Speech*. Disponível em: http://www.devmedia.com.br/artigo-net-magazine-51-tecnologia-speech/9336. Acesso em: 05 fev. 2017.

[6] Pereira, M. 2008. *Java Speech*. Disponível em: http://www.devmedia.com.br/artigo-java-magazine-04-javaspeech/8916. Acesso em: 25 fev. 2017.

[7] Hearst, M. A. 2011. *'Natural' search user interfaces*. Commun. ACM.

[8] Picheny, M. 2015. *Ibm watson now brings cognitive speech capabilities to developers*. Disponível em: https://developer.ibm.com/watson/blog/2015/02/09/ibm-watson-now-brings-cognitive-speech-capabilities-developers/. Acesso em: 19 jan. 2017.

[9] Marangoni, J. B. and Precipito, W. B. 2006. *Reconhecimento e sintetização de voz usando Java Speech*. Disponível em: http://faef.revista.inf.br/imagens_arquivos/arquivos_destaque/bjMnA2Zwc9685z8_2013-5-27-15-40-25.pdf. Acesso em: 14 fev. 2017.

[10] Guilhoto, P. J. S. and Rosa, S. P. C. S. 2002. *Reconhecimento de voz*. Disponível em: https://student.dei.uc.pt/~guilhoto/downloads/voz.pdf. Acesso em: 05 fev. 2017.

[11] Melo, A. M. and Pupo, D. T. 2010. *A Educação Especial na Perspectiva da Inclusão Escolar: livro acessível e informática acessível. Brasília: Ministério da Educação, Secretaria de Educação Especial*. Universidade Federal do Ceará.

[12] Ciriaco, D. 2009. *O que é API?* Disponível em: http://www.tecmundo.com.br/programacao/1807-o-que-e-api-.htm. Acesso em: 24 jan. 2017.

[13] Google. 2017. *Cloud Speech API*. Disponível em: https://cloud.google.com/speech/. Acesso em: 05 fev. 2017.

[14] Zeman, E. 2016. *Google to Offer Speech-to-Text API*. Disponível em: https://www.programmableweb.com/news/google-to-offer-speech-to-text-api/2016/03/24. Acesso em: 26 fev. 2017.

[15] Microsoft. 2017. *Bing Speech API*. Disponível em: https://www.microsoft.com/cognitive-services/en-us/speech-api. Acesso em: 24 fev. 2017.

[16] IBM. 2017. *Speech to Text*. Disponível em: http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/speech-to-text.html. Acesso em: 26 jan. 2017.

[17] Wagner, J. 2015. *Top 10 Machine Learning APIs: AT&T Speech, IBM Watson, Google Prediction*. Disponível em: http://www.programmableweb.com/news/top-10-machine-learning-apis-att-speech-ibm-watson-google-prediction/analysis/2015/08/03. Acesso em: 25 jan. 2017.

[18] W3C. 2012. *Web Speech API Specification*. Disponível em: https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html. Acesso em: 04 mar. 2017.

[19] Adorf, J. 2013. *Web Speech API*. Disponível em: http://home.in.tum.de/~adorf/pub/web-speech-api.pdf. Acesso em: 24 fev. 2017.