UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I - CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM COMPUTAÇÃO

MAYARA VICTÓRIA MEDEIROS PEREIRA

TURINGMS: UM SIMULADOR PARA APOIAR NO PROCESSO DE ENSINO-APRENDIZAGEM DE LINGUAGENS FORMAIS E TEORIA DA COMPUTAÇÃO

CAMPINA GRANDE
2021

MAYARA VICTÓRIA MEDEIROS PEREIRA

TURINGMS: UM SIMULADOR PARA APOIAR NO PROCESSO DE ENSINO-APRENDIZAGEM DE LINGUAGENS FORMAIS E TEORIA DA COMPUTAÇÃO

Trabalho de Conclusão de Curso (Artigo) apresentado ao Departamento do Curso de Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Computação.

Orientador: Prof. Me. Edson Holanda Cavalcante Júnior.

CAMPINA GRANDE
2021

P436t Pereira, Mayara Victória Medeiros.
Turingms [manuscrito] : um simulador para apoiar no processo de ensino-aprendizagem de linguagens formais e teoria da computação / Mayara Victoria Medeiros Pereira. - 2021.
30 p. : il. colorido.

Digitado.
Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia , 2021.
"Orientação : Prof. Me. Edson Holanda Cavalcante Júnior , Coordenação do Curso de Computação - CCT."

1. Ensino da Computação. 2. Linguagens formais. 3. Teoria da Computação. 4. Simulador. I. Título

21. ed. CDD 004

Elaborada por Giulianne M. Pereira - CRB - 15/714 BC/UEPB

MAYARA VICTÓRIA MEDEIROS PEREIRA

TURINGMS: UM SIMULADOR PARA APOIAR NO PROCESSO DE ENSINO-APRENDIZAGEM DE LINGUAGENS FORMAIS E TEORIA DA COMPUTAÇÃO

Trabalho de Conclusão de Curso (Artigo) apresentado ao Departamento do Curso de Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Computação.

Aprovada em: 09 / 06 / 2021.

BANCA EXAMINADORA

Prof. Me. Edson Holanda Cavalcante Júnior (Orientador)
Universidade Estadual da Paraíba (UEPB)

Prof. Dr. Wellington Candeia de Araújo
Universidade Estadual da Paraíba (UEPB)

Prof. Me. Antônio Carlos de Albuquerque
Universidade Estadual da Paraíba (UEPB)

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

GAM	Ginux Abstract Machine
IDE	Integrated Development Environment / Ambiente de Desenvolvimento Integrado
JFLAP	Java Formal Languages and Automata Package
LabLF	Laboratório de Linguagens Formais
MT	Máquina de Turing
OA	Objeto de Aprendizagem
SCTMF	Sistema de Criação e Teste de Modelos Formais
TuringMS	Turing Machine Simulator
UFPA	Universidade Federal do Pará
UI	User Interface / Interface do Usuário
VAS	Visual Automata Simulator

LISTA DE SÍMBOLOS

Σ	Alfabeto de símbolos de entrada
β	Branco
_	Branco - TuringMS
π	Função de transição ou programa
✪	Marcador de início de fita
▷	Marcador de início de fita - TuringMS
>	Movimento à direita - TuringMS
<	Movimento à esquerda - TuringMS

SUMÁRIO

TURINGMS: UM SIMULADOR PARA APOIAR NO PROCESSO DE ENSINO-APRENDIZAGEM DE LINGUAGENS FORMAIS E TEORIA DA COMPUTAÇÃO

TURINGMS: A SIMULATOR TO SUPPORT THE TEACHING-LEARNING PROCESS OF FORMAL LANGUAGES AND COMPUTER THEORY

Mayara Pereira[1]

RESUMO

Este estudo objetivou o desenvolvimento de um simulador capaz de facilitar a apresentação e assimilação de conteúdos relacionados à máquina de Turing nos cursos superiores de computação. Alguns dos passos galgados para a implementação desse simulador foram conhecer os conceitos ligados à máquina de Turing, identificar as principais dificuldades dos alunos e professores em disciplinas de Linguagens Formais e Teoria da Computação e como a utilização de um simulador gráfico pode auxiliar nesse processo de ensino-aprendizagem, bem como identificar a necessidade de diversificação de acesso ao simulador. Para tanto, a pesquisa bibliográfica foi utilizada como método para coleta de dados e formalização de conceitos. Já para o desenvolvimento do simulador foram realizadas consultas à documentação da linguagem e framework utilizados. A partir da coleta de dados foi possível perceber a clara dificuldade no processo de ensino-aprendizagem de conteúdos majoritariamente teóricos, onde a ausência de uma representação visual e a participação mais direta dos alunos na construção do saber destacaram-se como sendo pontos possíveis de ajustes através dos simuladores gráficos voltados ao ensino. Nesta visão, a facilidade na utilização e acesso ao simulador também contam como fatores decisivos para o sucesso ou fracasso da iniciativa. Enfim, por meio de todo levantamento e estudo realizado foi possível confirmar que a utilização de simuladores gráficos para a apresentação de conceitos relacionados à Linguagens Formais e Teoria da Computação pode auxiliar e tornar mais interativo e dinâmico o processo de ensino-aprendizagem.

Palavras-chave: Ensino da Computação. Linguagens Formais. Teoria da Computação. Simulador.

ABSTRACT

This study aimed to develop a simulator capable of facilitating the presentation and assimilation of contents related to the Turing machine in high education courses in computing. Some of the steps taken to implement this simulator were: knowing the concepts concerning the Turing machine; identifying the main difficulties of students and teachers in disciplines of Formal Languages and Computer Theory; and how the use of a graphic simulator can assist in the teaching-learning process, as well as identifying the need for diversification of access to the simulator. Therefore, we used bibliographic research as a method for data collection and formalization of concepts. For the development of the simulator, we consulted the language documentation and framework used. From data collection, it was possible to realize the main difficulty in the teaching-learning process of mostly theoretical contents, where the absence of a visual representation and the more direct participation of students in the construction of knowledge stood out as possible points of adjustments through graphic simulators focused on

[1] Graduanda em Ciência da Computação na Universidade Estadual da Paraíba (UEPB).
mayara.victoria@aluno.uepb.edu.br

teaching. In this point of view, the ease of usage and access to the simulator also count as decisive factors for the success or failure of the initiative. Finally, all the surveys and studies carried out made it possible to confirm that using graphic simulators to present the concepts related to Formal Languages and Computer Theory can support and make the teaching-learning process more interactive and dynamic.

Keywords: Computer Teaching. Formal Languages. Computer Theory. Simulator.

1. INTRODUÇÃO

Os temas abordados nas disciplinas relacionadas a Linguagens Formais e Teoria da Computação nos cursos superiores da área de Tecnologia são de extrema importância para uma formação abrangente do profissional. Esses temas compõem as Diretrizes Curriculares Nacionais para os cursos de graduação na área da Computação, tornando-se fundamental para qualquer currículo de graduação e pós-graduação neste âmbito.

De forma geral, essas disciplinas por si só possuem desafios atrelados ao processo de ensino-aprendizagem nada triviais de serem superados. Isso se dá devido à alta quantidade de abstrações e teorias nos conteúdos abordados, dificultando a visualização e fixação dos assuntos por parte do aluno e a demonstração por parte do professor. Com isso, a inclusão de técnicas de visualização e práticas é sugerida como forma de o aluno vivenciar os conceitos teóricos apresentados. Vê-se, nesse ponto, os simuladores gráficos como um OA (objeto de aprendizagem) - qualquer recurso digital destinado a apoiar o aluno no processo de aprendizagem[2] - que pode auxiliar no processo ensino-aprendizagem, possibilitando ao professor demonstrar de maneira mais simples o que antes seria apenas conceitos e ao aluno visualizar e praticar aumentando sua autonomia como agente receptor de conhecimentos. Essa pesquisa foca em explanar o desenvolvimento do TuringMS, assim como os conceitos e objetivos envolvidos em seu planejamento.

1.1 Contexto e Motivação

Diante de um cenário que desperta atenção pelas grandes dificuldades no processo de ensino-aprendizagem nas disciplinas de Linguagens Formais e Teoria da Computação devido à complexidade dos assuntos e altos níveis de abstração existentes, a exposição desses conteúdos sem associação à prática tende a ser enfadonha e sem grande produtividade. Um dos meios de otimização das aulas é a inclusão da prática juntamente com a explanação de conteúdos, estimulando o aluno a vivenciar o que está aprendendo (FURTADO, 2003). Para tanto, existem diversas ferramentas que possibilitam a prática desses assuntos. Dentre elas, os simuladores são uma alternativa eficaz para proporcionar ao aluno, através do seu manuseio, a ampliação e consolidação dos seus conhecimentos e habilidades na resolução de problemas.

Existem diversas ferramentas que simulam os diversos tipos de autômatos, processamentos, gramáticas, Máquina de Turing etc., cada qual com sua tecnologia e finalidade. Algumas das ferramentas são instaláveis, gratuitas e possuem o escopo bem restrito, já outras dão ao usuário uma boa autonomia do que irá simular. Com essas possibilidades, cabe principalmente ao professor, avaliar aquela que melhor atende a sua necessidade de ensino.

A partir da comparação de ferramentas realizada por Aguiar e Oeiras (2010), verificou-se que muitas possuem certas limitações em relação aos aparelhos compatíveis para sua utilização. Sendo assim, a acessibilidade também é um fator preocupante nesse cenário, pois além de impossibilitar muitos alunos de praticarem de forma autônoma a partir dos próprios dispositivos, nem sempre as instituições de ensino possuem laboratórios disponíveis para as aulas de Linguagens Formais e Teoria da Computação.

Nesse contexto, a motivação principal para o desenvolvimento do TuringMS é a escassez de simuladores gráficos que não tenham a necessidade de instalação ou dispositivos específicos para sua utilização, bem como uma ferramenta com livre acesso da comunidade.

---

[2] OBJETOS DE APRENDIZAGEM. NUTED, 2021. Disponível em: http://www.nuted.ufrgs.br/wordpress_beta/?page_id=2496. Acesso em: 22 de maio de 2021.

1.2 Objetivos

A partir do cenário já explicitado, a importância da inclusão dos softwares simuladores no ensino de conteúdos teóricos da Computação foi evidenciada. Para tanto, buscou-se reunir dados e informações com o propósito de responder ao seguinte problema de pesquisa: Como otimizar o processo ensino-aprendizagem de Linguagens Formais e Teoria da Computação através de um simulador gráfico?

Para tal pergunta, a resposta segue para a construção de um simulador gráfico - no escopo da Máquina de Turing - que:

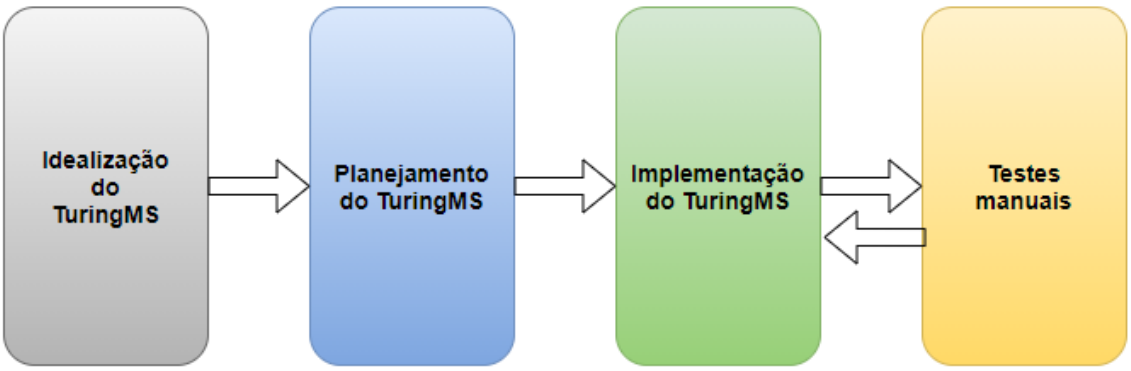
- Obedeça aos formalismos associados a Máquina de Turing;
- Possibilite uma visualização clara do processamento de palavras;
- Possibilite ao professor demonstrar de maneira prática os conceitos e abstrações envolvidas nos conteúdos relacionados à Máquina de Turing;
- Seja acessível em diversos dispositivos sem grandes prejuízos para visualização do processamento;
- Para sua execução, utilize recursos comuns do acesso à rede.

Dessa forma, o propósito geral deste trabalho é disponibilizar a alunos e professores de cursos de tecnologia um simulador gráfico acessível, de qualidade, totalmente gratuito e aberto à colaboração da comunidade visando agregar valor às aulas de Linguagens Formais e Teoria da Computação.

1.3 Metodologia

Na Figura 1 é demonstrada, através de um diagrama, a metodologia utilizada como base para a produção deste artigo:

Figura 1 - Fases da metodologia utilizada



Fonte: Elaborada pela autora, 2021.

- **Idealização do TuringMS**: analisando o cotidiano da ministração de aulas de Linguagens Formais e Teoria da Computação, a evidente dificuldade em produzir demonstrações visuais satisfatórias (muitas vezes através de desenhos manuais na lousa) para o ensino de conteúdos relacionados à Máquina de Turing e as limitações de acesso às ferramentas gráficas já conhecidas, demonstraram a necessidade de construir um simulador capaz de possibilitar acesso facilitado a partir de diversos dispositivos e, principalmente,

produzir uma simulação gráfica fácil e de qualidade respeitando todos os formalismos associados aos conceitos.

- **Planejamento do TuringMS**: para um melhor entendimento dos conceitos e formalismos envolvidos, pesquisas bibliográficas foram realizadas a fim de adquirir uma melhor percepção do nível de complexidade da ferramenta que seria implementada. Assim, nessa fase ocorreu a definição do escopo de processamento do simulador (máquina reconhecedora de linguagens), escolha de tecnologias a serem utilizadas, bem como estudos e definições sobre a usabilidade.
- **Implementação do TuringMS**: essa fase corresponde a todo desenvolvimento funcional do simulador, utilizando linguagens de programação, frameworks, IDE, etc. À medida que uma nova funcionalidade era implementada, testes manuais eram realizados. Melhorias na usabilidade e correções funcionais também foram implementadas nessa fase, sem se desviar do planejamento inicial.
- **Testes manuais**: essa fase ocorreu em paralelo à fase anterior, visto que testes eram realizados a cada nova funcionalidade implementada. Basicamente, esses testes foram realizados simulando necessidades de alunos e professores em sala de aula, bem como suas possíveis dúvidas no que tange à sequência de passos para simulação.

1.4 Organização do trabalho

O presente artigo foi dividido nas seguintes seções:

Na Seção 2 é apresentado o referencial teórico utilizado para contextualização dos conceitos de Máquina de Turing, introduzindo uma visão geral do conceito dessa máquina universal, destrinchando a abordagem intuitiva e como máquina. É abordado também os formalismos envolvidos e essenciais para o entendimento de cada componente necessário para o processamento, finalizando com uma breve explicação sobre o escopo de reconhecimento utilizado para o desenvolvimento do simulador.

Na seção 3 também é apresentado o referencial teórico utilizado para a contextualização da inclusão de simuladores no processo de ensino-aprendizagem. Através de dados e citações de autores, a importância dessas ferramentas no ensino é explicitada.

Na seção 4, o simulador desenvolvido (TuringMS) é apresentado. Nessa seção é descrito todo o planejamento da ferramenta, arquitetura, tecnologias utilizadas, ferramentas com propostas similares, funcionalidades, elementos visuais e limitações.

A seção 5 finaliza o artigo com as conclusões finais e algumas sugestões para trabalhos futuros.

2. CONCEITOS DE MÁQUINA DE TURING

Na presente seção, são apresentados os principais conceitos e definições aplicados no ensino do conteúdo da disciplina de Linguagens Formais e Teoria da Computação que estão relacionados ao simulador desenvolvido. No que se refere a máquinas universais, segundo Diverio e Menezes (1999), devem ser definidas de forma que sejam suficientemente simples para que suas propriedades relevantes sejam analisadas e conclusões gerais sejam estabelecidas, e suficientemente poderosa com a capacidade de simular qualquer máquina real ou teórica. De acordo com as Diretrizes Curriculares Nacionais para os cursos de graduação na área da Computação, é de caráter obrigatório que os fundamentos teóricos da área de Computação sejam explanados no curso, assegurando a formação de profissionais dotados desses conhecimentos.

Na subseção 2.1 e seus subtópicos é exposta uma visão geral da noção intuitiva e noção como máquina do modelo Máquina de Turing (MT). A subseção 2.2 trata dos conteúdos que

estão relacionados ao modelo formal da máquina: representações, definições, estados e transições. Na subseção 2.3 é abordado a MT como um reconhecedor de linguagens, discorrendo sobre seu escopo de processamento.

Tendo em vista que existem diversas variações de definições de MT, todo o escopo desse trabalho é baseado na definição exposta pelos autores Tiarajú Asmuz Diverio e Paulo Blauth Menezes em seus livros[3] publicados em 1999 e 2011 voltados à Teoria da Computação.

2.1 Visão Geral

Primeiramente, para entender os conceitos da MT, é importante que a definição de algoritmo esteja clara. Segundo Donato (2019), um algoritmo é uma sequência de instruções bem definidas que devem ser executadas até que uma condição seja satisfeita. Ainda, segundo o autor, um algoritmo é o conjunto de passos necessários para concluir uma tarefa e não, necessariamente, um programa de computador. Esse conceito foi formalizado em 1936 pelo cálculo lambda de Alonzo Church e através da MT.

As operações definidas em um algoritmo podem ser simuladas por uma MT, a qual "[...] possui, no mínimo, o mesmo poder computacional do que qualquer computador de propósito geral. " (DIVERIO E MENEZES, 1999, p. 83). Para que o processamento ocorra de forma satisfatória, é necessário que o conjunto de instruções seja bem definido, especificando o comportamento em todas as possíveis circunstâncias.

2.1.1 Noção Intuitiva

A noção intuitiva de um algoritmo é inspirada em ações que uma pessoa precisa executar para realizar cálculos. Assim, conforme Menezes (2013), essa noção intuitiva não é precisa em um contexto matemático. A MT é justamente uma proposta de formalização dessas ações e, devido à não exatidão atrelada, se torna impossível provar que a MT é o dispositivo mais genérico da computação. O que se pode afirmar é que existem evidências de que a capacidade computacional da MT é a máxima a ser atingida por outros modelos.

Ao imaginar o ato de resolução de cálculos, Alan Turing considerou que o agente ativo teria disponível um instrumento de escrita, uma fita ou folha de papel (quadriculados) contendo os dados iniciais do problema e um apagador (ALVES *et al.*, 2019). Diverio e Menezes (2011) descrevem que as ações desse agente se resumem a ler e alterar um símbolo do quadrado e mover os olhos para focar em outro quadrado adjacente. Considerando que o papel ou fita tem uma quantidade (potencialmente) infinita de quadrados, que os dados iniciais do problema são finitos (símbolos, estados), que há uma definição de início e fim e que o agente ativo manipula apenas dados de um quadrado por vez, se torna possível alcançar uma representação satisfatória para o problema. Alcançado este ponto, os cálculos são encerrados.

2.1.2 Noção como Máquina

A noção intuitiva apresentada na subseção 2.1.1, pode ser vista como uma máquina, onde, segundo Diverio e Menezes (1999), a fita se torna um dispositivo de entrada, saída e memória, o agente passivo é uma unidade de controle que lê e grava conteúdos na célula processada, movimentando-se para esquerda ou direita obedecendo as configurações da função de transição (ou programa) que definem movimentos, estados e símbolos a serem lidos e gravados pela unidade de controle.

[3] Teoria da computação: máquinas universais e computabilidade.

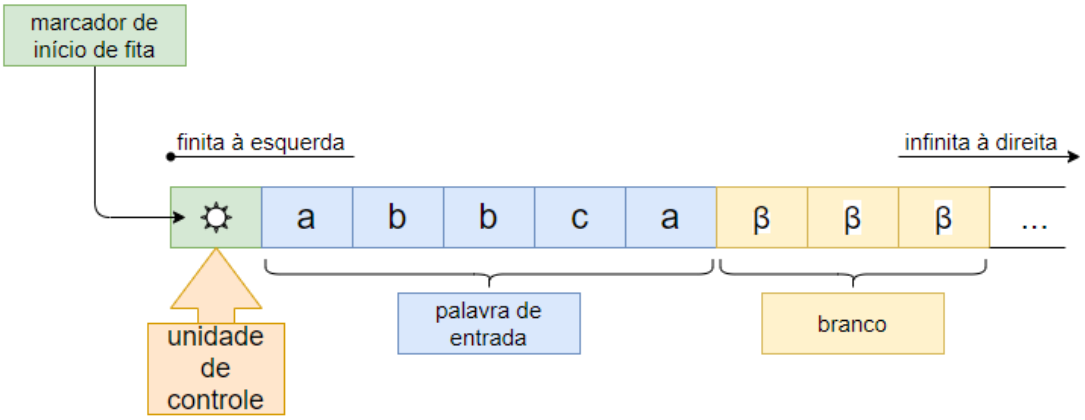
As restrições presentes no ato de calcular (descrito na subseção 2.1.1) permanecem na ideia de função de uma máquina. Sendo assim, em cada célula da fita apenas um símbolo pode ser armazenado, o qual deve estar previamente definido antes do início do processamento, bem como a finitude da fita à esquerda e a infinitude à direita.

Nesse contexto, os símbolos essenciais para o processamento são: símbolo marcador de início de fita; alfabeto de entrada ou auxiliar; símbolo branco.

2.2 Modelo Formal

Antes de iniciar a apresentação da formalização associada à MT, alguns elementos essenciais para o processamento estão evidenciados na Figura 2. Conforme descrito na subseção 2.1.2, a fita possui uma quantidade infinita de células à direita (quantas forem necessárias para o processamento da entrada), porém finita à esquerda, estando em sua primeira célula o símbolo marcador de início de fita, seguida de células povoadas pela palavra de entrada e o restante como o símbolo branco (β). A unidade de controle armazena o estado atual da máquina após cada movimentação definida na função de transição (programa).

Figura 2 - Representação da Fita da MT



Fonte: Teoria da Computação: Máquinas Universais e Computabilidade (1999, p. 84, com adaptações).

Segundo Diverio e Menezes (2011, p. 134), "uma MT é uma 8-upla [...]", ou seja, é uma sequência ordenada de oito elementos conforme a representação abaixo:

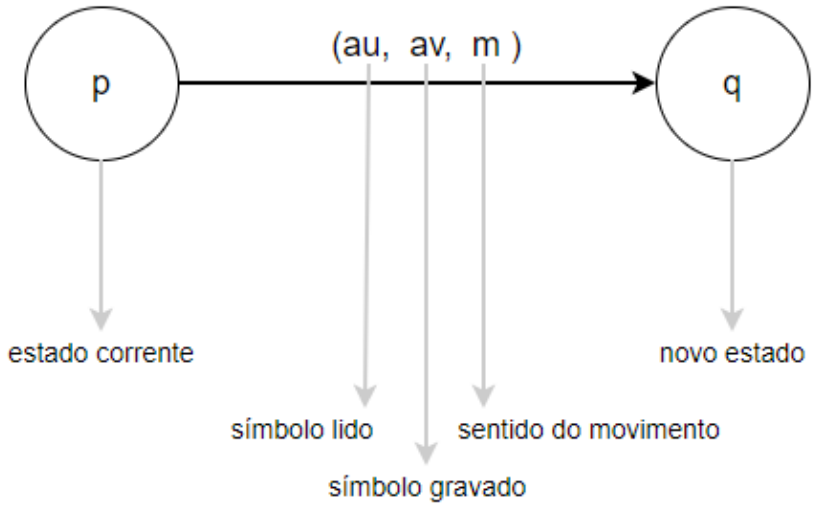
$$M = (\Sigma, Q, \pi, q0, F, V, \beta, ⌾)$$

O alfabeto de símbolos de entrada - representado por Σ - é um conjunto finito de símbolos que são reconhecidos durante o processamento. O Q representa o conjunto finito de estados que a máquina poderá assumir durante o processamento da entrada. A função de transição ou programa é representada por π e deve conter instruções de leitura, gravação e movimentação a serem seguidas durante o processamento. O estado inicial (q0) também deve ser previamente definido, pois é a partir dele que o processo é iniciado, bem como o conjunto de estados finais (F). Outros elementos como o alfabeto auxiliar (V), o símbolo branco (β) e o símbolo marcador de início de fita (⌾) na célula mais à esquerda da fita finalizam a definição da 8-upla.

As instruções de leitura, gravação e movimentação fornecidas pela função de transição são compostas pela condição atual (estado e símbolo lido) levando a uma nova condição

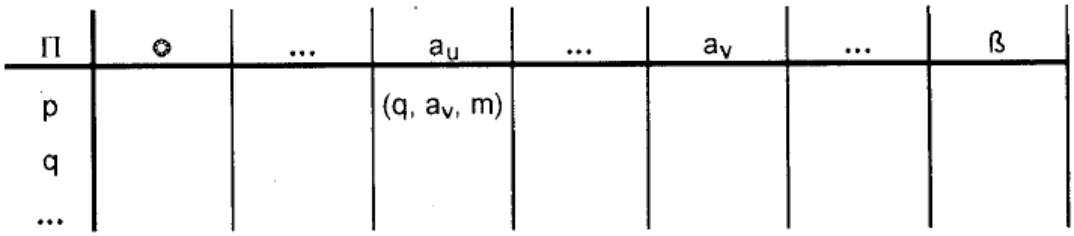
indicando o novo estado (podendo ser o estado corrente), um símbolo a ser gravado na célula lida da fita e a direção do movimento para mover a cabeça da fita. Essa função pode ser representada por grafo, conforme ilustrado na Figura 3, e como tabela conforme a Figura 4.

Figura 3 - Representação da Função Programa Como Um Grafo



Fonte: Teoria da Computação: Máquinas Universais e Computabilidade (1999, p. 86).

Figura 4 - Representação da Função Programa Como Tabela



Fonte: Teoria da Computação: Máquinas Universais E Computabilidade (1999, p. 86).

Na representação da função programa como tabela, todos os símbolos do alfabeto definido, o branco, o marcador de início de fita e o alfabeto especial povoam a primeira linha da tabela. A primeira coluna à esquerda é povoada pelos estados definidos. A célula de intersecção entre estado corrente e símbolo lido deverá conter as informações do próximo estado, símbolo gravado e movimento para seguir o processamento.

Conforme Diverio e Menezes (2011), o processamento consiste em repetir a análise das configurações de transição a cada novo cenário desde o estado inicial até chegar em um resultado de aceitação (atingiu o estado final) ou rejeição (entrada não correspondente ao alfabeto definido ou movimento inválido) da entrada processada. Também existe a possibilidade de ocorrer loop infinito e um resultado concreto não ser alcançado.

2.3 Reconhecedores de Linguagens

Os reconhecedores de linguagens, segundo Diverio e Menezes (1999), são dispositivos que identificam, através do alfabeto de entrada, se uma palavra pertence ou não a uma determinada linguagem. No contexto de processamento da MT, considera-se que uma palavra pertence a uma linguagem se em algum momento um estado final de aceitação é atingido,

garantindo que a palavra de entrada pertença ao alfabeto de símbolos definidos e que o estado final pertença ao conjunto de estados de aceitação da máquina. Já quando uma palavra não pertence a uma linguagem, o processamento se encerra sem atingir um estado final pertencente ao conjunto de estados de aceitação da máquina ou entra em loop infinito sem um resultado concreto.

As linguagens que permitem processamento de palavras pertencentes ao seu conjunto são linguagens enumeráveis recursivamente, sendo assim, "[...] a Classe das Linguagens Enumeráveis Recursivamente define a classe de todas as linguagens que podem ser reconhecidas mecanicamente". (DIVERIO E MENEZES, 1999, p. 91). Nessa classe mais genérica, também existem palavras que podem entrar em loop sem um resultado concreto atingido, para tanto a subclasse de Linguagens Recursivas foi definida para agrupar apenas as linguagens que possuem ao menos uma MT que resulta em aceitação ou rejeição, sem entrar em loop.

3. UTILIZAÇÃO DE SIMULADORES COMO FERRAMENTA DE APOIO NO PROCESSO DE ENSINO-APRENDIZAGEM DE DISCIPLINAS TEÓRICAS

Um simulador, como o próprio nome evidencia, simula algum estado, situação, comportamento ou características de dispositivos e softwares. Com a utilização de simuladores no ensino, espera-se que "[...] o aluno, por meio do manuseio dessas ferramentas computacionais, possa ampliar e consolidar seus conhecimentos e habilidades na resolução de problemas. " (AGUIAR E OEIRAS, 2010, p. 107). No contexto educacional, os simuladores são recursos poderosos para a otimização do ensino e aprendizado, pois, além de facilitar a transmissão de informações teóricas através de demonstrações, possibilita ao aluno ter um papel mais ativo nesse processo.

Como bem destaca Furtado (2003), o ensino de conteúdos teóricos sem a prática se torna improdutivo e sem empolgação, visto que os alunos permanecem em um papel totalmente passivo, principalmente quando a complexidade dos assuntos é elevada. Vale destacar também que os conteúdos abordados nas disciplinas de Linguagens Formais e Teoria da Computação tendem a ser apresentados de maneira puramente algébrica, destacando a exigência de uma boa formação matemática prévia dos alunos, comprovando ainda mais a dificuldade de aprendizado a partir de abstrações (RAMOS, 2009).

Conforme explicado acima, em todo esse processo, pode-se dizer que tanto docentes como discentes devem possuir papéis ativos na construção do aprendizado e, quando um desses papéis se desequilibra, o processo tende a ser prejudicado. É interessante, aliás, destacar que a utilização de simuladores pode auxiliar a manter este equilíbrio de papéis, promovendo ações de ambas as partes.

De acordo com Aguiar e Oeiras (2010), através de um estudo de caso, foi concluído que cerca de 30% dos alunos da disciplina de Linguagens Formais entre o segundo semestre de 2005 e o segundo semestre de 2009 na Universidade Federal do Pará (UFPA) não obtiveram um desempenho satisfatório para atingir a aprovação. Seguem afirmando que um dos possíveis fatores para esse baixo desempenho é a incerteza das respostas dos exercícios ou até a não resolução destes.

Pode-se dizer que a introdução de simuladores no processo de ensino-aprendizagem tem um caráter facilitador e otimizador, tanto para o desempenho dos professores como dos alunos (RAMOS, 2009). Nesse contexto, fica claro que a principal motivação da construção de simuladores para auxiliar no ensino-aprendizagem de Linguagens Formais e Teoria da Computação é possibilitar ao professor uma maneira mais clara e objetiva de demonstração de conteúdos altamente abstratos e proporcionar ao aluno uma visualização mais real dos

conceitos, bem como "poder reduzir as suas dificuldades ao tentarem resolver exercícios com diferentes graus de complexidade." (AGUIAR E OEIRAS, 2010, p. 106).

Em tese, no âmbito educacional, os simuladores devem ser utilizados de forma agregada a outros elementos essenciais para o ensino, como tarefas propostas, embasamento teórico das demonstrações, discussões etc. Caso contrário, corre-se o risco de que a absorção de conteúdos não ocorra de fato, mas, sim, apenas uma replicação de ações de forma superficial, como se seguisse um passo a passo sem questionamentos.

> Apesar de todas as vantagens apresentadas, o emprego de simuladores, sejam eles gráficos[4] ou não, fechados[5] ou abertos[6], deve sempre ser feito de modo criterioso, a fim de não preterir o pensamento algébrico do aluno. Se usadas de forma ampla e sem planejamento, tais ferramentas podem induzir uma espécie de preguiça mental que prejudica o alcance dos objetivos da disciplina. (RAMOS, 2010, p. 31).

O autor op. cit. deixa claro que a utilização dos simuladores no ensino não deve ser feita de qualquer maneira. Esse é o motivo pelo qual é importante frisar esse ponto, uma vez que a introdução de ferramentas computacionais ao processo de ensino-aprendizagem sem planejamento tende a comprometer o aprendizado e, possivelmente, a formação do aluno como um todo. Desse modo, a motivação e planejamento para a utilização desses recursos devem estar bem definidos, fazendo com que professores e alunos não caiam nas armadilhas da preguiça mental e acabem prejudicando o processo em geral.

Fica evidente, diante dos dados e informações apresentadas, que os simuladores são, de fato, ferramentas importantes e agregadores de qualidade no processo de ensino-aprendizagem, desde que utilizados de maneira responsável com planejamento e estudo, o resultado tende a ser extremamente satisfatório. Atentando-se a isso, a importância do papel do professor na condução da utilização desses recursos é ressaltada, bem como a autonomia do aluno como o agente a receber instrução.

4. SIMULADOR PARA APOIAR NO PROCESSO DE ENSINO-APRENDIZAGEM DE LINGUAGENS FORMAIS E TEORIA DA COMPUTAÇÃO

Essa seção apresenta o simulador desenvolvido neste trabalho, o TuringMS (Turing Machine Simulator)[7]. Ele foi desenvolvido levando em consideração os princípios teóricos da MT expostos pelos autores Tiarajú Asmuz Diverio e Paulo Blauth Menezes (1999 e 2011), e os principais propósitos da inclusão de simuladores no ensino de Linguagens Formais e Teoria da Computação, proporcionando aos alunos e professores uma ferramenta para construir uma experiência empolgante na relação ensino-aprendizagem.

Através desse simulador gráfico, o aluno pode se envolver com o conteúdo por meio da visualização e interação, manipulando de forma simplificada o processamento de palavras em uma MT. Para isso, basta que o aluno acesse o TuringMS por meio de um navegador e siga passo a passo as orientações de definição da máquina para iniciar as simulações de processamento da entrada.

A subseção 4.1 demonstra as principais características e a arquitetura de componentes do simulador. A subseção 4.2 mostra como é realizada a definição da máquina no simulador e o processamento. A subseção 4.3 apresenta as principais tecnologias utilizadas na construção do simulador. Por fim, na subseção 4.4 as considerações finais sobre o desenvolvimento.

---

[4] Simuladores que possibilitam que o usuário interaja com elementos computacionais visuais.
[5] Simuladores nos quais o usuário tem acesso apenas a simulações pré definidas pelo desenvolvedor.
[6] Simuladores nos quais o usuário tem a possibilidade de modificar os cenários de simulação.
[7] Disponível em: https://turing-machine-simulator.herokuapp.com/

4.1 Características Principais e Arquitetura do Simulador

Um dos obstáculos enfrentados na adoção de simuladores na educação é a limitação de equipamentos para executar as simulações de forma que os alunos não sejam meros espectadores durante as aulas. Contudo, os avanços tecnológicos têm contribuído para que esses simuladores se tornem cada vez mais acessíveis e adaptáveis a dispositivos móveis, possibilitando que, mesmo fora de um laboratório apropriado, alunos e professores tenham acesso às simulações onde e quando necessitarem.

Para tal, existem diversos simuladores voltados ao ensino de Linguagens Formais e Teoria da Computação, como o GAM (Ginux Abstract Machine)[8], VAS (Visual Automata Simulator)[9], Language Emulator[10], SCTMF (Sistema de Criação e Teste de Modelos Formais)[11], JFLAP (Java Formal Languages and Automata Package)[12], entre outros, sendo todos esses desenvolvidos em Java e caracterizados como ferramentas desktop em que o aluno/professor pode baixá-las e executá-las em seu computador. Também é conhecido o LabLF (Laboratório de Linguagens Formais)[13], o qual é desenvolvido em PHP utilizando funcionalidades da ferramenta JFLAP que possui código aberto.

Utilizando o TuringMS, os alunos poderão realizar uma imersão na fita da MT que eles próprios definiram, possibilitando visualizar passo a passo, manipular, e explorar o processamento da entrada recebida. Essa atividade de simulação gráfica tem por objetivo fortalecer e consolidar a compreensão dos alunos em relação aos conceitos abstratos ligados às máquinas universais. Para tanto, uma estratégia pedagógica bem delineada deve ser inserida com o intuito de assegurar que os alunos compreendam o conteúdo.

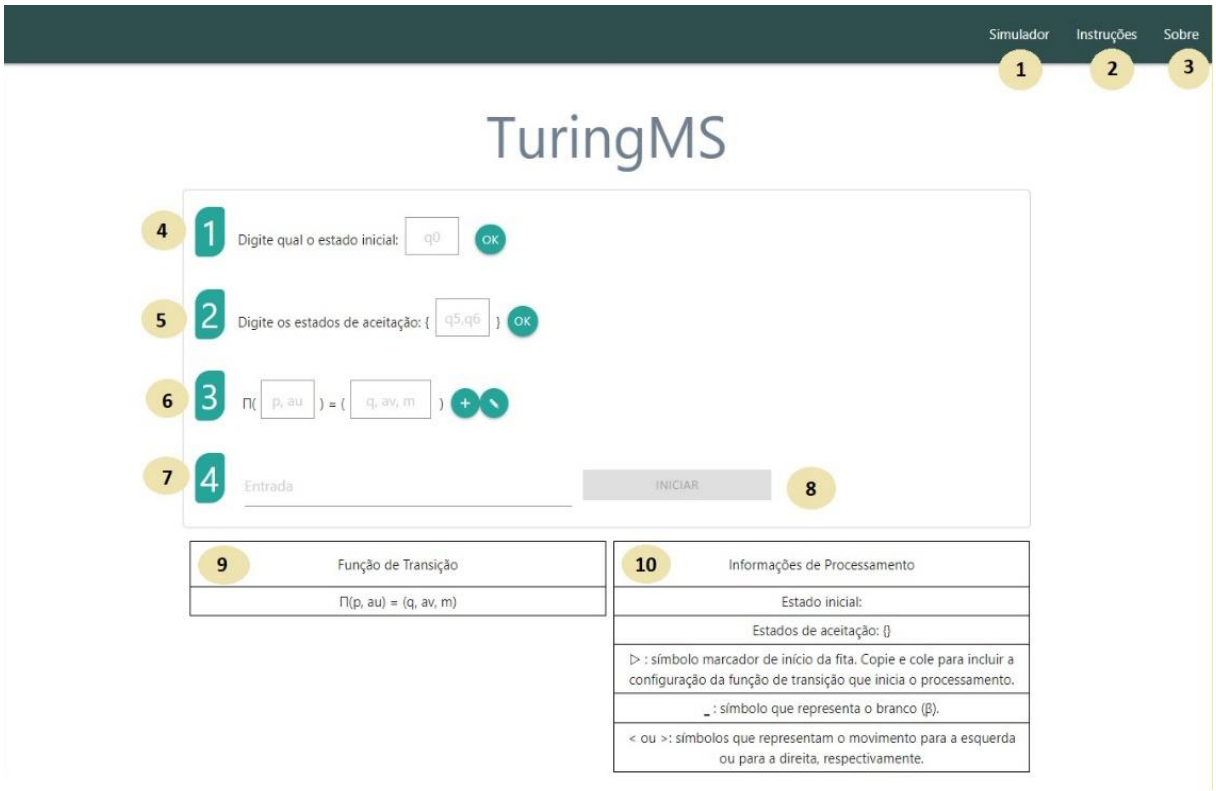
As principais atividades de simulação que podem ser executadas e observadas no TuringMS são:

- **Definição de uma Máquina de Turing:** nos primeiros passos antes de iniciar o processamento, o usuário deve definir o estado inicial, estados finais e função de transição/programa de acordo com sua necessidade;
- **Alteração da demonstração de processamento:** através de botões no *footer* da página de simulação, o usuário poderá progredir o processamento passo a passo ou realizá-lo de uma só vez, exibindo apenas o resultado final. Da mesma maneira, o usuário pode regredir os passos um a um ou voltar ao estado inicial da máquina;
- **Acompanhar as mudanças de estado:** a cada passo executado do processamento da entrada, é exibido ao usuário o estado atual da máquina na unidade de controle da fita;
- **Entender as transições dos estados:** a cada passo do processamento da entrada, a função de transição/programa executada para a configuração atual ser estabelecida é exibida, permitindo que cada movimentação do apontador, leitura e escrita fique clara ao usuário;
- **Alteração em tempo real:** é possibilitado ao usuário a modificação da definição da máquina e da entrada a qualquer momento, porém o processamento será reiniciado;
- ***Feedback* de processamento em *loop*:** quando a máquina atinge 100 repetições de um fluxo, é considerado *loop*. O processamento será encerrado e um feedback do estado de *loop* será exibido ao usuário.

[8] Mais detalhes em: http://repositorio.ufla.br/jspui/handle/1/9544
[9] Disponível em: https://github.com/yurifw/Visual-Automata-Simulator
[10] Disponível em: https://homepages.dcc.ufmg.br/~lfvieira/ftc.html
[11] Mais detalhes em: http://www2.sbc.org.br/csbc2008/pdf/arq0119.pdf
[12] Disponível em: http://www.jflap.org/
[13] Mais detalhes em: https://www.researchgate.net/publication/277160851_Laboratorio_de_Linguagens_Formais

A Figura 5 mostra a interface inicial de utilização do TuringMS, antes do usuário configurar a máquina e iniciar o processo de simulação.

Figura 5 - Página inicial do TuringMS



Fonte: Turing Machine Simulator (2021)[14]

É possível observar os principais elementos que fazem parte da página inicial do simulador, que são:

- **Elemento 1:** é a opção ‘Simulador’ que, ao ser selecionada, retorna para a página inicial no caso de estar em outra aba;
- **Elemento 2:** é o componente do menu que, ao ser selecionado, exibe a página de instruções passo a passo do que deve ser feito para iniciar o processamento;
- **Elemento 3:** é a opção ‘Sobre’ que, ao ser selecionada, exibe a página de informações sobre o desenvolvimento, autoria, finalidade, bem como o repositório com o código fonte da aplicação;
- **Elemento 4:** é onde o usuário define o estado inicial da máquina. É indicado iniciar a construção da MT por este elemento. Ao clicar no número 1, um modal com informações sobre o campo é exibido. Ao clicar no botão ‘OK’, o estado inicial é definido;
- **Elemento 5:** é onde o usuário deve definir qual(is) o(s) estado(s) de aceitação da máquina. Ao clicar no número 2, um modal com informações sobre o campo em questão é exibido. Ao clicar no botão ‘OK’, o(s) estado(s) final(is) da MT do usuário é/são definido(s);
- **Elemento 6:** é o local de definir os símbolos a serem lidos, gravados e movimentação através das configurações da função de transição. A título de processamento, os movimentos para direita são representados por ‘>’, para a esquerda por ‘<’, o símbolo marcador de início de fita como ‘▷’ e o branco como ‘_’. Ao clicar no número 3, um modal com informações

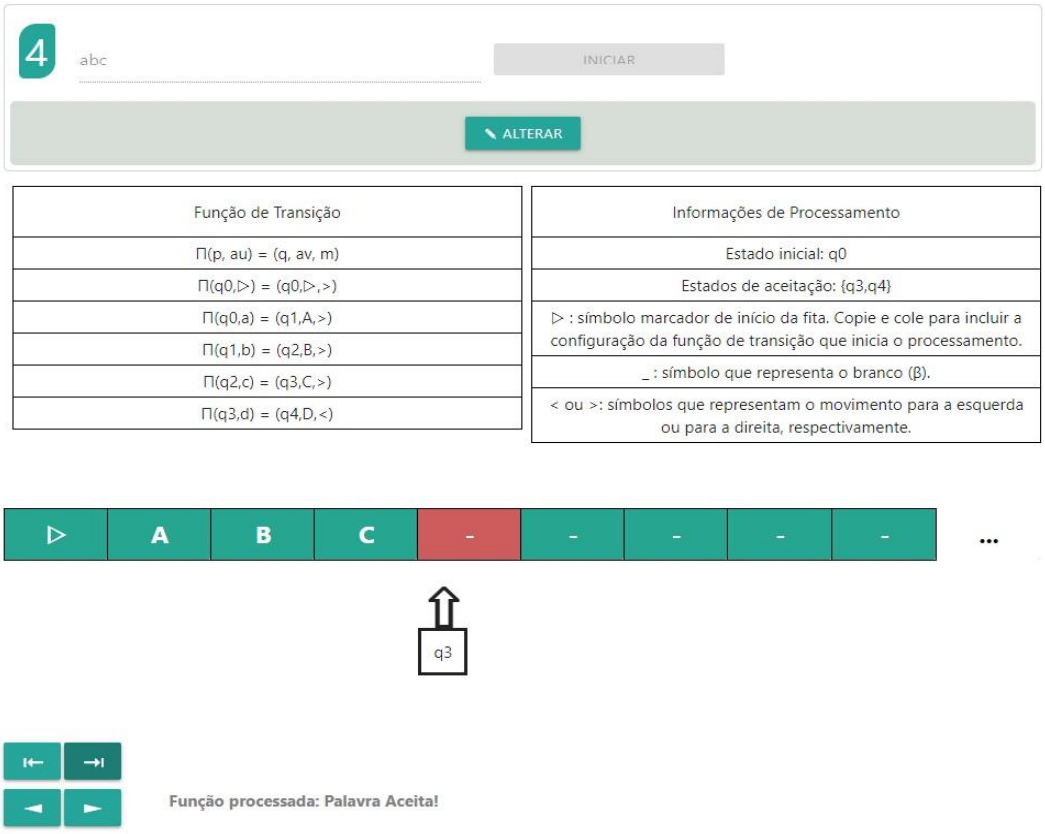
[14] Disponível em: https://turing-machine-simulator.herokuapp.com/. Acesso em: 06 de maio de 2021.

sobre a função a ser definida é exibido. Ao clicar no botão ‘OK’, a função de transição é armazenada. Já ao clicar no botão de edição (ícone de lápis), a página é estendida com opções de exclusão e edição de configurações e *reset* geral da máquina;

- **Elemento 7:** após a definição das estruturas dos elementos 4, 5 e 6, antes do processamento, uma entrada deve ser definida para ser processada pela máquina criada. Ao clicar no número 4, um modal com informações sobre o campo de entrada é exibido;
- **Elemento 8:** ao clicar neste botão, a fita é exibida e o processamento da entrada se inicia;
- **Elemento 9:** todas as configurações definidas da função de transição ficam registradas para exibição ao usuário;
- **Elemento 10:** a definição geral da máquina fica registrada para que o usuário possa consultar, bem como informações dos símbolos considerados no processamento.

A Figura 6 mostra o TuringMS sendo executado. Nesse exemplo, a máquina foi definida com o estado inicial ‘q0’, os estados de aceitação ‘q3’ e ‘q4’ e cinco configurações da função de transição que trocam letras minúsculas por suas equivalentes maiúsculas. O simulador inicia o processamento com a entrada ‘abc’ e finalizada aceitando a palavra.

Figura 6 - TuringMS simulando processamento



Fonte: Turing Machine Simulator (2021)[15]

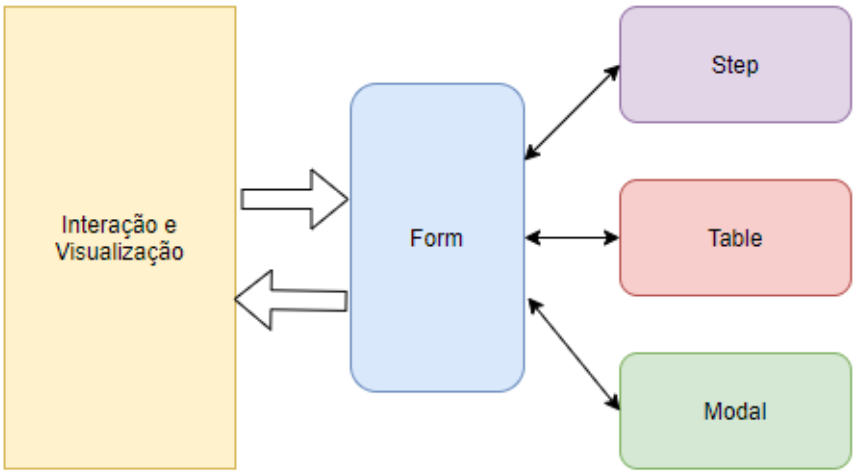
A Figura 7 ilustra a arquitetura dessa ferramenta que foi separada em componentes de acordo com suas responsabilidades:

- **Form:** é a estrutura principal, ela recebe todas as informações vindas do usuário e repassa esses dados somente para os outros componentes que necessitam deles;

[15] Disponível em: https://turing-machine-simulator.herokuapp.com/. Acesso em 06 de Maio de 2021.

- **Step:** recebe do Form todos os dados necessários para o processamento da entrada. É através dele que ocorre toda a visualização da simulação do processamento na fita.
- **Table:** recebe do Form as informações que precisam ser exibidas nas duas tabelas da página inicial. Quando o usuário realiza alguma mudança na definição da máquina, esses dados são novamente recebidos pelo Form e repassados para Table.
- **Modal:** é utilizado apenas para a exibição de informações sobre cada passo da definição da máquina. Ele não recebe informações, mas está contido no Form pois é ativado através da interação do usuário.

Figura 7 - Arquitetura do simulador TuringMS

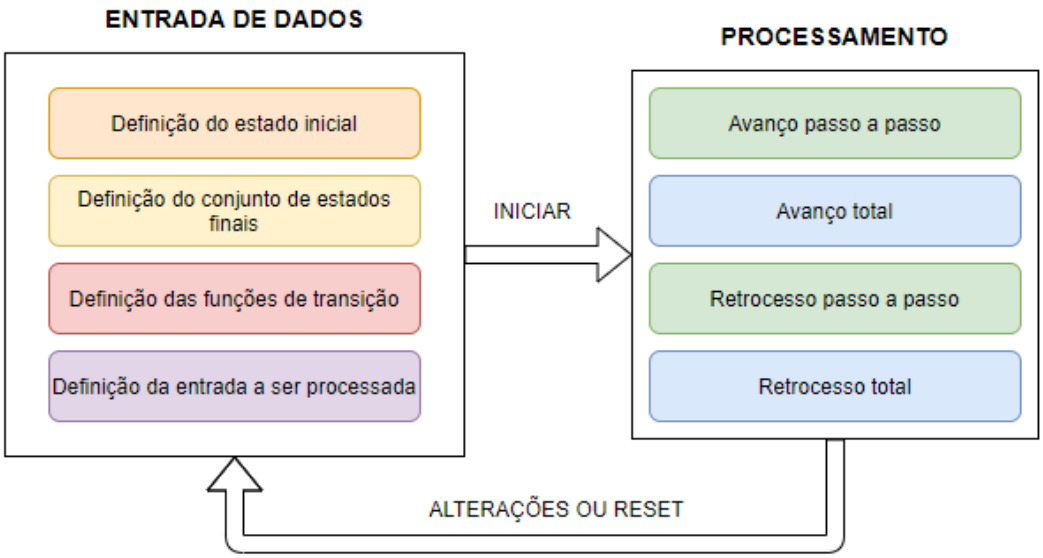


Fonte: Elaborada pela autora, 2021.

4.2 Simulação Gráfica de Processamento de Palavra

O funcionamento adequado do simulador depende do ciclo de ações ilustrado na Figura 8. Inicialmente, são 4 passos que devem ser cumpridos para realizar a simulação. Depois disso, o usuário poderá escolher como prefere visualizar o andamento do processamento da entrada, bem como também poderá alterar a máquina definida ou resetar todas as informações para uma nova definição.

Figura 8 - Ciclo de simulação do TuringMS

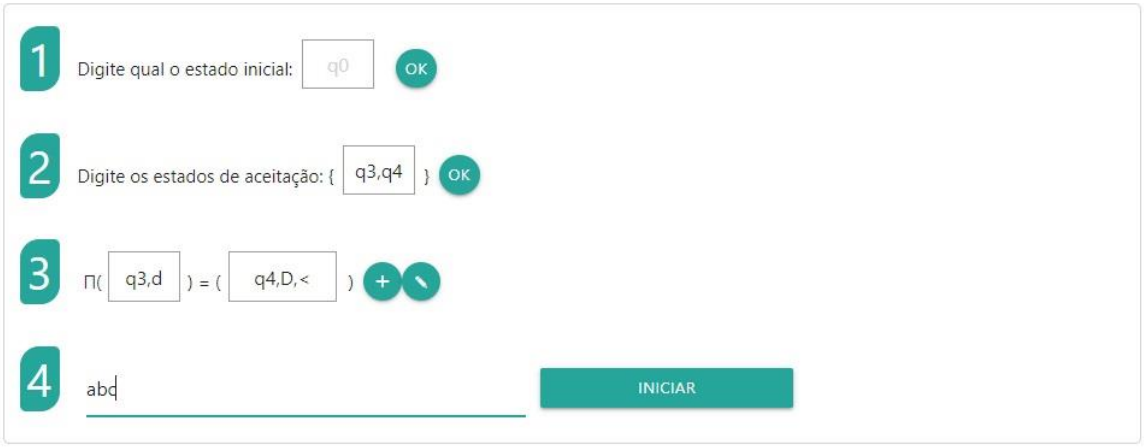


Fonte: Elaborada pela autora, 2021.

Na Figura 9, percebe-se que os três primeiros passos foram executados corretamente. No passo 1 o estado inicial foi definido e exibido na segunda linha da tabela de informações de processamento. Em seguida, no passo 2, o conjunto de estados finais foi definido (estados separados por vírgula) e exibido na terceira linha da mesma tabela. No passo 3, as transições foram definidas obedecendo as seguintes definições:

- No conjunto de configurações da função de transição, caso não deseje que sua entrada seja rejeitada no primeiro passo, deve-se definir uma transição que faça a leitura do símbolo marcador de início da fita, conforme exibido na terceira linha da tabela de função de transição. O símbolo marcador está informado na quarta linha da tabela de informações de processamento.
- Os campos para a inclusão das entradas da função de transição no passo 3 devem ser preenchidos levando em consideração o modelo **Π (p, au) = (q, av, m)**, onde ‘p’ é o estado corrente, ‘au’ o símbolo lido, ‘q’ o estado futuro, ‘av’ o símbolo a ser escrito e ‘m’ o movimento (conforme Figura 3). Conforme a tabela de informações de processamento, o ‘_’ é considerado o símbolo branco e ‘<’ / ‘>’ marcadores de movimento.

Figura 9 - Entrada de dados do TuringMS



Função de Transição
Π(p, au) = (q, av, m)
Π(q0,▷) = (q0,▷,>)
Π(q0,a) = (q1,A,>)
Π(q1,b) = (q2,B,>)
Π(q2,c) = (q3,C,>)
Π(q3,d) = (q4,D,<)

Informações de Processamento
Estado inicial: q0
Estados de aceitação: {q3,q4}
▷ : símbolo marcador de início da fita. Copie e cole para incluir a configuração da função de transição que inicia o processamento.
_ : símbolo que representa o branco (β).
< ou >: símbolos que representam o movimento para a esquerda ou para a direita, respectivamente.

Fonte: Turing Machine Simulator (2021)[16]

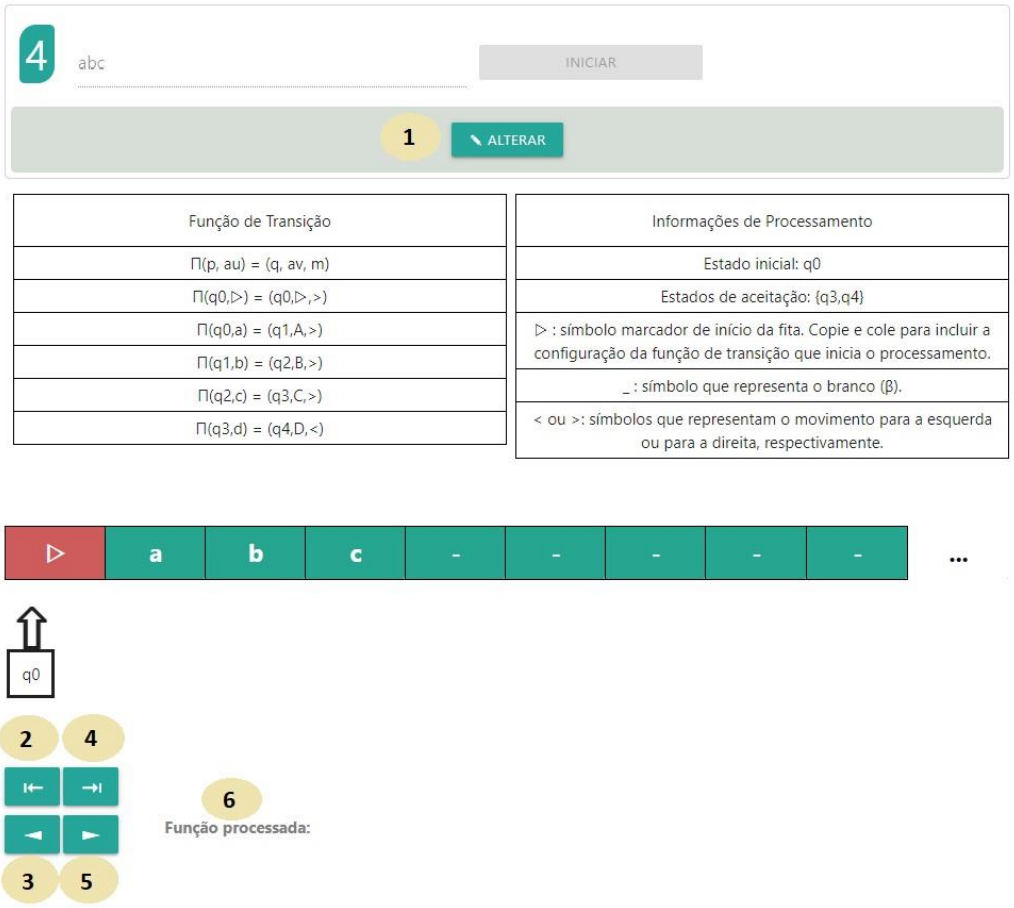
Para iniciar o processamento é preciso que uma entrada seja indicada no passo 4 e, em seguida,o botão 'INICIAR' seja pressionado. Na Figura 10, a entrada 'abc' foi definida e o processamento iniciado. Uma representação gráfica da fita da MT criada é exibida (conforme Figura 2). Quando uma das células está marcada em vermelho, significa que o apontador está naquela posição. Esse recurso foi pensado para facilitar a visualização do processamento em dispositivos móveis com telas menores.

Alguns novos elementos são exibidos nessa fase. São eles:

- **Elemento 1:** botão para liberar a edição dos dados fornecidos na etapa anterior;
- **Elemento 2:** botão para retroceder todo o processamento de uma única vez;
- **Elemento 3:** botão para retroceder apenas um passo do processamento por clique;
- **Elemento 4:** botão para avançar todo o processamento de uma única vez;
- **Elemento 5:** botão para avançar apenas um passo do processamento por clique;
- **Elemento 6:** *label* utilizada para dar *feedback* ao usuário sobre qual função de transição foi processada e sobre o resultado final do processamento (palavra aceita, rejeitada ou loop).

[16] Disponível em: https://turing-machine-simulator.herokuapp.com/. Acesso em 06 de Maio de 2021.

Figura 10 - Início da simulação do processamento de palavra no TuringMS

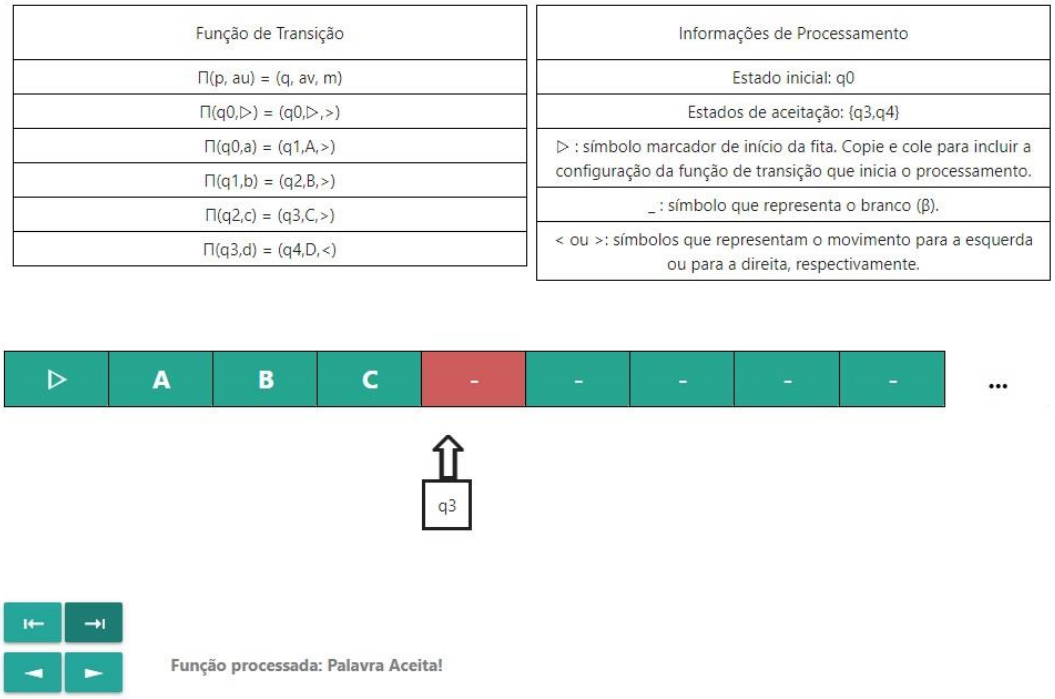


Fonte: Turing Machine Simulator (2021)[17]

Conforme já explicitado, fica a cargo do usuário escolher visualizar passo a passo o processamento da palavra ou seguir para o resultado final. Na unidade de controle da fita simulada, são exibidos todos os estados correntes durante o processamento. Ao finalizar todas as manipulações, conforme a definição da máquina, o *feedback* final é exibido, como ilustrado na Figura 11.

[17] Disponível em: https://turing-machine-simulator.herokuapp.com/. Acesso em 06 de Maio de 2021.

Figura 11 - Fim da simulação do processamento de palavra no TuringMS

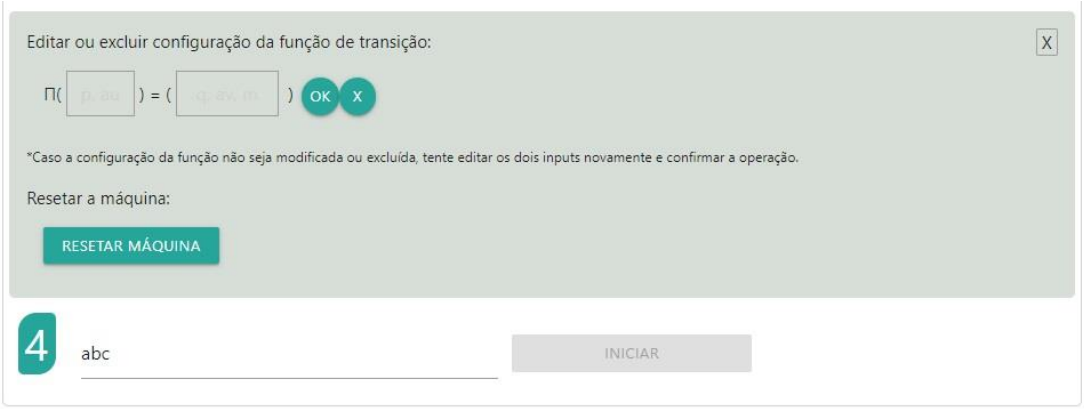


Fonte: Turing Machine Simulator (2021)[18]

Caso o usuário erre alguma definição ou apenas deseje alterar algum dado, basta clicar no botão ‘ALTERAR’ para liberar o formulário de informações novamente. Vale ressaltar que ao escolher alterar, a visualização da fita é omitida e, se escolher por voltar ao processamento com uma nova configuração, será do início. Também é possível realizar alterações ou exclusões no conjunto de configurações da função de transição através do *card* exibido na Figura 12. Para acessar o *card*, basta clicar no botão com o ícone de lápis ao lado dos campos de entrada de função.

[18] Disponível em: https://turing-machine-simulator.herokuapp.com/. Acesso em 06 de Maio de 2021.

Figura 12 - Alteração de definições no TuringMS



Função de Transição
Π(p, au) = (q, av, m)
Π(q0,▷) = (q0,▷,>)
Π(q0,a) = (q1,A,>)
Π(q1,b) = (q2,B,>)
Π(q2,c) = (q3,C,>)
Π(q3,d) = (q4,D,<)

Informações de Processamento
Estado inicial: q0
Estados de aceitação: {q3,q4}
▷ : símbolo marcador de início da fita. Copie e cole para incluir a configuração da função de transição que inicia o processamento.
_ : símbolo que representa o branco (β).
< ou >: símbolos que representam o movimento para a esquerda ou para a direita, respectivamente.

Fonte: Turing Machine Simulator (2021)[19]

Como pode-se observar na Figura 12, caso deseje alterar uma função, basta incluir a função alterada nos campos correspondentes no *card* de alteração e clicar no botão 'OK'. Caso deseje excluir, basta incluir a função a ser deletada nos campos correspondentes no *card* de alteração e clicar no botão 'X'. Essas mudanças são automaticamente refletidas na tabela de função de transição. Por fim, se necessitar *resetar* totalmente a máquina, basta clicar no botão 'RESETAR MÁQUINA' que todas as configurações definidas serão excluídas.

A utilização de recursos visuais gráficos tem por objetivo ilustrar a etapa ou o passo de transições de estado durante o processamento, explicitando a movimentação, leitura e escrita que ocorre durante toda execução. É possível, por exemplo, visualizar de forma clara a leitura de um caractere 'x' em uma célula, sendo substituído por um caractere 'y' na gravação e a movimentação para um dos lados logo após a escrita.

Através da visualização, torna-se mais fácil compreender o que acontece durante o processamento de reconhecimento de palavras por uma MT. Além disso, modais auxiliares ficam disponíveis durante todos os momentos da simulação para fornecer mais informações sobre cada passo necessário, bastando o usuário clicar no número do passo que desejar esclarecimentos.

Sobre o aspecto da navegação e facilidade de acesso, o TuringMS, além de não necessitar instalação, é responsivo, então todos os recursos disponíveis em uma visualização a partir de um desktop também estão disponíveis para dispositivos móveis, desde que ambos possuam conexão de rede.

4.3 Principais Tecnologias Utilizadas na Implementação

[19] Disponível em: https://turing-machine-simulator.herokuapp.com/. Acesso em 06 de Maio de 2021.

O TuringMS foi desenvolvido utilizando majoritariamente *ReactJs*[20] que, de acordo com Fernandes (2017), é uma biblioteca *JavaScript* que representa a camada *View/UI* da aplicação. Para a configuração geral do projeto foi utilizado o *Node.js* (versão 14.12.0), já que o *React* utiliza diversos pacotes e configurações para a aplicação. O TuringMS foi implementado levando em consideração o princípio básico do *React* que é a componentização, ou seja, componentes que possuem seu próprio estado se comunicando com outros componentes visando a simplificação da atualização de itens.

Como gerenciador de pacotes, o *Yarn*[21] (versão 1.22.5) é utilizado por padrão pelo o *React* e, segundo Souza (2020), serve para integrar pacotes prontos às aplicações, adicionando diretamente diversas funções. Essa é uma ferramenta que funciona basicamente como o *npm* (nativo do *Node.js*), mas é considerada mais segura e eficaz.

Muitos dos materiais de recursos visuais como botões, menus, modais, foram utilizados do framework *Materialize*, o qual é extremamente popular no desenvolvimento front-end. A utilização desses componentes facilitou alcançar o objetivo de implementar uma aplicação Web responsiva. Desse modo, como afirma Lopes (2014), não é nada trivial a criação de sites responsivos, pois é necessário que os componentes tenham a flexibilidade necessária para se adaptar ao dispositivo.

A IDE (*Integrated Development Environment* / Ambiente de Desenvolvimento Integrado) utilizada durante todo o desenvolvimento foi *Visual Studio Code* (versão 1.47.0), bem como suas integrações com o *Github*. Todo o projeto está versionado em um repositório[22] com o acesso totalmente livre e hospedado no *Heroku*[23] para acesso livre às funcionalidades do simulador pela rede.

4.4 Considerações (Parciais)

Essa seção apresentou o TuringMS, que é um simulador que visa apoiar o ensino de Linguagens Formais e Teoria da Computação por meio da simulação gráfica de processamento de palavras em uma MT. Esse simulador possibilita ao usuário aprendiz a execução de ciclos de processamento no seu próprio ritmo de entendimento, adiantando ou retrocedendo quando achar necessário. Ao usuário com domínio dos conceitos que regem o reconhecimento de palavras em uma MT, permite criar demonstrações gráficas mais complexas - desde que obedecendo às instruções e respeitando as limitações inerentes a cada navegador - conforme achar adequado.

Um dos pontos fortes do simulador é permitir que alunos e/ou professores naveguem e manipulem a aplicação a partir de dispositivos móveis, eliminando mais uma possível limitação de acesso e uso. Assim, seja em dispositivos desktop ou móveis, o usuário poderá construir uma simulação de processamento que o ajudará a transmitir os conceitos atrelados ou a absorvê-los de forma otimizada. No planejamento do TuringMS, foi priorizado que o processo de execução de simulações fosse baseado no ritmo de aprendizagem de cada um e, assim, o simulador foi concebido e construído.

5. CONCLUSÃO

[20] https://reactjs.org/
[21] https://yarnpkg.com/
[22] https://github.com/MayaraPereira/turing-machine-simulator
[23] https://turing-machine-simulator.herokuapp.com/

A relação ensino-aprendizagem tem estado cada vez mais em destaque através de discussões que indagam a efetividade de processos e padrões utilizados no ensino de maneira geral. Sendo assim, propostas visando a otimização dessa relação ganham mais atenção e a demanda cresce por soluções tecnológicas que agreguem valor às metodologias aplicadas. O TuringMS vem como um recurso gratuito, acessível e intuitivo para melhoria do ensino de Linguagens Formais e Teoria da Computação para os cursos superiores da área de tecnologia. Essa seção apresenta o fechamento do trabalho com suas conclusões. A subseção 5.1 demonstra as principais contribuições da pesquisa. A subseção 5.2 apresenta os trabalhos futuros. A subseção 5.3 faz as considerações finais deste trabalho.

5.1 Contribuições da Pesquisa

A pesquisa realizada neste artigo teve como resultado o simulador gráfico TuringMS. Foi desenvolvido para que o aluno obtenha, através da simulação gráfica, uma visão menos abstrata dos conteúdos relacionados à MT e para que o professor consiga de maneira prática demonstrar graficamente as abstrações teóricas envolvidas nesse assunto.

O TuringMS, diferentemente dos principais simuladores propostos para o ensino de Linguagens Formais e Teoria da Computação, é uma aplicação web, ou seja, não necessita instalação e também é totalmente utilizável a partir de dispositivos móveis. Fundamentado no princípio da componentização através do ReactJs, fazendo o uso dos propósitos dessa tecnologia, proporciona uma navegação intuitiva, necessitando apenas de conexão à rede e um navegador. Dessa maneira, o usuário se envolve visualmente em toda construção da máquina e, principalmente, na construção do seu próprio conhecimento de maneira exploratória e interativa. Por intermédio do TuringMS, é possível simular inúmeras configurações da MT, possibilitando uma otimização da demonstração e entendimento de conceitos abstratos na área do ensino da computação. A utilização dos conceitos de componentização voltada à interface do usuário permite que novas funcionalidades e/ou melhorias sejam incluídas sem grandes dificuldades. Por este motivo, toda a implementação deste simulador é aberta, visando somente a obtenção crescente de recursos para a otimização do ensino teórico nos cursos de tecnologia.

A principal contribuição dessa pesquisa é o produto obtido pela implementação do TuringMS: uma solução tecnológica acessível para otimizar a absorção do aprendizado dos conceitos abstratos encontrados no ensino de Linguagens Formais e Teoria da Computação.

5.2 Trabalhos Futuros

Para trabalhos futuros, seguem as propostas sugeridas:

- Realizar testes com grupos no que diz respeito à relação ensino-aprendizagem dos alunos utilizando o TuringMS nas disciplinas de Linguagens Formais e Teoria da Computação;
- Adicionar funcionalidades ao TuringMS para permitir a simulação de uma MT como processadora de funções computáveis;
- Implementar uma solução do TuringMS para permitir a configuração de Máquinas de Turing diversificadas (com duas fitas, por exemplo);
- Implementar uma solução do TuringMS para incluir uma visualização de processamento a partir de autômatos;
- Verificar, através de testes, o poder de processamento e limitações do TuringMS executado em navegadores de PCs (*Personal Computer* / Computador Pessoal) e dispositivos móveis;
- Fazer um comparativo entre o TuringMS e outros simuladores com a mesma proposta.

5.3 Considerações Finais

Na necessidade de melhorias, inovação e entusiasmo no processo de ensino-aprendizagem, os simuladores se tornam grandes aliados para suprirem essas demandas. Eles podem ser usados com esse fim desde o ensino básico. Contudo, ainda tem sido um recurso pouco utilizado no apoio ao ensino da Computação, especificamente nas disciplinas de Linguagens Formais e Teoria da Computação.

Esse trabalho explorou, no contexto computacional de disciplinas majoritariamente teóricas, os benefícios que a utilização de simuladores podem trazer para a relação de ensino-aprendizagem, bem como as cautelas a serem tomadas. É possível dizer que todo o planejamento inicial do simulador, desde o levantamento das principais dificuldades no ensino de disciplinas teóricas de Computação até a escolha da arquitetura e tecnologia, que seria utilizada para implementação do TuringMS, tem uma relação direta com as melhorias a ser obtidas no aprendizado por meio da utilização deste.

O TuringMS tem capacidade de contribuir para a melhoria do ensino de conceitos abstratos da MT nas disciplinas de Linguagens Formais e Teoria da Computação através da visualização e da possibilidade constante de interação do usuário com o que ocorre no processamento. Dessa maneira, professores e alunos são imergidos em uma experiência em que eles são os protagonistas. Sendo assim, o processo de ensino tende a se tornar mais real e empolgante, tanto para quem ensina como para quem aprende.

Esse trabalho, em sua totalidade, representou grandes desafios. Em particular, a utilização de recursos tecnológicos mais atuais para proporcionar a responsividade da aplicação e também o estudo e a preocupação em implementar algo que realmente agregue ao ensino de fundamentos teóricos tão importantes para os profissionais da computação (principalmente aos que desejam seguir carreira de pesquisador). Nenhum esforço teria sentido se a contribuição não fosse acessível a todos, por isso foi disponibilizado em repositório aberto, para que esse seja o início de grandes evoluções no processo de ensino-aprendizagem de Linguagens Formais e Teoria da Computação por meio do TuringMS.

REFERÊNCIAS

AGUIAR, R.S.; OEIRAS, J. Y. **Laboratório de Linguagens Formais - LabLF**. [S.l.]: Revista Brasileira de Informática na Educação, vol. 18, nº 1, 2010. Disponível em: <https://www.br-ie.org/pub/index.php/rbie/article/view/1208>. Acesso em: 30 outubro 2020.

ALVES, J. A.; OLIVEIRA, E. R. A.; FERREIRA, F.; MIGUENS, S.; PEREIRA, L. M.; POMBO, O.; QUARESMA, P.; VALENÇA, J. M. **Alan Turing**: cientista universal. Braga: UMinho Editora, 2019. *E-book.*

DIVERIO, T. A.; MENEZES, P. B. **Teoria da computação**: máquinas universais e computabilidade. Porto Alegre: Editora Sagra Luzzatto, 1999.

DIVERIO, T. A.; MENEZES, P. B. **Teoria da computação**: máquinas universais e computabilidade. 3. ed. Porto Alegre: Bookman, 2011.

DONATO, C. **Introdução à programação**. Joinville: Clube de Autores, 2019.

FERNANDES, D. **React do zero: componentização, propriedades e estado**. Blog Rocket Seat, 2017. Disponível em: https://blog.rocketseat.com.br/react-do-zero-componentizacao-propriedades-e-estado/. Acesso em: 11 maio 2021.

FURTADO, O. J. V. **O ensino de Linguagens Formais vinculado ao ensino de Compiladores**. In: XI WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO - SBC, 11. Campinas: 2003, p. 1-8.

LOPES, S. **A Web Mobile:** Design Responsivo e além para uma Web adaptada ao mundo mobile. São Paulo: Casa do Código, 2014. *E-book.*

MENEZES, P. B. **Matemática discreta para computação e informática**. 4. ed. Porto Alegre: Bookman, 2013.

RAMOS, M. V. M. **Ensino de linguagens formais e autômatos em cursos superiores de computação**. Revista de Computação e Tecnologia da PUC-SP (ReCeT), São Paulo, vol. 1, n° 1, p. 22-34. 2009.

SOUZA, I. **O que é Yarn e como funciona seu gerenciamento de pacotes**. Blog Rock Content, 2020. Disponível em: https://rockcontent.com/br/blog/yarn/. Acesso em: 11 maio 2021.

AGRADECIMENTOS

Ao professor Edson Holanda Cavalcante Júnior por toda dedicação e paciência ao longo dessa orientação.

Aos professores da banca examinadora pela honra de terem aceitado a incumbência de avaliar e colaborar com este trabalho.

À todos os professores que fazem parte do curso de Computação por tantos ensinamentos transmitidos com tanto empenho.