1 INTRODUÇÃO

A base da programação é ensinar máquina a realizar determinada tarefa, para que isso seja possível é necessário que o usuário comunique-se com a máquina, numa linguagem que ela entenda, a linguagem de programação. A construção da linguagem de programação envolve ações definidas em passos que se deseja que a máquina execute, a estes passos damos o nome de algoritmos. Para isso pode-se dizer que independente do paradigma utilizado no processo de desenvolvimento ele envolve algoritmos (MEDEIROS, 2015).

Algoritmos podem não necessariamente serem computacionais, já que são um conjunto de passos finitos. Analogias simplistas são os ingredientes e modo de preparo de uma receita ou instruções para uma tarefa do cotidiano como a troca de um pneu furado em um automóvel (MEDINA; FERTIG, 2006). Porém se tratando de computação deve ter maior formalidade, seguindo uma linguagem pois diferente do ser humano uma máquina não é tão interpretativa. Desconsideradas aqui inteligências artificiais que são desenvolvidas por algoritmos para terem comportamento de compreensão humana. Isto pode ser analogamente algo como não entender idioma estrangeiro e as instruções estarem nesta linguagem. Porém em informática isso tem que ser mais especificado com palavras ou caracteres predeterminados de início e fim, de blocos, repetições, etc (DERSHEM; JIPPING, 1990).

Sendo o berço mais popular da informática os Estados Unidos da América e seus criadores em muitos casos dessa origem ou de países com mesmo idioma as primeiras linguagens de computação e a maioria delas são em inglês (SEBESTA, 2009).

Isso muda quando falamos do Portugol que possui variações, mas resumidamente sua origem vem como uma tradução de uma linguagem de programação simples e comum, com seus termos no idioma nativo do Brasil.

Tendo como foco o aprendizado de algoritmos para uma boa fundamentação de futuros programadores ou profissionais da área, e muitas vezes a facilidade com o inglês não é algo comum em nosso país, a linguagem Portugol se torna um artifício vantajoso (JESUS et al., 2004).

1.1 OBJETIVOS DO TRABALHO

1.1.1 Objetivo Geral

Apresentar o conceito de ferramenta computadorizada para se tornar referência em criação, edição, visualização e interpretação de algoritmos em pseudolinguagem, como alternativa às existentes. Sendo seu diferencial o acesso nativo a partir de navegadores *Web* modernos.

Inclusive acessível quando uma conexão com a internet esteja indisponível.

1.1.2 Objetivos Específicos

Visando atingir o objetivo geral em sua totalidade lista-se a seguir objetivos específicos de modo sucinto:

- Desenvolver o protótipo, sendo seu código fonte aberto e disponível para que outros possam manter, ampliar e evoluir a aplicação;
- Fazê-lo compatível com a sintaxe desenvolvida na Universidade Estácio de Sá (UNESA) a UNESA *Algorithmic Language* (UAL). E mínimo produto viável para a primeira lição de ensino aprendizagem;
- Permitir o acesso via navegadores *Web* modernos, sem necessidade de instalações complementares;
- Aplicar aprimoramentos introduzidos na versão 5 da *HyperText Markup Language* (HTML) para uso sem conexão com a internet.

1.2 JUSTIFICATIVA DO TRABALHO

Praticar em *software* à abstração em papel auxilia e muito no processo de ensino e aprendizagem, em algoritmos além do pensamento lógico por etapas já pode ser desenvolvido e descrito para o repasse a um computador. “O Aprendizado de algoritmos não é uma tarefa muito fácil, só se consegue através de muitos exercícios” (LOPES; GARCIA, 2002, p. 1).

Para esse ensino é essencial conhecer dois conceitos principais lógica de programação e algoritmos. Primeiramente é apresentado um paralelo de situações do cotidiano com algoritmos, ao efetuar comparações de instruções passo a passo realizadas diariamente, como fritar um ovo ou trocar um pneu de automóvel. A seguir inicia-se a aplicação de linguagem estruturada em português para descrição dos algoritmos. Em alguns casos, utiliza-se de linguagens em inglês, mo por exemplo, Pascal. Há professores que preferem usar pseudo linguagem em português somente “em papel”, iniciando a codificação diretamente em linguagem C. Tem-se inclusive, também “em papel”, de analisar e interpretar os passos do algoritmo para saber quais serão os valores em memória e a saída na tela.

Simultaneamente, com a prática escrita, sem software específico, pode-se simular os mesmos algoritmos em interpretador ou compilador. Sendo em um interpretador sua construção dividida em partes, cada uma com uma função específica. Geralmente identifica-se: o analisador léxico, o analisador sintático, o analisador semântico, e o resultado de saída. Caso seja um

compilador há, ao invés de resultado de saída, o gerador de código, que transforma o programa em linguagem de máquina composta de 0s e 1s (DELAMARO, 2004).

Diante do exposto, este trabalho justifica-se pelo empenho em facilitar o processo de aprendizagem de algoritmos, desenvolvendo um um programa em navegador via Internet, fazendo com que seja desnecessária a instalação do programa, podendo ser acessado a partir de qualquer computador, com o objetivo que os exercícios sejam resolvidos com maior praticidade

1.3 LIMITAÇÕES DO TRABALHO

Limitou-se o trabalho numa imposição que vem preservar que o tempo disposto seja viável para sua execução.

Por seguir um processo bem diferenciado de construção optou-se por sua mínima execução. Permitindo apenas algorítmos muito simples pois um dos focos é a disponibilidade independente de sistema operacional ou instalações.

Outra limitação foi a não possibilidade de gravar e manter executáveis da saída gerada do algoritmo. Devido ao entendimento que é ponto não essencial para a prática do conhecimento e que uma possível solução no modelo pretendido seria dispendioso e demorado.

Devido a depender de estrutura adicional como servidores e hospedagem além de programação adicional desta interface também não há a gravação automática dos códigos para acesso irrestrito. Cabendo ao salvamento e abertura pela máquina. Levou-se em conta que há vários serviços para tal atividade com no caso cabendo ao usuário.

O sistema de tratamento de erros foi explorado até onde complexidade dos algoritmos possíveis dentro do proposto permitiu. Realizados assim neles os testes de erros possíveis.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em cinco capítulos que abordam o Processo de Ensino e Aprendizagem comum na atualidade e as Ferramentas Utilizadas, detalhes sobre o ensino com a utilização de Processamento de Linguagem Natural em Algoritmos.

Após o primeiro o anterior introdutório o capítulo dois descreve principais termos tem sua teoria descrita além de apresentar as principais ferramentas utilizadas no ensino e prática de lógica de programação.

Os capítulos seguintes apresentam no três a metodologia utilizada e no quatro aplicação em seu desenvolvimento até o seu resultado, apresentando alguns temos pertinentes somente ao processo.

Por fim o capítulo cinco conclui-se com as considerações finais, recomendações, sugestões e trabalhos futuros, podendo ser pelo autor ou outros interessados.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ALGORITMOS

A última das notas de Ada Lovelace, que foram republicadas em 1953, apresenta os primeiros conceitos sobre programação, descrevendo um algoritmo. Desde então estes foram difundidos, sendo utilizados até a atualidade (SANTIAGO; DAZZI, 2003).

Conforme Medina e Fertig (2006) algoritmo é um termo mais amplo, sendo inclusive destinado a outras áreas e finalidades além da programação. As encontradas na literatura são interpretativas porém todas chegam ao mesmo resultado, um conjunto de instruções para a resolução de uma situação. Algumas afirmações determinam instruções detalhadas ou mesmo que tipo de situação, se especificamente um problema ou tarefas em geral.

2.2 LINGUAGEM NATURAL

Com menor rigidez do que se faz necessário para uma máquina, esses conjuntos também podem ser descritos em uma linguagem humanamente mais natural. Numa conversa ou em uma explicação de rota de um destino ou uma receita culinária o entendimento ocorrerá. Por depender de formalidade e buscando essa aproximação é que o idioma estruturado é utilizado. Em nosso caso o português estruturado, mas há também o inglês estruturado e certamente outras (CITAR, 0000).

2.3 PROGRAMAS DE COMPUTADOR

Escritos em linguagens de programação normalmente de níveis elevados passam por camadas que são traspostas em sua maioria por um processo de compilação. Esse processo converte o código fonte escrito na variação padronizada de código para um linguagem binária (apenas zeros e uns) para comunicação direta o equipamento. Existem também linguagens de níveis mais baixos, porém de menor utilização (CITAR, 0000).

2.4 PSEUDO CÓDIGO OU LINGUAGEM

Visando ser uma ponte entre a linguagem natural e as linguagens de programação temos as pseudo-linguagem. Notadamente Portugol é evidênciada como a solução, porém não há uma podronização existindo inúmeras variações, no entando com muitas semelhanção entre si (CITAR, 0000).

3 MÉTODOS DE PESQUISA

4 DESENVOLVIMENTO DA APLICAÇÃO

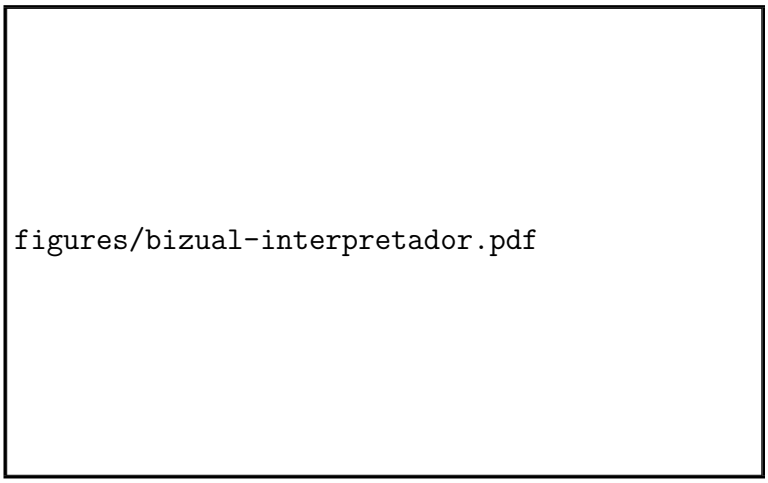
A criação do protótipo desse sistema foi o ponto principal desse trabalho, pois os objetivo foi oferecer opção para substituir *softwares* que vêm sendo utilizados atualmente por um prático, atual, acessível e com o código disponível para que possa ser corrigido e melhorado.

Nesse protótipo buscou-se eliminar algumas deficiências básica que existem em ferramentas similares, por exemplo funcionalidade de desfazer durante a edição, pois essas podem fazer com que o seu uso seja desgastante e desanimador. Como mencionado nas limitações a inclusão de funcionalidades que poderiam estar disponíveis ficarão como sugestões, algumas delas são encontradas por exemplo em Ambientes Integrados de Programação, comumente utilizada em outras linguagens.

Inclusive há caso em que a ferramenta inicialmente utilizada não é capaz de executar alguns algoritmos mais avançados. E isto leva a necessidade do uso de outro software na mesma disciplina até com possivelmente uma nova linguagem (por exemplo a linguagem de programação C ou C++), devido a está necessidade alguns educadores optam por esta sendo a única ferramenta utilizada, ignorando linguagem Portugol. Espera-se que diferente o UAL essa limitação possa realmente ser concluída em trabalho futuros, conforme proposto nas conclusões.

Essencialmente a aplicação foi desenvolvida em linguagens Web, o JavaScript sendo a linguagem de programação, HTML como linguagem de marcação e CSS como estilização. Nesse modelo já existem editores de código fonte de linguagens de programação,dos mais utilizados temos ACE e o CodeMirror, optado pelo primeiro pois em análise entendeu-se como sendo mais prático para o uso e criação do modelo da linguagem. Considerou-se a criação do modelo gramatical para destaque da sintaxe como dificuldade relativamente baixa.

Figura 1 – Esquemático do Interpretador



Fonte: Autor

Para resultar na árvore gramatical encontrou-se o ACORN um parser já escrito em JavaScript. De modo trabalhoso foi necessário adaptar o código fonte para compreender a estrutura do algorítmos proposto, principalmente devido à não utilização de parênteses para o comando de impressão em tela na linguagem UAL.

Como saída o interpretada o JS-Interpreter foi encontrado como a única opção. Ele já se utiliza do mesmo ACORN mencionado acima e interpreta a própria linguagem JavaScript. Pelo mesmo motivo citado acima, falta de parênteses no comando, se tornou uma tarefa um tanto dispendiosa fazer com que ele se adaptasse ao modelo.

Figura 2 – Interface Desktop

Fonte: Autor

Como interface gráfica foi mantido o mínimo necessário. Somente o editor de texto e um botão para executar. Foram visualizadas renderizações e tamanhos de telas variados, computadores (Figura 2), tablets, smartphones (Figura 3), etc. Seguindo conceitos do Google Material Design foi adicionada uma barra no topo como botão para menu o nome da aplicação, denominada Bizu, e um botão flutuante no canto inferior direito como ícone “play” para a execução do algoritmo.

Facilitando a tarefa de construção da interface foi utilizada a biblioteca Material Design Lite (MDL) disponibilizada oficialmente pelo Google. As cores foram escolhidas pela funcionalidade disponível no site da biblioteca.

A escolha do nome se deve a partir da palavra bizu adicionada a sigla da sintaxe.

Foi utilizado a funcionalidade *app cache* a partir de um arquivo manifesto listando o conteúdo que o navegador deve guardar cópia local. Podendo assim ter a internet desconectada após o primeiro acesso que a aplicação continuará acessível.

Todo o código foi desenvolvido em repositório disponível no github e quando concluído em sua versão mínima movido para seu destino final e podendo ser acessado.

Figura 3 – Interface Smartphone

Fonte: Autor

5 CONSIDERAÇÕES FINAIS

REFERÊNCIAS

AJAX.ORG. **Ace - The High Performance Code Editor for the Web**. 2016. <https://ace.c9.io/>. Acesso em: 24 agosto 2016.

CITAR, C. **Citar**. 0000.

DELAMARO, Márcio Eduardo. **Como Construir um Compilador Utilizando Ferramentas Java**. 1. ed. São Paulo: Novatec, 2004. v. 1. ISBN 8575220551.

DERSHEM, Herbert L; JIPPING, Michael J. **Programming languages: structures and models**. [S.l.]: Wadsworth Publ. Co., 1990.

FRASER, Neil. **JS-Interpreter - A sandboxed JavaScript interpreter in JavaScript**. 2016. <https://github.com/NeilFraser/JS-Interpreter>. Acesso em: 29 agosto 2016.

FUEGI, J.; FRANCIS, J. Lovelace babbage and the creation of the 1843 'notes'. **IEEE Annals of the History of Computing**, v. 25, n. 4, p. 16–26, Oct 2003. ISSN 1058-6180.

GOOGLE. **Material Design Lite**. 2016. <https://getmdl.io/>. Acesso em: 19 julho 2016.

HAVERBEKE, Marijn. **Acorn - A small, fast, JavaScript-based JavaScript parser**. 2016. <https://github.com/ternjs/acorn>. Acesso em: 29 agosto 2016.

HAVERBEKE, Marijn. **CodeMirror - In-browser code editor**. 2016. <https://codemirror.net/>. Acesso em: 24 agosto 2016.

JESUS, E. A. de; SANTIAGO; DAZZI, R. L. S. R. de. Ferramenta para criação e teste de algoritmos utilizando fluxogramas ou portugol. In: **Congresso Brasileiro de Computação, 2004, Itajaí**. Itajaí: Congresso Brasileiro de Computação, 2004.

LOPES, Anita; GARCIA, Guto. **Introdução à programação: 500 algoritmos resolvidos**. Rio de Janeiro: Campus, 2002. ISBN 9788535210194.

MEDEIROS, Antônio Vinícius Menezes. Um interpretador online para a linguagem portugol. 2015.

MEDINA, Marco; FERTIG, Cristina. **Algoritmos e programação**. São Paulo: Novatec, 2006.

SANTIAGO, Rafael de; DAZZI, Rudimar Luís Scaranto. Ferramentas que auxiliam o desenvolvimento da lógica de programação. In: **XII SEMINCO - Seminário de Computação, 2003, Blumenau. Anais do XII SEMINCO**. Blumenau: Furb, 2003. p. 113–120.

SEBESTA, Robert W. **Conceitos de linguagens de programação**. Porto Alegre: Bookman, 2009. ISBN 8577808629.

TAVARES, Gilberto E. **BizUAL - Portugol interpreter in HTML5**. 2016. <https://camaleaun.github.io/bizual/>. Acesso em: 4 outubro 2016.