DESTAQUES ACADÊMICOS
UNIVATES | CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Revista Destaques Acadêmicos, Lajeado, v. 10, n. 4, 2018. ISSN 2176-3070
DOI: http://dx.doi.org/10.22410/issn.2176-3070.v10i4a2018.2026
http://www.univates.br/revistas

SISTEMA DE APOIO AO USO DA PROBLEMATIZAÇÃO NO ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO

Evandro Franzen[1], Cleverton Hemming[2], Magda Bercht[3]

Resumo: As dificuldades enfrentadas por estudantes em disciplinas de programação são notórias e estão entre as principais causas de reprovação e evasão dos cursos da área de computação. Na maioria dos cursos disciplinas de programação ocorrem já nos semestres iniciais. Constata-se que muitos estudantes têm dificuldade para compreender o problema e para especificar a solução antes da elaboração do código fonte. Diferentes métodos e tecnologias têm sido pesquisados e desenvolvidos com o intuito de apoiar o ensino e aprendizagem nestes componentes curriculares. Entre os métodos é possível citar a problematização, que estimula a análise e compreensão de um problema antes da criação da solução final. Este trabalho apresenta o desenvolvimento e aplicação de um sistema baseado na metodologia da problematização como ferramenta de apoio ao desenvolvimento de atividades em disciplinas introdutórias de programação. O artigo descreve as etapas do processo de desenvolvimento do software, as principais funcionalidades, além dos resultados da utilização deste em uma disciplina de programação, em um curso de graduação. Para avaliar a percepção dos estudantes a respeito da metodologia e dos recursos da ferramenta foi aplicado um questionário, cujos resultados demonstraram que os estudantes reconhecem a importância de compreender e analisar o problema antes da construção do código fonte. Foi possível perceber também que os estudantes relataram um elevado grau de satisfação após a utilização do software.

Palavras-chave: Aprendizagem ativa. Ensino de algoritmos. Programação de computadores. Tecnologias educacionais. Resolução de problemas.

1 Professor na Universidade do Vale do Taquari – UNIVATES. Mestre em Ciência da Computação pela UFRGS - RS, Graduado como Tecnólogo em Processamento de dados pela UNISINOS. evandrofra@gmail.com

2 Graduado em Sistemas de Informação. Universidade do Vale do Taquari – UNIVATES. clevertonh@gmail.com

3 Doutora e Mestre em Ciências da Computação pela UFRGS –RS. Graduada em Licenciatura em Matemática pela UFRGS - RS. Professora no Programa de Pós-Graduação em Informática na Educação (PPGIE) da UFRGS – RS. bercht@inf.ufrgs.br

1 INTRODUÇÃO

As disciplinas de algoritmos e programação são fundamentais nos cursos da área de computação. É possível verificar que a reprovação em disciplinas introdutórias de programação, tem sido significativa. Watson e Li (2014) revisando o trabalho desenvolvido por Bennedsen e Caspersen (2007), comprovaram que a reprovação é na maioria dos casos, superior a 30%. 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).

As dificuldades, verificadas são muitas, diversos acadêmicos iniciantes nas disciplinas de programação consideram como uma barreira o fato das disciplinas contemplarem conteúdos envolvendo interpretação de texto, raciocínio lógico e matemática. Para Nobre e Menezes (2002) as principais dificuldades enfrentadas pelos professores consistem no desenvolvimento das habilidades dos estudantes, na compreensão e resolução de problemas e no desenvolvimento da abstração.

O desenvolvimento de programas de computador envolve tarefas que exigem a capacidade de resolver problemas escrevendo uma sequência de passos. O ensino na maioria dos casos se dá pelo desenvolvimento de atividades práticas usando enunciados que apresentam problemas a serem resolvidos pelo aluno (IEPSEN, BERCHT E REATEGUI, 2013). Antes da codificação em uma linguagem de programação, o estudante deve compreender o problema e definir uma estrutura lógica da solução, além dos especificar os elementos que serão utilizados (estruturas de armazenamento, condicionais, repetição).

Ao longo dos anos diferentes estratégias e recursos têm sido utilizados com o objetivo de qualificar o ensino de programação e com isso, reduzir as dificuldades e os índices de evasão e reprovação. Metodologias de ensino ativas, como a problematização, têm sido utilizadas com o objetivo de estimular o envolvimento, colocar o discente como sujeito fundamental no processo de ensino e aprendizagem. Entre as abordagens é possível destacar investigações sobre a influência de diferentes paradigmas ou linguagens de programação (PAILLARD, MOREIRA, 2017);

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.

Este trabalho apresenta o desenvolvimento de um sistema baseado na metodologia da problematização como ferramenta de apoio ao

desenvolvimento de atividades em disciplinas introdutórias de programação. O sistema foi denominado PROALG (Problematização aplicada ao ensino de algoritmos e programação), e tem como principal objetivo ser uma tecnologia de apoio em disciplinas de algoritmos e programação. Serão descritas as etapas do desenvolvimento da ferramenta, os principais recursos e os resultados decorrentes dos testes realizados em uma disciplina de programação introdutória que compõe a matriz de diferentes cursos de graduação na área de computação.

1.1 O ensino de programação

De acordo com Farrel (2010) as tarefas do programador vão além da escrita de instruções, sendo necessário entender o problema, especificar ou planejar a organização lógica, codificar o programa, executar e testar a solução. Uma interpretação incorreta do problema ou uma organização lógica mal formulada na maioria dos casos resulta em maiores dificuldades nas etapas finais. As diretrizes nacionais para os cursos da área de computação (MEC, 2016) definem as competências e habilidades para a formação para diferentes cursos. Embora as diretrizes não façam distinção entre habilidades e competências, é possível reconhecer as seguintes habilidades associadas a programação:

- Identificar problemas que tenham solução algorítmica.
- Resolver problemas desenvolvendo algoritmos.
- Identificar e analisar requisitos e especificações para problemas.
- Aplicar temas e princípios, como a abstração e complexidade.

Para identificar um problema e resolver o mesmo utilizando algoritmos é necessário antes de tudo compreender os principais aspectos, possíveis entradas, armazenamento e resultados esperados. Após a identificação dos pontos essenciais, é possível definir uma estratégia geral, um conjunto de passos ou etapas que podem compor uma ou mais soluções possíveis. A capacidade de utilizar a abstração é frequentemente citada por autores como uma habilidade importante no aprendizado de programação. Kramer (2007) define abstração como a habilidade de extrair características comuns a partir de exemplos e de especificar ou desenhar as características gerais de uma solução. Gomes e Mendes (2007) em seus estudos indicam que a dificuldade de interpretar um problema é uma característica comum entre os estudantes, especialmente nas disciplinas introdutórias.

Diversos autores têm investigado as dificuldades na aprendizagem de algoritmos e programação, especialmente aquelas apresentadas por programadores iniciantes (GOMES E MENDES, 2007; DA CUNHA, TONETTTI E SANAVRIA, 2017; SOUZA, DA SILVA E BARBOSA, 2016). Gomes e Mendes (2007) descrevem estes problemas sob as seguintes perspectivas: Os métodos de ensino, estratégias de estudo, habilidades e atitudes demonstradas pelos alunos, natureza da programação e os aspectos psicológicos. Um mapeamento

sobre a publicação de artigos relacionados ao ensino de programação é apresentado por Souza, Da Silva e Barbosa (2016).

Ao longo dos últimos anos foram desenvolvidos diversos ambientes ou sistemas com o intuito de apoiar o processo de ensino de algoritmos e programação. Esta seção descreve algumas destas inciativas, destacando a relação das mesmas com o presente trabalho.

O portal URI Online Judge é um projeto desenvolvido pela Universidade Regional Integrada (URI), com objetivo assessorar professores em disciplinas de programação, oferecendo desafios que estimulam os estudantes a desenvolver atividades práticas de programação (BEZ, TONIN E RODEGHERI, 2014). A plataforma disponibiliza um juiz online para que sejam realizados testes nas soluções propostas, como também correção em tempo real, fórum e aceitação de diversas linguagens de programação como C, C++, Java e Python. Professores podem gerenciar diferentes turmas, listas de exercícios, além de categorizar os problemas de acordo com os tópicos trabalhados na disciplina.

O sistema se constitui em um ambiente integrado, com a escrita e visualização do código diretamente no navegador. O ambiente permite que o estudante gere a saída esperada a partir de um conjunto de entradas para um problema e acompanhe o progresso das atividades que estão sendo desenvolvidas (BEZ, TONIN E RODEGHERI, 2014).

O trabalho desenvolvido por Amaral et al. (2015) teve como principal objetivo desenvolver um protótipo de um Portal Web baseado em teorias de aprendizagem para apoiar o ensino de programação para iniciantes. As principais teorias utilizadas tanto no projeto como na ferramenta foram o Ciclo de Aprendizagem experimental de Kolb e a Aprendizagem Significativa. A ferramenta foi criada para ser um objeto de aprendizagem, a qual concentra diversas informações e recursos que auxiliam o aluno durante a aprendizagem, além de se constituir de um instrumento que pode ser usado por professores no desenvolvimento do raciocínio lógico e capacidade de abstração.

As iniciativas descritas contribuíram com a especificação e o projeto do sistema apresentado neste artigo. O sistema apóia o desenvolvimento de atividades de programação de forma online, tal como ocorre no Online Judge. Uma das principais diferenças é que o sistema PROALG se baseia em uma metodologia ativa, composta de um conjunto de etapas anteriores a escrita do código. O fato de estar associado a uma metodologia de ensino e aprendizagem é uma das semelhanças com o trabalho descrito por Amaral (2015), embora estejam focados em teorias diferentes, em ambos os casos há uma preocupação em associar a tecnologia com um processo, com uma metodologia específica, característica importante em uma tecnologia educacional (RICHEY, SILBER E ELY, 2008).

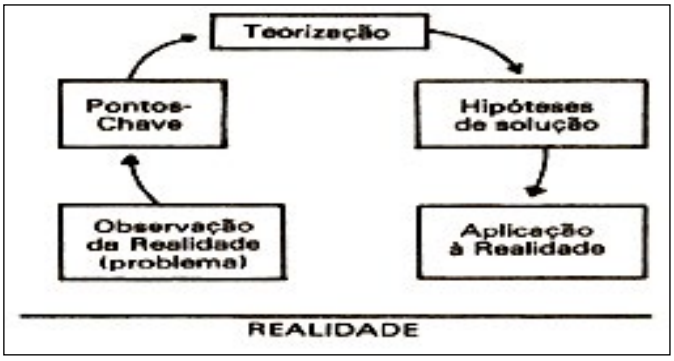
1.2. Metodologia da problematização

Juan Diáz Bordenave em Estratégias de Ensino-Aprendizagem (1982), apresenta os princípios da educação problematizadora, na qual o aluno se torna um sujeito ativo do processo, a partir do estímulo para a reflexão sobre problemas. Tais conceitos se opõe ao cenário do aluno passivo, memorizador e com baixa capacidade de resolver problemas. Na educação problematizadora, o aluno se torna um sujeito ativo do processo, a partir do estímulo para a reflexão sobre problemas.

A problematização teve como um dos primeiros métodos propostos o Arco de Maguerez, ilustrado na Figura 1. Uma das referências iniciais ao método é encontrada em Bordenave e Pereira (1982). Nela se enfatiza a ideia de que a aprendizagem deve acontecer a partir de uma visão global de um problema para uma visão analítica, que equivale a compreensão do mesmo.

A situação problema diz respeito à uma situação real ou um problema contextualizado, sendo possível abordar desde problemas mais complexos aos mais simples, de acordo com a atividade proposta. A definição dos pontos-chave do problema permite uma compreensão inicial dos principais elementos que o definem. Estabelece-se neste momento uma reflexão sobre o problema, a determinação da sua complexidade e as variáveis associadas. Busca-se identificar quais conceitos devem ser pesquisados ou revisados para a proposição de alternativas que possam solucioná-lo (BERBEL, 2012).

Figura 1 – Arco de Maguerez



Fonte: Bordenave, 1982.

A teorização é o momento no qual os estudantes pesquisam sobre os conceitos e elementos principais do problema, buscando aumentar o conhecimento sobre o mesmo. A reflexão e estudos devem fornecer subsídios para a elaboração de possíveis soluções, ou hipóteses de solução para o problema. Quais recursos serão utilizados, quais sequências de passos são adequadas para elaborar a solução, que técnicas serão usadas para elaborar,

testar e validar a solução, são exemplos de questionamentos comuns na etapa denominada "Hipóteses de solução" (BERBEL E GAMBOA, 2011).

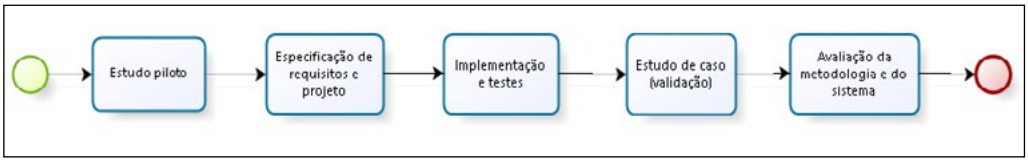
Estudos recentes têm demonstrado que as metodologias ativas contribuem para melhorar o desempenho dos estudantes em cursos ou disciplinas nas áreas de ciências exatas, tanto em cursos de graduação, quanto na educação profissional (JUNQUEIRA, WILDNER, 2017). O trabalho de Freeman et al. (2014) apresenta uma análise da performance de estudantes de graduação nos cursos de Matemática, Tecnologia e Engenharia comparando a aplicação de metodologias tradicionais com estratégias baseadas na Aprendizagem Ativa. A principal conclusão do estudo é que houve um incremento considerável na performance dos estudantes e que o uso sistemático da Aprendizagem Ativa ao longo do tempo é um aspecto importante para o sucesso desta. Dentre os métodos citados pelos autores estão as técnicas baseadas na resolução de problemas e a colaboração. Balduino e Ferreira (2015) apresentam os resultados do uso da metodologia da problematização combinada com tecnologias como SMS e modalidades de ensino semipresencial, como técnicas para apoiar o trabalho em grupos na resolução de problemas de programação.

2 METODOLOGIA

A metodologia adotada seguiu uma abordagem exploratória, baseada inicialmente em um estudo piloto que teve como principal objetivo avaliar a viabilidade de utilizar a problematização na resolução de problemas de programação. Durante o estudo piloto foi utilizado o AVEA (Ambiente Virtual de Aprendizagem) Moodle e dentro deste o recurso de Lição, que permite criar uma atividade para ser resolvida em uma sequência de passos. Os resultados do estudo piloto indicaram que os estudantes foram capazes de compreender a metodologia e que a coleta de dados realizada durante as etapas poderia contribuir para acompanhar as atividades (FRANZEN, BERCHT, DERTZBACHER, 2017).

A Figura 2 ilustra as etapas do processo, desde o estudo piloto até os testes e avaliação da ferramenta. Após o estudo inicial, foi realizada a etapa de especificação e análise dos requisitos, além do projeto que incluiu as definições sobre as tecnologias que foram utilizadas na implementação do software. A seguir foi realizado um estudo de caso, que contou com a participação de estudantes de uma disciplina de Algoritmos e programação, que é a primeira experiência de programação dos estudantes em seus cursos de graduação. O principal objetivo foi a utilização do sistema durante a resolução de diversas tarefas de programação e a avaliação do método e da ferramenta por parte dos estudantes no final do semestre.

Figura 2 – Processo de desenvolvimento do sistema PROALG



Fonte: Dos autores (2018).

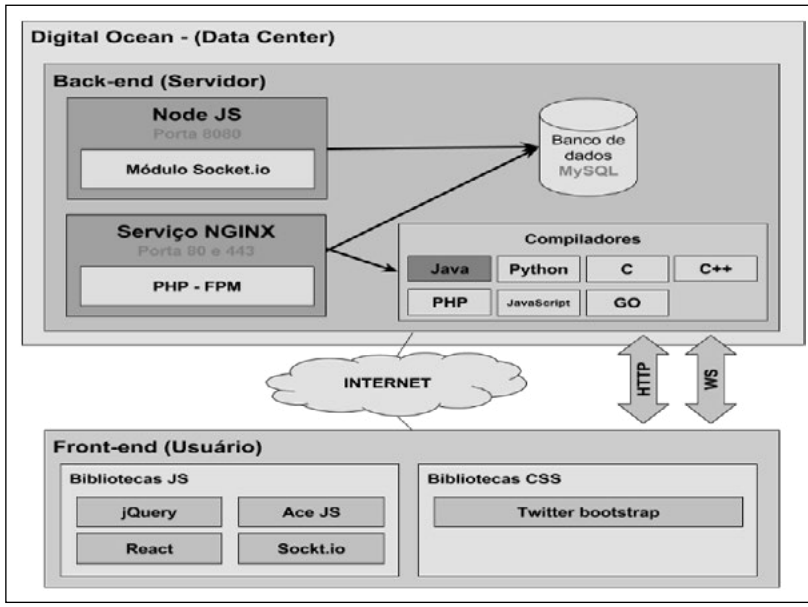
Um questionário foi o principal instrumento utilizado para conhecer a percepção dos estudantes. Além deste, foi utilizado um recurso de Fórum no ambiente virtual da disciplina para que os estudantes pudessem manifestar as observações e comentários sobre as atividades e o sistema. Quanto a abordagem utilizada para análise dos dados, foi adotado um método misto, com uma combinação de uma análise quantitativa dos resultados da aplicação do questionário, complementada por uma avaliação qualitativa das postagens no fórum (CRESWELL, 2010).

A problematização é incorporada na ferramenta com a proposição de um conjunto de passos, que iniciam por uma definição da situação problema por parte do professor, de forma textual. A identificação dos pontos-chave consiste em apontar os elementos que serão utilizados na resolução. O objetivo é que o aluno reflita sobre o problema e descreva os elementos (entradas, saídas, estruturas, construtos) necessários para elaborar a solução. A seguir deve ser elaborada uma hipótese para a solução do problema, uma sequência de passos, etapas, ações ou ainda uma especificação em alto nível da solução (algoritmo). Por fim, é elaborado o código fonte que constitui-se na solução final para a atividade. A partir da segunda etapa, o estudante pode acessar dicas ou visualizar pseudocódigo da solução, como apoio para esclarecer dúvidas.

2.1. Tecnologias utilizadas

Um dos requisitos não funcionais do sistema foi a disponibilização deste de forma online, que pudesse ser acessado pelos alunos em qualquer momento e em diferentes locais. A partir desta definição optou-se por hospedar a aplicação em um data center na nuvem, que disponibilizasse as tecnologias necessárias, mostradas na Figura 3.

Figura 3 – Tecnologias utilizadas



Fonte: Dos autores (2018).

O sistema utiliza diversas tecnologias, tanto no lado do servidor, quanto no front-end, que se constitui na interface e recursos utilizados pelo usuário. Como servidores web são utilizadas duas tecnologias:

- O NGINX (versão 7.1) é utilizado para interpretar os códigos fontes escritos na linguagem de programação PHP (Hypertext Preprocessor), que é a principal linguagem utilizada na implementação do software.
- O NodeJS é utilizado para interpretar os códigos fontes escritos na linguagem JavaScript, que é utilizada em algumas interfaces para permitir uma interação mais dinâmica, além da gravação de determinados eventos.

Diversos frameworks e bibliotecas como CCS (Cascading Style Sheets), React, Twitter Bootstrap, Socket.io e Ace.js são utilizadas na interface o usuário, para a coleta e gravação dos dados ou para a edição, compilação do código fonte de forma online. Cabe ressaltar que o software foi desenvolvido com o objetivo futuro de permitir a escrita de código fonte em diferentes linguagens, porém em um primeiro momento foi utilizado somente o compilador Java, em função desta ser a linguagem padrão para o ensino de programação na instituição em que ocorreu o estudo de caso.

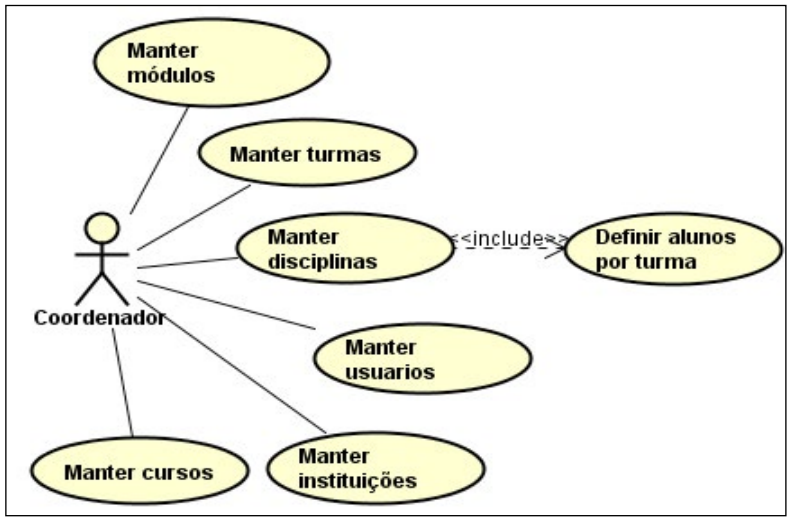
3 RESULTADOS E DISCUSSÃO

A seguir serão apresentadas as principais funcionalidades do sistema, os dados coletados durante a realização das atividades e por fim, os resultados de uma pesquisa que foi aplicada no final do estudo de caso, com o objetivo de conhecer a percepção dos estudantes sobre a utilização da problematização e sobre os recursos do sistema.

3.1 Funcionalidades e interfaces

Com o intuito de mostrar os recursos que serão disponibilizados na ferramenta, foram elaborados diagramas de casos de uso (BOOCH, RAUMBAUGH, JACOBSON, 2006).

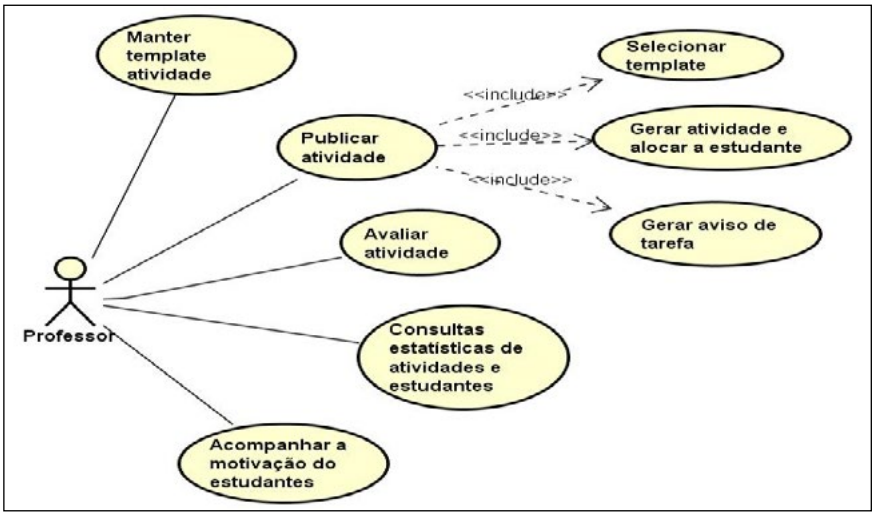
Figura 4 – Diagramas de casos de uso para coordenador



Fonte: Dos autores (2018).

A Figura 4 exibe os requisitos funcionais associados ao coordenador, que é responsável por manter os cadastros básicos, incluindo instituição, usuários, cursos e disciplinas, além das turmas que ocorrem a cada um dos semestres. Disciplinas são vinculadas às turmas, que por sua vez são compostas por um conjunto de alunos (usuários) aos quais é concedido acesso às atividades propostas pelo professor.

Figura 5 – Diagrama de casos uso relacionado ao professor

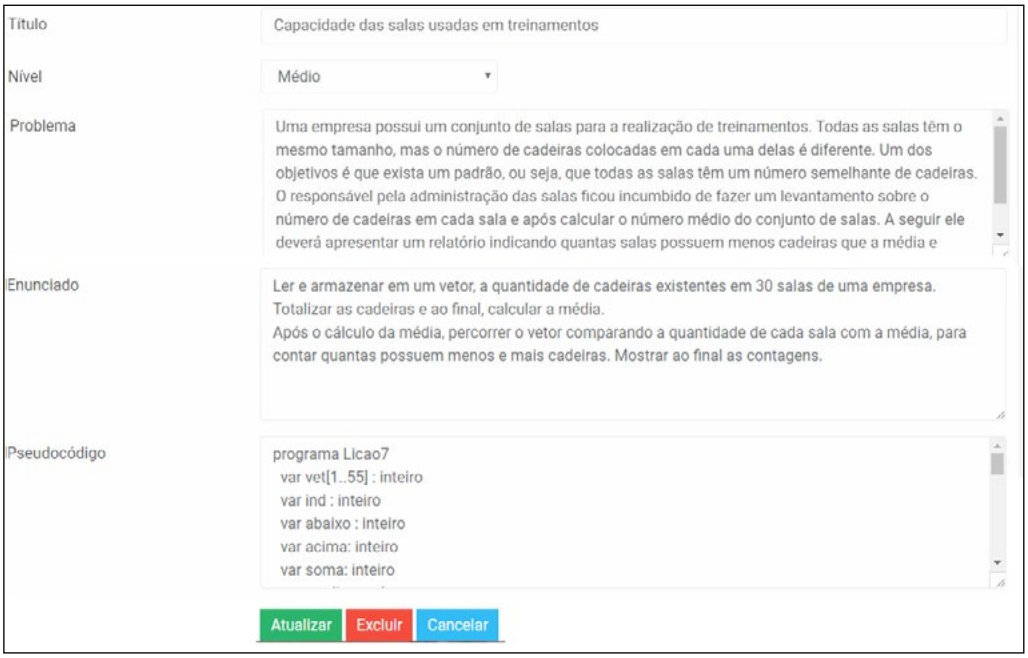


Fonte: Dos autores (2018).

Na Figura 5 é possível observar que cabe ao professor ou os diversos professores que ministram disciplinas de programação, manter listas de exercícios (templates) que poderão ser aplicados ao longo das ocorrências das disciplinas. O professor publica uma atividade que deverá ser realizada pelos alunos de uma determinada turma. A publicação envolve a seleção de um template, a geração e alocação a cada estudante, que é feita de forma automática pelo sistema, além da geração do aviso que existe uma tarefa a ser cumprida.

A interface mostrada na Figura 6 apresenta exemplos dos atributos definidos para um template e a Figura 7 mostra a interface para publicação da atividade para uma determinada turma. Um mesmo template pode ser utilizado como atividade para diferentes turmas, ao longo de vários semestres, o que facilita a manutenção de atividades. No momento da publicação, o professor determina se o método da problematização será utilizado ou se o estudante deverá somente elaborar o código fonte, utilizando um método tradicional.

Figura 6 – Cadastro de template



Fonte: Dos autores (2018)

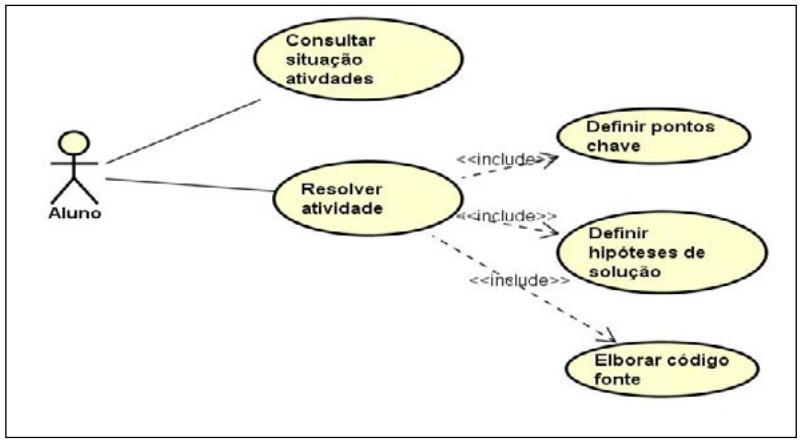
A descrição do problema deve trazer uma situação real, o contexto de um problema que estimule o estudante a pensar, refletir e pesquisar sobre os recursos a serem utilizados (BERBEL, 2012). Uma descrição que estimule o estudante a analisar o problema, sem fornecer elementos da solução, da construção do algoritmo é fundamental para a aplicação dos fundamentos da metodologia da problematização.

Figura 7 – Publicação de exercício

Fonte: Dos autores (2018).

O professor terá acesso ainda a recursos para visualização de estatísticas sobre os dados coletados durante as atividades e aos indicativos sobre a motivação demonstrada pelos estudantes durante a realização das tarefas. Tais recursos se relacionam diretamente aos objetivos deste trabalho, que busca apoiar o ensino de programação, considerando a dimensão afetiva. Os módulos de acompanhamento não foram desenvolvidos nesta versão do sistema, trabalhos futuros permitirão definir o modelo para acompanhamento da motivação e a implementação do respectivo módulo no software.

Figura 8 – Diagrama de casos de uso das funcionalidades relacionadas ao aluno



Fonte: Dos autores (2018).

As funcionalidades para o aluno em um primeiro momento permitem que ele verifique a lista das atividades e o status de cada uma delas, podendo acessar a mesma para a sua resolução, caso ela não esteja finalizada (Figura 8). A interface mostrada na Figura 9 exibe uma lista de possíveis atividades, sendo que a primeira não foi iniciada a segunda foi parcialmente resolvida.

Figura 9 – Exibição da lista de exercícios para o estudante

Fonte: Dos autores (2018)

Ao acessar uma determinada atividade, o aluno passa a visualizar cada uma dos passos que devem ser realizados para finalizar a atividade, sendo o primeiro deles a definição dos pontos-chave, cuja interface é mostrada na Figura 10. Nesta etapa o estudante não tem acesso a nenhuma dica, pois o objetivo principal é que ele concentre-se na compreensão do problema e na busca por conhecimento que lhe permita especificar possíveis soluções.

Figura 10 – Definição dos pontos-chave

Fonte: Dos autores (2018).

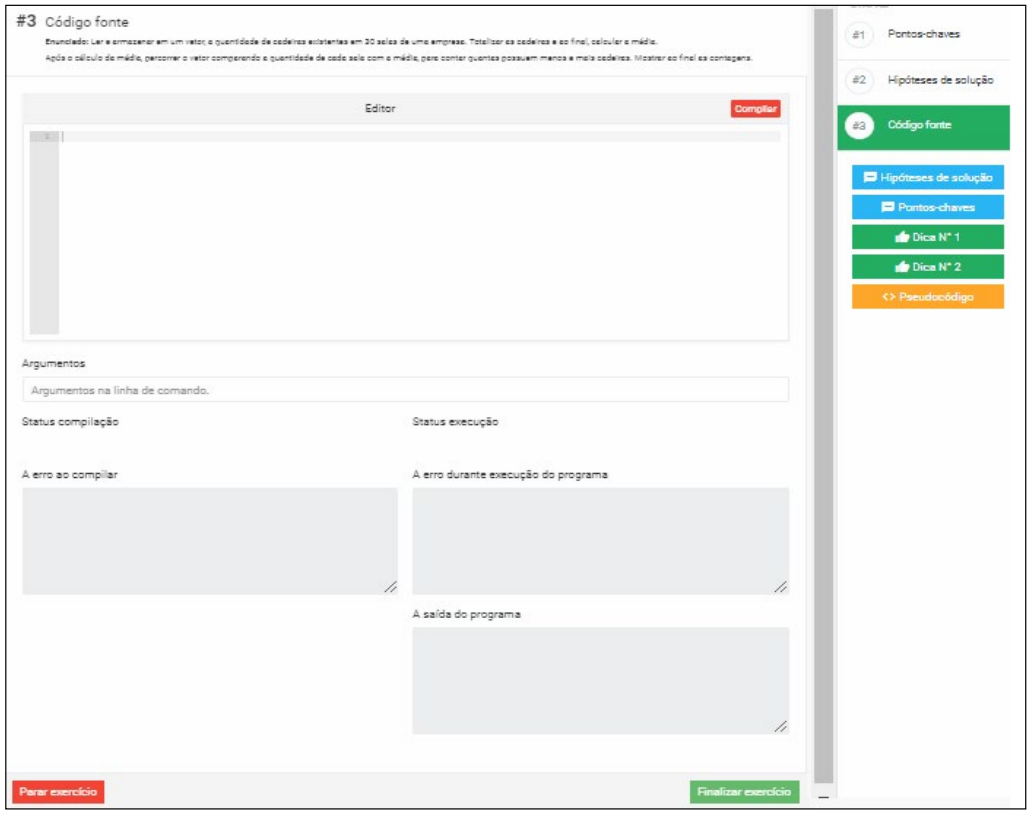
Na segunda etapa, ilustrada pela Figura 11, os botões com as dicas e pseudocódigo serão exibidos no lado direito, logo abaixo da indicação de qual etapa se encontra a atividade. A disponibilização das opções de ajuda e o registro do acesso ou não a estes recursos permitirá nos trabalhos futuros obter indicativos relacionados à independência e confiança demonstradas pelo estudante durante a resolução das tarefas.

Figura 11 – Descrição da hipótese de solução

Fonte: Dos autores (2018).

A última etapa consiste na elaboração do código fonte da solução e para esta tarefa o sistema exibe um editor que permite a escrita do código, a compilação e a execução de testes, a partir de um conjunto de parâmetros (Figura 12).

Figura 12 – Editor de código fonte



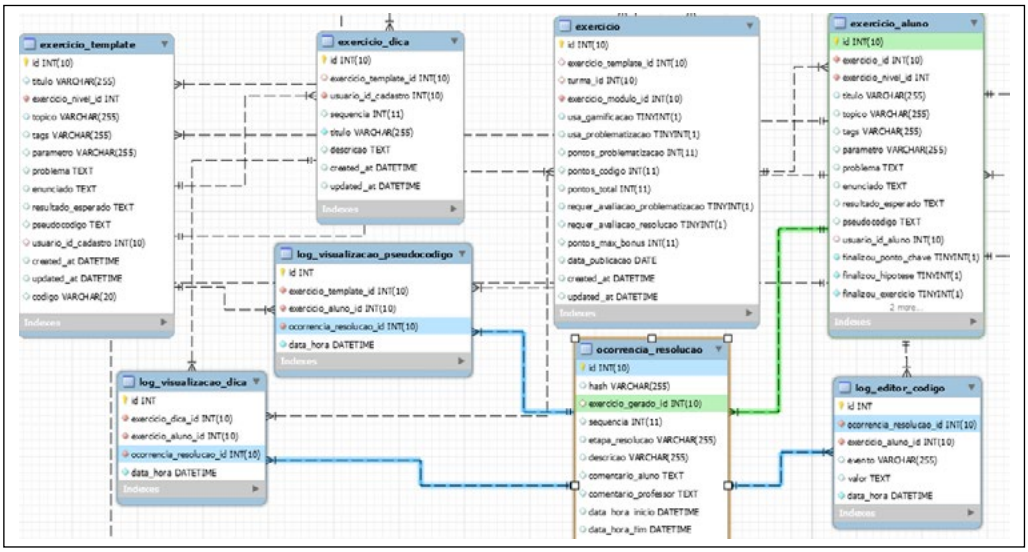
Fonte: Dos autores (2018).

3.2 Coleta de dados sobre atividades

Para atender os requisitos que permitem ao professor consultar as estatísticas relacionadas às atividades desenvolvidas e acompanhar a motivação dos estudantes, foram definidos um conjunto de dados a serem coletados durante a resolução dos problemas por parte dos estudantes. Embora estes módulos ainda não tenham sido implementados, durante o experimento que testou a ferramenta já foram coletados dados que serão usados no projeto e desenvolvimento destas funcionalidades.

Os dados incluem registros detalhados sobre data e hora de início e fim das atividades, registros de compilações e erros durante a elaboração do código, além dos conteúdos das respostas para cada uma das etapas. A Figura 13 exibe o modelo ER (Entidade Relacionamento) com as tabelas que armazenam as informações citadas.

Figura 13 – Tabelas para acompanhamento das atividades



Fonte: Dos autores (2018).

Observa-se que existem tabelas para definição de um template de exercício, com as respectivas dicas (exercicio_template e exercicio_dica) e outras para armazenar um exercício que o professor decidiu publicar para que os alunos de uma turma resolvessem (exercicio e exercicio_aluno). Há ainda um conjunto de tabelas cujo propósito é armazenar todas as ocorrências e ações do estudante, enquanto este realiza uma tarefa, sendo a principal delas denominada ocorrencia_resolucao. O conteúdo postado para cada questão, as incidências de entrada e saída da atividade, são registradas nesta tabela, juntamente com a data e a hora em que ocorreram. Foram propostas tabelas

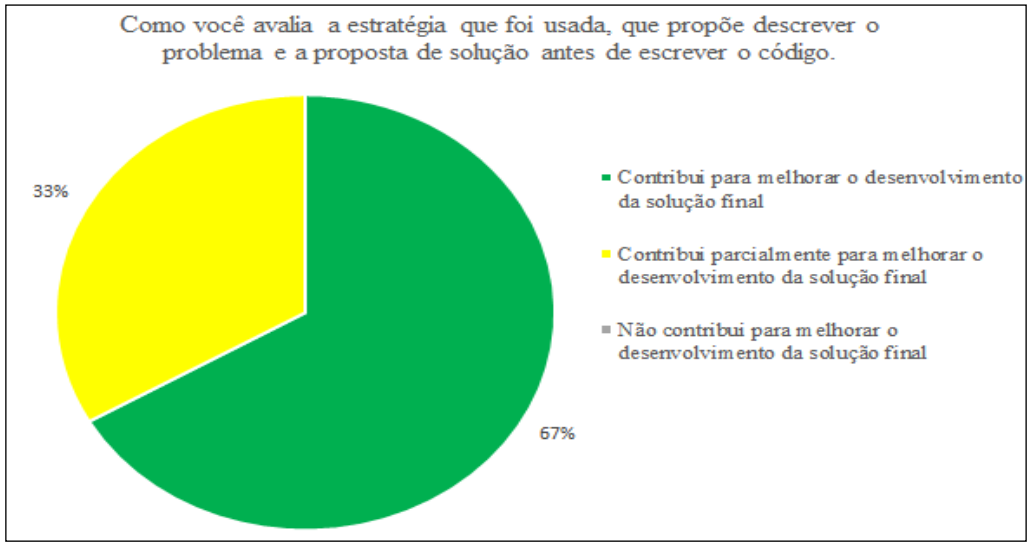
para armazenar ações específicas, como o acesso as dicas (log_visualizacao_dica, log_visualizacao_pseudocodigo), além de detalhes sobre a escrita do código fonte, como ações de copiar/colar, edição de trechos da soluções (log_editor_codigo).

Para cada registro de ocorrência de resolução, podem existir diversos registros de visualização de dica ou detalhes sobre a escrita do código, na Figura 13 é possível visualizar a relação da tabela de ocorrências com as demais. Os trabalhos para definição dos fatores da motivação do modelo para o reconhecimento destes fatores está em andamento e os dados coletados durante o estudo piloto (Franzen, Bercht, Dertzbacher, 2017) estão sendo usados para a elaboração e testes iniciais com o modelo afetivo.

3.3 Aplicação e avaliação do sistema

Esta seção descreve os resultados da aplicação do sistema em um estudo de caso realizado no segundo semestre de 2017. Participaram das atividades 28 estudantes de uma disciplina de Algoritmos e Programação, que faz parte da matriz curricular dos cursos de Engenharia de Software e Engenharia da Computação, em uma instituição de ensino comunitária, localizada no interior do Rio Grande do Sul. Os alunos eram em sua maioria de ingressantes, que estavam cursando o primeiro semestre dos referidos cursos na instituição, sendo assim, a maioria estava tendo a sua primeira experiência na área de programação.

Figura 14 – Questão relacionada à estratégia de resolução

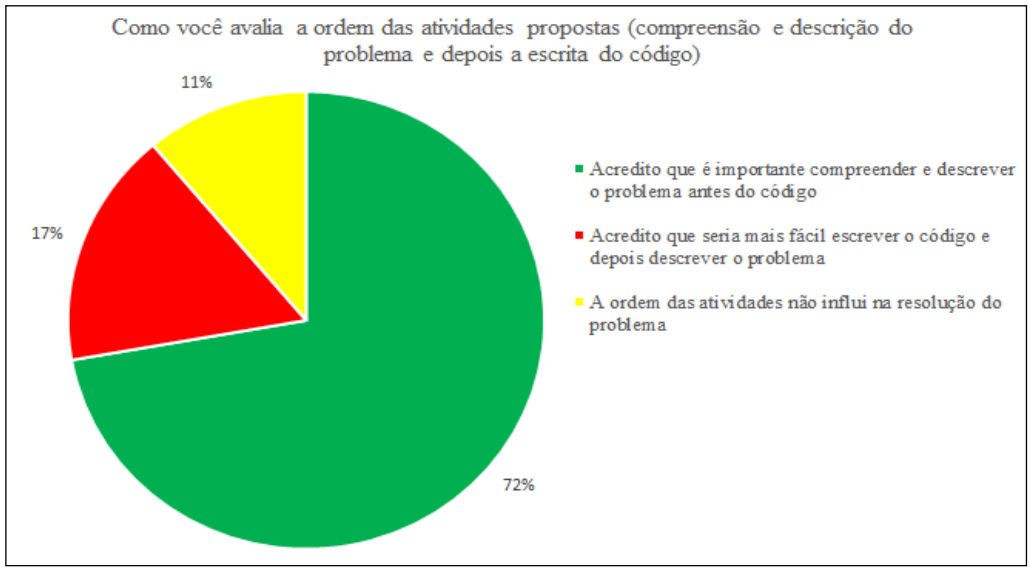


Fonte: Dos autores (2018).

Com o objetivo de avaliar a percepção dos estudantes sobre o uso da problematização e sobre o software utilizado durante a resolução das atividades, foi aplicado um questionário com diversas questões de múltipla escolha. A seguir serão apresentadas e analisadas estatísticas relacionadas a três destas questões. As duas primeiras questões buscam verificar se os acadêmicos compreenderam e consideraram positivo a forma de resolução, que incentivou a reflexão sobre o problema e a especificação de uma possível solução antes da escrita do código. Observa-se que a totalidade dos estudantes acredita que esta abordagem contribui total (67%) ou parcialmente (33%) para melhorar o desenvolvimento da solução final (Figura 14).

Quando questionados sobre a ordem dos passos, ou seja, a obrigatoriedade de definir pontos-chave e hipótese antes de escrever o código, a maioria (72%) entende que esta ordem é correta e que é importante compreender e descrever o problema primeiro. Entretanto, parte dos estudantes gostaria de elaborar primeiro o código para posteriormente descrever o problema, o que pode refletir uma visão tradicional utilizada na maioria das disciplinas de programação. Estes resultados são exibidos na Figura 15.

Figura 15 – Questão relacionada a ordem das atividades propostas

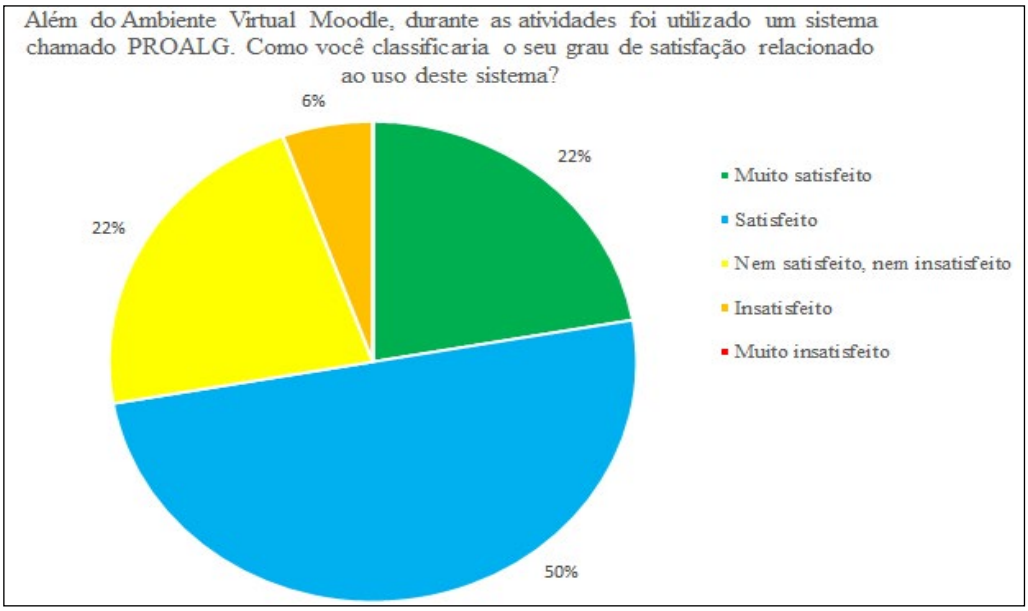


Fonte: Dos autores (2018).

A avaliação da satisfação dos estudantes sobre o sistema PROALG mostra que a grande maioria se mostra satisfeito (50%) ou muito satisfeito (22%). Apenas 6% dos alunos declarou estar insatisfeito com a ferramenta. 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 (Figura 16).

Figura 16 – Grau de satisfação relacionado ao uso do sistema



Fonte: Dos autores (2018)

Além do questionário, durante a disciplina foi disponibilizado um fórum no AVEA Moodle para que os alunos se manifestassem através de comentários e sugestões para a continuidade das atividades. Foram postados 35 comentários, estes feitos por 20 estudantes diferentes, o que permite inferir que a participação no fórum foi significativa, aproximadamente 64% dos membros da turma postaram algum comentário no fórum. Em sua maioria os comentários sobre a ferramenta foram positivos, 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.

Quadro 1 – Principais comentários postados no fórum

Comentários após testes iniciais	Comentários após ajustes no editor
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.	*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.*
Excelente programa. O aspecto negativo é a não interpretação de caracteres especiais.	*Com essas atualizações ficou muito melhor de programar, principalmente dos botões, notei diferença no compilador também, espero que continue evoluindo.*
O programa em si é bom, um pouco estranho no começo, mas a prática leva a perfeição. Mas é o programa é muito bom	*O programa melhorou bastante nessa segunda versão, na minha opinião não tem mais nada que interfere a sua utilização.*
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.	*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!*
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;	*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 envios precipitados.*

O Quadro 1 apresenta alguns destes comentários, que foram selecionados por representar a opinião da maioria dos estudantes. Embora não estejam descritos aqui todas as observações, os textos apresentados permitem estabelecer conclusões relevantes sobre a visão dos estudantes, além de complementar os resultados quantitativos apresentados anteriormente.

4 CONCLUSÕES

Após a exposição das etapas, dos recursos disponibilizados no sistema e dos resultados oriundos da aplicação do mesmo em uma disciplina de graduação, é possível concluir que a ferramenta apresenta potencial para qualificar o ensino de programação. Destaca-se a contribuição para que os

estudantes busquem compreender e especificar de forma mais detalhada o problema antes de iniciar a elaboração do código fonte, o que, de acordo com as pesquisas apresentadas foi percebido pela maioria dos alunos como algo importante e que pode contribuir para melhorar a aprendizagem.

As avaliações dos estudantes sobre o sistema, os recursos e a interface foi positiva, a grande maioria dos estudantes demonstrou algo grau de satisfação, inclusive postando comentários positivos, especialmente após os ajustes iniciais feitos no sistema. Tais resultados demonstram que o principal objetivo que era desenvolver e aplicar uma tecnologia educacional para utilização de uma metodologia ativa foi plenamente atingido, uma vez que houve uma percepção positiva tanto relacionada ao sistema em si, quanto a forma de resolução utilizada.

Cabe salientar que o estudo piloto realizado anteriormente foi fundamental para definir os requisitos e o projeto da ferramenta apresentada. Nesta etapa foram definidos principalmente os aspectos relacionados à coleta de dados e à viabilidade de usar as etapas da problematização, metodologia que serviu de base para a implementação do sistema.

Trabalhos futuros envolverão o desenvolvimento dos módulos de acompanhamento, das estatísticas e principalmente do modelo de reconhecimento dos fatores da motivação demonstrados pelo estudante. Este modelo utilizará os dados coletados durante a resolução das atividades e será implementado como um dos módulos do sistema, com o objetivo de apoiar as ações dos professores.

REFERÊNCIAS

AMARAL, E. et al. (2015). **Proposta de um Portal Web alinhado a Teorias de Aprendizagem para o Apoio ao Ensino de Programadores Iniciantes**. RENOTE: Revista Novas Tecnologias, v. 13, n. 1.

BALDUINO, J. D. O.; FERREIRA, F. S. S. (2015). **Proposta de uma nova abordagem para desenvolvimento de algoritmos de programação.** Linkscienceplace-Interdisciplinary Scientific Journal, v. 2, n. 1.

BENNEDSEN, J.; CASPERSEN, M. E. (2007). **Failure rates in introductory programming**. ACM SIGCSE Bulletin, v. 39, n. 2, p. 32-36.

BERBEL, N. A. N. (2012). **As metodologias ativas e a promoção da autonomia de estudantes**. Semina: Ciências Sociais e Humanas, v.32, n.1, p.25-40.

BERBEL, N. A. N.; GAMBOA, S. A. S. (2011). **A metodologia da problematização com o Arco de Maguerez: uma perspectiva teórica e epistemológica. Filosofia e Educação**, v. 3, n. 2, p. 264-287.

BEZ, J. L.; TONIN, N. A.; RODEGHERI, P. R. (2014). URI Online Judge Academic: **A tool for algorithms and programming classes**. In: Computer Science & Education (ICCSE), 2014 9th International Conference on. IEEE. p. 149-152.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. (2006**). UML: guia do usuário**. Elsevier Brasil.

BORDENAVE, J.D.; PEREIRA, A. M. (1982). **Estratégias de ensino aprendizagem**. 4. ed. Petrópolis: Vozes.

CRESWELL, J. W. (2010**). Projeto de pesquisa: métodos qualitativo, quantitativo e misto**. 3. ed. Porto Alegre: Artmed.

DA CUNHA, L. S.; TONETTI, P.; SANAVRIA, C. Z. (2017). **O Ensino de Informática no Brasil: Uma Análise da Produção Científica em Eventos da SBC (2010–2014)**. Anais do Computer on the Beach, p. 031-040.

FARREL, J. (2010). **Lógica e design de programação: introdução**. Cengage Learning.

FRANZEN, E.; BERCHT, M.; DERTZBACHER, J. **Problematização aplicada ao ensino e aprendizagem de algoritmos: Uma análise dos fatores associados a motivação dos estudantes.** RENOTE, v. 15, n. 1, 2017.

FREEMAN, S. et al. (2014). **Active learning increases student performance in science, engineering, and mathematics**. Proceedings of the National Academy of Sciences, v. 111, n. 23, p. 8410-8415.

GOMES, A.; MENDES, A. J (2007). **Learning to program-difficulties and solutions.** In: International Conference on Engineering Education–ICEE.

IEPSEN, E. F.; BERCHT, M.; REATEGUI, E. B. (2013). **Avaliando a Dimensão Afetiva para Apoio ao Processo de Aprendizagem na Disciplina de Algoritmos: um Estudo de Caso**. RELATEC: Revista Latinoamericana de Tecnología Educativa, v. 12, n. 2, p. 55-66.

JUNQUEIRA, A. M.; WILDNER, M. C. S. **Metodologia ativa aprendizagem por meio de problematização na educação profissional**. Revista **Destaques Acadêmicos**, v. 9, n. 4, 2017.

KRAMER, J. (2007). **Is abstraction the key to computing?.** Communications of the ACM, v. 50, n. 4, p. 36-42.

MEC: Ministério da Educação. (2016). **Diretrizes Curriculares Nacionais para os cursos de graduação na área da Computação**. Brasília.

NOBRE, I. A. M.; MENEZES, C. S. (2002). **Suporte à Cooperação em um Ambiente de aprendizagem para Programação (SAmbA).** In: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE). p. 337-347.

PAILLARD, G. A. L.; MOREIRA, L. O. (2017). **O impacto dos paradigmas e linguagens de programação no ensino intermediário da programação de computadores**. Revista Tecnologias na Educação. Ano 9. Vol.19.

RAMOS, V. et al. (2015). **A Comparação da Realidade Mundial do Ensino de Programação para Iniciantes com a Realidade Nacional: Revisão sistemática da literatura em eventos brasileiros.** In: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE), 2015. p. 318.

RICHEY, R. C.; SILBER, K. H.; Ely, D. P. (2008). **Reflections on the 2008 AECT Definitions of the Field.** TechTrends, v. 52, n. 1, p. 24-25.

SOUZA, D. M.; DA SILVA B. M. H.; BARBOSA, E. F. (2016). **Problemas e Dificuldades no Ensino e na Aprendizagem de Programação: Um Mapeamento Sistemático**. Revista Brasileira de Informática na Educação, v. 24, n. 1.

WATSON, C.; Li, F. W. B. (2014). **Failure rates in introductory programming revisited**. In: Proceedings of the 2014 conference on Innovation & technology in computer science education. ACM, p. 39-44.