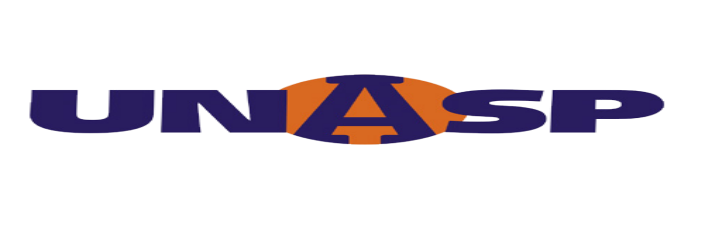
**UNASP**

**CENTRO UNIVERSITÁRIO ADVENTISTA DE SÃO PAULO**

**CAMPUS SÃO PAULO**

**CURSO DE CIÊNCIA DA COMPUTAÇÃO**



**Wellmmer Lucas de Oliveira Pinto**

**Simulador de Máquina de Turing: Uma Aplicação Web para o Aprendizado**

Trabalho de conclusão de curso apresentado ao Centro Universitário Adventista para obtenção do título de Bacharel em Ciência da Computação.

**São Paulo, SP, Novembro de 2016**

**Wellmmer Lucas de Oliveira Pinto**

**Simulador de Máquina de Turing: Uma Aplicação Web para o Aprendizado**

Trabalho de conclusão de curso apresentado ao Centro Universitário Adventista para obtenção do título de Bacharel em Ciência da Computação

**Área de Concentração:**

Ciência da Computação

**Orientador:**

Prof. Ms. Laercio Martins Carpes

**São Paulo, SP, Novembro de 2016**

Dedico este trabalho ao homem mais sábio deste mundo na minha opinião, e a mulher com a maior fé que já vi em toda minha vida.... Meus queridos e amados pais.

**AGRADECIMENTOS**

Agradeço à Deus em primeiro lugar pois “tudo foi possível nAquele que sempre me fortaleceu”. Também dirijo meus agradecimentos aos meus amigos que me ajudaram de diversas formas para conclusão deste trabalho: Guilherme, Lucas e Gabriel. Sem muitas delongas agradeço imensamente a minha fiel companheira Layla Camargo que esteve ao meu lado inúmeras manhãs, tardes, noites e madrugadas, me ajudando a escrever, revisar e desenvolver este trabalho. E por fim, agradeço aos meus amados pais por todo apoio, investimento e confiança.

**RESUMO**

O legado deixado por Alan Turing (1912–1954), com certeza, é inegavelmente grandioso e tem contribuído com estudos e pesquisas até os dias de hoje. O artigo nomeado *“On Computable Numbers, with an Application to the Entscheidungsproblem*” contempla uma de suas maiores invenções e que hoje usamos e chamamos de computadores. Turing detalha neste artigo sua teoria sobre os números computáveis, máquinas computáveis e como o que ele chamou de “máquina universal”, conhecida hoje por máquina de Turing, atenderia como resposta ao tão polêmico *“Problema de Decisão”*, ou do original alemão “*Entscheidungsproblem*”*,* de David Hilbert (1862-1943). No entanto, a linguagem usada por Turing para explanar suas ideias soa muitas vezes de forma complicada e por assim dizer, muito específica à matemática. Segundo LEAVITT (2007) Turing desbrava, em seu artigo, pântanos de símbolos não familiares, letras alemãs e gregas, e números binários, além de uma linguagem filosófica e matemática altamente técnica. No entanto, pensando em simplificar e auxiliar no estudo da máquina de Turing que este trabalho propõe uma aplicação web voltada para o aprendizado dos conceitos e funcionamento da máquina de Turing, incluindo um tutorial e um construtor e simulador de máquina de Turing.

**Palavra Chave:** Máquina de Turing; Simulador; Aplicação Web.

**ABSTRACT**

The legacy left by Alan Turing (1912-1954), certainly, is undeniably magnificent and has contributed to studies and research until these days. The article named "On Computable Numbers, with an Application to the Entscheidungsproblem" contemplates one of his greatest inventions and that today we use and call computers. Turing details in this paper his theory on computable numbers, computable machines and how what he called "universal machine", known today as Turing machine, would answer in response to the controversial "Decision Problem" or the German original "Entscheidungsproblem" of David Hilbert (1862-1943). However, the language used by Turing to explain his ideas often sounds complicated and, so to speak, very specific to mathematics. According to LEAVITT (2007) Turing raises, in his article, swamps of unfamiliar symbols, German and Greek letters, and binary numbers, in addition to a highly technical philosophical and mathematical language. However, thinking of simplifying and assisting in the study of the Turing machine, this project proposes a web application focused on learning the concepts and functioning of the Turing machine, including a tutorial and a Turing machine builder and simulator.

**Keywords:** Turing Machine; Simulator; Web Application.

**LISTA DE FIGURAS**

[Figura 1 - Turing Machine Simulator, Paul Rendell 18](#_Toc468371915)

[Figura 2 - Turing Machine, Martin Ugarte 19](#_Toc468371916)

[Figura 3 - Tela Inicial 20](#_Toc468371917)

[Figura 4 - Tutorial – Slides 01 20](#_Toc468371918)

[Figura 5 - Tutorial – Slides 02 21](#_Toc468371919)

[Figura 6 – Tutorial – Slides 03 22](#_Toc468371920)

[Figura 7 – Tutorial – Slides 04 22](#_Toc468371921)

[Figura 8 – Tutorial – Slides 05 23](#_Toc468371922)

[Figura 9 – Tutorial – Slides 06 24](#_Toc468371923)

[Figura 10 – Tutorial – Slides 07 24](#_Toc468371924)

[Figura 11 – Tutorial – Slides 08 25](#_Toc468371925)

[Figura 12 – Tutorial – Slides 09 25](#_Toc468371926)

[Figura 13 – Tutorial – Slides 10 26](#_Toc468371927)

[Figura 14 – Tutorial – Vídeo 26](#_Toc468371928)

[Figura 15 - Acesso RápidoFonte: Elaborada pelo autor 27](#_Toc468371929)

[Figura 16 - Construtor 28](#_Toc468371930)

[Figura 17 – Estados e Transições 29](#_Toc468371931)

[Figura 18 – Painel de Controle 30](#_Toc468371932)

[Figura 19 – Visualização dos Processos da Máquina 31](#_Toc468371933)

**SUMÁRIO**

[1 INTRODUÇÃO 3](#_Toc468372148)

[1.1 Argumentação Justificativa 10](#_Toc468372149)

[2 REFERENCIAL TEÓRICO 11](#_Toc468372150)

[2.1 Entscheidungsproblem: O Problema de Decisão 11](#_Toc468372151)

[2.2 A solução de Alan M. Turing 12](#_Toc468372152)

[2.2.1 O que são “números computáveis”? 12](#_Toc468372153)

[2.2.2 O que são “máquinas computáveis”? 12](#_Toc468372154)

[2.3 A Enigma 13](#_Toc468372155)

[2.4 Autômatos Finitos 14](#_Toc468372156)

[2.5 E o “Lambda Calculus”? 16](#_Toc468372157)

[3 PROJETO 17](#_Toc468372158)

[3.1 Objetivo Geral 17](#_Toc468372159)

[3.2 Metodologia 17](#_Toc468372160)

[3.3 Projetos Relacionados 18](#_Toc468372161)

[3.4 Desenvolvimento e Resultados 19](#_Toc468372162)

[3.5 Código Fonte 31](#_Toc468372163)

[4 CONCLUSÃO 40](#_Toc468372168)

[5 REFERÊNCIAS 42](#_Toc468372169)

# **INTRODUÇÃO**

De acordo com a versão online do *American Heritage Dictionary* (2016), a palavra “algoritmo” é um processo de solução de um problema passo a passo, especialmente um procedimento computacional recursivo estabelecido para a solução de um problema num número de passos finitos. Embora hoje seja conhecido o significado da palavra “algoritmo”, ao voltar para o século XIII onde tudo isso estava no escuro da mente de alguns homens ousados, contemplamos uma história por trás do que hoje é chamado de computador.

No terceiro capítulo do livro *“O homem que sabia demais”*, de David LEAVITT (2007), é esboçada uma introdução muito detalhada da história por trás do *Entscheidungsproblem*, ou melhor dizendo, o problema de decisão. Começando por Raimundus Lullus (1232-1316) e toda sua tese em cima do método geral de solução de problemas, intitulado por ele de “*ars magna”.* Mais tarde ampliado pelos estudos de Leibniz (1646-1716), resultando no estabelecimento de uma linguagem simbólica que efetivasse a solução do problema, a “*characteristica universalis”*, e na distinção de duas versões da *ars magna:* a “*ars inveniendi”*, encontrando as verdadeiras afirmações científicas, e a “*ars iudicandi”*, permitindo a decisão se uma afirmação científica é verdadeira ou não.

Quase um século depois, o matemático alemão David Hilbert (1862-1943) lança para a comunidade de matemáticos da época o *Entscheidungsproblem,* que caía na perspectiva do “*ars iudicandi”* deLeibniz, no qual, segundo Hilbert poderia se restringir a uma simples questão de “sim” ou “não” para o caso de haver um algoritmo que decida a validade de uma fórmula de primeira ordem. Isso gerou muita polêmica entre a comunidade de matemáticos, quando de um lado tinham matemáticos prós à uma descoberta para o problema de decisão, e do outro matemáticos contra, como o caso do matemático inglês G. H. Hardy (1877-1947), ao dizer que “isso seria uma infelicidade, pois se houvesse [o algoritmo], precisaríamos ter um conjunto mecânico de regras para a solução de todos os problemas matemáticos, e nossas atividades como matemáticos chegariam a um fim.”.

No entanto, havia um jovem britânico, por nome Alan Mathison Turing, que talvez por seu constante isolamento social, não estava de nenhum dos lados em relação a solução do *Entscheidungsproblem*, no contrário, o encarava como um mero problema que almejava uma resposta. Talvez pelo motivo claro de Turing não considerar o problema esperando um possível resultado negativo ou positivo, que ele obteve sucesso ao enfrentar de uma maneira totalmente inovadora. Todavia, foi com este modo tão literal de pensar, que Turing apresentou seus resultados, em 1936, no artigo publicado *“On Computable Numbers, with an Application to the Entscheidungsproblem”*, no qual ele define o conceito e o funcionamento da “máquina universal”, e desbanca o problema de decisão. (LEAVITT, 2007)

## **JUSTIFICATIVA**

Em um breve artigo, escrito sob encomenda e destinado à alunos de Ciência da Computação, o renomado Prof. Dr. Eng. Valdemar W. Setzer, do Depto. de Ciência da Computação da USP, destaca a importância dos estudos e legado deixados por Turing ao enfatizar:

*“Se o leitor está terminando um Bacharelado em Ciência da Computação, e nunca ouviu falar o nome Turing, é melhor começar tudo de novo em uma faculdade de um nível razoável. Isso se deve ao fato de Alan Mathison Turing (1912-1954) ter estabelecido alguns dos fundamentos mais importantes da Ciência da Computação, tanto do ponto de vista prático quanto teórico e, portanto, deve obrigatoriamente ser mencionado e suas descobertas estudadas em qualquer curso dessa área.” (SETZER, V. W. 2003).*

Seguindo este raciocínio, embora o aprendizado dos conceitos da *“máquina universal”* de Turing, contidos em seu artigo citado acima, sejam imprescindíveis tanto para um ingressante quanto para um formando de Bacharelado em Ciência da Computação, como em qualquer outro curso na área da tecnologia, é viável por antemão ser assumido que a compreensão de tais não é algo tão trivial. O estudo sobre a *“máquina universal”* de Turing é algo que pode parecer simples superficialmente, porém demanda um aprofundamento técnico matemático para compreender certas representações, equações e símbolos usados por Turing para explanar suas ideias. Segundo LEAVITT (2007) Turing desbrava, em seu artigo, pântanos de símbolos não familiares, letras alemãs e gregas, e números binários, além de uma linguagem filosófica e matemática altamente técnica.

## **OBJETIVOS**

Portanto, pensando carinhosamente no legado e todo conhecimento deixado por Alan Turing, que este trabalho procura por objetivo imputar os conceitos e funcionalidades da máquina de Turing em uma aplicação web e, através de uma forma simples e didática possa auxiliar na aprendizagem dos fundamentos da teoria da computação deixados por sua descoberta.

### **Objetivo Geral**

Desenvolver uma aplicação web para simplificar e auxiliar no aprendizado dos conceitos e funcionamento da máquina de Turing, de forma que tais sejam abstraídos e aplicados de forma coerente.

### **Objetivos Específicos**

Desenvolver uma aplicação web para simplificar e auxiliar no aprendizado dos conceitos e funcionamento da máquina de Turing.

## **RECURSOS E VIABILIDADE**

Como recurso necessário para a viabilização deste trabalho será utilizado um computador, ou notebook, no qual as ferramentas necessárias para o desenvolvimento e implementação dos objetivos propostos serão instaladas e configuradas , sendo elas:

* Visual Studio Code - versão 1.7.2
* Javascript - versão 1.8.5
* jQuery - versão 3.1.1
* jQuery UI - versão 1.11.4
* Bootstrap - versão 3.3.7

## **ORGANIZAÇÃO DO TEXTO**

O trabalho a seguir foi dividido em capítulos. O primeiro capítulo apresenta uma breve introdução ao tema, uma justificativa ao projeto proposto, assim como os objetivos, especificando tanto o geral quanto os específicos, e finalizando ao detalhar os recursos e viabilidade do projeto. O segundo capítulo é composto pela relação do referencial teórico utilizado como base do estudo preparatório para o desenvolvimento da aplicação. O terceiro capítulo vem abordando a metodologia aplicada ao projeto. O quarto capítulo explica o desenvolvimento da aplicação web, detalhando cada parte implementada. O quinto capítulo será discutido a análise dos resultados em relação aos testes aplicados em cima da aplicação web. O sexto capítulo é o desfecho do tema, comentando todas as ideias propostas e cumpridas ao decorrer do trabalho.

# **REFERENCIAL TEÓRICO**

## ***Entscheidungsproblem*: O Problema de Decisão**

Conforme HILBERT & ACKERMANN (1928), o *Entscheidungsproblem (“Problema de Decisão”),* é resolvido quando conhecemos um procedimento que permite, para qualquer expressão lógica dada, decidir sua validade ou satisfatibilidade. BOOLOS, BURGESS & JEFFREY (2007) completam afirmando que “uma expressão apenas é satisfazível se houver a possibilidade de uma interpretação ou modelo para torná-la verdadeira”.

Em 1928, David Hilbert propôs seu décimo desafio à comunidade matemática, chamado *“Problema de Decisão”* ou *“Entscheidungsproblem”*, do original alemão. O desafio se baseava em um certo algoritmo, por assim dizer, que recebe como entrada dados em linguagem formal e dados matemáticos, que outrora retornam ao usuário um sinal de “verdadeiro ou falso” quanto a determinada validação. O algoritmo tem o objetivo de validar qualquer expressão lógica à ele atribuída, retornando então se tal expressão é ou não válida e, também, o seu nível satisfatório.

QUEIROZ (2012) acrescenta dizendo:

*“Tal algoritmo seria capaz de decidir, por exemplo, se enunciados tais como a conjectura de Goldbach ou a hipótese de Riemann, são verdadeiras, muito embora nenhuma prova ou refutação desses enunciados seja conhecida.” (QUEIROZ, 2012)*

Antes da questão imposta pelo “*Problema de Decisão”* ser respondida, a noção do significado de algoritmo teria de ser formalmente definida. Tais definições foram feitas, de acordo com LEAVITT (2007) e TURING (1936), por [Alonzo Church](https://en.wikipedia.org/wiki/Alonzo_Church" \o "Alonzo Church) em 1936 com o conceito de *“calculabilidade efetiva”* baseado em seu “*[lambda calculus](https://en.wikipedia.org/wiki/Lambda_calculus" \o "Lambda calculus)”*e por Alan Turing, no mesmo ano, com seus conceitos ousados de computabilidade e máquinas de [Turing](https://en.wikipedia.org/wiki/Turing_machine" \o "Turing machine). É importante notar que Turing imediatamente reconheceu equivalência de seus conceitos com os modelos de computação, além de ter comparado seus resultados obtidos com os de Church, sendo também equivalentes em sua essência.

Seguindo LEAVITT (2007), uma resposta negativa foi dada ao *“Problema de Decisão”*, a partir dos resultados apresentados independentemente por Alonzo Church e Alan Turing em 1936. De um lado, Church provava que não há função computável que decida por duas expressões de cálculo lambda quer elas sejam equivalente ou não. Do outro lado, Turing comprimiu a questão da existência de que um algoritmo ou método geral capaz de solucionar o *“Problema de Decisão”* a uma simples questão de existir um método geral que decida se qualquer máquina de Turing é interrompida ou não. Ou seja, quer um algoritmo seja entendido como sendo equivalente à uma máquina de Turing, e tem resultado negativo como resposta, a questão de existir um algoritmo para o *“Problema de Decisão”* também deve ser negativa.

Aplicando seus conceitos das máquinas de Turing ao *Entscheidungsproblem (“Problema de Decisão”*), TURING (1936) conclui dizendo:

*"Correspondendo a cada máquina de computação “M”, construímos uma fórmula “Un (M)” e mostramos que, se existe um método geral para determinar se “Un (M)” é provável, então há um método geral para determinar se “M” nunca imprime 0 ".*

## **Hipótese Turing-Church**

De acordo com MENEZES (#), a máquina de Turing concebida em 1936 equivale à um modelo abstrato de computação, com o objetivo de explorar ao máximo os limites da capacidade de expressar solução de problemas. Os resultados de Turing tratam-se então de uma proposta de definição formal da noção intuitiva de algoritmo. Todo o esforço gerado a partir dos resultados obtidos de trabalhos independentes como o de Alan Turing e o de Alonzo Church serviram como reforço para a *Hipótese Turing-Church,* no qual diz, basicamente, que a capacidade computacional representada por uma máquina de Turing é o alcance máximo que pode ser atingido por qualquer outro método mecânico ou dispositivo capaz de expressar algoritmos.

## **Máquina de Turing**

Segundo DIVERIO & MENEZES (2011), a máquina de Turing é um modelo computacional universalmente conhecido e aceito como formalização de algoritmo. Portanto, trata-se de um simples mecanismo que formaliza a ideia de um indivíduo capaz de realizar cálculos.

TURING (1936) descreve que a *“máquina”* é alimentada, por assim dizer, com uma “*fita”* percorrendo sua estrutura. A *“fita”* é dividida em seções, como *“células”*, cada uma capaz de receber um *“símbolo”*. A *“célula”* que, no momento, estiver na *“máquina”* é chamada de *“célula registrada”,* quando que o *“símbolo”* da *“célula registrada” é* chamado *“símbolo registrado”*. O possível comportamento da *“máquina”*, para qualquer situação, pode ser determinado pela *“configuração-m”* juntamente com o *“símbolo registrado”*, definindo a *“configuração da máquina”.* LEAVITT (2007) resume os possíveis comportamentos da máquina ao dizer:

*“Dependendo de sua configuração a máquina vai escrever um símbolo numa célula em branco, apagar um símbolo já escrito lá, mover a fita um espaço para esquerda ou mover a fita um espaço para direita. O que determina como ela irá agir é uma ‘tabela de comportamento’ especificando a sequência das configurações-m de acordo com as quais a máquina pode executar seu algoritmo particular”. (LEAVITT, 2007)*

Todavia, para um melhor entendimento do conceito de máquina de Turing é interessante dividí-lo três partes: noção intuitiva, noção de máquina e modelo formal.

# **Noção Intuitiva**

Baseando-se na interpretação de DIVERIO & MENEZES (2011), Turing partiu do pressuposto de analisar a situação de um indivíduo equipado com um instrumento de escrita e um apagador, no qual realiza cálculos em uma folha de papel dividida em quadrados. De início, é viável supor que a folha de papel contém apenas os dados iniciais de determinado problema. Assim, presumimos que o trabalho deste indivíduo pode ser resumido em sequências de operações simples como:

* Ler um símbolo de um quadrado;
* Modificar um símbolo de um quadrado;
* Mover os olhos para outro quadrado.

Logo que encontrado alguma representação satisfatória para a resposta almejada, os cálculos deste indivíduo são finalizados. Portanto, para tornar viável cada parte deste procedimento, hipóteses a seguir são aceitáveis:

* A natureza limita e bidimensional da folha de papel não são critérios essenciais para os cálculos. Desde já assumindo que a folha de pape pode consistir de uma fila infinita e seccionada, ou melhor dizendo, dividida em quadrados;
* O conjunto de símbolos pode ser finito, podendo-se utilizar de sequências de símbolos;
* O conjunto de estados da mente do indivíduo durante o processo de cálculo é finito. Notando que dentre esses estados há dois especiais, o “estado inicial” e o “estado final”, no qual correspondem respectivamente ao início e fim do processo de cálculo do indivíduo;
* Todo o comportamento do indivíduo a cada momento é definido apenas pelo estado atual e pelo símbolo para o qual sua atenção está voltada;
* O indivíduo é capaz de observar e modificar o símbolo de apenas um quadrado de cada vez, assim como dar atenção à somente um quadrado adjacente

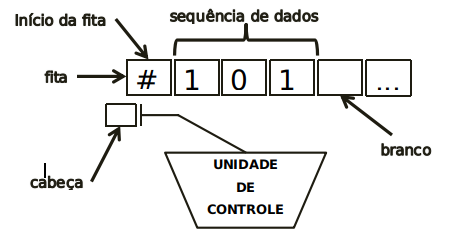
# **Noção de Máquina**

Ainda na ideia de DIVERIO & MENEZES (2011), a noção de um indivíduo calculando pode ser facilmente vista como uma máquina, composta por partes muito importantes, como segue:

* Uma *fita*, utilizada simultaneamente como mecanismo de memória e entrada e saída de dados;
* Um *unidade de controle*, que reflete o estado atual da máquina, possui uma *unidade de leitura e escrita*, por vezes chamada de *cabeça*, a qual acessa somente uma célula da fita de cada vez e pode movimentar-se para esquerda ou para direita;
* Um *programa* ou *função de transições*, onde é determinado anteriormente o estado inicial e o estado final da máquina, as ações que irão ser realizadas em determinado estado atual da máquina, incluindo comandos de leitura, escrita e sentido de movimento da cabeça.

A *fita* é finita para esquerda e infinita para direita, estritamente dividida em células, na qual cada um contém um símbolo. Os símbolos, por sua vez, podem pertencer ao alfabeto da máquina, ao alfabeto auxiliar ou até mesmo ser *“branco”* ou *“marcador de início da fita”.* Até então, a sequência de dados, a ser processada pela máquina, preenche as células mais à esquerda após o *marcador de início da fita*, como ilustrado na *Figura 1* abaixo.

Figura 1 - Ilustração de Máquina de Turing



Fonte: Elaborada pelo autor.

A *unidade de controle* possui um número finito e pré-determinado de estados. A *unidade de leitura e escrita*, ou *cabeça*, pode ler somente um símbolo de cada vez e escrever um novo símbolo em cima no símbolo lido. Após a leitura e escrita, a cabeça se move para esquerda ou para direita. Estes comandos são definidos pelo *programa* ou *função de transição*. O *programa* é uma função que determina o símbolo a ser escrito e o próximo estado da máquina de acordo com o estado atual e o símbolo lido.

# **Modelo Formal**

De modo formal, pode-se dizer que uma máquina de Turing é uma representada por 7-upla:

**Μ = (Q, ∑, Ґ, δ, q0, qA, qR)**

Sendo **Q, ∑, Ґ** são todos conjuntos finitos, onde:

1. **Q** representa o conjunto de estados da máquina;
2. **∑** representa o alfabeto de entrada da máquina, não contendo o símbolo especial *”branco”* **β**;
3. **Ґ** representa o alfabeto da fita, onde **β** pertence a **Ґ** e **∑** está incluído em **Ґ**;
4. **δ** é uma função parcial que define que representa as transições da máquina, sendo esta:

**δ: (Qatual , Ґlido) > (Qnovo , Ґnovo , {E,D})**

* 1. Ou seja, cada transição da máquina pode ser representada com a função acima quando expressado o estado atual **Qatual** da máquina e o símbolo lido **Ґlido**.
  2. Assim, a máquina tem a função de ir para um novo estado **Qnovo**, escrever por cima do símbolo lido **Ґlido** um novo símbolo **Ґnovo** e direcionar a *cabeça (unidade de leitura e escrita)* para esquerda**{E}** ou direita **{D}**da *fita*.

1. **q0** representa o estado inicial da máquina, ou seja, o ponto de partida.
2. **qA** representa o estado de aceitação da máquina, pode ser definido como um estado final da máquina.
3. **qR** representa o estado de rejeição da máquina, também pode ser definido como um estado final da máquina.

A representação formal da máquina de Turing pode sofrer alterações devido as inúmeras interpretações da máquina de Turing. No entanto, todas elas...

## **A *Enigma***

De acordo com WINTERBOTHAM (1978), *Enigma* é o nome pelo qual foi conhecida a máquina eletromecânica de criptografia com rotores, utilizada tanto para criptografar e descriptografar códigos de guerra. Tratava-se mais especificamente de uma máquina de porte médio, com medidas 65x45x35cm e pesando cerca de 50 kg.

Sua primeira versão foi patenteada em 1918, por Arthur Scherbius (1878-1929), e exibidas nos congressos da União Postal Universal em 1923 e 1924. Esta primeira versão foi chamada de *Enigma* *A*. Segundo WINTERBOTHAM (1978), a marinha alemã interessou-se pela *Enigma* e comprou alguns exemplares, adaptando-as ao seu uso em 1926. Estas primeiras máquinas de uso militar denominavam-se *Funkschlüssel C*. Porém, só em 1928 o exército elaborou a sua própria versão, a *Enigma G*. A partir de então, o seu uso estendeu-se a toda a organização militar alemã e a uma grande parte da hierarquia nazista.

Em meados 1938, Alan Turing foi convocado a se juntar ao GC&CS, em Bletchley Park, conhecido na época como o centro de decodificação de mensagens da inteligência britânica, para efetuar a criptoanálise da Enigma nazista. A máquina tinha por rotina mudar os padrões de códigos e criptografia diariamente, obrigando que o desafio de decifração fosse o mais rápido possível. Em 1940, Turing projetou a *"Bombe"*, uma máquina eletromecânica que ajudaria a decriptar as mensagens da Enigma.

A partir deste momento, a quebra desses conceitos e da suposta visão de “máquina indecifrável”, desvalorizaria a *Enigma* e seria responsável pelo fim prematuro da Segunda Guerra Mundial. WINTERBOTHAM (1978 comenta que o código da E*nigma,* foi de fato quebrado em 1933, por matemáticos da Polônia com a ajuda de meios *“bombas eletromecânicas”*. No entanto, versões aperfeiçoadas, de tais *“bombas eletromecânicas”,* pelos britânicos em Bletchley Park, sob a liderança do matemático Alan Turing, que aceleraram o processo de decodificação das *Enigmas* usadas pela marinha alemã.

## **Autômatos Finitos**

As máquinas de Turing são uma categoria de autômatos finitos caracterizada pela linguagem de recursivamente enumerável, do qual se aplica a resposta ao “Problema de Decisão” de David Hilbert. Izabela Melo, do Departamento de Sistemas & Computação da UFCG nos traz uma definição mais abstrata e “humana”, do que vem a ser um autômato finito:

*“Você já parou para pensar em como funcionam as portas que abrem e fecham automaticamente? E as lavadoras de louça/roupa, termômetros eletrônicos, relógios digitais, calculadoras e máquinas de venda automática? Todos esses dispositivos eletromecânicos têm uma controladora que nada mais é do que um autômato finito.*

*Além desses dispositivos (controladores), os autômatos finitos também são importantes para reconhecer padrões em dados, como, por exemplo, na análise léxica de um compilador; projetar uma nova linguagem para uma aplicação específica (desenvolvendo seu compilador); processamento de voz; etc.” (MELO, 2011)*

Os autômatos finitos são modelos simples computacionais, usados em máquinas com memórias extremamente limitadas, podendo ser exemplificada com uma porta automática de um supermercado: Geralmente, existem dois tapetes, um à frente e outro atrás da porta e neles estão alinhados os eixos do sensor de presença. Neste exemplo, o autômato funciona de forma bem simples: Quando alguém se enquadra em um dos tapetes a porta se abre e só se fecha quando a pessoa complete a passagem pelos dois tapetes, evitando assim, que a porta se feche atingindo o usuário.

Em uma linguagem mais computacional:

“O controlador pode estar em dois estados (aberto ou fechado) e passa de um estado para outro dependendo do estímulo (entrada) que recebe. Nesse caso, tem-se 4 estímulos diferentes: frente (uma pessoa está no tapete da frente); retaguarda (uma pessoa está no tapete de dentro); ambos (há pessoas sobre os dois tapetes) e nenhum (não há ninguém sobre os tapetes). Para cada estímulo, o controlador vai responder de uma forma diferente, realizando as transições de um estado para outro. A representação das possíveis transições de acordo com cada estímulo, chamamos essa representação de diagrama de estados.” (MELO, 2011)

## **Javascript**

Falar sobre o Javascript

## **jQuery**

Falar sobre o jQuery

## **Bootstrap**

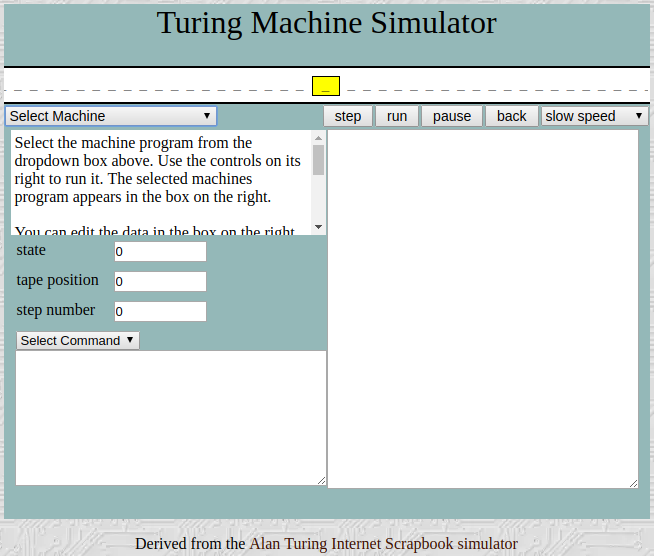
Falar sobre o Bootstrap

## **Projetos Relacionados**

### ***Turing Machine Simulator***

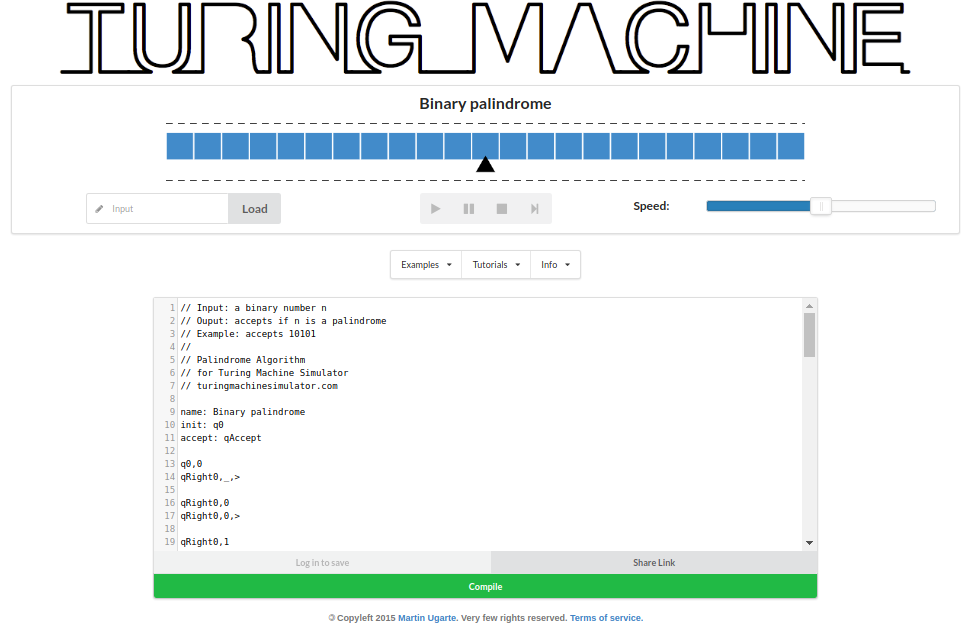
Falar melhor sobre o projeto de Paul Rendell

Figura 2 - Turing Machine Simulator, de Paul Rendell



### ***Turing Machine***

Figura 3 - Turing Machine, de Martin Ugarte



# **METODOLOGIA**

Antes de iniciar o desenvolvimento da aplicação, será feito um levantamento das possíveis funcionalidades e esboços do design e características visuais da aplicação. Todo o projeto estará seguindo o tradicional modelo de engenharia de software, ou mais conhecido como “Modelo Cascata”, criado por Royce em 1970, justo selecionado pela facilidade de implementação e organização do cronograma de atividades. (BOEHM, 1987) Então, após toda a parte de análise do projeto, serão desenvolvidos e implementados cada conceito e funcionalidade da máquina de Turing, tendo como aplicação final um simulador de máquina de Turing que auxiliará no ensino e na aprendizagem dos conceitos de Turing para a máquina.

Devido à uma melhor portabilidade do projeto em geral, falando mais apropriadamente do código fonte da aplicação, será utilizado o editor de código Visual Studio Code, recentemente lançado pela Microsoft e de alto índice de aceitação pela comunidade de desenvolvedores. O Visual Studio Code é de categoria *open-source,* disponibilizado para os principais sistemas operacionais Windows, Linux e Mac, e vem com um “leque” considerável de *plugins*, extensões e recursos para diversas linguagens de programação.

Para o desenvolvimento do projeto web foi utilizado como linguagem nativa o Javascript com plugins de jQuery para tratamento de elementos HTML, jQueryUI para tratamento gráfico do CSS3, HTML5 para elaboração das páginas web e pacote de temas e estilos do Bootstrap. Todas essas linguagens e ferramentas de linguagem de programação foram utilizadas em sua versão mais atualizada e estável, sendo importante notar que também são de categoria *open-source* como todo o projeto.

Seguindo para a parte final do projeto, “o teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado.” (DIAS, 2008). Portanto, serão realizados testes de funcionalidade e testes de usabilidade para garantir que o objetivo principal da aplicação seja alcançado.

# **DESENVOLVIMENTO**

## **Aplicação Web**

Pensando na usabilidade e em uma leitura mais amigável do que a aplicação, por objetivo, pretende oferecer, dividimos a mesma em quatro seções muito importantes: *Tutorial e Máquina de Turing,* no qual a *Figura 3* abaixo exemplifica.

Figura 4 - Tela Inicial



Fonte: Elaborada pelo autor

* **Tutorial:** *a* seção *Tutorial* tem como objetivo imergir o usuário em um tipo de estudo dirigido resumido, porém cuidadosamente detalhado, para que seja absorvido uma carga de conhecimentos e conceitos básicos sobre a máquina de Turing, antes mesmo de começar a usar o *Simulador* ou o *Construtor*. Neste estudo dirigido será possível aprender basicamente o que é uma máquina de Turing e como usar as seções *Simulador* e *Construtor,* através de slides e um vídeo, para abstrair tais conhecimentos. As figuras *Figura 4, Figura 5, Figura 6, Figura 7, Figura 8, Figura 9, Figura 10, Figura 11, Figura 12 e Figura 13*  mostram um pouco melhor do que é a realmente a seção.

Figura 5 - Tutorial – Slides 01



Fonte: Elaborada pelo autor

* **Máquina de Turing:**a seção *Máquina de Turing* é dividida em três subseções muito importantes para o entendimento dos conceitos da máquina de Turing.
  + **Acesso Rápido*:*** *a seção Acesso Rápido* tem como objetivo trazer um tipo de “teste rápido”, no caso de usuários que já aprenderam ou saibam usar a máquina, ou até porventura não dependem da seção *Construtor* para criar suas *“m-configurações”* (lógicas estados e transições) para máquina. Então, como podemos observar na *Figura 15 abaixo,* a seção é composta por basicamente:

Figura 6 - Acesso RápidoFonte: Elaborada pelo autor

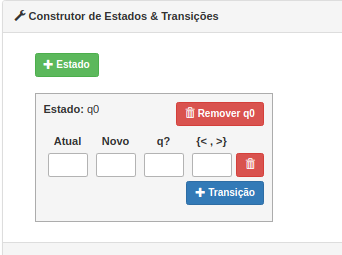
* + - **Exemplo:** um botão para baixar um arquivo de texto (.txt) com uma máquina de Turing programada. Se o conteúdo do arquivo de texto (.txt) for inserido na caixa de texto *“M-Configurações”* e em seguida clicarmos no botão *“Carregar M-Configurações”* a aplicação irá carregar as configurações na seção *Construtor* e montar o escopo de animação da máquina na seção *Simulador.*
    - **M-Configurações:** uma caixa de texto para inserir as m-configurações anteriormente programadas;
    - **Nova Máquina:** um botão para limpar todas as entradas do *Construtor* e iniciar a criação de uma nova máquina;
    - **Carregar Máquina:** um botão para carregar as configurações de máquina de Turing que já foram programadas;
    - **Salvar Máquina:** um botão para salvar as configurações de máquina de Turing que foram programadas pelo nas seções *Construtor* e *Simulador.*
  + **Construtor:** Na seção *Construtor*, a aplicação permite configurar uma máquina de Turing para uma determinada função a escolha do usuário. A princípio, a seção *Tutorial* já terá ensinado como programar uma máquina de Turing através da criação de estados da máquina e transições de estados na máquina, que segundo Turing são as *“configurações-m”.* O programa de máquina de Turing a ser ensinado inicialmente será algo bem simples como um “detector de sentenças equivalentes”, que também está disponibilizado no botão *“Exemplo Pronto”* da seção *Acesso Rápido*. Assim a seção *Construtor* é composta, como mostra a *Figura 16,* abaixo,pelos elementos:

Figura 7 - Construtor

Fonte: Elaborada pelo autor

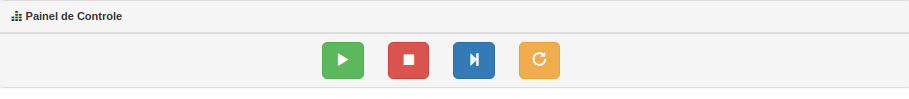
* + - **Adicionar Estado:** um botão para adicionar os estados da máquina. Ao adicionar um estado é criado um “*container”* de opções *(caixa de opções)* para configurar o estado;
    - **Remover Estado:** um botão para remover o estado e o correspondente *“container”* de opções;
    - **Adicionar Transição:** um botão para adicionar uma transição no *“container”* de opções do estado. Ao adicionar uma transição é inserida uma “linha” com caixas de texto à serem preenchidas para configurar a transição. Estas linhas são uma abstração simplificada das “*tuplas”* da máquina de Turing, que no caso serão compostas, de acordo com a *Figura 17,* por:

Figura 8 – Estados e Transições



Fonte: Elaborada pelo autor

* + - **Atual:** uma caixa de texto para inserir o valor atual na fita lido pelo leitor;
    - **Novo:** uma caixa de texto para inserir o novo valor que irá sobrescrever o valor atual lido da fita;
    - **q?:**  uma caixa de texto para inserir o próximo estado após a transição;
    - **{< , >}:** uma caixa de texto para ser inserido a direção em que o leitor da máquina deve seguir para ler o próximo valor da fita. A sintaxe deste campo foi definida da seguinte forma: “**<**”para esquerda e “**>**” para direita (sem aspas);
    - **Remover Transição:** um botão para remover a transição correspondente ao botão.
  + ***Simulador*:** Na seção do Simulador *haverá* um espaço para a inserção do estado inicial da máquina e da sequência de dados de entrada que configuram a fita da máquina. Por fim, após os atributos da máquina estiverem completamente definidos é apenas clicar no botão de iniciar para ver a máquina em funcionamento. O Simulador tem alguns controles de máquina e elementos de tela muito importantes, que podem ser observados na *Figura 18 abaixo*, eles são:

Figura 9 – Painel de Controle

Fonte: Elaborada pelo autor

* + - **Estado Inicial:** uma caixa de texto para inserir o estado inicial da máquina;
    - **Fita da Máquina:**  uma caixa de texto para inserir a sequência de valores (qualquer carácter antes definido nas transições da máquina);
    - **Iniciar:** um botão com a funcionalidade de dar início aos procedimentos e funcionamento da máquina;
    - **Parar:** um botão com a funcionalidade de parar todo o processo que estiver sendo executado pela máquina;
    - **Passo a Passo:** um botão com a funcionalidade de executar a máquina de modo que o usuário consiga visualizar literalmente “passo a passo” os processos de transição da máquina e movimentação do leitor na fita;
    - **Reiniciar:** um botão com a funcionalidade de reiniciar a máquina para o seu estado inicial anterior ao momento de sua inicialização.
    - **Máquina de Turing (Animação):** um espaço reservado na tela onde ocorrerão as animações correspondentes aos processos de transição e movimentação do leitor na fita. Podemos dividir a animação, de acordo com a *Figura 20,* em:

Figura 10 – Visualização dos Processos da Máquina

Fonte: Elaborada pelo autor

* + - * **Estado Atual:** espaço na tela onde é informado o estado atual em que a máquina está processando as transições;
      * **Posição do Leitor:** espaço na tela onde é informado a posição em que o leitor está na fita;
      * **Resultado:** espaço na tela onde é informado o status da máquina durante e no final da análise da fita;
      * **Leitor:** espaço onde é simulado a animação de um cabeçote (olho) de leitura percorrendo a fita;
      * **Fita da Máquina:** espaço onde é mostrado os valores da fita e as alterações feitas pelas transições da máquina.

## **Código-fonte**

Para um melhor entendimento técnico da aplicação esta seção tem como objetivo disponibilizar o código-fonte utilizado no desenvolvimento do projeto através de um link compartilhado no GitHub:

<https://github.com/wellmmeR/tm-web-simulator-unasp>.

# **ANÁLISE E RESULTADOS**

## **Dificuldades e Desafios**

Realizar testes com 3 pessoas no mínimo

## **Testes Realizados**

Realizar testes com 3 pessoas no mínimo

## **Resultados Obtidos**

Realizar testes com 3 pessoas no mínimo

# **CONCLUSÃO**

Melhorar conclusão

# **REFERÊNCIAS**

American Heritage Dictionary. AHDictionary of The English Language. Copyright 2016, Houghton Mifflin Harcourt. All rights reserved. Disponível em: [https://ahdictionary.com](https://ahdictionary.com/word/search.html?q=algorithm&submit.x=69&submit.y=13) - Acesso em 30 de agosto de 2016.

BOEHM, B. l. W. "Software Process Management: Lessons Learned from History" in ICSE '87 Proceedings of the 9th International Conference on Software Engineering, 1987.

BOOLOS, George S., BURGESS, John P., JEFFREY, Richard C. Computability and Logic. 5th Edition, 2007. Princeton University, New Jersey. ISBN: 9780521877527

DIAS N., A. C. Introdução a Teste de Software. Engenharia de Software Magazine. Nº 1, 2008.

DIVERIO, Tiaraju Asmuz & MENEZES, Paulo Blauth. Teoria da Computação: Máquinas Universais e Computabilidade, 2011. 3ª Edição, Volume 5. Editora Bookman.

HILBERT, David & ACKERMANN, [Wilhelm](https://en.wikipedia.org/wiki/Wilhelm_Ackermann" \o "Wilhelm Ackermann). Grundzüge der Theoretischen Logik (Princípios de Lógica Matemática). Springer-Verlag, [ISBN: 0-8218-2024-9](https://en.wikipedia.org/wiki/Special:BookSources/0821820249).

JACOBSON, I. & Pan-Wei, Ng. Aspect-Oriented Software Development with Use Cases (Addison-Wesley Object Technology Series), Publisher: Addison-Wesley Professional © 2004. ISBN: 0321268881.

LEAVITT, David. O homem que sabia demais. Tradução de Samuel Dirceu. 1ª edição. Ribeirão Preto, São Paulo. Editora Novo Conceito Ltda., 2007.

MELO, IZABELA. “Relembrando conceitos básicos sobre autômatos finitos”. 2011. Departamento de Sistemas & Computação da UFCG, Disponível em:<http://www.dsc.ufcg.edu.br/~pet/jornal/marco2011/materias/recapitulando.html> - Acesso em 21 de Novembro de 2016.

QUEIROZ, RUY J. G. B. de. Problemas Decidíveis e Problemas Indecidíveis: O Legado de Alan Turing, 2012. Disponível em: <http://www.ufrgs.br/alanturingbrasil2012/presentation-RuyQueiroz-ptBR.pdf> - Acesso em 21 de Novembro de 2016.

RENDELL, PAUL. Turing Machine Simulator, 2015. Disponível em: <http://rendell-attic.org/gol/TMapplet/index.htm> - Acesso em 24 de Novembro de 2016.

SETZER, V. W. “Alan Turing e a Ciência da Computação”. Depto. de Ciência da Computação da USP. Disponível em: https://www.ime.usp.br/~vwsetzer/Turing-teatro.html. Acesso em 30 de agosto de 2016.

TURING, A. M. “On Computable Numbers, with an Application to the Entscheidungsproblem”. Recebido em 28 de Maio de 1936 - Lido em 12 de Novembro de 1936. Proceedings of the London Mathematical Society.

UGARTE, MARTIN. Turing Machine. Disponível em: <https://turingmachinesimulator.com/> - Acesso em 24 de Novembro de 2016.

WINTERBOTHAM, F. W. “Enigma - O Segredo de Hitler”. Biliex, 1978.