UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA DE APOIO AO ENSINO INTERATIVO DE EXPRESSÕES REGULARES

GABRIEL TAKASHI KATAKURA

BLUMENAU
2018

GABRIEL TAKASHI KATAKURA

FERRAMENTA DE APOIO AO ENSINO INTERATIVO DE EXPRESSÕES REGULARES

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Profa. Joyce Martins, Mestre - Orientadora

BLUMENAU
2018

FERRAMENTA DE APOIO AO ENSINO INTERATIVO DE EXPRESSÕES REGULARES

Por

GABRIEL TAKASHI KATAKURA

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Profa. Joyce Martins, Mestre – Orientadora, FURB
Membro:	Prof. Alexander Roberto Valdameri, Mestre – FURB
Membro:	Prof. Aurélio Faustino Hoppe, Mestre – FURB

Blumenau, 12 de julho de 2018

Dedico este trabalho a minha mãe, que sempre me guiou e apoiou para atingir a conclusão do meu curso de graduação.

AGRADECIMENTOS

Agradeço principalmente minha mãe, que me levou até as oportunidades de estudo que me guiaram para a minha formação acadêmica e profissional.

À minha orientadora Joyce, que dedicou seu tempo com paciência e competência, me guiando por cada etapa que passei neste trabalho.

À comunidade de desenvolvimento de React e Node.js, que compartilharam seu conhecimento, me permitindo escolher as ferramentas mais adequadas para concluir este trabalho.

All problems in computer science can be solved by another level of indirection.

David John Wheeler

RESUMO

Este trabalho apresenta o desenvolvimento de uma ferramenta de apoio ao ensino de expressões regulares. A ferramenta permite o professor gerenciar o cadastro de exercícios, atividades e turmas, distribuir os exercícios para os estudantes e efetuar o acompanhamento em tempo real, enquanto a ferramenta auxilia o estudante na resolução do exercício. Na implementação foi utilizada a biblioteca React para o desenvolvimento da interface e, em conjunto de WebSockets, foram desenvolvidos os mecanismos de notificação em tempo real. Para a persistência de dados foi utilizado o MongoDB, um banco NoSQL que se adequou a modelagem definida para o desenvolvimento da ferramenta. A avaliação da ferramenta foi feita com um teste prático em sala de aula com estudantes do curso de Sistemas de Informação da FURB. O trabalho atingiu todos os objetivos proposto, e com base nas análises dos resultados obtidos, mostrou ser uma ferramenta adequada para o ensino de expressões regulares com o supervisionamento de um professor em sala de aula.

Palavras-chave: Expressões regulares. Ferramenta de ensino. Acompanhamento em tempo real.

ABSTRACT

This work presents the development of a support tool in the teaching of regular expressions. The tool allows the teacher to manage the register of exercises, activities and classes, distribute the exercises to the students and carry out the monitoring in real time. In the implementation, the React library was used to develop the interface and, together with WebSockets, the real-time notification mechanisms were developed. For the data persistence was used MongoDB, a NoSQL database that adapted to the modeling defined for the development of this tool. The test of the tool was done with a practical test in the classroom with students of the course of Information Systems of FURB. The proposed work reached all the objectives and, based on the analysis of the obtained results, proved to be an adequate tool for the teaching of regular expressions with the supervision of a teacher in the classroom.

Key-words: Regular expressions. Teaching tool. Real-time tracking.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

JFLAP – Java Formal Languages and Automata Package

REST – Representational State Transfer

RF - Requisito Funcional

RNF – Requisito Não Funcional

SUMÁRIO

1 INTRODUÇÃO

Atualmente vive-se em uma era onde os avanços tecnológicos em diversas áreas estão acontecendo “[...] mais depressa do que nossa capacidade para acompanhá-los [...]” (PIVA JUNIOR, 2008, p. 1). Uma área em especial que tornou-se tópico de diversos debates de como as pessoas deveriam reagir a essas mudanças é a educação, na qual, conforme Rinaldes (2013), o uso do computador como ferramenta de ensino acabou gerando a necessidade das “[...] redefinições do papel dos professores nesse novo contexto” (RINALDES, 2013, p. 1). Pode-se destacar o ensino na área da computação como uma das mais afetadas por estes avanços tecnológicos tanto em relação aos conteúdos abordados, já que surgem novas tecnologias e consequentemente novos conteúdos a serem estudados, quanto à forma como são ensinados, preferencialmente usando ferramentas ou metodologias ativas que facilitem o aprendizado.

Nesse sentido, a área da computação está repleta de sites e ferramentas com foco no ensino, onde vários destes seguem um modelo de ensino autodidata e geralmente dispondo um ambiente de desenvolvimento interativo, conforme proposto por Victor (2017) em Learnable Programming. Além destes, existem iniciativas para o ensino de programação, como a Computação na Escola (WANGENHEIM et. al., 2013, p. 1), que “[...] é dedicada a aumentar o ensino de computação no Ensino Fundamental e Médio”. Porém, mesmo com a existência de tantos sites e iniciativas, a área da computação é muito ampla e com várias áreas de conhecimento, o que dificulta o desenvolvimento de ferramentas capazes de concentrar todo o material necessário para um ensino adequado de cada tópico desta área, necessitando de ferramentas com um foco mais específico para o ensino de certos conteúdos.

Um dos conteúdos estudados nos cursos de Ciência da Computação é o de Linguagens Formais, que possui como foco o estudo da “[...] análise léxica e sintática de linguagens de programação” (MENEZES, 2000, p. 1). Uma abordagem de ensino sobre Linguagens Formais apresenta os tipos de linguagens definidas pela Hierarquia de Chomsky, que classifica as linguagens formais em uma hierarquia de quatro níveis. O primeiro nível dessa hierarquia é conhecido como linguagem regular, sendo um conceito relacionado o de expressões regulares. As expressões regulares, além de servirem para a especificação de linguagens regulares, são comumente disponibilizadas nas linguagens de programação para a validação de padrões como telefones, datas e códigos em geral. O aprendizado de expressões regulares por desenvolvedores é importante, visto que, conforme Goyvaerts e Levithan (2011, p. 16), “Se [...] expressões regulares [forem utilizadas] com habilidade, elas podem simplificar muitas

tarefas de programação e processamento de texto, além de permitir outras que não seriam possíveis sem elas".

Diante do exposto, este trabalho apresenta o desenvolvimento de uma ferramenta web para apoio ao ensino de expressões regulares, modelada de forma que cada ação de um estudante notifique o professor em tempo real, objetivando tornar as aulas mais ágeis e o aprendizado mais efetivo.

1.1 OBJETIVOS

O objetivo deste trabalho é criar uma ferramenta web de ensino interativo de expressões regulares.

Os objetivos específicos são:

a) disponibilizar um canal interativo para o desenvolvimento e compartilhamento de atividades;
b) ter um mecanismo de notificações em tempo real para o acompanhamento das atividades;
c) validar as expressões regulares especificadas, determinando a equivalência entre duas expressões regulares - a especificada pelo professor e a especificada pelo estudante.

1.2 ESTRUTURA

Este trabalho foi dividido em quatro capítulos. O segundo capítulo apresenta a fundamentação teórica necessária para o entendimento desta ferramenta. O terceiro capítulo apresenta o desenvolvimento da ferramenta, abordando tópicos como os requisitos funcionais e não funcionais levantados, a especificação da ferramenta com casos de uso e diagrama de classes, a implementação e os resultados obtidos com testes aplicados em sala de aula. Por fim, o quarto e último capítulo, apresenta as conclusões obtidas com o desenvolvimento deste trabalho e as sugestões de possíveis extensões.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo descreve os assuntos que fundamentaram o trabalho desenvolvido, ou seja, é apresentado o conceito de expressões regulares e a aplicabilidade delas no dia-a-dia dos desenvolvedores, bem como é dada uma visão geral sobre as plataformas de ensino modernas. Também são descritos três trabalhos com características similares ao trabalho desenvolvido.

2.1 EXPRESSÕES REGULARES

A origem das expressões regulares "[...] começou nos anos 40, quando Warren McCulloch e Walter Pitts, neurologistas, descreveram [uma tese sobre] o funcionamento dos neurônios. Mais tarde, Stephen Kleene passou o estudo para uma notação matemática" (CUNHA, 2005, p. 1). Com a criação dessa notação matemática, o estudo das expressões regulares foi aprofundado pelos matemáticos da época por cerca de 20 anos antes de seu uso efetivo nos computadores (JARGAS, 2012).

Foi somente em 1968 que um algoritmo de busca desenvolvido para o editor de texto qeq incluiu o uso de expressões regulares, algo que originou a criação do aplicativo grep, um comando para sistemas baseados em Unix que aceita o uso de expressões regulares (JARGAS, 2012). Já em 1986 foi criado um pacote chamado regex para a linguagem de programação C, que acabou tornando o uso cada vez mais popular das expressões regulares nos programas (JARGAS, 2012).

As expressões regulares servem para especificar um padrão de texto, normalmente com o uso de metacaracteres que "[...] são caracteres com funções especiais" (CUNHA, 2005, p. 1). Uma expressão regular simples como `abc` é capaz de capturar o texto literal `abc`, mas somente este texto, ou seja, possui um conjunto finito de valores válidos. Já a expressão `a+|bc?`, que usa os metacaracteres `+` (uma ou mais vezes) e `?` (zero ou uma vez) pode ser entendida como: o texto é composto pelo caractere `a` uma ou mais vezes ou pelo caractere `b` seguido ou não do caractere `c`, aceitando assim textos como `a`, `aa`, `b`, `bc`, entre outros, permitindo desta maneira um conjunto infinito de valores válidos já que o caractere `a` pode ser repetido infinitamente.

Conforme Cunha (2005, p. 1), existem "[...] 14 metacaracteres [primitivos que] podem ser divididos em 04 (quatro) categorias: representantes, quantificadores, âncoras, e outros.". Com o uso desses metacaracteres, pode-se criar expressões regulares que identificam padrões. Um cenário onde o uso de expressões regulares é algo comum é na implementação de

compiladores para validar *tokens*, isto é, símbolos válidos escritos em um programa, ou para descartar caracteres de formatação e comentários, desnecessários para a etapa da análise sintática. Ainda, o uso de expressões regulares é tão importante para encontrar e validar padrões de texto, tais como um telefone, um e-mail ou uma data, normalmente presentes no cadastro de perfil de um usuário de um sistema ou de um cliente de uma loja, que linguagens como JavaScript, Perl e Ruby possuem suporte para expressões regulares literais, ou seja, possuem uma definição na gramática para suporte sintático a expressões regulares, enquanto linguagens como C#, Java e Python dispõem de bibliotecas que implementam a funcionalidade.

Alguns dos metacaracteres mais comuns e simples de serem utilizados são os de ocorrência, de alternativa, de definição de grupo e de conjunto, descritos no Quadro 1.

Quadro 1 – Metacaracteres

METACARACTERE	CONHECIDO COMO	PALAVRAS QUE ACEITA
`{n}`	exatamente `n` ocorrências	a definição precedente ocorre exatamente `n` vezes
`+`	uma ou mais ocorrências	a definição precedente ocorre uma ou mais vezes
`*`	nenhuma ou mais ocorrências	a definição precedente ocorre zero ou mais vezes
`?`	nenhuma ou uma ocorrência	a definição precedente ocorre zero ou uma vezes
`|`	alternativa	uma ou outra definição
`(...)`	grupo	definição de grupo
`[...]`	conjunto	definição de conjunto de caracteres

Fonte: elaborado pelo autor.

Os metacaracteres de ocorrência, também conhecidos como quantificadores, permitem identificar padrões de texto que denotam uma repetição de caracteres. Um exemplo disso é uma expressão regular para definir um número inteiro positivo, que pode ser especificado como `((1|2|3|4|5|6|7|8|9)0*)+`. Essa expressão capturar qualquer número inteiro como `1`, `10`, `12` e `1204`, além de definir que o dígito mais à esquerda deve ser qualquer um diferente de `0` de tal forma que palavras como `010` são descartadas pela expressão regular. Nesse exemplo, tem-se também o uso dos metacaracteres de grupo e alternativa.

2.2 PLATAFORMAS DE ENSINO MODERNAS

Atualmente a quantidade de informação gerada é excessivamente maior que a capacidade que o ser humano tem de apreender informação útil. James (2012) cita que a cada minuto são gerados mais de 48 horas de vídeos no YouTube e são criados mais de 571 sites. Ainda, a forma com que o conteúdo se propaga está afetando diretamente a metodologia de

ensino usada nas salas de aula. Além do conteúdo mais acessível, novas tecnologias digitais têm tornado o aprendizado cada vez mais flexível em termos de tempo e espaço.

Sendo assim, conforme Rinaldes (2013, p. 1), "[...] o papel do professor deve ser não mais o de ensinar, mas o de facilitador/orientador/mediador da aprendizagem, instigando a curiosidade do aluno". Seguindo esse pensamento, as plataformas de ensino modernas têm se apresentado como ferramentas que servem como um canal de comunicação entre o professor e o aluno, permitindo que dúvidas e interações entre os alunos e o professor sejam feitas a qualquer momento. Tem-se também plataformas de ensino-aprendizagem que permitem que estudantes tenham mais flexibilidade sobre o que e quando querem aprender, pois as informações estão disponíveis em todos os momentos (ESPÍNDOLA, 2016). Nesse sentido, ferramentas deste estilo criam um caminho mais acessível para que os estudantes consigam buscar conhecimento fora do modelo tradicional de ensino.

Espíndola (2016) escreve que uma plataforma de ensino-aprendizagem oferece um ambiente que centraliza as informações e que serve como meio para a formação, desempenho e desenvolvimento dos conteúdos. Ainda, conforme Espíndola (2016, p. 1), é "[...] uma plataforma [... que] proporciona não só um melhor aproveitamento para os usuários, como auxilia na agilidade de diversos processos". Essas plataformas podem ser aplicadas de diversas maneiras, uma delas, por exemplo, é a utilização no ensino a distância.

As plataformas de ensino-aprendizagem possuem geralmente recursos tecnológicos como "[...] mecanismos para envio de mensagens, bate-papo, recepção e envio de materiais e gerenciadores de tarefas" (LMS..., 2012, p. 1). Apesar das plataformas de ensino-aprendizagem serem desenvolvidas para contextos pedagógicos diferenciados, a evolução deste modelo demonstrou historicamente que existe um conjunto de funcionalidades básicas como (LMS..., 2012, p. 1): acesso protegido; gestão de perfis, de acesso a conteúdos e de acesso; comunicação entre os envolvidos, controle de atividades. Além destas funcionalidades, uma característica importante para as plataformas de ensino é a usabilidade do sistema, onde foi constatado por Reitz, Lima e Axt (2011, p. 149), através de experimentos aplicados em uma sala com dois grupos, "[...] que a usabilidade técnica e pedagógica influência [sic] significativamente a aprendizagem e o desempenho dos alunos".

2.3 TRABALHOS CORRELATOS

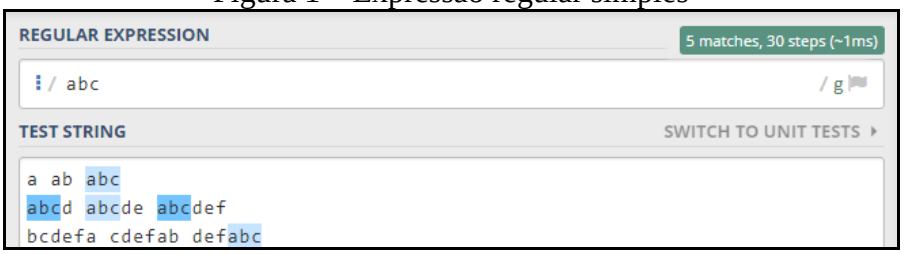
Não foram encontradas ferramentas de ensino interativo sobre o tema proposto neste trabalho. Desta forma, abaixo são apresentados trabalhos semelhantes que podem ser utilizados como ferramenta de ensino. O primeiro é a aplicação web Regex101 (DIB, 2014),

que permite visualizar detalhadamente a composição de uma expressão regular bem como analisar os valores especificados a partir da mesma. O segundo é o JFLAP (RODGER; FINLEY, 2008), uma ferramenta para criação e simulação de alguns tipos de autômatos e que possui suporte para conversões de expressões regulares para autômatos finitos e vice-versa. O terceiro é a aplicação web Regulex (CHENG, 2014), que permite visualizar uma representação gráfica de uma expressão regular.

2.3.1 REGEX101

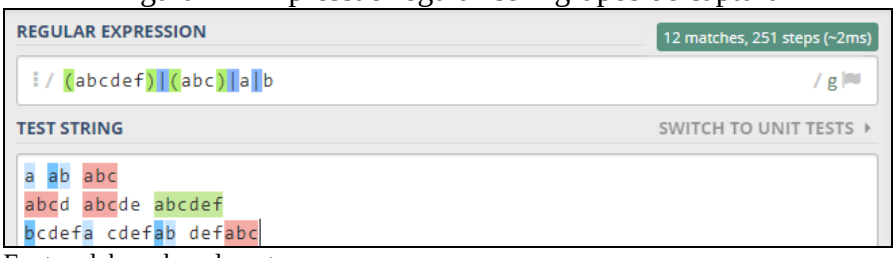
O Regex101 foi criado por Dib (2014), sendo originalmente desenvolvido como um simples projeto. Com o tempo tornou-se um dos maiores sites para testar expressões regulares. O site é intuitivo e o seu uso é simples. Na página inicial, conforme ilustra a Figura 1, o usuário deve informar uma expressão regular no campo `REGULAR EXPRESSION` e valores para serem testados no campo `TEST STRING`. Os valores válidos pela expressão regular são destacados no texto com uma cor diferente. No caso da Figura 1, estão destacadas as sequências de símbolos compostas por `abc`. Caso a expressão regular possua grupos de captura, os valores que são capturados por estes são destacados com uma cor diferente para cada grupo (Figura 2).

Figura 1 – Expressão regular simples



Fonte: elaborado pelo autor.

Figura 2 – Expressão regular com grupos de captura

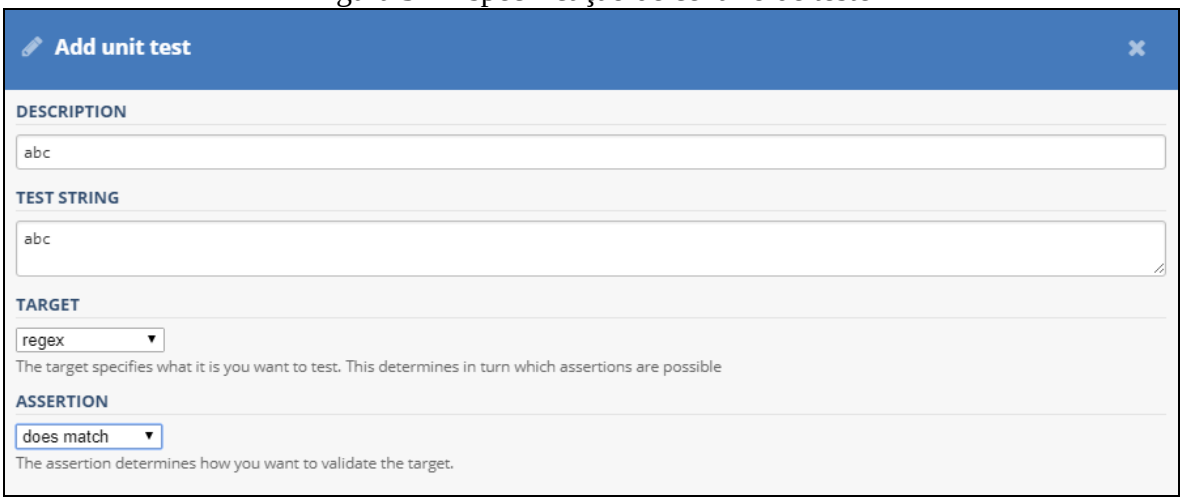


Fonte: elaborado pelo autor

A aplicação também permite criar cenários de teste em vez de entrar com somente um texto para validação. Um cenário de teste, visualizado na Figura 3, pode conter uma descrição (`DESCRIPTION`), um texto para validação (`TEST STRING`) e uma afirmação (`ASSERTION`) que permite informar se o texto é válido ou não. Depois de criados os cenários, pode-se executá-los. Os testes que finalizarem com sucesso estarão em verde (como os dois primeiros testes da

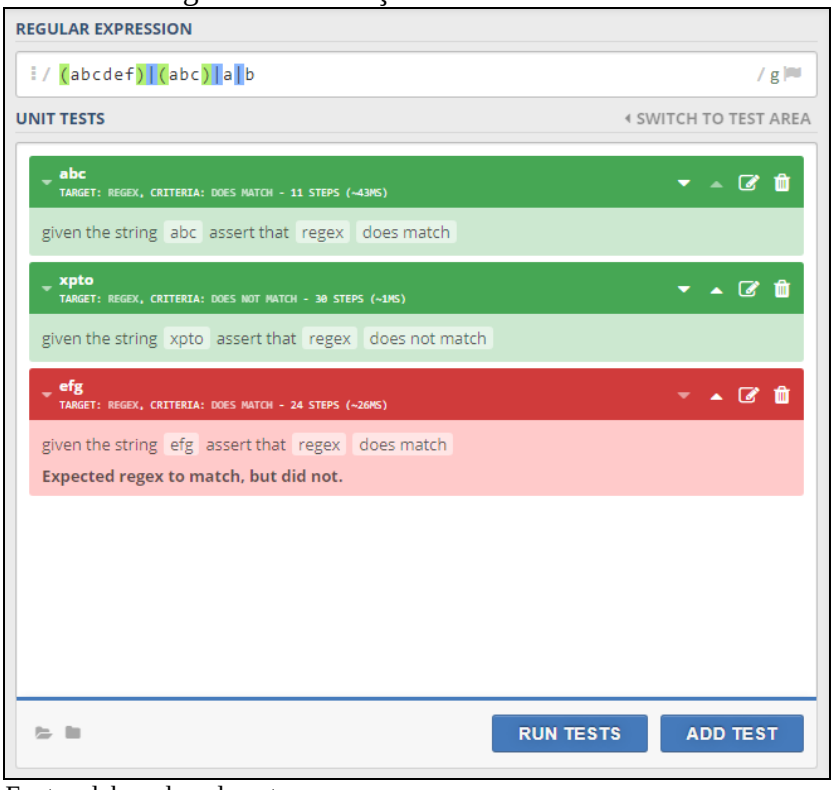
Figura 4), enquanto os que falharem estarão em vermelho (como o último teste da Figura 4). É possível expandir os testes para uma informação mais detalhada, através das setas ao lado do identificador do teste.

Figura 3 – Especificação de cenário de teste



Fonte: elaborado pelo autor.

Figura 4 – Execução de cenários de teste

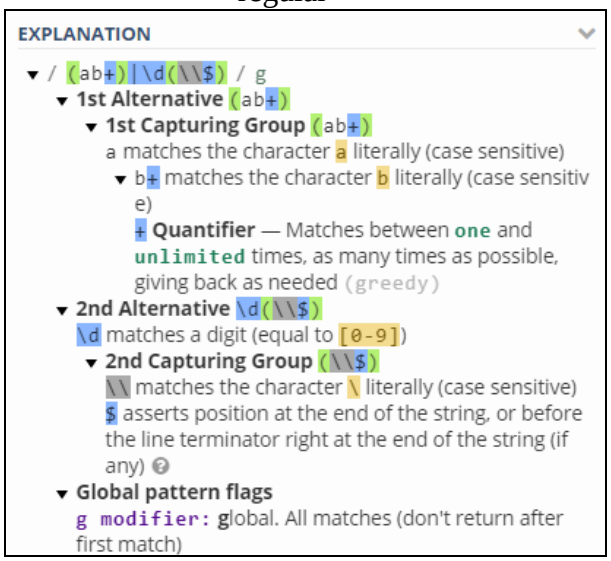


Fonte: elaborado pelo autor.

No canto superior direito da ferramenta existe uma área explicando a expressão regular informada. A aplicação mostra uma visualização em forma de árvore da expressão regular (Figura 5), ramificando em alternativas de captura, identificando grupos de captura e detalhando os metacaracteres usados (quantificadores, âncoras, entre outros). Já no canto inferior direito existe um guia de referência que explica os metacaracteres que podem ser

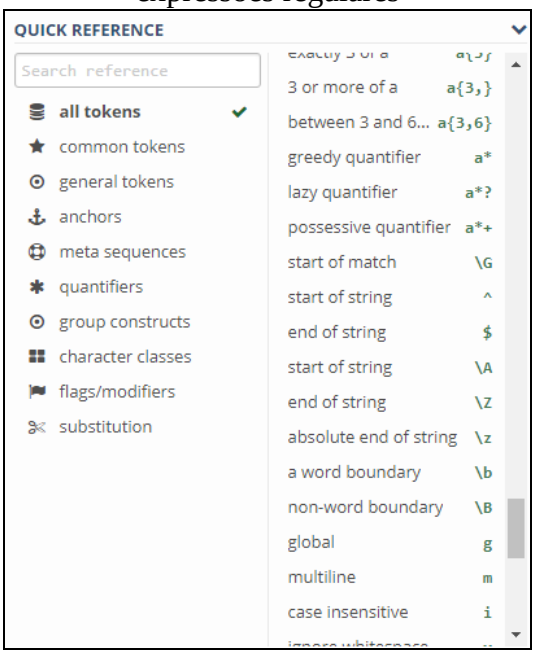
usados para especificar uma expressão regular. Esse guia, visualizado na Figura 6, possui agrupadores para facilitar a busca e o entendimento de cada tipo de metacaractere que é aceito no Regex101.

Figura 5 – Detalhamento de uma expressão regular



Fonte: elaborado pelo autor.

Figura 6 – Guia de referência sobre expressões regulares



Fonte: Dib (2014).

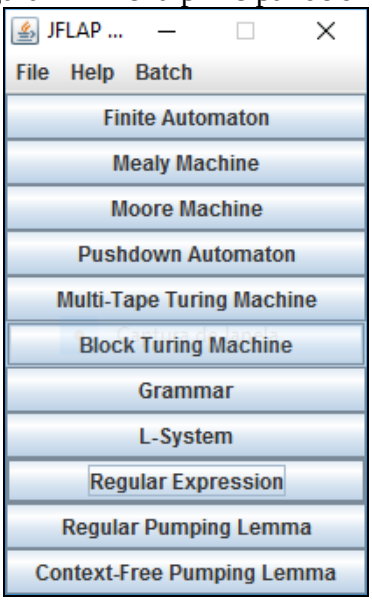
A aplicação é intuitiva e auxilia no entendimento de expressões regulares, mas não possui uma maneira de gerar exemplos de valores válidos e inválidos a partir de uma expressão regular e não possui uma área de aprendizado interativo.

2.3.2 JFLAP

O Java Formal Languages and Automata Package (JFLAP) foi criado por estudantes do Instituto Politécnico Rensselaer sob a orientação de Susan Rodger por volta de 1990. O software é basicamente um pacote de ferramentas gráficas para trabalhar com autômatos finitos, autômatos de pilha, máquinas de Turing, expressões regulares e gramáticas (RODGER; FINLEY, 2008). Os principais tópicos abordados pela ferramenta são apresentados no menu principal, visto na Figura 7.

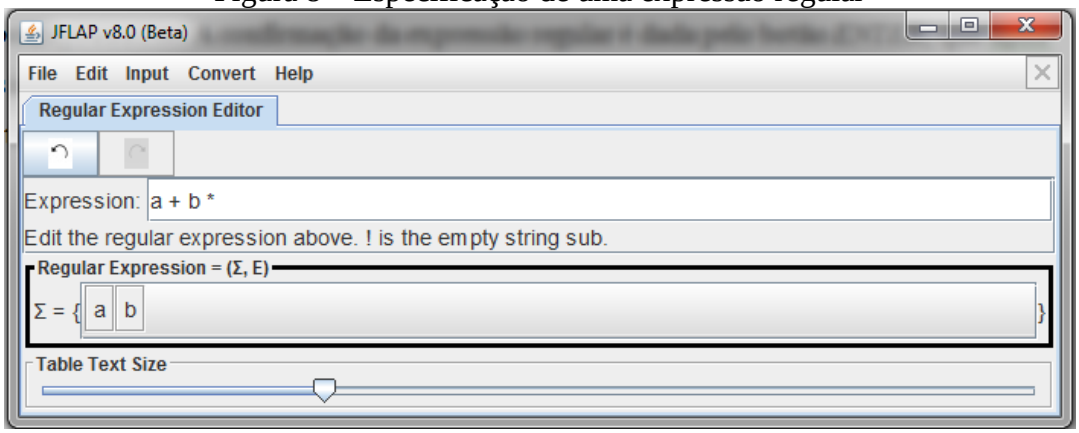
Ao clicar na opção `Regular Expression`, é apresentada a interface (Figura 8) para a especificação de uma expressão regular. Para isto, deve-se informar a expressão regular no campo `Expression`. Ao pressionar `ENTER`, o campo `Regular Expression = (Σ, E)` é preenchido com os símbolos do alfabeto, presentes na expressão regular.

Figura 7 – Menu principal do JFlap



Fonte: elaborado pelo autor.

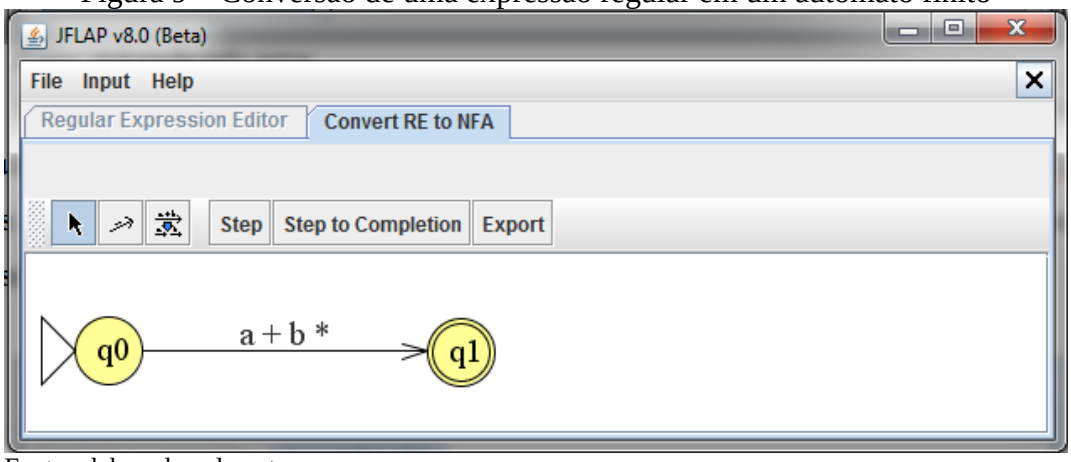
Figura 8 – Especificação de uma expressão regular



Fonte: elaborado pelo autor.

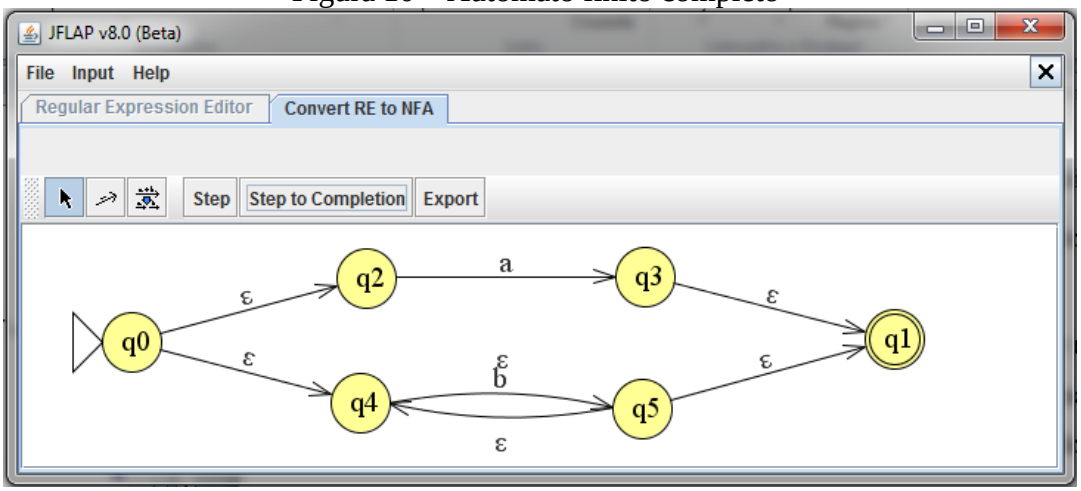
É possível converter uma expressão regular em um autômato finito. Na interface para especificação de expressões regulares, tem-se o menu `Convert` e a opção `RE to FA`, usada para criar um autômato finito corresponde à expressão regular informada anteriormente, conforme pode ser visto na Figura 9. Para visualizar o autômato finito completo, contendo transições rotuladas com a palavra vazia, também chamadas de ε-transições, basta clicar na opção `Step to Completion` (Figura 10). Ao clicar na opção `Export`, obtém-se uma nova tela (Figura 11) onde é possível editar o autômato finito, adicionando e removendo estados ou transições, ou marcando um estado como inicial ou final.

Figura 9 – Conversão de uma expressão regular em um autômato finito



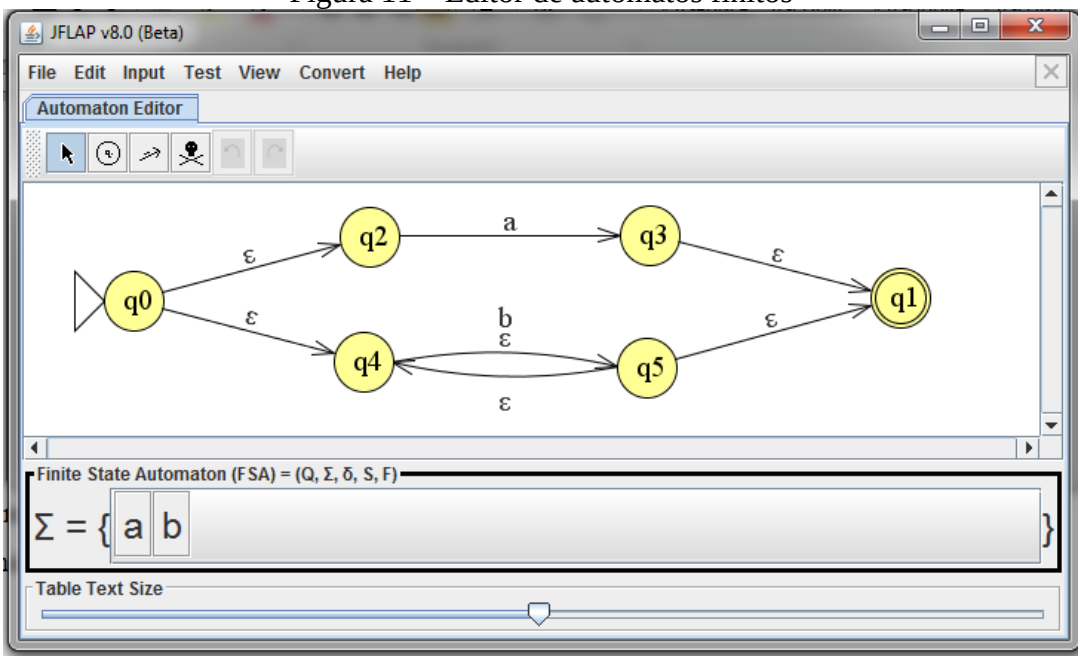
Fonte: elaborado pelo autor.

Figura 10 – Autômato finito completo



Fonte: elaborado pelo autor.

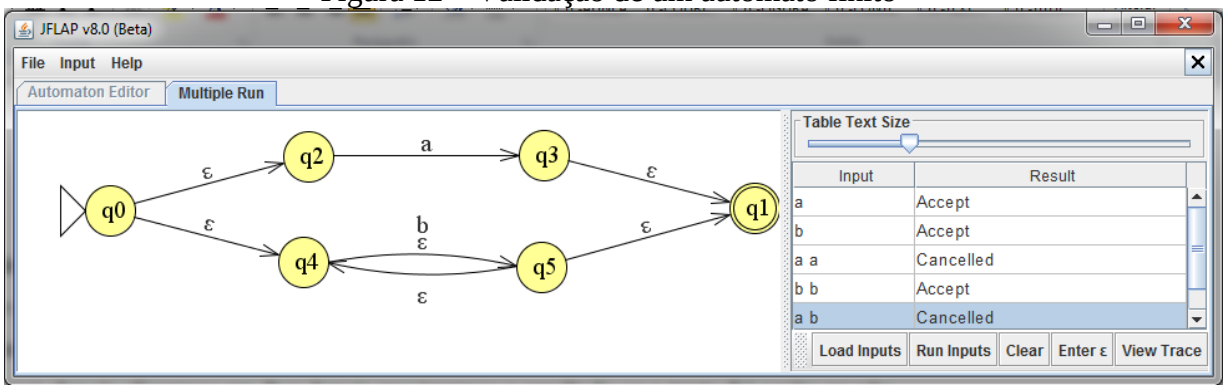
Figura 11 – Editor de autômatos finitos



Fonte: elaborado pelo autor.

Com o autômato finito gerado a partir da expressão regular especificada, é possível efetuar testes com valores de entrada para validar a expressão regular. No menu `Input`, deve-se selecionar a opção `Multiple Run` e uma nova tela será exibida (Figura 12) com um componente onde devem ser informados na coluna `Input` valores para serem validados. Ao clicar em `Run Inputs`, obtém-se o resultado, ou seja, se a sentença de entrada foi aceita ou não pelo autômato finito e, consequentemente, pela expressão regular. Nesse caso, é necessário ter conhecimento sobre autômatos finitos.

Figura 12 – Validação de um autômato finito



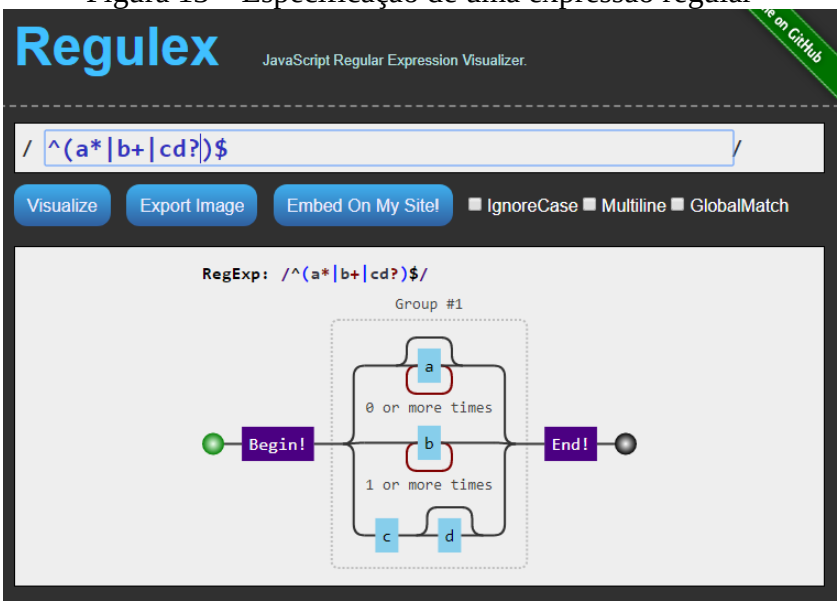
Fonte: elaborado pelo autor.

Considera-se o JFLAP uma ferramenta adequada para trabalhar com autômatos finitos. Já o suporte para expressões regulares limita-se basicamente na conversão de expressões regulares para autômatos finitos e na conversão de autômatos finitos determinísticos para expressões regulares.

2.3.3 REGULEX

O Regulex foi desenvolvido por Cheng (2014) e é uma ferramenta que possui como propósito representar uma expressão regular de forma ilustrativa, em um modelo semelhante a um diagrama de sintaxe. O site é simples e o seu uso também. Na página inicial o usuário deve informar uma expressão regular no único campo de texto disposto na tela e, conforme é especificada a expressão regular, a representação gráfica é atualizada. Na Figura 13 está definida a expressão regular `(a*|b+|cd?)` seguida de sua representação gráfica. A aplicação cria uma representação gráfica da expressão regular informada, porém esta é sua única funcionalidade.

Figura 13 – Especificação de uma expressão regular



Fonte: elaborado pelo autor.

3 DESENVOLVIMENTO

Nesta seção são abordados os requisitos funcionais e não funcionais levantados para o desenvolvimento da ferramenta (seção 3.1), seguidos da especificação do trabalho (seção 3.2) e da implementação (seção 3.3), que apresenta as técnicas e ferramentas utilizadas e a validação de respostas na resolução de exercícios. Depois é apresentada a operacionalidade da implementação, desde o gerenciamento efetuado pelo professor até a resolução dos exercícios pelo estudante, bem como o acompanhamento das atividades em tempo real pelo professor. Por fim, na seção 3.4, é mostrada a análise de resultados de testes obtidos a partir de uma avalição da ferramenta em sala de aula.

3.1 REQUISITOS

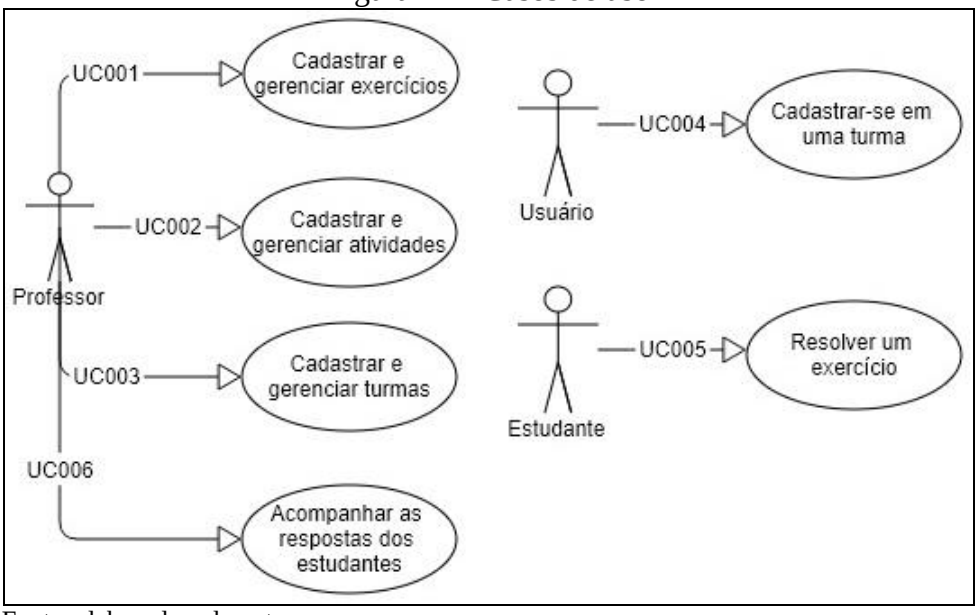
A ferramenta proposta neste trabalho deve:

a) permitir que o usuário cadastre-se através do Google e Facebook (Requisito Funcional - RF);
b) permitir a utilização de perfis de usuário – professor e estudante (RF);
c) permitir que o professor crie e gerencie turmas (RF);
d) permitir que o professor crie e gerencie atividades sobre expressões regulares (RF);
e) gerar um conjunto de valores válidos e inválidos para a atividade conforme a expressão regular especificada (RF);
f) permitir que o professor distribua as atividades para as turmas (RF);
g) permitir que os estudantes resolvam as atividades propostas (RF);
h) validar as atividades submetidas pelos estudantes conforme a resposta fornecida pelo professor (RF);
i) mostrar o andamento das atividades em tempo real para agilizar o processo de validação (RF);
j) seguir o protocolo OAuth para autenticação (Requisito Não Funcional - RNF);
k) ser desenvolvida em HTML5, CSS3 e JavaScript (RNF);
l) utilizar o MongoDB para persistência de dados (RNF);
m) utilizar WebSocket para a comunicação em tempo real (RNF);
n) ter suporte para os navegadores Chrome e Firefox (RNF).

3.2 ESPECIFICAÇÃO

A ferramenta foi desenvolvida como uma aplicação cliente-servidor. O professor pode cadastrar exercícios, atividades e turmas e acompanhar a resolução de atividades pelos estudantes. Já os estudantes podem resolver os exercícios das atividades em andamento propostas para a sua turma. A partir disso, podem ser visualizados os casos de uso na Figura 14.

Figura 14 – Casos de uso



Fonte: elaborado pelo autor.

O uso da ferramenta começa pelo professor, que pode: gerenciar o cadastro de exercícios (caso de uso UC001), gerenciar o cadastro de atividades (UC002) e gerenciar o cadastro de turmas (UC003). Para criar um exercício, o professor deve informar: (1) a descrição ou enunciado; (2) a expressão regular, solução do enunciado proposto; (3) um conjunto de etapas para auxiliar os estudantes na resolução do exercício. Essas etapas são compostas por palavras válidas e inválidas, com uma determinada quantidade de símbolos, que serão apresentadas ao estudante no momento da resolução do exercício. Por exemplo, considerando o seguinte enunciado $L=\{w \mid w \in \{a, b\}^+ \text{ e } |w| \text{ é par}\}$, tem-se que a linguagem L é composta por palavras que possuem quantidade par de símbolos (a ou b). Assim, o professor pode determinar que o exercício pode ser resolvido pela expressão regular `((a|b){2})+` ou equivalente. Ainda, se o professor definir que a 1ª etapa será composta por palavras com até três símbolos, serão apresentadas para o estudante as palavras `aa`, `ab`, `ba` e `bb` como válidas e as palavras ε (palavra vazia), `a`, `b`, `aaa`, `aab`, `aba`, `abb`, `baa`, `bab`, `bba` e `bbb` como inválidas. Considerou-se que a visualização do conjunto de palavras válidas e inválidas pode auxiliar o estudante na resolução do exercício.

Uma vez definidos os exercícios, o professor pode agrupá-los em atividades ou listas de exercícios, permitindo entregar o mesmo conjunto de exercícios para diversas turmas. O professor deve então definir quais atividades uma turma deverá resolver e qual o prazo de entrega de cada atividade. Depois de efetuar o gerenciamento destas entidades, o professor pode disponibilizar para qualquer um o link de acesso de uma turma, permitindo um usuário tornar-se um estudante desta turma (UC004). O ator usuário é qualquer pessoa que acessa a ferramenta e não está vinculado a uma turma. É considerado somente um estado transitório, até acessar o link de acesso da turma, tornando-se um estudante e, assim, podendo resolver exercícios (UC005) referentes aquela turma. Como a resolução de exercícios é considerada a principal funcionalidade da ferramenta, o caso de uso encontra-se detalhado no Quadro 2. Por fim, o professor pode efetuar o acompanhamento por e-mail do envio de respostas dos estudantes (UC006). O professor também recebe notificações na própria ferramenta e tem acesso ao detalhamento da resolução dos exercícios, atualizado em tempo real.

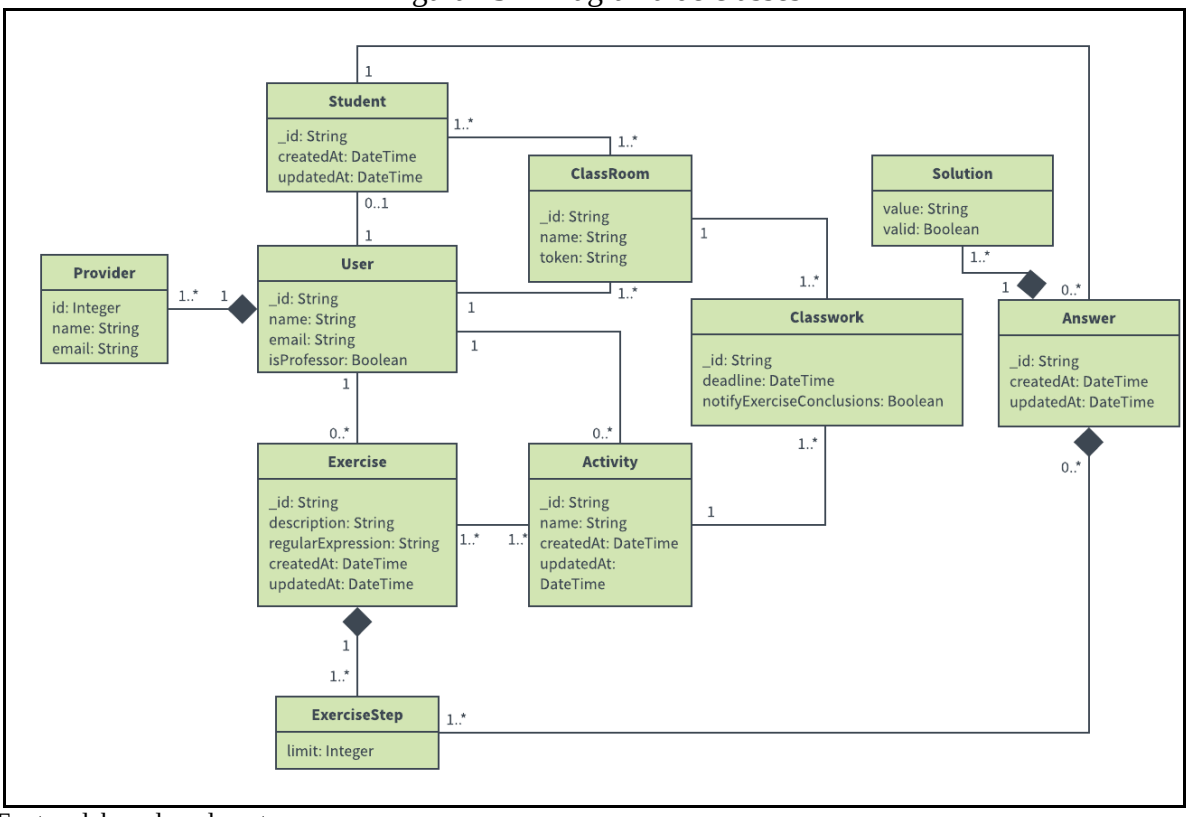
Na Figura 15 tem-se o diagrama com as classes da ferramenta. Os exercícios foram modelados como `Exercise`, que possui a expressão regular e o enunciado do exercício, além de uma lista de etapas (`ExerciseStep`), onde cada `ExerciseStep` indica a quantidade de símbolos que a etapa possui, sendo essa quantidade usada posteriormente para gerar as palavras válidas e inválidas. As atividades correspondem à `Activity`, que possui o nome da atividade e uma lista de exercícios. As turmas são representadas por `ClassRoom`, que possui o nome da turma, o *token* de acesso e uma lista de `Classworks`, onde cada `Classwork` é composto por uma atividade, o prazo de entrega e a indicação de envio de e-mail na conclusão de exercícios. As respostas enviadas pelos estudantes foram especificadas como `Solution`. Cada resposta é cadastrada com uma indicação se ela é válida ou não e, para facilitar na busca pelas soluções por exercício, `Answer` agrupa essas respostas por exercício. Um usuário (`User`) possui um nome e um email, além de um vínculo com um `Provider`, que contém as informações do serviço utilizado para se cadastrar na ferramenta (Google ou Facebook).

Quadro 2 – Caso de uso: `Resolver exercício`

CASO DE USO:	Resolver exercício
BREVE DESCRIÇÃO:	Permite que o estudante resolva atividades propostas para uma turma pelo professor.
ATOR(ES):	Estudante
FLUXO PRINCIPAL:	1. O estudante seleciona uma atividade entre as atividades em andamento relacionadas para a sua turma. 2. O estudante seleciona um exercício da atividade escolhida no passo anterior. 3. Com base na descrição do exercício e nas palavras válidas e inválidas dispostas, o estudante escreve uma expressão regular para o exercício. 4. A ferramenta verifica se a expressão regular é válida. Em caso negativo, faz o tratamento de erro. Em caso positivo, permite que o estudante envie a resposta. 5. O estudante submete a resposta especificada para o exercício. 6. A ferramenta notifica o estudante que a resposta está correta. 7. O estudante pode repetir os passos anteriores para os demais exercícios e atividades da sua turma.
FLUXO ALTERNATIVO 01:	No passo 4, se a expressão regular não for válida, a ferramenta emite uma mensagem com o erro detectado para as seguintes situações: - se possuir algum erro léxico ou sintático; - se usar símbolos que não pertencem ao alfabeto extraído do exercício; - se não aceitar todas as palavras válidas relacionadas para a etapa; - se aceitar alguma das palavras inválidas relacionadas para a etapa.
FLUXO ALTERNATIVO 02:	No passo 5, se o professor estiver acessando a ferramenta, independente da resposta do estudante estar correta, uma notificação será exibida no canto inferior direito da ferramenta, informando que o estudante acabou de submeter uma solução.
FLUXO ALTERNATIVO 03:	No passo 6, se a resposta submetida pelo estudante estiver correta e o professor configurou a ferramenta para ser notificado das conclusões dos exercícios por e-mail, o professor receberá um e-mail com o nome da turma e do estudante que enviou a resposta, e a descrição do exercício junto da resposta do estudante.
FLUXO ALTERNATIVO 04:	No passo 7, se a resposta submetida pelo estudante não estiver correta, ou seja, não for equivalente à expressão regular previamente informada pelo professor, a ferramenta emite uma mensagem e envia outras palavras válidas e inválidas, a partir da especificação feita pelo professor, para auxiliar a resolução dos exercícios, e o estudante retorna ao passo 3 do fluxo principal. Se não houverem palavras válidas e inválidas a serem apresentadas, a ferramenta informa que não existe mais palavras para auxiliar na resolução do exercício.
PRÉ-CONDIÇÕES	O estudante deve acessar a sua turma com atividades cadastradas na ferramenta através do *token* de acesso fornecido pelo professor.
PÓS-CONDIÇÕES	Exercícios resolvidos.

Fonte: elaborado pelo autor.

Figura 15 – Diagrama de classes



Fonte: elaborado pelo autor.

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas as ferramentas e técnicas utilizadas no desenvolvimento, assim como a operacionalidade da ferramenta.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento dessa ferramenta foram utilizadas as linguagens JavaScript, HTML e CSS. Foi utilizado o React para os controles visuais da interface, o Redux para efetuar o gerenciamento de estado da aplicação e o Bootstrap para criar um site responsivo e com suporte para os principais navegadores utilizados. Do lado do servidor, foi utilizado o Node.js para permitir escrever código em JavaScript, o Express para servir de camada de comunicação REST e a biblioteca Mongoose para manipular os dados do MongoDB, que foi escolhido como banco de armazenamento de dados por três motivos: (1) possuir diversas bibliotecas que facilitam seu manuseio com o JavaScript; (2) permitir um controle mais flexível dos dados; (3) possuir um baixo custo para ser mantido em produção, visto que o desenvolvimento deste trabalho está disponível para acesso na plataforma Heroku (Apêndice B). Para permitir usar as funcionalidades do ECMAScript 6 e do JSX na aplicação, foi utilizado o webpack em conjunto do Babel, tanto para montar um ambiente de

desenvolvimento quanto para servir de ferramenta de compilação para gerar os arquivos finais para ambiente de produção. Por último, foi utilizado o PegJS para especificar uma linguagem para escrever expressões regulares. Essa definição gera um *parser* que valida léxica e sintaticamente uma entrada de texto e, caso nenhum erro seja encontrado, retorna o alfabeto extraído da expressão regular.

3.3.2 Validação de respostas na resolução de exercícios

A funcionalidade principal da aplicação é a validação de expressões regulares, que serve para auxiliar o estudante a resolver um exercício proposto pelo professor. A validação possui duas etapas principais, uma efetuada durante o momento que o estudante está resolvendo o exercício, outra efetuada depois que o estudante submete uma resposta.

A primeira etapa de validação é realizada por duas funções. A primeira função chama-se `regex` (Quadro 3) e identifica erros léxicos ou sintáticos. A função basicamente delega o texto para um `parser` (linha 3) e, no caso de detecção de erros, formata a mensagem incluindo a coluna em que o erro foi encontrado (linha 6).

Quadro 3 – Função `regex`

```
1  const regex = value => {
2    try {
3      if (value) parser.parse(value);
4      return undefined;
5    } catch (error) {
6      return `Coluna ${error.location.start.column}: ${error.message}`;
7    }
8  };
```

Fonte: elaborado pelo autor.

O `parser` é gerado a partir do PegJS usando a gramática especificada no Quadro 4. Essa gramática foi projetada para capturar o alfabeto de uma expressão regular simples, que é composto por símbolos (letras minúsculas e dígitos) encontrados na expressão, descartando adequadamente dígitos usados em ocorrências, que não são considerados como parte do alfabeto de uma expressão regular. Nas linhas 24 a 28 tem-se as regras que definem lexemas numéricos e letras minúsculas, denominadas, respectivamente de `Integer` e `LowerCase`. Ambas as regras são simples, não possuem quantificadores, alternativa ou concatenação, efetuando o reconhecimento de apenas um caracter. Já na linha 21 é definida a regra `Alphanumeric` para agrupar o alfabeto disponível para a especificação de expressões regulares e na linha 19 é usado o operador `+` para denotar que o `AlphanumericExpression` é composto por uma ou mais ocorrências de `Alhpanumeric`. Isso significa que `AlphanumericExpression` reconhecerá uma lista de *tokens*. Na linha 16, a regra `CaptureExpression` nomeia uma variável `symbols`, permitindo que os demais caracteres

sejam descartados e somente o retorno de `Expression+` seja analisado. Na linha 9 é efetuado o descarte do `RepetitionPart?` porque dígitos utilizados na especificação de uma ocorrência não são considerados como parte do alfabeto de uma expressão regular. Na linha 5 tem-se a união das expressões específicas para permitir que diversas expressões sejam alinhadas umas com as outras. Por último, na linha 2 é feita a normalização dos símbolos encontrados pela função `normalize` que remove símbolos repetidos e ordenar a lista de forma alfabética.

Quadro 4 – Gramática descrita usando PegJS

```
1  Grammar
2    = symbols:Expression+ { return normalize(symbols) }
3
4  Expression
5    = left:TermExpression rights:("|" TermExpression)* { return
6  left.concat(rights ? flatten(rights.map(right => right[1])) : []) }
7
8  TermExpression
9    = symbols:(AlphanumericExpression / CaptureExpression) RepetitionPart? {
10 return symbols }
11
12 RepetitionPart
13  = ("+" / "*" / "{" Integer+ "}")? "?"?
14
15 CaptureExpression
16   = "(" symbols:Expression+ ")" { return flatten(symbols) }
17
18 AlphanumericExpression
19   = Alphanumeric+
20
21 Alphanumeric
22   = (Integer / LowerCase)
23
24 Integer "integer"
25   = [0-9]
26
27 LowerCase "lower case"
28   = [a-z]
```

Fonte: elaborado pelo autor.

A segunda função `solutionIsValid` (Quadro 5) da primeira etapa identifica problemas relacionados à resposta do exercício em si. Primeiro é usada na linha 3 a instrução `symbolParser.parse` para identificar os símbolos que estão sendo utilizados. Se algum dos símbolos extraídos não estiver na lista de símbolos da solução do exercício, previamente cadastrada pelo professor (linhas 5 e 6), um erro é lançado para o estudante (linha 7). Se os símbolos forem válidos, é verificado se todas as palavras válidas apresentadas são aceitas pela resposta (linhas 12 até 19) e se todas as palavras inválidas são rejeitadas (linhas 21 até 28). Em caso positivo, o estudante poderá submeter a resposta para o servidor.

Quadro 5 – Função `solutionIsValid`

```
1  const solutionIsValid = currentStep => solution => {
2    if (solution) {
3      const lettersFromSolution = symbolParser.parse(solution);
4
5      if (lettersFromSolution.some(letter =>
6  !currentStep.symbols.includes(letter))) {
7        return 'A expressão regular especificada usa símbolos que não pertencem
8  ao alfabeto da linguagem';
9      }
10   }
11
12   const allInvalid = currentStep.words.invalids.every(
13     words => words.every(word => !toRegex(solution).test(word)),
14   );
15
16   if (!allInvalid) {
17     return 'Algumas palavras inválidas foram capturadas pela expressão
18 regular';
19   }
20
21   const allValid = currentStep.words.valids.every(
22     words => words.every(word => toRegex(solution).test(word)),
23   );
24
25   if (!allValid) {
26     return 'Algumas palavras válidas não foram capturadas pela expressão
27 regular';
28   }
29
30   return undefined;
31 };
```

Fonte: elaborado pelo autor.

A segunda etapa de validação é realizada no servidor pela função `hasSameMinimization` (Quadro 6). A validação basicamente verifica se a resposta enviada pelo estudante é equivalente à expressão regular (solução) cadastrada pelo professor no exercício. Primeiro é verificado se a resposta é exatamente igual à solução (linha 4). Se não for, é usada a biblioteca `regex2dfa` para transformar as expressões regulares (resposta e solução) em autômatos finitos determinísticos mínimos (linha 5). Considerando que toda linguagem regular possui apenas um autômato finito determinístico mínimo (MENEZES, 2000), caso os autômatos resultantes sejam iguais, significa que as expressões regulares possuem equivalência e então é a resposta do aluno é válida para o exercício.

Quadro 6 – Função `hasSameMinimization`

```
1  const { regex2dfa } = require('regex2dfa/regex2dfa.js');
2
3  const hasSameMinimization = (exercise, solution) => (
4    exercise.regularExpression === solution
5    || regex2dfa(exercise.regularExpression) === regex2dfa(solution)
6  );
```

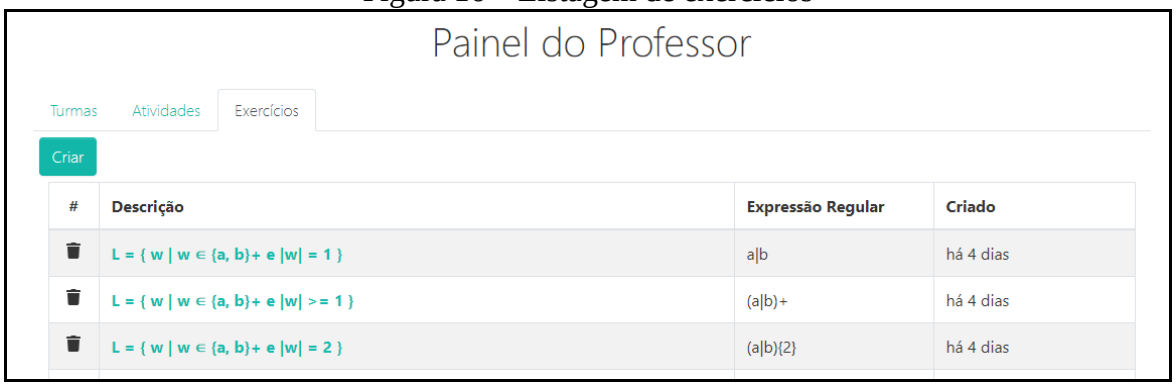
Fonte: elaborado pelo autor.

3.3.3 Operacionalidade da implementação

A ferramenta, denominada Learn REGEX, foi desenvolvida como uma aplicação web com suporte para os principais navegadores. Então, em princípio, qualquer usuário pode

utilizar a ferramenta desde que tenha uma conexão com a internet. O acesso é feito através de uma conta Facebook ou Gmail, sendo esse um pré-requisito para acessar a ferramenta. A tela principal para professores, mostrada na Figura 16, possui três abas: `Turmas`, `Atividades` e `Exercícios`. Por padrão, a ferramenta abre com a aba `Exercícios` selecionada. Cada aba contém: (1) o botão `Criar` no canto superior esquerdo, que permite efetuar o cadastro do respectivo registro (turma, atividade ou exercício); (2) uma listagem com as informações principais do registro; (3) um ícone de lixeira, que permite efetuar a exclusão do registro da linha associada; (4) um link na coluna `Descrição` (ou `Nome`, no caso de turmas), que permite navegar para o registro em si.

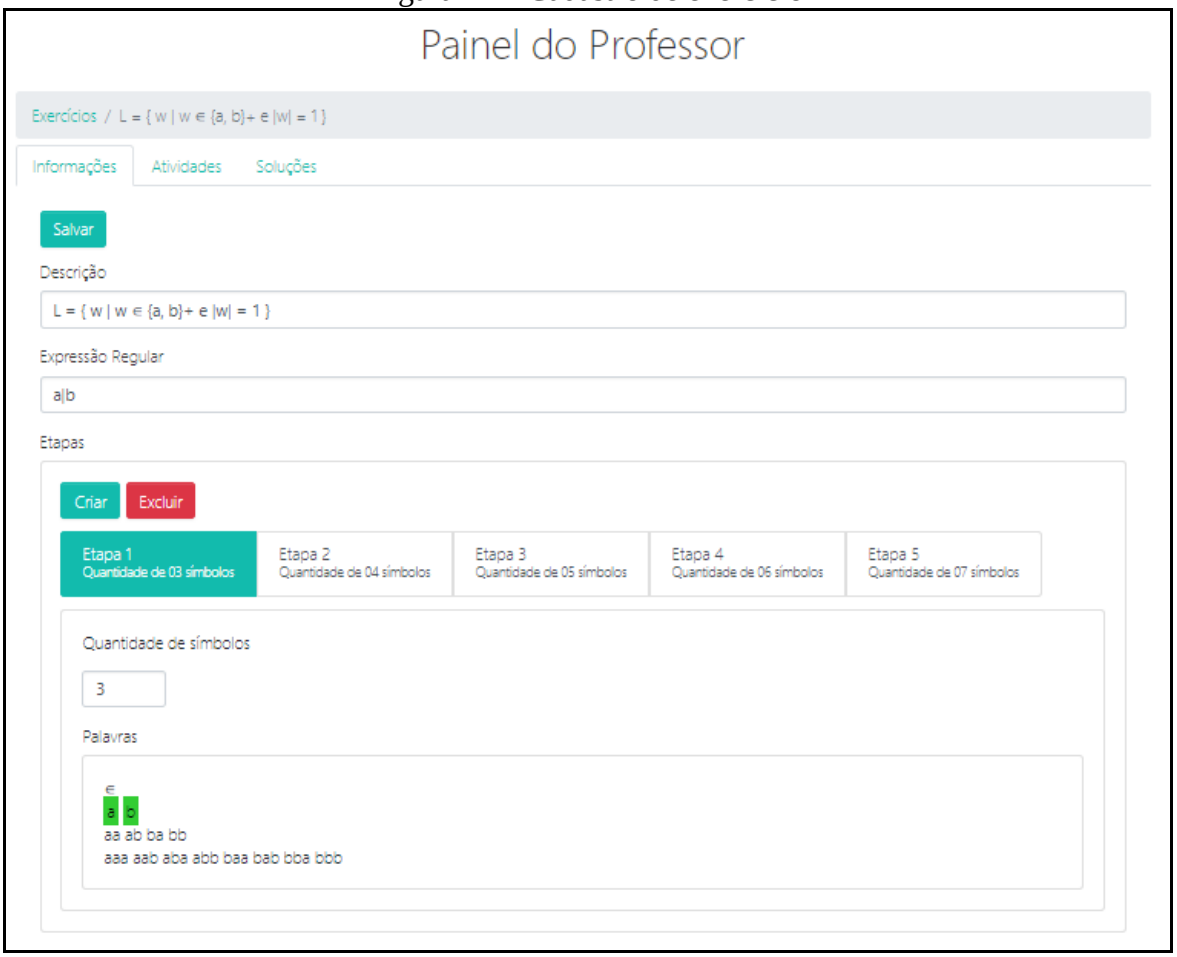
Figura 16 – Listagem de exercícios



Fonte: elaborado pelo autor.

No caso da Figura 16, ao clicar na descrição de um exercício, será exibida a tela de edição do exercício, conforme a Figura 17. Para exercícios tem-se: (1) a aba `Informações`, que contém as informações do exercício; (2) a aba `Atividades`, que lista as atividades das quais o exercício faz parte; (3) a aba `Soluções`, que contém expressões regulares válidas e inválidas submetidas pelos estudantes. Na aba `Informações` são apresentados: (1) o campo `Descrição`, com o enunciado do exercício, que será apresentado para o estudante no momento da resolução de atividades; (2) o campo `Expressão Regular`, que é a solução do exercício; (3) um conjunto de etapas. Cada etapa possui o campo `Quantidade de símbolos`, que indica o número máximo de símbolos que as palavras possuem e, consequentemente, limita a quantidade de palavras que serão apresentadas ao estudante para auxiliar na resolução de um exercício. As palavras geradas podem ser vistas logo abaixo no campo `Palavras`, sendo que as palavras que estão destacadas em verde são as palavras aceitas pela expressão regular especificada e as demais são palavras não aceitas pela expressão. A ideia por trás das etapas é: a cada resposta errada do estudante, apresentar um conjunto maior de palavras válidas e inválidas que possam auxiliá-lo na resolução do exercício.

Figura 17 – Cadastro de exercício



Fonte: elaborado pelo autor.

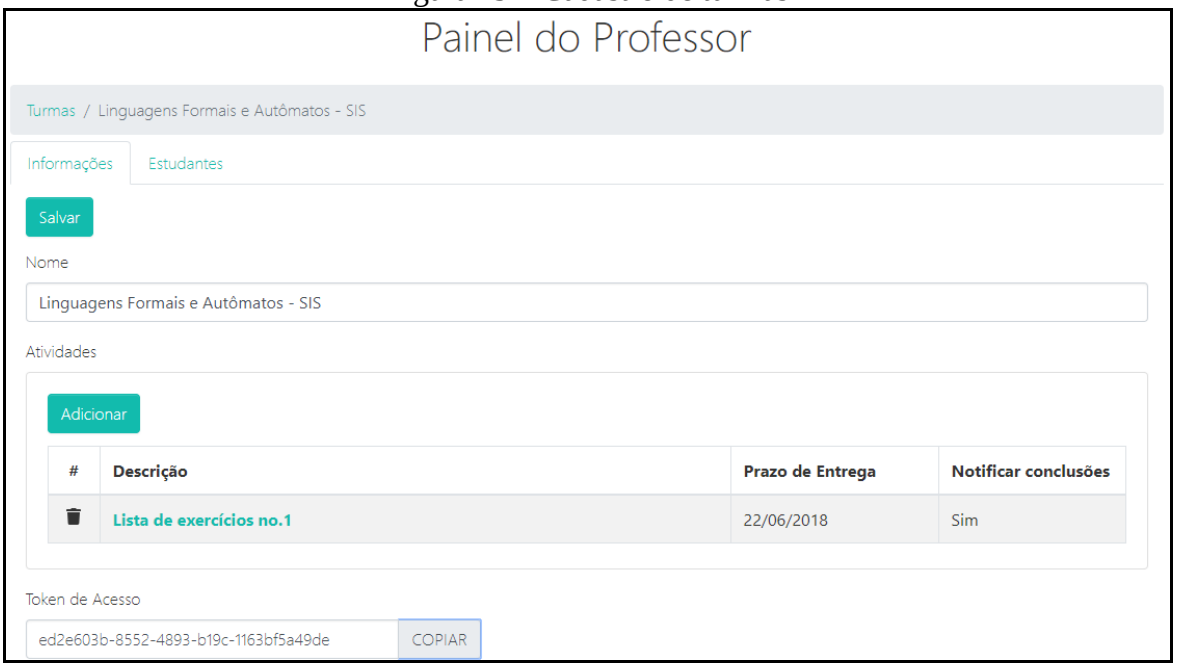
Na aba `Soluções`, conforme Figura 18, são apresentadas duas listagens, uma com as expressões regulares válidas e outra com as inválidas, ou seja, resposta corretas e erradas submetidas pelos estudantes. Em ambas as listagens existe a coluna `Ocorrências` que indica a quantidade de vezes que aquela solução foi submetida. Ao clicar em alguma das expressões, é apresentada a listagem dos estudantes que a submeteram. Ambas as telas, tanto da aba `Soluções` quanto da listagem de estudantes por expressão regular submetida, funcionam em tempo real, isto é, qualquer resposta submetida irá atualizar automaticamente estas listagens.

Figura 18 – Listagem de soluções

Fonte: elaborado pelo autor.

O cadastro de atividades é o mais simples. Possui somente o campo `Descrição` e uma listagem de `Exercícios`. A ideia é que uma atividade seja basicamente um agregador de exercícios. Já no cadastro de turmas mostrado na Figura 19, tem-se a aba `Informações` e a aba `Estudantes`.

Figura 19 – Cadastro de turmas

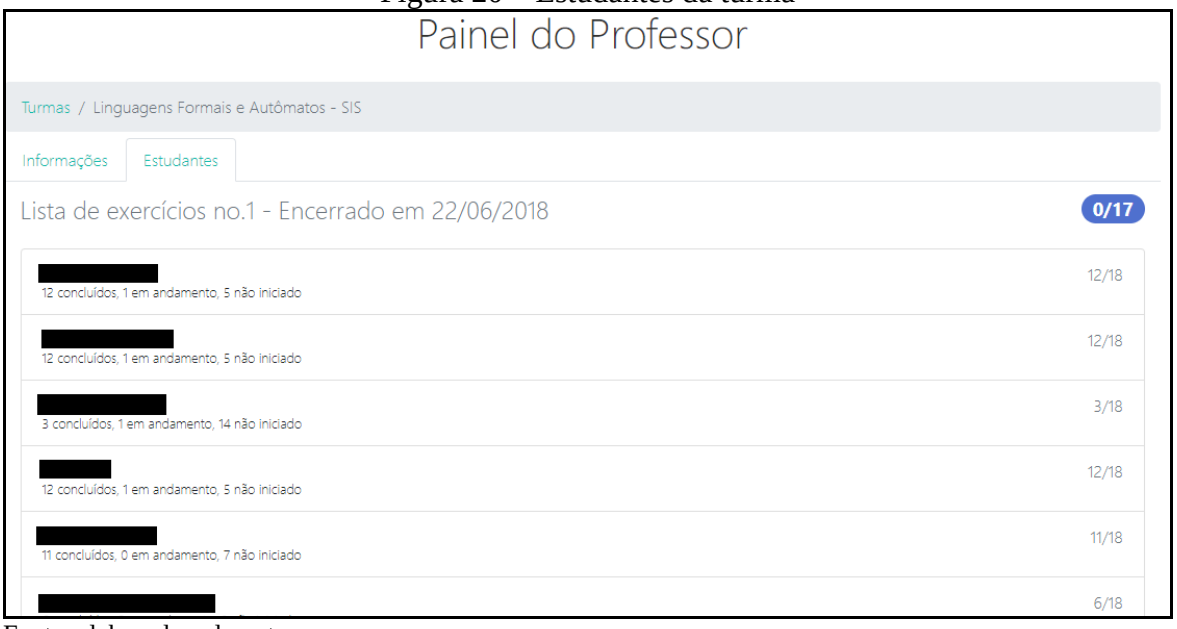


Fonte: elaborado pelo autor.

Na aba `Informações`, são apresentados: o campo `Nome` da turma, uma listagem de `Atividades` e um `Token de Acesso` que possui um botão `COPIAR` ao lado direito. Cada atividade possui um prazo de entrega e se devem ser enviadas notificações via e-mail para o professor quando os estudantes submeterem respostas corretas para os exercícios. Ao clicar no botão `COPIAR`, um link para a turma é copiado para a área de transferência para que o

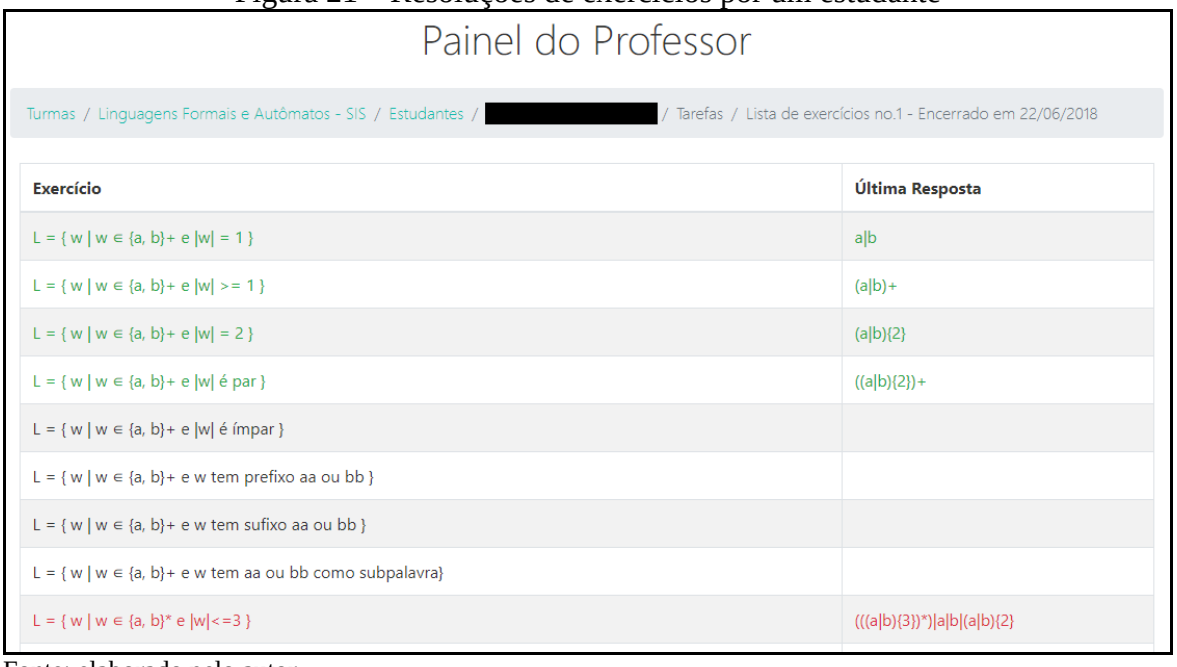
professor possa fornece-lo aos estudantes da turma. Assim, qualquer estudante que acessar este link será registrado na turma. A aba `Estudantes`, conforme visualizado na Figura 20, apresenta, para cada atividade, um resumo da resolução dos exercícios pelos estudantes. Tem-se: (1) o número de estudantes que concluíram todos os exercícios da atividade e o número de alunos que a estão realizando (no canto superior direito); (2) para cada um dos estudantes, quantos exercícios foram concluídos, quantos estão em andamento e quantos não foram iniciados ainda. É possível ainda ver o desempenho de um estudante ao clicar no nome dele (Figura 21).

Figura 20 – Estudantes da turma



Fonte: elaborado pelo autor.

Figura 21 – Resoluções de exercícios por um estudante

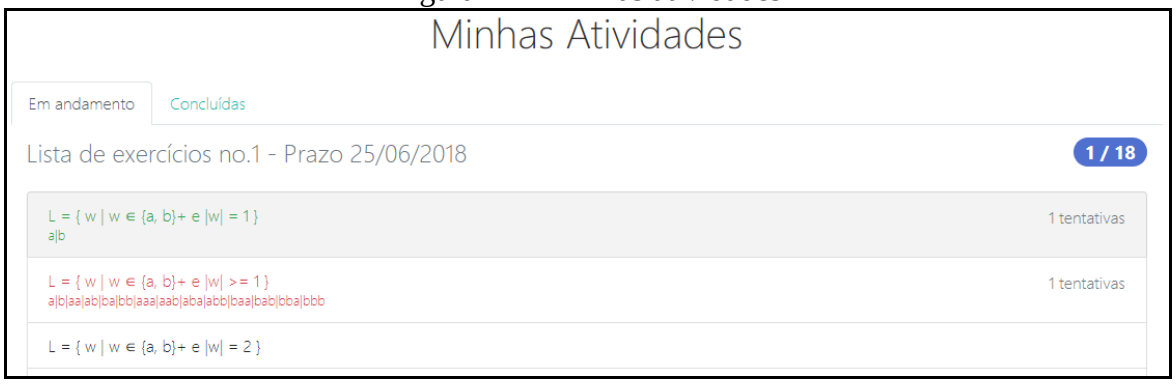


Fonte: elaborado pelo autor.

Na Figura 21, é mostrada uma listagem dos exercícios de um determinado estudante e a última resposta enviada para cada um dos exercícios, além de destacar em verde as respostas corretas e em vermelho as respostas incorretas. Ambas as telas também funcionam em tempo real.

A tela principal dos estudantes (Figura 22) apresenta duas abas. A aba `Em andamento` mostra as atividades que ainda estão dentro do prazo de entrega estipulado pelo professor, enquanto a aba `Concluídas` mostra as atividades com o prazo encerrado. Em ambas as abas, a apresentação é a mesma, tem-se a relação dos exercícios de cada uma das atividades, a descrição da atividade e o prazo de entrega. No canto superior direito, é possível visualizar o número de exercícios corretamente resolvidos e o número de exercícios da atividade. Ainda nessas abas, os exercícios com texto em verde indicam que o exercício foi resolvido, exercícios com texto em preto não foram iniciados, enquanto os com texto em vermelho indicam que uma resposta foi enviada, mas não é válida.

Figura 22 – Minhas atividades



Fonte: elaborado pelo autor.

Os exercícios de atividades que ainda estão em andamento são clicáveis. Ao clicar em um exercício, é apresentada a tela de resolução de exercício, conforme a Figura 23. O campo `Expressão Regular` é o único campo editável na tela. Logo abaixo do campo é apresentado o conjunto de palavras válidas e inválidas para o exercício. Conforme a expressão regular é informada, qualquer palavra capturada pela expressão é marcada em verde nos campos abaixo. Caso sejam detectados erros na resposta, tais como a definição de uma expressão regular inválida, falta de captura de palavras válidas ou a captura indevida de palavras inválidas, o campo `Expressão Regular` tem a borda alterada para vermelho e logo abaixo do campo é apresentada uma mensagem indicando o problema encontrado. O botão `Enviar Solução` só é habilitado quando nenhum dos problemas relacionados anteriormente for encontrado, permitindo que o estudante submeta a resposta para análise. Assim sendo, caso a expressão regular submetida não seja equivalente à solução do exercício, e se houver uma

próxima etapa cadastrada para o exercício, o estudante é notificado que a resposta não está correta e novas palavras são adicionadas aos campos de palavras válidas e inválidas para auxiliar na resolução do exercício. Se não houver uma nova etapa, então somente é informado que a resposta não está correta e que não existem mais etapas cadastradas para auxiliar na resolução do exercício. Se a resposta estiver correta, o estudante é informado que finalizou o exercício com sucesso e é redirecionado para a listagem de atividades em andamento.

Figura 23 – Resolução de um exercício

Fonte: elaborado pelo autor.

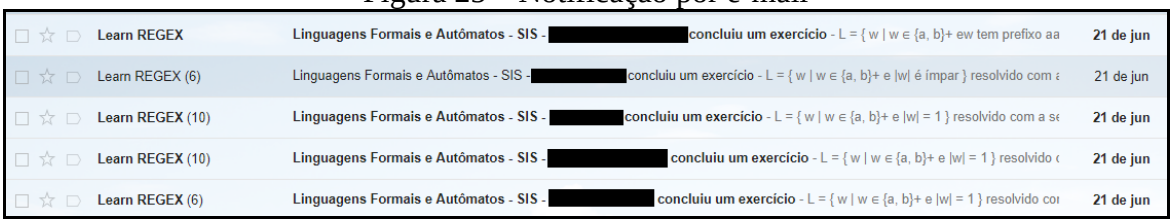
Por fim, toda vez que uma resposta é enviada para o servidor, o professor é notificado em tempo real que um estudante acabou de enviar uma resposta. Essa notificação é apresentada no canto inferior direito da tela do professor, independente de qual tela ele esteja no momento, conforme a Figura 24.

Figura 24 – Notificação em tempo real

Fonte: elaborado pelo autor.

Adicionalmente, caso o professor tenha informado no cadastro da turma para ser notificado por e-mail sobre as conclusões de exercícios desta atividade, se a resposta estiver correta, um e-mail é enviado com o nome da turma, o nome do estudante, a descrição do exercício e a resposta submetida, conforme a Figura 25.

Figura 25 – Notificação por e-mail



Fonte: elaborado pelo autor.

3.4 ANÁLISE DOS RESULTADOS

Nesta seção são abordados os resultados do trabalho. Primeiramente é descrito como foi efetuada a coleta de dados de um teste prático em sala de aula com estudantes da disciplina de Linguagens Formais e Autômatos do curso de Sistemas de Informação. Depois é feita uma análise dos resultados sobre a usabilidade da ferramenta e, por fim, uma análise das funcionalidades e aplicabilidades da ferramenta.

3.4.1 Teste prático e coleta de dados

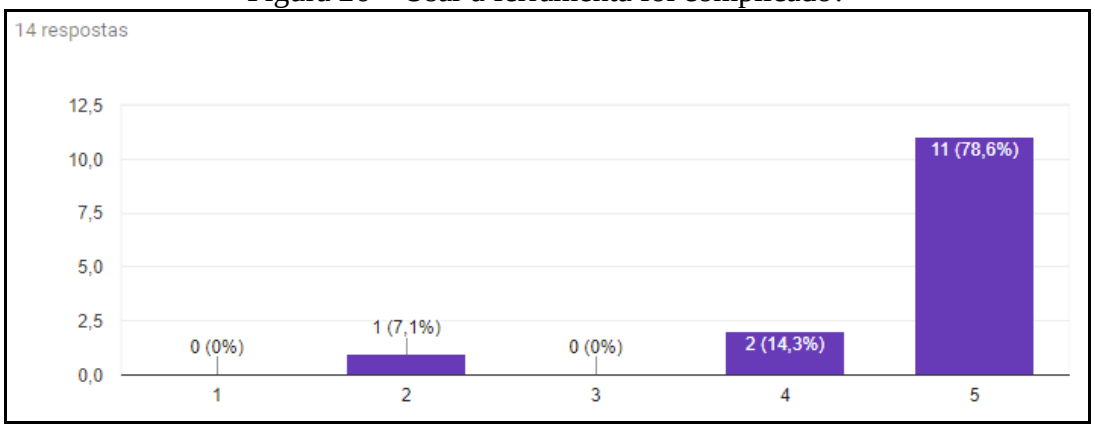
Como o foco deste trabalho é servir como uma ferramenta de auxílio para o aprendizado de expressões regulares em sala de aula, em 21 de junho de 2018, entre às 18:40 e 20:10, foi aplicado um teste prático com 14 estudantes do curso de Sistemas de Informação. O teste consistiu na resolução de uma atividade contendo 18 exercícios sobre expressões regulares durante uma hora. Depois os estudantes tiveram 30 minutos para responder um questionário (Apêndice A) que serviu para coletar os dados sobre os ambientes de testes utilizados, sobre a usabilidade e funcionalidades da ferramenta. Quanto aos ambientes de testes usados, uma análise das respostas encontra-se no Apêndice A. O questionário foi criado usando o Google Docs, que permite gerar gráficos com base nas respostas e garante a anonimidade de cada um dos estudantes.

3.4.2 Usabilidade

A ferramenta foi desenvolvida com foco em simplicidade e praticidade para a resolução dos exercícios pelos estudantes. Por causa disto foi desenhada para condensar as informações em uma única tela para os estudantes, possuindo somente um mecanismo de abas para trocar a visualização entre as atividades em andamento e as atividades concluídas. Com

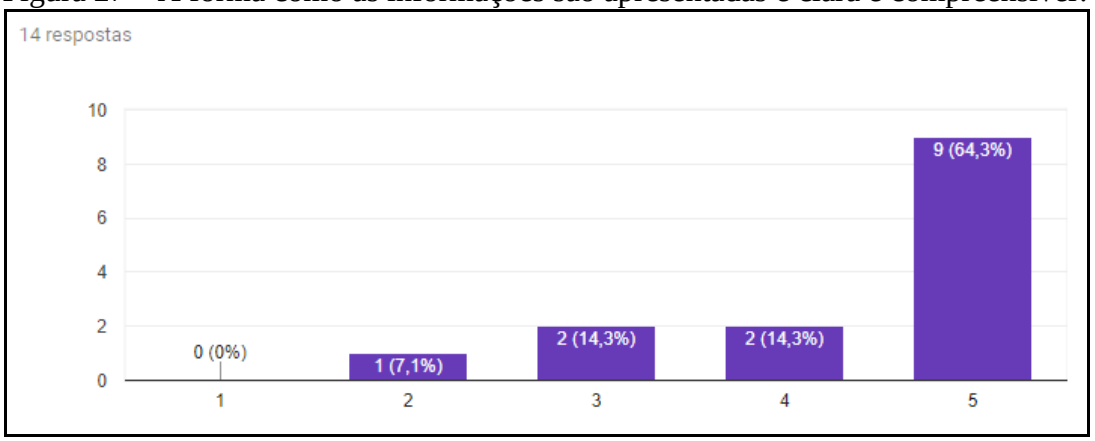
base nesses princípios, três questões foram elaboradas para identificar se a ferramenta conseguiu atingir esses objetivos: (1) Usar a ferramenta foi complicado? (gráfico com respostas na Figura 26); (2) A forma como as informações são apresentadas é clara e compreensível? (Figura 27); (3) É fácil navegar entre as telas da ferramenta? (Figura 28). Considerando que quanto maior o valor da resposta (muito ruim – 1 até muito bom – 5), melhor a nota, pode-se afirmar pelos gráficos que pelo menos 78,6% dos estudantes acha o uso da ferramenta simples e prático.

Figura 26 – Usar a ferramenta foi complicado?



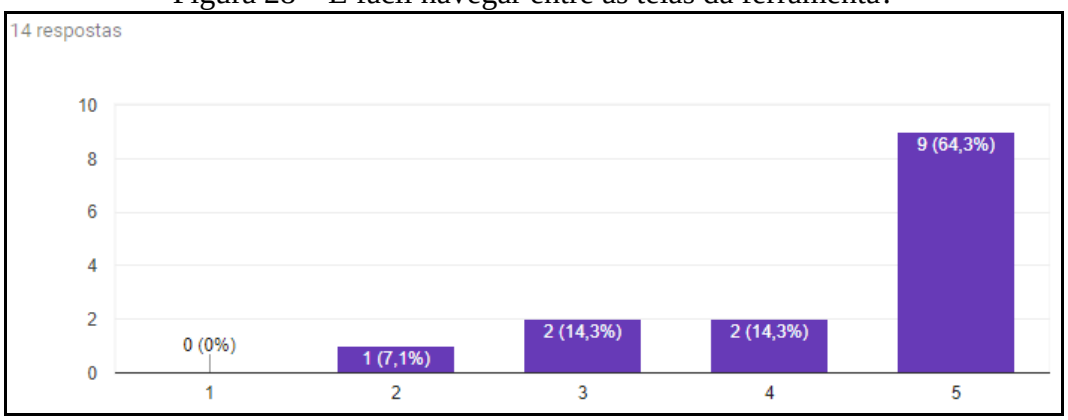
Fonte: elaborado pelo autor.

Figura 27 – A forma como as informações são apresentadas é clara e compreensível?



Fonte: elaborado pelo autor.

Figura 28 – É fácil navegar entre as telas da ferramenta?



Fonte: elaborado pelo autor.

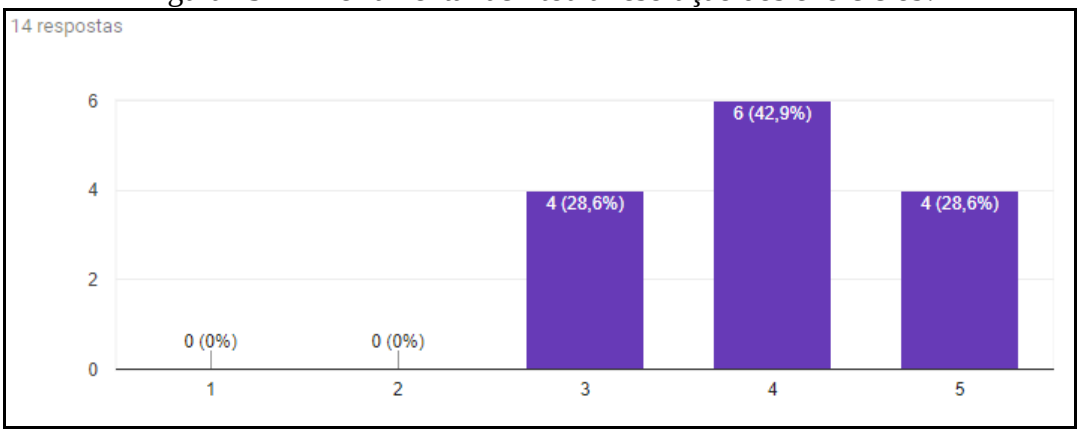
3.4.3 Funcionalidades e aplicabilidades

Quanto às funcionalidades, o primeiro questionamento feito foi sobre o mecanismo de acesso, pois é possível acessar a ferramenta somente através do Facebook ou do Google, sem a opção de cadastro de *login* na própria ferramenta. Foi questionado se a falta dessa funcionalidade incomodou os estudantes, sendo que: 5 estudantes afirmaram que não sentiram falta da funcionalidade e sequer perceberam que a ferramenta não tinha opção de cadastro manual, 6 estudantes não sentiram-se incomodados mesmo notando a falta da opção, 1 estudante afirmou que preferia possuir esta opção disponível e 2 estudantes afirmaram que não tinham conta no Facebook e no Google e precisaram criar uma para acessar a ferramenta.

A resolução de exercícios para 71,5% dos estudantes, conforme a Figura 29, foi facilitada com o uso da ferramenta. Para mais da metade dos estudantes (57,1%) as mensagens de erros (Figura 30) auxiliaram na resolução dos exercícios, mas 2 estudantes afirmaram que as mensagens de erros não foram o suficiente. Foi aberto no questionário uma área de sugestões e críticas e notou-se que a maioria das críticas estava voltada para como a ferramenta deveria mostrar mais adequadamente quais palavras válidas ou inválidas tiveram problemas durante o momento da validação da expressão regular. Dependendo da quantidade de símbolos configurados no cadastro de um exercício, quando um estudante chega na última etapa, muito texto é apresentado na interface e a mensagem de erro não é suficiente para auxiliar o estudante a descobrir o que está errado na solução.

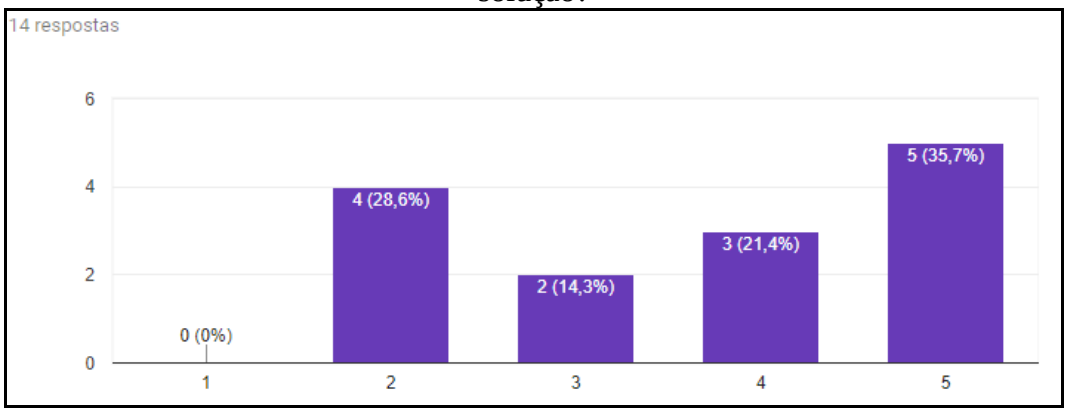
Observa-se que esses estudantes já tiveram aulas sobre expressões regulares no início do semestre. Com base nisto, 78,6% responderam que resolver as atividades se tornou mais fácil com o uso da ferramenta e 92,9% dos estudantes afirmaram que gostariam de ter usado essa ferramenta no aprendizado de expressões regulares.

Figura 29 – A ferramenta facilitou a resolução dos exercícios?



Fonte: elaborado pelo autor.

Figura 30 – As mensagens de erros na resolução do exercício auxiliaram a encontrar a solução?



Fonte: elaborado pelo autor.

3.4.4 Comparativo com outras ferramentas

Como os estudantes que efetuaram o teste prático já haviam estudado expressões regulares no semestre corrente, as últimas duas perguntas do questionário eram discursivas e permitiram os estudantes: (1) informar o nome da ferramenta utilizada para auxiliar na resolução de exercícios sobre expressões regulares; (2) descrever um comparativo entre a ferramenta citada e o Learn REGEX. As ferramentas citadas foram o Regex101 e o RegExr. RegExr é semelhante à ferramenta Regex101, descrita anteriormente. Possui um campo para escrever uma expressão regular e um campo de texto onde são destacadas visualmente todas as palavras válidas para a expressão regular especificada. Os estudantes apontaram os seguintes pontos positivos para o Learn REGEX, comparando-o com as demais ferramentas: (1) ser uma ferramenta com foco na aprendizagem, enquanto as demais servem somente como ferramenta de teste; (2) permite centralizar as informações necessárias para resolver os exercícios, compactando todas as informações em uma única tela com os enunciados na própria ferramenta, eliminando a necessidade de uma segunda ferramenta para gerenciamento dos enunciados; (3) possuir uma interface agradável, amigável e prática para a resolução de exercícios. Como ponto negativo, os estudantes informaram que tanto o Regex101 quanto o RegExr possuem um mecanismo de destaque visual mais robusto, isto é, essas ferramentas efetuam destaques visuais com cores diferentes para diferenciar qual termo da expressão regular capturou determinada palavra no campo de texto. Ainda, o usuário pode posicionar o ponteiro do mouse sobre os destaques visuais para visualizar um texto informativo sobre o termo da expressão regular que efetuou aquela captura. O Learn REGEX apenas destaca as palavras capturadas pela expressão regular especificada, sem distinguir qual termo em específico validou a palavra.

4 CONCLUSÕES

Com o desenvolvimento deste trabalho, todos os objetivos propostos foram alcançados. O resultado deste trabalho foi uma aplicação web, que serve como uma ferramenta do apoio ao ensino sobre expressões regulares e o acompanhamento em tempo real do andamento das atividades.

Para o desenvolvimento deste trabalho foi realizado um estudo sobre o React, que serviu como a principal ferramenta de montagem das telas. O React foi uma escolha adequada. Como o mesmo é somente uma biblioteca para controles visuais, sua simplicidade permitiu a criação de novas telas na ferramenta com facilidade, além de um reaproveitamento de código adequado para a padronização das telas. Porém, por ser somente uma biblioteca, outras ferramentas precisaram ser estudadas para atender todos os objetivos propostos neste trabalho. A biblioteca Redux, que serve como um gerenciador de estado global da aplicação, facilitou o desenvolvimento de telas que acompanhassem alteração de dados em tempo real, permitindo reaproveitar os mesmos modelos de dados entre toda a aplicação. Contudo, o Redux criou várias camadas de indireção, isto acabou consumindo mais tempo em como definir uma arquitetura adequada para a ferramenta. Por fim, a escolha de usar JavaScript como linguagem principal da ferramenta, sendo utilizada tanto no servidor como no cliente, foi o maior benefício encontrado no desenvolvimento deste trabalho. Foi possível reaproveitar regras de negócios em ambos os lados e a maioria das funcionalidades da ferramenta foram encontradas e integradas ao projeto por causa da vasta quantidade de bibliotecas já desenvolvidas para esta linguagem.

A maior dificuldade encontrada no trabalho foi a escolha da biblioteca `regex2dfa` para a transformação de expressões regulares em autômatos finitos determinísticos mínimos. Como a biblioteca retorna um texto do autômato finito analisado, dependendo da expressão regular utilizada, o processamento torna-se pesado demais para a máquina e causa um estouro de memória no Node.js. A expressão `(a|b)*(aa|bb)(a|b){10}`, por exemplo, causa este tipo de problema, em função da quantidade de ocorrências (`{10}`) da subexpressão `(a|b)`. A única solução encontrada foi trocar a biblioteca, mas por causa do estágio de desenvolvimento em que o problema foi encontrado, essa troca não foi efetuada. Contudo, com os testes aplicados em sala de aula, não houve nenhum momento em que está falha tenha ocorrido. O problema ocorre somente com casos bem específicos.

Por fim, com o desenvolvimento deste trabalho, acredita-se que o mesmo possa contribuir para o estudo de como desenvolver ferramentas voltadas ao ensino de expressões

regulares, visto que não foi encontrado nenhum trabalho correlato que tivesse a mesma abordagem apresentada neste trabalho. Também presume-se que este trabalho contribua para o estudo de como desenvolver uma aplicação usando somente uma linguagem de programação, permitindo um alto grau de reaproveitamento de código entre todas as partes de um projeto.

4.1 EXTENSÕES

O trabalho atingiu todos os objetivos propostos. Contudo, durante o desenvolvimento desta ferramenta e com análise dos resultados aplicados em sala de aula, algumas melhorias e funcionalidades foram identificadas e podem ser implementadas. São elas:

a) trocar o `regex2dfa` por outra ferramenta capaz de validar a equivalência entre duas expressões regulares sem causar estouros de memória em casos específicos;
b) desenvolver um mecanismo para controle de cadastro de estudantes para que o professor possa decidir quais usuários podem cadastrar-se na turma através do token de acesso;
c) desenvolver uma solução mais adequada para auxiliar o estudante a resolver um exercício quando ele atingiu a última etapa e não conseguiu resolver ainda o exercício;
d) desenvolver telas que permitam o estudante verificar todo o histórico de soluções enviadas por exercício;
e) destacar visualmente na expressão regular especifica por um estudante quando um problema for encontrado, apontando onde o problema foi encontrado na expressão regular;
f) melhorar a exibição das palavras válidas e inválidas na resolução de exercícios, para quando possuir muitas palavras o usuário não precisar ficar utilizando a barra de rolagem para encontrar qual palavra ele precisa validar adequadamente;
g) melhorar as telas de acompanhamento em tempo real, para destacar visualmente alterações efetuadas na tela, em vez de somente alterar os dados da tela;
h) colocar uma opção de cadastro de usuário na própria ferramenta, bem como abranger outras opções por redes sociais como o Github e o Twitter.

REFERÊNCIAS

CHENG, Jex. **Regulex**: JavaScript regular expression visualizer. [S.l.], 2014. Disponível em: <https://jex.im/regulex/>. Acesso em: 7 set. 2017.

CUNHA, José A. **Desmistificando expressões regulares.** [S.l.], 2005. Disponível em: <https://docente.ifrn.edu.br/josecunha/disciplinas/desenvolvimento-web/pdfs/expressoes-regulares>. Acesso em: 23 out. 2017.

DIB, Firas. **Online regex tester and debugger**: PHP, PCRE, Python, Golang and JavaScript. [S.l.], 2014. Disponível em: <https://regex101.com>. Acesso em: 20 ago. 2017.

ESPÍNDOLA, Rafaela. **Plataformas gratuitas**: por que não optar por elas? [S.l.], 2016. Disponível em: <https://www.edools.com/plataformas-gratuitas>. Acesso em: 12 set. 2017.

GOYVAERTS, Jan; LEVITHAN, Steven. **Expressões regulares bookbook.** São Paulo: Novatec, 2011.

JAMES, Josh. **How much data is created every minute?** [S.l.], 2012. Disponível em: <https://www.domo.com/blog/how-much-data-is-created-every-minute/>. Acesso em: 23 out. 2017.

JARGAS, Aurelio M. **Expressões regulares:** uma abordagem divertida. 4. ed. São Paulo: Novatec Editora, 2012.

LMS: principais características. [S.l.], 2012. Disponível em: <http://gestordeconteudos.blogspot.com.br/2012/01/lms-principais-caracteristicas.html>. Acesso em: 26 out. 2017.

MENEZES, Paulo F. B. **Linguagens formais e autômatos**. 3. ed. Porto Alegre: Sagra Luzzato, 2000.

PIVA JUNIOR, Dilermando. **As implicações dos avanços tecnológicos em educação**. [S.l.], 2008. Disponível em: <http://www.edigital.com.br/educacao/42-superior/65-implicacoes-dos-avancos-tecnologicos-em-educacao>. Acesso em: 20 ago. 2017.

REITZ, Doris S.; LIMA, José V.; AXT, Margarete. Usabilidade e desempenho de alunos em E-Learning. **Informática na Educação: teoria & prática**, Porto Alegre, v. 14, n. 1, p.137-151, jan./jun. 2011.

RINALDES, Marcília. **O uso da tecnologia como ferramenta no processo ensino-aprendizagem**. [S.l.], 2013. Disponível em: <https://www.portaleducacao.com.br/conteudo/artigos/direito/o-uso-da-tecnologia-como-ferramenta-no-processo-ensino-aprendizagem/30114>. Acesso em: 20 ago. 2017.

RODGER, Susan H.; FINLEY, Thomas. **JFLAP.** 2008. Disponível em: <http://www.jflap.org/>. Acesso em: 20 ago. 2017.

VICTOR, Bret. **Learnable programming**: designing a programming system for understanding programs. [S.l.], 2017. Disponível em: <http://worrydream.com/LearnableProgramming/>. Acesso em: 20 ago. 2017.

WANGENHEIM, Christiane G. et al. **Computação na Escola**. [Florianópolis], 2013. Disponível em: <http://www.computacaonaescola.ufsc.br/>. Acesso em: 20 ago. 2017.

APÊNDICE A – Questionário

Nos quadros a seguir são apresentadas as questões do questionário de avaliação, aplicado em sala de aula com os estudantes de Sistemas de Informação após testarem a ferramenta com um total de 18 exercícios cadastrados. As questões sobre a usabilidade da ferramenta (Quadro 7) serviram para levantar se os problemas encontrados para resolver os exercícios foram em decorrência do uso da ferramenta ou da atividade em si. Conforme as respostas recebidas, 42,9% dos estudantes responderam que tiveram dificuldade para resolver as questões (pergunta nº6). Mas, as respostas dadas às perguntas nº5 e nº7 deixa claro que os estudantes não conseguiram concluir os exercícios ou por falta de tempo ou faltava conhecimento de expressões regulares para alguns exercícios. Conclui-se que os problemas não estão relacionados com o uso da ferramenta. Ainda assim, observa-se que 4 estudantes deram sugestões de que como a ferramenta pode melhorar o auxílio na resolução dos exercícios.

Quadro 7 – Questões sobre o uso da ferramenta

PERGUNTA	TIPO DE RESPOSTA
1. Você conseguiu acessar a ferramenta através do link disponibilizado?	sim ou não
2. Caso tenha respondido NÃO para a pergunta anterior, por favor, descreva o porquê.	descritiva
3. Você teve algum problema para efetuar o login na ferramenta?	sim ou não
4. Caso tenha respondido SIM para a pergunta anterior, por favor, descreva o porquê.	descritiva
5. Você conseguiu resolver todas as 18 questões? Em caso negativo, por favor, descreva o porquê.	descritiva
6. Você teve alguma dificuldade ou dúvida para resolver as questões?	sim ou não
7. Caso tenha respondido SIM para a pergunta anterior, por favor, descreva o porquê.	descritiva

Fonte: elaborado pelo autor.

As questões sobre o ambiente de testes (Quadro 8) serviram para verificar se a ferramenta funcionava adequadamente em cada sistema operacional e navegador. Pelos dados coletados, não houve problemas no uso da ferramenta nos sistemas operacionais Windows e Mint e nos navegadores Google Chrome e Mozilla Firefox, usados na avaliação.

Quadro 8 – Questões sobre o ambiente de testes

PERGUNTA	TIPO DE RESPOSTA
1. Em qual sistema operacional você usou a ferramenta?	múltipla escolha: Windows, Mac OSX, Ubuntu, Debian, Fedora, Mint
2. Qual a versão do sistema operacional que você usou?	resposta curta
3. Em qual(is) navegador(es) você usou a ferramenta?	seleção: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Internet Explorer
4. Qual a versão do navegador que você usou?	resposta curta
5. Você sentiu alguma dificuldade no uso da ferramenta que aparenta ser um problema relacionado ao sistema operacional ou navegador? Em caso afirmativo, por favor, descreva a dificuldade enfrentada.	descritiva

Fonte: elaborado pelo autor.

As demais questões foram sobre a usabilidade da ferramenta (Quadro 9), as funcionalidades da ferramenta (Quadro 10) e a aplicabilidade dessa ferramenta em sala de aula (Quadro 11), já discutidas nas seção 3.4.

Quadro 9 – Questões sobre a usabilidade

PERGUNTA	TIPO DE RESPOSTA
1. Usar a ferramenta foi complicado?	escala linear (muito difícil – 1 até muito fácil – 5)
2. A forma como as informações são apresentadas é clara e compreensível?	escala linear (muito ruim – 1 até muito bom – 5)
3. É fácil navegar entre as telas da ferramenta?	escala linear (muito difícil – 1 até muito fácil – 5)
4. A ferramenta em algum momento parou inesperadamente?	sim ou não

Fonte: elaborado pelo autor.

Quadro 10 – Questões sobre as funcionalidades

PERGUNTA	TIPO DE RESPOSTA
1. Você sentiu falta de uma conta/senha específica para usar a ferramenta ou somente o acesso a partir da conta do facebook/google atendeu suas necessidades?	múltipla escolha: • Sim, eu não tinha uma conta facebook/google e precisei fazer uma para entrar no software • Sim, eu prefiro sempre possuir essa opção, mesmo podendo me cadastrar com o facebook/google • Não, notei a falta da opção, mas não me incomodei com isto • Não, nem percebi a falta dessa opção
2. A ferramenta facilitou a resolução dos exercícios?	escala linear (pouco – 1 até muito – 5)
3. As mensagens de erros na resolução do exercício auxiliaram a encontrar a solução?	escala linear (pouco – 1 até muito – 5)
4. Resolver atividades sobre expressões regulares se tornou mais fácil com o uso dessa ferramenta?	escala linear (mais fácil – 1 até mais difícil – 5)

Fonte: elaborado pelo autor.

Quadro 11 – Questões sobre a aplicabilidade

PERGUNTA	TIPO DE RESPOSTA
1. Você gostaria de ter usado essa ferramenta no aprendizado sobre expressões regulares?	sim ou não
2. Você usou alguma ferramenta para auxiliar na resolução dos exercícios sobre expressões regulares na matéria de Linguagens Formais e Autômatos? Em caso afirmativo, por favor, informe o nome da ferramenta.	descritiva
3. Caso tenha respondido sim para a pergunta anterior, por favor, se possível, descreva um comparativo entre a ferramenta citada na pergunta anterior e o Learn REGEX	descritiva
4. Qual a sua opinião sobre a ferramenta Learn REGEX quando ao uso e funcionalidades? Sinta-se à vontade para fazer críticas e sugestões.	descritiva

Fonte: elaborado pelo autor.

APÊNDICE B – FURB Learn Regex

A ferramenta está disponível para acesso no link https://furb-learn-regex.herokuapp.com. O código fonte da aplicação está hospedado no GitHub, no link https://github.com/gtkatakura/furb-learn-regex.