colloquium exactarum

RECONHECIMENTO DE VOZ NO AUXÍLIO À EDIÇÃO DE CÓDIGO FONTE EM JAVA

SPEECH RECOGNITION IN AID TO THE ISSUE OF SOURCE CODE IN JAVA

Francisco Saburo Misucochi Yui[1]; Silvio Antonio Carro[2]

Discente da Faculdade de Informática da UNOESTE[1]; Docente da Faculdade de Informática da UNOESTE[2];
E-mail: francisco_yui@hotmail.com[1]; silvio@unoeste.br[2];

RESUMO - Com a proposta no desenvolvimento de um aplicativo para a edição de código fonte, este trabalho descreve a metodologia de implementação de uma interface de software dotada de reconhecimento de voz para edição de código fonte na linguagem de programação Java. O aplicativo por meio do reconhecimento sintático de sentenças na linguagem de programação e técnicas que dão assistência à edição de código fonte propicia a escrita de código fonte de forma rápida e segura para pessoas com deficiência motora.
Palavras-chave: reconhecimento de voz; áudio digital; síntese da fala; Google Speech API; Apache Metaphone.

ABSTRACT - This work describes the methodology of implementation of a software interface endowed with voice recognition for Java source editing. The application through the syntactic recognition of sentences in the programming language and techniques that support the editing of source code provides the writing of source code quickly and safely for people with motor disabilities.
Keywords: voice recognition; digital audio; speech synthesis; Google Speech API; Apache Metaphone.

Recebido em: 03/05/2016
Revisado em: 05/10/2016
Aprovado em: 10/11/2016

1 INTRODUÇÃO

O reconhecimento por comandos de voz via aplicações computacionais é uma tecnologia que fascina o homem desde o surgimento dos computadores. A possibilidade de conversar com um computador, ditar um texto ou emitir comandos de voz faz parte de sistemas desde o surgimento da multimídia, mas somente nos últimos anos nos deparamos com a tecnologia relacionada ao áudio sendo utilizada de forma popular e acessível.

Encontramos aplicativos e bibliotecas que permitem aos usuários, ditarem palavras e frases ao computador onde aplicativos as convertam em texto ou executem funções específicas de controle.

Pessoas com deficiências motoras, impedidas de digitar também adotam sistemas de reconhecimento de voz para desempenhar atividades profissionais ou de entretenimento.

O principal foco do projeto é o estudo da síntese, identificação e o reconhecimento da voz e sua utilização em um aplicativo de edição de código fonte direcionado por comandos de voz. O aplicativo ainda oferece recursos de assistência na digitação de comandos e identificadores na linguagem de programação o que torna o processo de codificação bastante rápido.

A proposta de um projeto com estas características justifica-se pelo fato de ser um tema pouco explorado na área para programação. A linguagem de programação a ser codificada foi à linguagem Java, por questões de popularidade e de uso livre.

Este trabalho esta organizado no seguinte modo: no capítulo 2 são descritas as principais APIs para o processamento da voz. O capítulo 3 descreve outros trabalhos relacionados. O capítulo 4 mostra os recursos que fizeram parte do desenvolvimento do sistema. O capítulo 5 apresenta os resultados experimentais. Finalmente, o capítulo 6 conclui o trabalho e sugere pesquisas futuras.

2 APIS RELACIONADAS AO RECONHECIMENTO DA VOZ

Várias APIs foram disponibilizadas para possibilitar o uso do áudio e de seu processamento em um programa de computador. Muitas delas são dependentes da arquitetura do computador ou de softwares externos, portando não são fáceis de utilizar e suscetíveis a problemas externos, como software e hardware obsoletos ou não compatíveis. Outras pertencem a soluções proprietárias e são cobradas pelo uso, dentre essas podemos destacar a Microsoft, Nuance, Dragon e IBM Via Voice. Outro problema encontrado é a ausência de soluções que interpretam o

idioma português, mais precisamente o português brasileiro.

A seguir, foram relacionadas às APIs estudadas no desenvolvimento do projeto e citadas os prós e contras para sua adoção.

2.1 JSAPI (JAVA SPEECH APPLICATION PROGRAMMING INTERFACE)

A JSAPI (ORACLE, 2015) desenvolvida pela Sun Microsystems, Inc é uma interface de programação de aplicativos Java para suporte multiplataforma de comando e controle, sistemas de reconhecedores de voz e síntese.

Ela foi desenvolvida para trabalhar com o apoio de outras APIs como FreeTTS, IBM's "Speech for Java", The Cloud Garden, Lernout & Hauspie's TTS, Conversa Web 3.0, Festival, Elan Speech Cube.

O uso desta API foi descartada por não garantir o funcionamento em sistemas operacionais mais recentes como o Windows 8 e Windows 10.

2.2 CMUSPHINX

A CMUSphinx (CMUSPHINX, 2015) é um grupo de sistemas de reconhecimento de voz desenvolvidas na Carnegie Mellon University. Como a série de reconhecimento de voz: Sphinx 2-4 e um treinador de modelo acústico: SphinxTrain. A versão testada foi a Sphinx4, essa API é uma biblioteca puramente feita em Java para reconhecimento de voz. Ela não necessita de um treinamento para usar, porém devemos cadastrar as palavras com suas fonéticas em seu banco de palavras.

Ela tem suporte para varias línguas como inglês (US/UK), francês, mandarim, alemão, holandês e russo, porém foi descartada por não ter suporte em português.

2.3 MICROSOFT SPEECH SDK C#

A Microsoft Speech SDK (MICROSOFT, 2015) desenvolvida pela Microsoft e faz a síntese e reconhecimento de voz em C#. A versão testada necessitava de treinamento para poder utilizar, sendo necessário cadastrar as palavras na forma ditada.

Foi descartado por precisar cadastrar as palavras por meio da voz, assim deixando o sistema fixo para um narrador e ser um recurso pago.

2.4 GOOGLE SPEECH API

A bibioteca Google Speech API (MEY, 2016) foi desenvolvida pela Google e destina ao reconhecimento de voz em vários idiomas. Não necessita de um treinamento para usar, mas é preciso o acesso à internet e uma chave para liberação. Apesar de ser uma tecnologia proprietária e paga, há uma versão livre, limitada a 50 acessos por dia. Para obter a chave de acesso é necessário se cadastrar no website disponível em:

http://www.chromium.org/ developers/how-tos/api-keys.

A Google Speech API foi utilizada no desenvolvimento do projeto.

3 TRABALHOS RELACIONADOS

Foram pesquisadas e estudadas diversas bibliotecas que poderiam gerar subsídios para o desenvolvimento do projeto. Neste capítulo foram relacionadas as bibliotecas que efetivamente foram utilizadas no desenvolvimento do protótipo criado no presente estudo.

3.1 RECONHECIMENTO DE VOZ COM SPHINX-4

Um projeto para uma aplicação com reconhecimento de voz foi desenvolvido no Eclipse, API Sphinx-4 e CloudGarden, produzido por Gabriel Francisco Silva (SILVA, 2013).

O aplicativo reconhece a fala a partir de um banco de dados com a gramática e escreve em texto. O software grava a fala do usuário, compara com sua gramática e transforma em texto.

3.2 RECONHECIMENTO E SINTETIZAÇÃO DE VOZ UTILIZANDO JAVA SPEECH API - UCSAL

O trabalho foi um dicionário de voz. Foi desenvolvido no Eclipse e JSAPI, utilizando bibliotecas como: Text to Speech (TTS), IBM ViaVoice e Java Speech API, desenvolvido por Alã Assis e Hemerson Santos (BURLAMAQUI, 2015).

O aplicativo captura a voz, verifica no seu banco de dados e retorna para usuário a palavra com seu significado, antônimo e sinônimo em texto e sintetiza o texto retornado.

3.3 J.A.R.V.I.S. (JUST A RELIABLE VOCAL INTERPRETER & SYNTHESIZER) API

O J.A.R.V.I.S. (KUSA, 2015) é um conjunto de APIs utiliza os mecanismos de fala pela Google. Desenvolvida por Luke Kusa e feito em Java, inclui um reconhecedor, sintetizador, e um utilitário de captura de microfone.

O projeto utiliza várias APIs como: Captura microfone API, um reconhecedor de voz usando o serviço reconhecedor do Google, um sintetizador de voz usando o serviço de sintetizador do Google, FLAC API (converter arquivos WAVE do microfone para FLAC) e tradutor do Google Translator. (Kusa)

3.4 APACHE METAPHONE

O Apache Metaphone é um algoritmo fonético para a geração de índices de como as palavras são pronunciadas. Portanto as palavras com sons parecidos geram índices iguais.

Foi desenvolvido por Lawrence Philips, a versão original (para a língua inglesa) foi implementada em PHP.

A versão usada no programa foi para a língua portuguesa e a desenvolvido em Ruby (VÁRZEA PAULISTA, Prefeitura Municipal, 2015).

A API recebe a palavra e substitui suas consoantes por suas fonéticas, assim gerando a fonética da palavra.

4 O PROJETO FJE

Como forma de testar e validar as técnicas de reconhecimento de voz disponíveis foi desenvolvido um aplicativo denominado de FJE, acrônimo de Fala Java Editor.

A linguagem Java para a edição de código fonte foi utilizada, pois é a mais utilizada como a primeira linguagem a ser aprendida em cursos de programação e como visto na Figura 1 abaixo é a que possui maior popularidade.

Programming Language	Ratings	Change
Java	20.956%	+4.09%
C	13.223%	-3.62%
C++	6.698%	-1.18%
C#	4.481%	-0.78%
Python	3.789%	+0.06%
PHP	2.992%	+0.27%
JavaScript	2.340%	-0.79%
Ruby	2.338%	+1.07%
Perl	2.326%	+0.51%
Visual Basic .NET	2.325%	-0.64%

Figura 1. Ranking das linguagens mais populares usadas para programação do Tiobe.
Fonte: (TIOBE, 2016).

O aplicativo foi organizado em módulos, os quais serão discutidos nesse capítulo.



Figura 2. Módulos para captura e reconhecimento do FJE.

4.2 CAPTURA DA VOZ

Para a captura de voz foi utilizada a biblioteca “Microphone Capture API” com recursos para analisar a frequência do microfone e armazenar somente as informações relacionadas à fala, suprimindo os ruídos e os barulhos do ambiente. As informações capturadas são convertidas em arquivo padrão FLAC (PAYTON, 2014) e enviado ao WebService do Google Speech API.

Há dois modos para acionar a gravação de voz: o automático e o manual.

No modo automático, uma *thread* de captura é acionada a todo o momento que se detecta mudanças na amplitude e volume do som ambiente. Quando detectado que as mudanças se assemelham a pronuncia de palavras ou frases, é iniciado o processo de armazenamento do áudio.

No modo manual, a captura é realizada por meio de uma opção da interface do aplicativo.

4.3 GOOGLE DUPLEX SPEECH API

O Google Duplex Speech API favorece a conexão do aplicativo com o WebService do Google Speech API, sendo responsável pelo controle de *upstream* e *downstream* de áudio com o WebService. São necessários alguns parâmetros para que possa efetivar as transações: chave de serviço, linguagem da gravação, tipo do áudio e suas especificações, tipo do app, seu output no formato json e cliente.

4.4 RESPOSTA DO GOOGLE

As Figuras 3 e 4 representam respectivamente a chamada ao WebService e a resposta efetuada pelo serviço. Na relação de palavras retornadas no processo de reconhecimento, denominadas de palavras candidatas há uma ordenação do maior para o menor grau de confiabilidade.

```
public void onResponse(GoogleResponse gr) {
    System.out.println("Google pensa que você disse: " + gr.getAllPossibleResponses());
    System.out.println("com "
            + ((gr.getConfidence() != null) ? (Double.parseDouble(gr.getConfidence()) * 100) : null)
            + "% confiança.");
```

Figura 3. Código Fonte do quando recebe alguma resposta do Google.

No algoritmo utilizado foi decidido considerar todas as palavras candidatas e compará-las com o conjunto de identificadores do FJE. Caso não ocorra o reconhecimento, um segundo processo é executado o qual envolve a comparação fonética da biblioteca Apache Metaphone.

```
Google pensa que você disse: [novo",
"confidence":0.91206837}, {"transcript":"jogo"},
{"transcript":"logo"}, {"transcript":"Globo"},
{"transcript":"Google] com 91.206837% confiança.
```

Figura 4. Resposta do Google Speech API.

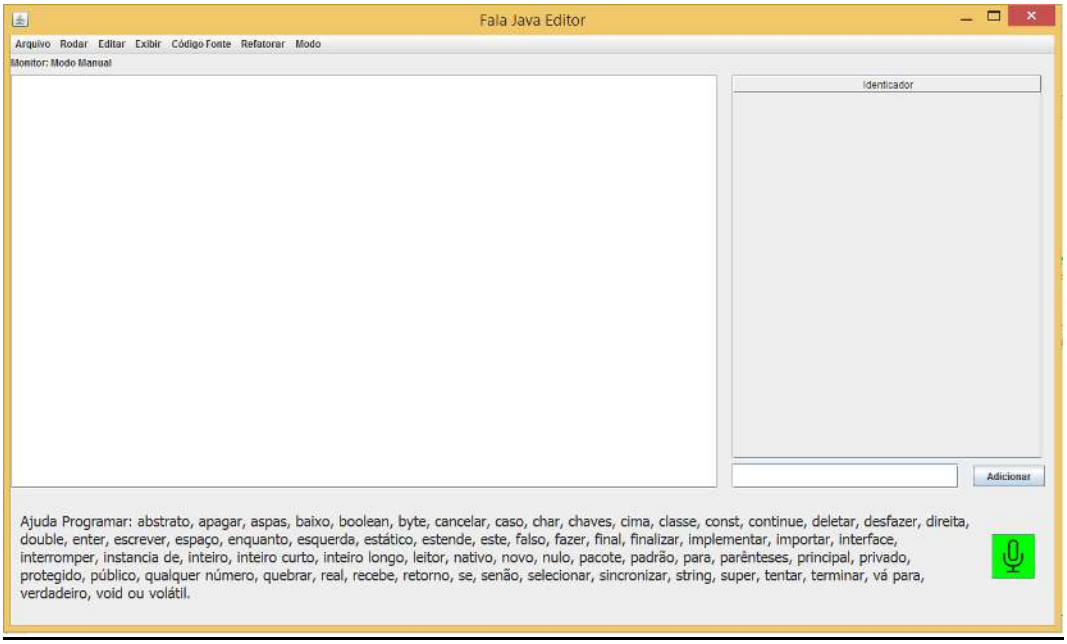


Figura 5. Interface principal do FJE.

4.5 INTERFACE DO APLICATIVO

A Figura 5 apresenta a interface principal do FJE. A interface é segmentada em três blocos, identificados como: Menu, Programa e Identificar.

Na figura, o ícone na cor verde sinaliza que o sistema esta esperando alguma palavra ser pronunciada, se mudar para vermelho significa que se iniciou o processo de captura e gravação, seguindo pelo envio e resposta do Google Speech API.

4.6 O MENU

O módulo denominado de Menu oferece acesso às opções tradicionais em um editor de código fonte: Organizado em uma barra de menus, oferece as opções Arquivo, Rodar, Editar, Exibir, Código fonte, Refatorar e Modo. Cada menu possui itens pertinentes à suas ações.

Novo:

- Abrir
- Sair

Rodar:

- Executar

Editar

- Desfazer
- Copiar
- Colar
- Recortar
- Deletar

Exibir:

- Editores

Código fonte:

- Formatar

Refatorar:

- Renomear
- Mover
- Exclusão
- Retirar
- Colocar

Modo:

- Manual
- automático

4.7 PROGRAMAR

O Módulo Programar é utilizado para a edição de um código fonte. O usuário pode falar ou escrever o texto e manipular o cursor do editor livremente por voz. As palavras reservadas envolvem comandos e expressões da linguagem e controles de edição. As seguintes palavras chaves são identificadas: abstrato, apagar, aspas, baixo, boolean, byte, cancelar, caso, char, chaves, cima, classe, const, continue, deletar, desfazer, direita, double, enter, escrever, espaço, enquanto, esquerda, estático, estende, este, falso, fazer, final, finalizar, implementar, importar, interface, interromper, instancia de, inteiro, inteiro curto, inteiro longo, leitor, nativo, novo, nulo, pacote, padrão, para, parênteses, principal, privado, protegido, público,

qualquer número, quebrar, real, recebe, retorno, se, senão, selecionar, sincronizar, string, super, tentar, terminar, vá para, verdadeiro, void ou volátil.

4.8 IDENTIFICAR

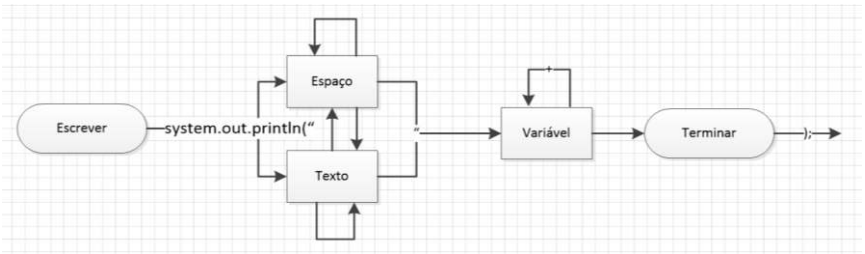
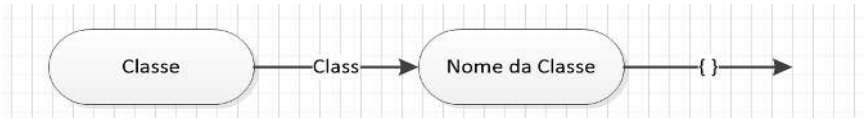
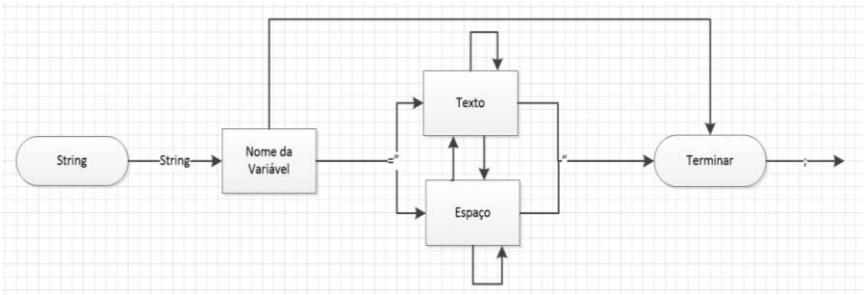
O módulo Identificar é caracterizado por um conjunto de identificadores gerados durante a edição de um código fonte. Envolve nomes dados às variáveis, classes e demais estruturas da linguagem Java. As palavras identificadas foneticamente são: novo, cima, baixo, selecionar, adicionar ou limpar.

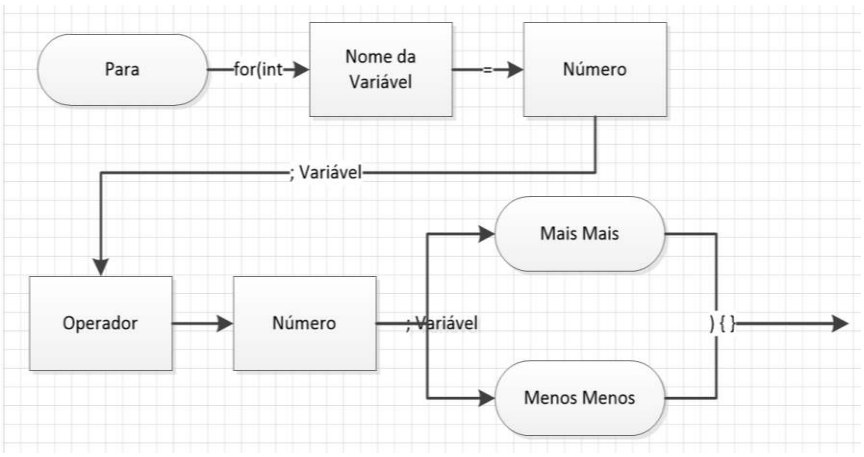
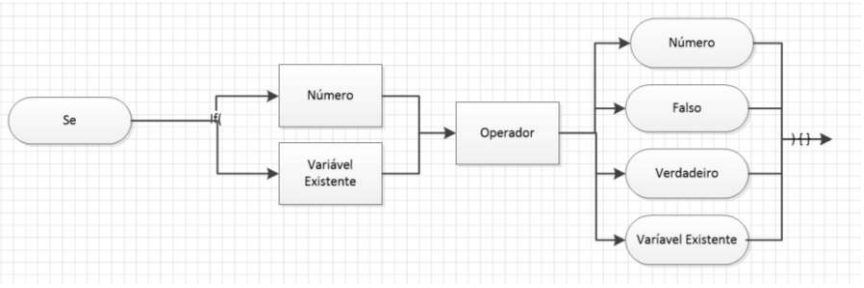
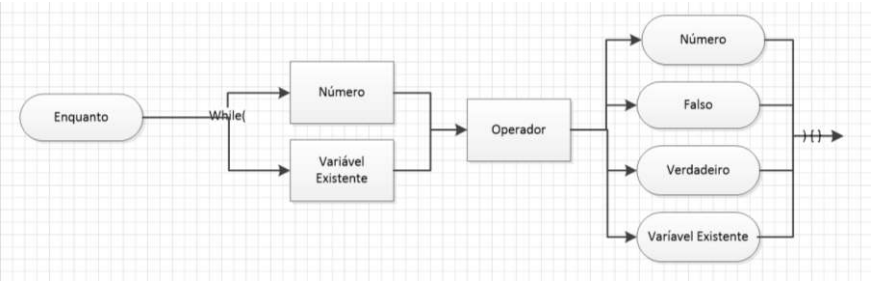
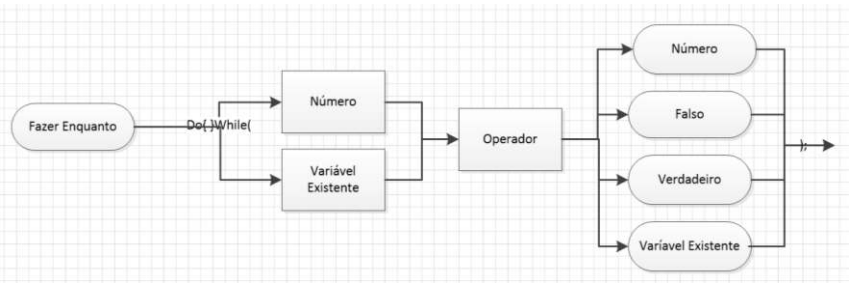
As variáveis, classes ou outros identificadores da linguagem adicionados podem ser utilizados no modo Programar, falando-as ou selecionando-as com o mouse.

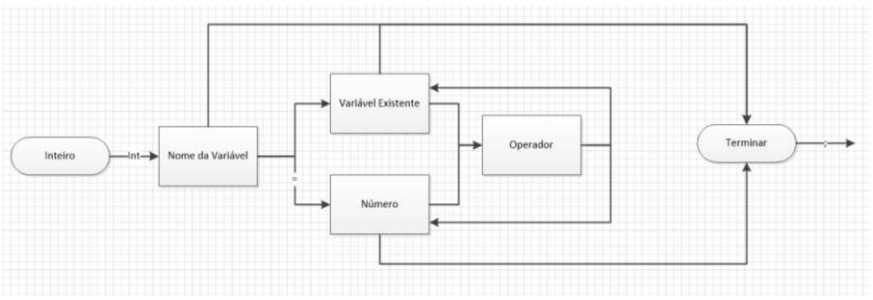
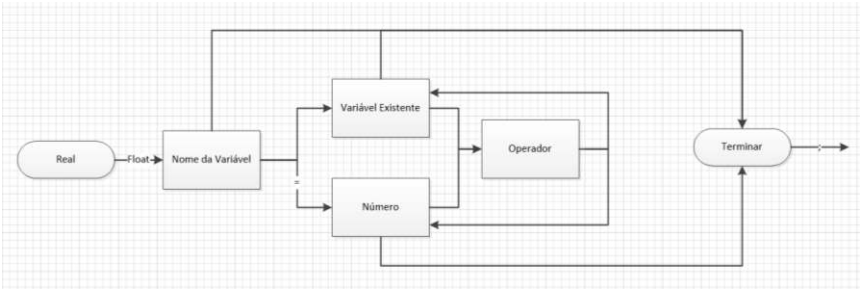
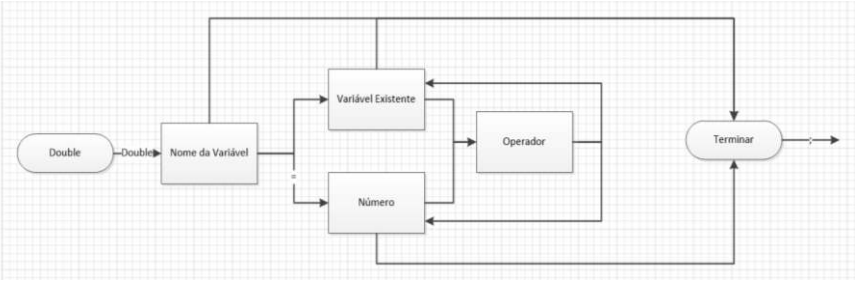
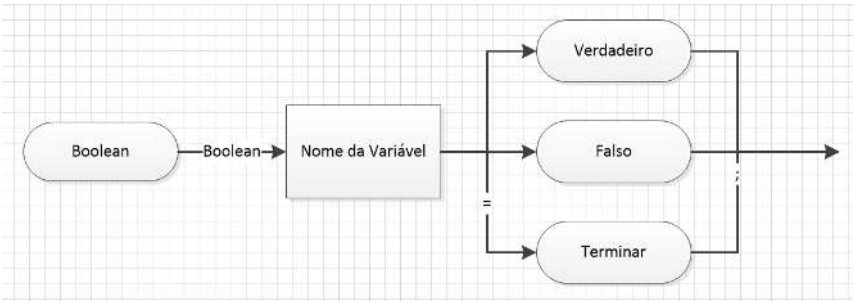
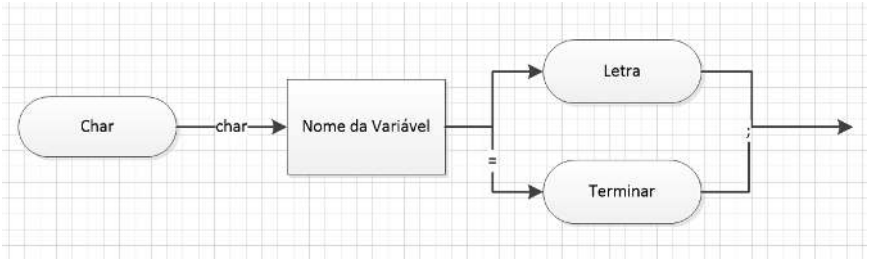
4.9 REGRAS GRAMATICAIS

As regras gramaticais são caracterizadas por uma sequencia de passos que originam uma expressão completa na linguagem. O beneficio que ela trás ao usuário é a escrever, não cometer erros e facilitar a reconhecimento de voz. A seguir uma Tabela 1 com cada passo das regras:

Tabela 1. Regras Gramaticais.

Comando Principal	Comando Secundário	Diagrama
Escrever	Espaço Variável Terminar Texto	
Classe	Nome da Classe	
String	Nome da Variável Terminar Espaço Texto	

<table>
<tr><td rowspan="5">Para</td><td>Nome da Variável</td><td rowspan="5">
</td></tr>
<tr><td>Número</td></tr>
<tr><td>Operador (Maior, Menor, Maior Igual, Menor Igual, Igual, Diferente)</td></tr>
<tr><td>Mais Mais</td></tr>
<tr><td>Menos Menos</td></tr>
<tr><td rowspan="5">Se</td><td>Variável Existente</td><td rowspan="5">
</td></tr>
<tr><td>Operador (Maior, Menor, Maior Igual, Menor Igual, Igual, Diferente)</td></tr>
<tr><td>Falso</td></tr>
<tr><td>Verdadeiro</td></tr>
<tr><td>Número</td></tr>
<tr><td rowspan="5">Enquanto</td><td>Variável Existente</td><td rowspan="5">
</td></tr>
<tr><td>Operador (Maior, Menor, Maior Igual, Menor Igual, Igual, Diferente)</td></tr>
<tr><td>Falso</td></tr>
<tr><td>Verdadeiro</td></tr>
<tr><td>Número</td></tr>
<tr><td rowspan="5">Fazer enquanto</td><td>Variável Existente</td><td rowspan="5">
</td></tr>
<tr><td>Operador (Maior, Menor, Maior Igual, Menor Igual, Igual, Diferente)</td></tr>
<tr><td>Falso</td></tr>
<tr><td>Verdadeiro</td></tr>
<tr><td>Número</td></tr>
</table>

Inteiro	Nome da Variável Terminar Variável Existente Número Operador (Somar, Multiplicar, Dividir, Subtrair)	
Real	Nome da Variável Terminar Variável Existente Número Operador (Somar, Multiplicar, Dividir, Subtrair)	
Double	Nome da Variável Terminar Variável Existente Número Operador (Somar, Multiplicar, Dividir, Subtrair)	
Boolean	Nome da Variável Terminar Verdadeiro Falso	
Char	Nome da Variável Terminar Letra	

4.11 ATALHOS

Os atalhos foram criados para facilitar a geração de código fonte, seu uso agiliza significativamente a geração de código fonte e suprime muitas vezes a demora gerada durante o processo de soletração individual dos códigos da linguagem. A seguir uma tabela com os atalhos disponíveis e seus respectivos significados.

Tabela 2. Atalhos.

Atalho	Significado
Leitor	scanner leitor = new scanner(system.in); \n
Principal	public static void main(string[] arg){ \n \n} \n
Tentar	try{ \n\n} \n catch(exception e){}
Aspas	""
Parênteses	()
Chaves	{\n}\n
Terminar	; \n
Ler inteiro	= leitor.nextint();\n
Ler real	= leitor.nextfloat();\n
Ler double	= leitor.nextdouble();\n
Ler string	= leitor.nextline(); \n

5 EXPERIMENTOS E RESULTADOS

Os experimentos foram realizados no sentido de testar a viabilidade e a usabilidade de oferecer uma ferramenta destinada ao reconhecimento e edição de código fonte pela voz. Os testes foram realizados em um ambiente com poucos ruídos externos e por pessoas com pouca ou nenhuma experiência no uso do aplicativo. Os usuários tiveram o conhecimento dos atalhos oferecidos (Tabela 2). Nos testes foram utilizadas duas pessoas do sexo feminino e duas do sexo masculino. Cada usuário foi convidado para soletrar um código fonte de exemplo e na Tabela 3 demonstra os resultados da quantidade de comandos que cada usuário teve que falar, erros de reconhecimento, acertos diretos (palavra que não teve que repetir), palavras pronunciadas, tempo com FJE, tempo ao digitar o código e vezes teve desfazer. Na Figura 6 temos o código usado para o teste.

```
class programa
{
        static public void main(string[] args)
        {
                boolean pode = false;
                for(int tecla=0;tecla<5;tecla++)
                {
                        string nome;
                        scanner leitor = new scanner(system.in);
                        nome = leitor.nextline();
                        int idade;
                        idade = leitor.nextint();
                        int dias = idade*365;
                        if(dias>500)
                        {
                                pode=true;
                                System.out.println("Idade em dias"+dias);
                        }
                }
        }
}
```

Figura 6. Código usado no teste.

Tabela 3. Resultados do experimento.

Usuário	Quantidade de comandos	Erros	Acertos Diretos	Palavras pronunciadas	Tempo com FJE	Tempo sem FJE	Desfazer
1	40	0	33	49	8:24.50	3:57.00	0
2	40	0	36	46	6:30.98	2:06.65	0
3	40	1	33	71	9:52.01	2:10.54	1
4	40	0	31	50	6:07:76	2:33.58	0

O sistema proposto utilizando as APIs do Google Speech API e Apache Metaphone podem resolver problemas relacionados à inserção digital e profissional de indivíduos com necessidades motoras. A identificação e edição via voz se torna real e com usabilidade bastante satisfatória.

A morosidade inicialmente encontrada no processo de captura e reconhecimento dos comandos individuais pode ser amenizada com o uso de atalhos e demais facilidades oferecidas pelo aplicativo. Ainda assim, o experimento constata que na digitação convencional o usuário desempenha a edição em aproximadamente um terço do tempo. A latência apresentada é ocasionada principalmente pela demora no processo de transação com o WebService do Google Speech. A utilização de recursos locais com certeza agilizaria o processo, mas infelizmente nenhuma biblioteca ou recurso local possui o mesmo grau de precisão oferecida pelo serviço do Google.

6 CONSIDERAÇÕES FINAIS

Como trabalhos futuros espera-se a criação de demais atalhos para facilitar a edição do código fonte, melhorar o algoritmo de captação automática das palavras soletradas e desenvolver o módulo de compilação e execução do código fonte.

7 AGRADECIMENTOS

Gostaria de agradecer a Universidade do Oeste Paulista por disponibilizar os recursos para pesquisa e desenvolvimento deste projeto.

Agradeço também ao professor orientador Silvio Antonio Carro pelas orientações do desenvolvimento da pesquisa.

8 REFERÊNCIAS

BURLAMAQUI, A.M.F. **Java Speech – Tutorial passo a passo.** 2015. Disponível em: <http://aquilesburlamaqui.wikidot.com/javaspeech>. Acesso em: 07 set. 2015.

CMUSPHINX. **Open Source Speech Recognition Toolkit.** 2015. Disponível em: <http:// cmusphinx.sourceforge.net/>. Acesso em: 6 dez. 2015.

KUSA, L. **J.A.R.V.I.S. (Java-Speech-API**). 2015. Disponível em: <https://github.com/lkuza2/java-speech-api>. Acesso em: 12 nov. 2015.

MEY, G. **Reverse Engineering Google's Speech To Text API.** 2016. Disponível em: <https://github.com/gillesdemey/google-speech-v2/>. Acesso em: 30 mai. 2016.

MICROSOFT. **Microsoft Speech Platform SDK 11 Documentation.** 2015. Disponível em: <https://msdn.microsoft.com/en-us/library/dd266409%28v=office.14%29.asp>. Acesso em: 6 dez. 2015.

ORACLE. **Java Speech API Frequently Asked Questions.** 2015. Disponível em: <http://www.oracle.com/technetwork/pt/java/jsapifaq-135248.html>. Acesso em: 6 dez. 2015.

PAYTON, T. **Google Speech API.** 2014. Disponível em: <http://blog.travispayton.

com/wp-content/uploads/2014/03/Google-Speech-API.pdf>. Acesso em: 6 dez. 2015.

SILVA, G.F. **We have science.** 2013. Disponível em: <http://wehavescience.com/2012/11/03/reconhecimento-de-voz-com-sphinx-4/>. Acesso em: 09 nov. 2013.

TIOBE. **TIOBE Index for May 2016.** 2016. Disponível em: <http://www.tiobe.com/tiobe_index>. Acesso em: 30 mai. 2016.

VÁRZEA PAULISTA. Prefeitura Municipal. **pt_metaphone().** 2015. Disponível em: <http://informatica.varzeapaulista.sp.gov.br/metaphone/>. Acesso em: 12 nov. 2015.