

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO CORREIA GONZALEZ

**UM ESTUDO SOBRE O USO DO MECANISMO DE
DICAS PERSONALIZADAS NO ENSINO DE
CONCEITOS BÁSICOS DE PROGRAMAÇÃO**

MONOGRAFIA

CAMPO MOURÃO

2016

GUSTAVO CORREIA GONZALEZ

**UM ESTUDO SOBRE O USO DO MECANISMO DE
DICAS PERSONALIZADAS NO ENSINO DE
CONCEITOS BÁSICOS DE PROGRAMAÇÃO**

Proposta de Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 1, do Curso de Bacharelado em Ciência da Computação do Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Igor Scaliante Wiese

Coorientador: Prof. Dr. Marco Aurélio Graciotto Silva

CAMPO MOURÃO

2016

Resumo

Gonzalez, Gustavo. Um estudo sobre o uso do mecanismo de dicas personalizadas no ensino de conceitos básicos de programação. 2016. 27. f. Monografia (Curso de Bacharelado em Ciência da Computação), Universidade Tecnológica Federal do Paraná. Campo Mourão, 2016.

O elevado nível de reprovação em disciplinas onde são ensinados conceitos básicos de programação, em qualquer grau de ensino, é um problema enfrentado por muitos alunos e tem sido alvo de variadas pesquisas (Holcomb; Simone, 2016). Existe um conjunto de razões que estão relacionadas com a origem do problema, como o método de ensino e aprendizagem, a falta de algumas competências e interesse por parte dos alunos, e a própria dificuldade do tema (Sinclair *et al.*, 2015).

Contexto: O desenvolvimento do estudo será realizado na Universidade Tecnológica Federal do Paraná, com alunos selecionados entre o primeiro ao oitavo semestre.

Objetivo: Avaliar o impacto do uso do mecanismo de dicas personalizadas no ensino de conceitos básicos de programação, mais especificamente em estrutura de condição e laço de repetição. Para atingir esse objetivo, foram criadas duas questões de pesquisas descritas como:

- **QP₁:** *Existe impacto do uso do mecanismo de dicas para minimizar erros e melhorar soluções de exercícios?*

Esta questão de pesquisa tem como objetivo investigar se o mecanismo de dicas irá ajudar os alunos a obterem melhores resultados na execução de exercícios de programação.

- **QP₂:** *Quais dicas personalizadas ajudam mais os alunos?*

Esta questão de pesquisa avaliará qual tipo de dica é melhor, podendo ser aleatorizadas em função do tipo do exercício, geradas a partir de erros cometidos em exercícios passados e sem dicas. Levando em consideração a fonte da dica gerada, podendo ser por alunos iniciantes, alunos experientes ou professores.

Método: Será desenvolvido um software para auxiliar o aprendizado de programação que utiliza um sistema de dicas personalizadas aos alunos no momento da realização de exercícios.

Para avaliar o mecanismo de dicas será realizado um estudo controlado com três grupos de alunos distribuídos de forma homogênea em relação ao nível de conhecimento em programação. O primeiro grupo receberá dicas classificadas a partir de erros anteriores, o segundo será com dicas aleatorizadas levando em consideração o assunto abordado no exercício e o terceiro grupo não utilizará o mecanismo de dicas. Assim, os voluntários realizaram uma lista de exercícios e em cada submissão ao sistema de dicas será gravado um *log* com dados que irão mensurar o desempenho de cada voluntário, assim será realizado o estudo desses dados para responder as questões de pesquisa.

Resultados esperados: Espera-se que o software ajude no aprendizado dos alunos tornando o ensino de programação mais dinâmico e personalizado. Deste modo, espera-se diminuir o número de reprovações nas matérias introdutórias de programação e em desistências do curso de Ciência da Computação.

Palavras-chaves: Sistema, Mecanismo, Dicas, Dicas Personalizadas, Learnersourcing, Crowdsourcing

Lista de figuras

3.1	Visão geral do método proposto.	12
3.2	Visão geral do sistema de dicas.	13
3.3	Fluxos para criação de dicas.	20
3.4	Visão geral da aplicação do estudo.	21
4.1	Tela de registro de usuário.	22
4.2	Tela de login no sistema.	23
4.3	Tela para resetar a senha.	23
4.4	Tela de cadastro de professor.	24
4.5	Tela de cadastro de aluno.	24

Lista de tabelas

3.1	Formulário para aquisição de voluntários.	17
3.2	Questionário do estudo presencial.	18
5.1	Cronograma	25

Sumário

1	Introdução	7
2	Referencial Teórico	9
2.1	Reprovações em Disciplinas de Programação	9
2.2	Novas Técnicas de Ensino	10
2.3	Mecanismo de Dicas	10
2.4	Considerações Finais	11
3	Proposta	12
3.1	Implementação do Mecanismo de Dicas	12
3.1.1	Cadastro do Usuário	12
3.1.2	Cadastro de Exercícios	13
3.1.3	Cadastro de Turma	14
3.1.4	Realização de Exercícios	14
3.1.4.1	Utilização das Dicas	15
3.1.5	Cadastro de Dica	15
3.1.6	Cadastro do Diário	15
3.1.7	Boca (BOCA Online Contest Administrator)	15
3.2	Formato do Estudo	16
3.2.1	Teste do sistema	16
3.2.2	Banco de Exercícios	16
3.2.3	Banco de Dicas	16
3.2.4	Avaliação do Sistema de Dicas	17

4	Resultados Preliminares	22
5	Cronograma	25
	Referências	26

Introdução

O ensino de linguagens de programação tem o propósito de conseguir desenvolver nos alunos um conjunto de competências necessárias para conceber programas e sistemas computacionais capazes de resolver problemas reais. O insucesso na aprovação dos estudantes em disciplinas de programação, é um tema que tem sido alvo de alguns estudos (Bosse; Gerosa, 2015), (Cukierman, 2015).

O problema do insucesso é evidenciado por Lahtinen *et al.* (2005) realizaram uma pesquisa com 559 alunos e 34 professores de diferentes universidades para estudar as dificuldades na aprendizagem de programação. Como resultado foi percebido que as questões mais difíceis na programação são: a compreensão de como projetar um programa para resolver uma tarefa determinada, dividir as funcionalidades em procedimentos e encontrar erros de seus próprios programas. Estas são as capacidades que o aluno deve obter para entender as maiores entidades do programa em vez de apenas alguns detalhes sobre eles. Já os conceitos de programação mais difíceis foram recursão, ponteiros e referências, tipos de dados abstratos, manipulação de erro.

Com a necessidade de prover um suporte personalizado aos alunos, Elkherj e Freund (2014) criaram um sistema de dicas que permite o professore enviar uma dica personalizada em tempo real após verificar que o aluno realizou várias tentativas para resolver o exercício. Entretanto, com o aumento do número de alunos utilizando o sistema, os professores não conseguem oferecer um suporte a todos os alunos, assim utilização desse tipo de abordagem na construção do sistema de dicas apresenta limitações em relação a quantidade de alunos realizando os exercícios.

Nós utilizamos a abordagem do *learnersourcing* com o intuito de que os alunos, através de sua própria experiência resolvendo exercícios, podem criar dicas úteis através de suas

implementações ou *bugs* que resolvem. Estes alunos podem então gerar sugestões para colegas com base em sua própria perícia. Mostraremos que, dada essa escolha de *design*, os alunos podem criar dicas que podem até mesmo substituir a assistência personalizada dos professores, quando essa assistência não está disponível.

O objetivo desse trabalho é criar um software de código aberto para auxiliar na aprendizagem de conceitos básicos de programação, implementando um sistema colaborativo de dicas escritas pelos próprios usuários. Para investigar se o uso do mecanismo de dicas personalizadas é capaz de melhorar o rendimento dos alunos nos exercícios de estrutura de condição e laço de repetição, nós iremos aplicar um experimento controlado com três grupos distintos de alunos. O primeiro grupo utilizará o mecanismo de dicas normal que consiste em prover dicas de acordo com o exercício que o aluno está realizando. O segundo grupo utilizará o mecanismo de dicas personalizado que disponibiliza dicas de acordo com o exercício e o erro cometido na submissão. Por fim, o terceiro grupo utilizará o sistema sem o mecanismo de dicas.

No software serão implementadas funcionalidades que permitam o usuário resolver exercícios nas linguagens de programações C, C++ e Java, fornece dicas para a solução de exercícios, um diário para relatar as dificuldades enfrentadas durante a execução dos exercícios. Estas informações são úteis e podem ser personalizar e melhorar o ensino de programação.

Para realizar a validação da ferramenta será realizado um estudo controlado na Universidade Tecnológica Federal do Paraná com três grupos de estudantes escolhidos aleatoriamente de forma homogênea em relação ao nível de conhecimento em programação. Com o experimento será possível avaliar se o grupo de alunos com dicas tem melhor desempenho que alunos sem o suporte do mecanismo. O desempenho será medido por: tempo, número de tentativas até a solução correta, qualidade do código gerado medida complexidade ciclomática do código, número de linhas entre as tentativas e tamanho da solução. Por fim, será avaliado se a qualidade das dicas está diretamente relacionada com o nível de conhecimento do aluno, e se as dicas de alunos são melhores ou não em relação as dicas dos professores.

Os próximos capítulos estão organizados em três partes. No capítulo de referencial teórico será apresentado e discutido os principais conceitos que envolvem o estudo. Este procedimento é importante, pois discutiremos os pontos de vista de diversos autores, assim como diversas abordagens alternativas. O objetivo do capítulo de proposta é apresentar a metodologia que será utilizada para desenvolver o estudo e o software. O objetivo do capítulo de resultados preliminares é apresentar os avanços que o estudo obteve durante o seu desenvolvimento.

Referencial Teórico

Neste capítulo apresentaremos uma revisão da literatura agrupando trabalhos relacionados a reprovações e dificuldades de alunos em disciplinas de programação, novas abordagens de ensino e sistemas de dicas.

Reprovações em Disciplinas de Programação

O entendimento e aplicação dos conceitos estudados em disciplinas de programação é fundamental para que o aluno consiga desenvolver programas mais complexo. Helminen e Malmi (2010) afirmam que a programação é uma competência essencial no curso de Ciência da Computação. Segundo Bosse e Gerosa (2015) os estudantes normalmente apresentam uma grande dificuldade com os conteúdos abordados, ocasionando em reprovação ou desistência.

Sinclair *et al.* (2015) agruparam pesquisas internacionais como a *National Survey of Student Engagement* (NSSE) feita na América do Norte e Canadá, a pesquisa *Nacional Universidade Experience* (UES) e a pesquisa *Survey Student Participation* (SES) que realizam a medida dos esforços dos alunos em atividades associadas com a aprendizagem efetiva. Esse estudo reúne dados relacionadas à Ciência da Computação sendo que os resultados desta meta-análise indicam que o curso de Ciência da Computação apresenta taxas mais baixas do que a média em muitos dos principais pontos de referência de engajamento.

Novas Técnicas de Ensino

Mecanismo de Dicas

Nas abordagens citadas a baixo, uma dica equivale a um auxílio ao aluno para que ele consiga realizar um exercício que esteja com dificuldade, essa dica pode ser oferecida tanto pelo professor ou colega de turma. Um sistema que utiliza esse conceito pode ser chamado de sistema de dicas, onde reproduz esse contato que o aluno teria com uma pessoa mais instruída quando precisa tirar uma dúvida ou pedir uma dica para realizar o exercício. Mas o aluno com dificuldades não pode ter esse apoio todo o tempo, então estão sendo criados sistema de dicas para suprir essa necessidade de auxiliar o aluno no seu aprendizado.

Elkherj e Freund (2014) apontam que as sugestões utilizadas em sistemas de aprendizagem on-line para ajudar os alunos quando eles estão tendo dificuldades são fixados antes do tempo e não dependem das tentativas mal sucedidas que o aluno já fez. Isto limita severamente a eficácia das sugestões. Eles desenvolveram um sistema alternativo para dar dicas aos estudantes. A principal diferença é que o sistema permite um instrutor enviar uma dica para um estudante após o aluno ter feito várias tentativas para resolver o problema e falhou. Depois de analisar os erros do aluno, o instrutor é mais capaz de entender o problema no pensamento do aluno e enviar-lhes uma dica mais útil. O sistema foi implantado em um curso de probabilidade e estatística com 176 alunos, obtendo *feedback* dos alunos muito positivo. Mas o desafio que os autores enfrentam é como escalar efetivamente o sistema de forma que todos os alunos que precisam de ajuda obtenham dicas eficazes. Pois com uma grande quantidade de alunos ficaria exaustivo para os instrutores avaliarem cada submissão dos exercícios de cada aluno para retornarem uma dica específica do erro cometido. Então eles criaram um banco de dados de dica que permite que os instrutores reutilizem, compartilhem e melhorem em cima de sugestões escritas anteriormente.

Price (2015) está utilizando um sistema de tutores inteligentes (STI) que podem manter os alunos no caminho certo, na ausência de instrutores, fornecendo sugestões e advertências para os alunos que precisam de ajuda. Além disso, as técnicas baseadas em dados podem gerar feedback automaticamente a partir de tentativas de resolução de um problema. O autor está aplicando o seu estudo permitindo que os alunos escrevam códigos que se conectam com seus interesses, tais como jogos, aplicações e histórias. Como exemplo concreto, imagine um aluno que esteja desenvolvendo um jogo simple, quer requer a utilização de variáveis, laços de repetição, condicionais. Mas o aluno está com dificuldades em relação a uma funcionalidade que o jogo terá, ele poderá pedir uma dica para implementar essa funcionalidade que será gerada automaticamente pelo sistema com relação as tentativas anteriores de outros alunos.

Cummins *et al.* (2016) investigaram o uso de dicas para 4.652 usuários qualificados em um ambiente de aprendizagem on-line de grande escala chamado Isaac, que permite aos usuários para responder a perguntas de física com até cinco dicas. Foi investigado o comportamento do usuário ao usar dicas, engajamento dos usuários com desvanecimento (o processo de tornar-se gradualmente menos dependentes das dicas fornecidas), e estratégias de dicas incluindo decomposição, correção, verificação ou comparação. Como resultados obtidos, os alunos apresentaram estratégias para as resoluções dos exercícios sendo a mais comum é ver o conceito da dica para realizar a decomposição do problema e, em seguida, enviar uma resposta correta. A outra estratégia é usar os conceitos da dica para determinar se a pergunta pode ser respondida, uma grande proporção dos usuários que utilizaram essa estratégia acabou não tentando responder a pergunta

Considerações Finais

Proposta

Neste capítulo apresenta o método utilizado para a elaboração deste trabalho. A Figura 3.1 ilustra cada etapa a ser realizada para o desenvolvimento da abordagem proposta.

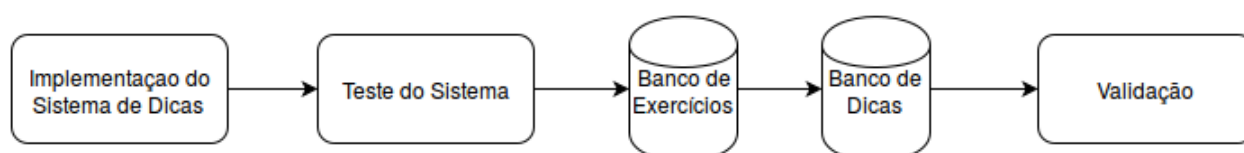


Figura 3.1. Visão geral do método proposto.

Para avaliar o impacto do uso do mecanismo de dicas personalizadas no ensino de conceitos básicos de programação, primeiramente será realizado a implementação do sistema de dicas utilizando o *framework* de desenvolvimento Laravel na versão 5.1.

Implementação do Mecanismo de Dicas

Esta seção, descreve o sistema de dicas a ser desenvolvido. O processo de realização de exercícios, que é onde o sistema deve atuar, pode ser dividido em quatro fases: cadastro do usuário, realização de exercícios, utilização de dicas (providas por usuários do sistema) e validação do exercício. A descrição apresentada será usada para a modelagem do sistema.

Cadastro do Usuário

Quando um usuário deseja realizar o cadastro no nosso sistema, ele poderá escolher entre a opção de aluno ou professor. O aluno, realiza exercícios disponíveis no banco de dados do sistema, preenche um diário não obrigatório para cada exercício, cria uma dica para cada

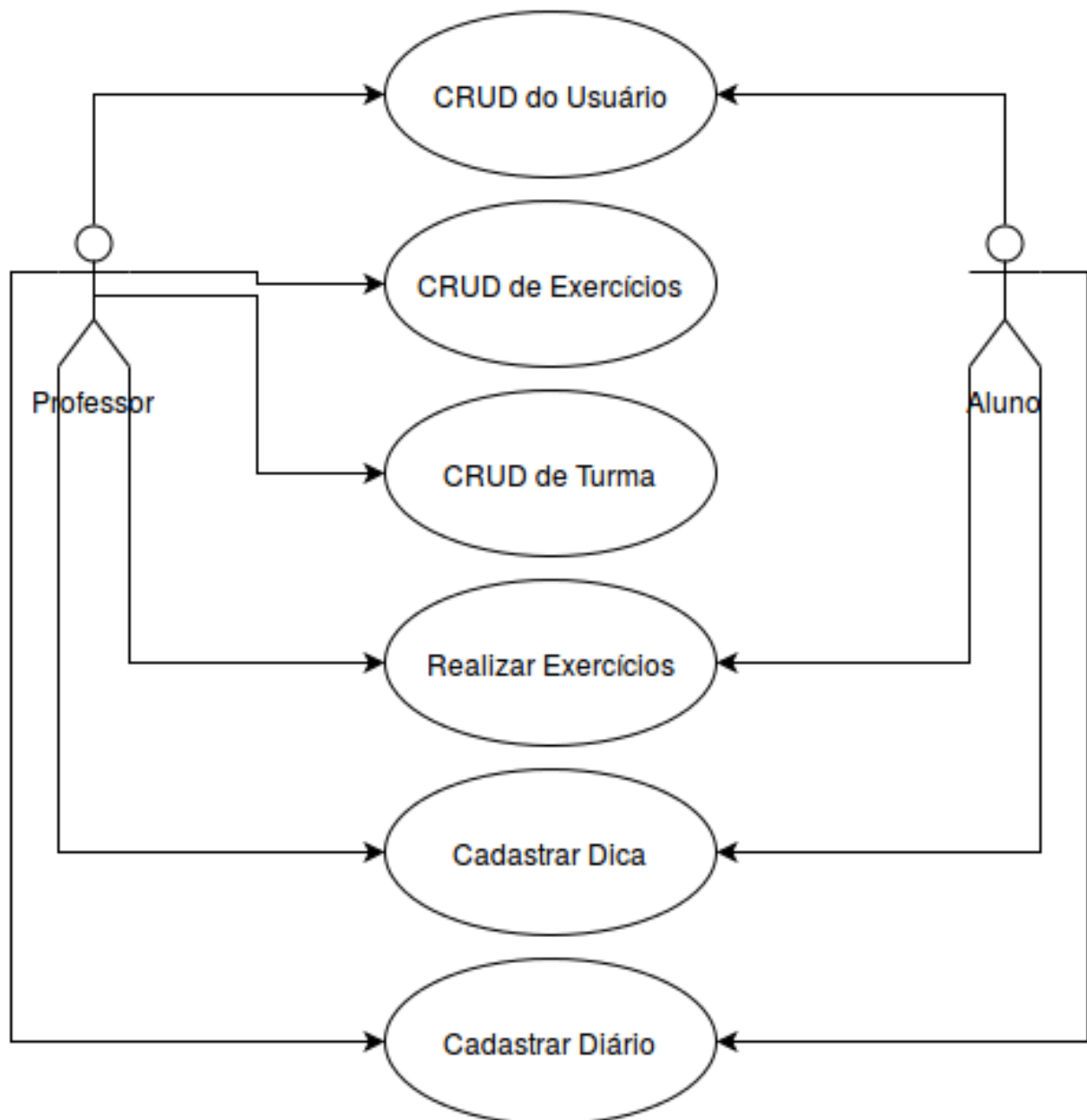


Figura 3.2. Visão geral do sistema de dicas.

exercício e consulta seu perfil com seus dados pessoais e um relatório de suas submissões de exercícios. O professor, além de realizar as mesmas atividades do aluno, também poderá criar salas com exercícios pré-definidos convidando alunos para fazer parte dela e submeter exercícios para o sistema.

Cadastro de Exercícios

Essa funcionalidade do sistema só poderá ser acessada se o usuário for um professor. Sendo assim, o professor terá que cadastrar um enunciado, qual linguagem é para ser utilizada na resolução, o nível (fácil, médio ou difícil), o tipo do exercício (condicional ou laço de

repetição), a resposta que poderá ser escrita em um campo ou um *upload* do arquivo com as respostas. Cada exercício cadastrado será agregado a uma lista de exercícios que o professor irá criar antes de cadastrar os exercícios.

Cadastro de Turma

O professor poderá realizar o cadastro de uma classe para adicionar seus alunos e passar as listas de exercícios e acompanhar o rendimento de cada aluno. Para cadastrar uma turma é necessário um nome da turma e um professor, com isso o professor poderá adicionar os alunos desejados.

Realização de Exercícios

Após a conclusão do cadastro de usuário, o aluno ou professor terão acesso as funcionalidades específicas de cada tipo de usuário. Dessa forma, o usuário poderá resolver uma lista de exercícios de duas formas: estando vinculado a uma turma ou individualmente. Caso o usuário esteja vinculado a uma turma, a lista de exercícios que ele poderá resolver será a que o professor responsável pela turma criou. Caso o usuário não faça parte de uma turma, ele terá que pesquisar por uma lista de exercícios informando qual assunto, estrutura de condição ou laço de repetição, ele deseja realizar.

Depois que o usuário encontrar uma lista de exercícios, o sistema apresentará os exercícios à serem resolvidos na ordem em que foram adicionados na lista. Assim o usuário poderá trabalhar em sua solução e submeter ao sistema para correção utilizando a linguagem de programação escolhida pelo professor na criação da lista de exercícios, essa resolução poderá estar correta ou não. Se estiver correta, o sistema apresentará uma mensagem de parabenização e redirecionará o usuário para o próximo exercício. Caso a resolução estiver errada o sistema retornará qual o erro ocorrido na compilação, assim o usuário poderá realizar outras submissões até que consiga obter sucesso na resolução ou requisitar uma dica para o sistema para auxiliá-lo na construção da resposta.

Cada submissão, tanto correta ou incorreta, é gravada no banco de dados na forma de *log*, sendo salvo os dados: a usuário que realizou o exercício, todas as submissões, os erros cometidos, o tempo demorado para obter sucesso no exercício, (ver a onde foi marcado no papel).

Utilização das Dicas

O usuário apenas poderá consultar uma dica caso o exercício enviado para validação não esteja correto e ele escolha a opção de disponibilização de dicas, assim ele terá direito a três dicas selecionadas aleatoriamente pelo mecanismo. O usuário irá escolher consultar uma das três dicas e após a consulta ele poderá realizar uma nova submissão do exercício.

Cadastro de Dica

O cadastro de uma dica pode ser realizado quando o usuário submete um exercício para validação e obtém êxito, assim o sistema apresentará uma tela perguntando se o usuário deseja contribuir com uma dica, caso aceite, o sistema irá redirecionar o usuário para a tela onde será realizado o cadastro da dica, onde ele irá descrever em um campo a dica que deseja compartilhar. Também será apresentada a solução do mesmo exercício de outro usuário para que seja escrito uma dica.

Cadastro do Diário

No processo de execução de um exercício o usuário poderá preencher um diário relatando suas experiências, essa funcionalidade será apresentada aos usuários caso o professor deseja receber o preenchimento do diário dos exercícios da lista.

Caso o professor pretende receber os diários dos exercícios da lista de sua turma, o usuário após cada submissão da resolução no sistema produzirá um diário que será informado os dados referentes a experiência de realizar o exercício.

Boca (BOCA Online Contest Administrator)

Para realizar a compilação dos exercícios no sistema de dicas, será utilizado funções do *software* BOCA Campos e Ferreira (2004) que é escrito predominantemente em PHP, o que permitiria sua execução em qualquer plataforma compatível com tal linguagem. No entanto, o sistema de julgamento depende de scripts Bash e de funcionalidades específicas do sistema operacional Linux (isolamento de recursos/jail). O Boca é amplamente utilizado em maratonas de programação em todo o Brasil, são exemplos de maratonas: Maratona de Programação da USP, Maratona de Programação da FACENS, Maratona de Programação da Faculdade de Informática de Presidente Prudente e Copa de programação da PUC-SP.

Formato do Estudo

Esta seção apresentará o formato do estudo, descrevendo como serão realizados o teste do sistema, a criação do banco de exercícios e banco de dicas e como serão respondidas as questões de pesquisa.

Teste do sistema

Após o desenvolvimento do sistema ser concluído. Antes de realizar os estudos, será necessário executar uma etapa de teste para encontrar possíveis erros no sistema. Para que essa etapa aconteça, será con. Rese vocado um grupo de alunos de forma voluntária para utilizarem as funcionalidades do sistema e reportar se existe algum erro no sistema. Os alunos deverão reportar os erros encontrados através de criação de *issues* no *GitHub* onde está o repositório do sistema. Todos os erros serão corrigidos antes de realizar os estudos para avaliar o sistema.

Banco de Exercícios

O banco de dados de exercícios será criado a partir de exercícios selecionados para diferir dos exercícios realizados na matéria de Algoritmos oferecida pelo curso, a seleção terá a finalidade de prevenir que o voluntário do estudo de avaliação do sistema já tenha realizado o exercício. Os exercícios serão divididos em três grupos, o primeiro abordará conceitos básicos da linguagem de programação como entrada e saída de dados, declaração de variáveis e constantes, operações e funções matemáticas. O segundo grupo refere-se a estruturas condicionais e o terceiro grupo trata de estruturas de repetição.

Banco de Dicas

O banco de dicas será provido após a conclusão do banco de exercícios, assim será realizada uma convocação por formulário *online* aos estudantes da Universidade Tecnológica Federal do Paraná do curso de Bacharelado em Ciência da Computação para se voluntariarem a utilizar o sistema de dicas por um determinado período em um dia que será estipulado no formulário. Os voluntários utilizaram o sistema de dicas em um ambiente controlado sendo esse um laboratório da Universidade Tecnológica Federal do Paraná com um professor ou alunos monitorando as atividades e tirando as possíveis dúvidas dos participantes.

A criação das dicas conforme a Figura 3.3 irá respeitar dois fluxos de execução, sendo eles: fluxo de reflexão e fluxo de comparação. Os dois fluxos são iniciados quando o usuário recebe uma lista de exercícios para ser realizada. Após o usuário ter uma lista de exercícios o fluxo de reflexão começa, o sistema irá apresentar um exercício da lista para ser resolvido e o usuário poderá começar a realizar a solução. Após a solução ser concluída, o usuário submete a solução para validação, o sistema irá executar a solução do exercício e verificar se está correta ou não. Caso a solução não estiver correta, o sistema redireciona para a etapa de realização de solução, nessa etapa o usuário poderá pedir uma dica para o sistema. Caso a solução estiver correta, o sistema irá redirecionar o usuário para a tela aonde ele criará a dica para o exercício e irá realizar o cadastro dela no banco de dicas.

Após o usuário gerar uma dica do exercício realizado, o sistema irá seguir o fluxo de comparação. Deste modo, o sistema irá procurar no banco de dados soluções de outros usuários do exercício realizado. O sistema irá apresentar ao usuário a pior e a melhor solução do exercício para que ele crie uma dica para melhorar as duas soluções. As dicas de otimização serão cadastradas no banco de dados de dicas para auxiliar outros usuários na resolução do exercício.

Avaliação do Sistema de Dicas

Nesta subseção será explicado as duas etapas que serão realizadas para responder as duas questões de pesquisa. A primeira etapa consiste em aplicar um estudo controlado na Universidade Tecnológica Federal do Paraná para gerar os dados de *log* de execução das listas de exercícios necessários. Dessa forma, após a geração dos dados de *log* a segunda etapa será inicializada. A segunda etapa apresenta as análises que serão feitas para responder as duas questões de pesquisa.

Tabela 3.1. Formulário para aquisição de voluntários.

Perguntas
Nome?
Idade?
Telefone?
Email?
Curso?
Tempo que está cursando o curso?
Semestre?

A Figura 3.4 representa a primeira etapa da avaliação do sistema, onde será necessário a colaboração de voluntários do curso de Bacharelado em Ciência da Computação para o experimento presencial que será realizado na Universidade Tecnológica Federal do Paraná, para

isso será criado um formulário no *Google Forms* representado na Tabela 3.1 e disponibilizado *online* para os alunos do curso.

Os voluntários serão divididos em três grupos distintos de acordo com o nível de conhecimento em programação, o nível de cada voluntário será estimado de acordo com o tempo que está na graduação e o semestre que se encontra. Os três grupos irão realizar os mesmos exercícios mas utilizaram funcionalidades diferentes do sistema,

- **Grupo 1:** utilizará o mecanismo de dicas que prove dicas de acordo com o exercício que o aluno está realizando.
- **Grupo 2:** utilizará o mecanismo de dicas personalizado que disponibiliza dicas de acordo com o exercício e o erro cometido na submissão.
- **Grupo 3:** utilizará o sistema sem o mecanismo de dicas.

Toda submissão realizada pelos três grupos de voluntários será gravada no *log* de submissões, sendo salvo os dados: a usuário que realizou o exercício, todas as submissões, os erros cometidos, o tempo demorado para obter sucesso no exercício. Após todos os voluntários tiverem finalizados todos os exercícios da lista, o estudo presencial será finalizado com a aplicação de um questionário representado na Tabela 3.2 para avaliar a usabilidade do sistema e a experiência da utilização de um sistema *web* para solucionar exercícios de programação.

Tabela 3.2. Questionário do estudo presencial.

Perguntas
pergunta1?
pergunta2?
pergunta3?
pergunta4?
pergunta5?
pergunta6?
pergunta7?

A segunda etapa da avaliação do sistema responderá as duas questões de pesquisa, será analisado os dados do *log* de submissões realizados na primeira etapa. A primeira questão de pesquisa tem como objetivo investigar se o mecanismo de dicas irá ajudar os alunos a obterem melhores resultados na execução de exercícios de programação, essa questão será respondida a partir da avaliação do desempenho dos grupos que utilizaram o sistema com o mecanismo de dicas em relação ao grupo que utilizou o sistema sem o mecanismo, também será avaliado o questionário respondido pelos voluntários.

A segunda questão: Quais dicas personalizadas ajudam mais os alunos? Será respondida com a análise da comparação do desempenho do primeiro grupo de voluntário com o segundo, e verificando se as dicas de voluntários mais experientes possuíram melhor avaliação em

relação as dicas dos voluntários menos experientes.

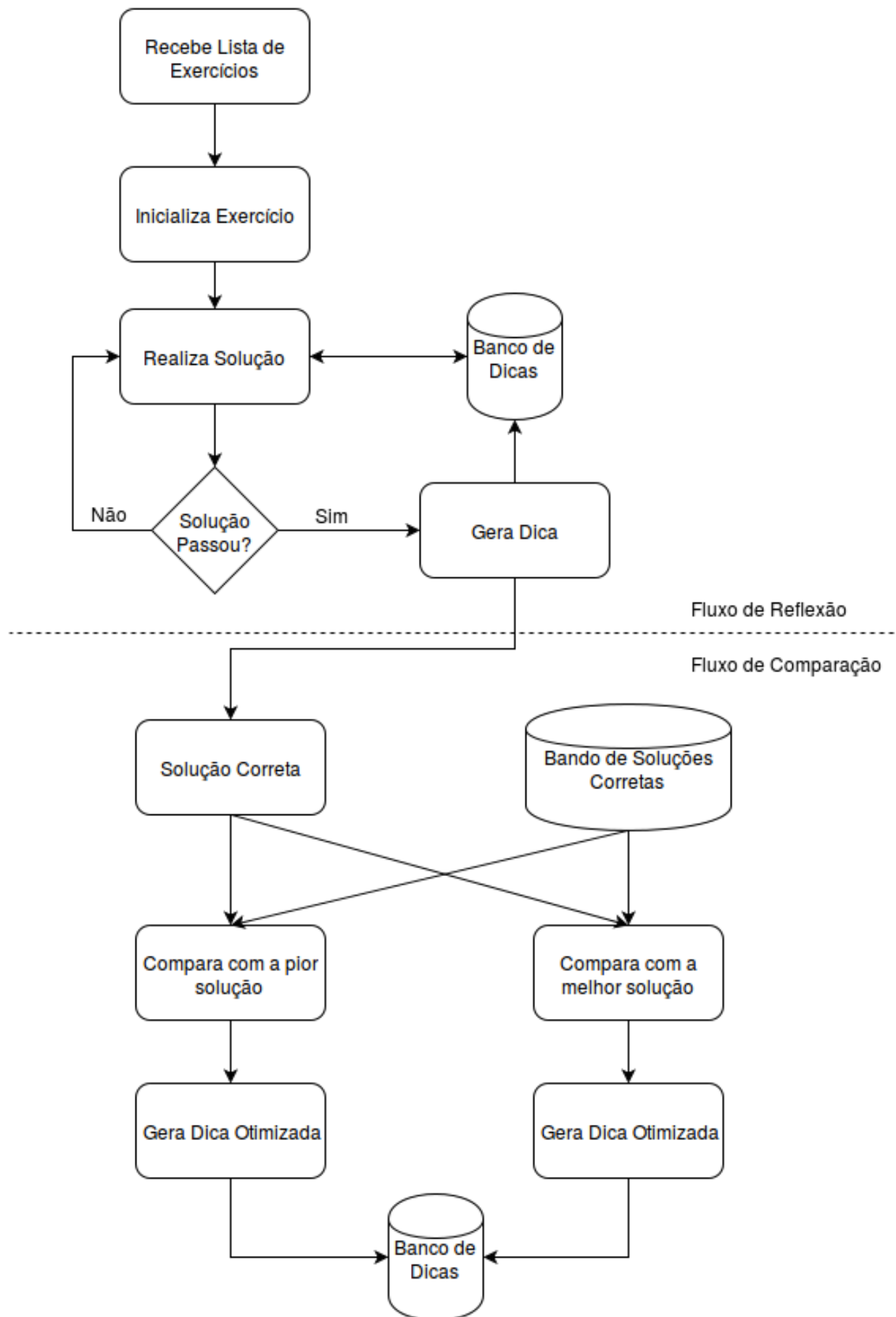


Figura 3.3. Fluxos para criação de dicas.

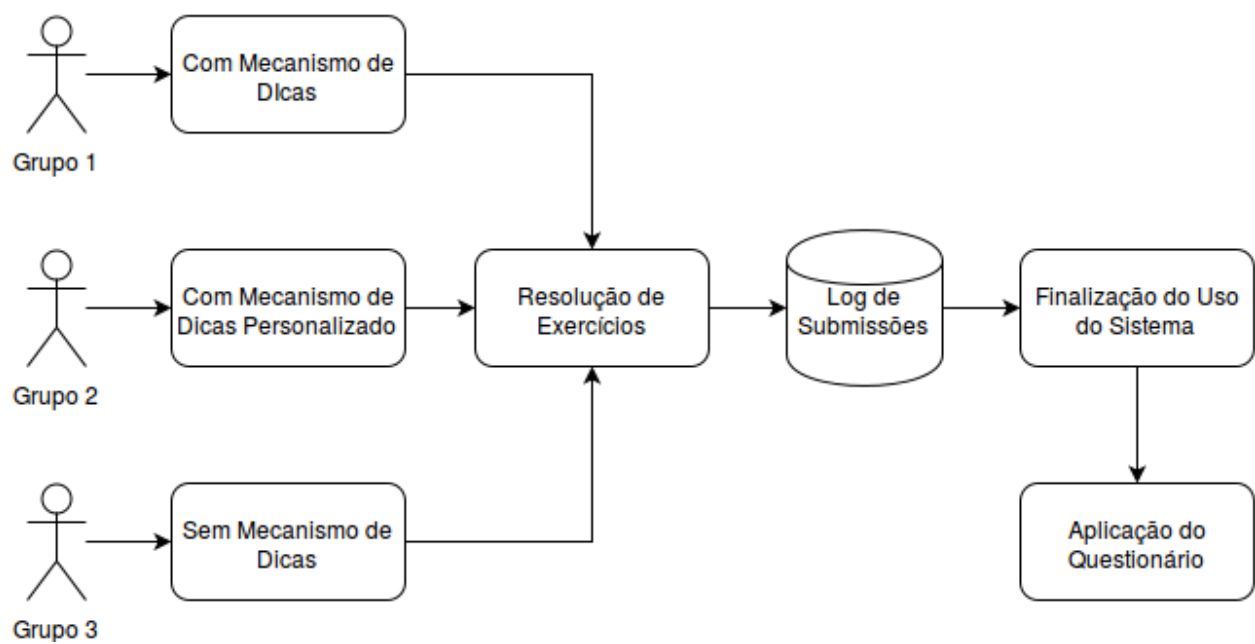


Figura 3.4. Visão geral da aplicação do estudo.

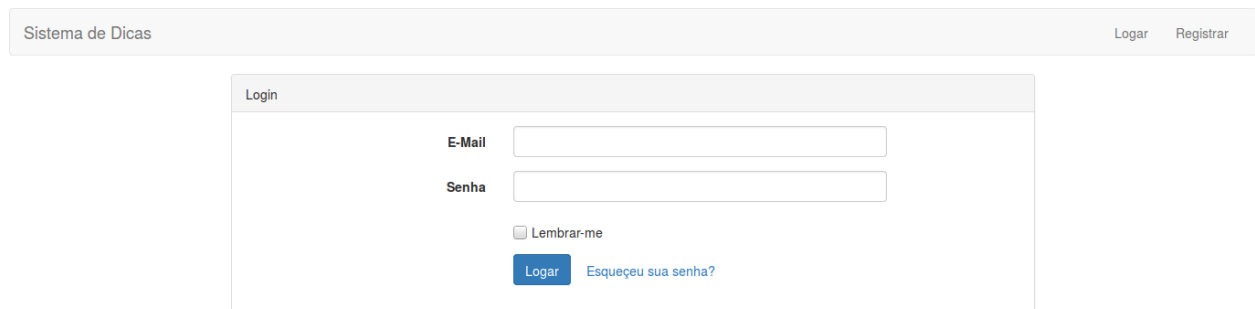
Resultados Preliminares

Neste capítulo será apresentado os resultados preliminares, onde utilizamos o diagrama de caso de uso apresentado na Figura 3.2 encontrada no Capítulo 3 para realizar o desenvolvimento do sistema no *framework* de desenvolvimento Laravel na versão 5.1.

Assim foi desenvolvido o registro de usuário, o login, a tela de resetar senha, o cadastro de professor e o cadastro de aluno representadas respectivamente nas figuras a seguir.

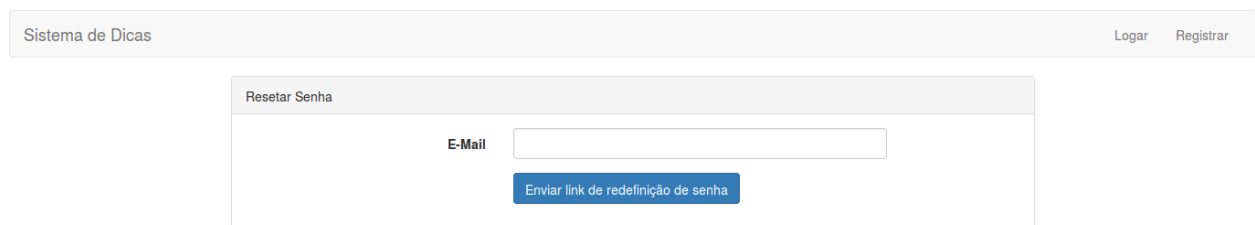
A imagem mostra a interface de registro de usuário de um sistema web. No topo, há uma barra cinza com o texto "Sistema de Dicas" à esquerda e os links "Logar" e "Registrar" à direita. Abaixo, centralizado, há um formulário branco com o título "Registrar Usuário" em uma barra cinza. O formulário contém quatro campos de entrada: "Nome", "E-Mail", "Senha" e "Confirmar Senha", cada um com um rótulo à esquerda e um campo de texto à direita. Abaixo dos campos, há um botão azul com o texto "Registrar".

Figura 4.1. Tela de registro de usuário.



The image shows a web interface for a system named "Sistema de Dicas". At the top, there is a header bar with the system name on the left and two links, "Logar" and "Registrar", on the right. Below the header, there is a "Login" form. The form has a title "Login" in a light gray box. Inside the form, there are two input fields: "E-Mail" and "Senha". Below the "Senha" field, there is a checkbox labeled "Lembrar-me". At the bottom of the form, there is a blue button labeled "Logar" and a link labeled "Esqueceu sua senha?".

Figura 4.2. Tela de login no sistema.



The image shows a web interface for a system named "Sistema de Dicas". At the top, there is a header bar with the system name on the left and two links, "Logar" and "Registrar", on the right. Below the header, there is a "Resetar Senha" form. The form has a title "Resetar Senha" in a light gray box. Inside the form, there is one input field labeled "E-Mail". Below the input field, there is a blue button labeled "Enviar link de redefinição de senha".

Figura 4.3. Tela para resetar a senha.

Sistema de Dicas Home Respostas Alunos Professores teste ▾

Novo Professor

Professor:

Email:

Telefone:

Endereço:

Cidade:

Estado:

País:

Universidade:

CEP:

[Criar Professor](#)

Figura 4.4. Tela de cadastro de professor.

Sistema de Dicas Home Respostas Alunos Professores teste ▾

Novo Aluno

Aluno:

Email:

Telefone:

Endereço:

Cidade:

Estado:

País:

Universidade:

CEP:

[Criar Aluno](#)

Figura 4.5. Tela de cadastro de aluno.

Cronograma

A Tabela 5.1 apresenta o cronograma das tarefas a serem realizadas futuramente.

Tabela 5.1. Cronograma

	Janeiro	Fevereiro	Março	Abril	Maior	Junho
Implementação do Sistema	X	X				
Experiemntos			X	X	X	
Escrita do Trabalho		X	X	X	X	X
Defesa						X

Referências

BOSSE, Yorah; GEROSA, Marco Aurélio. Reprovações e trancamentos nas disciplinas de introdução à programação da universidade de são paulo: Um estudo preliminar. In: *XXIII WEI-Workshop sobre Educação em Informática. Recife, Julho, 2015.*

CAMPOS, CP De; FERREIRA, CE. Boca: um sistema de apoio a competições de programação (boca: A support system for programming contests). In: *Workshop de Educacao em Computacao (Brazilian Workshop on Education in Computing), Congresso anual da SBC, 2004.*

CUKIERMAN, Diana. Predicting success in university first year computing science courses: The role of student participation in reflective learning activities and in i-clicker activities. In: *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, 2015. (ITiCSE '15), p. 248–253. ISBN 978-1-4503-3440-2. Disponível em: <http://doi.acm.org/10.1145/2729094.2742623>.*

CUMMINS, Stephen; STEAD, Alistair; JARDINE-WRIGHT, Lisa; DAVIES, Ian; BERESFORD, Alastair R.; RICE, Andrew. Investigating the use of hints in online problem solving. In: *Proceedings of the Third (2016) ACM Conference on Learning @ Scale, 2016. (L@S '16), p. 105–108. ISBN 978-1-4503-3726-7. Disponível em: <http://doi.acm.org/10.1145/2876034.2893379>.*

ELKHERJ, Matthew; FREUND, Yoav. A system for sending the right hint at the right time. In: *Proceedings of the First ACM Conference on Learning @ Scale Conference, 2014. (L@S '14), p. 219–220. ISBN 978-1-4503-2669-8. Disponível em: <http://doi.acm.org/10.1145/2556325.2567864>.*

HELMINEN, Juha; MALMI, Lauri. Jype - a program visualization and programming exercise tool for python. In: *Proceedings of the 5th International Symposium on Software Visualization, 2010. (SOFTVIS '10), p. 153–162. ISBN 978-1-4503-0028-5. Disponível em: <http://doi.acm.org/10.1145/1879211.1879234>.*

HOLCOMB, Kayla M.; SIMONE, Nevan F. The role of chronology in analyzing introductory programming assignments (abstract only). In: *Proceedings of the 47th ACM Technical*

Symposium on Computing Science Education, 2016. (SIGCSE '16), p. 725–726. ISBN 978-1-4503-3685-7. Disponível em: <http://doi.acm.org/10.1145/2839509.2851062>.

LAHTINEN, Essi; ALA-MUTKA, Kirsti; JÄRVINEN, Hannu-Matti. A study of the difficulties of novice programmers. In: Acm. *ACM SIGCSE Bulletin*, 2005. v. 37, n. 3, p. 14–18.

PRICE, Thomas W. Integrating intelligent feedback into block programming environments. In: *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, 2015. (ICER '15), p. 275–276. ISBN 978-1-4503-3630-7. Disponível em: <http://doi.acm.org/10.1145/2787622.2787748>.

SINCLAIR, Jane; BUTLER, Matthew; MORGAN, Michael; KALVALA, Sara. Measures of student engagement in computer science. In: *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, 2015. (ITiCSE '15), p. 242–247. ISBN 978-1-4503-3440-2. Disponível em: <http://doi.acm.org/10.1145/2729094.2742586>.