UNIVERSIDADE DO VALE DO TAQUARI - UNIVATES
CURSO DE SISTEMAS DE INFORMAÇÃO

FERRAMENTA DE APOIO AO ENSINO E APRENDIZAGEM DE ALGORITMOS E PROGRAMAÇÃO

Cléverton Heming

Lajeado, junho de 2018

Cléverton Heming

FERRAMENTA DE APOIO AO ENSINO E APRENDIZAGEM DE ALGORITMOS E PROGRAMAÇÃO

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Tecnológicas, daUniversidade do Vale do Taquari- UNIVATES, como parte dos requisitos para a obtenção do título de bacharel em Sistemas de informação.

Orientador: Prof. Me. Evandro Franzen

Lajeado, junho de 2018

Agradeço a minha família, em especial aos meus pais Antoninho e Maria e minha esposa Roseane, pelo apoio, compreensão e incentivo, principalmente nos momentos mais difíceis desta minha caminhada.

Aos professores da Univates, em especial ao meu orientador Evandro Franzen, pelo incentivo, dedicação e paciência durante a realização deste trabalho.

A todos muito obrigado!

“A imaginação é mais importante que a ciência, porque a ciência é limitada, ao passo que a imaginação abrange o mundo inteiro”.

(Albert Einstein)

RESUMO

Diante das dificuldades apresentados pelos estudantes nas disciplinas de programação, é necessário estudar como melhorar o processo de aprendizagem e como reduzir os índices de reprovação. O método de aprendizagem também é relevante neste processo, uma vez que na maioria dos casos são utilizadas listas de exercício com problemas propostos e aguardando um retorno do aluno. As dificuldades, verificadas são muitas, porém diversos alunos consideram como uma barreira, a necessidade de ter conhecimento em conteúdos que envolvam interpretação de texto, raciocínio lógico e matemático. Este trabalho baseia-se metodologia de problematização no ensino de algoritmos, com o objetivo direcionar o aluno observar de forma sistemática, denominar os pontos-chaves realizando uma investigação, buscar a fundamentação teórica e após isso colocar em prática a solução. O principal objetivo foi o desenvolvimento de uma ferramenta que utiliza a problematização para auxiliar o ensino de algoritmos e programação. O software permite ainda a coleta de dados que serão usados para apoiar a análise dos resultados, além da correção das atividades por parte dos professores. A ferramenta foi testada em duas turmas na disciplina de algoritmos e programação no segundo semestre de 2017 e no primeiro semestre de 2018. Foi realizada uma pesquisa com os alunos participantes, a qual apontou uma boa aceitação do sistema e um alto grau de satisfação.

Palavras-chaves: Programação. Algoritmo. Problematização. Arco de Maguerez.

ABSTRACT

Faced with the difficulties presented by the students in the programming disciplines, it is necessary to study how to improve the learning process and how to reduce the failure rates. The learning method is also relevant in this process, since in most cases exercise lists with proposed problems are used and awaiting a student's return. The difficulties, verified are many, but several students consider as a barrier, the need to be knowledgeable in content that involves interpretation of text, logical reasoning and mathematical. This work is based on a methodology of problematization in the teaching of algorithms, with the objective to direct the student to observe in a systematic way, to denominate the key points conducting an investigation, to seek the theoretical foundation and after that to put into practice the solution. The main objective was the development of a tool that uses the problematization to aid the teaching of algorithms and programming. The software also allows the collection of data that will be used to support the analysis of the results, besides the correction of the activities by the teachers. The tool was tested in two classes in the discipline of algorithms and programming in the second half of 2017 and in the first half of 2018. A research was carried out with the participating students, which pointed out a good acceptance of the system and a high degree of satisfaction.

Keywords: Programming. Algorithm. Problematization. Arch of Maguerez.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ACM	Association for Computing Machinery
AVA	Ambiente virtual de aprendizagem
CCS	Cascading Style Sheets
HTML	HyperText Markup Language
ICPC	International Collegiate Programming Contest
IOI	International Olympiad in Informatics
MEC	Ministério da Educação
MIT	Massachusetts Institute of Technology
MP	Metodologia de problematização
PHP	Hypertext Preprocessor
SBC	Sociedade Brasileira de Computação
SMS	Short Message Service
TIC	Tecnologias da informação e comunicação
WS	Web Sockets

SUMÁRIO

1 INTRODUÇÃO

As disciplinas de programação são parte essencial da maioria dos cursos de graduação da área de computação e são consideradas por boa parte dos estudantes como difíceis. Existem na maioria dos cursos, disciplinas de programação ou algoritmos nos primeiros semestres, ou seja, os alunos deparam-se com a necessidade de aprender programação, logo no início do curso.

As dificuldades, verificadas são muitas, porém diversos alunos iniciantes nas disciplinas de programação consideram como uma barreira o fato de contemplar conteúdos envolvendo interpretação de texto, raciocínio lógico e matemática (LIMA JUNIOR et al. 2015). Para Nobre e Menezes (2002) as seguintes dificuldades podem ser encontradas pelo professor neste processo: Reconhecer as habilidades de seus alunos; apresentar técnicas para solução de problemas; exercitar a capacidade de abstração; promover ao trabalho em grupo entre os alunos.

Tradicionalmente a reprovação em disciplinas introdutórias de programação, tem sido significativa. Watson e Li (2014) ao revisar o trabalho desenvolvido por Bennedsen e Caspersen (2007), comprovaram que a média de reprovação é na maioria dos casos, superior a 30%, aproximadamente um terço dos alunos é reprovado em disciplinas de programação. A pesquisa foi realizada em 15 países e 51 instituições diferentes, deixando o Brasil como terceiro país com maior reprovação, com percentual de reprovação superior a 50% (RAMOS et al., 2015).

Várias pesquisas indicam que os problemas ou dificuldades tem se mantido em anos recentes. O trabalho publicado por Hoed (2017) apresenta um levantamento realizado no

período de 2010 a 2015, indicando que os cursos na área da Matemática e Computação apresentam percentuais de evasão acima das demais áreas, com taxas superiores a 20%. Comparando as taxas médias nacionais de evasão para os demais cursos com os cursos de computação, observa-se que as médias para a computação apresentam entre 4 e 5 pontos percentuais a mais em todos os anos exibidos no levantamento.

Segundo relatório da Sociedade Brasileira de Computação (SBC, 2015) foi criado diversos novos cursos na área de computação, no Brasil durante os dois últimos anos. Entre estes cursos, Engenharia de Software pode ser considerado o que apresenta a maior carga de disciplinas relacionadas a programação de computadores e, ao mesmo tempo apresenta o maior crescimento em vagas oferecidas e matriculados (SBC, 2015). Pode-se observar que esse crescimento demonstra a importância de qualificar o ensino de programação para assim evitar um número cada vez maior de evasão dos cursos.

As TIC (Tecnologias da Informação e Comunicação) podem disponibilizar recursos importantes para apoiar o desenvolvimento de atividades como também acompanhar a interação entre professores e estudantes. Existem diversas iniciativas e trabalhos que tem como objetivo criar ferramentas, softwares de apoio para o ensino de programação, entre estas destacam-se ambientes para elaboração e correção de código, recursos baseados em jogos, entre outras.

Entretanto, a simples utilização de uma ferramenta, sem um método ou abordagem pedagógica adequada pode trazer resultados abaixo da expectativa. Outro aspecto importante está relacionado a coleta e disponibilização de dados que permitam ao professor acompanhar o desempenho do aluno, a forma como este interage e resolve os problemas.

O tema do presente trabalho está relacionado com o ensino de algoritmos de programação utilizando a problematização. O trabalho está limitado a criação e testes de uma ferramenta de ensino em algoritmos de programação, para utilização como ferramenta de apoio em disciplinas de programação. As dificuldades enfrentadas pelos estudantes e possíveis reprovações e desistências podem estar relacionadas com a dificuldade de problematizar, ou seja, compreender um problema e buscar possíveis alternativas de solução para o mesmo.

Considerando esta realidade, propõe-se o desenvolvimento e aplicação de um sistema para apoio ao aprendizado de Algoritmos e Programação baseado no método da

problematização. O sistema deverá permitir também a coleta de dados relacionados aos passos utilizados durante a realização das atividades e exercícios, o que poderá contribuir para compreender melhor o esforço empreendido pelo aluno na resolução das tarefas.

1.1 Objetivos

Este trabalho tem como objetivo implementar e testar um sistema de apoio ao aprendizado de programação que suporte a metodologia de problematização.

1.1.1 Objetivos específicos

São designados como objetivos específicos:

- Efetuar um levantamento sobre o cenário atual das pesquisas relacionadas ao ensino de programação.
- Conhecer os fundamentos da problematização.
- Realizar uma pesquisa sobre ferramentas utilizadas no apoio do ensino de algoritmos.
- Identificar tecnologias adequadas para implementação do sistema proposto.
- Especificar requisitos para o sistema.
- Disponibilizar a ferramenta como apoio ao trabalho do professor, o que poderá contribuir na melhora do ensino e aprendizagem de programação.

1.2 Justificativa

Segundo Hiltz e Turoff (2005) durante os últimos anos a educação sofreu diversas modificações, de um cenário com cursos presenciais utilizando objetivismo com uma didática centrada no professor para então cursos online e híbridos utilizando tecnologias digitais para assim ter apoio construtivismo, a colaboração e a uma pedagogia centrada no estudante, além

de atender alunos em escala global.

Diante das dificuldades apresentados pelos estudantes nas disciplinas de programação, é necessário estudar como melhorar o processo de aprendizagem e como reduzir os índices de reprovação. Revisões sobre diferentes abordagens e o seu impacto nos resultados podem ser encontradas em (VIHAVAINEN, WATSON, 2014; MICAEL SOUZA et al., 2016). É apresentado pelos autores, estatísticas sobre diversos tipos de intervenções realizadas por professores, como a colaboração e o trabalho em grupo e a criação de cursos preliminares, entre outros. Técnicas baseadas em colaboração, na resolução de problemas e na contextualização com situações vivenciadas no mundo real apresentaram bons resultados (VIHAVAINEN; WATSON, 2014).

As características dos novos estudantes, que se mostram ansiosos, pouco tolerantes a frustrações, associadas ao reconhecimento de que eles dominam tecnologias digitais contribui para criar uma ideia de que estes terão facilidades nos cursos de computação e que a satisfação e o sucesso serão rápidos (TWENGE; CAMPBELL, 2011). O que se verifica, porém é que muitos destes alunos se decepcionam, o que reduz o engajamento e leva a resultados abaixo do esperado.

Outro aspecto essencial é que devido ao perfil imediatista, os estudantes nem sempre estão dispostos a refletir sobre o problema, estruturar mentalmente uma solução antes de partir para a construção do código fonte, em muitos casos o desenvolvimento da solução final é baseado em estratégias de tentativa e erro, o que ocasiona problemas no momento da compilação e muitos erros de execução. O estímulo para que exista uma colaboração e um conjunto de passos que levem a problematização, especialmente durante a análise do problema e antes da construção da solução é o foco principal deste trabalho.

1.3 Estrutura do trabalho

Este trabalho está dividido em capítulos e sua ordem é definida da seguinte forma: o primeiro capítulo consiste em uma apresentação introdutória ao contexto do ensino de algoritmos de programação. O segundo capítulo compreende a revisão da literatura que inclui nos conceitos e referenciais sobre o ensino de algoritmos e programação, como também uma revisão em alguns ambientes de virtuais de aprendizado; por seguinte o terceiro capítulo,

apresenta os trabalhos relacionados ao tema, os quais foram utilizados como base para a elaboração da proposta. O quarto capítulo contém a metodologia utilizada na elaboração deste estudo. Traz a abordagem do problema, objetivo geral, procedimentos técnicos e tecnologias utilizadas. O quinto capítulo do estudo incide na proposta do trabalho, detalhando todos os recursos presente no sistema desenvolvido como também a estrutura de dados utilizada por ele.

O sexto contempla os resultados, mostrando uma análise da pesquisa realizada com os alunos sobre a ferramenta e sua metodologia, os comentários de alunos e os exercícios propostos, como também uma análise das informações coletadas na resolução dos exercícios. O sétimo e último capítulo finaliza o presente trabalho com as conclusões obtidas.

2 REFERENCIAL TEÓRICO

Neste capítulo é apresentada uma revisão bibliográfica com abordagem dos conteúdos que fundamentam o tema escolhido para o trabalho.

2.1 Ensino de algoritmos de programação

O ensino de algoritmos e programação tradicionalmente inicia-se pela apresentação e utilização de algoritmos, segundo Ascencio e Campos (2007) para elaborar um algoritmo é necessário seguir alguns passos tais como: Compreender o problema a ser resolvido, definir dados de entrada e saída, definir o processamento, construir o algoritmo, realizar testes e simulações.

Uma linguagem de programação é formada por um conjunto de instruções e regras e uma sintaxe a qual deve ser respeitada durante a escrita de um programa. Podemos citar como exemplo algumas linguagens de programação utilizadas no ensino, tais como Pascal, Java, C, Python entre outras (FARREL, 2010). A programação consiste na elaboração de algoritmos, posteriormente codificados utilizando uma linguagem de programação, produzindo um código fonte de um sistema. O código fonte é submetido a um processo de compilação, que realiza a transformação do mesmo em uma representação binária, uma linguagem de máquina, que é compreendida pelos computadores.

Na mesma linha Puga e Risseti (2009) destacam que ao fazer um programa devemos saber produzi-lo para prever qualquer intenção do usuário ao usar o mesmo, ressalta ainda a sequência de fatos, entrada, processamento e saída para solução de problemas.

Para Knuth (1998) um algoritmo pode ser caracterizado através de uma sequência finita de passos que serão realizados para resolver um problema, para isso ele define um conjunto de regras para a criação de um algoritmo.

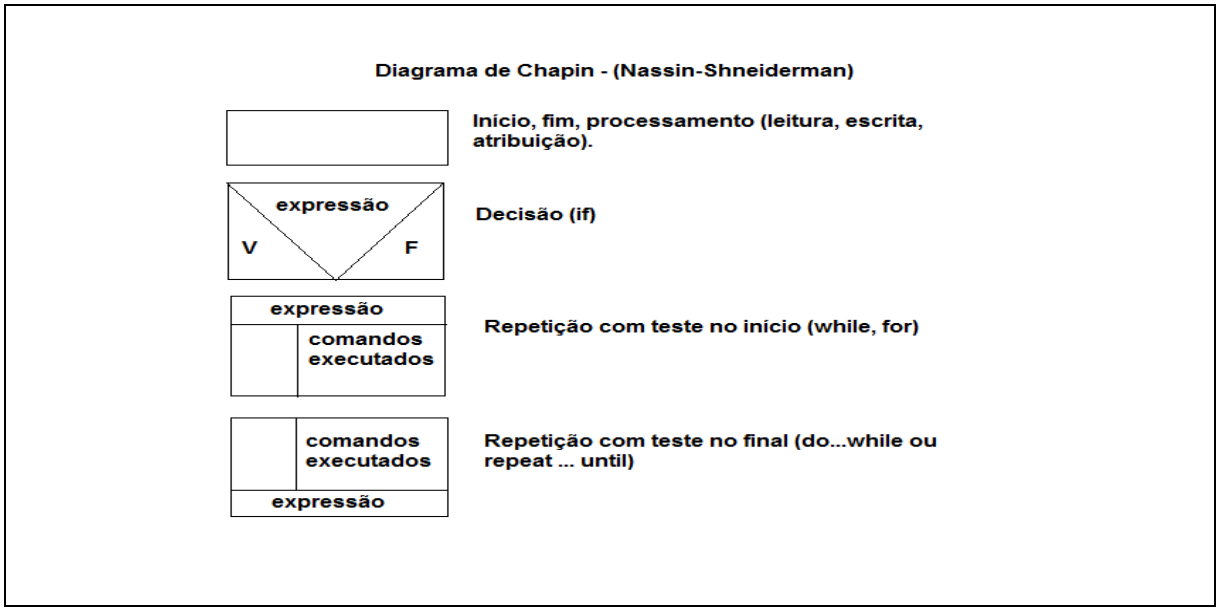
- Finitude: O algoritmo deve terminar após um conjunto finito de passos.
- Precisão: Ações devem ser definidas com precisão, não podendo haver ambiguidade na sua interpretação.
- Entradas e saídas: O algoritmo recebe um conjunto de entradas que são processadas, gerando as saídas correspondentes.

Além da escrita de instruções, o programador deve entender o problema, planejar a estrutura lógica, codificar o programa e após, traduzir para uma linguagem de máquina para executar e testar a solução (FARREL,2010). Todo o processo é diretamente influenciado para compreensão e interpretação do problema.

Após expor a maneira de construir um algoritmo são apresentados os seus tipos, segundo Puga e Risse (2009) possuem quatro: A primeira é o pseudocódigo[1] forma que utiliza uma linguagem flexível, que fica entre a linguagem natural e a linguagem de programação. A segunda é a descrição narrativa, que utiliza a linguagem natural para explicar os passos para realização da tarefa, sua utilização pode causar muitas interpretações distintas por isso não é muito utilizada. A terceira é o fluxograma como uma forma universal de representação, uma vez que apresenta figuras geométricas para ilustrar as etapas a serem seguidas para solucionar os problemas. A quarta e última é o Diagrama de Chapin (FIGURA 1) também conhecido como Nassi-Shneiderman ou diagrama N-S, apresenta a resolução por meio de um diagrama com uma visão hierárquica e estruturada, assim como a descrição narrativa não é muito utilizada.

[1] Pseudocódigo é uma forma genérica de escrever um algoritmo, utilizando uma linguagem simples, sem necessidade de conhecer a sintaxe de nenhuma linguagem de programação <https://pt.wikipedia.org/wiki/Pseudocódigo>.

Figura 1 - Simbologia do Diagrama de Chapin



Fonte: Ascencio e Campos (2007).

Para Ascencio e Campos (2007) o pseudocódigo consiste em analisar o problema e escrever, passo a passo por meio de regras para chegar à solução, e o fluxograma utiliza símbolos gráficos predefinidos para encontrar a solução.

Manzano e Oliveira (2016) definem diagrama de blocos ou de quadros, como instrumentos que estabelecem visualmente a sequência de operações a ser efetuada por um programa de computador e pseudocódigo como ferramenta textual que permite descrever de forma simples e sem o rigor técnico de uma linguagem de programação formal (uso de parênteses, pontuações e parâmetros) as etapas que o programa de computador deve executar.

Para Pimentel et al. (2006) a primeira experiência no aprendizado de algoritmos e programação termina com uma grande barreira, devido às dificuldades expostas pelos alunos durante as disciplinas que requerem como base o conhecimento de programação para assim prosseguir nas demais disciplinas.

As diretrizes nacionais para os cursos da área de computação estabelecidas pelo MEC, contam com um conjunto de competências e habilidades para a formação de profissionais nos cursos de Engenharia de Computação, Engenharia de Software, Sistemas de Informação, Ciência da Computação e Licenciatura em Computação. É possível notar que as competências relacionadas à especificação, projeto e implementação de sistemas computacionais são aplicadas em diferentes contextos. É possível perceber as seguintes habilidades associadas a programação de computadores:

- Identificar problemas que tenham solução algorítmica.
- Resolver problemas usando recursos de programação.
- Identificar e analisar requisitos e especificações para problemas específicos.
- Definir e avaliar estratégias para solucionar problemas computacionais.
- Aplicar a abstração na elaboração de programas.

Segundo Gomes e Mendes (2007) os seus estudos indicam que a dificuldade de interpretar um problema é uma característica comum entre os estudantes, sendo maior em disciplinas introdutórias. Embora o foco seja nos problemas de menor complexidade, é possível identificar que o aluno deve ser capaz de realizar todas as etapas, mesmo em situações de menor dificuldade. Estudos comprovam que erros e dificuldades acarretam em uma possível desmotivação do estudante e como consequência no aumento das dificuldades e na possível desistência ou reprovação.

Solloway e Ehrlich (1984) realizaram um estudo com objetivo de identificar os conhecimentos utilizados por programadores durante o desenvolvimento de sistemas. Foi utilizado por Nowaczyk (1984) um conjunto de testes para medir a performance de programadores e identificar suas habilidades relacionadas a capacidade de solucionar problemas. Foi identificado que a falha no entendimento e resolução de problemas em conjunto com as dificuldades de leitura e ao baixo conhecimento em matemática aponta tendência de insucesso nestas disciplinas. Muitas das dificuldades não estão ligadas com a linguagem de programação e sim nas estratégias ou à capacidade de compreender o problema e planejar sua solução.

Jenkins (2002) aponta inúmeras causas para o fracasso no aprendizado de algoritmos, entre elas a pouca capacidade de abstração, a baixa aptidão para resolver problemas, além dos métodos pouco adequados ou que considerem os diferentes estilos de aprendizagem. Outro fator que pode afetar o ensino diz respeito às sintaxes e estruturas das linguagens de programação, consideradas inadequadas para o aluno.

É possível encontrar diversos trabalhos que apresentam mapeamentos sobre as dificuldades, estratégias e recursos utilizados no ensino e aprendizagem de programação.

Embora alguns estudos apontem para estratégias promissoras que podem ser adotadas no ensino de programação, poucos deles se propõe a validar de forma quantitativa o impacto das diferentes estratégias. O trabalho de Vihavainen et al. (2014) analisou quantitativamente as intervenções e os princípios pedagógicos utilizados nestas. Quanto ao tipo da intervenção, as mesmas foram agrupadas e entre os assuntos mais relevantes estão:

- Colaboração: Estratégias que encorajam alunos a colaborar entre si.
- Contextualização: Alinhamento das atividades e conteúdos às situações reais ou contextos específicos.
- Temática de jogos: Utilização de jogos ou contextos baseados em games.
- Alterações ou atualização nos materiais utilizados.
- Suporte em pares ou grupos de trabalho.
- Cursos ou treinamentos preliminares: Criação de cursos ou treinamentos introdutórios, que, em sua maioria utilizam ambientes de apoio com Scratch ou Alice.

2.2 Ambientes ou sistemas de apoio ao ensino de algoritmos de programação

Os Ambientes Virtuais de Aprendizagem (AVAs) surgiram com o avanço da tecnologia e principalmente da internet, assim como da sociedade, que verificou neste método um novo conceito de aprender ou ensinar determinado assunto, seja ele qual for. Hoje os AVAs apoiam qualquer disciplina seja ela matemática, letras ou línguas estrangeiras. Alguns autores definem com outra nomenclatura os ambientes virtuais de aprendizagem, sendo eles ambientes digital de aprendizagem, mas ambos com a mesma finalidade.

Para Almeida (2003), ambientes virtuais de aprendizagem são sistemas disponíveis online, com objetivo de fornecer suporte durante as atividades, permitindo entregar diversos recursos, mídias e linguagens, expor informações de maneira organizada, elaborar e socializar conteúdo para atingir determinados objetivos. As atividades se desenvolvem no ritmo de trabalho e espaço de cada aluno, e de acordo com um planejamento prévio chamado de design educacional (CAMPOS; ROCHA, 1998; PAAS, 2002).

A aprendizagem através de um AVA promove uma interatividade entre alunos, professores, além de proporcionar uma organização maior em arquivos disponibilizados através da ferramenta, possibilitando a aquisição de conhecimento de uma maneira eficaz, devido que os alunos tendem a ser também protagonistas do processo de ensino/aprendizagem. O aprendizado mediado por AVA contribuir para que diversas fontes de informação e conhecimentos possam ser criados e disseminados através de conteúdos disponibilizados através de hipertextual, multimídia e recursos de simulações. Além do acesso às leituras específicas o aprendiz que interage com o conteúdo e poderá também se comunicar com outros alunos de forma síncrona e assíncrona (SANTOS, 2003, p. 4).

A seguir serão apresentados alguns ambientes ou iniciativas para apoiar o ensino da programação de computadores. Busca-se estabelecer uma relação entre estas ferramentas e o software desenvolvido neste trabalho.

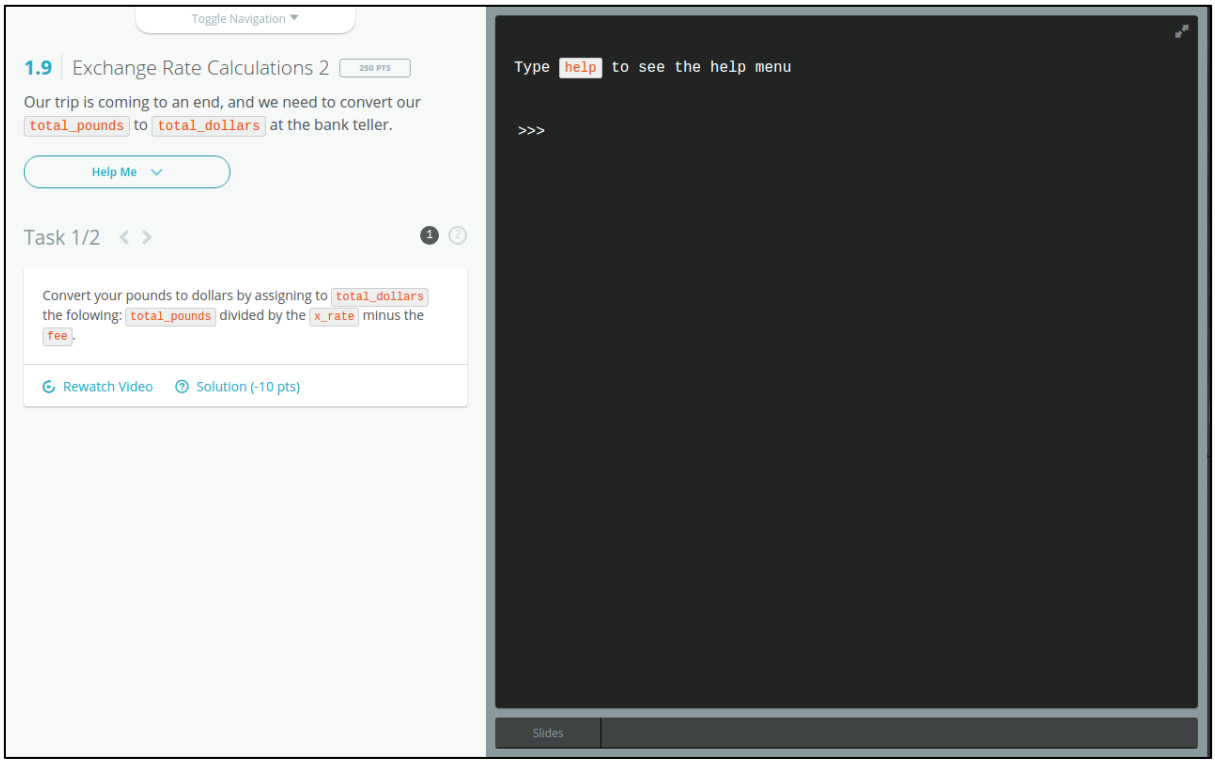
2.2.1 Code School

Para Lauber (2014), está ocorrendo um grande interesse na ciência da computação, da mesma forma na criação de ferramentas de aprendizado online. É possível citar a Code School, a qual foi fundada em 2011 pela empresa Envy Labs, que consiste em uma plataforma de aprendizado online para desenvolvedores ou iniciantes, que ensina através de conteúdos divertidos. Cada curso é construído em torno de um tema criativo e um enredo para que assim pareça que o aluno esteja jogando um jogo e não sentado em uma sala de aula. É combinada a mecânica de jogos, utilizando vídeos e desafios de codificação para tornar assim a aprendizagem mais divertida. Hoje a plataforma disponibiliza mais de 60 cursos que cobre HTML, CSS, JavaScript, Ruby, Python, .NET, iOS, Git, bancos de dados entre outros. A Code School conta com instrutores experientes e conteúdo de alta qualidade, o qual é produzido cuidadosamente e inspirado pela comunidade e rede de membros, que conta com mais de dois milhões usuários.

A plataforma oferece a possibilidade do aluno testar o código diretamente no seu navegador (FIGURA 2) sem a necessidade de ter um ambiente de desenvolvimento local, enquanto isso recebe comentários sobre o código enviado, outra característica é a utilização da experiência de gamificação no aprendizado, encorajando o aluno a continuar. Ao completar com êxito os módulos e cursos, o aluno acumulará pontos e recebera medalhas para

que assim possa mostrar suas habilidades para amigos e comunidade.A plataforma da Code School apresenta muito bem o aprendizado do código fonte, disponibilizado pequenos exercícios, focados especificamente em um assunto.

Figura 2 - Interface do Code School



Fonte: Code School (2017).

2.2.2 Khan Academy

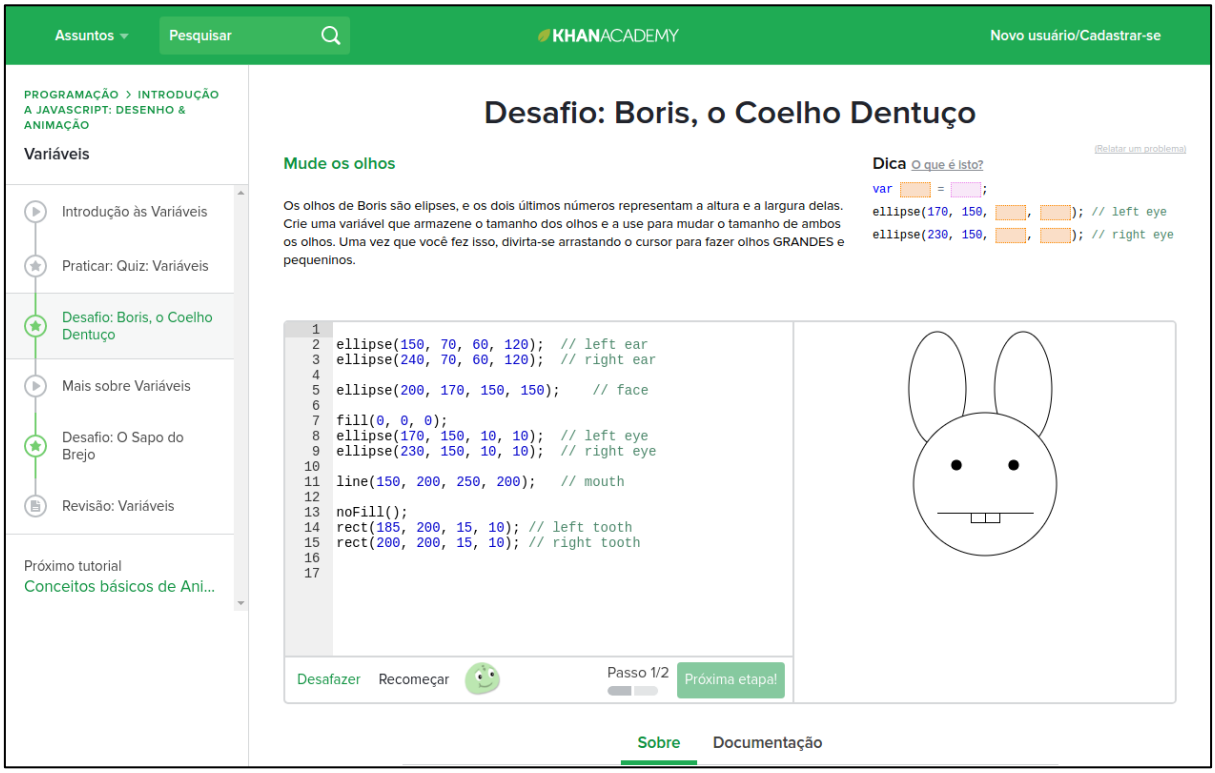
Segundo Storm (2011) a Khan Academy começou em 2006 como um site gratuito, disponibilizando vídeos curtos de 10 a 20 minutos que ensinam uma variedade de assuntos, especialmente em matemática e ciência. Atual a plataforma se constitui como ONG educacional sem fins lucrativos, com o objetivo levar o ensino de alta qualidade para qualquer pessoa, em qualquer lugar e de forma totalmente gratuita. Para isso disponibiliza mais de 3800 vídeos catalogados em diversos assuntos como biologia, ciência da computação, economias e finanças, como também física, matemática, saúde e química.

A Khan Academy disponibiliza exercícios, vídeos e um sistema de aprendizado personalizado (FIGURA 3) que habilita os estudantes a aprender no seu próprio ritmo. O projeto tem como missão especial levar a matemática para os estudantes desde o jardim até o

cálculo, utilizando tecnologias e metodologias adaptativas que identificam os pontos fortes e fracos no aprendizado. Outro ponto forte é as parcerias oferecidas, podemos citar a Academia de Ciências da Califórnia, NASA, MIT e o Museu de Arte Moderna os quais oferecem conteúdos especializados em diversos assuntos.

Na categoria da ciência da computação, temos o ensino de algoritmos e programação, o qual é separado em diversos níveis, cada um abordando um determinado assunto, como: busca binária, ordenação, algoritmos recursivos e grafos.Vale ressaltar que a Khan Academy investiu muito tempo em deixar a interface de sua plataforma o mais simples possível, adicionando os principais conceitos de gamificação, como uso de medalhas, visualização de progresso e pontos.

Figura 3 - Interface do Khan Academy



Fonte: Khan Academy (2017).

2.2.3 Codecademy

Codecademy é uma empresa de educação, criada por Zach Sims em 2011, com objetivo de criar uma plataforma interativa de ensino. Atualmente disponibiliza diversas linguagens de programação como: Javascript, Python, Ruby, PHP, JAVA como também

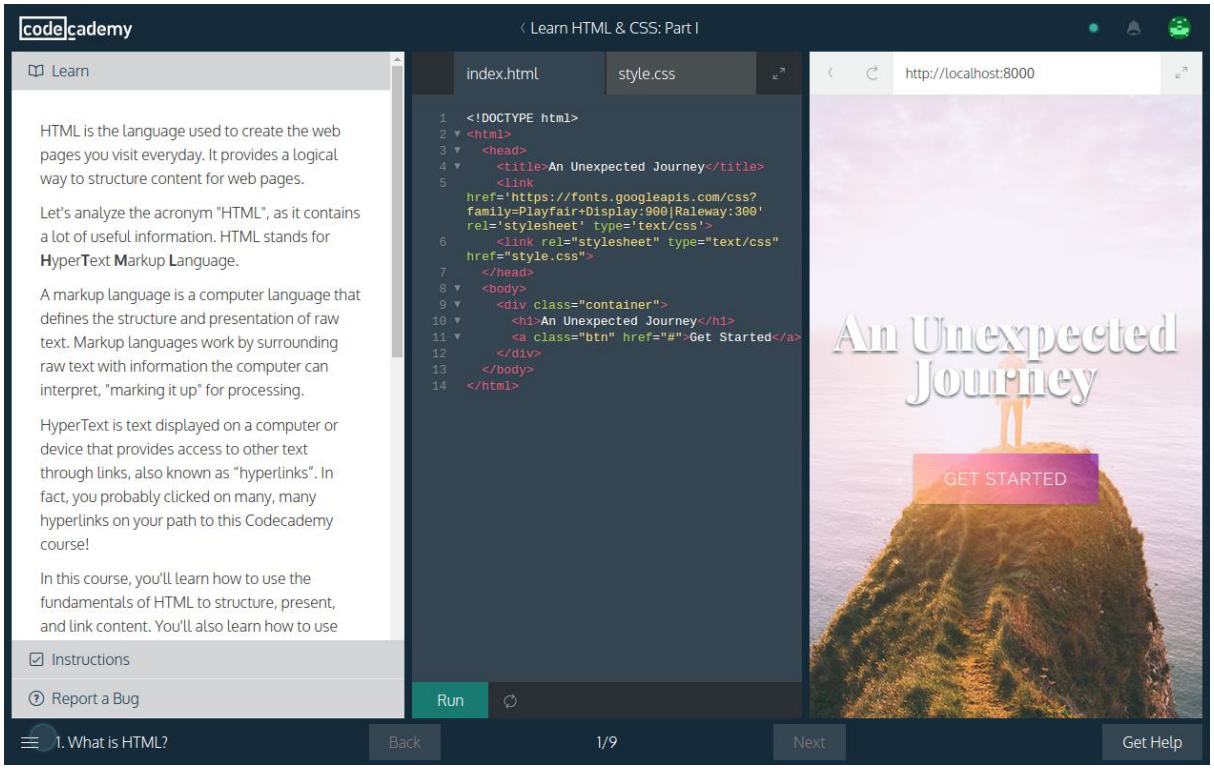
linguagens de marcação HTML e CSS, possibilitando o desenvolvimento de páginas simples até sites mais elaborados utilizando as diversas tecnologias oferecidas.

O aluno deve resolver os desafios impostos a ele (FIGURA 4), para que assim ele consiga avançar de nível e receber o próximo conteúdo. Todas as atividades são realizadas dentro do próprio site, dispensando a instalação de qualquer programa local pelo aluno.

Segundo Dantas e Consta (2013) a Codecademy está disponibilizando seus recursos na plataforma CODE, a qual é uma grande colaboração de várias empresas da área da tecnologia, como Google, Facebook, Twitter, Dropbox, Microsoft e as Universidades de Stanford, Harvard e Indiana, como também de algumas plataformas de ensino: KhanAcademy, CodeHS, Codecademy e Scratch.

A plataforma da Codecademy utiliza um conceito de interface muito diferente da Khan Academy e Code School, fornecendo na mesma tela o material didático, um editor de código fonte para diversos arquivos e um local onde o sistema criado durante os exercícios já pode ser testado.

Figura 4 - Interface do Codecademy



Fonte: Codecademy (2017).

2.2.4 Code Chef

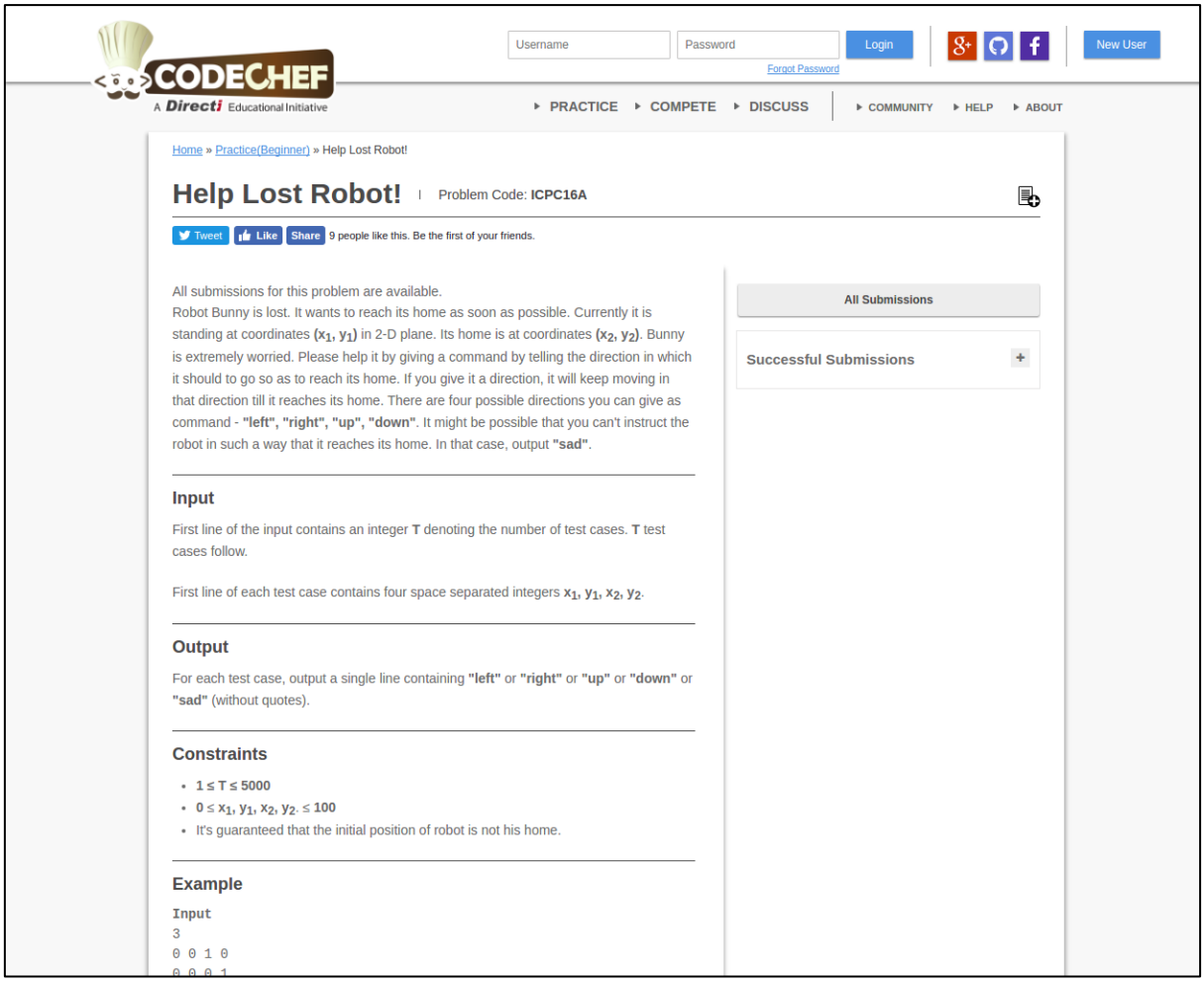
Criado em 2009 pela empresa indiana Directi, Code Chef é uma iniciativa sem fins lucrativos, de uma plataforma de competição de desafios em programação, atualmente com suporte a mais de 50 linguagens, contando com uma grande comunidade de estudantes e profissionais.

A plataforma disponibiliza o desafio ao aluno de forma contextual, como também os dados de entrada, saída e constante como pode ser visto na Figura 5. O programa deve ser escrito pelo aluno e enviado para o site para que assim possa ser analisado.

Em 2010 foi lançado a iniciativa ‘Go for Gold’ preparando as equipes indianas para as finais do Concurso Internacional de Programação Colegial (ACM-ICPC). Outra iniciativa da empresa Directi foi lançada em 2013, onde inaugurou o programa ‘Code Chef for Schools’ promovendo o talento de programação em níveis iniciais para assim criar uma comunidade independente de desenvolvedores de tecnologias. Tendo como objetivo levar os estudantes indianos para as Olimpíadas Internacionais de Informática (IOI), a qual exige que mostrem habilidades básicas de TI, como também análise de problemas, estrutura de dados, algoritmos, programação e testes.

A plataforma Code Chef, imcorpora uma ideia muito diferente dos demais sistemas apresentados acima, ela é específica para competição de algoritmo, não sendo utilizada para o ensino inicial com alunos e sim na criação de desafios para alunos com algum conhecimento em programação.

Figura 5 - Interface do Code Chef

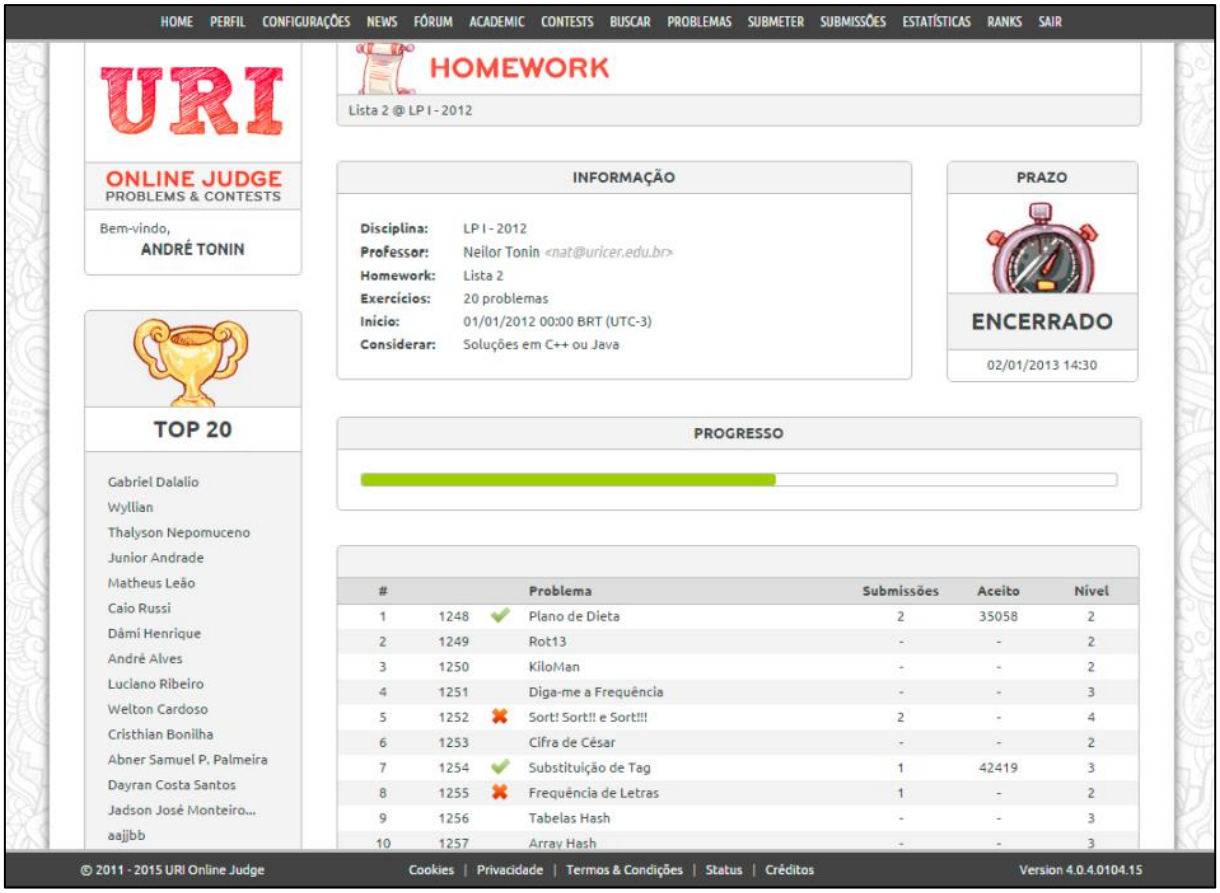


Fonte: Code Chef (2017).

2.2.5 URI Online Jungle

Segundo Selivon et al. (2013) o portal URI Online Jungle é um projeto desenvolvido desde 2011 pela Universidade Integrada – Campos de Erechim, com objetivo de oferecer desafios no padrão da ICPC (International Collegiate Programming Contest). A plataforma disponibiliza um juiz online para que seja realizado testes nas soluções propostas, como também correção em tempo real, fórum e aceitação de diversas linguagens de programação como C, C++, Java e Python (FIGURA 6).

Figura 6 - Interface Judge disponibilizada aos estudantes



Fonte: Selivon (2013).

O sistema URI Online Jungle compartilha diversas funções com o sistema Proalg o qual foi desenvolvido e testado para este trabalho, oferecendo aos professores a possibilidade de acompanhar o rendimento e a linha de raciocínio utilizada pelo aluno, como também fornecendo uma flexibilidade nos horários de estudo para os alunos. Outra característica compartilhada entre os dois sistemas é a correção dos exercícios, fornecendo um retorno para o aluno de forma fácil e direta.

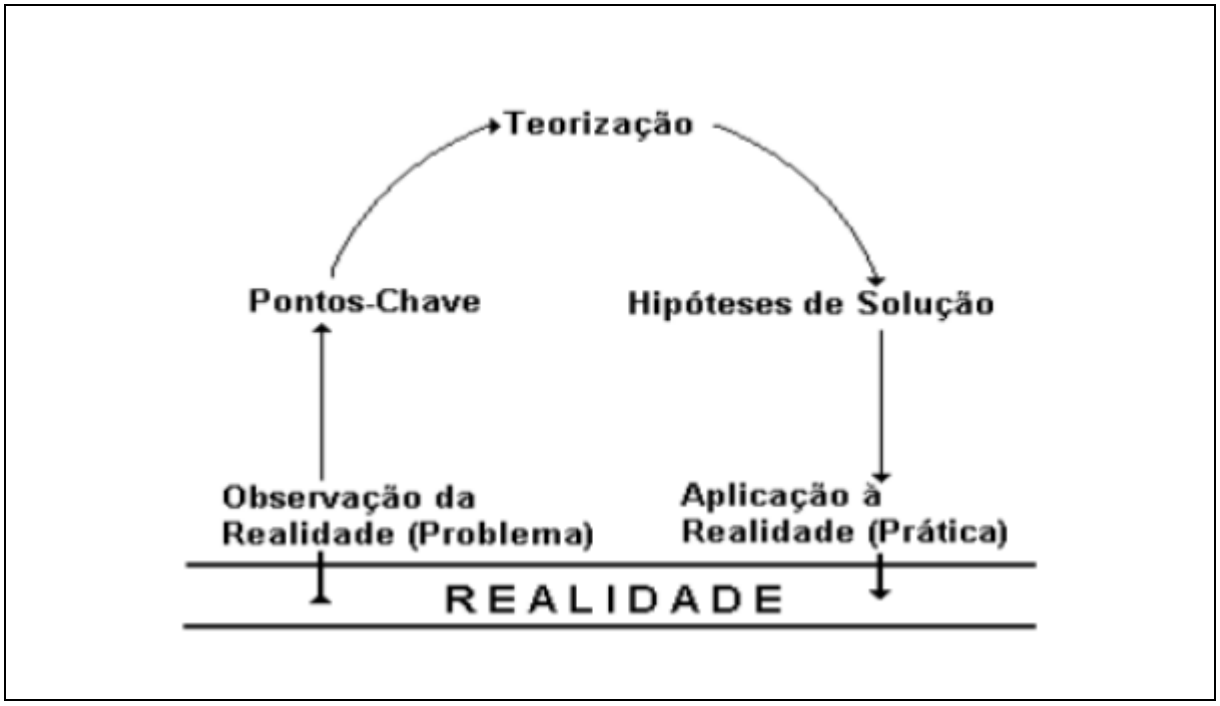
2.3 Problematização no ensino

Nesta seção serão apresentados os fundamentos da metodologia da problematização com o Arco de Maguerez. As etapas do método serão implementadas na ferramenta desenvolvida para a resolução das atividades de programação.

2.3.1 Arco de Maguerez

Segundo Giannasi e Berbel (1998) o Arco de Maguerez de Charles Maguerez, tem como primeira referência teórica de MP, conhecendo o esquema apresentado por Bordenave e Pereira (1982), podemos ver o esquema mostrado na Figura 7.

Figura 7 - Arco de Maguerez



Fonte: Bordenave e Pereira (1982 apud COLOMBO; BERBEL, 2007).

Na primeira parte denominada, Observação da Realidade (Problema), o professor deve orientar os alunos a observar de forma sistemática o problema em questão, para que assim tenham uma visão ampla do problema (SILVA; DELIZOICOV, 2008). Com essa análise os alunos conseguem identificar as dificuldades, carências, diferenças de vários níveis, que possam ser transformadas em problemas (GIANNASI; BERBEL, 1998).

Em uma segunda etapa, denominada Pontos-Chave, os alunos devem realizar uma investigação mais aprofundada, focando primeiramente nas possíveis causas do problema e por meio de análise reflexiva, identificar os pontos chaves para o entendimento do problema, da mesma forma encontrar forma para resolver ou amenizar o problema estuda (GIANNASI; BERBEL, 1998; MITRE et al., 2008; SILVA; DELIZOICOV, 2008).

A parte da Teorização tem como objetivo buscar a fundamentação teórica para que assim possa ser explicado o problema. Nesse momento o professor deve orientar os alunos a

buscar elementos científicos que possam ajudar o entendimento do assunto, dentro de cada um dos pontos chaves já identificados. O aluno deve buscar as informações em diversas fontes, como: livros em bibliotecas, artigos, palestras. Durante esse processo de pesquisa o professor deve ficar mediando, orientando e estimulando a participação do aluno (GIANNASI; BERBEL, 1998; MITRE et al., 2008; SILVA; DELIZOICOV, 2008).

A quinta e última parte, aplicação à realidade, o aluno deve colocar em prática as idéias criadas como solução do problema, e aprenderá a generalizar o que foi aprendido para aplicar em outras situações, tendo a possibilidade de avançar na fundamentação teórica, criando estratégias que o ajudem pôr em prática, de alguma maneira, as soluções do problema (GIANNASI; BERBEL, 1998; MITRE et al., 2008; SILVA; DELIZOICOV, 2008).

Após a conclusão do Arco de Maguerez, o aluno pode treinar a relação prática – teoria - prática, tendo sempre como etapa inicial e final do processo de ensino e aprendizagem, a realidade social (GIANNASI; BERBEL, 1998; MITRE et al., 2008; SILVA; DELIZOICOV, 2008).

3 TRABALHOS RELACIONADOS

É possível encontrar diversos estudos que buscam aprimorar o ensino de algoritmos e programação ou compreender as dificuldades apresentadas pelos estudantes. Um dos estudos mais relevantes foi conduzido por Benedsen e Caspersen (2007). Os autores demonstram que a média de reprovação nas disciplinas introdutórias de programação é superior a 30%, ou seja, aproximadamente um terço dos alunos é reprovado.

Para chegar a esta conclusão foi desenvolvida uma pesquisa que envolveu universidades em diferentes países, com uma participação maior, porém, nos Estados Unidos. Aproximadamente 66% das respostas foi de instituições americanas. O levantamento buscou verificar também qual o paradigma de programação mais adotado, onde percebeu-se que em metade dos casos a opção principal é pelo uso da orientação a objetos. Uma discussão importante apresentada no artigo é se o número de alunos que falha, desiste ou reprova é considerado alto demais. Na visão dos autores a taxa é significativa, mas encontra-se dentro do que era esperado.

Pesquisas no Brasil são mais raras, é possível encontrar um número reduzido de levantamentos estatísticos sobre a taxa de reprovação ou desistência nestas disciplinas, em nível nacional. Encontram-se diversos levantamentos regionais, locais ou mesmo com o foco em uma determinada instituição, mas a realidade brasileira ainda é pouco conhecida. Um destes levantamentos foi realizado por, Ramos (2015), que apresenta resultados ainda piores, se comparados com cenários internacionais, estimando que as taxas de reprovações que podem variar de 40 a 50% em instituições brasileiras.

Considerando este cenário, diversas iniciativas exploram técnicas ou estratégias

diversas para obter melhorias no processo de ensino e aprendizagem. Serão apresentadas a seguir iniciativas que possuem relação com este trabalho.

A utilização de abordagens baseadas em problemas tem crescido, em alguns casos a aprendizagem baseada em problemas é aplicada como modelo para um curso completo ou como estratégia para direcionamento da matriz curricular, como em Qiu e Chen (2010). Os autores apresentam iniciativa semelhante a outras, quando problemas são empregados para nortear os estudos de grupos de alunos em disciplinas ou em cursos. Nesta abordagem, o problema é debatido, discutido por grupos de estudantes e todas as atividades, incluindo a programação são relacionadas ao problema analisado.

De Oliveira e Ferreira (2015), apresentam os resultados da aplicação da problematização com o Arco de Maguerez no ensino de programação. A técnica foi combinada com tecnologias para troca de mensagens, especialmente no ensino semipresencial. Foram definidas quatro etapas:

- Etapa inicial: O aluno analisa individualmente o problema e identifica os pontos chaves do mesmo.
- Atividades em grupos: Os problemas são analisados em grupos.
- Definição da solução final: Os líderes de todos os grupos escolhem e apresentam a solução final a partir das diversas soluções analisadas.

As tecnologias são usadas como recursos para colaboração e a interação entre os participantes. As principais tecnologias usadas no trabalho foram o Google Docs e recursos móveis baseados em SMS. Como forma de justificar a aplicação da metodologia e o uso das tecnologias citadas, os autores apresentaram levantamentos relativos ao número de estudantes que possuem celulares e utilizam com frequência SMS e a internet a partir destes dispositivos.

Iepsen (2013) utilizou técnicas de mineração de dados para encontrar padrões associados à frustração no aprendizado de algoritmos. Foi desenvolvido um ambiente de apoio para o desenvolvimento de exercícios de programação e para a coleta de dados usados na mineração. São coletados dados relativos ao número de compilações com erros, número total de erros, tempo entre o início e a última compilação do programa, número de programas anteriores sem solução (sem uma compilação correta), número de compilações sem erros de sintaxe.

O aluno utiliza o ambiente para escrever e compilar o algoritmo e enquanto resolve a questão tem a possibilidade de indicar se está frustrado clicando em um botão disponível no software. Posteriormente o processo de mineração encontra regras que relacionam alto número de compilações ou a incidência recorrente de erros em programas com o estado de frustração indicado pelo usuário.

3.1 Gerador de exercícios para apoio no ensino de programação

A ferramenta de geração automática de exercício foi desenvolvida por Pedro Tramontina (2015) em seu trabalho de conclusão de curso, a qual possibilita a geração de exercícios para as disciplinas de algoritmos de programação, assim auxiliando os alunos na melhora do seu processo de aprendizado.

O sistema está separado em três componentes, que são: gerador, tradutor e uma lista de *templates*[2] como podem ser visto na Figura 9. O sistema está estruturado em seu módulo de *templates*, os quais são arquivos estruturados que contém informações para a geração dos exercícios, como o nível de dificuldade, seu tópico principal, enunciado genérico e sua solução no formato de um pseudocódigo (FIGURA 8).

Figura 8 – Pseudocódigo

```
## Pseudocode

    declare integer i
    declare array iArray of integer size ${n}

    for i from 1 up to ${n} step 1
        iArray[i] = random(${ini}, ${end})

    for i from 1 up to ${n} step 1
        print iArray[i]
        print " "
    println ""
```

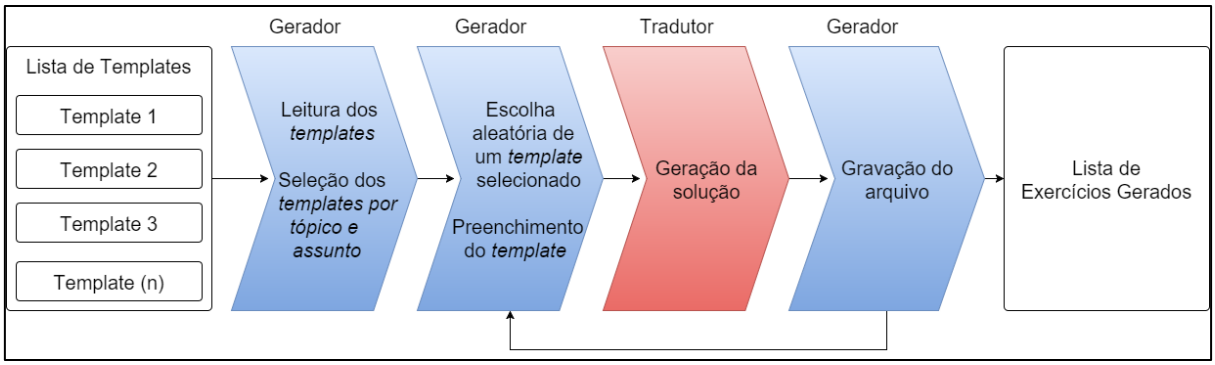
Fonte: Tramontina (2015).

Cada *template* corresponde a um tipo de exercício, e em seu conteúdo existem lacunas as quais serão preenchidas de forma automática e aleatória pela ferramenta, assim gerando

[2] Template: é um documento de conteúdo, com apenas a apresentação visual e instruções sobre onde e qual tipo de conteúdo deve entrar a cada parcela da apresentação (https://pt.wikipedia.org/wiki/Web_template).

diversos exercícios de um mesmo *template*. Para que a geração seja realizada, devem ser informados os seguintes parâmetros: quantidade de exercícios, assunto principal e o nome da linguagem de programação de saída, que pode ser C, C++, Java e Python. Os exercícios serão salvos individualmente em arquivos.

Figura 9 – Processo de geração de exercícios



Fonte: Tramontina (2015).

Com o trabalhado de Tramontina (2015) pode ser realizado um estudo sobre o nível e a qualidade do aprendizado de algoritmo e programação, já que ferramenta criada por ele nos ajuda na criação de exercícios únicos, assim dificultando o compartilhamento das respostas entre os alunos.

A criação de uma ferramenta que possa utilizar tecnologia de compartilhamento e trabalho em grupo, como citado por Oliveira e Ferreira (2015) e exercícios de forma automática como proposto por Tramontina (2015) podem facilitar o estudo sobre a compreensão das dificuldades dos alunos no aprendizado de algoritmo e programação.

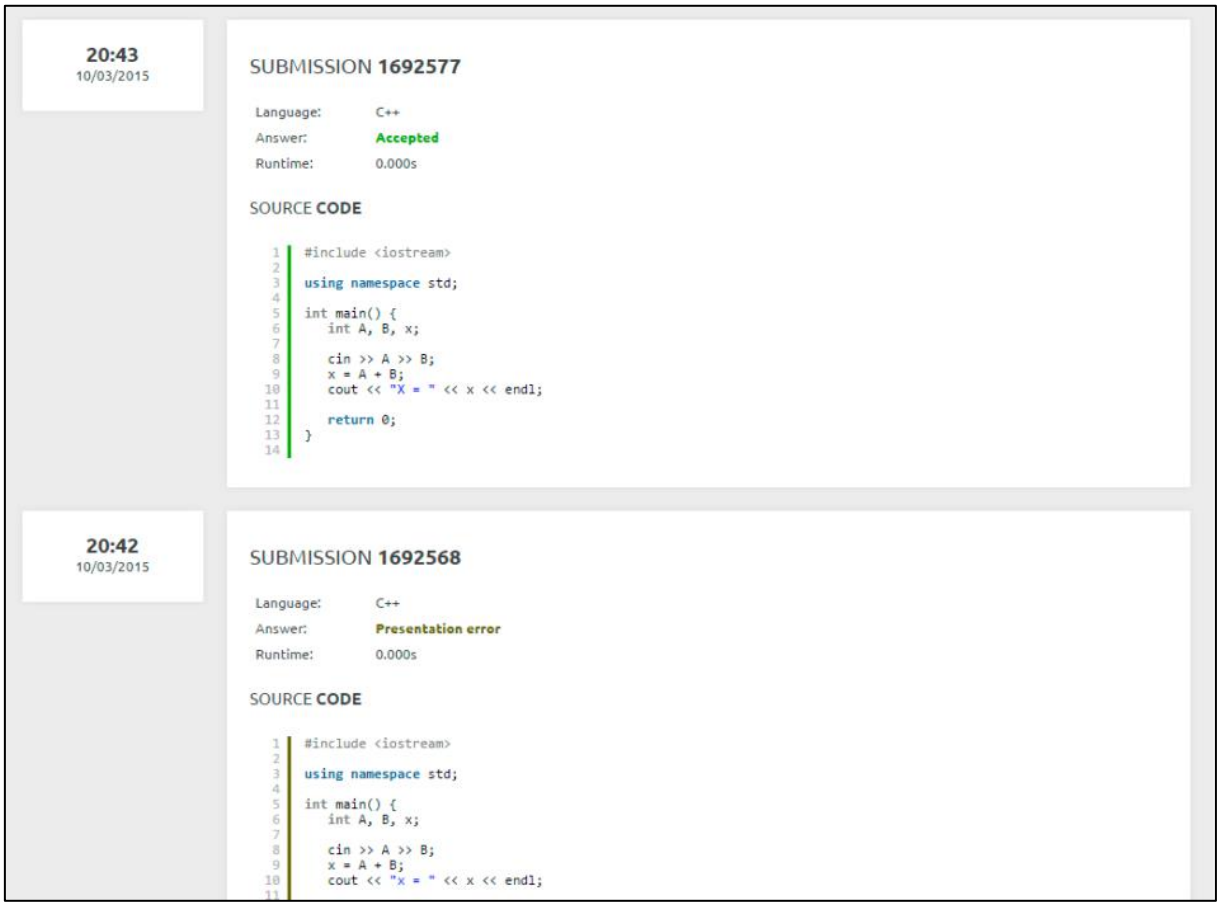
Para o desenvolvimento do sistema Proalg, foi utilizando os modelos de *Template* criados por Tramontina (2015), já o sistema de geração de exercícios será utilizado futuramente.

3.2 URI Online Jungle

Segundo Selivon et al. (2013) o portal URI Online Jungle é um projeto desenvolvido desde 2011 pela Universidade Integrada – Campos de Erechim e posteriormente criado um novo módulo chamado ‘Academic’, o qual oferece aos professores a possibilidade de acompanhar o rendimento e a evolução de seus estudantes, como também o controle das

atividades propostas em horários de aulas e extraclasse. Outra característica presente na ferramenta é a definição de períodos para a resolução de uma lista de exercícios e a possibilidade do professor visualizar os diversos envios realizados pelo aluno (FIGURA 10).

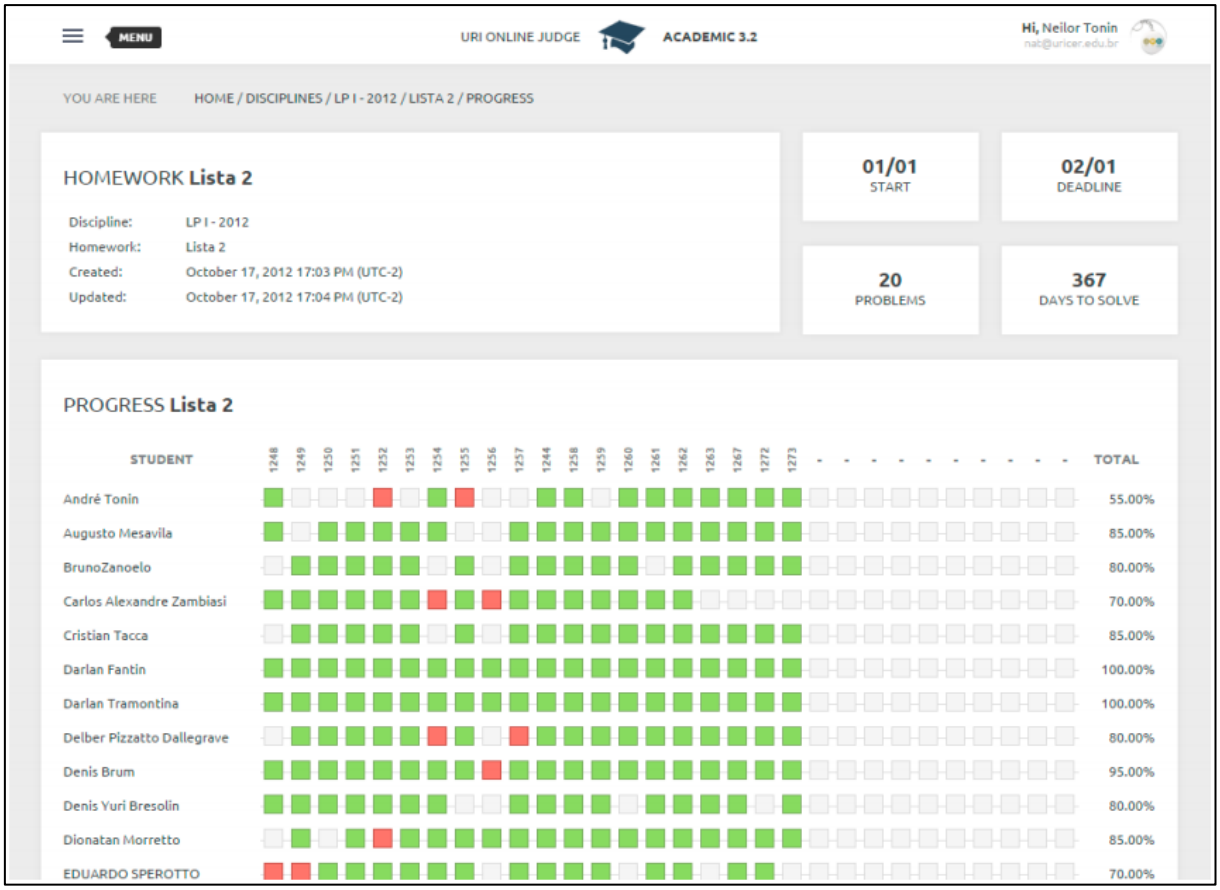
Figura 10 - Envio realizados por um aluno



Fonte: Selivon (2013).

Para o professor é entregue uma interface simples, na qual ele pode acompanhar de forma fácil e objetiva a evolução de seus alunos, a mesma disponibiliza diversas informações agrupando-as por uma lista de exercícios. É fornecido um percentual das atividades concluídas e também de forma visual através de cores nos itens da tabela, como mostrado na Figura 11.

Figura 11 - Interface Judge disponibilizada ao professor



Fonte: Selivon (2013).

4 PROCEDIMENTOS METODOLÓGICOS

Apresentam-se neste capítulo os métodos utilizados no desenvolvimento do trabalho, esta seção está subdividida em quatro grupos: Método de Pesquisa, Modo de abordagem, objetivos, procedimentos técnicos da pesquisa e tecnologias utilizadas.

4.1 Métodos de pesquisa

O método de pesquisa adotado pelo trabalho será o indutivo, pois será realizada uma observação no processo de aprendizagem de algoritmos, utilizando uma ferramenta para então fornecer exercícios e coletar informações dos alunos. Segundo Visser, Plomp e Kuipler (2002) mesmo por meio de utilização de tecnologias no aprendizado, os estudantes acabam enfrentando os mesmos problemas motivacionais do ensino tradicional, inclusive sendo uma das principais causas de reprovação e desistência. Para Zichermann e Cunningham (2011), deve ser utilizado a abordagem de Gamificação, sendo um pensamento e processo de jogo para assim envolver os alunos na resolução do problema.

4.2 Modo de abordagem da pesquisa

A metodologia de pesquisa adotada pelo trabalho enquanto ao seu modo de abordagem será o Quantitativo, de acordo com Petersen et al. (2008) mapeamento de uma forma sistemática é uma metodologia que pode envolver a busca por literatura, tendo como objetivo verificar a natura, a extensão e a quantidade de estudos publicados na área de interesse. Segundo Squire (2011) os jogos são uma experiência que merece ser analisada, com o

objetivo de aprender a cativar os estudantes que cresceram em um contexto estimulado por tecnologia.

4.3 Objetivos da pesquisa

A metodologia de pesquisa adotada pelo trabalho enquanto ao seu objetivo será a pesquisa exploratória, envolvendo exercícios, questionários e sugestões de melhorias. Para Kapp (2012) a gamificação como um mecanismo e pensamento para que assim possa engajar as pessoas, motivá las, promover conhecimento e resolver problemas. Para Petersen et al. (2008), pode ser conduzido um mapeamento em cinco passos essenciais tais como:

1. Definir questões para uma pesquisa;
2. Realizar uma pesquisa de estudos;
3. Realizar uma triagem de documentos;
4. Definir palavras chaves dos resumos;
5. Extração de dados e mapeamento.

O presente trabalho se caracteriza com o primeiro e quinto passo, de modo que o sistema utilizará um cadastro de exercícios para aplicação em aula, e durante os exercícios serão coletados informações para que em um próximo trabalho seja realizado um mapeamento das informações.

4.4 Procedimentos técnicos usados na pesquisa

A metodologia de pesquisa adotada pelo trabalho enquanto ao seu procedimento técnico será a pesquisa de campo, de acordo com Berbel (1996) a metodologia de problematização não é um mero exercício intelectual, pois as decisões tomadas deverão ser executadas ou encaminhadas. Nesse momento, o componente social e político está mais presente. A prática que corresponde a essa etapa implica num compromisso dos alunos com o seu meio. Do meio observaram os problemas e para o meio levarão uma resposta de seus estudos, visando transformá-lo em algum grau. Para Diaz Bordenave e Pereira (1980) a

educação problematizadora parte do pressuposto de que "uma pessoa só conhece bem algo quando o transforma, transformando-se ela também no processo".

4.5 Tecnologias utilizadas

O sistema para o apoio no ensino de algoritmo de programação utiliza diversas tecnologias como pode ser visto na Figura 12, em seu *backend*[8] dois sistemas como servidores web são utilizados:

- NGINX[3] utilizado para interpretar os códigos fontes escritos na linguagem de programação PHP (Hypertext Preprocessor), a qual utilizamos a versão 7.1.

- NodeJS[4] utilizado para interpretar os códigos fontes escritos na linguagem JavaScript, utilizamos o módulo Socket.IO[5] para trabalhar com webscokets.

No *frontend*[9] está presente o JavaScript na versão ES6 e CCS (*Cascading Style Sheets*) na versão 3, para isso utilizamos o *framework* React na versão 15 para a criação da interface do editor de código fonte e o Twitter Bootstrap na versão 3 para a criação das telas do sistema. Para o desenvolvimento do sistema será utilizado o gerador de códigos Code Charge Studio na versão 5.1 para geração dos códigos em PHP.

A coleta de informações do usuário será feira por WebSockets, utilizando a biblioteca Socket.io[5], com o servidor NodeJS na versão 6.1. Será utilizado o banco de dados relacional Mysql na versão 5.7 a *engine* de armazenamento innoDB. O Ace.js será utilizado como editor de códigos *online*, na ferramenta que será desenvolvida.

Para a compilação do código fonte criado pelo aluno, será utilizado o compilador JAVAC[6] na versão 8 e para o controle de recursos e tempo utilizando durante o processo de

---

[3] Nginx é um servidor web rápido, leve, e com inúmeras possibilidades de configuração para melhor performance (https://pt.wikipedia.org/wiki/Nginx).

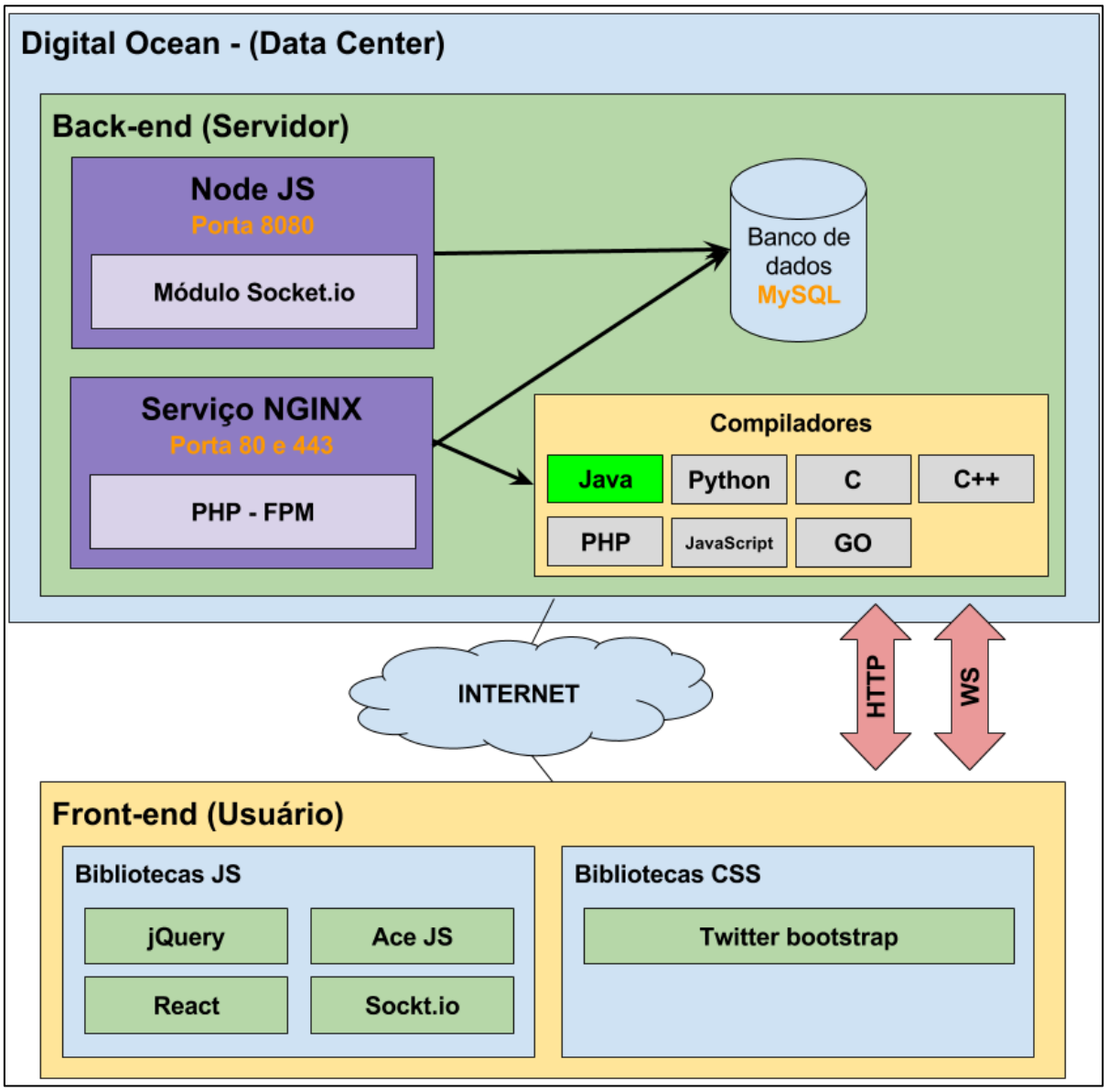
[4] NodeJS é um interpretador de código JavaScript que funciona do lado do servidor (https://pt.wikipedia.org/wiki/Node.js).

[5] Socket.IO é uma biblioteca de JavaScript para aplicações web em tempo real . Permite comunicação em tempo real e bidirecional entre clientes e servidores da Web (https://en.wikipedia.org/wiki/Socket.IO).

[6] JAVAC é o compilador primário da linguagemJava, incluído noJava Development Kit(JDK) daOracle Corporation. Apesar de existirem outros compiladores, o criado pelaSun Microsystemsé o mais usado (https://pt.wikipedia.org/wiki/Javac).

compilação, será utilizado o comando ULIMIT[7] do linux.

Figura 12 - Diagrama das tecnologias utilizadas



Fonte: Elaborada pelo autor (2018).

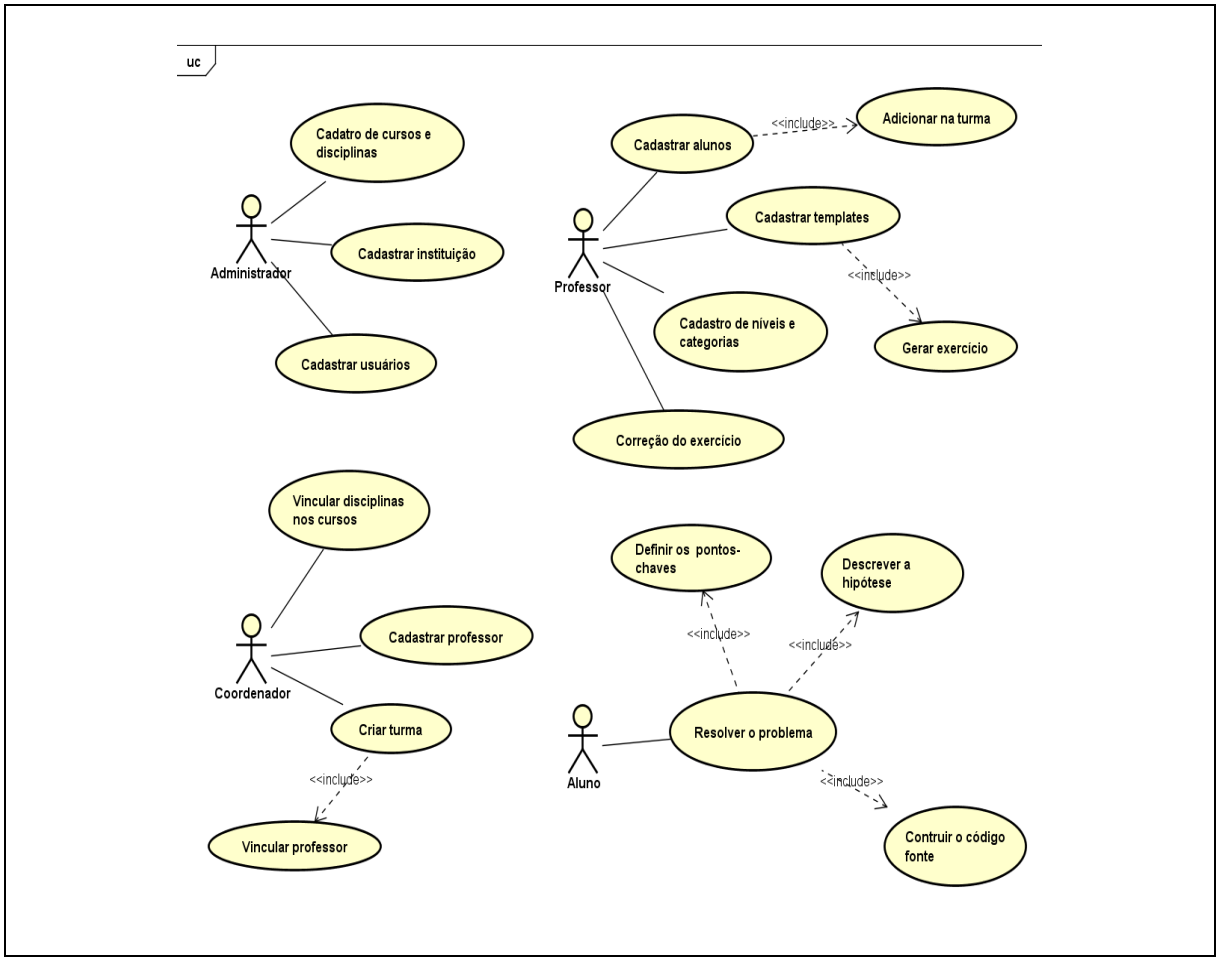
[7]ULIMIT é um comandoshellinternousado para mostrar e definir várias restrições no uso de recursos para um shell (http://wiki.linuxquestions.org/wiki/Ulimit).
[8]Backend Sistema responsável pela regra de negócios, webservices e APIs de uma aplicação (https://www.oficinadanet.com.br/post/13541-afinal-o-que-e-frontend-e-o-que-e-backend-).
[9]Frontend Interface de interação com o usuário (https://www.oficinadanet.com.br/post/13541-afinal-o-que-e-frontend-e-o-que-e-backend-).

5 SISTEMA PROALG

O presente capítulo tem objetivo descrever o sistema para o apoio no aprendizado de algoritmos de programação, serão abordados assuntos referentes à estrutura, modelagem e tecnologias utilizadas. O sistema será desenvolvido de forma complementar em dois trabalhos.

O sistema PROALG foi desenvolvido na plataforma web e disponibilizado para acesso pela internet. Consta no sistema uma definição de tipos de usuário (FIGURA 13), que são nomeados como administrador, coordenador, professor e aluno, dessa forma foi definido o tipo de perfil do usuário no sistema, liberando assim as funcionalidades pertinentes ao seu tipo.

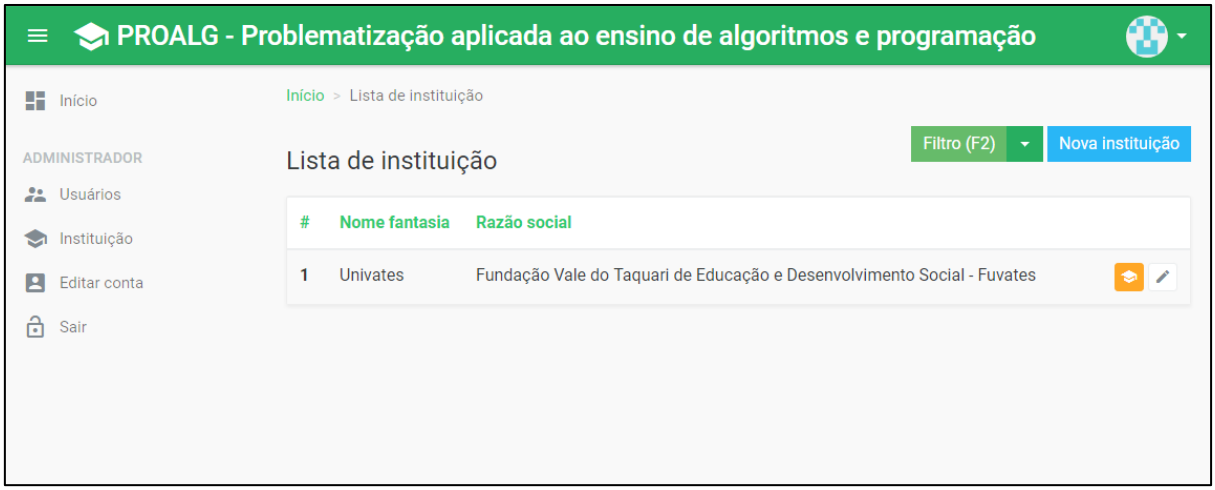
Figura 13 - Diagrama de casos de uso do PROALG



Fonte: Elaborada pelo autor (2018).

O administrador tem como objetivo o cadastro das instituições de ensino (FIGURA 14), coordenadores, cursos e disciplinas, dessa maneira mantendo uma uniformidade entre os cursos e disciplinas disponibilizados por todas as instituições.

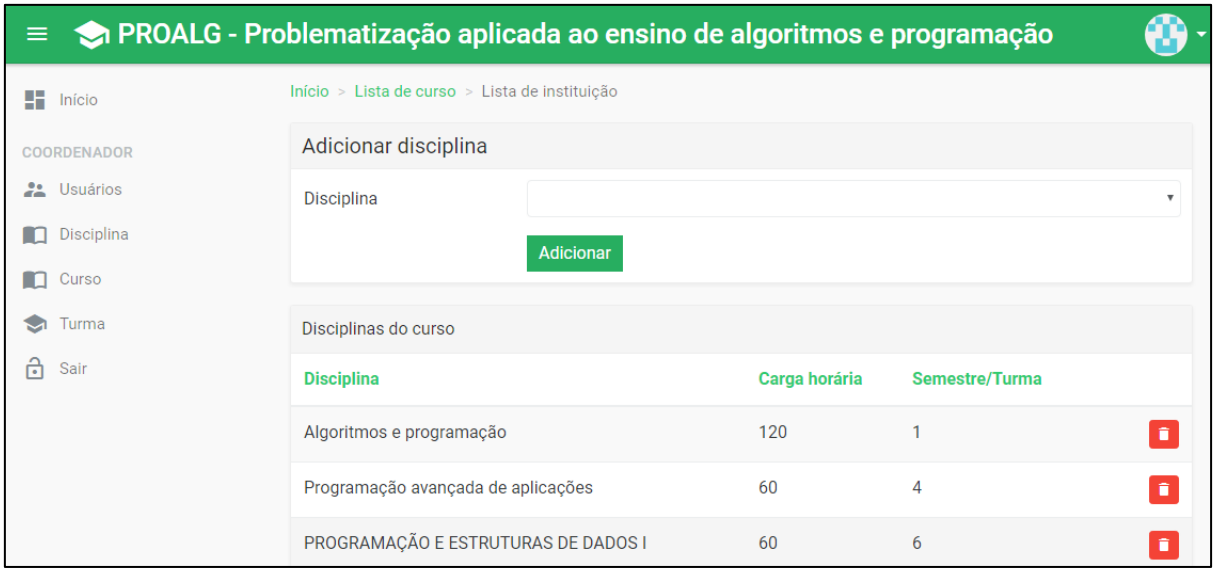
Figura 14 - Lista de instituições



Fonte: Elaborada pelo autor (2018).

O coordenador deve vincular os cursos a sua instituição e preencher alguns dados adicionais, que são as características, os diferenciais como também o mercado de trabalho, carga horária, turno e duração do curso. Após isso será vinculado às disciplinas ao curso (FIGURA 15) já selecionado e novamente serão fornecidas novas informações, que neste caso é a carga horária, semestre e ementa. Tendo a configuração de cursos e disciplinas concluídas, será realizado o cadastro de professores, que serão usuários, porém com o tipo ‘professor’, após isso o coordenador deve criar as turmas de cada disciplina e posteriormente vincular um professor em cada turma.

Figura 15 - Cadastro de disciplina



Fonte: Elaborada pelo autor (2018).

O professor deve realizar o cadastro das turmas oferecidas na instituição a qual ele pertence, para isso basta acessar o menu turma no menu lateral e pressionar o botão “Nova turma” disponível na tela de listagem (FIGURA 16).

Figura 16 - Lista de turmas

Fonte: Elaborada pelo autor (2018).

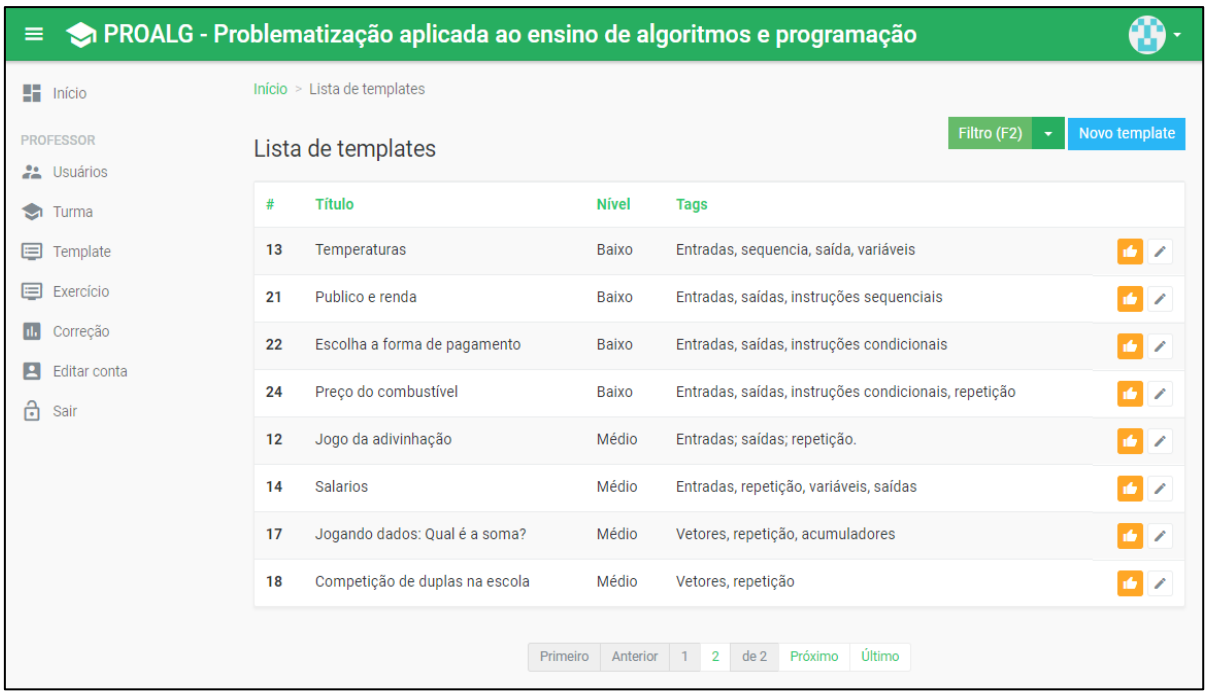
Na tela de cadastro de turmas (FIGURA 17), é selecionado a disciplina, o professor e adicionado uma breve descrição como também a data de início e fim do período.

Figura 17 - Cadastro de turmas

Fonte: Elaborada pelo autor (2018).

Será de responsabilidade do professor cadastrar os alunos no sistema, ou apenas fazer a vinculação deles nas turmas caso o usuário já esteja cadastrado. Outra tarefa do professor será o cadastro dos *templates* de exercícios (FIGURA 18) que serão utilizados durante as aulas, os quais estão vinculados ao professor, desse modo o mesmo *template* pode ser utilizado em diversas turmas. Outra característica é disponibilizar o *template* como público, para que outros professores possam usufruir do exercício. Para realizar a geração de exercício é necessário que antes sejam criados os níveis, para que assim sejam classificados os exercícios.

Figura 18 - Lista de *templates*



Fonte: Elaborada pelo autor (2018).

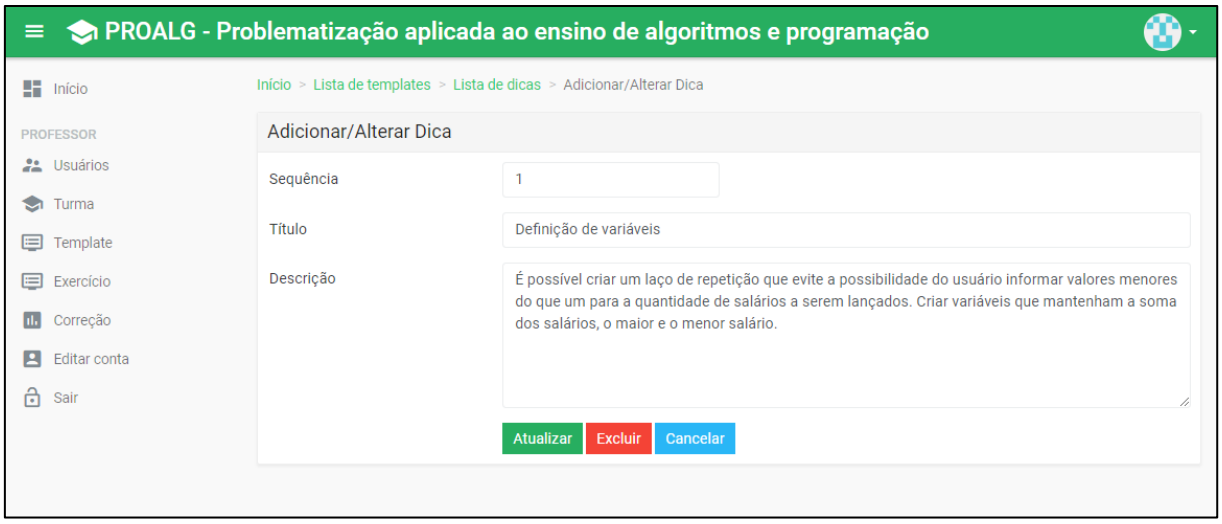
Durante o cadastro do *template* (FIGURA 19) é solicitado a descrição do problema e enunciado, o problema deve ser uma contextualização de uma situação real e não um enunciado tradicional de um exercício.A descrição do problema é fundamental para que o estudante seja estimulado a refletir sobre o contexto, a respeito da situação epartir desta defina os pontos chave e uma hipótese de solução, antes de iniciar a codificação do problema.

Figura 19 - Cadastro de *templates*

Fonte: Elaborada pelo autor (2018).

É possível adicionar dicas no *template* (FIGURA 20), que serão disponibilizados aos alunos durante a realização dos exercícios, não existe um limite, todas serão mostradas no painel lateral da tela de exercício do aluno. As dicas são compostas por um título e uma breve descrição que possa auxiliar o aluno no momento da resolução do exercício.

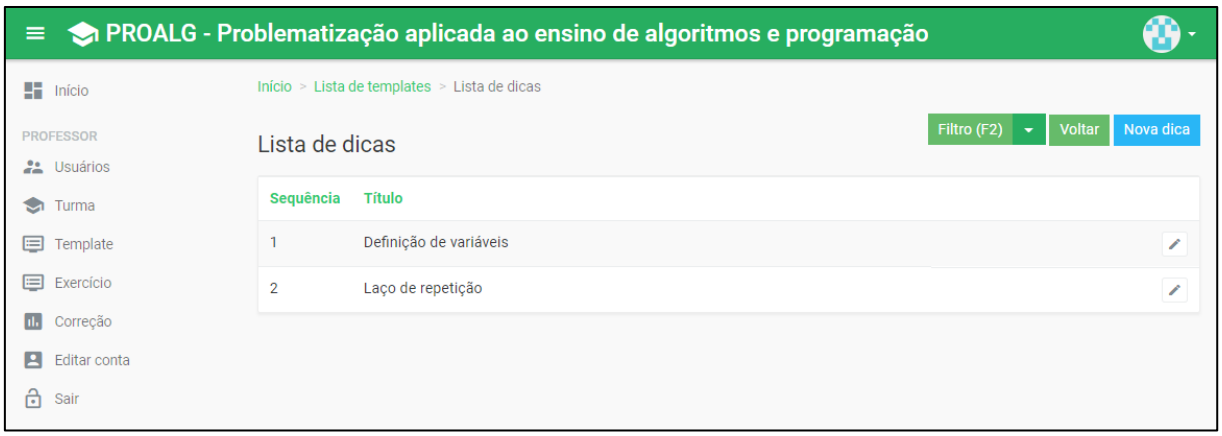
Figura 20 - Cadastro de dicas



Fonte: Elaborada pelo autor (2018).

As dicas que foram cadastradas e vinculadas ao *template* (FIGURA 21), podem ser acessadas pelos alunos posteriormente durante a resolução dos exercícios, por botões que estão disponíveis na tela de exercício. O acesso às dicas ficam registrados para que o professor possa saber quantas vezes e quais dicas o aluno utilizou.

Figura 21 - Lista de dicas do *template*

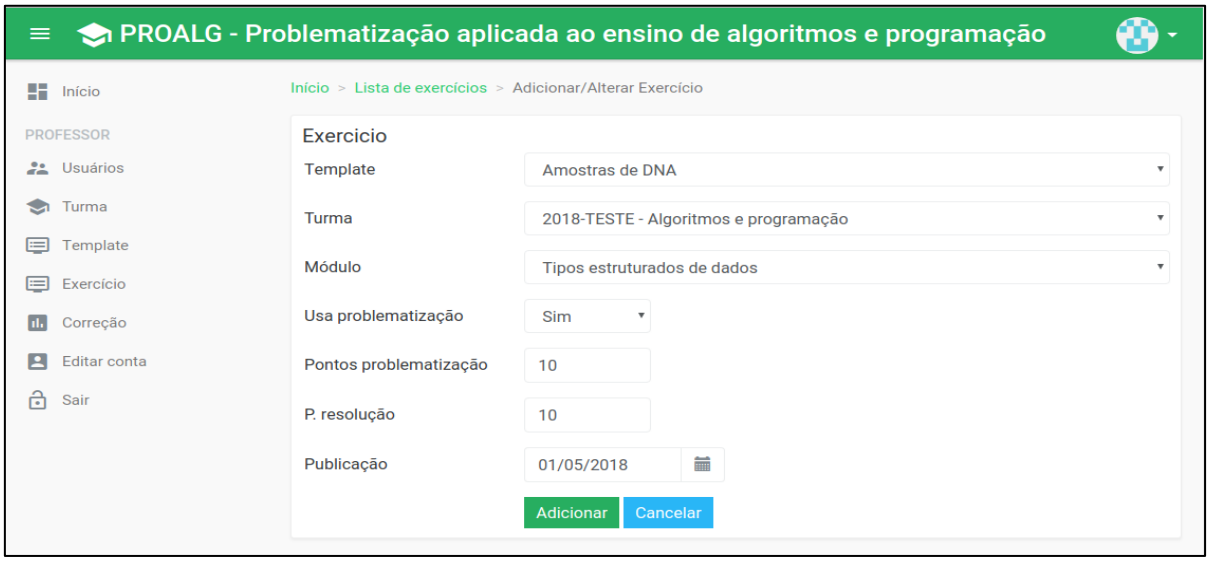


Fonte: Elaborada pelo autor (2018).

Após todos os cadastrados auxiliares serem realizados, o professor deve cadastrar os exercícios (FIGURA 22), selecionando o *template* cadastrado anteriormente, a turma a qual terá acesso ao exercício, um módulo, se deve utilizar problematização, o máximo de pontos da

problematização, o máximo de pontos da resolução e a data da publicação do exercício. Se no exercício não for utilizado problematização, então será solicitado somente o código fonte do aluno e não as etapas de pontos-chaves e hipóteses da solução.

Figura 22 - Cadastro de exercício



Fonte: Elaborada pelo autor (2018).

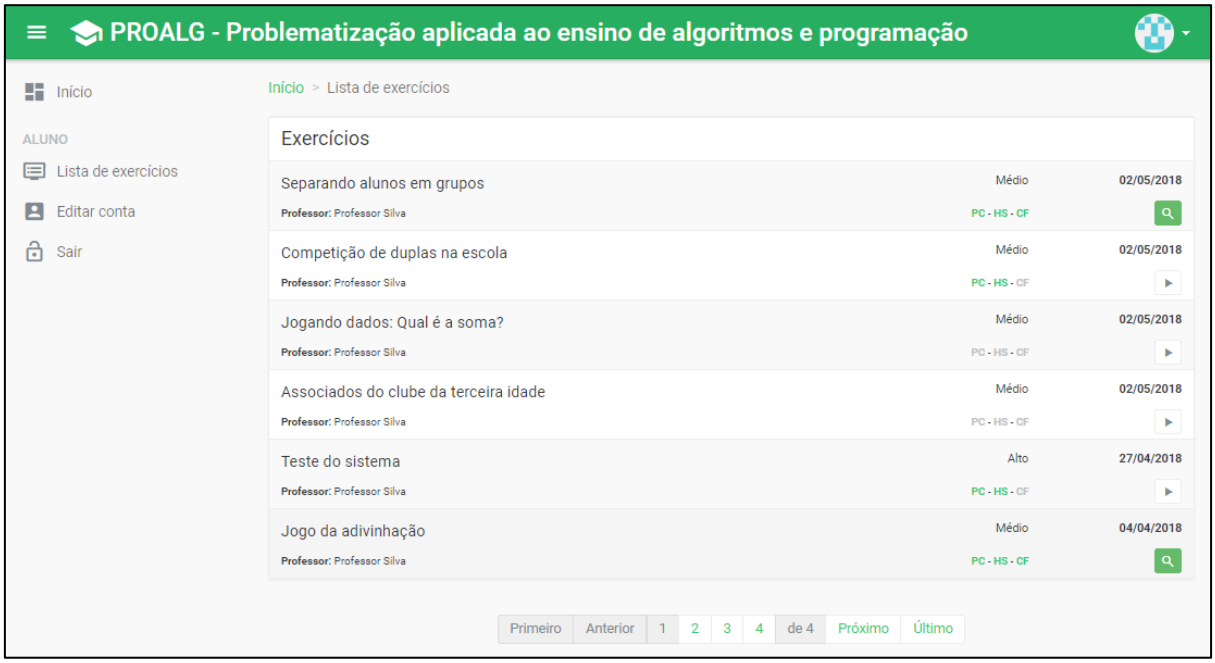
O próximo passo após o cadastro do exercício, é fazer a geração, esse processo vai criar um registro de exercício para cada aluno vinculado na turma. Para realizar a geração deve ser acessado a tela de listagem de exercício (FIGURA 23), clicado no botão gerar, que fica ao lado do botão de alterar. Após o sistema concluir o processo o botão será removido e adicionado um contador em seu lugar, informando a quantidade exercícios gerados.

Figura 23 - Lista de exercícios

Fonte: Elaborada pelo autor (2018).

O Aluno ao acessar o sistema, visualizará uma lista de cursos as quais ele está vinculado, ao selecionar o curso desejado, serão mostradas as disciplinas que ele cursou ou está cursando. Ao acessar a disciplina, será disponibilizada a lista de exercício do aluno (FIGURA 24), agrupada pelos seus respectivos níveis, acompanhado de algumas informações como o andamento do exercício.

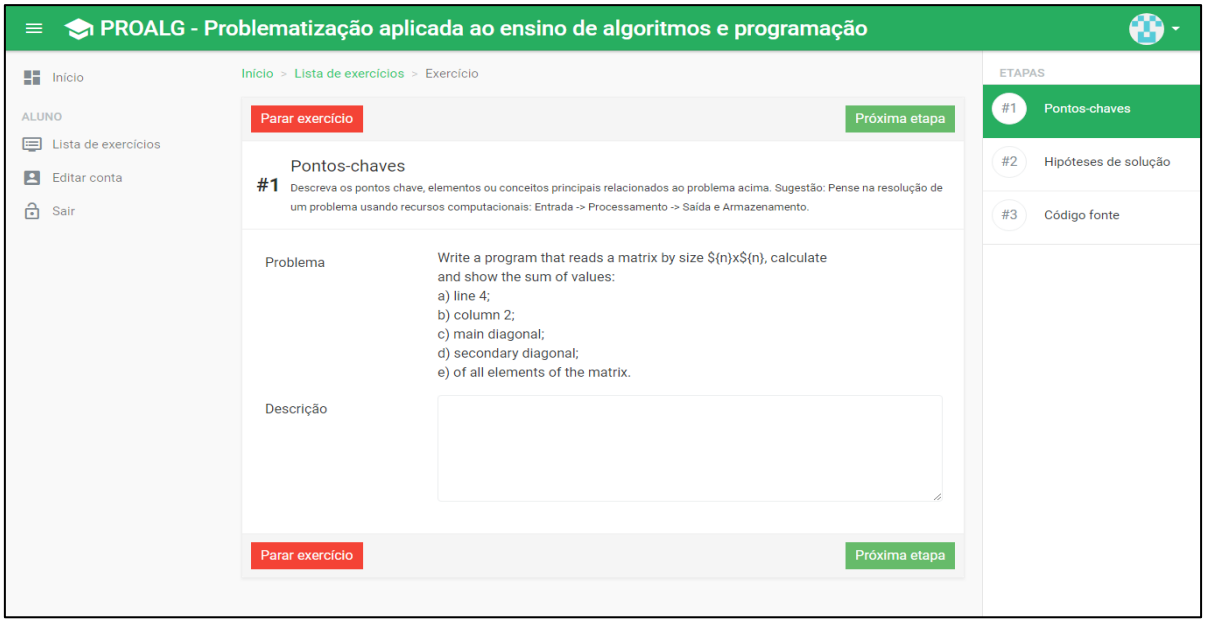
Figura 24 - Lista de exercício



Fonte: Elaborada pelo autor (2018).

Ao acessar o exercício o aluno será direcionado para uma tela onde serão listados os passos para a resolução do mesmo. Conforme mostrado na Figura 25 podemos ver que o primeiro passo é o pontos-chaves, neste momento será fornecido para o aluno a descrição do problema e instigado o mesmo a definir os pontos-chaves para a resolução do exercício. Durante todo o processo de resolução serão coletados diversos tipos de informação, para que posteriormente o professor possa utilizar as informações para uma análise no rendimento do aluno.

Figura 25 - Pontos chaves

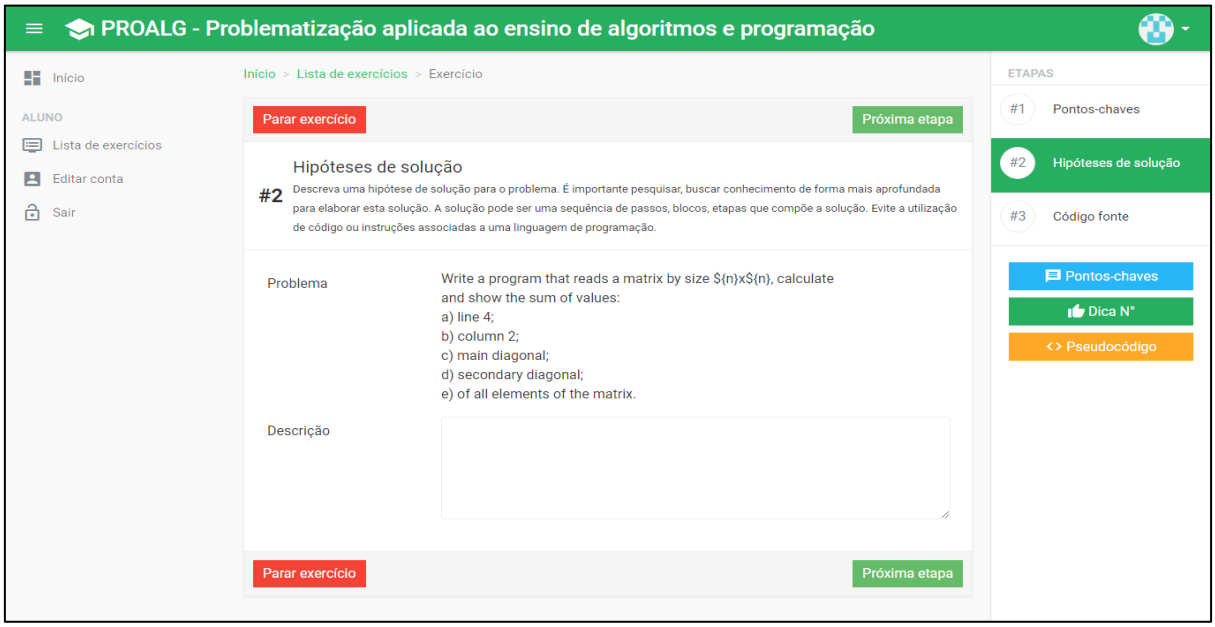


Fonte: Elaborada pelo autor (2018).

Após a elaboração dos pontos-chaves, o aluno deverá confirmar a conclusão, clicando no botão “Próxima etapa”, neste momento ele será direcionado para a próxima etapa do exercício, podendo visualizar somente como leitura a etapa anterior.

A hipótese será desenvolvida na segunda etapa do exercício, onde o aluno novamente com base no problema, deve elaborar uma resposta, tendo como base os pontos-chaves, dicas e pseudocódigo, conforme mostrado na Figura 26.

Figura 26 - Hipóteses de solução



Fonte: Elaborada pelo autor (2018).

Tendo o aluno desenvolvido os pontos chaves e hipótese do exercício, chegou o momento de realizar a criação do código fonte, para isso é fornecido um editor de código como mostrado na Figura 27, onde o aluno deve criar um programa que satisfaça os requisitos solicitados no exercício.

Figura 27 - Código fonte

Fonte: Elaborada pelo autor (2018).

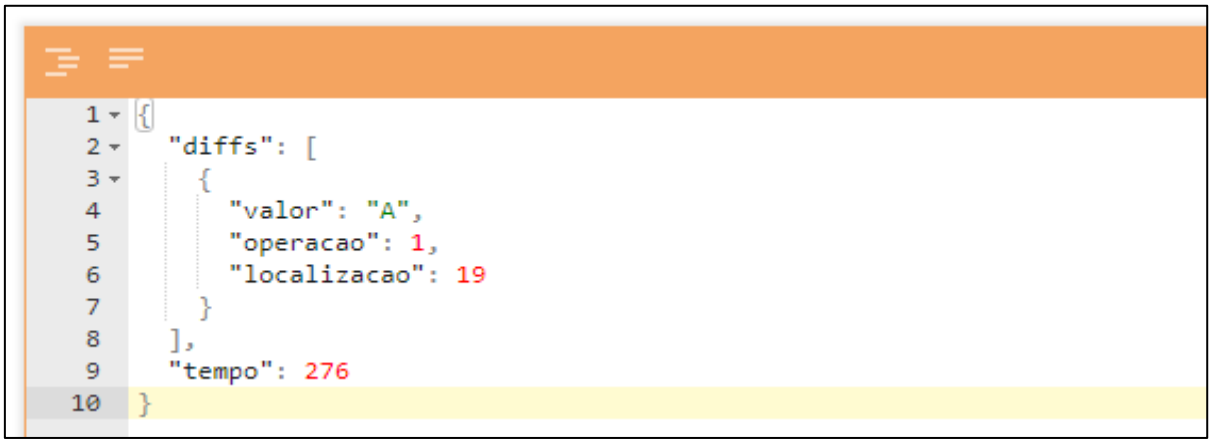
Durante o processo de criação do algoritmo pelo aluno, são gravados todos os movimentos do cursor dentro do editor, como também ações de copiar e colar. Com o armazenamento destas informações, o professor pode realizar uma análise no desenvolvimento realizado pelo aluno, isso tudo para que possa entender qual a linha de raciocínio utilizada por ele.

Para o editor de código, foi utilizado a biblioteca ACE.JS, a qual fornece um componente de edição de código orientada a eventos, desta forma pode ser coletado as informações monitorando os eventos do componente.

O monitoramento da escrita do código é realizada em tempo real, enviando os dados para o servidor de forma contínua, para isso foi utilizado o evento ‘onChange’ do editor, o qual é acionado sempre que o conteúdo do editor é alterado. Nesse momento é aplicado um algoritmo de diferencial, comparando o valor antigo com o valor novo do editor, assim localizando as alterações realizadas pelo aluno. O resultado dessa análise é um vetor chamado

DELTA, o qual possui duas posições, uma com o tempo gasto para a alteração em relação a última alteração realizada e outra com uma lista de vetores, do tipo DIFF como mostrado na Figura 28.

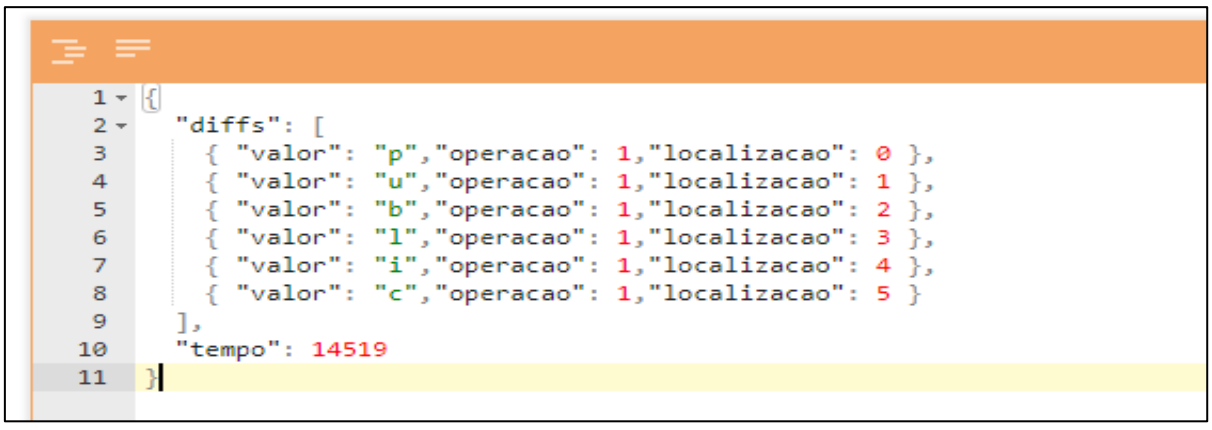
Figura 28 - Objeto delta



Fonte: Elaborada pelo autor (2018).

Cada objeto DIFF contém três propriedades, uma com o valor adicionado, outra com a operação realizada, a qual pode ser 1 (adicionado), -1 (removido) e 0 (sem alteração) e por último a localização da alteração, que é a sequência do caractere na linha. Quando for realizado uma colagem de código no editor, o evento 'onChange' é adicionado somente uma vez, e nesse momento é realizado diversas alterações ao mesmo tempo, neste caso será um objeto DELTA, com diversos objetos DIFFs, como pode ser visto na Figura 29.

Figura 29 - Objeto delta com múltiplos diffs



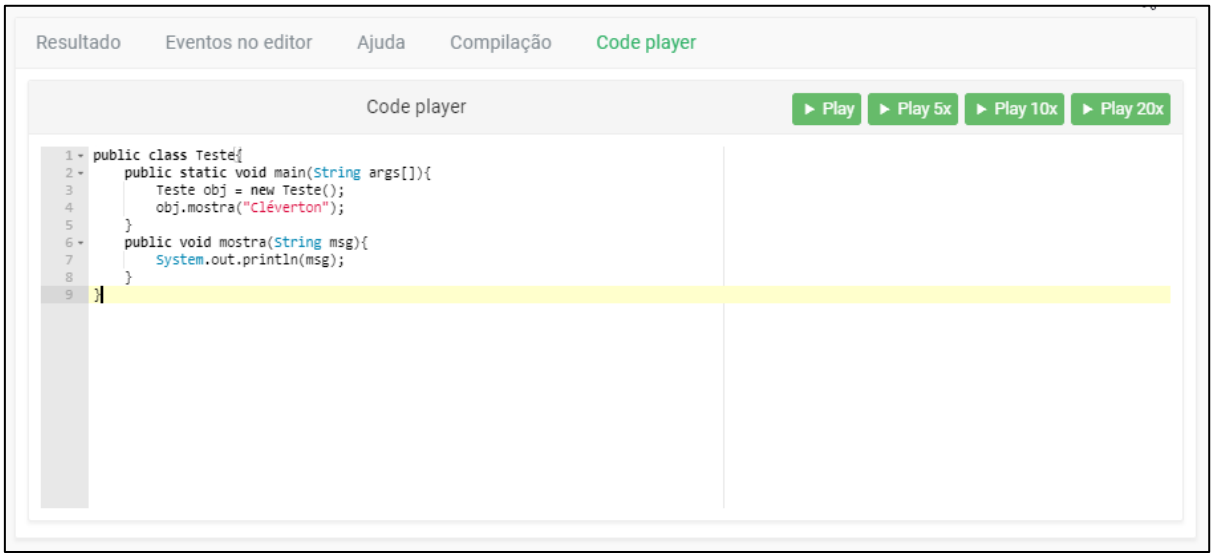
Fonte: Elaborada pelo autor (2018).

Foi utilizada a biblioteca 'google-diff-match-patch'[8]a qual implementa o algoritmo de diferencial de Myers (2003), que geralmente é considerado o melhor diferencial de uso geral.

[8] google-diff-match-patch disponibiliza algoritmos robustos para executar as operações necessárias para sincronizar texto sem formatação (https://code.google.com/archive/p/google-diff-match-patch/).

A visualização dos movimentos realizados no editor é através de um player (FIGURA 30), que reproduz de forma fiel todos os movimentos realizados pelo aluno durante a resolução do problema.

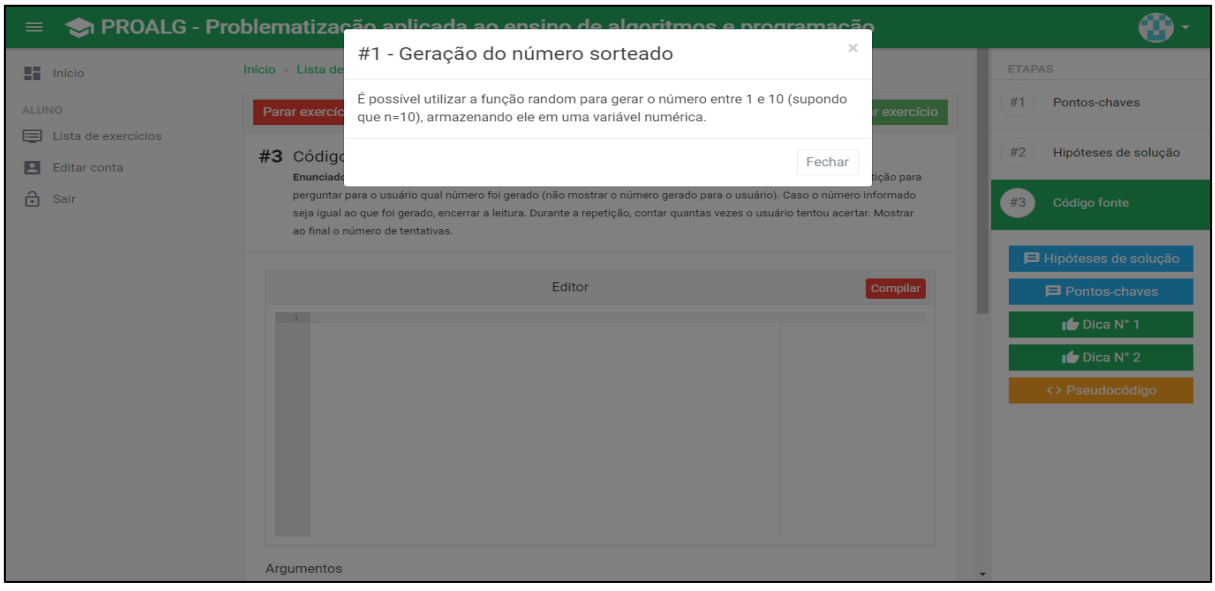
Figura 30 - Player do código fonte



Fonte: Elaborada pelo autor (2018).

Outra característica da tela de exercícios, é fornecer ajuda para o aluno durante o processo, está disponíveis botões de dicas (FIGURA 31) e outro com o pseudocódigo, quando clicado em cada um deles será mostrado por uma janela a informação solicitada. A ajuda somente é permitida na segunda e terceira etapa e todas as solicitações serão gravadas.

Figura 31 - Tela de dicas



Fonte: Elaborada pelo autor (2018).

Outra característica da tela de exercício será o botão ‘parar exercício’, com o qual o aluno informa que está desistindo de resolver o problema.O aluno pode em qualquer momento e em qualquer etapa decidir parar de trabalhar no exercício podendo retornar onde parou em outro momento. Para cada tentativa, será criado um novo registro, sendo assim possível analisar quantas vezes o aluno tentou trabalhar em cada exercício.

Durante a terceira e última etapa do exercício, o aluno pode testar o código fonte codificado por ele. Durante esse processo de testes, é disponibilizado diversas informações ao aluno, como o estado e possíveis erros ocorridos durante a construção ou execução do programa, como também a saída gerada pelo programa, conforme mostrado na Figura 32.

Figura 32 - Teste do programa

Fonte: Elaborada pelo autor (2018).

Para realizar uma interação com o programa, é disponibilizado um campo de entrada de dados, que neste caso são os argumentos, os quais devem ser informados antes da execução do programa. Esses parâmetros podem ser acessados dentro do método ‘main’ do programa, através de um vetor recebido por este método, podemos ver na Figura 33 um caso de uso.

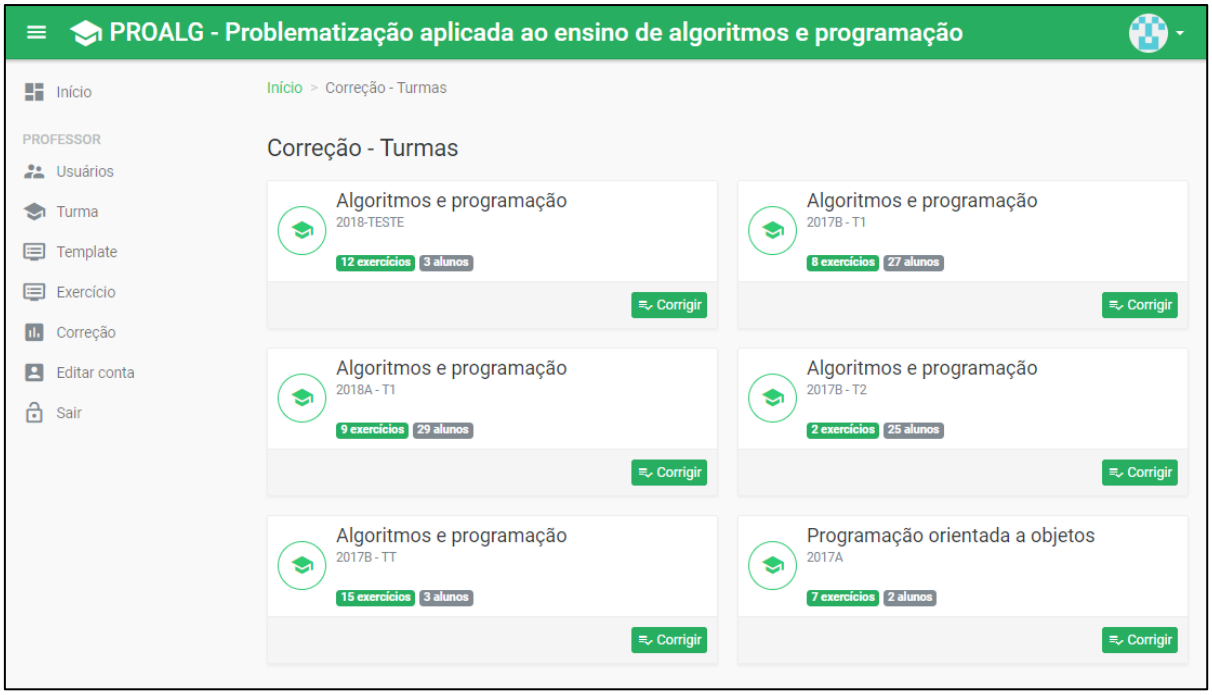
Figura 33 - Programa de exemplo

```
1  public class teste{
2      public static void main(String args[]){
3          //Primeiro argumento
4          System.out.println(args[0]);
5          //Segundo argumento
6          System.out.println(args[1]);
7          //Terceiro argumento
8          System.out.println(args[2]);
9      }
10 }
11
```

Fonte: Elaborada pelo autor (2018).

Após o aluno concluir o exercício, o mesmo fica disponível ao professor para realizar a correção, para isso deve ser acessado a opção “Correção” do menu. O primeiro passo é selecionar a turma (FIGURA 34) que deseja realizar a correção, as turmas são mostradas em forma de bloco, informando a quantidade de exercícios como também a quantidade de alunos, para continuar deve ser pressionado o botão “Corrigir”.

Figura 34 - Lista de turmas na correção

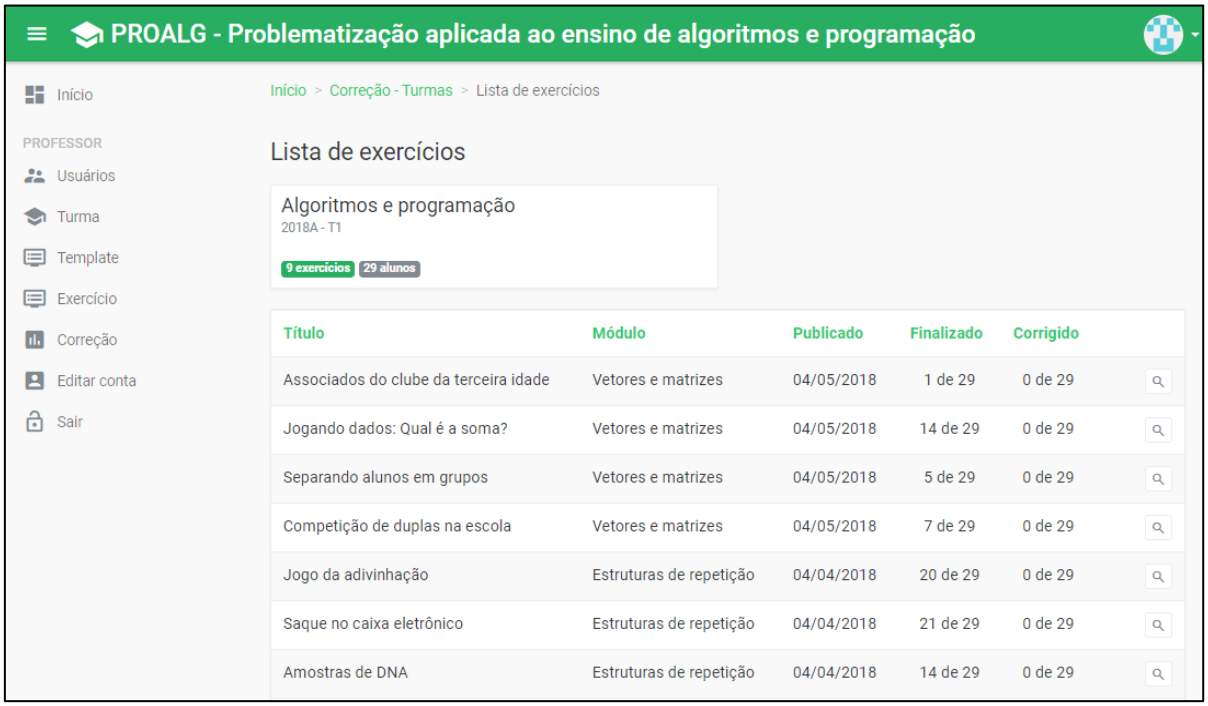


Fonte: Elaborada pelo autor (2018).

Ao selecionar a turma desejada, será carregado outra tela, na qual está disponível as mesmas informações da tela anterior sobre a turma e também uma lista de exercícios (FIGURA 35), essa lista está ordenada pela data de publicação, fornecendo também algumas características do exercício como título, módulo, totalizador de exercícios finalizados e

corrigidos. O próximo passo é escolher o exercício para então visualizar os alunos que entregar a atividade.

Figura 35 - Lista de exercícios de uma turma na correção



Fonte: Elaborada pelo autor (2018).

O terceiro passo na correção dos exercícios é escolher um aluno para avaliar, na listagem (FIGURA 36) é mostrado o nome dos alunos e as etapas que foram finalizadas, que pode ser os Pontos-chaves, Hipótese da solução e Código fonte. Quando as três etapas forem finalizadas, fica disponível a data e hora da entrega como também o acesso da tela de correção do exercício. Após finalizar o processo de correção pode ser visualizado na listagem as notas e as avaliações de níveis.

Figura 36 - Lista de alunos de um exercício na correção

Fonte: Elaborada pelo autor (2018).

A tela de correção de exercício (FIGURA 37) disponibiliza diversas informações, algumas em forma de métricas, as quais ficam disponíveis em um painel na parte superior:

- Tentativas: tentativas do aluno para realizar a atividade.
- Ajuda: quantidade de vezes que o aluno utilizou as ajudas, que podem ser as dicas, etapas do exercício e pseudocódigo.
- Tempo ponto-chave: tempo gasto pelo aluno para realizar a primeira etapa do processo, é utilizado somente o tempo da tentativa que foi entregue, as demais não são computadas.
- Tempo hipótese: tempo gasto pelo aluno para realizar a segunda etapa do exercício, é utilizado somente o tempo da tentativa que foi entregue.
- Tempo código fonte: tempo gasto pelo aluno para realizar a escrita do código fonte, esse valor inclui todo o processo de teste do programa.
- Tempo total: soma dos três tempos anteriores, tempo ponto-chave, tempo hipótese e tempo código fonte.

- Avaliação do aluno: ao realizar a entregar da última etapa, é solicitado ao aluno uma pequena avaliação composta por uma pergunta solicitando o grau de dificuldade do exercício, sendo possível escolher entre “Baixo”, “Médio” e “Alto” e um campo de comentário.

- Compilações abortadas: quantidade de compilação que foram abortadas de forma automática pelo sistema, isso ocorre quando o tempo de compilação ultrapassa os 5 segundos.

- Compilação OK: quantidade de compilações realizadas com sucesso.

- Compilação ERRO: quantidade de compilação não finalizadas.

- Execução OK: quantidade de execuções realizadas com sucesso.

- Execução ERRO: quantidade de execuções com problemas.

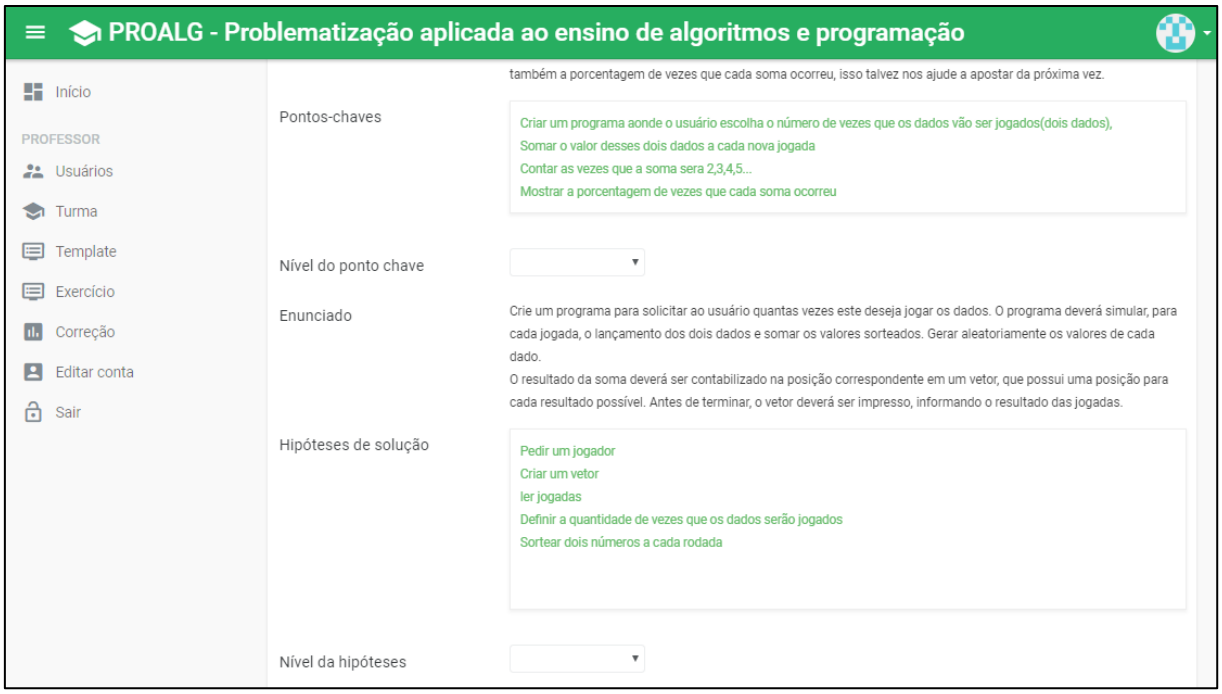
Abaixo do painel está disponível as informações do exercício acompanhada da resposta do aluno. Tudo isso disponível para que o professor perceba como os alunos fizeram a atividade, acompanhando o andamento e desta forma possa contribuir para melhorar o processo de ensino e aprendizagem.

Figura 37 - Tela de correção de exercício

Fonte: Elaborada pelo autor (2018).

Abaixo da descrição do problema está os pontos-chaves elaborados pelo aluno (FIGURA 38), o professor deve avaliar essa informação e selecionar uma opção no campo “Nível dos pontos-chaves”, as opções disponíveis são “Baixo”, ”Médio” e “Alto”. A hipótese da solução também deve ser avaliação, selecionado uma opção no campo “Nível da hipótese”.

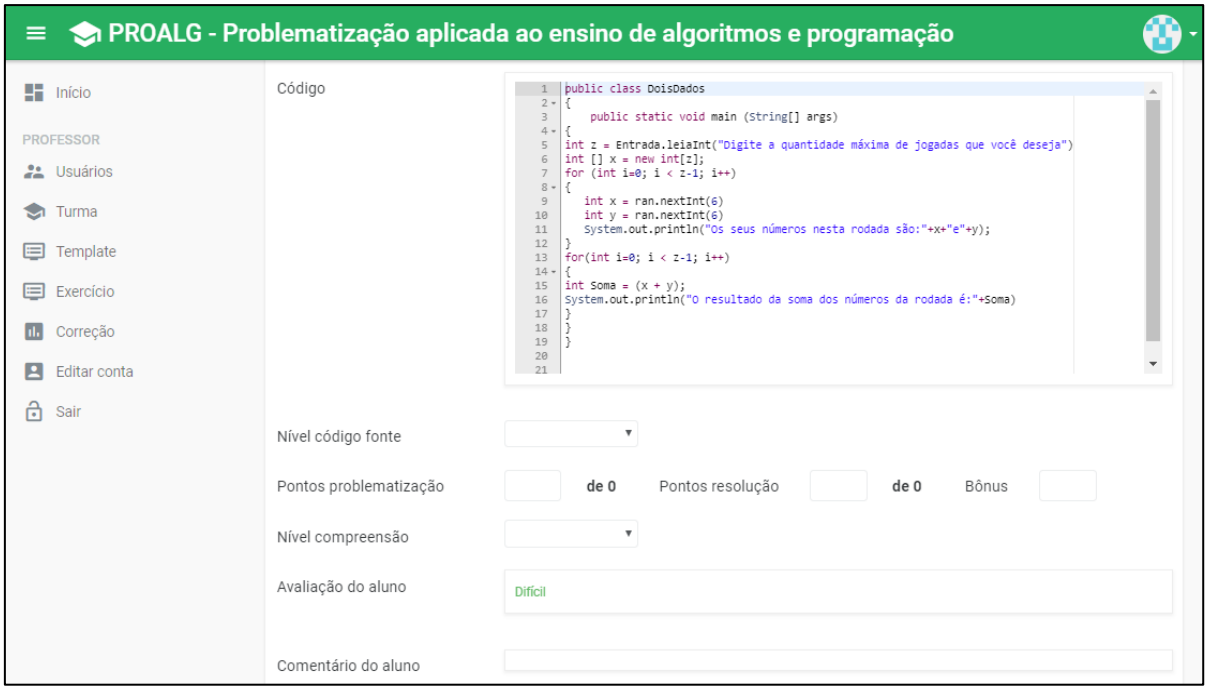
Figura 38 - Tela de correção de exercício



Fonte: Elaborada pelo autor (2018).

Logo abaixo do campo de “nível de hipóteses” está disponível o código fonte criado pelo aluno (FIGURA 39), esse código também deve passar pelo processo de avaliação, selecionando uma opção no campo “Nível do código fonte” que são “Ruim”,“Regular” e “Bom”. Também está disponível os campos para o professor preencher os valores das notas do aluno, logo abaixo pode ser visualizado a avaliação do aluno.

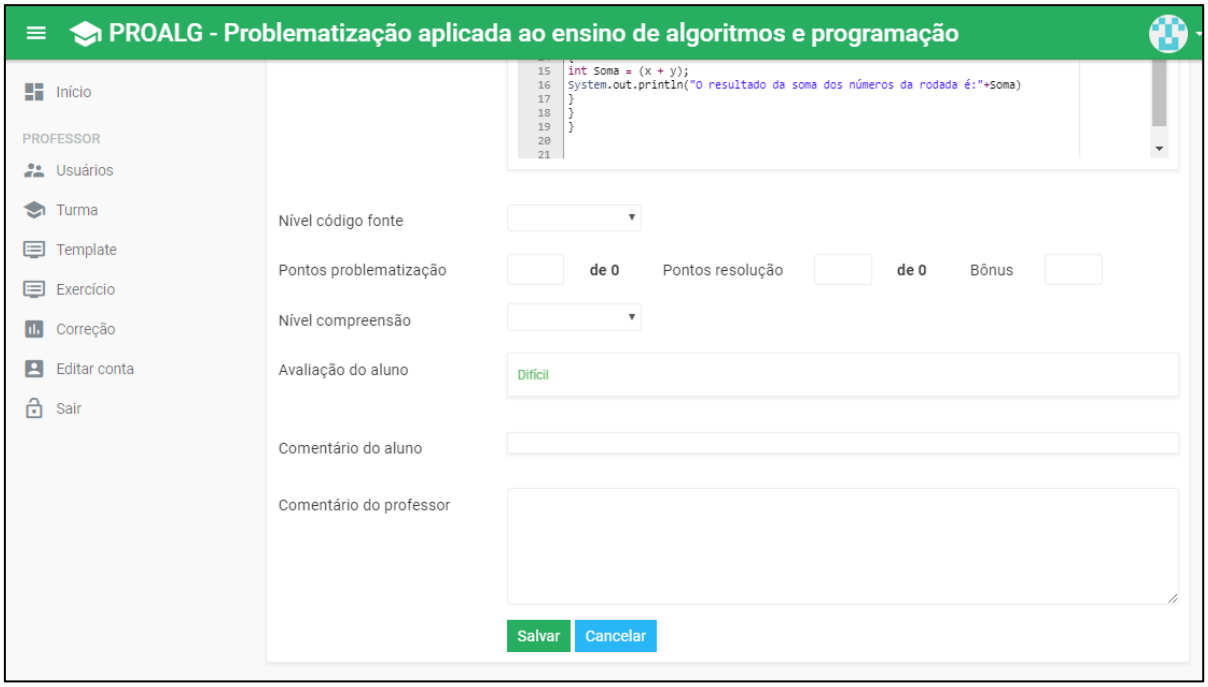
Figura 39 - Tela de correção de exercício



Fonte: Elaborada pelo autor (2018).

Por último o professor pode deixar um comentário para o aluno, descrevendo sua correção, uma sugestão de melhora como também um elogio (FIGURA 40). O comentário do professor será mostrado para o aluno após a correção na tela de resumo do exercício, que é mostrada quando o aluno acessa o exercício após finalizar o mesmo.

Figura 40 - Tela de correção de exercício



Fonte: Elaborada pelo autor (2018).

Para que o professor possa realizar a correção dos exercícios, fica disponível algumas informações as quais são registradas durante a realização da atividade, podemos visualizar os eventos no editor, visualização de ajudas e todos os testes do código fonte. Os eventos de copiar e colar dentro do editor são registrados, com a sua data e hora e valor copiado ou colado como mostrado na Figura 41.

Figura 41 - Eventos no editor na correção de exercício

Fonte: Elaborada pelo autor (2018).

Outra informação muito importante é a quantidade de ajuda solicitada pelo aluno, esses pedidos de ajuda podem ser visualizados na listagem que está disponível na aba “Ajuda” (FIGURA 42). Fica registrado em qual etapa foi visualizado e o tipo da ajuda que neste caso pode ser dica, etapa ou pseudocódigo. Com esse tipo de histórico o professor pode conhecer melhor a forma que o aluno resolveu o problema, se foi por tentativa e erro ou buscando soluções já existentes.

Figura 42 - Ajudas solicitadas durante o exercício

Fonte: Elaborada pelo autor (2018).

Durante a elaboração do código fonte, o aluno pode testar o seu programa, para isso é disponibilizado a funcionalidade de compilar e executar, esse processo é monitorado e o professor pode visualizar todas as tentativas posteriormente, como visto na Figura 43. É apresentado uma lista de compilações, nela constam diversas informações como: a data e hora, código fonte, argumentos, status da compilação e execução, tempo de compilação e execução, como também se o processo foi abortado pelo sistema.

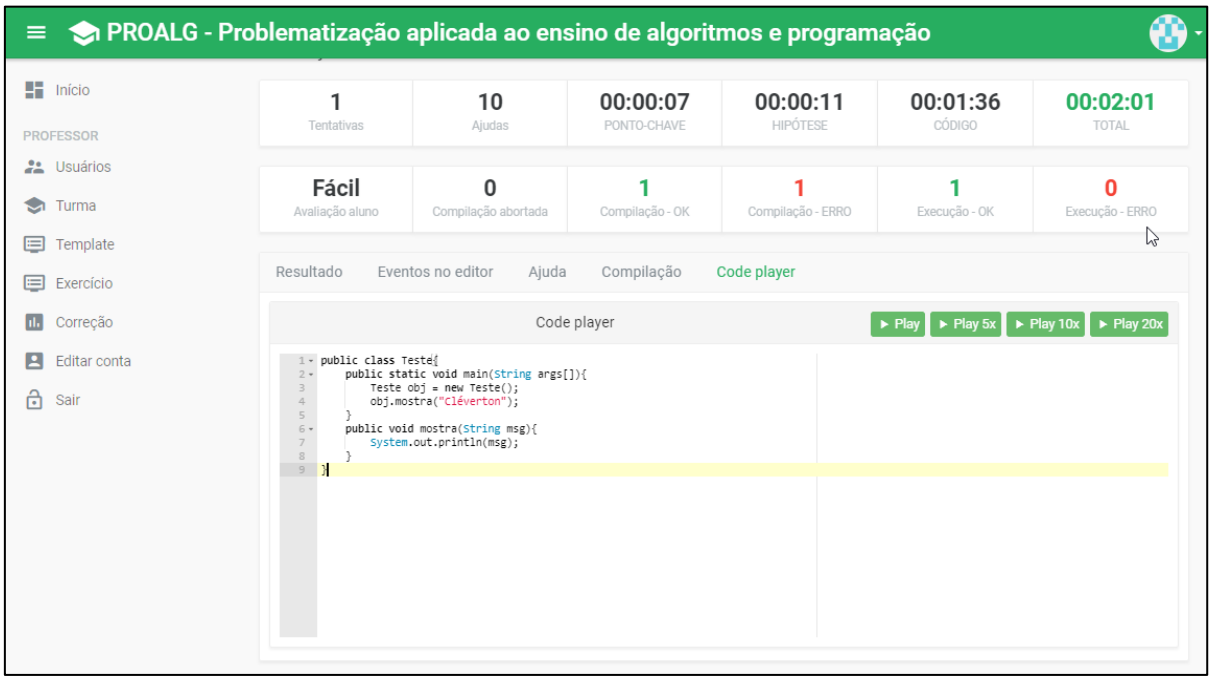
Figura 43 - Lista de compilações

Fonte: Elaborada pelo autor (2018).

Outro recurso da tela de correção é o player de código (FIGURA 44), pode ser visualizado todos os passos no desenvolvimento do algoritmo que o aluno utilizou, está disponível vários botões para que possa ser utilizado diversas velocidades na visualização.

A visualização pode ser parada em qualquer momento e copiado o código fonte, já que não se trava de uma gravação em vídeo e sim de uma sequência de passos utilizadas pelo aluno no editor. O player utiliza o mesmo editor de código disponibilizado na tela de exercício para o aluno.

Figura 44 - Player do código fonte



Fonte: Elaborada pelo autor (2018).

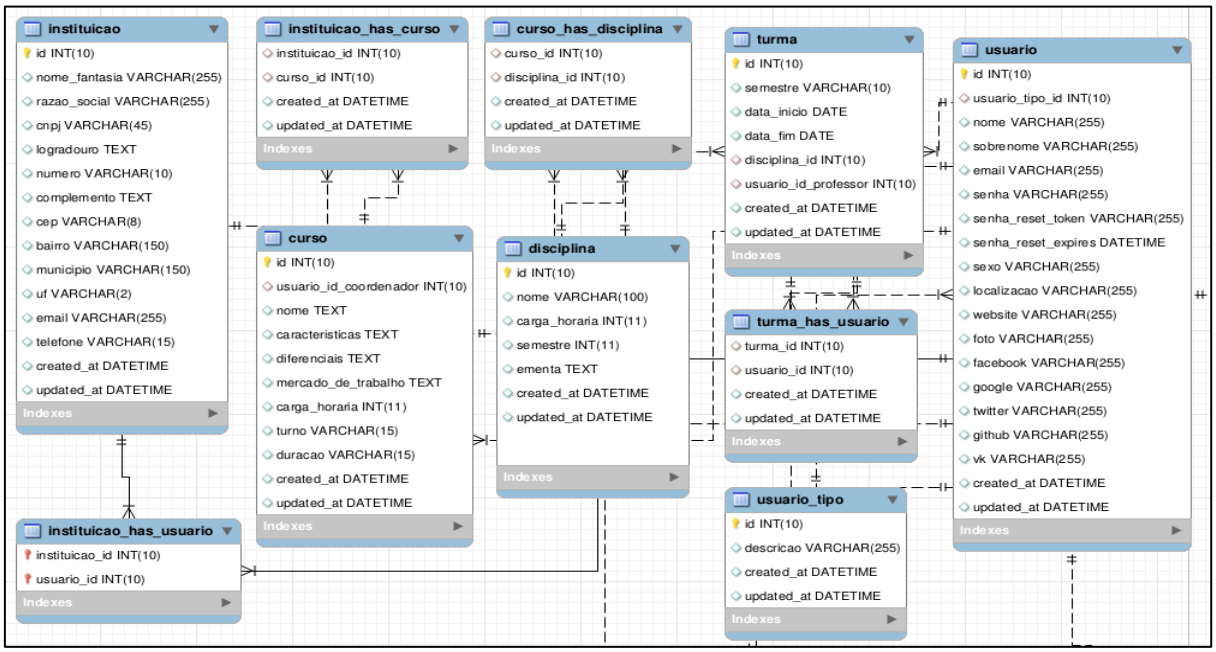
Após o término da correção, é direcionado para a tela de listagem de alunos mas agora com as informações de notas e níveis preenchidos (FIGURA 45). Essa tela ordena os registro de uma forma que fique na parte superior sempre os exercícios finalizados e que não foram corrigidos, assim permitindo uma melhor organização do professor na hora de realizar a correção dos trabalhos.

Figura 45 - Lista de alunos de um exercício na correção

Fonte: Elaborada pelo autor (2018).

O modelo ER (Entidade Relacionamento) é uma ferramenta utilizada para demonstrar o projeto da base de dados que suporta um sistema. O modelo ER relacional apresenta detalhes sobre as tabelas, atributos, além das restrições de integridades (chaves primárias, chaves estrangeiras). O modelo de dados para o sistema proposto é mostrado nas Figura 46, Figura 47 e Figura 48. Na primeira imagem são apresentadas as tabelas básicas, cadastros de usuários (alunos e professores), tipos de usuário, cursos, disciplinas, instituição e turmas.

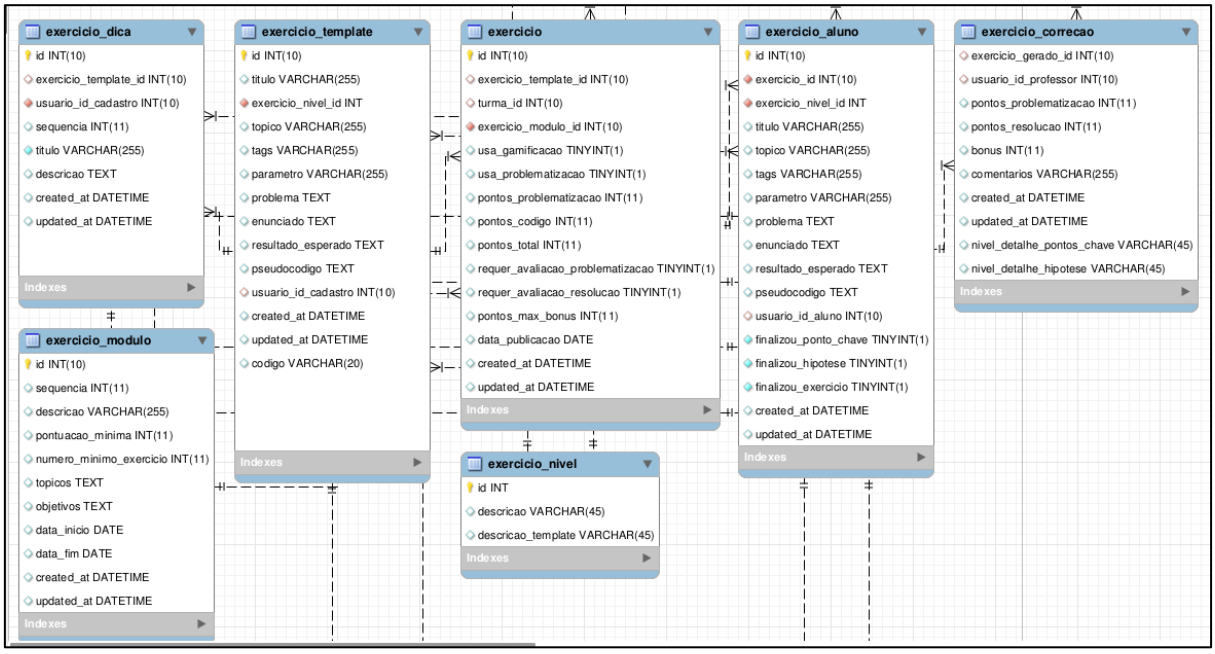
Figura 46 - Tabelas básicas



Fonte: Elaborada pelo autor (2018).

Além dos cadastros citados, é possível destacar as tabelas que armazenam os *templates* e os exercícios, a alocação destes para cada estudante, a correção por parte dos professores como também o cadastro de dicas, módulos e níveis do exercício (exercicio_template, exercicio, exercicio_aluno, exercicio_correcao, exercicio_dica, exercicio_modulo e exercicio_nivel).

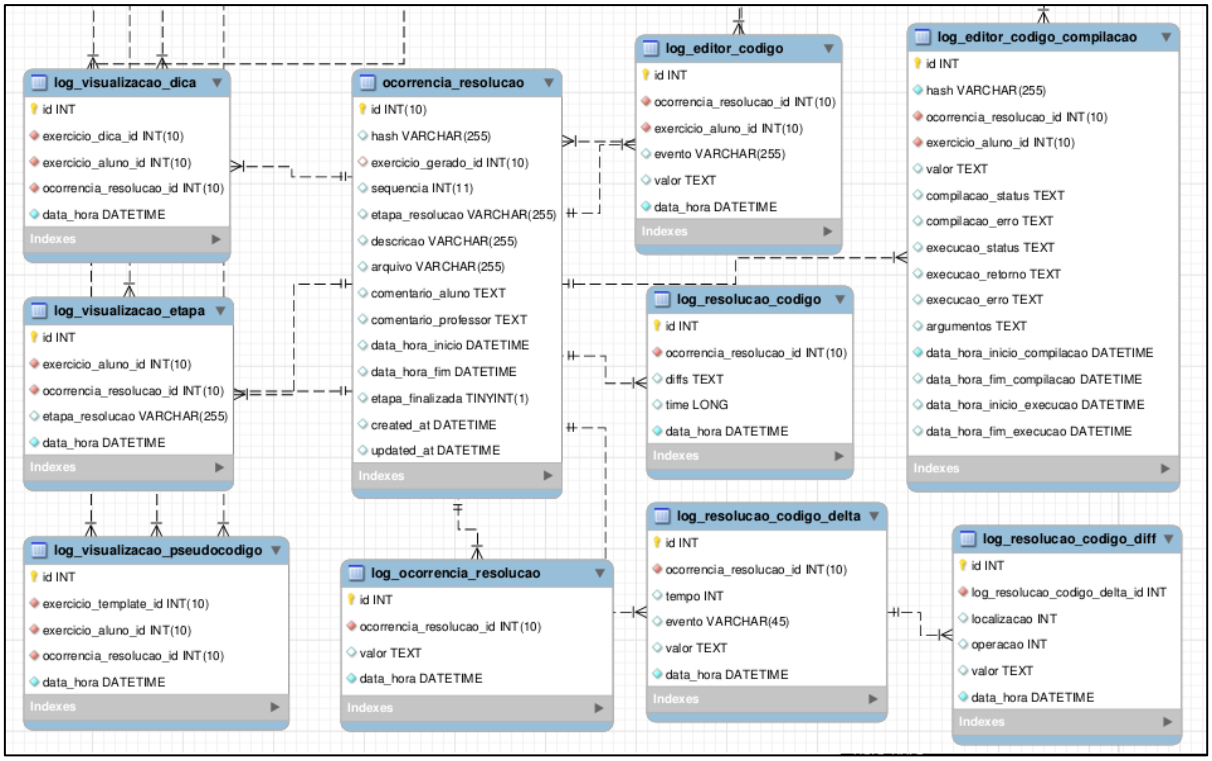
Figura 47 - Tabela de exercícios



Fonte: Elaborada pelo autor (2018).

Também é utilizado um conjunto de tabelas com o propósito de armazenar todas as ocorrências e ações do estudante, enquanto este realiza um exercício, sendo a principal delas denominada ocorrencia_resolucao (FIGURA 48). O conteúdo postado para cada questão, as incidências de entrada e saída da atividade, são registradas nesta tabela, acompanhada da data e hora em que ocorreram. Foram propostas tabelas para armazenar ações específicas, como o acesso as dicas (log_visualizacao_dica, log_visualizacao_etapa, log_visualizacao_pseudocodigo), além de detalhes sobre a escrita do código fonte, como ações de copiar/colar, edição de trechos da solução (log_editor_codigo, log_ocorrencia_resolucao, log_resolucao_codigo, log_resolucao_codigo_dela, log_resolucao_codigo_diff, log_editor_codigo_compilacao).

Figura 48 - Tabela de ocorrências



Fonte: Elaborada pelo autor (2018).

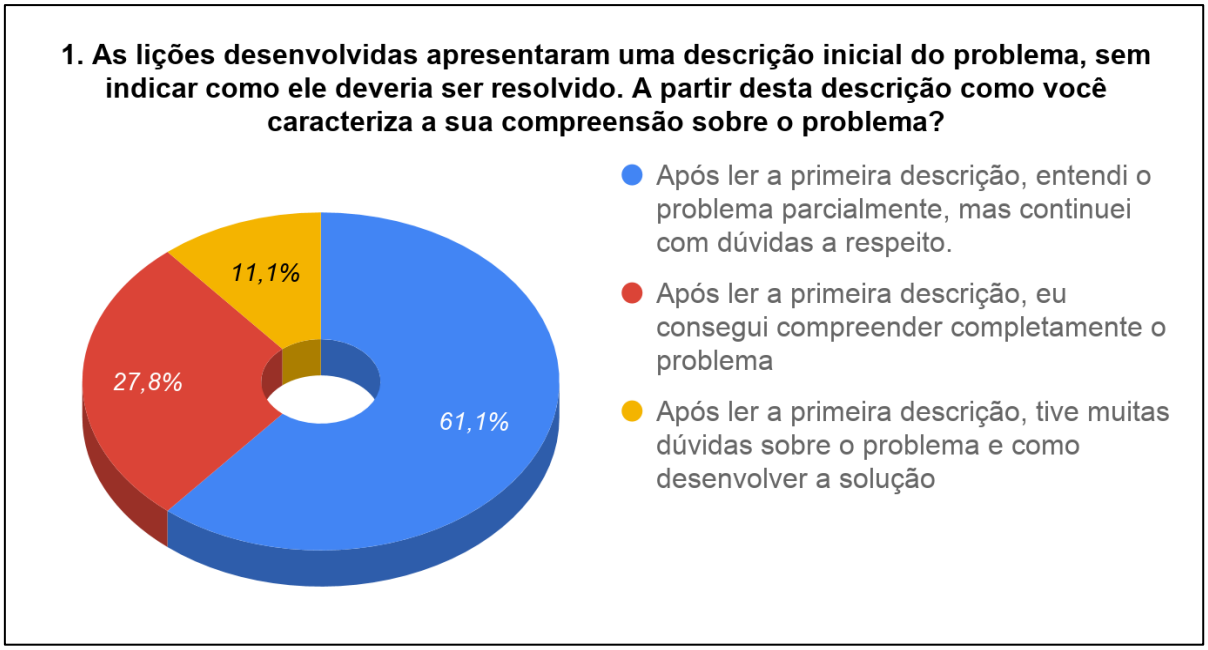
6 RESULTADOS E DISCUSSÕES

Esta seção descreve os resultados da aplicação do sistema em um teste realizado no segundo semestre de 2017 e primeiro semestre de 2018. Participaram das atividades 28 estudantes de uma disciplina de Algoritmos e Programação, dos cursos de Engenharia de Software e Engenharia da Computação da Univates, os alunos em sua maioria eram de ingressantes, os quais estavam cursando o seu primeiro semestre, sendo assim, a maioria estavatendo o seu primeiro contato com a programação.

Foi aplicado um questionário com diversas questões de múltipla escolha, com objetivo de avaliar a percepção dos alunos na utilização da metodologia de problematização e do software utilizado durante a realização das atividades.

A seguir serão apresentadas e analisadas estatísticas relacionadas ao questionário. A primeira questão busca verificar se os alunos compreenderam como realizar a resolução do exercício apenas lendo a descrição do problema. Observa-se que a maioria dos estudantes (61,1%) entenderam parcialmente o problema após ler a primeira descrição, mas continuam com dúvidas, já 27,8% conseguiram entender completamente o problema após a leitura e o restante da turma (11,1%) tiveram diversas dúvidas sobre o problema e como desenvolver a solução (GRÁFICO 1).

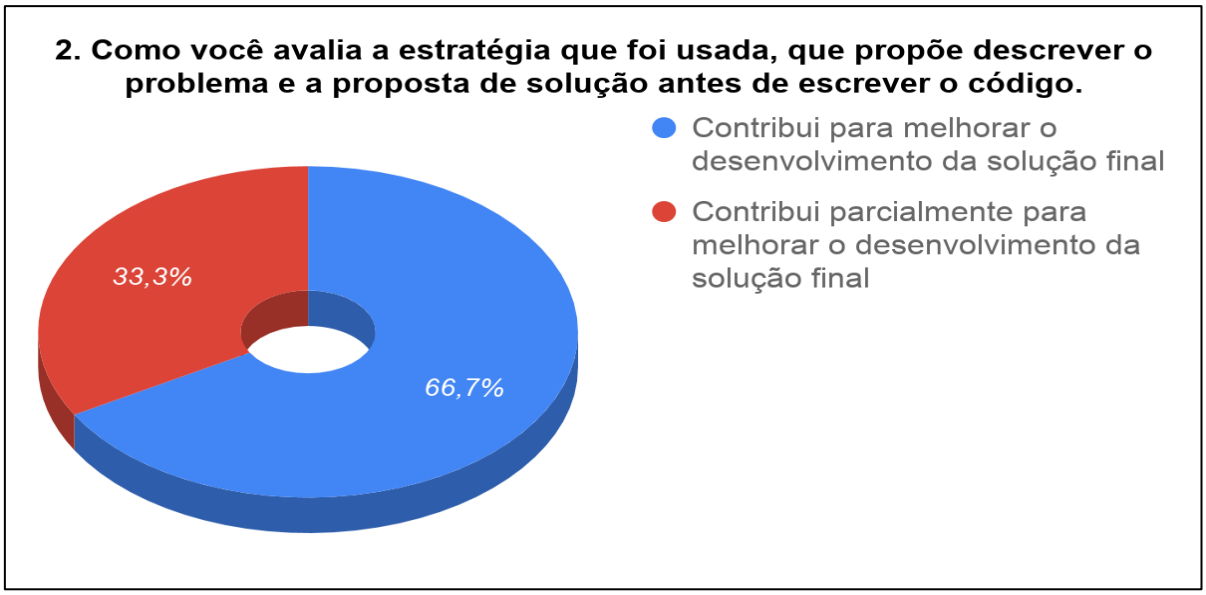
Gráfico 1 - Resultado da primeira pergunta



Fonte: Elaborado pelo autor (2018).

A segunda questão avaliou a percepção dos alunos em relação a metodologia utilizada pelo sistema, na resolução do exercício. Observa-se que a totalidade dos estudantes acredita que esta abordagem contribui total (67%) ou parcialmente (33%) para melhorar do desenvolvimento da solução (GRÁFICO 2).

Gráfico 2 - Resultado da segunda pergunta

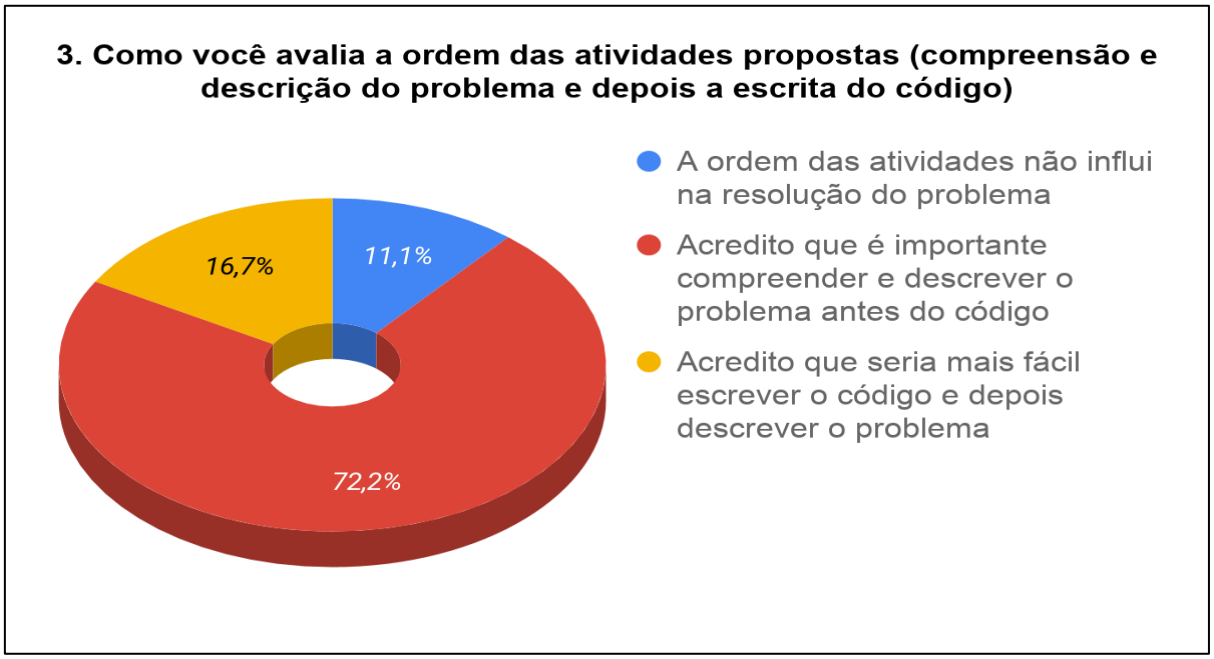


Fonte: Elaborado pelo autor (2018).

Quando os alunos foram questionados na ordem dos passos (GRÁFICO 3), obrigando a elaborar os pontos-chaves e hipótese da solução antes de escrever o código fonte, a maioria

(72,2%) entende que é importante compreender e descrever o problema primeiro. Contudo, parte dos alunos gostaria de elaborar primeiro o código fonte e após isso descrever o problema, o que acaba mostrando a visão tradicional da maioria das disciplinas de programação.

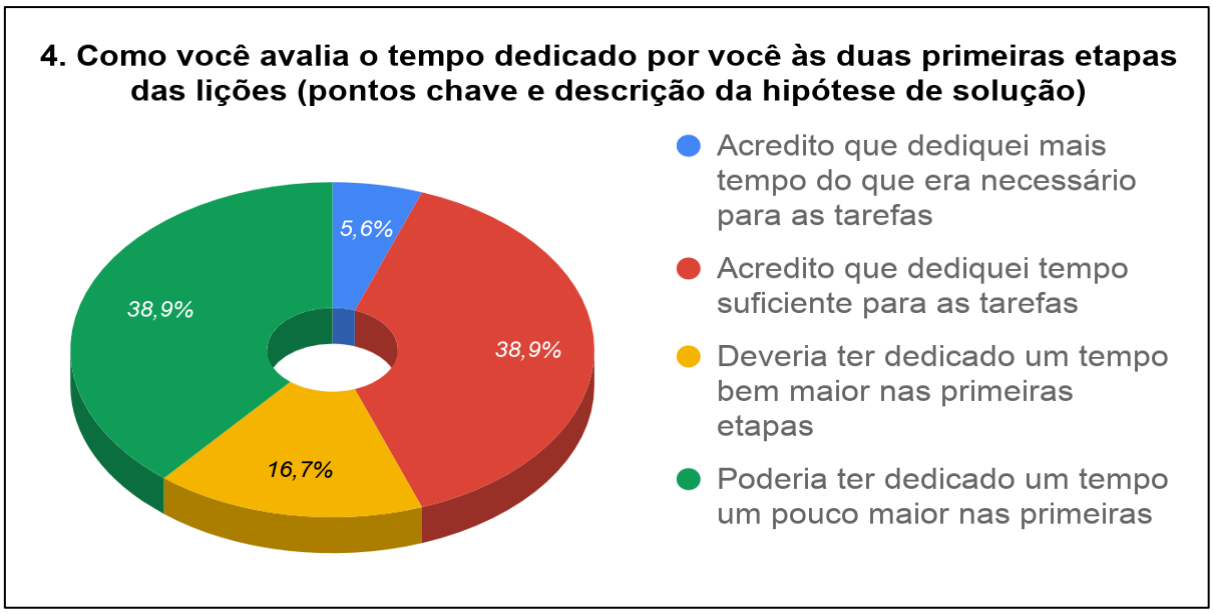
Gráfico 3 - Resultado da terceira pergunta



Fonte: Elaborado pelo autor (2018).

Os alunos também foram questionados em relação ao tempo que destinaram para a realização das duas primeiras etapas do exercício (pontos-chaves e descrição da hipótese de solução). A turma acabou ficando dividida em relação ao tempo, já que 38,9% dos alunos acreditam que poderia ter dedicado mais tempo e 38,9% acreditam ter utilizado o tempo necessário para realizar as duas primeiras etapas como mostrado noGráfico 4.

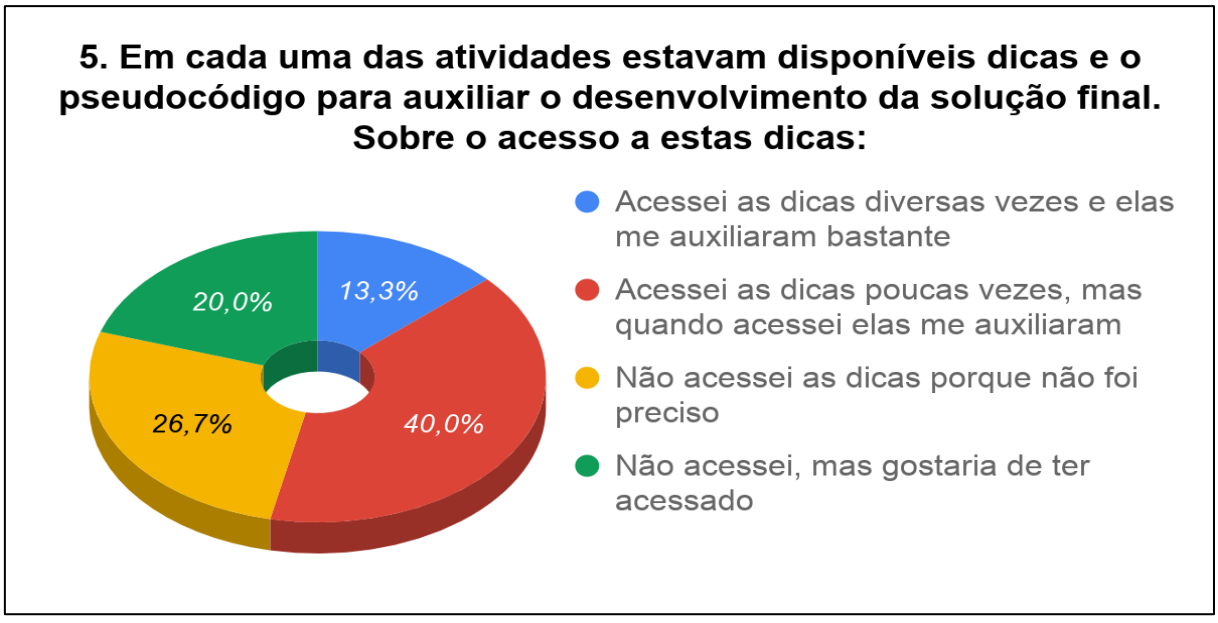
Gráfico 4 - Resultado da quarta pergunta



Fonte: Elaborado pelo autor (2018).

Outro ponto avaliado foi em relação aos recursos de ajuda disponibilizados no sistema, a maioria dos alunos declarou ter acessado os recursos de ajuda pelo menos uma vez, sendo que para 26,7% não julgaram necessário acessar e 20% acreditam que deveria ter acessado (GRÁFICO 5).

Gráfico 5 - Resultado da quinta pergunta

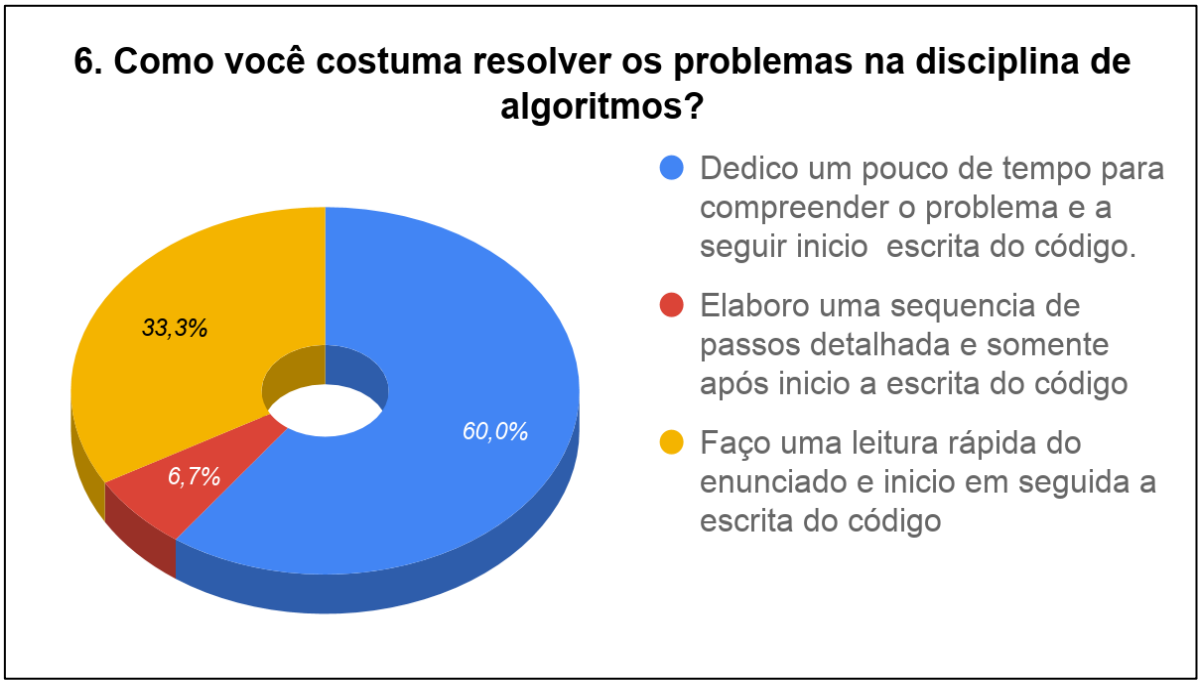


Fonte: Elaborado pelo autor (2018).

A sexta questão abordou como os alunos costumam resolver os problemas nas disciplinas de algoritmos, observa-se que a maioria dos alunos (60%) dedica um tempo para

compreender o problema antes de iniciar a escrita do código, já 33,3% dos alunos realizam uma pequena leitura do enuncia e 6,7% costumam primeiramente elaborar uma sequencia de passos e após isso iniciam a escrita do código.

Gráfico 6 - Resultado da sexta pergunta

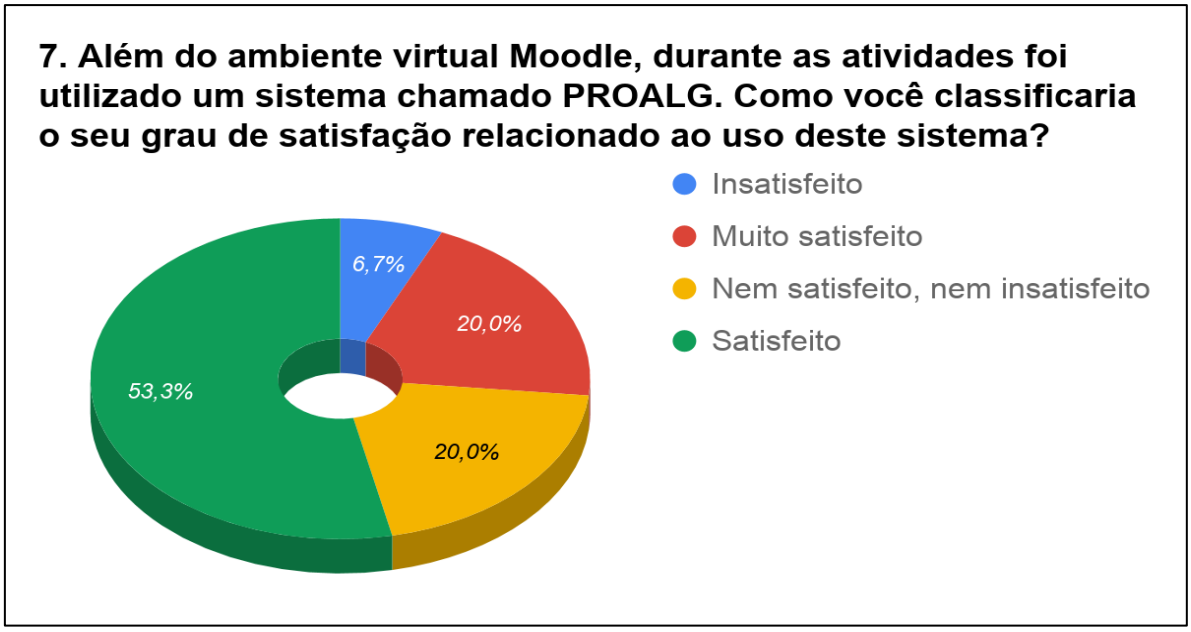


Fonte: Elaborado pelo autor (2018).

Por último foi realizado uma avaliação de satisfação sobre o sistema PROALG, a qual mostrou que a grande maioria dos alunos se demonstra satisfeito (50%) ou muito satisfeito (22%), apenas 6% dos alunos declarou estar insatisfeito com a ferramenta (GRÁFICO 7).

Os resultados apontam para um cenário promissor, mesmo que existam ajustes e novos recursos a serem implementados, o sistema demonstrou ser uma boa alternativa, tanto no aspecto do método usado, quanto na qualidade dos recursos disponibilizados aos estudantes.

Gráfico 7 - Resultado da sétima pergunta



Fonte: Elaborado pelo autor (2018).

Além do questionário, durante a disciplina foi disponibilizado um fórum no ambiente virtual da Univates para que os alunos se manifestassem através de comentários e sugestões. Foram postados 35 comentários, estes feitos por 20 estudantes diferentes, o que permite concluir que a participação no fórum foi significativa, tendo a participação de 64% dos alunos. A maioria os comentários sobre a ferramenta foi positivo, os aspectos que foram considerados negativos pelos alunos ou as sugestões de melhoria, em geral, estavam relacionadas ao editor de código. Foram relatados problemas com acentuação, dificuldades relacionadas a visualização de erros e outras observações sobre a etapa de escrita do código fonte. Após um período inicial de testes foram realizadas algumas melhorias que foram consideradas positivas pelos estudantes (QUADRO1).

Quadro 1 – Comentários dos alunos no fórum

Comentários dos alunos sobre o sistema	
Primeira versão do sistema	**Segunda versão, após ajustes**
Excelente programa. O aspecto negativo é a não interpretação de caracteres especiais.	O programa melhorou bastante nessa segunda versão, na minha opinião não tem mais nada que interfere a sua utilização.
Erros de difícil entendimento ao começo do uso; uso de pontuação e acentuação estão fora de funcionamento; Servidor trava quando está sendo usado por muitas pessoas. Alguns erros são sem nexo; programa com bom intuito e ideia; prático e com um bom suporte;	Com essas atualizações ficou muito melhor de programar, principalmente dos botões, notei diferença no compilador também, espero que continue evoluindo.

(Continua...)

(Conclusão)

Comentários dos alunos sobre o sistema	
Primeira versão do sistema	**Segunda versão, após ajustes**
A parte 1 e a parte 2 dos exercícios foi bem tranquila, mas na 3, onde pede para digitarmos o código do java, estava meio complicado de entender os erros. Porém, é um programa/site bem acessível e inteligente, fácil de entender. As dicas que o programa dava não cheguei a usar muito.	Na primeira versão já estava legal o layout com os passos a serem seguidos, na segunda versão melhorou bastante a questão dos botões que agora são melhores visualizados. O compilador achei muito bom e prático na hora da realização da tarefa e nessa segunda versão ele ficou mais otimizado. No geral a ferramenta é ótima para o aprendizado de quem está começando, parabéns!
O programa se mostra bem simples e direto. Facilitaria a possibilidade dos acentos e caracteres especiais e de inserir uma entrada do usuário durante a execução, tornaria o programa mais interativo. A compilação também é um pouco demorada e difícil de interpretar os erros.	O programa ficou muito melhor, agora gostei de trabalhar com ele, acho que nos exercícios anteriores tive dificuldade de trabalhar com o modelo de entradas do programa agora utilizando variáveis e com sorteios de números aleatórios achei muito mais fácil.
O programa em si é bom, um pouco estranho no começo, mas a prática leva a perfeição. Mas é o programa é muito bom	Programa teve uma grande melhora em comparação com a versão anterior. Porém a tela de erros de execução não detecta onde o erro ocorre, e poderia pedir uma confirmação se realmente quer enviar a tarefa para evitar envíos precipitados.
Bom programa, pena que não funcione com o modelo de entradas do Java.	Achei muito interessante essa ferramenta desta vez, pois eu consegui fazer o código tudo direitinho e deu tudo certo! Porém podem continuar inovando, acho que seria muito bom.
Em primeiro momento demorei um pouco para entender as diferenças entre o sistema e o Dr Java, mas de uma forma geral achei uma plataforma muito boa para realizações das questões.	As últimas atualizações já melhoraram bastante o programa em si. Mt bom...
É um bom programa mas foi um pouco difícil para eu entender, mas eu o usaria mais vezes.	O programa está cada vez melhor.
	Na minha opinião, o programa melhorou bastante, podendo voltar nas etapas visualizando o enunciado e também, para compilar não trancou/ travou. Está muito bom!
	Melhorou muito o programa, não tive dificuldades para utilizar. Seria legal sublinhar a linha do erro. No geral achei muito bom.
	Melhorou bastante, na minha opinião, os erros são mostrados de forma mais clara.

Fonte: Elaborado pelo autor (2018).

6.1 Exercícios utilizados

Foram realizadas diversas atividades durante o semestre, disponibilizando exercícios

para que os alunos praticassem o conteúdo da disciplina, como operadores lógicos, estruturas de dados e repetição como também funções, vetores e matrizes. Com a realização das atividades o sistema foi testado e utilizando os retornos dos alunos foi possível realizar as correções e melhorias necessárias, abaixo são mostrados os exercícios realizados.

No segundo semestre de 2017 os exercícios foram disponibilizados para 27 alunos, dos quais 23 conseguiram entregar pelo menos a primeira etapa do 'Jogo da adivinhação' sendo que 15 concluíram a tarefa. No segundo exercício denominado 'Salários', 16 entregaram a primeira parte e somente 10 concluíram a tarefa como mostrado nos Quadros 2 e 3.

Quadro 2 – Exercício de um jogo da adivinhação

Título	**Jogo da adivinhação**
Problema	Você deve criar um sistema que simule um jogo no qual o usuário tenta acertar um número aleatório. O número sorteado deve estar entre 1 e n (limite determinado). Enquanto o usuário não acertar o número sorteado, o programa continua perguntando o próximo palpite e conta o número de tentativas. Quando o jogador acertar o número sorteado o programa termina e mostra o número de tentativas necessárias para acertar.
Enunciado	Elaborar um programa para gerar e armazenar um número aleatório entre 1 e n. O programa deverá perguntar para o usuário qual número foi gerado (não mostrar o número gerado para o usuário). Quando o usuário informar o número correto, o sistema não irá mais perguntar e mostrará em quantas tentativas o usuário acertou.

Fonte: Elaborado pelo autor (2018).

Quadro 3 - Exercício simulando um cálculo de salário

Título	**Salários**
Problema	O gerente do RH deseja obter algumas estatísticas sobre os salários dos funcionários. Ele necessita de uma solução que permita informar um determinado número de salários de um conjunto de funcionários. Após informar os salários o gerente deseja saber a média, o maior e o menor salário dentre os funcionários informados.
Enunciado	Criar um algoritmo que faça a leitura do valor dos salários de N funcionários, sendo que o número de funcionários é definido pelo usuário. Após ler os salários destes funcionários, o sistema deverá mostrar o valor do menor, do maior salário, além da média desses salários.

Fonte: Elaborado pelo autor (2018).

Novos testes foram realizados no primeiro semestre de 2018 com 29 alunos, nosquadros abaixo são mostrados os exercícios como também apresentado as análises dos resultados.

O primeiro exercício (QUADRO4) realizado, tem como objetivo treinar os operadores lógicos, foi solicitado a criação de um programa que recebe o valor de uma compra e sua forma de pagamento, o mesmo deve realizar o cálculo do valor final da compra utilizando a

regra informada no enunciado do exercício. A tarefa foi enviado para 29 alunos e concluído por 26, dos quais a maioria classificou como médioa dificuldade, já 5 alunos acharam fácil e 4 difícil.

Quadro4 - Exercício que calcula o valor de cada parcela de uma compra

Título	**Escolha a forma de pagamento**
Problema	Uma loja que vendia somente pelo preço a vista, com pagamento no ato da compra resolveu criar uma nova forma de pagamento, com parcelamento em até 3x. A empresa definiu uma regra que define que na compra a vista será dado 5% de desconto e nas compras parceladas será acrescido 1% a cada parcela, ou seja, em duas vezes, 2% de acréscimo em 3 vezes, 3% e assim por diante, até o número máximo de 5 parcelas. O dono da loja solicitou auxílio para obter um aplicativo que calcule e mostre o valor a pagar e o valor da parcela, de acordo com o valor original da compra e a opção de pagamento do cliente, que pode ser a vista ou parcelado (1, 2, 3 4 ou 5 parcelas).
Enunciado	Criar um programa para receber e armazenar o valor de uma compra e a opção de pagamento (a vista ou parcelado), além do número de parcelas nos casos em que o cliente paga de forma parcelada. Considerando que a vista o cliente recebe 5% de desconto e cada parcela a mais acarreta em 1% de acréscimo, calcule o valor total da compra. Aplicar o percentual correspondente ao número de parcelas sobre o valor original. A partir do valor total da compra calcule o valor das parcelas, dividindo o valor total pelo número de parcelas. Mostre o valor a pagar e o valor de cada parcela.

Fonte: Elaborado pelo autor (2018).

Para finalizar foi utilizado um exercício de vetores e matrizes (QUADRO5), solicitando a criação de um programa que realize a apuração de um jogo entre alunos. Apenas 7 alunos finalizaram esta atividade, onde 2 classificaram como médio, 5 como difícil e nenhum com fácil.

Quadro5 - Exercício que realiza a apuração de um jogo entre alunos

Título	**Competição de duplas na escola**
Problema	Uma escola resolveu fazer uma competição entre duas turmas de estudantes. As duas turmas possuem o mesmo número de alunos, 10. As regras definidas foram as seguintes: - Um nome de cada turma é sorteado e os dois estudantes competem entre si respondendo um determinado conjunto de perguntas. Ao final, o número de acertos de cada um é contado para saber o vencedor. - Alunos que já participaram não são mais sorteados e o processo se repete até que todos os alunos participaram. No final, é anunciada a turma vencedora, que é aquela que teve um número maior de alunos vencedores. Além disso, será indicado ainda quantos alunos em cada turma acertaram menos do que a média geral (considerando todos os estudantes). Fomos contratados para elaborar um programa que permita automatizar este processo. O principal requisito é que seja possível registrar o número de pontos de cada aluno, a cada rodada de competição. A partir deste registro, é possível indicar o vencedor em cada dupla e no final a turma vencedora, além do número de estudantes com pontuação abaixo da média.
Enunciado	Criar um programa no qual são utilizados dois vetores que armazenam o número de acertos que cada estudante teve em uma competição de duplas.

(Continua...)

(Conclusão)

Enunciado	A cada rodada dois alunos da turma competem e os acertos são registrados nos vetores, na mesma posição. Além disso compara-se os acertos para ver qual turma venceu. No fim, compara-se o número de vitórias das turmas para indicar qual venceu. É necessário ainda percorrer o vetor para saber quantos alunos, por turma, tem acertos abaixo da média geral, que pode ser calculada durante a leitura dos dados.

Fonte: Elaborado pelo autor (2018).

7 CONCLUSÃO

Este trabalho de conclusão de curso teve como objetivo o desenvolvimento e teste de uma ferramenta para o auxílio do ensino das disciplinas de algoritmos e programação. A proposta de desenvolver o sistema surgiu devido ao alto índice de reprovação registrado nas disciplinas de programação conforme mostrado por diversos autores presente neste trabalho.

Com a criação do sistema, foi possível auxiliar os alunos no entendimento do enunciado das atividades, utilizando a metodologia de problematização do Arco de Maguerez, a qual forçou os alunos a elaborar os pontos-chaves e a hipótese da solução antes de escrever o código fonte, fornecendo diversos recursos durante este processo, como dicas e um ambiente onde o código fonte pode ser criado e testado antes de ser entregue.

Nos primeiros testes realizados com os alunos, foram identificados alguns problemas com a ferramenta, tanto no leiaute o qual dificultava a utilização do sistema como também na parte técnica, devido a alguns erros existentes. Entre os problemas na parte técnica, podemos citar a lentidão e erros ocorridos durante o teste do código fonte, para resolver essas dificuldades foi necessário algumas configurações no servidor para limitar o tempo e os recursos consumidos pelo sistema no momento do teste.

Após a realização dos ajustes, foram realizados novos testes e obtido um novo retorno dos alunos, que relataram uma melhora na facilidade de utilização, devido às mudanças de leiaute como também na velocidade do sistema.

Conclui-se que após os testes realizados com os alunos a ferramenta e a metodologia de problematização se demonstraram eficazes, melhorando o entendimento e a compreensão dos exercícios propostos como também fornecendo um ambiente de estudo fácil, rápido e

simples.

Recomenda-se, para futuros trabalhos, que outros acadêmicos possam realizar novas pesquisas com outras metodologias de ensino, adicionando novos recursos ao sistema desenvolvido neste trabalho, visto que o trabalho confirma que os alunos tiveram uma melhora no entendimento dos exercícios propostos em aula.

REFERÊNCIAS

BERBEL, Neusi Aparecida Navas. A Metodologia da Problematização no Ensino Superior e sua contribuição para o plano da Praxis. Semina: **Ci.Soc./Hum.**, [S.l.], v. 17, p. 7-17, 1996. Ed. Especial.

COLOMBO, Andréa Aparecida; BERBEL, Neusi Aparecida Navas. A Metodologia da Problematização com o Arco de Maguerez e sua relação com os saberes de professores. Semina: **Ciências Sociais e Humanas**, Londrina, v. 28, n. 2, p. 121-146, jul./dez. 2007. Disponível em: <http://www.sgc.goias.gov.br/upload/links/arq_390_ametodologiadaproblematizacaocomoarcodemaguerez.pdf>. Acesso em: 27 out. 2016.

DANTAS, Ricardo Fidelis; COSTA, Francisco Eudes Almeida da.CODE:O ensino de linguagens de programação educativas como ferramentas de ensino/aprendizagem.In: V SIMPÓSIO HIPERTEXTO TECNOLOGIAS NA EDUCAÇÃO, 2013, [S.l.]. **Anais...** [S.l.], 2013.Disponível em: <http://www.nehte.com.br/simposio/anais/Anais-Hipertexto-2013/CODE%20-%20O%20ensino%20de%20linguagens%20de%20programa%C3%A7%C3%A3o%20educativas%20como%20ferramentas%20de%20ensino-aprendizagem.pdf>.Acesso em: 23 jun. 2017.

GIANNASI, Maria Júlia; BERBEL, Neusi Aparecida Navas.**Metodologia da problematização como alternativa para o desenvolvimento do pensamento crítico em cursos de educação continua à distância**. Londrina, 1998. Disponível em: <http://www.unibarretos.com.br/faculdade/wp-content/uploads/2015/11/METODOLOGIA-DA-PROBLEMATIZACAO-8.pdf>. Acesso em: 03 out. 2017.

GIRAFFA, Lucia Maria Martins; MARCZAKO, Sabrina S.; ALMEIDA, Gláucio.**O ensino de algoritmos e programação mediado por um ambiente na Web**. Porto Alegre, 2015. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2003/003.pdf>. Acesso em: 28 out. 2016.

GIRAFFA, Lucia Maria Martins. **Uma odisséia no ciberespaço:** o software educacional dos tutoriais aos mundos virtuais. Porto Alegre, 2009. Disponível em: <http://br-ie.org/pub/index.php/rbie/article/view/3/3>. Acesso em: 28 out. 2016.

GOMES, Anabela de Jesus; HENRIQUES, Joana; MENDES, Antonio Jose.**Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores**. Coimbra, Portugal, 2015. Disponível em: <http://eft.educom.pt/index.php/eft/article/view/23/16> Acesso em: 25 out. 2016.

GOMES, Anabela de Jesus; MENDES, Antonio Jose.**À procura de um contexto para apoiar a aprendizagem inicial de programação**. Coimbra, Portugal, 2015. Disponível em: <http://eft.educom.pt/index.php/eft/article/viewFile/439/212>.Acesso em: 25 out. 2016.

HILTZ, Starr Roxanne; TUROFF, Murra. **Education goes digital:** the revolution of online learning and the revolution in higher education. New York: [s.n.], 2005. Disponível em: <https://cyber.harvard.edu/communia2010/sites/communia2010/images/Hiltz_Turoff_2005_The_Evolution_of_Online_Learning_and_the_Revolution_in_Higher_Education.pdf> Acesso em: 17 out. 2016.

HOED, Raphael Magalhães. **Análise da evasão em cursos superiores:** o caso da evasão em cursos superiores da área de computação. 2016. 188 f. Dissertação (Mestrado Profissional em Computação Aplicada) - Universidade de Brasília. Instituto de Ciências Exatas Departamento de Ciência da Computação,Brasília, 2016. Disponível em: <http://repositorio.unb.br/bitstream/10482/22575/1/2016_RaphaelMagalh%C3%A3esHoed.pdf>. Acesso em: 21 ago. 2017.

IEPSEN, Edécio Fernando. **Ensino de algoritmos:** detecção do estado afetivo de frustração para apoio ao processo de aprendizagem. Tese (Doutorado) - PPGIE/UFRGS, Porto Alegre, 2013.
IEPSEN, Edécio Fernando; BERCHT, Magda; REATEGUI, EliseoBerni. Avaliando a Dimensão Afetiva para Apoio ao Processo de Aprendizagem na Disciplina de Algoritmos: um Estudo de Caso. RELATEC:**Revista Latinoamericana de Tecnología Educativa**, [S.l.], v. 12, n. 2, p. 55-66, 2013.

KAPP, Karl M. **The Gamification of Learning and Instruction:** Game-Based Methods and Strategies for Training and Education. 1. ed. [S.l.]: Pfeiffer & Company, 2012.

LAUBER, J. **Code School**.Middletown, 2014.Disponível em: <http://search.proquest.com/docview/1628571119/>. Acesso em: 23 jun. 2017.

NOBRE, Isaura Alcina Martins; MENEZES, Crediné Silva de. Suporte à Cooperação em um Ambiente de aprendizagem para Programação (SAmbA).In: XIIISIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇAO - SBIE, 2002, São Leopoldo. **Anais...**São Leopoldo: Unisinos, 2002.p. 337-347. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/viewFile/195/181>.Acesso em: 17 mai. 2017.

PETERSEN, Kai; FELDT, Robert; MUJTABA, Shahid; MATTSSON, Michael.**Systematic Mapping Studies in Software Engineering**.Karlskrona, 2008. Disponível em: <http://www1.bcs.org.uk/upload/pdf/ewic_ea08_paper8.pdf>.Acesso em: 18 out. 2016.

PUGA, Sandra; RISSETI, Gerson. **Lógica de programação e estrutura de dados, com aplicações em java**. 2. ed. São Paulo: Pearson Prentice Hall, 2009.

RAMOS, Vinicius et al. A Comparação da Realidade Mundial do Ensino de Programação

para Iniciantes com a Realidade Nacional: Revisão sistemática da literatura em eventos brasileiros. In: XXVI SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO-SBIE. 2015, [S.l.]. **Anais...** [S.l.], 2015. p. 318. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/download/5178/3566>. Acesso em: 21 ago. 2017.

RAMOS, Vinicius; FREITAS, Mateus; GALIMBERT, Maurício. **A Comparação da Realidade Mundial do Ensino de Programacão para Iniciantes com a Realidade Nacional:** revisão sistemática da literatura em eventos brasileiros.Florianópilos, 2015. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/5178>. Acesso em: 25 out. 2016.

SANTOS, Pricila Kohls dos; GIRAFFA, Lucia Maria Martins. **Evasão na educação superior:** um estudo sobre o censo da educação superior no Brasil. Porto Alegre, 2012. Disponível em: <http://repositorio.pucrs.br/dspace/bitstream/10923/8689/2/EVASAO_NA_EDUCACAO_SUPERIOR_UM_ESTUDO_SOBRE_O_CENSO_DA_EDUCACAO_SUPERIOR_NO_BRASIL.pdf>. Acesso em: 23 out. 2016.

SILVA, Priscylla; TENÓRIO, Maria Cristina; FECHINE, Joseana. **Um Mapeamento Sistemático sobre Iniciativas Brasileiras em Ambientes de Ensino de Programação.** Campina Grande, 2015. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/5188/3576>.Acesso em: 25 out. 2016.

SQUIRE, Kurt. **Video Games and Learning -** teaching and participatory culture in the digital age.1. ed. [S.l.]: Teachers College Press, 2011.

VIHAVAINEN, Arto; AIRAKSINEN, Jonne; WATSON, Christopher. In:Proceedings of the tenthannualconference on International computingeducationresearch. **ACM**, New York, p. 19-26, 2014.

VISSER, Lya; PLOMP, Tjeerd; AMIRAULT, Ray J.; KUIPER., Wilmad.Motivating students at a distance: The case of an international audience.**Journal of Educational Technology Researchand Development**, [S.l.], 2002. Disponível em: <http://www.learndev.org/dl/ETR%26D2002-LyaEtAl-FinalDraft.pdf>.Acesso em: 17 out. 2016.

ZICHERMANN, Gabe; CUNNINGHAM, Christopher.**Gamification by Design**.1.ed. [S.l.]: O'Reilly, 2011.

UNIVATES

R. Avelino Talini, 171 | Bairro Universitário | Lajeado | RS | Brasil
CEP 95914.014 | Cx. Postal 155 | Fone: (51) 3714.7000
www.univates.br | 0800 7 07 08 09