UNIVERSIDADE FEDERAL DO PAMPA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

EDSON AZEVEDO DE CAMARGO

SAELF - SISTEMA DE APOIO AO ESTUDO EM LINGUAGENS FORMAIS

Bagé
2022

EDSON AZEVEDO DE CAMARGO

SAELF - SISTEMA DE APOIO AO ESTUDO EM LINGUAGENS FORMAIS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientadora: Sandra Dutra Piovesan

Bagé
2022

Ficha catalográfica elaborada automaticamente com os dados fornecidos pelo(a) autor(a) através do Módulo de Biblioteca do Sistema GURI (Gestão Unificada de Recursos Institucionais).

C172 Camargo, Edson Azevedo de

SAELF – SISTEMA DE APOIO AO ESTUDO EM LINGUAGENS FORMAIS / Edson Azevedo de Camargo. – 2022.

121 f.: il.

Orientadora: Sandra Dutra Piovesan

Trabalho de Conclusão de Curso (Graduação) – Universidade Federal do Pampa, Campus Bagé, Bacharelado em Engenharia de Computação, 2022.

1. Expressão Regular. 2. Linguagens. 3. Autômatos. 4. PHP. I. Sandra Dutra Piovesan. II. Título.

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
Universidade Federal do Pampa

EDSON AZEVEDO DE CAMARGO

SAELF - SISTEMA DE APOIO AO ESTUDO EM LINGUAGENS FORMAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 10 de Agosto de 2022.

Banca examinadora:

_____

Prof. Dr.a Sandra Dutra Piovesan
Orientadora - UNIPAMPA

_____

Prof. Dr. Fábio Luis Livi Ramos

UNIPAMPA

_____

Prof. Dr. Julio Saraçol Domingues Junior
UNIPAMPA

Assinado eletronicamente por **JULIO SARACOL DOMINGUES JUNIOR**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/08/2022, às 17:04, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **SANDRA DUTRA PIOVESAN**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/08/2022, às 17:04, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **FABIO LUIS LIVI RAMOS**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/08/2022, às 17:15, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0903546** e o código CRC **D7A56CB1**.

Referência: Processo nº 23100.017413/2022-14 SEI nº 0903546

Dedico este trabalho à minha mãe, por todo apoio e dedicação, posso afirmar com toda certeza que essa vitória é nossa!

AGRADECIMENTO

Agradeço a meus pais Solange e Nivaldo, padrasto Sr. Abner e familiares, vocês se sacrificaram, se dedicaram, me apoiaram de forma incondicional, abdicaram de tempo e de muitos projetos pessoais para que eu tivesse a oportunidade de estudar e de ter uma boa formação profissional, mas também pessoal. Eu devo tudo que sou a vocês, e se sinto orgulho de mim e do lugar onde cheguei, é porque sei que vocês vieram segurando a minha mão. Aos meus amigos, que na hora que precisei de alguma ajuda em relação ao meu trabalho, sempre estavam ali prontos a me ajudar, em especial ao meu saudoso amigo Dionatan Pinto de Souza (*in memoriam*) e Eugênio Pierazzoli, que estiveram sempre ao meu lado durante todo esse trabalho. E aos meus professores, por me ensinarem e ajudarem durante toda trajetória acadêmica, em especial a professora Dr.ª Sandra Dutra Piovesan por todo apoio e dedicação para que este trabalho fosse concluído da melhor forma possível. Eu dedico este título a vocês. Obrigado! Sem vocês, nada disso seria possível.

“Você nunca sabe que resultados virão da sua ação, mas se você não fizer nada, não existirão resultados.”

Mahatma Gandhi

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema online, implementado em PHP para que os acadêmicos do curso de Engenharia de Computação da Unipampa tenham a possibilidade de efetuar os exercícios da disciplina de Linguagens Formais. Dessa forma, auxiliando no aprendizado e facilitando a compreensão dos conteúdos abordados em sala de aula e com o propósito de tornar esta disciplina mais atraente aos olhos dos discentes do curso. O conteúdo da disciplina de linguagens formais é muito técnico e de difícil assimilação, desta forma o desenvolvimento do SAELF se mostra extremamente importante. Assim, o sistema conta com uma opção para realizar testes de expressões regulares, possibilidade de construção de autômatos finitos determinísticos e não-determinísticos e geração de palavras. O acadêmico poderá construir autômatos finitos determinísticos ou não-determinísticos, conseguindo efetuar o reconhecimento de palavras. No reconhecimento de palavras, têm-se duas opções: o reconhecimento direto, que indica somente se a palavra foi reconhecida ou não, e o reconhecimento passo a passo, que permite que o acadêmico visualize a análise de uma palavra a cada símbolo reconhecido. Durante o teste de expressões regulares, ele poderá inserir uma determinada expressão e realizar o reconhecimento ou não de uma palavra. Na opção geração de palavras o acadêmico deverá inserir os símbolos de um alfabeto qualquer e a quantidade de palavras que ele deseja listar. O sistema passou por uma validação com os alunos da disciplina de Linguagens Formais baseando-se em um questionário e a utilização do sistema, além disso, também foi avaliado pelo professor Esdras Lins Bispo Junior da Universidade Federal do Jataí (UFJ) em Goiás, onde foi emitido um parecer com a sua avaliação.

Palavras-chave: Expressão Regular; Linguagens; Autômatos; PHP.

ABSTRACT

This work presents the development of an online system, implemented in PHP so that the students of the Computer Engineering course at Unipampa have the possibility to carry out the exercises of the discipline of Formal Languages. In this way, assisting in learning and facilitating the understanding of the contents covered in the classroom and with the purpose of making this discipline more attractive in the eyes of the students of the course. The content of the discipline of formal languages is very technical and difficult to assimilate, in this way the development of the SAELF is extremely important. The system has an option to test regular expressions, the possibility of building deterministic and non-deterministic finite automata and generating words. The academic will be able to build deterministic or non-deterministic finite automata, being able to perform word recognition. In word recognition, there are two options: direct recognition, which indicates only whether the word was recognized or not, and step-by-step recognition, which allows the academic to visualize the analysis of a word for each recognized symbol. During the regular expression test, it will be able to enter a certain expression and perform the recognition or not of a word. In the word generation option, the student must insert the symbols of any alphabet and the number of words he wants to list. The system underwent validation with the students of the Formal Languages subject based on a questionnaire and the use of the system, in addition it was also evaluated by Professor Esdras Lins Bispo Junior of the Federal University of Jatai (UFJ) in Goiás, where it was issued an opinion with its assessment.

Keywords: Regular Expressions, Languages, Automata, PHP.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AFD	Autômato Finito Determinístico
AFND	Autômato Finito Não Determinístico
CSS3	Cascading Style Sheets
DSR	Design Science Research
ER	Expressão regular
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
JS	Java Script
MVC	Model View Controller
PHP	Hypertext Preprocessor
UML	Unified Modeling Language
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A Disciplina de Linguagens formais é uma disciplina importante do currículo de Engenharia de Computação na Universidade Federal do Pampa (UNIPAMPA), porém é uma disciplina difícil, por isso os discentes possuem dificuldade ao tentar absorver os conteúdos abordados em sala de aula e entender de forma plena. De acordo com Terra (2015) o índice de reprovação desta disciplina é de aproximadamente 50% e é necessário um conhecimento solido para o perfil do aluno na área de computação. Outro problema é o material de estudo de linguagens formais que é bastante complexo por ser puramente técnico, assim acaba por dificultar a sua compreensão pelos discentes do curso.

As disciplinas teóricas que utilizam formalismos matemáticos, demonstrações de teoremas são complexas e podem ocasionar o desinteresse por parte dos alunos. Em Linguagens formais não difere, essa disciplina contempla conteúdos extremamente abstratos, áridos, complexos e que aparentemente não parece fazer parte de sua futura profissão, dessa forma contribuindo na redução do interesse por parte do aluno (RAMOS, 2009). As Linguagens Formais e Autômatos finitos, estuda a teoria das classes de linguagens que possuem uma sintaxe precisa e sua semântica é bem definida e também analisa modelos computacionais que realizam o seu reconhecimento (VIEIRA, 2006).

Segundo Ramos (2009) a disciplina de Linguagens formais e autômatos finitos surgiu a partir da derivação de duas subáreas que eram independentes na década de 1960 a subárea de linguagens formais e a subárea de autômatos finitos.

Ainda em meados de 1950, Chomsky um matemático e linguista, iniciou os estudos de uma formalização gramatical de linguagens, porém não obteve muito sucesso nessa área. Outrora, ele apresentou uma classificação das linguagens estruturadas e organizadas em níveis de complexidade de forma crescente que até hoje é uma referência para os estudos de linguagens formais, mais tarde essa classificação ficou conhecida como a hierarquia de Chomsky (RAMOS, 2009).

A hierarquia de Chomsky foi proposta por Noam Chomsky em 1956 e submetese a uma classificação para linguagens, gramáticas e autômatos. Na hierarquia de Chomsky existe quatro partes, em que cada uma dessas partes é um subconjunto próprio da parte superior. Essas partes são denominadas: Linguagens Recursivamente Enumeráveis, Linguagens Sensíveis ao Contexto, Linguagens Livres de Contexto e Linguagens Regulares. Em computação, nas linguagens de programação e na implementação de compiladores são utilizados amplamente as partes de linguagens livres de contexto e

linguagens regulares (MENEZES, 2000).

Sabendo disso, neste trabalho de conclusão de curso foi implementado um sistema computacional online que favorece este aprendizado. Este sistema teve como objetivo principal apoiar os discentes do curso na disciplina de Linguagens Formais, sendo mais especificamente em Expressões Regulares e na representação gráfica de Autômatos de estados finitos.

O propósito que se pretende alcançar é tornar a disciplina de Linguagens Formais mais atraentes aos olhos dos discentes do Curso de Engenharia de Computação, facilitando assim o seu aprendizado por meio de exercícios em um sistema online onde o discente visualiza como os exercícios são resolvidos e possa interagir com o objeto de ensino, contribuindo assim no processo de aprendizagem.

O sistema possui exercícios a respeito de autômatos de estados finitos, onde o usuário poderá realizar estes exercícios e logo após, conferir se estão corretos ou verificar onde existiu o erro e no fim da tarefa enviar diretamente ao professor da disciplina para que este faça a devida avaliação. A existência de uma área para a realização de testes de Expressões Regulares também está disponível. Nela, o usuário poderá inserir uma expressão regular e testar se uma determinada palavra é reconhecida ou não pela sua expressão. Também possui um gerador de palavras para um determinado alfabeto Σ (*sigma*), onde o usuário insere os símbolos do alfabeto e a quantidade de palavras que pretende gerar; com isso o sistema gera a sequência de símbolos denominado como "palavra" a partir da menor sequência possível que seria a palavra vazia ϵ (*epsilon*) até o número máximo de símbolos solicitado

1.1 Problema de pesquisa

Em um cenário onde o índice de reprovação de uma disciplina na universidade é alto e o grau de dificuldade é elevado, é possível auxiliar os estudantes da disciplina de Linguagens formais, tornando-a mais atrativa, disponibilizando conteúdos e exercícios através de uma plataforma online?

1.2 Objetivos

Nesta sessão serão abordados o objetivo geral e os objetivos específicos deste trabalho.

1.2.1 Objetivo Geral

O objetivo geral do presente trabalho foi construir um sistema computacional denominado SAELF – Sistema de Apoio ao Estudo em Linguagens Formais, a fim de auxiliar o estudo de Linguagens Regulares e Autômatos Finitos no curso de Engenharia de Computação.

1.2.2 Objetivos Específicos

- Pesquisar, identificar, estudar e comparar os sistemas correlatos;
- Buscar tecnologias para o desenvolvimento do SAELF;
- Implementar uma solução adequada do sistema;
- Realizar os testes dos recursos disponibilizados e validar o sistema proposto.

1.3 Organização do trabalho

Este trabalho está organizado como se segue: no Capítulo 2 são apresentados os aspectos teóricos estudados para o desenvolvimento do trabalho. São abordados temas como Linguagens Formais, Expressões Regulares, Autômatos finitos determinísticos, não determinísticos. O Capítulo 4 apresenta os trabalhos correlatos. O Capítulo 5 descreve a metodologia e as ferramentas utilizadas para o desenvolvimento deste trabalho. Já o Capítulo 6 apresenta o desenvolvimento do SAELF – Sistema de Apoio aos Estudos de Linguagens Formais, detalhando os requisitos, especificação e implementação do sistema. No Capítulo 7 a análise dos dados do formulário aplicado. Por fim, o Capítulo 8 apresenta as considerações finais.

2 REFERENCIAL TEÓRICO

Esta seção é destinada à apresentação do referencial teórico que apoiou o trabalho técnico realizado. O subcapítulo 2.1 apresenta a revisão sistêmica da literatura, seguido do subcapítulo 2.2 apresenta os conceitos fundamentais deste trabalho; O subcapítulo 2.3 apresenta Linguagens Formais, Linguagens Regulares 2.4 e Expressões Regulares 2.5. O subcapítulo 2.6 apresenta os Autômatos finitos.

2.1 Revisão sistêmica da literatura

Um estudo que possui o objetivo reunir material semelhante de vários autores e realizar uma análise é denominado revisão sistêmica da literatura. Como este trabalho possui caráter exploratório de pesquisa bibliográfica e suas aplicações, dessa forma, esta revisão sistêmica tem por objetivo instrumentalizar e conceituar o processo de desenvolvimento.

Nesta etapa de análise e revisão bibliográfica a organização foi realizada da seguinte forma:

- Etapa 1 – definir as questões de pesquisa relacionadas à revisão;
- Etapa 2 – definir as fontes de pesquisa;
- Etapa 3 – definições das strings de busca;
- Etapa 4 – análise dos resultados encontrados;
- Etapa 5 – classificação dos trabalhos encontrados;
- Etapa 6 – revisão e análise dos trabalhos selecionados.

A partir da definição das etapas de desenvolvimento da revisão sistemática a seguir serão descritos os métodos e resultados destas etapas.

Etapa 1 – Definição das questões de pesquisa:

I. Quais são as ferramentas existentes para o ensino de linguagens formais e autômatos finitos?

II. Quais são as áreas de linguagens formais que estas ferramentas abrangem?

III. Existem ferramentas que abrangem todas as áreas de linguagens formais como, por exemplo, expressões regulares, autômatos finitos determinísticos e não determinísticos?

Na Tabela 1 estão elencadas as fontes de pesquisas selecionadas na etapa 2 da

revisão sistêmica da literatura.

Tabela 1 – Fontes de pesquisa na revisão sistêmica

Tipo	Fonte
Bibliotecas e repositórios digitais	ACM Digital Library <https://dl.acm.org/>
	IEEE Xplorer <https://ieeexplore.ieee.org/>
	Wiley Online Library <https://onlinelibrary.wiley.com/>
Periódicos	ACM Transactions on Computing Education <https://dl.acm.org/journal/toce>
Anais de eventos	International Computing Education Research Workshop <https://dl.acm.org/conference/icer>
	Information Technology Education Conference <https://dl.acm.org/conference/ite>
	Innovation and Technology in Computer Science Education <https://dl.acm.org/conference/iticse>
	SIGCSE: Computer Science Education <https://dl.acm.org/conference/sigcse>
	Workshop sobre Educação em Computação <https://sol.sbc.org.br/index.php/wei>

Fonte: Próprio autor

Na etapa 3 foram definidas as strings de busca baseadas em palavras-chave com o apoio do referencial teórico utilizado neste trabalho e a seguir criada a Tabela 2 para uma melhor visualização destas *strings*.

Tabela 2 – Tabela de strings de busca

Termo	Sinônimos
Autômatos finitos	Máquina, finite automata
Ensino	Aprendizado, método, estratégia, learning, teaching,
Exp. regular	regular expression
Linguagens	linguagem, sistema, software, tool, language, system

Fonte: Próprio autor

As strings de busca nas fontes de pesquisa foram construídas por meio da conjunção de cláusulas que contenham a disjunção dos sinônimos em seu interior, estão sendo ilustradas na Tabela 3.

Tabela 3 – Strings de busca utilizadas na pesquisa

String	String e operador
1	Autômatos finitos OR finite automata OR Expressão regular OR Regular expression AND software AND Máquina AND Linguagem OR language OR Method OR system
2	Autômatos finitos OR finite automata AND Expressão regular OR Regular expression AND software AND Máquina NOT Linguagem OR language NOT Method OR system
3	Autômatos finitos OR finite automata AND Expressão regular OR Regular expression AND software AND Máquina

Fonte: Próprio autor

A primeira fonte de pesquisa utilizada foi a biblioteca ACM Digital Library, a string de busca utilizada foi Autômatos finitos OR finite automata OR Expressão regular OR Regular expression AND software AND Máquina AND Linguagem OR language OR Method OR system buscando referenciais por título e resumo. A partir dessa string foram obtidos 164.949 resultados referentes a artigos publicados nos últimos cinco anos.

Foi necessário realizar uma redução no demasiado número de resultados da busca, assim, foram analisados os títulos dos artigos que traziam assuntos não relacionados ao trabalho e a *string* foi alterada após análise dos resultados. Retiramos a palavra **linguagem** da busca por trazer os mais diversos resultados de trabalho que não estavam

correspondendo a nossa pesquisa, para isso utilizamos o operador ***NOT***, porém essa alteração não surtiu efeito, então retiramos da busca as palavras **linguagem**, ***method*** e ***system***, ainda para refinarmos melhor a busca inserimos a palavra ***software*** após cada um dos termos.

Ao final dessas alterações a *string* de busca ficou dessa forma: **Autômatos finitos OR *finite* automata AND Expressão regular OR *Regular expression* AND software AND *NOT* Máquina**, nesse momento reduzimos a busca para apenas 14 trabalhos onde foi possível realizar a leitura de seus resumos e introduções, a fim de se obter uma pesquisa eficiente.

É importante ressaltar que essa revisão sistemática está sendo realizada no período de junho de 2017 á junho de 2022, ou seja, nos últimos 5 anos e que desses 14 trabalhos encontrados com a *string* de busca nenhum correspondeu exatamente a busca, encontramos trabalhos relacionados a autômatos finitos ou expressões regulares, mas nenhum que se auxilia ao ensino de ambos os temas.

Na pesquisa da ferramenta *Google Scholar* as mesmas strings foram utilizadas anteriormente na busca por título da biblioteca *ACM Digital Library*, com filtro para o período de 2016 a 2021. Obteve-se aproximadamente 100 resultados. Foram realizados alguns testes com strings mais curtas e foi possível perceber que os resultados melhoraram, mas nenhum resultado exatamente como desejamos foi encontrado. Posto que a pesquisa não trouxe resultados satisfatórios, selecionamos dois artigos da plataforma Google Scholar que se mostraram interessantes para o auxílio ao ensino de linguagens formais.

O primeiro trabalho selecionado trata-se de uma ferramenta para auxiliar o ensino de expressões regulares. Esta ferramenta dá a possibilidade do professor gerenciar o cadastro de exercícios e atividades para uma determinada turma e também é possível realizar o acompanhamento em tempo real. Na implementação foi utilizado *Websockets* e a biblioteca *React* para desenvolvimento do *frontend*, também foram desenvolvidos sistemas de notificação em tempo real, além disso, a ferramenta conta com a utilização do banco de dados NoSQL MongoDB (KATAKURA, 2018).

O segundo trabalho refere-se a um jogo baseado em autômatos finitos no estilo *escape room* onde os jogadores precisam achar um jeito de encontrar a saída de cada ambiente. Os autores utilizaram para o desenvolvimento a *engine Unity3D*, e também utilizaram os recursos gráficos disponibilizados pela própria *Unity*, além disso, por meio dela ainda é possível escolher a plataforma de destino do jogo, ou seja, permite que o

jogo seja executado em multiplataformas. Durante o desenvolvimento foram criadas um sistema de pontuação no jogo, condições de vitória e derrota e desafios para o jogador (CARVALHO; JUNIOR; COSTA, 2021).

2.2 Conceitos fundamentais

A fim de facilitar a compreensão e simplificar a leitura, especificamos aqui alguns conceitos fundamentais a respeito de linguagens formais. É denominado um Alfabeto um conjunto finito de símbolos, e é normalmente adotado como símbolo o Σ, portanto, um conjunto infinito não é um alfabeto, por outro lado, o conjunto vazio é um alfabeto e seu símbolo é ϕ (*phi*). Uma palavra sobre um alfabeto Σ é uma sequência finita de símbolos e denotada por ω (*omega*) já o conjunto de todas as palavras de um alfabeto Σ é denotado por Σ^* e para denotarmos uma palavra vazia, ou seja, sem nenhum símbolo, utilizaremos ε (MENEZES, 2000).

Em uma palavra sobre um alfabeto ε podemos ter Prefixo, Sufixo e Subpalavra. Prefixo é qualquer sequência inicial de uma palavra, sufixo é qualquer sequência final e subpalavra é qualquer sequência de símbolos contíguos de uma palavra, sendo assim qualquer prefixo ou sufixo é uma subpalavra (MENEZES, 2000).

Exemplo:
Σ = {a, b} e ω = abbaa.
Prefixos: ε, a, ab, abb, abba, abbaa.
Sufixos: ε, a, aa, baa, bbaa, abbaa.

Concatenação de palavras é uma operação binária definida sobre o produto cartesiano de dois conjuntos, onde a primeira palavra é associada com a segunda exatamente nesta ordem.

Exemplo:
Σ = {a, b}, ω_1 = {ab} e ω_2 = {ba}, logo a concatenação será:
$\omega_1.\omega_2$ = {abba}

Podemos também concatenar uma palavra com a palavra vazia ε.

$\omega_1.\varepsilon$ = {ab}

Concatenação sucessiva ou Fecho de Kleene é uma operação unária e infinita, porém, contável. O fecho de Kleene de *L* é denotado por L^* onde o * significa que o "*L*" pode se repetir zero ou mais vezes. Definição fecho de Kleene: Seja Σ um alfabeto e L $\subseteq \Sigma^*$, logo $L^* = \{\omega_1, \omega_2, \cdots \omega_n | \omega_1 \in L, 1 \leq i \leq n, n \geq 0\}$ (FURTADO, 2007).

Exemplo: Σ = {a, b, c}*

L^* = { ε, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab $\cdots$ }

Uma linguagem é um conjunto de palavras sobre Σ e denotada por L. Definição concatenação de linguagens: Seja Σ um alfabeto e L1, L2 $\subseteq \Sigma^*$ a concatenação de L1 e L2 é denotada por L1.L2 = $\{\omega_1\omega_2, | \omega_1 \in L1 \wedge \omega_2 \in L2 \}$ (FURTADO, 2007).

Exemplo:

L1 = {ε, a, b}

L2 = {0, 1}

L1.L2 = $\{0, 1, a_0, a_1, b_0, b_1\}$

Propriedades das Linguagens:

Segundo Furtado (2007) a classe das Linguagens Regulares (LR) é fechada para as operações de União, Concatenação, Complemento e Intersecção, abaixo na Tabela 4 veremos suas propriedades:

Tabela 4 – Propriedades das operações

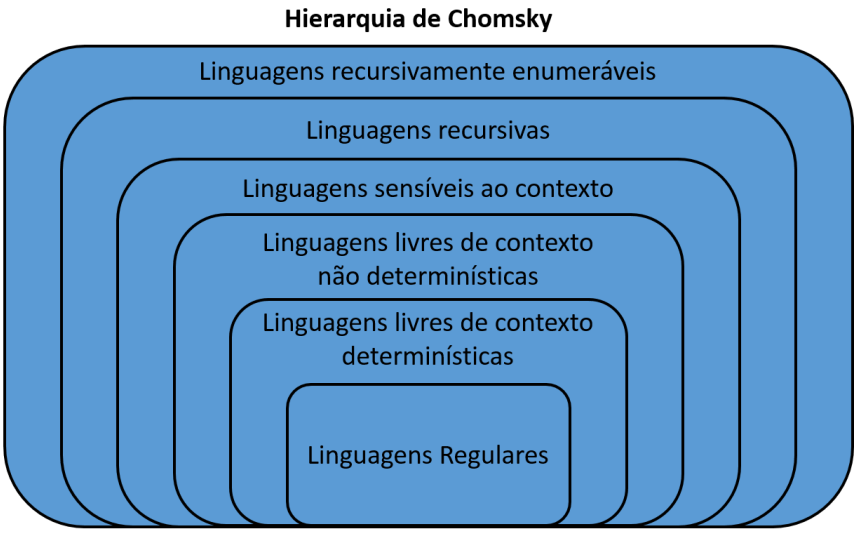
Operação	Propriedade
União	Seja L1 e L2 (LR), então L1 ∪ L2 = $\{\omega \mid \omega \in L1 \vee \omega \in L2\}$ também é uma LR
Concatenação	Seja L1 e L2 (LR), então L1.L2 = $\{\omega\beta | \omega \in L1 \wedge \beta \in L2\}$ também é uma LR
Complemento	Se L1 $\subseteq \Sigma^*$ é um LR, então $\Sigma^* -$ L1 também é uma LR
Intersecção	Seja L1 e L2 (LR), então L1 ∩ L2 = $\{\omega | \omega \in L1 \wedge \omega \in L2\}$ também é uma LR

Fonte: Adaptado de Furtado (2007)

2.3 Linguagens Formais

Em 1956 Noam Chomsky descreveu o que hoje é conhecido como a Hierarquia de Chomsky. A classificação das Linguagens Formais é definida nesta hierarquia pelo grau de complexidade de suas regras de reescrita, a Figura 1 demonstra esta hierarquia. Assim as linguagens são classificadas conforme o tipo variando de 0 a 3. As de nível 0 não possuem restrições, as outras são cada vez mais restritas respectivamente e denominadas como sensíveis ao contexto, livres de contexto e regulares (REINALDO et al., 2003).

Figura 1 – Hierarquia de Chomsky



Fonte: Adaptado de Menezes (2000)

Uma linguagem sobre um alfabeto Σ, é um conjunto de palavras sobre Σ, ou seja:

$$L \subseteq \Sigma^*$$

2.3.1 Linguagem do tipo 0

São denominadas enumeráveis recursivamente ou irrestritas, nela não existem restrições quanto às regras de reescrita ou produção, exceto que as regras de produção estejam na forma: $\alpha \rightarrow \beta$, sendo que $\alpha \neq \varepsilon$. As linguagens geradas por essa classe de gramática são reconhecidas por uma máquina de Turing com fita ilimitada. (REINALDO et al., 2003; SANTOS; COELHO, 2010)

2.3.2 Linguagem do tipo 1

São denominadas de sensíveis ao contexto, nessa classe é exigido apenas que em cada regra de produção, o comprimento da palavra original seja igual ou menor que a palavra derivada.

Ac $\rightarrow$ Aab é válida.

Aab $\rightarrow$ "Ac" é inválida, pois a palavra original "Aab" é maior que à palavra derivada.

As linguagens geradas por essa classe de gramática são reconhecidas por uma máquina de Turing com fita limitada. (REINALDO et al., 2003; SANTOS; COELHO, 2010).

2.3.3 Linguagem do tipo 2

São denominadas livres de contexto, neste tipo de linguagem todos os antecedentes das regras têm apenas um elemento, por exemplo, A $\rightarrow$ Ba, sendo que o resultado da transformação de A independente do contexto.

As linguagens geradas por essa classe de gramática são reconhecidas por um autômato com pilha. (REINALDO et al., 2003; SANTOS; COELHO, 2010).

2.3.4 Linguagem do tipo 3

São denominadas de Linguagem Regular e é aquela em que as regras de produção, tem apenas como consequência um terminal ou um terminal seguido de um não terminal e estão na forma A $\rightarrow$ Bw ou A $\rightarrow$ wB.

As linguagens geradas por essa gramática são reconhecidas por um autômato de estados finitos. (REINALDO et al., 2003; SANTOS; COELHO, 2010).

A fim de definir um tipo de linguagem podemos utilizar um dispositivo chamado de reconhecedor que nos permite submeter uma palavra ou uma cadeia de símbolos a um teste de aceitação, onde ele consegue determinar se tal palavra é ou não reconhecida pela linguagem. Um dispositivo reconhecedor é, na verdade, um modelo matemático que

descreve o funcionamento de uma máquina (SILVA, 2007).

A Tabela 5 deixa claro o tipo de linguagem reconhecido por cada reconhecedor.

Tabela 5 – Tipos de reconhecedores de linguagem

Linguagem Tipo	Linguagem	Reconhecedor
3	Regulares (LR)	Autômatos finitos
2	Livres de Contexto (LLC)	Autômatos de pilha
1	Sensíveis ao Contexto (LSC)	Máquina Turing
0	Recursivamente Enumerável (LRE)	Máquina Turing

Fonte: Apostila Teoria de Computação – UTFPR

Segundo Furtado (2007), podemos classificar as Linguagens Formais como um estudo de modelos matemáticos que viabilizam o reconhecimento de linguagens, suas propriedades, classificações e estruturas. A representação de uma Linguagem formal e realizada de forma finita e exata, utilizando sistemas baseados em modelos matemáticos. Todos os problemas computacionais podem ser tratados ou estudados sob a ótica da teoria das linguagens formais.

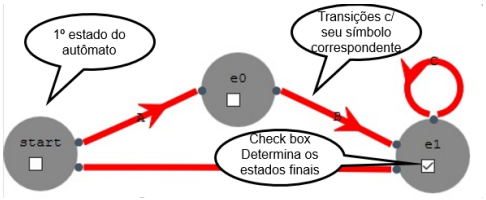
2.4 Linguagens Regulares

Segundo a Hierarquia de Chomsky, a Linguagem Regular trata-se de uma classe de linguagens um pouco mais simplificada, sendo possível desenvolver algoritmos de reconhecimento ou de geração com um nível baixo de complexidade, possuindo alta eficiência e de fácil implementação (MENEZES, 2000).

As linguagens regulares são aquelas que podem ser representadas por uma expressão regular (ER) ou por um autômato finito.

Na Figura 2 é demonstrada uma legenda explicando as partes principais dos autômatos desenvolvidos no SAELF.

Figura 2 – Partes principais do autômato



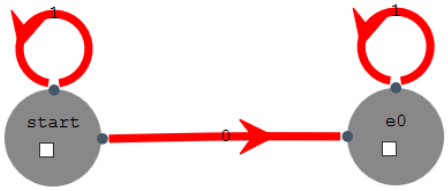
Fonte: Próprio autor

A seguir, mostraremos alguns exemplos de ER (Expressão regular) e seus autômatos finitos correspondentes, para facilitar a compreensão.

Exemplos:

1. L = {$\omega \in \{0,1\}^*$ | ω possui apenas um 0}
   ER = $1^* 0 1^*$
   L = {0, 01, 10,011,101, 110, 0111, 1011, 1101, 1110 $\cdots$ }

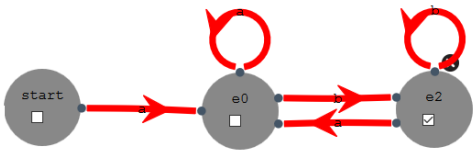
Figura 3 – Autômato correspondente ao exemplo 1.



Fonte: Próprio autor

2. L = {$\omega \in \{a,b\}^*$ | ω começa com “a” e termina com “b”}
   ER = a (a | b)* b
   L = {ab, aab, abb, aaab, aabb, abab, abbb, aaaab, aaabb, aabab $\cdots$ }

Figura 4 – Autômato correspondente ao exemplo 2

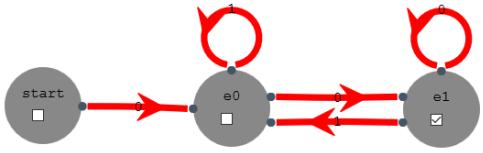


Fonte: Próprio autor

3. L = {$\omega \in \{0,1\}^*$ | ω começa e termina com 0}
   ER = (0 (0 | 1)* 0) | 0

L = {0, 00, 000, 010, 0000, 0100, 0110, 0010, 00000, 01000 ⋯ }

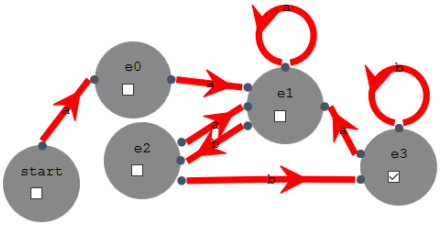
Figura 5 – Autômato correspondente ao exemplo 3



Fonte: Próprio autor

4. L = {$\omega \in \{0,1\}^*$ | ω começa e termina com bb}
   ER = aa (a | b)* bb
   L = {aabb, aaabb, aabbb, aaaabb, aaabbb, aababb, aabbbb, aaaaabb }

Figura 6 – Autômato correspondente ao exemplo 4



Fonte: Próprio autor

Uma linguagem L é dita regular se, e somente se, é possível construir um Autômato Finito (Determinístico ou Não-Determinístico) ou uma expressão Regular que reconheça a linguagem (OLIVEIRA, 2009).

2.4.1 Teorema Linguagem regular

Se L é uma linguagem regular, então existe uma expressão regular “r” tal que Gera (r) é uma linguagem regular (PALAZZO, 2007).

2.5 Expressões Regulares

Menezes (2000) diz que para determinarmos uma linguagem regular podemos desenhar um diagrama de estados em forma de grafo, realizar o processo de concepção de um reconhecedor de linguagem ou escrever sua expressão regular. Definição: Uma expressão regular (ER) sobre um alfabeto Σ é indutivamente definida como se segue:

1. ϕ é uma expressão regular que denota uma linguagem vazia.
2. ε é uma expressão regular que denota uma linguagem contendo exclusivamente a palavra vazia $\{\varepsilon\}$.
3. Qualquer símbolo X pertencente ao alfabeto Σ é uma expressão regular.
4. Se "r" e "s" são expressões regulares e denotam respectivamente as linguagens L = (r) e L = (s), respectivamente, então:
   - (r + s) é uma expressão regular e denotam a linguagem R ∪ S.
   - (rs) é uma expressão regular e denotam a linguagem $\{\alpha\beta \mid \alpha \in R \wedge \beta \in S\}$.
   - (r^*) é uma expressão regular que denota a linguagem R^* (MENEZES, 2000).

A Tabela 6 exemplifica algumas expressões regulares e demonstra os tipos de palavra que são reconhecidas.

Tabela 6 – Exemplos de Expressões Regulares para Σ = {a,b}

Exp. Regular	**Linguagens Representadas**
ab	Somente a palavra ab
ba^*	Todas as palavras iniciam com b, seguido de zero ou mais a_s
$(a \mid b)^*$	Todas as palavras sobre o alfabeto {a,b} incluindo a palavra ε
$(a \mid b)^*aab(a \mid b)^*$	Todas as palavras contendo aab como sub palavra.
$a^*ba^*ba^*$	Todas as palavras contendo exatamente dois b
$(a \mid b)^*(aa \mid bb)$	Todas as palavras que terminam com aa ou bb
$(a \mid \varepsilon)(b \mid ba)^*$	Todas as palavras que não possuem dois a consecutivos

Fonte: Palazzo (2007)

Um exemplo prático de utilização de expressões regulares, seria como verificar a estrutura de um número de CPF. Na Figura 7 que está ilustrando o formato de uma expressão regular para validar um número de CPF, temos que o trecho [0-9] aceita apenas número de zero até nove, o trecho 3 garante que apenas três números sejam digitados, o trecho [/.] garante que a cada três números um ponto tenha que ser digitado. Portanto, através dessa expressão regular é possível garantir que apenas números no formato de um

CPF sejam digitados em um formulário.

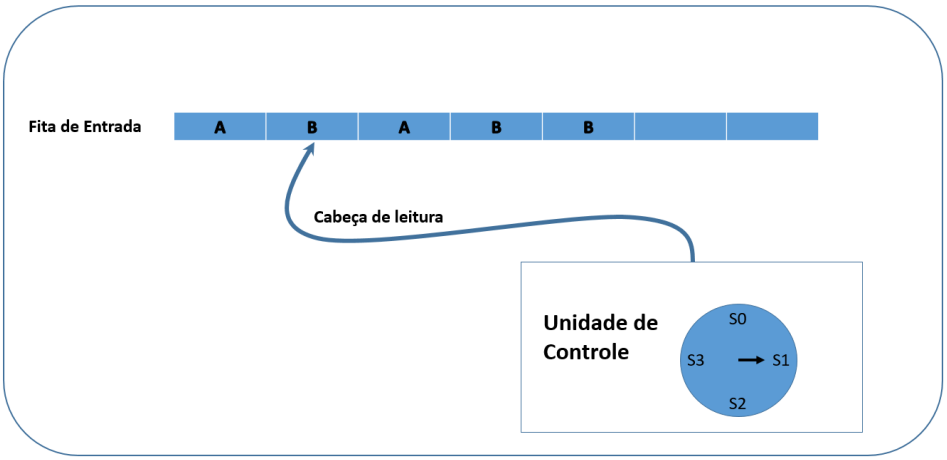
Figura 7 – Expressão regular para validação de um CPF

Fonte: Próprio autor

2.6 Autômatos Finitos

Os autômatos finitos são basicamente reconhecedores de linguagens regulares. Um reconhecedor de uma linguagem L qualquer é um dispositivo que, recebendo uma sequência ω como entrada, confirmam o reconhecimento de ω se e somente se, $\omega \in \varepsilon$ L ou se a sentença não for reconhecida $\omega \notin$ L (FURTADO, 2007). Um Autômato Finito pode ser dividido em três partes: fita de entrada, unidade de controle e cabeça de leitura como exemplificado no diagrama da Figura 8.

Figura 8 – Diagrama de representação Autômato



Fonte: Próprio autor

Os símbolos que compõe a palavra a ser testada encontram-se escritos em uma fita de entrada, existe uma unidade de controle que indica qual é o estado atual e existe uma cabeça de leitura que lê um símbolo por vez da fita. Um Autômato é dito finito porque o conjunto de estados possíveis é finito.

Neste trabalho abordaremos dois tipos de Autômatos Finitos: Autômato Finito Determinístico (A.F.D.) e Autômato Finito Não Determinístico (A.F.N.D.).

2.6.1 Autômatos Finitos Determinísticos

Segundo Furtado (2007), formalmente definimos um A.F.D. como:

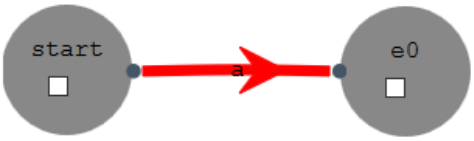
$$M = (K , \Sigma, \delta, q_0, F)$$

Onde:

- K é um conjunto finito, não vazio de estados;
- Σ é um Alfabeto, finito, de entrada;
- δ é uma função de transição definida em: K x $\Sigma \rightarrow$ K;
- $q_0 \in$ K, é o estado inicial;
- F $\subseteq$ K, é o conjunto de estados finais;

A interpretação de uma transição δ (*start*, a) = e_0 , onde *start* $\wedge$ $e_0 \in$ K $\wedge$ a $\in \Sigma$, é a seguinte: se o "Controle de M" está no estado "start" e o próximo símbolo de entrada é "a", então "a" deve ser reconhecido e o controle passar para o estado "e_0" na Figura 9 exemplificamos esta interpretação.

Figura 9 – Interpretação de δ (função de transição)



Fonte: Próprio autor

Um autômato é dito determinístico quando o estado atual de M, ao ler um determinado símbolo na fita de entrada, existe apenas um próximo estado possível.

Estado inicial: O estado inicial q_0 indica que ao executarmos o AFD, a unidade de controle automaticamente se posiciona no estado inicial, ou seja, q_0. Deve existir no máximo um estado inicial em cada autômato.

Reconhecimento: Uma palavra é reconhecida, se e somente se, após a leitura de todos os símbolos contidos na fita de entrada, o marcador se encontrar parado em um estado final, caso contrário a palavra não foi reconhecida.

Exemplo de autômato finito determinístico:

Seja M = (K , Σ, δ, q_0, F) tal que Σ = {a,b}, K = {e_0, e_1, e_2}, F = {e_2} e a função transição δ definida segundo a Tabela 7.

Tabela 7 – Tabela função de transição δ

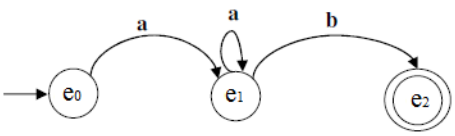
δ	**a**	**b**
e_0	e_1	-
e_1	e_1	e_2
e_2	-	-

Fonte: Próprio autor

Ao interpretarmos a Tabela 7, partirmos do estado inicial e_0 e lendo o símbolo "a" na fita de entrada, M muda para o estado e_1. Se o marcador estiver em e_0 e a fita de entrada com o símbolo "b" a palavra é rejeitada automaticamente por M, pois não existe transição definida para esta situação. Já estando no estado e_1, e lendo o símbolo "a", M permanece no estado e_1. Quando M está no estado e_1 e lê um símbolo "b", ele muda para o estado e_2. Estando no estado e_2 qualquer símbolo que seja lido, a palavra será rejeitada, pois não existem transições definidas a partir de e_2.

Comumente a representação de δ é realizada utilizando um grafo orientado. A Figura 10 apresenta o grafo correspondente a função de transição do autômato finito determinístico utilizado no exemplo anterior (SILVA, 2007).

Figura 10 – Representação AFD



Fonte: Silva (2007)

Os estados possíveis do AFD estão representados na Figura 10 como e_0, e_1 e e_2, sendo e_0 o estado inicial indicado por uma seta sem origem e com destino o próprio estado e e_2 o estado final indicado por dois círculos concêntricos, enquanto as transições estão representadas pelas arestas. A origem de uma seta indica o estado atual e ela aponta para

qual estado ela vai caso o símbolo lido seja o correspondente. O símbolo a ser lido pela transição é colocado sobre a aresta.

O destino da aresta indica o qual será o próximo estado após ser feita a leitura do símbolo.

O estado e_1 pode ser traduzido como: “o estado onde se lê um ou mais $\{a_s\}$ ”.

O estado e_2 pode ser expresso por: “o estado que indica a leitura de exatamente um $\{b\}$ ”.

No exemplo da Figura 10 o AFD reconhece a linguagem L = $\{\omega \mid \omega$ é formada por uma sequência de um ou mais “as”, seguido de exatamente um b sobre $\Sigma^*\}$, onde $\Sigma = \{a,b\}$. Portanto, toda linguagem (LR) possui um AFD correspondente (SILVA, 2007).

2.6.2 Autômatos Finitos Não Determinísticos

Um autômato finito não determinístico (AFND) é similar a um AFD, porém existe pelo menos um estado tal que ao ler um mesmo símbolo há mais de uma possibilidade de destino.

Um AFND é um sistema formal

$$M = (K , \Sigma, \delta, e_0, F)$$

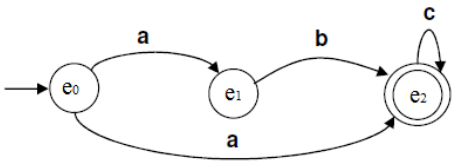
Onde:

- K, Σ e_0, F possuem a mesma definição de um AFD.
- δ É uma função de transição, definida em K x Σ = ρ(K); sendo que ρ(K) é um subconjunto de K; isto equivale a dizer que $\delta(q, a) = p_1, p_2, ...,p_n$.

A interpretação de δ é que M no estado “q”, com o símbolo “a” na entrada irá testar os dois caminhos disponíveis e escolher entre o estado p1, estado p_2, ..., como para o estado p_n (MENEZES, 2000).

Na Figura 11 ilustra de forma sucinta como funciona um AFND.

Figura 11 – Representação AFND



Fonte: Menezes (2000)

Na Figura 11, existem duas transições de estados possíveis para o símbolo “a” estando o autômato em seu estado inicial.

Aceitação. Uma palavra é reconhecida por um AFND se ao testarmos todas as transições possíveis à medida que se lê a palavra, o AFND para em um estado final após ler toda a palavra. Sendo assim, o não-determinismo do próximo estado pode ser interpretado como um teste de todas as possibilidades. Alguns exemplos de palavras reconhecidas pelo autômato da Figura 11: {a, ac, ab, abcc, ...} (SILVA, 2007).

Rejeição. Uma palavra é rejeitada por um AFND se nenhum caminho de transições leva o autômato a um estado final após ler toda a palavra. Alguns exemplos de palavras rejeitadas pelo autômato da Figura 11: {b, bc, abb, aca, ...} (SILVA, 2007).

Teorema: Para todo AFND é possível construir um AFD equivalente.

Na prática, AFND e AFD são equivalentes, pois a facilidade de não Determinismo dos AFND, não representa um aumento de poder computacional em relação aos AFD, ou seja, a classe de linguagens reconhecidas por AFNDs e AFDs é a mesma (MENEZES, 2000).

2.7 Modelo Espiral

Na década de 80 Barry Boehm criou o modelo de desenvolvimento de software que, na verdade é uma melhoria do modelo incremental e possui o princípio de ser iterativo e direcionado a riscos, porque a cada iteração é realizada uma análise dos riscos (BOEHM, 1988). Esse nome se dá por sua representação, aonde cada volta no espiral passa por todas as fases do desenvolvimento do software. No modelo em espiral as voltas podem ser repetidas quantas vezes sejam necessárias, até que o software possa ser finalizado. O modelo em espiral é dividido em quatro etapas:

2.7.1 Primeira etapa – Definição dos objetivos

Nesta etapa, serão definidos os objetivos específicos para depois identificar as limitações do processo e do software, além de ser quando o planejamento da gerência é detalhado e os riscos são identificados.

2.7.2 Segunda etapa – Avaliação e redução de riscos

Nesta etapa, é realizada uma análise detalhada para cada um dos riscos encontrados a partir do projeto, a seguir os passos são definidos de modo a reduzir os riscos e após essa análise são planejadas estratégias alternativas.

2.7.3 Terceira etapa – Desenvolvimento e validação

Nesta etapa, após a avaliação e redução dos riscos, é escolhido um paradigma para o desenvolvimento do software.

2.7.4 Quarta etapa – Planejamento

Nesta etapa é onde todo o projeto é analisado para verificar o que foi realizado e definir os próximos passos para continuar o ciclo do espiral ou finalizar o sistema.

Dentre diversos modelos de ciclo de software neste trabalho optou-se pela utilização do modelo em espiral em razão da sua relação direta com a análise de riscos, a Figura 12 demonstra o modelo em espiral de Boehm.

A volta mais interna deve se preocupar com a viabilidade do sistema e o ciclo seguinte se preocupa com a definição dos requisitos, o próximo tem o objetivo com o projeto do sistema e assim por diante (SOMMERVILLE, 2011).

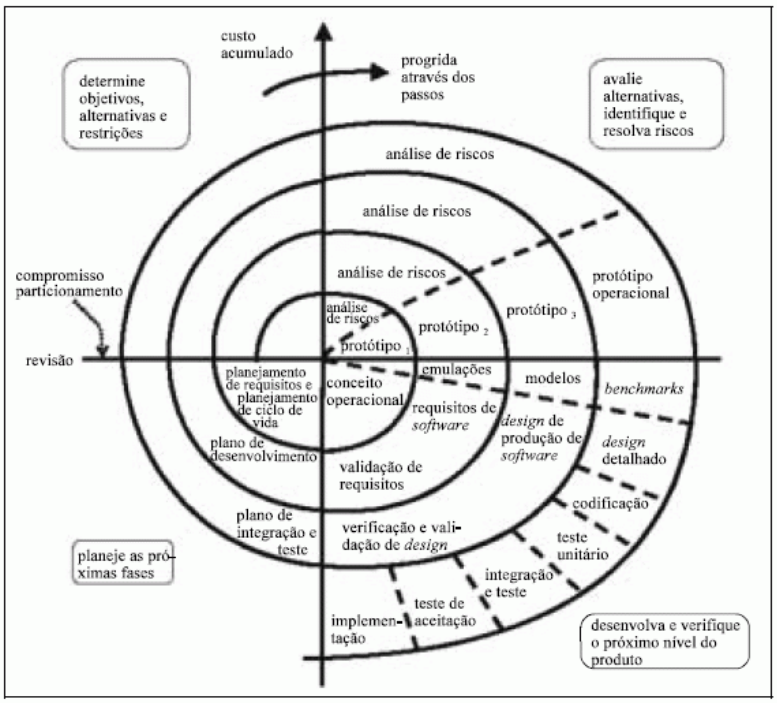
Segundo Sommerville (2007), a principal diferença entre o modelo em espiral e outros modelos de software é a evidente consideração dos riscos.

Em um software os riscos são de suma importância e a parte principal, exatamente como ocorre em outras áreas (CHADBOURNE; SANDERS, 1999).

Obviamente, desde que Boehm desenvolveu o modelo em espiral, a área que realiza o tratamento de riscos em engenharia de software já evoluiu muito. Essa evolução pode ser notada por que passou de uma análise dentro do modelo de desenvolvimento como Boehm havia proposto para se tornar um leme que deve transpor todos os processos de ciclo do software.

O processo que realiza o gerenciamento dos riscos tem seu início a partir das incertezas, preocupações, desconhecimento e dúvidas que por fim irão se tornar riscos aceitáveis (MACHADO, 2002).

Figura 12 – Modelo espiral de Boehm



Fonte: Adaptado do modelo espiral de Boehm (1988)

De acordo com Nogueira e Abe (2008), um produto de software necessita de um gerenciamento de riscos eficiente e sistemático, a fim de tratar os fatores de risco para minimizar os seus efeitos e cumprir prazos, manter custos estimados e, além disso atender as necessidades do cliente.

3 TEORIAS DE APRENDIZAGEM E DESIGN PARA DESENVOLVIMENTO

Neste Capítulo traremos as teorias de aprendizagem estudadas durante o desenvolvimento deste trabalho.

3.1 Design instrucional

A elaboração de material instrucional de qualidade para ambientes de ensino é uma ferramenta de relevância considerável que permite a aprendizagem a partir de técnicas, métodos, estratégias, atividades, materiais e recursos em situações didáticas, sistema de avaliação, que serão utilizados no ensino-aprendizagem, isso é denominado de Design Instrucional (FILATRO, 2008).

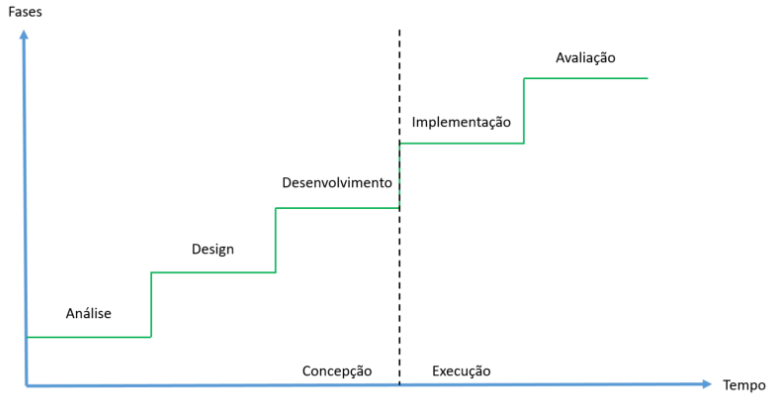
O design Instrucional ou projeto instrucional é um termo utilizado para se referir a Engenharia Pedagógica e está baseado em três pilares fundamentais, são eles: o primeiro é o conteúdo e necessita que o ensino seja praticamente autônomo, o segundo é o recurso tecnológico utilizado sendo usabilidade, custo e facilidade os critérios de relevância importantíssima, já o terceiro pilar está relacionado ao *design*, desta forma facilidade de utilização, *layout* de qualidade e conforto são os princípios fundamentais. Segundo Romiszowski e Romiszowski (2005), existem variados tipos de materiais instrucionais e estes não atendem a um único formato, o Design Instrucional demonstra que o formato resultante deriva dos objetivos de aprendizagem, do tipo de conteúdo e do público-alvo.

A criação do Design Instrucional iniciou durante a Segunda Guerra Mundial, os americanos precisavam de uma forma de treinamento em massa e rápido para realizar tarefas complexas e de alto nível. As tarefas eram divididas em subtarefas e depois separadas com um único objetivo de aprendizagem, esta divisão utilizou a teoria de Burrhus Frederic Skinner, ele foi um psicólogo americano Filatro e Piconez (2007).

O *design* instrucional é baseado em modelos, o utilizado no projeto do SAELF, foi o ADDIE (*Analyze, Design, Develop, Implement, Evaluate*), neste modelo inicialmente é necessária a realização da primeira etapa a *Analyze* 3.1.1 (análise), onde são reunidas as informações sobre futuros aprendizes, as tarefas a serem concluídas e todos os objetivos do projeto. A próxima etapa é o *Design* 3.1.2, que para o ADDIE significa planejamento a partir da análise, desta forma, as etapas seguintes são descritas nas subseções: *Development* 3.1.3 (desenvolvimento), *Implementation* 3.1.4 (implementação) e *Evaluation* 3.1.5 (avaliação).

Segundo Kurt (2018), desenvolvedores, designers instrucionais e educadores acham este tipo de abordagem eficiente e muito útil por possuir etapas definidas, facilitando a implementação e desenvolvimento de ferramentas de treinamento eficientes. Desta forma, o modelo ADDIE é de ampla aceitação e utilização. A Figura 13 se refere as etapas ou fases do modelo ADDIE em relação ao tempo, no eixo y temos as fases do modelo e no eixo x temos a relação de tempo.

Figura 13 – Fases Modelo ADDIE



Fonte: Filatro (2008)

3.1.1 *Analyze*

A etapa de análise consistiu em estudar o perfil do aluno ou usuário que utilizaria o sistema e dos fatores que contribuiriam para auxiliá-lo no aprendizado de linguagens formais. Desta forma, foram inicialmente selecionados os requisitos do sistema analisando as necessidades do público-alvo.

Os requisitos selecionados foram:

- Fundamentação teórica relevante para a compreensão do objeto de estudo;
- Recursos materiais e humanos disponíveis para modelagem do sistema e design do material de apoio;
- Grau de interesse do aluno ao sistema e conteúdo apresentado;
- Grau de aproveitamento do sistema.

3.1.2 *Design*

Utilizando as necessidades de atender ao conteúdo da disciplina como uma ferramenta de aprendizagem, a ferramenta deve:

- Realizar o teste de Expressões Regulares;
- Realizar a criação e teste de Autômatos Finitos Determinísticos;
- Realizar a criação e teste de Autômatos Finitos Não Determinísticos;
- Compreensão do material de apoio e vídeo aulas.

3.1.3 *Development*

No desenvolvimento do sistema e do material de apoio foi necessário responder às perguntas básicas como assinaladas pelo modelo, assim a tecnologia utilizada é:

- Utilização de softwares como Xampp;
- Ferramentas para PHP e Frameworks;
- Utilização de ferramentas para edição de vídeos;
- O material utilizado foi um computador com acesso à internet.

Na etapa de análise foi estudado o tipo de utilização do sistema e quais ferramentas seriam importantes para que o sistema oferecesse um apoio ao usuário da disciplina de linguagens formais que realmente fosse eficiente. Baseado nesses dados, foi possível desenhar um diagrama para determinar as etapas do desenvolvimento e verificar a existência de possíveis dificuldades de aprendizado ou problemas no uso do sistema (entendimento do uso do sistema, material de apoio, vídeo aulas e tutoriais, diagramas, "etc").

3.1.4 *Implement*

O sistema e o material de apoio é utilizado de forma interativa pelo usuário. Baseado no material de apoio e utilização do sistema, foram determinados os módulos:

- Módulo de geração de palavras: realizando a combinação de palavras de um determinado alfabeto.

- Módulo de teste de Expressões Regulares: onde o usuário insere a Expressão Regular e a cadeia de teste e verifica se ela pertence ou não a Expressão.
- Modulo de Criação e teste de Autômatos Finitos Determinísticos: onde o usuário cria, edita e testa o seu autômato conforme a sua necessidade.
- Modulo de Criação e teste de Autômatos Finitos Não Determinísticos: onde o usuário cria, edita e testa o seu autômato conforme a sua necessidade
- Modulo de material de apoio: onde o usuário encontra diversas vídeo aulas, listas de exercícios e apostilas para auxílio da aprendizagem.

3.1.5 *Evaluate*

A fim de realizar a avaliação do modelo de ensino apresentado, foi desenvolvido um questionário de avaliação quantitativo, a fim de transformar o resultado em estatísticas.

Os resultados apresentados na avaliação demonstram os níveis de satisfação do usuário, permitindo também o uso dessa avaliação como *feedback* de modo a realizar melhorias no sistema.

3.2 *Design Science Research*

Aproximadamente na década de 1960 surgiu o *Design Science Research* (DSR), e um dos primeiros autores a utilizar este termo em seus projetos de arquitetura, engenharia e sustentabilidade foi Richard Buckmister Fuller. Segundo Hevner (2007), a *Design Science* passou a ser bastante difundida já em meados da década de 1970, especialmente nas áreas de engenharia. Dessa forma, foi amplamente adotada também em projetos que envolvessem engenharia elétrica, computação e ciência da computação devido à sistematização na concepção de artefatos. É denominado um artefato: *frameworks*, arquiteturas, métodos, modelos, princípio de *design* e instanciações.

Segundo Dresch, Lacerda e Júnior (2015), a DSR possui como base fundamental duas áreas de pesquisa sendo o design e a ciência do comportamento, essas duas áreas de pesquisa se complementam e possuem uma relação cíclica.

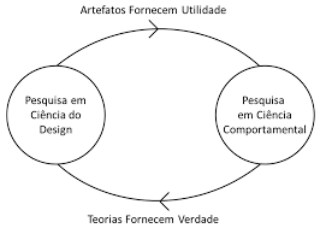
De acordo com Simon (1980), a DSR, é um paradigma de pesquisa que necessita a criação de artefatos inovadores, a fim de resolver os diversos problemas relacionados ao mundo real. O *design Science Research* é uma abordagem que possui dois objetivos:

- Desenvolver um artefato que resolva um problema;
- Gerar novos conhecimentos técnicos e científicos.

Quando se fala em conhecimento técnico e científico a maioria das pessoas tem dificuldade em diferenciá-los, mas de acordo com Wazlawick (2014) a ciência constrói as teorias a fim de explicar o que se é observado, já a tecnologia é prática, e existe para mudar o mundo, e não para criar teorias.

Hevner e Chatterjee (2010, p.11) elaboraram a Figura 14 para ilustrar a base fundamental do *Design Science Research*.

Figura 14 – Base fundamental do *Design Science Research* (DSR)



Fonte: Hevner e Chatterjee (2010, p.11)

Nesta imagem é possível observar que o desenvolvimento de um artefato possui a necessidade de se apoiar na ciência (Pesquisa em Ciência) e a utilização dos artefatos possibilita a pesquisa científica. Assim, o que se pode compreender é que o conhecimento técnico e o conhecimento científico andam lado a lado, e é necessário estabelecer uma relação entre eles, além disso, essa relação deve se estender também entre artefato e teoria e tecnologia e ciência.

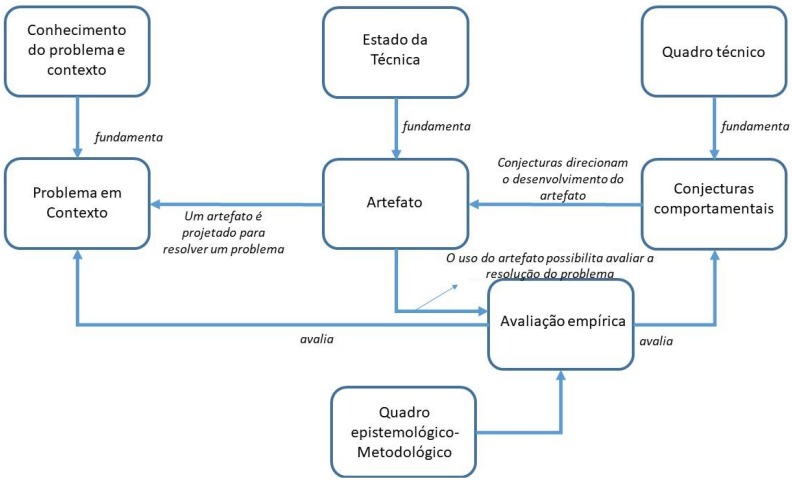
Normalmente existem diversas abordagens a fim de direcionar as pesquisas relacionadas a DSR, mas a meta é sempre a mesma: atingir ou alcançar os dois objetivos principais.

Na Figura 15 estão representados os principais elementos do modelo DSR, neste modelo o desenvolvimento de um artefato é baseado em suposições sobre como as pessoas adquirem conhecimento, trabalham, comunicam, pensam, etc. Assim o artefato é desenvolvido para resolver um problema de contexto.

Com base nisso, ao utilizar o artefato é possível analisar se o problema em questão foi resolvido e se as conjecturas são válidas. Portanto, é desenvolvido o conhecimento técnico e científico.

Esta Figura 15 apresenta os principais elementos do *Design Science Research*, sendo necessário possuir conhecimento a respeito do problema e em qual contexto ele

Figura 15 – Elementos principais DSR



Fonte: Adaptado de Pimentel, Filippo e Santos (2020)

se enquadra, sendo assim para encontrar uma solução para um determinado problema antes é necessário estudá-lo. Ao criar um projeto de um artefato devemos direcioná-lo por conjecturas comportamentais que devem ser embasadas em teorias que por fim fazem parte do quadro teórico. Por isso é necessário estabelecer uma conexão entre conhecimento técnico e teórico.

Na criação de um novo artefato nem sempre deve seja algo totalmente novo, pode ser algo que já exista, porém, é preciso realizar um levantamento das técnicas já existentes e identificar as suas soluções, a partir disso efetuar uma pesquisa do design a fim de analisar e projetar a solução.

Na avaliação empírica podemos utilizar:

- sessão de uso do artefato;
- casos;
- oficinas;
- técnicas de coleta;
- produção de dados;
- testes;
- questionários;
- técnicas de análise;
- entre outros métodos.

Neste tipo de avaliação é necessário considerar alguns procedimentos que

tenham como objetivo o rigor científico e exige a elaboração de um quadro epistemológico-metodológico. Este tem por objetivo listar as principais referências relacionadas a metodologia de pesquisa e também deve incluir:

- Abordagem epistemológica;
- Método de pesquisa;
- Técnicas de produção/coleta de dados;
- Técnicas de analise/interpretação;

Incluso no módulo do artefato existe os critérios de verificação que basicamente servem para avaliar se o artefato definitivamente funciona, se não existem problemas que impossibilitem a utilização. E também temos os critérios de aceitação, que basicamente servem para sinalizar se um problema foi resolvido ou não, esses critérios de aceitação são importantes para indicar se o artefato cumpre o que promete na resolução do problema proposto.

4 TRABALHOS CORRELATOS

Esta seção contém uma análise sobre três ferramentas que possuem características similares ao sistema proposto e relacionadas a simulação de autômatos, destacando suas principais funcionalidades, limitações e carências. Todas essas ferramentas permitem a criação e simulação de modelos computacionais e podem ser utilizadas por alunos e professores em diversos contextos.

4.1 Auger Simulador de Autômatos

O Auger[1] é um ambiente para construção e simulação de autômatos finitos, criado por Charbel Szymanski, a versão beta deste simulador foi implementada em Delphi e lançada em 2004. O objetivo principal deste ambiente é o apoio didático no ensino de linguagens formais, mais especificamente no ensino de autômatos finitos. Ele foi implementado em Java, e através dele é possível realizar as seguintes tarefas:

- Criar autômatos finitos de forma gráfica (a partir do diagrama de estados);
- Executar algoritmos de manipulação de autômatos finitos determinísticos e não determinísticos;
- Testar autômatos finitos através do recurso de simulação de cadeias de entrada;
- Utilizar expressões regulares e gramáticas regulares para gerar autômatos finitos;
- Gerar expressões regulares e gramáticas regulares a partir de autômatos finitos;
- Gerar programas em linguagem Java para testar os autômatos criados ou criar analisadores léxicos;
- Criar diagramas de estados e utilizá-los como imagens em outros softwares.

Este software está disponível para download no site do projeto, portanto não é online e também não é necessário ser instalado. O usuário realiza o download e executa em seu computador apenas quando desejar utilizá-lo.

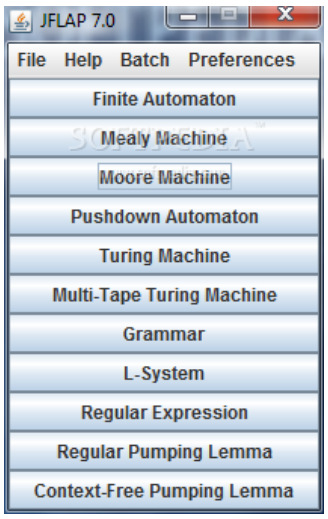
No site do projeto é possível encontrar alguns tutoriais de utilização para simplificar a sua utilização e também um breve histórico das versões que já estiveram disponíveis para download.

[1] C. Szymanski. Auger - **Ambiente para construção e simulação de autômatos finitos.** Disponível em: <www.evoluma.com/auger/> Acesso em: 21 jul. 2017

4.2 JFLAP

O JFLAP[2] é um software para facilitar o aprendizado de linguagens formais, nele estão inclusos autômatos finitos determinísticos e não determinísticos, autômatos *pushdown* não determinísticos, máquinas de Turing multi-fita, vários tipos de gramáticas, análise e sistemas L. Além de construir e testar exemplos para estes, o JFLAP permite que se experimente conversões de um tipo de autômato para outro, como, por exemplo, converter um AFND para um AFD ou um AFD de estado mínimo, para uma expressão regular ou gramática regular. O JFLAP foi implementado em Java então também não é online, não é necessária sua instalação no computador do usuário, a ferramenta é apenas executada, porém, exige que o mesmo possua a instalação do Java 1.4 ou superior. No sistema não existem tutoriais explicando a sua utilização, porém, existe um link para um site externo que possui algumas informações. A ferramenta é totalmente em inglês, a Figura 16 demonstra como é o menu de utilização da ferramenta.

Figura 16 – Menu de utilização do JFLAP



Fonte: jflap.org

[2] **JFLAP**. Disponível em: <www.jflap.org/jflaptmp>. Acesso em: 21 jul. 2017

4.3 Simulador de Autômatos

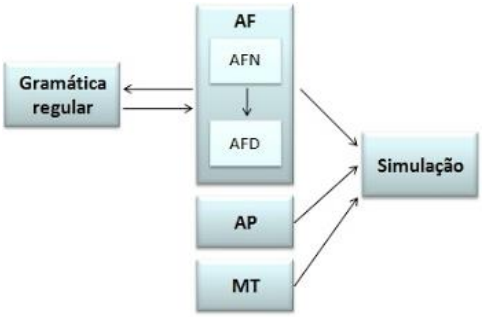
O Simulador de Autômatos[3] é uma ferramenta para a criação, simulação e conversão de modelos formais, desenvolvida como projeto de iniciação científica na Universidade de Uberaba (UNIUBE) e depois continuado como Trabalho de conclusão de curso visando auxiliar o aprendizado de Linguagens Formais e Autômatos (RIBEIRO JUNIOR, 2007). É uma ferramenta de fácil utilização, porém, não é online, é necessário efetuar o download e executá-la no computador do usuário, ou seja, não é necessário instalar, ela é apenas executada no computador. Como ela é uma ferramenta com bastante utilidade e fácil de utilizar, não possui tutoriais em seu site, existe apenas um vídeo explicando como utilizar a ferramenta. A sua implementação foi realizada em Java.

Ele reconhece e simula os seguintes modelos abaixo:

- Autômatos Finitos Determinísticos;
- Autômatos Finitos Não-Determinísticos;
- Autômatos com Pilha;
- Máquinas de Turing;
- Gramáticas Regulares.

A Figura 17 representa a arquitetura da ferramenta, ilustrando as funcionalidades citadas acima.

Figura 17 – Arquitetura da ferramenta



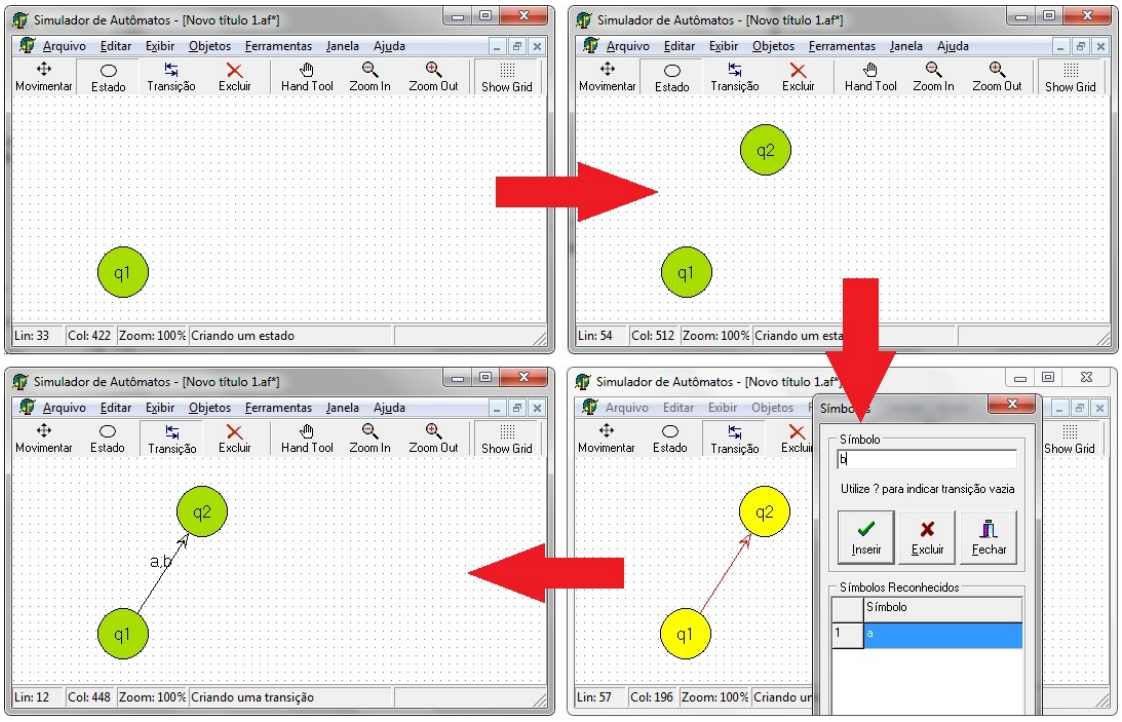
Fonte: Site do projeto Simulador de Autômatos

A Figura 18 ilustra como criar um autômato a partir desta ferramenta seguindo um passo a passo. Inicialmente é criado o estado inicial q_1, após isso é criado o segundo estado do autômato q_2, por fim é inserida uma transição com o símbolo correspondente

[3]Ribeiro Junior (2007). **Simulador de Autômatos**. Disponível em: <www.simuladordeautomatos.com>. Acesso em: 20 jul. 2017

do autômato.

Figura 18 – Exemplo de criação de autômato – Simulador de Autômatos



Fonte: Página do projeto do Simulador de Autômatos

4.4 Outros trabalhos

Outros sistemas também foram analisados como, por exemplo, o RegexPal[4] implementado para a plataforma *Android* e é um testador de Expressões Regulares, ele é *off-line* e necessita ser instalado no *Smarthphone*. É um aplicativo totalmente em inglês, porém de fácil utilização.

FSM Simulator[5] é um simulador de autômatos totalmente em inglês e online sua utilização é um pouco complexa devido a não deixar explicita toda a sua utilidade e não possuir tutoriais de auxílio ao usuário.

Regex101[6] é um testador de Expressões Regulares totalmente em inglês e online, é uma ferramenta bastante complexa de se utilizar devido à falta de tutoriais, ou seja, tem que aprender utilizando.

[4]F. DIB. **RegexPal**. Disponível em: <www.regexpal.com>. Acesso em: 17 mai. 2017
[5]F. PIERSON. **FSM UI Test**. Disponível em: <www.radford.edu/npierson/fsm/fsmSimulator.html>. Acesso em: 21 mai. 2017
[6]F. DIB. **Regex101**. Disponível em: <https://regex101.com/>. Acesso em: 19 mai. 2017

A fim de demonstrar as vantagens de implementação do SAELF, foi elaborada a Tabela 8, que realiza uma comparação com os sistemas analisados. Com isso fica evidente a importância desta ferramenta de aprimoramento de estudos na disciplina de Linguagens Formais. O SAELF é um sistema mais completo que os outros, ao atender na totalidade os requisitos observados.

Tabela 8 – Comparação de sistemas encontrados

Nome	Simular Autômatos	Teste Exp. Regular	Geração Palavras	Online	Manual Tutoriais	Vídeos
SAELF	Sim	Sim	Sim	Sim	Sim	Sim
RegexPal	Não	Sim	Não	Não	Não	Não
JFLAP	Sim	Sim	Não	Não	Não	Não
Sim. Autômatos	Sim	Não	Não	Não	Sim	Não
FSM Simulator	Sim	Não	Não	Sim	Não	Não
Regex101	Não	Sim	Não	Sim	Não	Não
Auger	Sim	Não	Não	Não	Sim	Não

Fonte: Próprio autor

5 METODOLOGIA

Para a implementação inicial do sistema foi necessária a utilização de algumas ferramentas de apoio. Este Capítulo apresenta as ferramentas utilizadas e as etapas seguidas.

5.1 Etapas de Execução

Nesta seção, estão informações pertinentes relacionadas as etapas de execução do projeto.

Etapa 1: levantamento bibliográfico

Nesta etapa foi realizado um estudo de materiais bibliográficos que abordassem MVC, PHP, expressões regulares, autômatos finitos, desenvolvimento de *software* e engenharia de *software*. Também foi realizada uma busca por simuladores de autômatos e testadores de expressões regulares já existentes.

Etapa 2: análise dos sistemas encontrados

Esta etapa se efetivou a partir da análise das ferramentas encontradas na etapa anterior. Dentre os diversos simuladores de autômatos e ferramentas para teste de expressões regulares encontrados, foram analisados apenas os que apresentavam características mais semelhantes aos objetivos deste trabalho. O objetivo é a observação de principais funcionalidades como tipo de autômatos reconhecidos, execução no reconhecimento de expressões, *layout*, limitações e disponibilidade.

Etapa 3: levantamento de requisitos

Durante esta etapa, foram levantados requisitos funcionais e não funcionais para que a ferramenta fosse desenvolvida e estes se encontram no apêndice A. Como referência, foram utilizados os resultados da etapa anterior que previa observar funcionalidades de ferramentas existentes em relação à simulação de autômatos, teste de expressões regulares e informática na educação.

Etapa 4: ambiente de desenvolvimento do sistema

Desenvolver uma aplicação Web com uma interface visual atrativa que seja útil e simplificada pode ser uma tarefa complexa se as ferramentas disponíveis não forem utilizadas corretamente. Durante esta etapa, algumas bibliotecas do Java Script como Jquery, JSplumb[1] foram analisadas, verificando seu suporte para o desenvolvimento de uma interface, com isso obteve-se também um conhecimento na área de CSS3 e HTML. Paralelamente, foi realizada uma busca por APIs abertas que auxiliassem no desenvolvimento de aplicações que manipulassem conectores e objetos em aplicações Web.

Etapa 5: desenvolvimento do sistema

A etapa de desenvolvimento do sistema primeiramente se deu através da criação de uma estrutura básica que permitia dividir o sistema em três pilares fundamentais que seriam o modelo, a visão e o controlador do padrão MVC (*Model View controler*) facilitando assim, a escrita do código e, também a sua modificação se necessário. Normalmente a maioria das modificações que eventualmente sejam necessárias neste padrão é realizada na parte da visão que é onde existe interação com o usuário, já na parte do modelo e controlador as alterações são respectivamente muito mais raras. Em seguida foram implementadas funcionalidades como: conexão com um banco de dados para gerenciar o cadastro de alunos, professores e, também, o *login/logout*, menu de navegação, a geração de palavras a partir dos símbolos inseridos por um usuário, teste de expressões regulares, simulação de autômatos finitos. A função de salvar os exercícios e enviá-los ao professor responsável foi o próximo recurso a ser implantado, permitindo que os discentes realizassem os exercícios e enviassem de forma simplificada. No final desta etapa, foram implementadas melhorias no código e sua interface para que a proposta deste trabalho fosse concluída da melhor forma possível.

[1] **JSPLUMB**. Disponível em: <https://jsplumbtoolkit.com/>. Acesso em: 21 jul. 2017.

5.2 Ferramentas utilizadas

Este subcapítulo é destinado a apresentar as ferramentas utilizadas durante o desenvolvimento do sistema.

5.2.1 PHP (Editor)

O PHP é um acrônimo para (*Hypertext PreProcessor*) é uma linguagem de programação baseada em scripts embutida no HTML. Existem diversos editores gratuitos de PHP como, por exemplo, o Notepad++[2] e Kate entre outros, neste sistema ambos foram utilizados, porém, o mais utilizado foi o Notepad++ (versão 7.3.3) por possuir suporte a diversos tipos de linguagem de programação e também pela facilidade e praticidade em sua utilização. Este editor, além de possuir uma vasta documentação em português, permite uma rápida implementação sugerindo as sintaxes e palavras mais utilizadas, com apoio do CSS3, JavaScript e HTML5 o desenvolvimento de uma interface gráfica em alto nível fica muito facilitado permitindo assim efeitos importantes na visualização do sistema

5.2.2 JavaScript (JS)

Para ser possível criar os autômatos finitos, é necessário primeiro poder desenhar e mover os estados, inserir as transições com seus símbolos. Os autômatos devem poder ter seus componentes criados, removidos e remanejados em tempo real. A parte da implementação responsável sendo o jsplumb e jquery que são *Frameworks* do JavaScript[3], o primeiro é responsável pelas conexões ou transições, ele é completamente livre e disponível sob licença MIT, a estrutura funciona muito bem para o desenvolvimento Web.

O jquery[4] foi criado sobre o mantra "*Write less, do more*", ou seja "escreva menos, faça mais" e é por isso que ele é prático de se utilizar com poucas linhas de código, pode-se criar os mais diversos efeitos, ele funciona baseado em eventos, o jquery é responsável pela animação das transições durante o teste, janelas de diálogo e também pelo movimento

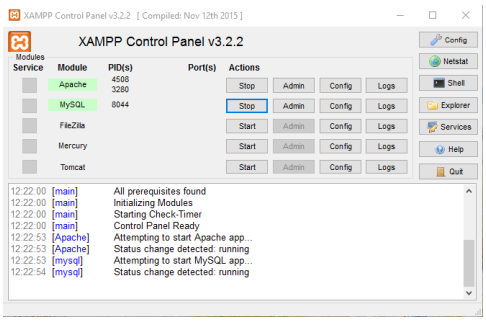
[2]**Notepad++.** Disponível em: <https://notepad-plus-plus.org/download/v7.4.1.html>. Acesso em: 21 mai. 2017.
[3]**JS JavaScript**. Disponível em: <https://www.javascript.com/> Acesso em: 21 mai. 2017.
[4]**JQUERY**. Disponível em: <https://jquery.com/>. Acesso em: 21 jul. 2017

dos estados do autômato e transições recalculando sua posição dinamicamente. Este *framework* possui funcionalidades voltadas para animações e efeitos, é de código aberto.

5.2.3 XAMPP

O *XAMPP*[5] é um software que possui um pacote com os principais servidores web de código aberto do mercado, incluindo *FTP*, banco de dados *Apache, MySQL, FileZilla, Mercury e Tomcat* com suporte às linguagens como *PHP* e *Perl* além de outras linguagens e recursos, como um cliente *FTP*. Com ele, é possível executar sistemas como *WordPress* e *Drupal* localmente, o que facilita e agiliza o desenvolvimento. O software possui um pacote de instalação automática que cria um ambiente local pronto para gerir página Web. O objetivo do *XAMPP* é construir uma distribuição fácil de instalar para desenvolvedores entrarem no mundo do Apache. Para torná-lo conveniente para os desenvolvedores, o *XAMPP* é utilizado atualmente para servir sites Web, e com algumas modificações é geralmente seguro para uso em servidor público. Uma ferramenta especial é incluída para proteger facilmente as partes mais importantes e sensíveis do pacote. O *XAMPP* possui muitos aplicativos, dentre eles o, *phpMyAdmin, FTP Server, OpenSSL*, e sua principal vantagem é a versatilidade, em alguns minutos são instalados todos esses programas. (SEIDLER, 2014). A Figura 19 demonstra como é o painel de controle do *XAMPP* e suas funcionalidades, nela existem os módulos, botões de *start/stop*, admin, *config*, *logs* entre outros serviços.

Figura 19 – Painel de controle do Ambiente de desenvolvimento PHP – XAMPP



Fonte: XAMPP

[5] **XAMPP Ambiente de desenvolvimento PHP**. Disponível em: <https://www.apachefriends.org/pt_br/index.html>. Acesso em: 01 jun. 2017.

5.2.4 MVC – Model View Controller

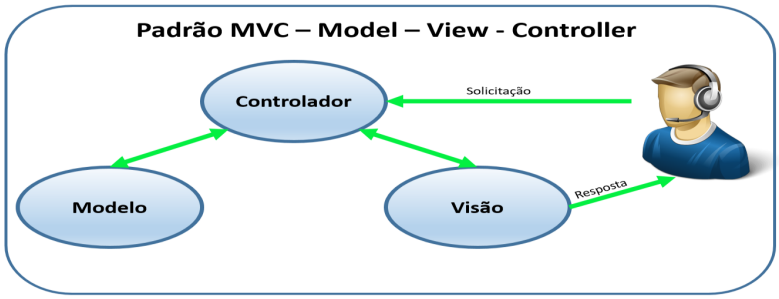
MVC é o acrônimo de *Model, View e Controller* um modelo de padrão de projeto altamente difundido em Engenharia de Software e largamente utilizado em aplicações WEB, sua facilidade e praticidade consiste em organizar o sistema em 3 partes fundamentais sendo elas Modelo, Visão e Controlador (PEREIRA, 2014).

O Modelo possui as funcionalidades e os dados principais, a visão é responsável por apresentar os dados a um determinado usuário e solicitar ao controlador o que deve ser mostrado, ou seja, a interface Web, o Controlador é onde os eventos acontecem, pode-se dizer que o controlador é responsável por receber as requisições do usuário e ele sabe qual modelo utilizará e envia a resposta para a visão que no que lhe concerne é enviada ao usuário (MINETTO, 2007).

A Figura 20 está exemplificando como funciona a sequência de um padrão de projeto MVC.

Neste padrão o usuário realiza uma solicitação na parte da visão, enviada ao controlador, onde este solicita ao modelo. Logo após isso, o processo é revertido, sendo assim, o modelo envia ao controlador a informação, que envia para a visão responsável por exibir ao usuário aquela informação que foi solicitada.

Figura 20 – Diagrama Padrão MVC



Fonte: Próprio autor

6 DESENVOLVIMENTO

Este Capítulo tem por finalidade demonstrar o funcionamento do Sistema de Apoio ao Estudo em Linguagens Formais – SAELF.

6.1 Introdução

O sistema SAELF, desenvolvido neste trabalho, tem por objetivo auxiliar os alunos da disciplina de linguagens formais. Para isso, o sistema fornece exercícios da disciplina e possui ferramentas para construção de autômatos finitos determinísticos e não determinísticos, teste de expressões regulares e combinação de símbolos, gerando assim, palavras sobre um determinado alfabeto. Os professores que solicitarem o seu cadastro terão a possibilidade de inserir exercícios e disponibilizar aos seus alunos que devem resolvê-los e enviar ao seu professor para sua devida avaliação, embora o sistema baseado nos testes já informe ao seu usuário se o exercício está ou não correto. A avaliação do professor, também disponível no sistema, pode ser uma alternativa interessante para que o usuário saiba onde errou ou até mesmo para atribuir um conceito aos seus alunos.

Qualquer pessoa poderá se cadastrar como um usuário comum, ou seja, para simplesmente utilizar o sistema. Para inserir exercícios, apenas professores possuirão esse privilégio por intermédio de solicitação de cadastro ao administrador do sistema, essa opção se faz necessária para tentar garantir que o conteúdo corresponda com a proposta do trabalho.

Para representarmos um sistema web, de forma intuitiva e atrativa ao usuário, se faz necessário o desenvolvimento de uma interface simplificada, porém, ao mesmo tempo, repleta de recursos, permitindo assim a sua utilização.

O protótipo visa uma solução para a especificação do sistema em alto nível através de testes de expressões regulares, autômatos finitos e construção de palavras, onde não só a construção de autômatos é possível, como também salvar em um formato específico para validações e geração do código de forma automática ele armazena as posições dos estados e transições. A linguagem escolhida para o desenvolvimento deste protótipo foi o PHP, por ter suporte para os mais diversos navegadores e contar com bibliotecas que facilitam a criação e manejo dos estados de um autômato, da mesma maneira que facilitam a manipulação dos eventos.

Para que o protótipo pudesse ser implementado, foi realizado um levantamento

de requisitos que resultou em uma lista de requisitos funcionais e não funcionais. Os requisitos que a ferramenta deve atender estão listados a seguir:

6.1.1 Requisitos Funcionais

- O sistema deve permitir ao usuário testar expressões regulares;
- O sistema deve permitir ao usuário a criação de palavras;
- O sistema deve permitir ao usuário a construção de autômatos finitos determinísticos;
- O sistema deve permitir usuário a construção de autômatos finitos não determinísticos;
- O sistema deve permitir ao usuário salvar o progresso do seu autômato;
- O sistema deve permitir ao usuário retomar o progresso do seu autômato;

6.1.2 Requisitos Funcionais

- Apresentar interface amigável de utilização;
- Apresentar compatibilidade em navegadores;
- Apresentar segurança;
- Apresentar internacionalização.

6.2 Protótipo Inicial

A implementação do protótipo teve como passo inicial um estudo o *framework Jquery* responsável pelas animações, como, por exemplo, movimentar o estado de um autômato de um lado para o outro, também é responsável pelas janelas de diálogos ao inserir as transições, outra responsabilidade do *Jquery* é durante o teste do autômato alterar a cor da transição que está sendo testada isso pode ser denominado como um evento. Outro *framework* que também foi analisado é o *JsPlumb* responsável pelos conectores, ou seja, as transições. Esse estudo foi realizado através da análise de exemplos disponíveis no site oficial dos *frameworks* e através da leitura da documentação específica. Testes iniciais foram realizados com códigos exemplos de *Jquery* e *Jsplumb*

até que a implementação do sistema fosse iniciada. O projeto iniciou estruturando a base do sistema utilizando o padrão MVC e a criação do arquivo *app.php* que recebe dois parâmetros obrigatórios, o primeiro determina o arquivo de *controller* e o segundo determina o método a ser executado dentro da classe de controle a partir disso. O arquivo de controle é responsável por fazer o processamento e a coleta de dados nas classes de modelo e prepara a variável para a impressão nos arquivos da *view*. As classes de modelo são acessadas somente por arquivos de controle, recebendo parâmetros e retornando resultados para serem exibidos indiretamente pela *view*, porém o responsável por passar o resultado é totalmente das classes de controle. A *view* recebe os parâmetros *PHP* e os imprime em meio a *tags* HTML, facilitando assim a manutenção, com isso pouco processamento é requerido neste trecho. A partir disso foram dados os primeiros passos no desenvolvimento do sistema e começam a ser adicionadas as funcionalidades propostas.

6.3 Características do Sistema

O sistema de Apoio ao Estudo de Linguagens Formais, denominado SAELF, foi desenvolvido para a plataforma web, utilizando como linguagem de programação PHP e o padrão de projetos MVC (Modelo, Visão e Controle).

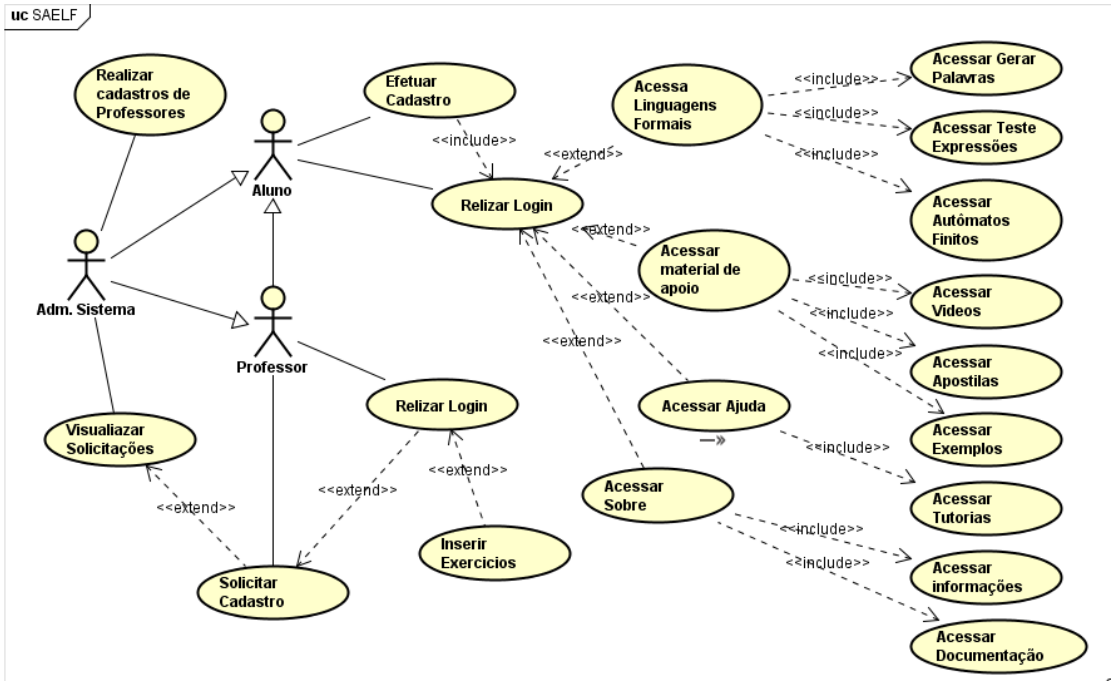
O sistema tem por objetivo possibilitar ao usuário criar e manipular autômatos finitos determinísticos, autômatos finitos não determinísticos, testar expressões regulares, e gerar combinações de símbolos. Também disponibilizar tutoriais de utilização, vídeos explicativos, materiais de estudos e listas de exercícios.

A fim de que o sistema possa ser utilizado por qualquer pessoa se pretende passar o sistema por um processo de internacionalização, ou seja, oferecer uma opção para a utilização do sistema em inglês e espanhol.

Para a fase de projeto de software foi utilizada a Linguagem de Modelagem Unificada (UML – *Unified Modeling Language*) em que os diagramas foram construídos utilizando a ferramenta Astah versão de estudantes, esses modelos são utilizados para representar tanto a estrutura quanto o comportamento do sistema, na Figura 21 é possível visualizar o caso de uso geral do sistema, onde todas as funcionalidades são exibidas.

Pode-se perceber que na Figura 21 estão sendo utilizados três atores principais que são eles: Administrador do sistema, Professor e Aluno. O administrador do sistema tem permissão geral, ele pode cadastrar alunos e professores, além de utilizar todo o sistema.

Figura 21 – Diagrama de caso de uso do SAELF



Fonte: Próprio autor

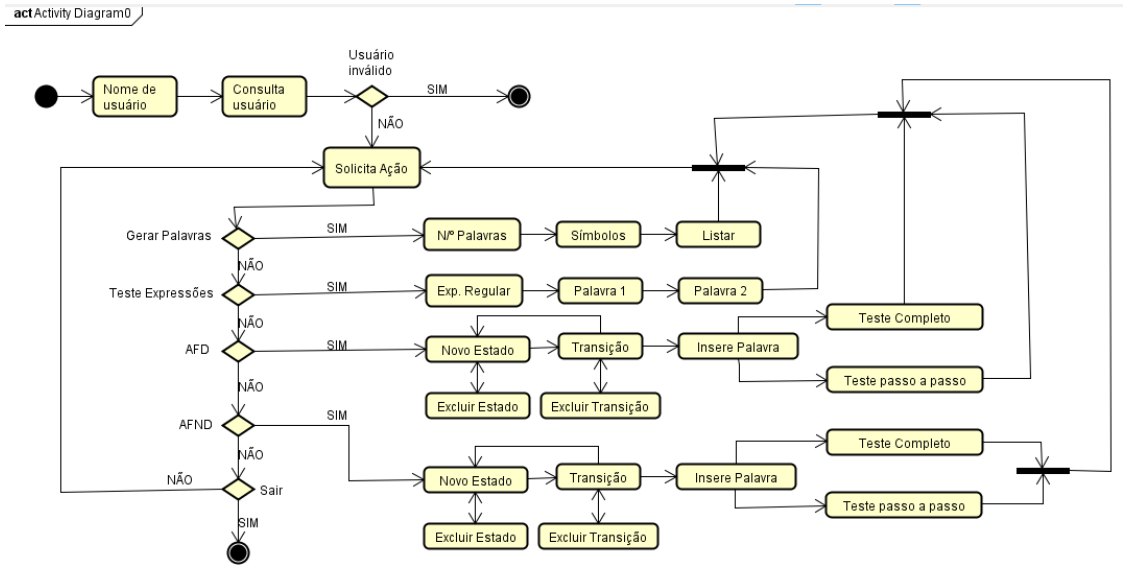
Já os professores têm permissões para utilizar o sistema e inserir exercícios, porém para garantir a segurança do sistema só o administrador pode cadastrar um professor, portanto os professores devem solicitar o cadastro ao administrador. Os alunos têm permissão apenas para utilização do sistema e se cadastrar.

O diagrama deixa explícita essa hierarquia utilizando a generalização, em outras palavras significa que o ator que possui a generalização pode fazer tudo que lhe está atribuído e também tudo que está atribuído ao seu ator subordinado. O diagrama de atividade ilustra graficamente como será o funcionamento do SAELF, então a Figura 22 demonstra a lógica do sistema.

No diagrama de atividades da Figura 22 pode-se perceber que o usuário deve realizar o login e o sistema realiza o teste de validação se for válido solicita a ação, sendo inválido retorna para a tela inicial de login e cadastro de usuários.

Sendo o usuário válido, este pode optar por algumas opções de navegação como gerar palavras, testar expressões regulares, autômatos finitos, autômatos finitos não determinísticos e sair. Qualquer uma dessas ações permite ao usuário realizar a devida tarefa correspondente.

Figura 22 – Diagrama de atividades SAELF



Fonte: Próprio autor

6.4 Criação de Autômatos

O SAELF permite criar autômatos através de uma interface gráfica de fácil manipulação, ou seja, a ferramenta possui um editor gráfico onde o usuário adiciona estados, pode movimentá-los e criar as transições apenas com movimentos do mouse. O funcionamento ocorre do seguinte modo, a ferramenta permite ao usuário determinar inicialmente se deseja criar um AFD ou AFND, em seguida é só começar a criar os estados e suas transições. O estado inicial do autômato é sempre criado automaticamente, já o estado final como pode existir mais de um, houve a necessidade de inserir um *checkbox* em todos os estados para que fossem determinados quais deles eram estados finais.

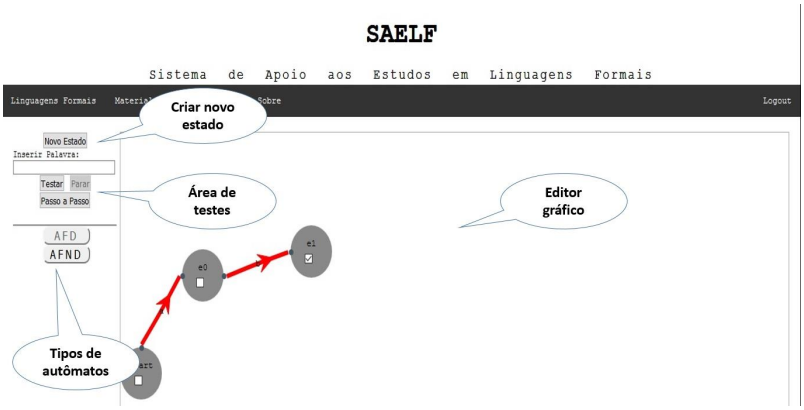
A Figura 23 mostra a tela de construção de autômatos do sistema, onde podemos visualizar o editor gráfico e outros componentes importantes do sistema, como: os tipos de autômatos e local de teste.

Outra característica importante do SAELF é como ele valida tanto um AFD quanto um AFND, a ferramenta permite ao usuário processar sua entrada de duas formas (Figura 24):

1. Validação passo a passo: o usuário determina o andar do processamento da entrada;
2. Validação direta: o usuário determina o resultado imediatamente.

Com esse tipo de interface o usuário poderá visualizar melhor como é feito

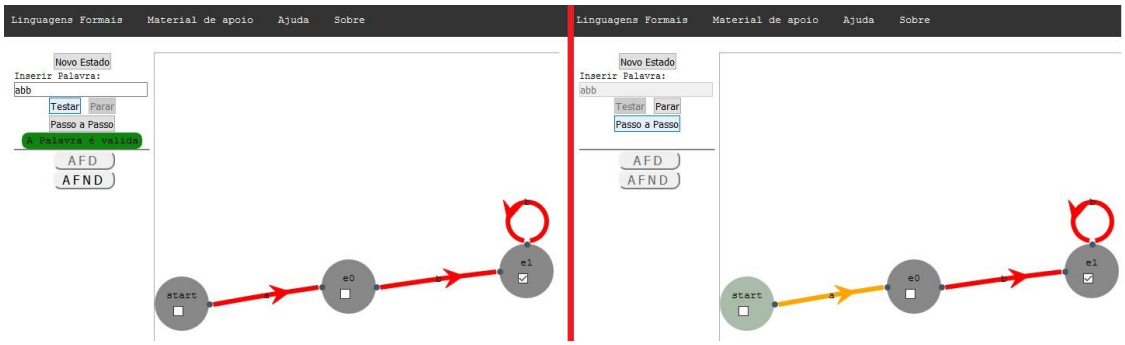
Figura 23 – Tela de construção de autômatos SAELF



Fonte: Próprio autor

um processamento de um AFD ou AFND, verificando se sua palavra de entrada foi reconhecida ou não pelo seu autômato.

Figura 24 – Tela de reconhecimento de autômatos SAELF



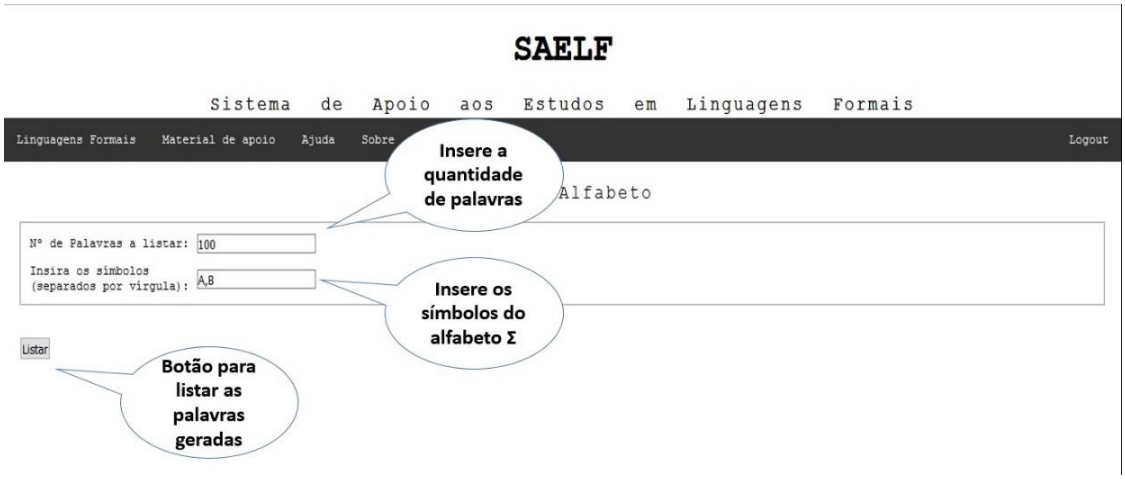
Fonte: Próprio autor

6.5 Criação de palavras

A Figura 25 demonstra o funcionamento da opção criar palavras detalhando os campos que nela existem. Na opção de menu criar palavras, em primeiro momento é solicitado ao usuário para inserir através do formulário para listagem das combinações de símbolos o número de palavras que ele deseja listar e os símbolos do alfabeto separados por vírgulas. Este formulário submete essas informações para o método de processamento *ProcessarAlfabeto* na classe *linguagensCtrl.class.php* esse método trata o número de palavras e os símbolos e os envia para a função *geraAlfabeto* na classe *alfabetoModel.class.php*. Essa classe trata recursivamente as possibilidades utilizando

a função *geraProximaPalavra* e retorna esse tratamento em uma variável para a classe de controle *linguagensCtrl.class.php* só então ela é utilizada na instância do arquivo *ListarAlfabeto.php* que está incluído na *view*.

Figura 25 – Tela geração de palavras do SAELF



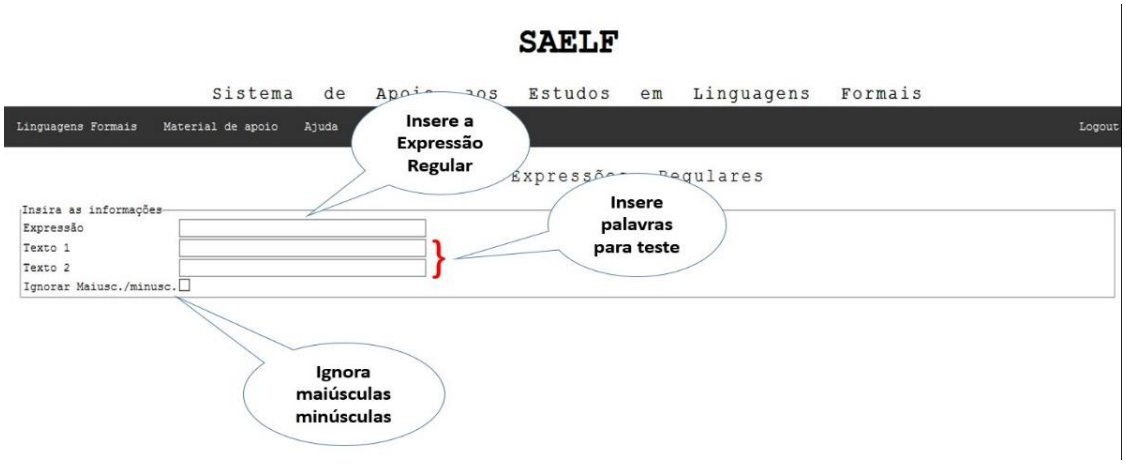
Fonte: Próprio autor

6.6 Teste de Expressões Regulares

Na opção de menu testar expressões regulares, em primeiro momento é solicitado ao usuário para inserir através do formulário de testes uma expressão regular e pode-se testar até duas palavras simultaneamente. Em JavaScript as expressões regulares são objetos, sendo utilizados os seguintes métodos *exec, test, match, replace e Split do objeto RegExp*[1]. Ao utilizar o construtor *RegExp* a compilação da Expressão Regular é realizada imediatamente. A Figura 26 demonstra o detalhamento da opção teste de expressões regulares com os seus campos.

[1]O construtor RegExp cria um objeto de expressão regular para realizar uma correspondência de texto com um padrão. <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/RegExp>

Figura 26 – Tela teste de expressões regulares do SAELF



Fonte: Próprio autor

A Figura 27 mostra um teste de expressão regular em execução e o reconhecimento de uma das palavras inseridas, neste exemplo foi inserida uma expressão regular que reconhece qualquer palavra que possua como subpalavra “abb”, portanto a primeira palavra do *box* Texto 1 ela foi reconhecida, pois atende o requisito de possuir “abb”, já a palavra do *box* Texto 2 não foi reconhecida.

Figura 27 – Tela de teste expressão regular SAELF

Fonte: Próprio autor

6.7 Material de apoio

No menu de material de apoio do sistema, foi disponibilizado para o usuário nove vídeos a respeito de linguagens formais, a maioria deles é de autoria do Professor Esdras Lins Bispo Junior o qual nos autorizou a inseri-los no sistema web, no Apêndice B é possível verificar a autorização que ele nos concedeu por email e listados na Tabela 9.

Tabela 9 – Lista de vídeos disponibilizados no sistema.

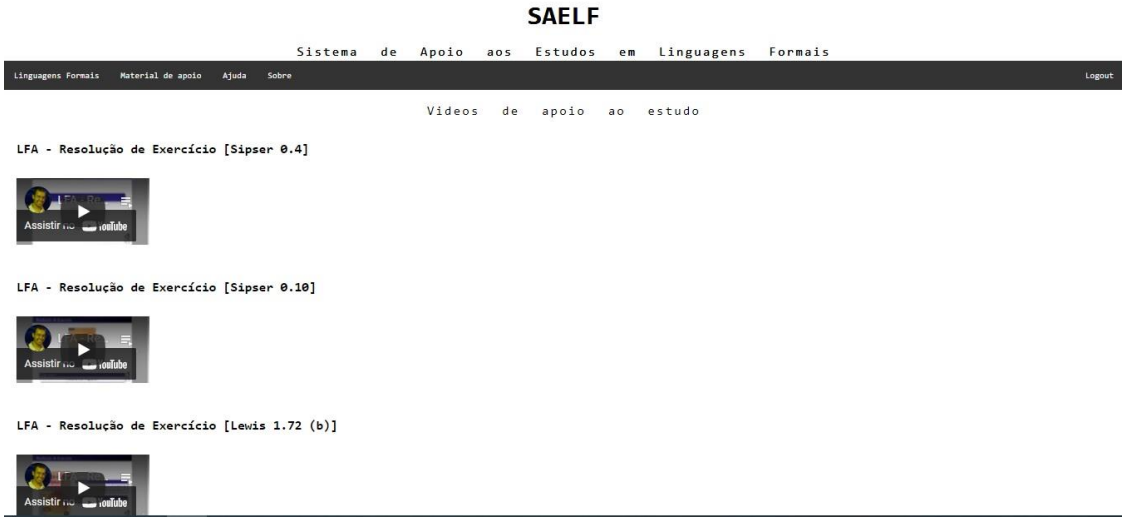
Vídeo	Lista de vídeos disponibilizados no sistema
1	LFA - Resolução de Exercício [Sipser 0.4] <https://youtu.be/53PKpTkyULo>
2	LFA - Resolução de Exercício [Sipser 0.10] Disponível em: <https://youtu.be/nsDYvu5Glek>
3	LFA - Resolução de Exercício [Lewis 1.72(b)] Disponível em: <https://youtu.be/_Smac5M3pMk>
4	LFA - Resolução de Exercício [Sipser 1.6 (b)] Disponível em: <https://youtu.be/CO-gBM2CbRo>
5	LFA - Resolução de Exercício [Bispo Jr 001] Disponível em: <https://youtu.be/vDzjLl1pvjQ>
6	LFA - Resolução de Exercício [Sipser 1.6 (l)] Disponível em: <https://youtu.be/ewcQAU_UpP4>
7	LFA - Resolução de Exercício [Bispo Jr 002] Disponível em: <https://youtu.be/sOB5VRRJa-s>
8	LFA - Resolução de Exercício [Queiroz 001] Disponível em: <https://youtu.be/LA8zGYfMnXE>
9	Autômatos Finitos Determinísticos e Não-Determinísticos Disponível em: <https://youtu.be/mCwQoM8KaZk>

Fonte: Próprio autor

Em sua grande maioria, os vídeos disponibilizados tratam de resoluções de exercícios do livro Introdução a Teoria da computação de Michael Sipser 2ª edição norte-americana e do livro Elementos de Teoria da Computação de Harry R. Lewis 2ª edição. A Figura 28 demonstra com mais detalhes a disposição dos vídeos no sistema web SAELF, onde foi utilizado a opção de incorporação de vídeos do YouTube que basicamente gera um código HTML denominado *EMBED*, com ele é possível assistir os vídeos diretamente no sistema, ou seja, sem a necessidade de ir para outro site.

A disposição dos usuários também está uma lista de exercícios selecionados do

Figura 28 – Disposição dos vídeos



Fonte: Próprio autor

livro de Introdução a Teoria da Computação de Michael Sipser e alguns foram adaptados facilitando ou dificultando o seu nível. Em cada um dos exercícios propostos existem um botão logo abaixo que permite visualizar uma imagem com uma solução possível do problema, onde foi utilizado o próprio SAELF para a resolução. Na Figura 29 pode-se ver como os exercícios ficam dispostos no Sistema.

Figura 29 – *Print Screen* da lista de exercícios no SAELF

Fonte: Próprio autor

7 ANÁLISE DOS DADOS

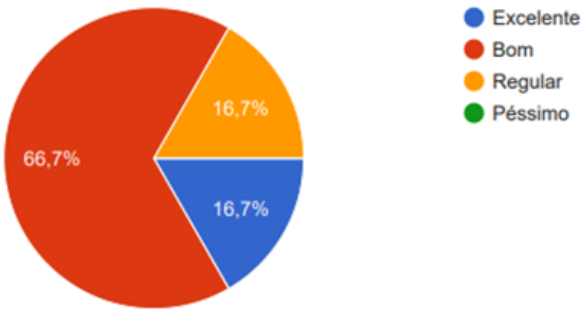
A avaliação de um sistema computacional é um processo contínuo e constante e sempre será possível adicionar melhorias. Após a conclusão de um sistema web educacional e sua disponibilização para o usuário é possível coletar dados ou efetuar pesquisas como uma forma de *feedback* destes usuários, a fim de fornecer ideias para melhorias ou até mesmo correção de erros que ainda podem persistir mesmo após a realização dos mais diversos testes a que são submetidos. Esse *feedback* também pode ser utilizado para fornecer melhorias nas próximas versões do sistema ou até mesmo criar funcionalidades. De forma geral, o sistema web "SAELF – Sistema de apoio aos estudos em linguagens formais" foi bem-conceituado pelos avaliadores, onde estes contribuíram com comentários de incentivo e positivos sobre o trabalho desenvolvido. Em cada etapa de avaliação diversos aspectos foram analisados através de um questionário de avaliação e de uma avaliação realizada pelo professor Esdras Lins Bispo Junior da Universidade Federal do Jataí (UFJ).

No questionário, diversos aspectos foram analisados por meio de itens classificados pelos avaliadores como: excelente, bom, regular e péssimo.

Participaram dessa avaliação alunos da Universidade Federal do Pampa, ao todo foram obtidas 6 respostas no questionário, aparentemente um número baixo de respostas, mas é possível ainda afirmar que está avaliação ocorreu fora do período em que a disciplina foi ofertada em 2019, portanto, agora existiam alguns alunos que tinham conhecimento da disciplina e diversos que ainda não tinham conhecimento, impossibilitando dessa forma participar de forma efetiva da avaliação. Segundo o ranking da Unipampa Campus Bagé de 2019, foram listados 190 alunos do curso de engenharia de computação (BAEC) como população, assim uma amostra de 6 pessoas com um intervalo de confiança de 95% gera uma margem de erro de 40% na pesquisa, utilizando a fórmula de Slovin.

A fim de iniciar a apresentação dos resultados obtidos, foi perguntado aos avaliadores qual a sua opinião "quanto a forma de login do sistema SAELF" e obteve-se 4 respostas com "bom" totalizando 66,7%, 1 resposta com "excelente" e uma resposta com "regular" totalizando assim 16,7% cada. A Figura 30 ilustra este percentual de respostas.

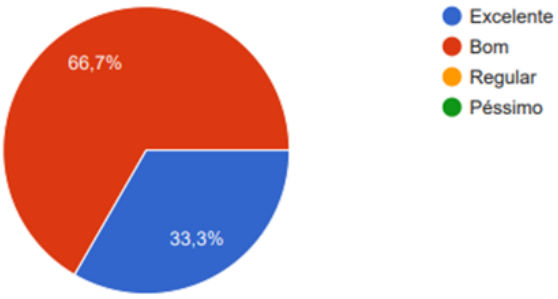
Figura 30 – Gráfico de avaliação de login do SAELF



Fonte: Próprio autor

Em relação à segunda pergunta do questionário que era referente “Quanto a forma de recuperação de senha do sistema SAELF”, obteve-se 2 respostas com “Excelente” e 4 respostas com “Bom” totalizando assim 33,3% e 66, 7% das respostas respectivamente. A Figura 31 ilustra este percentual de respostas.

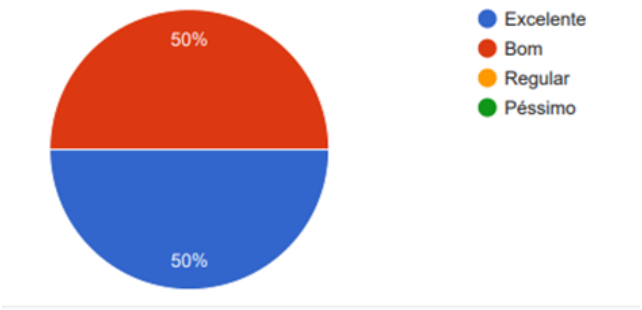
Figura 31 – Gráfico de avaliação da recuperação de senha



Fonte: Próprio autor

Também foi perguntado aos avaliadores “quanto a forma de disposição dos menus do sistema” e a resposta dividiu as opiniões entre “Excelente” e ”Bom” totalizando assim 50% para cada resposta. A Figura 32 ilustra este percentual de respostas.

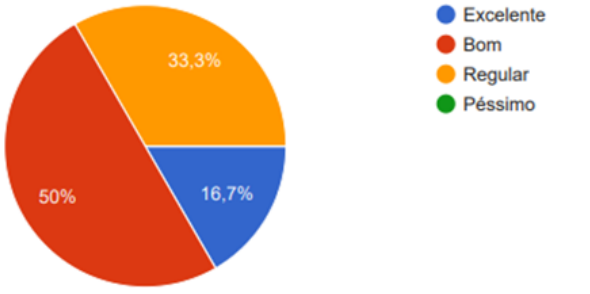
Figura 32 – Gráfico de avaliação da disposição dos menus



Fonte: Próprio autor

Em relação ao questionamento sobre o "Módulo de geração de palavras", obteve-se 3 respostas que consideraram o módulo de geração de palavras "Bom" totalizando 50% das respostas, 2 avaliadores acharam "regular" com 33,3% do percentual e uma resposta ficou com "Excelente" ficando com 16,7% do total. A Figura 33 ilustra este percentual de respostas.

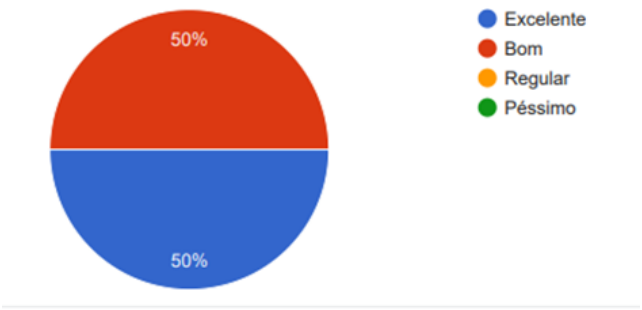
Figura 33 – Gráfico de avaliação do módulo de geração de palavras



Fonte: Próprio autor

Na utilização do "módulo de geração de palavras", 3 pessoas disseram ser "Bom" e 3 pessoas disseram ser "Excelente" totalizando 50% para cada uma das respostas. A Figura 34 ilustra este percentual de respostas.

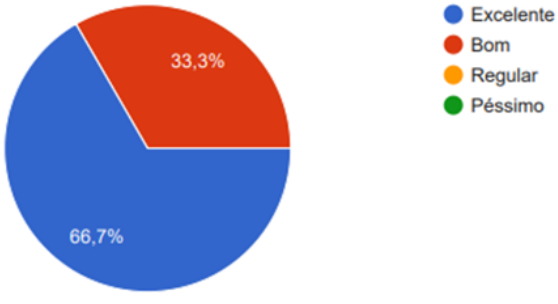
Figura 34 – Gráfico de avaliação da utilização de geração de palavras



Fonte: Próprio autor

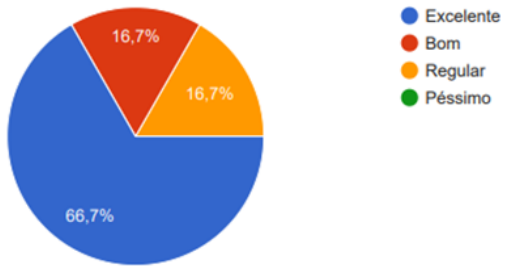
Segundo os dados obtidos pode-se observar que a respeito do módulo de Expressões regulares 4 avaliadores optaram por “Excelente” e 2 optaram por “bom”, sendo assim a porcentagem ficou 66,7% e 33,3%, esta ilustração pode ser observada na Figura 35. Sobre a utilização do módulo, foram 4 respostas para “Excelente”, 1 resposta para “Bom” e 1 resposta para regular, sendo assim as porcentagens de cada resposta ficou em 66,7%, 16,7% e 16,7% respectivamente para cada uma das respostas. A Figura 36 ilustra este percentual de respostas.

Figura 35 – Gráfico de avaliação do módulo de expressão regular



Fonte: Próprio autor

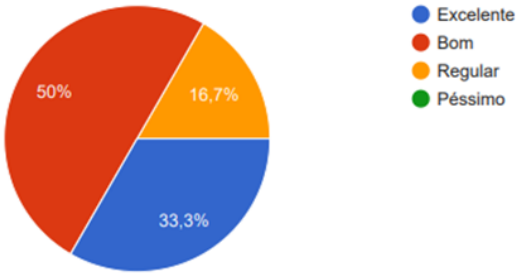
Figura 36 – Gráfico de avaliação da utilização do módulo de expressão regular



Fonte: Próprio autor

A questão a seguir era referente ao módulo de autômatos finitos e as respostas foram, 2 opções com “excelente” e percentual de 33,3%, 3 opções com “Bom” e percentual de 50% das respostas e por fim 1 resposta “Regular” com 16,7%. A Figura 37 ilustra esse percentual de respostas.

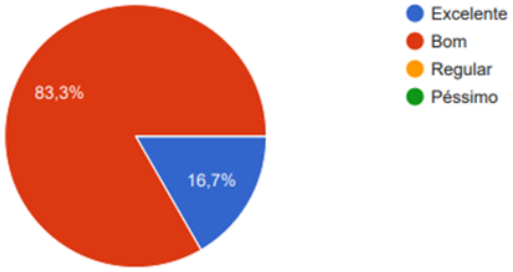
Figura 37 – Gráfico de avaliação do módulo de autômatos finitos



Fonte: Próprio autor

Quanto a utilização do módulo de autômatos finitos determinísticos, a avaliação ficou com 5 respostas na opção “Bom” e 1 resposta na opção “Excelente” e o percentual de 83,3% para “Bom” e 16,7% para “Excelente”. A Figura 38 ilustra esse percentual de respostas.

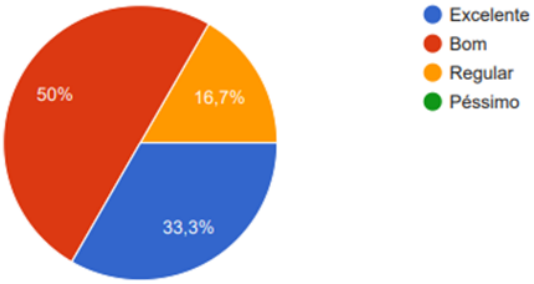
Figura 38 – Gráfico de avaliação do módulo de AFD



Fonte: Próprio autor

Também havia uma pergunta a quanto a utilização do módulo de autômatos finitos não determinísticos onde nas respostas 2 avaliadores optaram por “Excelente” com 33,3% do resultado, 3 optaram por “Bom” com 50% do resultado e 1 optou por “regular” com 16,7% do percentual total de respostas. A Figura 39 ilustra esse percentual de respostas.

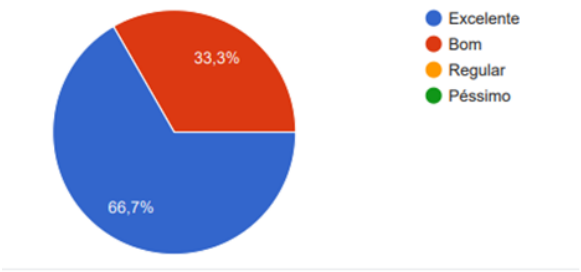
Figura 39 – Gráfico de avaliação do módulo de AFND



Fonte: Próprio autor

A respeito do material de apoio inserido no sistema web atingimos 100% das respostas avaliando como “Bom”. Nessa mesma área ainda perguntou a respeitos das vídeo aulas do professor Esdras Lins Bispo Junior que estavam em seu canal do YouTube sendo incorporadas no sistema, nesse quesito as respostas foram, 4 opções com “Excelente” 66,7% das respostas e 2 opções com “Bom” sendo 33,3% das respostas. A Figura 40 ilustra esse percentual de respostas.

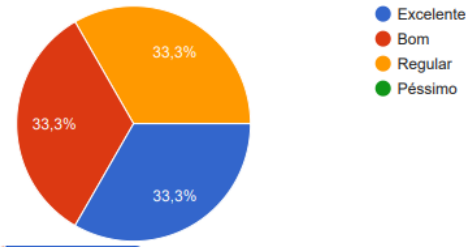
Figura 40 – Gráfico de avaliação das vídeo aulas



Fonte: Próprio autor

Também foram inseridos alguns exercícios no sistema e a avaliação ficou com 2 respostas para “Excelente”, 2 respostas para “Bom” e duas respostas para ”Regular” sendo assim o percentual ficou em 33,3% para cada. A Figura 41 ilustra esse percentual de respostas.

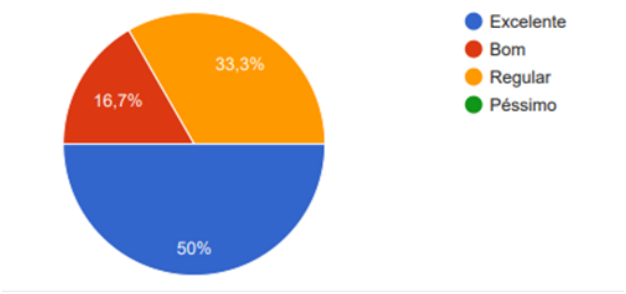
Figura 41 – Gráfico de avaliação dos exercícios de autômatos



Fonte: Próprio autor

Por fim, as duas últimas perguntas ficaram a respeito da clareza do módulo de ajuda do sistema web e obtivemos 3 respostas para “Excelente” com 50%, 2 respostas para “Regular” com 33,7% e uma resposta para “Bom” com 16,7% do total de respostas. A Figura 42 ilustra esse percentual de respostas.

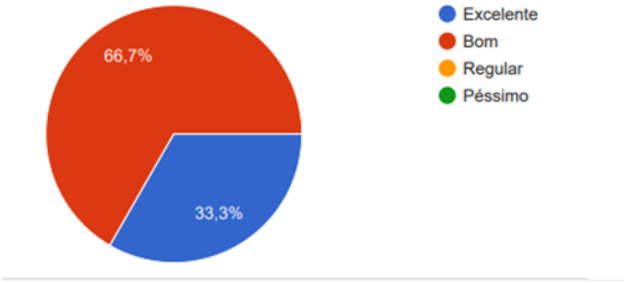
Figura 42 – Gráfico de avaliação do módulo de ajuda



Fonte: Próprio autor

Em nossa última pergunta foi sobre os vídeos tutoriais de ajuda e 4 avaliadores optaram por “Bom” e 2 optaram por “Excelente” e o percentual ficou em 66,7% e 33,3% para cada uma das respostas. A Figura 43 ilustra esse percentual de respostas.

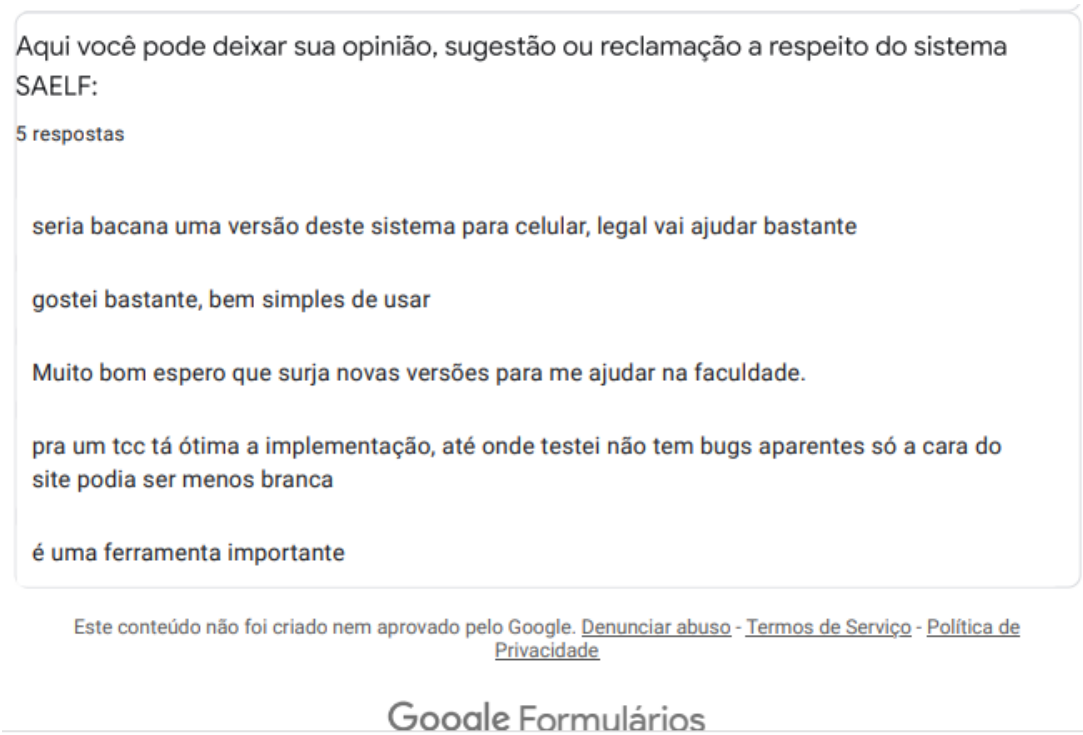
Figura 43 – Gráfico de avaliação dos vídeos tutoriais



Fonte: Próprio autor

Na Figura 44 estão elencadas as considerações que os avaliadores responderam no questionário de avaliação do SAELF.

Figura 44 – Considerações dos Avaliadores



Fonte: Próprio autor

O referido sistema também foi avaliado pelo professor Esdras Lins Bispo Jr.do Departamento de Ciência da Computação na Universidade Federal de Goiás (UFG) – Jataí Regional.

A avaliação foi realizada em 13/05/2019, neste documento estão descritas todas as etapas de avaliação.

O professor realizou uma avaliação bem detalhada e criteriosa de cada um dos módulos do sistema web, onde achou o módulo de expressões regulares muito bom e achou importante receber o *feedback* imediato se a palavra pertence ou não a linguagem regular gerada, inclusive foi sugerido por ele uma alteração no rótulo “texto 1” por “Cadeia” que foi aceita e alterada.

Na avaliação do módulo de autômatos finitos ele concluiu estar bastante intuitivo e durante a criação dos autômatos é de fácil utilização, porém foi difícil de perceber ser necessário clicar no rótulo para gerar a seta de transição, mas com o apoio do vídeo tutorial foi possível entender.

O módulo do material de apoio é importante e interessante possuir as vídeo aulas, como sugestão de usabilidade foi sugerido que os vídeos ficassem como uma lista de tópicos como exemplo ele citou a própria lista de busca do YouTube, julgou ser mais fácil

e agradável à visualização e também seria possível ver um resumo do conteúdo do vídeo.

Na avaliação dos vídeos tutoriais de ajuda a respeito de expressões regulares e autômatos finitos achou bastante interessante e direto ao ponto, fez algumas sugestões a respeito de edição de vídeos. Na aba “Sobre” do sistema web ele sugeriu que fosse inserida após a defesa uma cópia do texto da monografia, além de agradecimentos.

De modo geral, a opinião dele foi que o SAELF é uma importante ferramenta para auxiliar no ensino aprendizagem de linguagens formais.

8 CONSIDERAÇÕES FINAIS

A disciplina de linguagens formais possui um conteúdo muito técnico, resultando em uma disciplina complexa, dessa forma, a ferramenta se mostrou viável para auxiliar os alunos, onde Terra (2015) afirmou ter um índice de reprovação de aproximadamente 50%.

No caso específico deste trabalho, onde o principal objetivo era desenvolver sistema online para auxiliar os alunos dos cursos na área da informática, foi fundamental garantir que a experiência dos usuários, como, por exemplo, garantir uma acessibilidade multinavegadores, sendo elencada no topo da lista de importâncias e garantir que as ferramentas e técnicas sejam corretamente utilizadas para o sucesso. O sistema SAELF foi desenvolvido para ser utilizado totalmente online, de forma que com acesso à internet e com um computador os alunos possam continuar a estudar em qualquer lugar que estiverem. Além disso, é possível, que seja utilizado por professores ou monitores da disciplina de linguagens formais como uma ferramenta de avaliação ou de diagnóstico de dificuldades.

A importância de fazer um planejamento com fatores que compõem suas etapas como um levantamento de requisitos, a montagem de um cronograma bem definindo para a conclusão dos mesmos. Reservar uma parcela boa de tempo para pesquisar ferramentas e tecnologias para desenvolver uma aplicação web foi fundamental. A utilização do design instrucional foi importante na criação dos materiais para facilitar o ensino aprendizagem e se mostrou útil durante o desenvolvimento, implementação e avaliação que são os seus três pilares. Durante o desenvolvimento da pesquisa utilizou-se o DSR que é um design de implementação focado na computação, utilizado enquanto se faz a pesquisa e com desenvolvimento, onde contribuiu significativamente na construção e projeto do sistema web SAELF.

Com a realização deste trabalho foi possível colocar em prática os conceitos adquiridos durante a graduação, e a realização do processo de desenvolvimento de um software com o potencial de auxiliar os estudantes de linguagens formais e reduzir o índice de reprovação.

8.1 Trabalhos futuros

Apesar de adequada, a solução ao problema tem alguns pontos que ainda podem ser melhorados e novas tecnologias adicionadas, a fim de incrementar, atualizar ou até mesmo dar um upgrade no sistema, principalmente na interface porque neste trabalho o foco foi na parte lógica do sistema, então podemos dizer que a parte estética da interface poderia ser mais atrativa.

Seguindo a linha de pesquisa do projeto proposto neste trabalho, podem ser adicionados novos módulos como, por exemplo, um simulador de máquina de Turing e máquinas de pilha, além de ser possível trabalhar com gramáticas regulares, utilizando o mesmo tipo de *framework* utilizado neste trabalho. Ademais pode se fazer uma melhoria no *front-end* do sistema, que nada mais é que a parte do sistema que os usuários podem ver e interagir, como a interface gráfica do usuário (GUI) incluindo o design, menus de navegação, textos, imagens, etc.

Outro possível trabalho futuro pode ser uma versão mobile do sistema SAELF, para que o usuário tivesse a oportunidade de utilizar o sistema em qualquer lugar onde estivesse, já que hoje em dia praticamente todos tem acesso a um *smartphone*.

REFERÊNCIAS

BOEHM, B. W. A spiral model of software development and enhancement. **Computer**, IEEE, v. 21, n. 5, p. 61–72, 1988. Disponível em: <https://ieeexplore.ieee.org/abstract/document/59/.>

CARVALHO, F. E. A.; JUNIOR, M. M. C.; COSTA, Y. M. Jogos educativos no ensino de autômato finito determinístico: Um estudo de caso com o jogo a factory disaster. In: SBC. **Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital**. [S.l.], 2021. p. 472–478.

CHADBOURNE, B. C.; SANDERS, B. To the heart of risk management: teaching project teams to combat risk. In: **the 30th Annual Project Management Institute 1999 Seminar & Symposium**. [S.l.: s.n.], 1999.

DRESCH, A.; LACERDA, D. P.; JÚNIOR, J. A. V. A. **Design science research: método de pesquisa para avanço da ciência e tecnologia**. [S.l.]: Bookman Editora, 2015.

FILATRO, A. C. **Learning design como fundamentação teórico-prática para o design instrucional contextualizado.** Tese (Doutorado) — Universidade de São Paulo, 2008.

FILATRO, A. C.; PICONEZ, S. Planejamento, design, implementação e avaliação de programas de educação on-line. **Curitiba: Escola de Governo do Paraná**, 2007.

FURTADO, O. J. V. **Linguagens Formais e Compiladores**. Universidade Federal de Santa Catarina, Centro Tecnol̸gico, Departamento de Inform´tica e de Estat´stica, 2007. Acesso em: 21 mai. 2017. Disponível em: <https://www.ime.usp.br/~jef/tc_gramaticas.pdf.>

HEVNER, A.; CHATTERJEE, S. Design science research in information systems. In: ____. **Design Research in Information Systems: Theory and Practice**. Boston, MA: Springer US, 2010. p. 9–22. ISBN 978-1-4419-5653-8. Disponível em: <https://doi.org/10.1007/978-1-4419-5653-8_2.>

HEVNER, A. R. A three cycle view of design science research. **Scandinavian journal of information systems**, v. 19, n. 2, p. 4, 2007. Disponível em: <https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1017&context=sjis.>

KATAKURA, G. T. Ferramenta de apoio ao ensino interativo de expressões regulares. 2018. Acesso em: 06 jul. 2022. Disponível em: <http://dsc.inf.furb.br/arquivos/tccs/monografias/2018_1_gabriel-takashi-katakura_monografia.pdf.>

KURT, S. **ADDIE Model: Instructional Design**. 2018. Acesso em: 20 mai. 2019. Disponível em: <https://educationaltechnology.net/the-addie-model-instructional-design/.>

MACHADO, C. A. F. **A-Risk: um método para identificar e quantificar risco de prazo em projetos de desenvolvimento de software**. 239 p. Dissertação (Mestrado) — Pontifícia Universidade Católica do Paraná, Curitiba, 2002. Acesso em: 06 jul. 2022. Disponível em: <https://www.ppgia.pucpr.br/pt/arquivos/mestrado/dissertacoes/2002/cristina_filipak_2002.pdf.>

MENEZES, P. B. **Linguagens formais e autômatos**. 5. ed. [S.l.]: Sagra-Dcluzzato, 2000.

MINETTO, E. L. **Frameworks para Desenvolvimento em PHP**. [S.l.]: Novatec Editora, 2007. 192 p.

NOGUEIRA, M.; ABE, J. M. Normas e modelos de qualidade como política de produção de software no contexto brasileiro. **XV SIMPEP Simpósio Engenharia de Produção, Bauru-SP**, 2008.

OLIVEIRA, P. V. N. **LFA Virtual–Uma ferramenta de ensino para a disciplina de linguagens formais e autômatos**. 2009. Acesso em: 14 mai. 2017. Disponível em: <http://www.bcc.ufla.br/wp-content/uploads/2013/2009/LFA_Virtual-Uma_Ferramenta_de_Ensino_para_a_Disciplina_de_Linguagens_formais_e_Autonomatos.pdf.>

PALAZZO, L. A. M. **Expressões Regulares e Gramáticas Regulares**. 2007. 7 p. Acesso em: 20 mai. 2017. Disponível em: <http://infocat.ucpel.tche.br/disc/lfa/docs/LFA-T02.pdf.>

PEREIRA, M. H. R. **AngularJS: Uma abordagem prática e objetiva**. [S.l.]: Novatec Editora, 2014. 208 p.

PIMENTEL, M.; FILIPPO, D.; SANTOS, T. M. Design science research: pesquisa científica atrelada ao design de artefatos. **RE@ D-Revista de Educação a Distância e eLearning**, v. 3, n. 1, p. 37–61, 2020.

RAMOS, M. V. M. Ensino de linguagens formais e autômatos em cursos superiores de computação. **CEP**, Universidade Federal do Vale do São Francisco (UNIVASF), v. 48902, p. 13, 2009. Acesso em: 10 jul. 2022.

REINALDO, F. A. et al. Representação gramatical nebulosa livre de contexto. p. 11, 2003. Acesso em: 19 mai. 2017. Disponível em: <https://www.researchgate.net/profile/Francisco-Reinaldo/publication/276202921_Representacao_Gramatical_Nebulosa_Livre_de_Contexto/links/5551f91e08ae6943a86d67f0/Representacao-Gramatical-Nebulosa-Livre-de-Contexto.pdf.>

RIBEIRO JUNIOR, N. G. **Ambiente para Auxilio ao Ensino de Linguagens Formais e Autômatos**. 2007. Acesso em: 20 jul. 2017. Disponível em: <www.simuladordeautomatos.com.br.>

ROMISZOWSKI, A.; ROMISZOWSKI, L. P. Retrospectiva e perspectivas do design instrucional e educação a distância: análise da literatura. **Revista Brasileira de Aprendizagem Aberta e a Distância**, v. 4, p. 46, 2005. Disponível em: <https://doi.org/10.17143/rbaad.v4i0.168.>

SANTOS, E.; COELHO, R. C. Obtaining l-systems rules from strings. EDGRAF, p. 15, 2010. Acesso em: 18 mai. 2017. Disponível em: <https://seer.furg.br/vetor/article/view/1345/1374.>

SEIDLER, K. O. "about the xampp project". 2014. Acesso em: 20 mai. 2018. Disponível em: <https://www.apachefriends.org/about.html.>

SILVA, R. C. **Apostila de Teoria da computação**. 2007. Acesso em: 18 mai. 2017. Disponível em: <www.dainf.ct.utfpr.edu.br/~fabro/LFA/TeoriaComputacaoNovo.pdf.>

SIMON, H. A. Cognitive science: The newest science of the artificial. **Cognitive science**, Elsevier, v. 4, n. 1, p. 33–46, 1980. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0364021381800031.>

SOMMERVILLE, I. **Engenharia de software**. 8. ed. São Paulo: Pearson Addison-Wesley, 2007.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TERRA, R. **Dados da disciplina de Linguagens Formais e Autômatos**. [S.l.], 2015. 547 p. Disponível em: <http://professores.dcc.ufla.br/~terra/public_files/2015_apostila_lfa.pdf.>

VIEIRA, N. J. **Introdução aos fundamentos da computação: linguagens e máquinas**. 1. ed. [S.l.]: Pioneira Thomson Learning, 2006. 334 p.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. [S.l.]: CAMPUS, 2014. v. 2. 168 p.

APÊNDICE A – DOCUMENTO DE REQUISITOS

A.1 Introdução

Este documento especifica os requisitos do SAELF Sistema Aprimoramento Educacional em Linguagens Formais para o apoio do ensino da disciplina de Linguagens Formais, fornecendo aos desenvolvedores as informações necessárias para o projeto de implementação, assim como a realização de testes.

Visão geral do documento

Além desta seção introdutória, as seções seguintes estão organizadas como descrito abaixo.

Seção A.2 – Descrição geral do sistema: apresenta uma visão geral do sistema, caracterizando qual é o seu escopo e descrevendo seus usuários.

Seção A.3 – Requisitos funcionais: relacionam a maneira de como o sistema deve operar, onde se especificam as entradas e saídas do sistema.

Seção A.4 – Requisitos não-funcionais: especifica todos os requisitos não funcionais do sistema, divididos em requisitos de usabilidade, confiabilidade, desempenho, segurança, distribuição, adequação a padrões e requisitos de hardware e software.

Seção A.5 – Referências: apresenta referências para outros documentos utilizados para a confecção deste documento.

Identificação dos requisitos

Para uma correta interpretação deste documento é pertinente o conhecimento de algumas convenções e termos específicos, descritos a seguir. Por convenção, a referência a requisitos é feita através do nome da subseção onde eles estão descritos, seguidos do identificador do requisito, conforme a especificação a seguir: [nome da subseção. Identificador do requisito].

Prioridades dos requisitos

Para estabelecer a prioridade dos requisitos, na seção 3, foram adotadas as denominações “essencial”, “importante” e “desejável”.

Essencial é o requisito sem o qual o sistema não entra em funcionamento.

Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.

Importante é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.

Desejável é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele.

Requisitos desejáveis podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

A.2 Descrição geral do sistema

SAELF é um sistema *online* que será disponibilizado para o apoio de estudos aos discentes na área de informática. Ele possuirá teste de expressões regulares, geração de palavras e construção de autômatos finitos.

Abrangência do sistema O sistema *online* SAELF contribuirá para o ensino da disciplina de Linguagens Formais aos discentes de cursos na área de informática e de Engenharia de Computação.

A.3 Requisitos Funcionais

Esta seção compõe a lista de requisitos funcionais do sistema SAELF.

A.3.1 Utilização

Requisito 1: [RF01] Teste de expressões regulares

- **Descrição:** Permite ao usuário inserir e testar suas expressões.
- **Entradas:** teclado e mouse;
- **Processo:** A função receberá uma expressão e suas palavras a serem testadas e executará;
- **Saída:** Reconhecimento ou não das palavras inseridas.

Prioridade:

X	Essencial		Importante		Desejável

Requisito 2: [RF02] Sistema de autômatos finitos.

- **Descrição:** O *script* que permite ao usuário construir autômatos.
- **Entradas:** Mouse para inserir novos estados e suas transições, teclado para inserir os símbolos das transições e inserir palavras.
- **Processo:** São exibidas opções como inserir um novo estado, inserir transição, teste completo e teste passo a passo.
- **Saída:** Reconhecimento ou não da palavra inserida.

Prioridade:

X	Essencial		Importante		Desejável

Requisito 3: [RF03] Sistema de autômatos finitos não determinísticos.

- **Descrição:** O *script* que permite ao usuário construir autômatos não determinísticos.
- **Entradas:** Mouse para inserir novos estados e suas transições, teclado para
- inserir os símbolos das transições permitindo também deixar transições sem
- símbolos ou (ε) e inserir palavras.
- **Processo:** São exibidas opções como inserir um novo estado, inserir
- transição, teste completo e teste passo a passo.
- **Saída:** Reconhecimento ou não da palavra inserida.

Prioridade:

X	Essencial		Importante		Desejável

Requisito4: [RF04] Geração de palavras.

- **Descrição:** Sistema que permite ao usuário gerar combinações de símbolos.
- **Entradas:** teclado;
- Processo: É exibido um menu com duas opções: quantidade de palavras e símbolos do alfabeto.
- **Saída:** A quantidade de combinações de símbolos solicitada.

Prioridade:

X	Essencial		Importante		Desejável

Requisito 5: [RF05] Serialização.

- **Descrição:** *Script* que permite ao usuário salvar o progresso do seu autômato.
- **Entradas:** Mouse e teclado;
- **Processo:** É exibido um botão com a opção salvar e uma janela será aberta para inserir um nome.
- **Saída:** Autômato salvo no banco de dados.

Prioridade:

X	Essencial		Importante		Desejável

Requisito 6: [RF06] Deserialização.

- **Descrição:** *Script* que permite ao usuário retomar o progresso de seu autômato.
- **Entradas:** Mouse e teclado;
- **Processo:** É exibido um botão com a opção abrir e uma janela será aberta para escolher o autômato a ser reconstruído.
- **Saída:** Reconstrução do autômato.

Prioridade:

X	Essencial		Importante		Desejável

A.4 Requisitos Não Funcionais

Esta seção compõe a lista de requisitos não funcionais do sistema SAELF.

Requisito 1: [RNF01] Usabilidade

- A *interface* tem vital importância para o sucesso do sistema. O *layout* será amigável, simples e de fácil utilização.

Prioridade:

X	Essencial		Importante		Desejável

Requisito 2: [RNF02] Compatibilidade

- Requisito fundamental para que o sistema possa ser executado em diversos tipos de navegadores.

Prioridade:

	Essencial	X	Importante		Desejável

Requisito 3: [RNF03] Segurança

- Característica fundamental em um sistema online que exige usuário e senha para sua utilização.

Prioridade:

	Essencial		Importante	X	Desejável

Requisito 4: [RNF04] Tipo de interface

- O sistema só poderá ser acessado online via HTTP ou HTTPS.

Prioridade:

	Essencial		Importante	X	Desejável

Requisito 5: [RNF05] Necessidades de internacionalização

- O sistema poderá ter opções para ser acessado em idiomas como inglês e espanhol, mas o idioma principal será o português.

Prioridade:

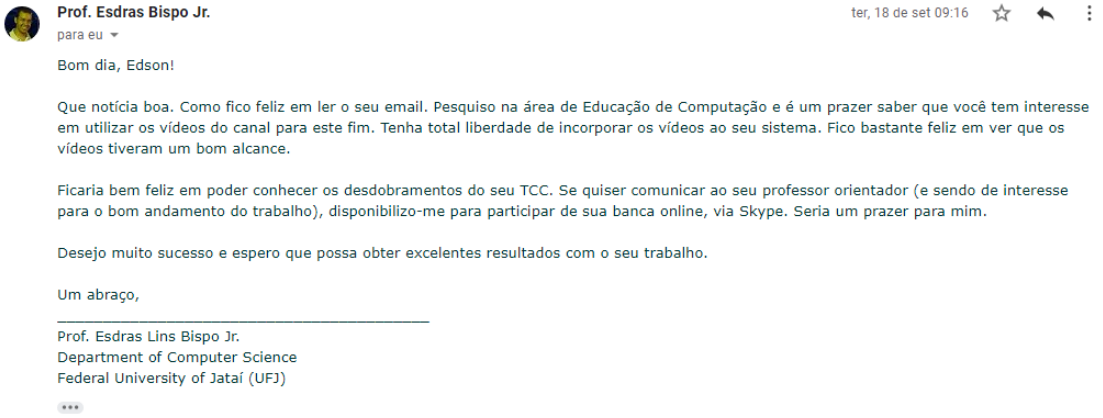
	Essencial		Importante	X	Desejável

A.5 Referências Bibliográficas

SOMMERVILLE, I. **Engenharia de software**. 8. ed. São Paulo: Pearson Addison-Wesley, 2007.

PRESSMAN, R. S; **Software Engineering: A Practitioner's Approach**, 7 ed., Mc Graw Hill, 2010

APÊNDICE B – EMAIL DE AUTORIZAÇÃO INCORPORAÇÃO DE VÍDEOS



Prof. Esdras Bispo Jr. ter, 18 de set 09:16

para eu

Bom dia, Edson!

Que notícia boa. Como fico feliz em ler o seu email. Pesquiso na área de Educação de Computação e é um prazer saber que você tem interesse em utilizar os vídeos do canal para este fim. Tenha total liberdade de incorporar os vídeos ao seu sistema. Fico bastante feliz em ver que os vídeos tiveram um bom alcance.

Ficaria bem feliz em poder conhecer os desdobramentos do seu TCC. Se quiser comunicar ao seu professor orientador (e sendo de interesse para o bom andamento do trabalho), disponibilizo-me para participar de sua banca online, via Skype. Seria um prazer para mim.

Desejo muito sucesso e espero que possa obter excelentes resultados com o seu trabalho.

Um abraço,

Prof. Esdras Lins Bispo Jr.
Department of Computer Science
Federal University of Jataí (UFJ)

Fonte: Próprio autor

APÊNDICE C – EMAIL DE AVALIAÇÃO DO SISTEMA SAELF

Gmail **Edson Camargo <edinho.camargo1@gmail.com>**

Consulta Acadêmica

Prof. Esdras Bispo Jr. <esdraspiano@gmail.com> 13 de maio de 2019 15:25
Para: Edson Camargo <edinho.camargo1@gmail.com>

Olá Edson, tudo joia?

Envio para você as minhas primeiras impressões do sistema. Achei muito legal a ferramenta como um todo. É um bom recurso para ser utilizado no processo de ensino e aprendizagem de Linguagens Formais.

Tentei dar um retorno a cada bloco de links da sua página, joia? No fim, faço uma sugestão de avaliação mais padronizada.

==========
Linguagens Formais
==========

Gerar palavras/alfabeto
Não entendi o propósito desta página em termos de funcionalidade.

Expressão Regular
A página está bem legal. E é bem interessante poder receber o feedback imediato para saber se a palavra pertence ou não à linguagem regular gerada.
Como sugestão, eu substituiria o rótulo "Texto 1" por "Cadeia".

Autômatos Finitos
Criação de AFDs
Uso do AFD pronto: Bastante intuitivo
Criação do AFD:
- Fácil de criar um novo estado
- Difícil de perceber que é necessário clicar no rótulo para gerar a seta de transição

==========
Material de Apoio
==========

Vídeos
Legal ver os vídeos na página. Muito massa.
Eu só sugeriria, em termos de usabilidade, que os vídeos ficassem como uma lista de tópicos (tipo o resultado de busca do YouTube).
É mais fácil e eu julgo ser mais agradável de visualizar. E ainda seria possível ter um resumo do conteúdo do vídeo.

Exercícios
Muito legal os exercícios. E bem interessante utilizar a sua própria ferramenta para mostrar a solução. Muito joia!
Acredito que seja necessário deixar explícito qual é o alfabeto de referência para cada uma das linguagens. É necessário pois para a criação do AFD exige-se que cada estado dê destino obrigatório para todos os símbolos do alfabeto da linguagem.

Exercício 7
Por "cadeia de "0" e "1", não fica claro se a cadeia vazia é permitida. Eu só sei deste aspecto porque você põe a ER ao lado. Talvez fosse importante reescrever o exercício para não gerar dúvida no estudante.

Exercício 9
Você está utilizando um AFD ou um AFN na resolução? O estado e3 não dá destino para nenhum símbolo do alfabeto. Pelo que percebi, a tela capturada não apresenta distinção nenhuma quando está

Fonte: Próprio autor

Gmail **Edson Camargo <edinho.camargo1@gmail.com>**

Consulta Acadêmica

Prof. Esdras Bispo Jr. <esdraspiano@gmail.com> 13 de maio de 2019 15:25
Para: Edson Camargo <edinho.camargo1@gmail.com>

Olá Edson, tudo joia?

Envio para você as minhas primeiras impressões do sistema. Achei muito legal a ferramenta como um todo. É um bom recurso para ser utilizado no processo de ensino e aprendizagem de Linguagens Formais.

Tentei dar um retorno a cada bloco de links da sua página, joia? No fim, faço uma sugestão de avaliação mais padronizada.

==========
Linguagens Formais
==========

Gerar palavras/alfabeto
Não entendi o propósito desta página em termos de funcionalidade.

Expressão Regular
A página está bem legal. E é bem interessante poder receber o feedback imediato para saber se a palavra pertence ou não à linguagem regular gerada.
Como sugestão, eu substituiria o rótulo "Texto 1" por "Cadeia".

Autômatos Finitos
Criação de AFDs
Uso do AFD pronto: Bastante intuitivo
Criação do AFD:
- Fácil de criar um novo estado
- Difícil de perceber que é necessário clicar no rótulo para gerar a seta de transição

==========
Material de Apoio
==========

Vídeos
Legal ver os vídeos na página. Muito massa.
Eu só sugeriria, em termos de usabilidade, que os vídeos ficassem como uma lista de tópicos (tipo o resultado de busca do YouTube).
É mais fácil e eu julgo ser mais agradável de visualizar. E ainda seria possível ter um resumo do conteúdo do vídeo.

Exercícios
Muito legal os exercícios. E bem interessante utilizar a sua própria ferramenta para mostrar a solução. Muito joia!
Acredito que seja necessário deixar explícito qual é o alfabeto de referência para cada uma das linguagens. É necessário pois para a criação do AFD exige-se que cada estado dê destino obrigatório para todos os símbolos do alfabeto da linguagem.

Exercício 7
Por "cadeia de "0" e "1", não fica claro se a cadeia vazia é permitida. Eu só sei deste aspecto porque você põe a ER ao lado. Talvez fosse importante reescrever o exercício para não gerar dúvida no estudante.

Exercício 9
Você está utilizando um AFD ou um AFN na resolução? O estado e3 não dá destino para nenhum símbolo do alfabeto. Pelo que percebi, a tela capturada não apresenta distinção nenhuma quando está

Fonte: Próprio autor

lidando com AFD ou AFN. Quando utilizo para a criação, é possível perceber isso, embora ainda não seja tão fácil de ver.

Apostilas
Pelo que vi, ainda não foram incluídas, né?

==========
Ajuda
==========

Tutorial Expressão Regular
Simples e direto. Muito Legal.
Sugestão. Poderia ser mais direto se fosse dividido em dois vídeos.
O primeiro mostrando diretamente como logar (se julgar necessário)
O segundo mostrando o uso da ER diretamente.
A vinheta poderia ser mais curta (10-15s)
Você poderia utilizar o "Loading" com a chamada principal
(se possível... sei que edição de vídeo é bem chatinho)

Tutorial Autômatos
Muito legal o tutorial. Bem claro.
Pelo que percebi, é necessário clicar no rótulo para gerar a transição? Ou é impressão minha, Edson? Um duplo clique não seria melhor?

==========
Sobre
==========

Seria legal, após a defesa, colocar o texto da sua monografia aqui, nesta seção. Também é comum colocar os agradecimentos nesta seção.

==========
Rodapé
==========

O rodapé ainda está com o ano de 2017.

==========
Sugestões para avaliação
==========

Não sei se você está pensando em fazer uma avaliação padronizada. Mas poderia utilizar o questionário de avaliação proposto por Vieira e colegas (2012).

VIEIRA, M. M. S.; SIMÕES, L. L. F.; BARRETO, A. L. O. AVALIAÇÃO DE SOFTWARE EDUCATIVO: ASPECTOS PEDAGÓGICOS E TÉCNICOS. In. Faculdade Cearense em Revista. Vol 5, n. 1, 2012.

Link: http://www.faculdadescearenses.edu.br/revista2/index.php/representantes/2013-03-15-16-49-17/85-avaliacao-de-software-educativo-aspectos-pedagogicos-e-tecnicos

==========

Posso fazer esta avaliação mais padronizada. Você deseja?
Como estão as suas datas? Tem época prevista para a defesa?

Um abraço,

Prof. Esdras Lins Bispo Jr.
Department of Computer Science
Federal University of Goiás (UFG) - Jataí Regional

[Texto das mensagens anteriores oculto]

Fonte: Próprio autor

APÊNDICE D – EMAIL DE CONFIRMAÇÃO CADASTRO DO SISTEMA SAELF

23/05/2022 12:53 Gmail - Bem vindo à Linguagens Formais!

Gmail

Edson Camargo <edinho.camargo1@gmail.com>

Bem vindo à Linguagens Formais!

webmaster@linguagensformais.com <webmaster@linguagensformais.com> 21 de maio de 2018 21:49
Para: edinho.camargo1@gmail.com

Bem vindo à pagina de linguagens formais.

https://mail.google.com/mail/u/0/?ik=db211a5c38&view=pt&search=all&permmsgid=msg-f%3A1601124096239103879&simpl=msg-f%3A1601124… 1/1

Fonte: Próprio autor

APÊNDICE E – QUESTIONÁRIO AVALIAÇÃO DO SISTEMA SAELF

Questionário de Avaliação do SAELF

Questionário de avaliação do SAELF - (Sistema de Apoio aos Estudos em Linguagens Formais)

Responda as questões objetivas como forma de avaliação do sistema SAELF

Obs: Ao responder este questionário você concorda com a utilização das respostas como instrumento de avaliação do presente trabalho.

*Obrigatório

1. Quanto à forma de realização de login? *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

2. Quanto à forma de recuperação de senha? *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

3. Quanto à disposição dos menus? *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

4. Modulo geração de palavras? *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

5. Utilização do módulo geração de palavras? *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

6. Módulo Teste de Expressões Regulares? *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

7. Utilização do Teste de Expressões regulares? *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

8. Módulo Autômatos finitos? *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

9. Utilização do módulo Autômatos finitos Determinísticos? *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

10. Utilização do módulo Autômatos finitos Não Determinísticos? *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

11. Material de apoio *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

12. Vídeo aulas a respeito de linguagens formais? *

Marcar apenas uma oval.

- Excelente
- Bom
- Regular
- Péssimo

13. Exercícios a respeito de autômatos finitos? *

Marcar apenas uma oval.

- Excelente
- Bom
- Regular
- Péssimo

14. Qualidade das apostilas? *

Marcar apenas uma oval.

- Excelente
- Bom
- Regular
- Péssimo

15. Clareza do módulo de ajuda? *

Marcar apenas uma oval.

- Excelente
- Bom
- Regular
- Péssimo

16. Clareza dos vídeos tutoriais de ajuda? *

Marcar apenas uma oval.

- ◯ Excelente
- ◯ Bom
- ◯ Regular
- ◯ Péssimo

17. Aqui você pode deixar sua opinião, sugestão ou reclamação a respeito do sistema SAELF:

APÊNDICE F – RESULTADO QUESTIONÁRIO AVALIAÇÃO DO SISTEMA SAELF

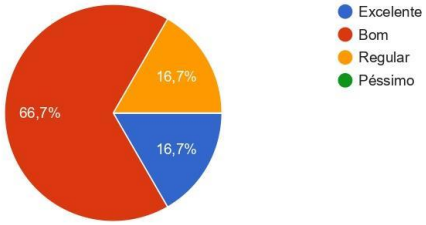
Questionário de Avaliação do SAELF

6 respostas

Publicar análise

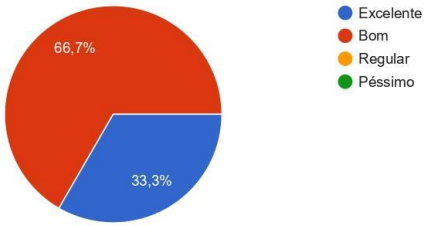
Quanto à forma de realização de login? Copiar

6 respostas



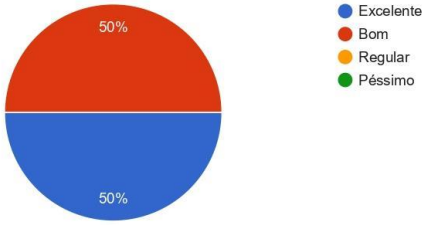
Quanto à forma de recuperação de senha? Copiar

6 respostas



Quanto à disposição dos menus? Copiar

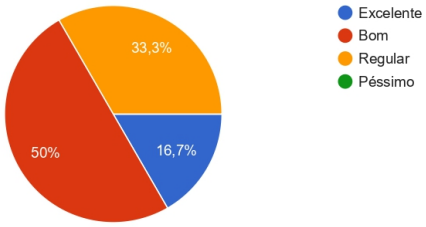
6 respostas



Modulo geração de palavras?

Copiar

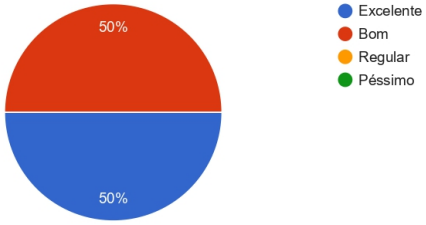
6 respostas



Utilização do módulo geração de palavras?

Copiar

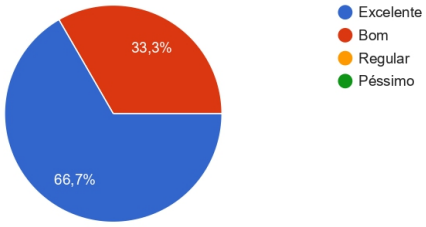
6 respostas



Módulo Teste de Expressões Regulares?

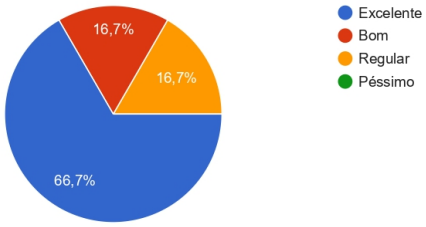
Copiar

6 respostas



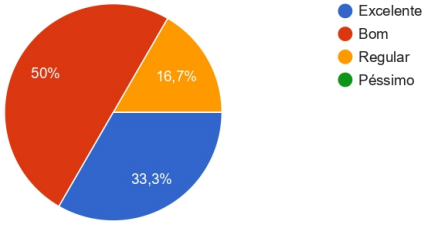
Utilização do Teste de Expressões regulares? Copiar

6 respostas



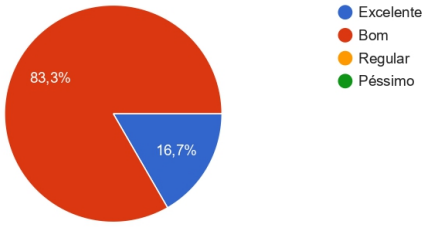
Módulo Autômatos finitos? Copiar

6 respostas



Utilização do módulo Autômatos finitos Determinísticos? Copiar

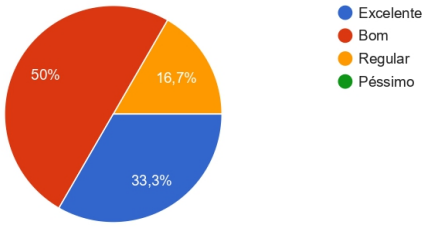
6 respostas



Utilização do módulo Autômatos finitos Não Determinísticos?

Copiar

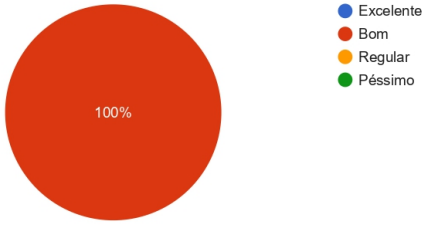
6 respostas



Material de apoio

Copiar

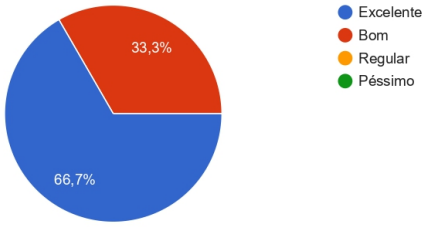
6 respostas



Vídeo aulas a respeito de linguagens formais?

Copiar

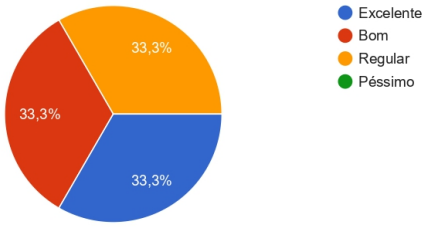
6 respostas



Exercícios a respeito de autômatos finitos?

Copiar

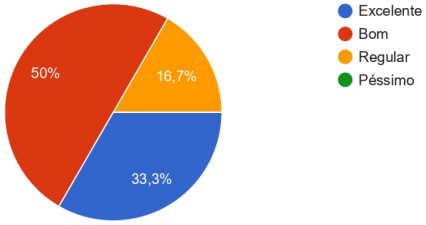
6 respostas



Qualidade das apostilas?

Copiar

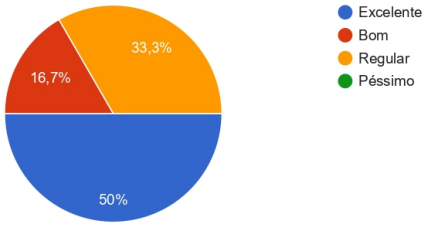
6 respostas



Clareza do módulo de ajuda?

Copiar

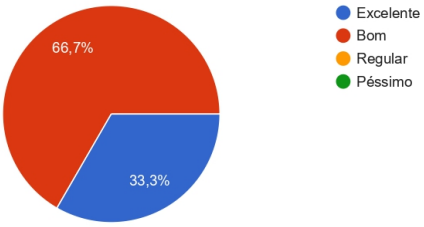
6 respostas



Clareza dos vídeos tutoriais de ajuda?

Copiar

6 respostas



Aqui você pode deixar sua opinião, sugestão ou reclamação a respeito do sistema SAELF:

5 respostas

seria bacana uma versão deste sistema para celular, legal vai ajudar bastante

gostei bastante, bem simples de usar

Muito bom espero que surja novas versões para me ajudar na faculdade.

pra um tcc tá ótima a implementação, até onde testei não tem bugs aparentes só a cara do site podia ser menos branca

é uma ferramenta importante

APÊNDICE G – PLANO DE TESTES DO SISTEMA SAELF

SAELF

Plano de Teste

Versão 01.00

Histórico da Revisão

DATA	VERSÃO	DESCRIÇÃO	AUTOR
17/05/2022	01.00	Criação dos Casos de Teste	Edson Camargo

Sumário

1. INTRODUÇÃO

Este documento de Plano de Teste tem o objetivo de documentar as informações necessárias para planejar e controlar os testes de validação do projeto SAELF – (Sistema de Apoio aos Estudos em Linguagens Formais). O documento descreve o plano geral de testes referente aos cadastros básicos de forma a direcionar os esforços de teste e os Casos de Teste a serem executados para validar o produto.

1.1 Escopo

*Este documento descreve o Plano de Testes a ser usado pelo SAELF – (Sistema de Apoio aos Estudos em Linguagens Formais). para avaliar a qualidade funcional, confiabilidade e performance. O teste que será coberto por este documento será: Validação de Expressões Regulares, validação de Autômatos finitos, determinísticos e não determinísticos, geração de palavras.

2. PLANEJAMENTO PARA OS TESTES

2.1 Necessidades de Hardware

TIPO DE HARDWARE	DETALHAMENTO	QUANTIDADE	FORMA DE DISPONIBILIZAÇÃO	DATA LIMITE
Computador	Uso pessoal com acesso a internet	1		17/05/2022

2.2 Necessidades de Software

TIPO DE SOFTWARE	DETALHAMENTO	QUANTIDADE	FORMA DE DISPONIBILIZAÇÃO	DATA LIMITE
Navegador WEB – Google Chrome	*Versão 69.0.3497.100 (Versão oficial) ou superior*	1	Corporativo	20/05/2022
Navegador WEB - Mozilla Firefox	Versão 2.x e Versão 3.x	1	Corporativo	20/05/2022

2.3 Necessidade de Pessoas

PAPEL	ENVOLVIMENTO ESTIMADO	QUANTIDADE	PERÍODO DE ENVOLVIMENTO NO PROJETO
Testador	2 horas	1	20/05/2022 à 25/05/2022

2.4 Necessidade de Capacitação

TREINAMENTO	DURAÇÃO	DATA DE REALIZAÇÃO
Não é necessário nenhum tipo de capacitação		

2.5 Cronograma de Testes

TESTES DE SISTEMA

ATIVIDADE	DATA DE INÍCIO	DURAÇÃO (HORAS)	PAPEL RESPONSÁVEL/ENVOLVIDOS
Não é necessário nenhum tipo de teste de sistema			

3. CASOS DE TESTE

Para a definição dos Casos de Teste foram utilizados os módulos desenvolvidos e a modelagem do banco de dados do SAELF.

CASO Nº	CT001 – Realizar cadastro no sistema SAELF
OBJETIVO DO TESTE	Verificar se o usuário consegue efetuar o login.
PASSOS	1. Acessar a página de login do SAELF: Menu iniciar → Programas → Navegador. 2. No campo endereço, digite: *https://www.linguagensformais.com* 3. Informe o nome de usuário: 4. Informe a senha:
CRITÉRIOS DE ÊXITO	O usuário deve conseguir acessar a próxima tela onde irá selecionar qual tipo de tarefa deseja executar

CASO Nº	CT002 – Recuperação de senha via e-mail.
OBJETIVO DO TESTE	Verificar se o usuário consegue recuperar a senha para realizar login no SAELF
PASSOS	**1. Realizar logout no sistema.** 1. O usuário será redirecionado para a página de login; 2. Clicar em recuperar senha; 3. Inserir nome de usuário e e-mail utilizado no cadastro. 4. Clicar no botão recuperar senha. **2. Realizar login em sua conta de email.** 1. Abrir o email com o titulo: Webmaster@linguagensformais.com Assunto: Recuperação de senha ; 2. Clicar sobre o link de redefinição de senha. 3. Na nova janela aberta o usuário deverá inserir a nova senha a ser utilizada e confirma-la no campo abaixo. 4. Clicar em Alterar a senha.
CRITÉRIOS DE ÊXITO	**1. Receber o email corretamente.** 1. O usuário deve conseguir receber o email corretamente. **2. Realizar atualização da senha.**

	1. O usuário deve conseguir realizar a atualização de sua senha. **3. Realizar login com a nova senha corretamente** 1. O usuário deverá conseguir realizar o login e utilização do sistema após a atualização da senha

CASO Nº	CT003 – Utilizar a função geração de palavras.
OBJETIVO DO TESTE	Verificar se as funcionalidades: Inserir símbolos, quantidade de palavras e gerar estão funcionando corretamente
PASSOS	**1. Acesso ao módulo.** 1. Realizar login; 2. Selecionar no Menu Linguagens Formais a opção gerar palavras. **2. Estado inicial da tela.** Deverão aparecer na tela as seguintes opções; 1. Campo 1 Inserir o número de palavras que deseja gerar; 2. Campo 2 inserir o alfabeto desejado separados por virgulas. **3. Cadastrar Tema.** 1. Clicar sobre o botão "Listar". 2. O sistema irá carregar a tela e mostrar as palavras geradas conforme o usuário solicitou.
CRITÉRIOS DE ÊXITO	Verificar se a quantidade de palavras está correta e facilidade de utilização

CASO Nº	CT004 – Realizar a criação e teste de autômatos finitos determinísticos utilizado o teste passo a passo
OBJETIVO DO TESTE	Verificar a execução e criação de autômatos
PASSOS	1. Acessar a página de principal do SAELF: 2. No menu inicial linguagens formais escolher a opção autômatos 3. Clicar no botão novo estado, clicar no meio do estado e arrastar em direção ao novo estado para inserir uma transição, inserir o símbolo desejado, repetir essa ação quantas vezes forem necessárias. 4. Marcar o Checkbox que desejar como estado final 5. Inserir a palavra que deseja testar e ir clicando em passo a passo para verificar a execução do autômato

CRITÉRIOS DE ÊXITO	O usuário deve conseguir criar e realizar o teste do autômato

CASO Nº	CT005 – Realizar a criação e teste de autômatos finitos determinísticos utilizado o teste direto
OBJETIVO DO TESTE	Verificar a execução e criação de autômatos
PASSOS	1. Acessar a página de principal do SAELF: 2. No menu inicial linguagens formais escolher a opção autômatos 3. Clicar no botão novo estado, clicar no meio do estado e arrastar em direção ao novo estado para inserir uma transição, inserir o símbolo desejado, repetir essa ação quantas vezes forem necessárias. 4. Marcar o Checkbox que desejar como estado final 5. Inserir a palavra que deseja testar e clicar em "Teste" para verificar se o autômato está correto.
CRITÉRIOS DE ÊXITO	O usuário deve conseguir criar e realizar o teste do autômato

CASO Nº	CT006 – Realizar a criação e teste de autômatos finitos não determinísticos utilizado o teste passo a passo
OBJETIVO DO TESTE	Verificar a execução e criação de autômatos
PASSOS	1. Acessar a página de principal do SAELF: 2. No menu inicial linguagens formais escolher a opção autômatos 3. Clicar no botão "AFND" 4. Clicar no botão novo estado, clicar no meio do estado e arrastar em direção ao novo estado para inserir uma transição, inserir o símbolo desejado, repetir essa ação quantas vezes forem necessárias. 5. Marcar o Checkbox que desejar como estado final 6. Inserir a palavra que deseja testar e ir clicando em passo a passo para verificar a execução do autômato
CRITÉRIOS DE ÊXITO	O usuário deve conseguir criar e realizar o teste do autômato

CASO Nº	CT007 – Realizar a criação e teste de autômatos não finitos determinísticos utilizado o teste direto
OBJETIVO DO TESTE	Verificar a execução e criação de autômatos
PASSOS	1. Acessar a página de principal do SAELF:

	2. No menu inicial linguagens formais escolher a opção autômatos 3. Clicar no botão “AFND” 4. Clicar no botão novo estado, clicar no meio do estado e arrastar em direção ao novo estado para inserir uma transição, inserir o símbolo desejado, repetir essa ação quantas vezes forem necessárias. 5. Marcar o Checkbox que desejar como estado final 6. Inserir a palavra que deseja testar e clicar em “Teste” para verificar se o autômato está correto.
CRITÉRIOS DE ÊXITO	O usuário deve conseguir criar e realizar o teste do autômato

CASO Nº	CT008 – Realizar utilização dos menus, assistir vídeos, acesso aos exercicios
OBJETIVO DO TESTE	Verificar a execução e criação de autômatos
PASSOS	1. Acessar a página de principal do SAELF: 2. Navegar nos menus do sistema 3. Clicar na opção vídeos e executa-los. 4. Clicar na opção exercícios tentar efetuar alguns dos exercícios e verificar sua solução. 5. Avaliar via questionário a utilização do sistema.
CRITÉRIOS DE ÊXITO	O usuário deve conseguir criar e realizar a utilização dos menus com facilidade

3.1 Testes de Aceitação do Usuário

Esse teste é conduzido pelo analista de negócio juntamente com os usuários finais do sistema (Homologação do sistema), a fim de simular operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado.

APÊNDICE H – APOSTILA DE EXERCÍCIOS DO SISTEMA SAELF

Apostila de auxilio e fixação de exercícios SAELF

Algumas considerações são necessárias para que este conteúdo de Linguagens Formais fique claro o suficiente para a realização dos exercícios propostos, neste contexto a criação desta apostila é de suma importância para auxiliar os alunos da disciplina. Desta forma devido à falta de conteúdo simplificado e objetivo deste assunto foi elaborada esta apostila.

- Alfabeto

Um Alfabeto é um conjunto finito de símbolos, e normalmente é adotado como símbolo o **Σ**, portanto, um conjunto infinito não é um alfabeto, por outro lado o conjunto vazio é um alfabeto e seu símbolo é **∅**. Uma palavra sobre um alfabeto **Σ** é uma cadeia finita de símbolos e denotada por **ω**, já o conjunto de todas as palavras de um alfabeto **Σ** é denotado por **Σ*** e para denotarmos uma palavra vazia, ou seja, sem nenhum símbolo, utilizaremos **ε**. (MENEZES, 2000)

- Palavra

Em uma palavra sobre um alfabeto **Σ** podemos ter Prefixo, Sufixo e Subpalavra. Prefixo é qualquer sequência inicial de uma palavra, sufixo é qualquer sequência final e subpalavra é qualquer sequência de símbolos contíguos de uma palavra sendo assim qualquer prefixo ou sufixo é uma subpalavra. (MENEZES, 2000)

Exemplo:

Σ = {a, b}

ω = abbaa

Prefixos: ε, a, ab, abb, abba, abbaa.

Sufixos: ε, a, aa, baa, bbaa, abbaa.

- Concatenação de palavras

A concatenação de palavras é uma operação binária definida sobre o produto cartesiano de dois conjuntos, onde a primeira palavra é associada com a segunda exatamente nesta ordem.

Exemplo:

Σ = {a, b}

ω_1 = {ab} e ω_2 = {ba}

Logo a concatenação será:

$\omega_1 \bullet \omega_2 = \{abba\}$

Podemos também concatenar uma palavra com a palavra vazia ε.

$\omega_1 \bullet \varepsilon = \{ab\}$

- Concatenação Sucessiva ou Fecho de Kleene

Concatenação sucessiva ou Fecho de Kleene é uma operação unária e infinita, porém, contável. O fecho de Kleene de **L** é denotado por **L*** onde o * significa que o "**L**" pode se repetir 0 ou mais vezes. Definição fecho de Kleene: Seja **Σ** um alfabeto e **L** ⊆ **Σ***, logo **$L^* = \{\omega_1, \omega_2, \ldots \omega_n \mid \omega_i \in L, 1 \leq i \leq n, n \geq 0\}$** (FURTADO, 2007)

Exemplo:

$\Sigma = \{a, b, c\}^*$

$L^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab...\}$

- Linguagem

Uma linguagem é um conjunto de palavras sobre **Σ** e denotada por **L**. Definição concatenação de linguagens: Seja **Σ** um alfabeto e **$L_1, L_2 \subseteq \Sigma^*$** a concatenação de **$L_1$** e **$L_2$** é denotada por **$L_1 \circ L_2 = \{\omega_1 \omega_2 \mid \omega_1 \in L_1 \wedge \omega_2 \in L_2\}$** (FURTADO, 2007)

Exemplo:

$L_1 = \{\varepsilon, a, b\}$

$L_2 = \{0, 1\}$

$L_1 \circ L_2 = \{0, 1, a_0, a_1, b_0, b_1\}$

- Propriedades das Linguagens:

Segundo Furtado (2007) a classe das Linguagens Regulares (LR) é fechada para as operações de União, Concatenação, Complemento e Intersecção, abaixo veremos suas propriedades:

- UNIÃO:
  - Seja L_1 e L_2 (LR), então $L_1 \cup L_2 = \{\omega \mid \omega \in L_1 \vee \omega \in L_2\}$ também é uma LR
- CONCATENAÇÃO:
  - Seja L_1 e L_2 (LR), então $L1 \circ L2 = \{\omega\beta \mid \omega \in L_1 \wedge \beta \in L_2\}$ também é uma LR.

- COMPLEMENTO:
  - Se $L_1 \subseteq \Sigma^*$ é um LR, então $\Sigma^* - L_1$ também é uma LR.
- INTERSECÇÃO:
  - Seja L_1 e L_2 (LR), então $L_1 \cap L_2 = \{\omega \mid \omega \in L_1 \wedge \omega \in L_2\}$ também é uma LR.

➢ Linguagens Regulares

Segundo a Hierarquia de Chomsky, a Linguagem Regular trata-se de uma classe de linguagens um pouco mais simplificada, sendo possível desenvolver algoritmos de reconhecimento ou de geração com um nível baixo de complexidade possuindo alta eficiência e de fácil implementação. (MENEZES, 2000)

As linguagens regulares são aquelas que podem ser representadas por uma expressão regular (ER) ou por um autômato finito.

➢ EXPRESSAO REGULAR

Uma Expressão Regular é qualquer padrão que possa ser encontrado em uma linguagem. Afim de explicar de forma resumida e clara, é uma linguagem onde dizemos o padrão de um texto ou cadeia de caracteres que queremos encontrar, informamos onde o padrão deve ser encontrado e iniciamos a busca. De forma análoga podemos comparar essa busca com a função localizar de um arquivo Word ou até mesmo pdf, porem de forma mais detalhada e profunda.

Para exemplificar utilizaremos uma expressão regular que valida o número do CPF de um sujeito qualquer.

O CPF (Cadastro de Pessoa Física) é um documento brasileiro emitido pela Secretaria da Receita Federal do Ministério da Fazenda. Seu número é composto por 11 dígitos, sendo os dois últimos os dígitos verificadores, que atestam se o número do CPF é válido.

O formato de um CPF é: 000.000.000-00

A partir dessa expressão regular é possível garantir que o usuário de um sistema, digite o CPF no formato correto.

Expressão regular = **[0-9]{3}\.?[0-9]{3}\.?[0-9]{3}\-?[0-9]{2}**

[0-9]{3} Faixa de caracteres: 0 a 9, quantidade: 3 caracteres;

[0-9]{3} Faixa de caracteres: 0 a 9, quantidade: 3 caracteres;

[0-9]{3} Faixa de caracteres: 0 a 9, quantidade: 3 caracteres;

[-]? Um traço, opcional (se acrescentar outros caracteres, comece pelo - sempre);

[0-9]{2} Faixa de caracteres: 0 a 9, quantidade: 3 caracteres;

[\.]?Um ponto, opcional. Foi usado \ no ponto, pois ele sozinho é caractere especial;

Com essa expressão regular é possível garantir que o usuário de um sistema digite no formato correto o CPF.

Obs: Garante que o CPF está no formato correto, mas não garante que seja um CPF válido.

Alfabeto → Σ = {a, b}

Palavra → ω = abbaa ω = ε

Linguagem.

Σ = {a, b, c} *

L*= {ε, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab...}

As linguagens regulares são aquelas que podem ser representadas por uma expressão regular ou por um autômato finito.

L= {ω ϵ {0,1} * | ω possui apenas um 0}

ER = 1* 0 1*

L= {0, 01, 10,011,101, 110, 0111, 1011, 1101, 1110...}

As expressões regulares são utilizadas para validar uma cadeia de caracteres.

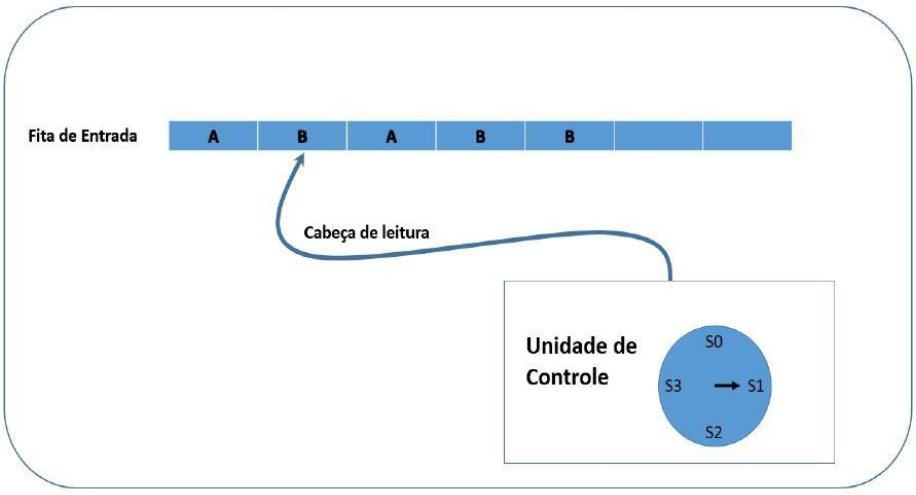
Tabela 2 – Exemplos de Expressões Regulares

Exp. Regular	Linguagens Representadas
ab	Somente a palavra ab
ba*	Todas palavras que iniciam com b, seguido de zero ou mais as
(a | b)*	Todas as palavras sobre o alfabeto {a,b}
(a | b)*aab(a | b)*	Todas as palavras contendo aab como sub palavra.
a*ba*ba*	Todas as palavras contendo exatamente dois b
(a | b)*(aa | bb)	Todas as palavras que terminam com aa ou bb
(a | ε)(b | ba)*	Todas as palavras que não possuem dois a consecutivos

Fonte: Arquivo pessoal

Por definição, uma linguagem L é dita regular se, e somente se, é possível construir um Autômato Finito (Determinístico ou Não-Determinístico) ou uma expressão Regular que reconheça a linguagem. (OLIVEIRA, 2009)

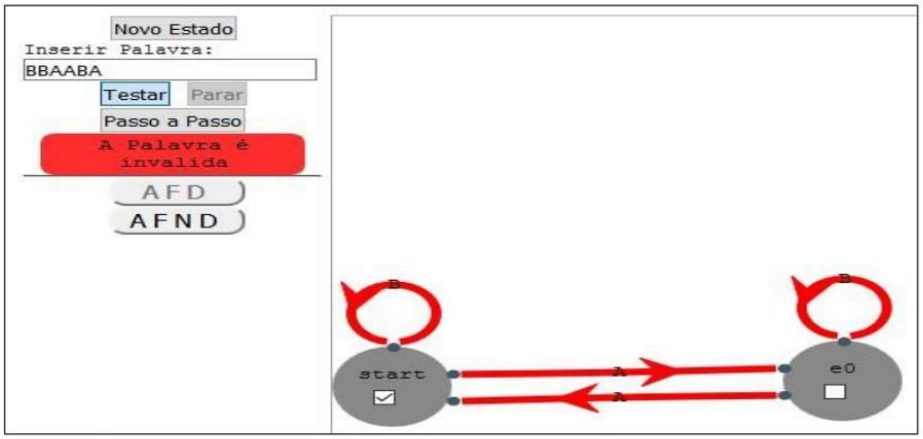
- Um autômato pode ser dividido em três partes.



- Um AFD é uma tupla definida formalmente como:

M = (K, Σ, δ, qo, F)

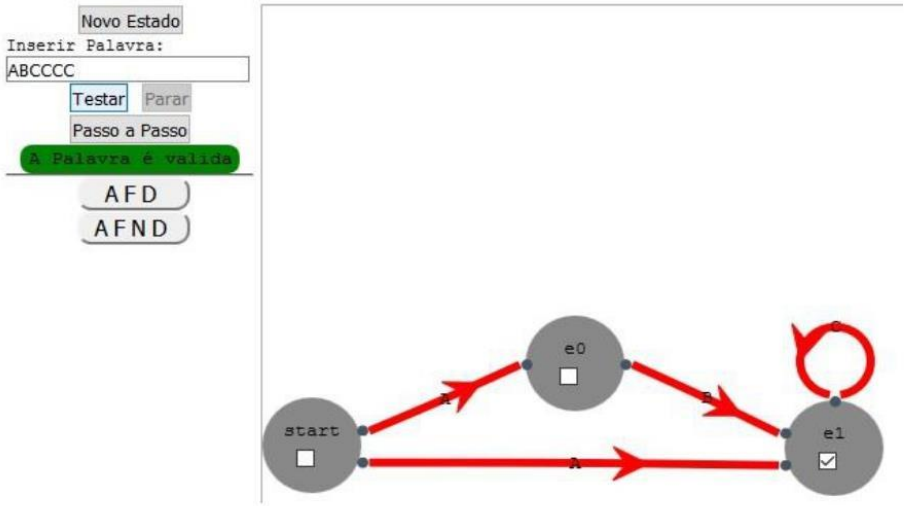
L= {ω ϵ {A,B} * | ω possui apenas um Nº par de As}



Fonte: Arquivo pessoal

- Um AFND é uma tupla definida formalmente como:

M = (K, Σ, Δ, S, F)



Exemplos: Expressões Regulares e Autômatos

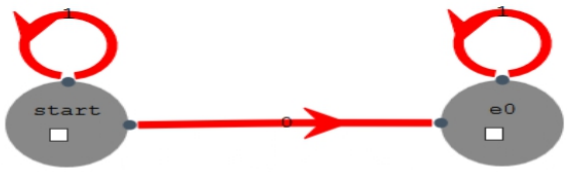
Alguns exemplos de expressão regular (ER) e autômatos finitos estão sendo abordados a seguir para facilitar o aprendizado.

1) L= {ω ϵ {0,1} * | ω possui apenas um 0}

ER = 1* 0 1*

L= {0, 01, 10,011,101, 110, 0111, 1011, 1101, 1110...}

Figura 2 - Autômato correspondente ao exemplo 1.



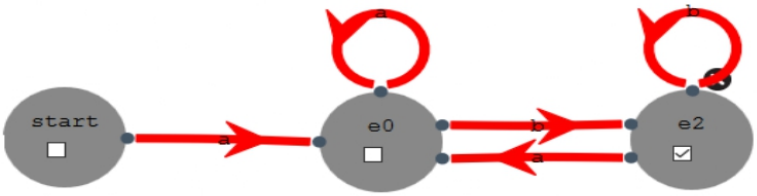
Fonte: Arquivo Pessoal

2) L = {ω ϵ {a,b} * | ω começa com “a” e termina com “b”}

ER = a (a | b)* b

L= {ab, aab, abb, aaab, aabb, abab, abbb, aaaab, aaabb, aabab...}

Figura 3 - Autômato correspondente ao exemplo 2.



Fonte: Arquivo Pessoal

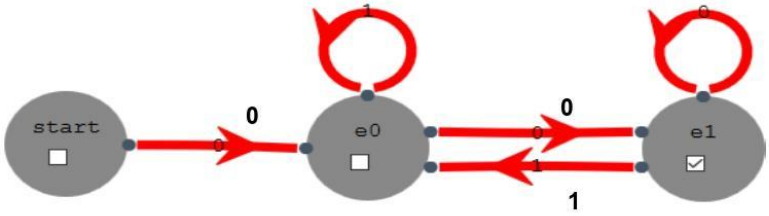
3) L = {ω ϵ {0,1} * | ω começa e termina com 0}

ER = (0 (0 | 1)* 0) | 0

L= {0, 00, 000, 010, 0000, 0100, 0110, 0010, 00000, 01000...}

Figura 4 - Autômato correspondente ao exemplo 3.

1 **0**



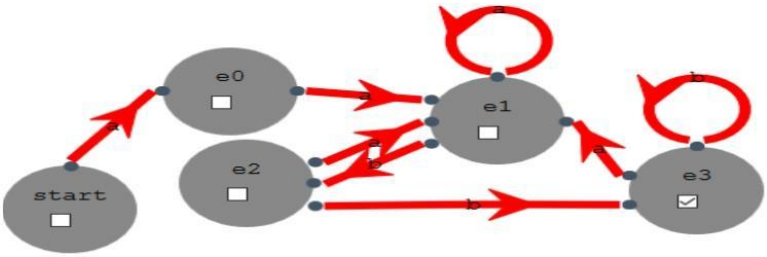
Fonte: Arquivo Pessoal

4) L = {ω ϵ {0,1} * | ω começa com aa e termina com bb}

ER = aa (a|b)* bb

L = {aabb, aaabb, aabbb, aaaabb, aaabbb, aababb, aabbbb, aaaaabb}

Figura 5 - Autômato correspondente ao exemplo 4.



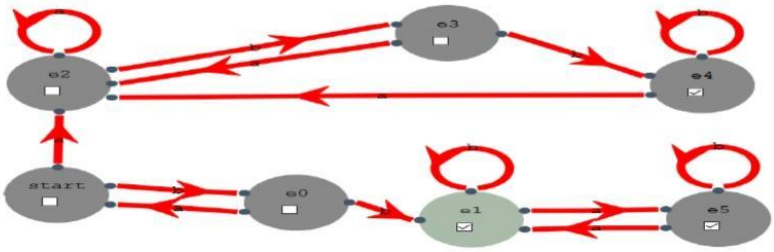
Fonte: Arquivo Pessoal

5) L= {ω ϵ {a,b} * | ω começa ou terminam com bb}

ER= bb (a|b)*| (a|b)* bb

L= {bb, bba, abb, bbb, aabb, abbb, bbaa, bbab, bbba, bbbb...}

Figura 6- Autômato correspondente ao exemplo 5.



Fonte: Arquivo Pessoal

Exercícios propostos: Expressões Regulares

Nível iniciante:

1) L={ω ∈ {a,b}* | ω começa e termina “a” }.
2) L={ω ∈ {a,b}* | ω começa e termina “a”}.
3) L={ω ∈ {a,b}* | ω começa “a” e termina com “b”}.
4) L={ω ∈ {a,b}* | ω começa e termina “a” ou começa e termina com “b”}.
5) L={ω ∈ {a,b}* | ω começa e termina “a” e possui a subpalavra “abb”}.

Nível básico:

1) L={ω ∈ {a,b}* | ω possui exatamente três “a” }.
2) L={ω ∈ {a,b}* | ω possui um número par de “a”}.
3) L={ω ∈ {a,b}* | ω começa com “b” e tem número ímpar de “a”}.
4) L={ω ∈ {a,b}* | ω começa e termina “a” e tem exatamente três “b”}.
5) L={ω ∈ {a,b}* | ω não possui mais de dois “b” consecutivos}.

Nível difícil:

1) L={ω ∈ {a,b}* | ω começa e termina “a” tem exatamente dois “a”, começa e termina com “b” e tem número ímpar de “b” }.
2) L={ω ∈ {a,b}* | ω não tem dois “b” consecutivos e possui um número par de “a”}
3) L={ω ∈ {a,b}* | ω não possui “ababba”}
4) L={ω ∈ {0,1}* | ω começa com “1” termina com “0” ou começa com “0” e termina com “1” e não possui três “1” consecutivos}
5) L={ω ∈ {a,b}* | ω começa com “a”, não possui três “a” consecutivos, e o número de “b” deve ser ímpar }

Exercícios propostos: Autômatos Finitos Determinísticos

Nível iniciante:

1) Construa um autômato finito determinístico que reconheça a linguagem L={ω ϵ {a,b}* | ω possui a subpalavra “aba” }.
2) Construa um autômato finito determinístico que reconheça a linguagem L={ω ϵ {a,b}* | ω possua “ab” como prefixo }
3) Construa um autômato finito determinístico que reconheça a linguagem L={ω ϵ {a,b}* | ω possua “ba” como sufixo}
4) Construa um autômato finito determinístico que reconheça a linguagem L={ω ϵ {a,b}* | ω possui “ab” como prefixo, “ba” como sufixo e “baab” como subpalavra }.
5) Construa um autômato finito determinístico que reconheça a linguagem L={ω ϵ {a,b}* | ω possui uma sequência de “a” e “b”, com número ímpar de “b” }.

Nível básico:

1) Construa um autômato finito determinístico que reconheça a linguagem L={ω ϵ {a,b}* | ω possui a subpalavra “aba” }.
2) Construa um autômato finito determinístico que reconheça a linguagem L(M) = { ω ϵ {0,1}* | ω ∈ {0,1}* e possui um número par de ocorrências de “0” e ímpar de “1” }
3) Construa um autômato finito que reconheça a linguagem gerada pela expressão regular 1(011*0)*.
4) Construa um autômato finito que reconheça a linguagem gerada pela expressão regular 1(01*0 | 11)*1*0.
5) Construa um autômato finito determinístico que reconheça a linguagem L={ω ϵ {a,b}* | ω tem exatamente três “b”, começa e termina com “b” e e tem um número ímpar de “a”}.

Nível difícil:

1) Construa um autômato finito determinístico sobre o alfabeto Σ = {1, 2, 3} que reconheça a linguagem L = {ω ϵ {0,1}* | ω possui 312 como prefixo, 211 como subpalavra e 121 como sufixo}.
2) Construa um autômato finito determinístico sobre o alfabeto Σ ={0,1} tal que L(M) = { ω ϵ {0,1}* | ω tenha dois 0's consecutivos OU dois 1's consecutivos}.
3) Construir um autômato finito determinístico sobre o alfabeto Σ = {a, b} que reconheça a linguagem L = {ω ϵ {a,b}* | ω possui um número arbitrário de repetições "a" seguidos pelo mesmo número de "b", sendo no máximo 3 repetições}. Exemplo: L(M) = (ab | aabb | aaabbb)*