

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. 29. 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(UTFPR), 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 será realizado um estudo para investigar se o mecanismo de dicas irá ajudar os alunos a obterem melhores resultados na resolução de exercícios de programação e avaliar 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 baseado no artigo da (Glassman *et al.*, 2016). 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: Ensino de Algoritmo, Dicas, Dicas Personalizadas, Learnersourcing

Lista de figuras

2.1	Distribuições de FIN, MID e WIC. O gráfico não inclui os alunos que se retiraram do curso, nem com $FIN = 0$. Fonte (Cukierman, 2015)	12
3.1	Visão geral do método proposto.	15
3.2	Visão geral do sistema de dicas.	16
3.3	Tela de registro de usuário.	17
3.4	Tela de login no sistema.	17
3.5	Tela de cadastro de professor.	18
3.6	Tela de cadastro de aluno.	18
3.7	Fluxos de reflexão para criação de dicas.	
3.8	Fluxos de comparação para criação de dicas.	
3.9	Visão geral da aplicação do estudo.	

Lista de tabelas

2.1	Indicadores da pesquisa NSSE 2013, com fonte em (Sinclair <i>et al.</i> , 2015) . . .	11
2.2	Resumo dos resultados da UES, com fonte em (Sinclair <i>et al.</i> , 2015)	11
3.1	Formulário para aquisição de voluntários.	
3.2	Questionário do estudo presencial.	
3.3	Cronograma	

Sumário

1	Introdução	7
2	Referencial Teórico	10
2.1	Reprovações em Disciplinas de Programação	10
2.2	Novas Técnicas de Ensino	11
2.3	Mecanismo de Dicas	13
2.4	Considerações Finais	14
3	Proposta	15
3.1	Implementação do iHint	15
3.1.1	Gerenciar Usuário	16
3.1.2	Gerenciar Exercício	
3.1.3	Lista de Exercício	
3.1.4	Gerenciar Turma	
3.1.5	Realizar Exercício	
3.1.5.1	Utilização das Dicas	
3.1.5.2	Boca (<i>BOCA Online Contest Administrator</i>)	
3.1.6	Gerenciar Dica	
3.1.7	Gerenciar Diário	
3.1.8	Teste do sistema	
3.2	Estudo de Avaliação	
3.2.1	Banco de Exercícios	
3.2.2	Banco de Dicas	

3.2.3	Avaliação do Sistema de Dicas
-------	---

Cronograma

Referências

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), que realizaram uma pesquisa com 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 os alunos devem obter para entender as maiores entidades do programa em vez de apenas alguns detalhes sobre eles.

Estas dificuldades contribuem para a desistencia dos alunos nas disciplinas de programação. Entretanto, os pesquisadores estão utilizando várias metodologias e *softwares* para minimizar esse problema. Este estudo pretende realizar a implantação de um mecanismo de dicas para resolução de exercícios afim de descobrir se o desempenho e interesse dos alunos melhoram na disciplina.

Com a necessidade de prover um suporte personalizado aos alunos, Elkherj e Freund (2014) criaram um sistema de dicas que permita o professor 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 suporte a todos os alunos. Assim, a utilização desse tipo de abordagem na construção do sistema de dicas apresenta limitações em relação a quantidade de alunos

realizando os exercícios.

O método que será aplicado para a implementação do sistema é o *learnersourcing* que gerencia as atividades dos alunos através de uma interface que coleta os dados de aprendizagem dos alunos e suas avaliações de explicações e elicit a geração de novas explicações de futuros alunos.

Nós utilizamos a abordagem do *learnersourcing* apresentada por Glassman *et al.* (2016) com o intuito de que os alunos, através de sua própria experiência resolvendo exercícios, podem criar dicas úteis através de suas implementações. Estes alunos podem então gerar sugestões para colegas com base em sua própria experiência.

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 UTFPR 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 através da 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 e a pesquisa *Nacional Universidade Experience* (UES). 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. A Tabela 2.1 apresenta um resumo das pontuações dos indicadores do NSSE 2013 sendo o máximo de 60 pontos para cada indicador e quanto maior melhor. Assim, a pontuação do curso de Ciência da Computação estão abaixo da média geral para todas as categorias exceto Collaborative Learning em que é igual. Em vários indicadores, o curso de Ciência da Computação está apenas 1 ou 2 pontos (em 60) atrás, mas em *Reflective Learning* e *Learning Strategies* em particular, a diferença é maior. Em ambos esses indicadores as Ciências Físicas e Engenharia são geralmente de baixa pontuação e pode ser considerado como comparações sujeitas próximos do curso de Ciência da Computação.

Tabela 2.1. Indicadores da pesquisa NSSE 2013, com fonte em (Sinclair *et al.*, 2015)

Subject Area	CS	Phys. Sci.(not CS)	Eng	Overall
Higher Order Learning	38	40	39	39
Reflective Learning	32	34	33	39
Learning Strategies	34	39	36	41
Quantitative Reasoning	28	38	37	29
Collaborative Learning	32	36	40	32
Discussions with Diverse Others	38	41	41	41
Student Faculty Interaction	20	28	23	24
Student Faculty Interaction	37	41	38	41
Quality of interactions	42	43	41	43
Supportive environment	31	34	32	33

A Tabela 2.2 apresenta um resumo dos resultados da UES. Das categorias gerais nos UES, o curso de Ciência da Computação apresenta mal resultados em duas: segundo mais baixo no *Skills Development* em comparação dentro das 45 áreas específicas, e décimo mais baixo em *Teach Quality*. Outras categorias apresentam melhores resultados, sendo uma acima da média para *Learner Engagement* e também *Student Support* e apenas dois abaixo da média para *Learning Resource*.

Tabela 2.2. Resumo dos resultados da UES, com fonte em (Sinclair *et al.*, 2015)

Subject Area	Skills Dev	Learner Engagement	Teach Quality	Learning Resource	Student Support
CS	72	58	74	81	54
Overall	79	57	79	83	53

Novas Técnicas de Ensino

Knobelsdorf *et al.* (2014) aplicaram uma abordagem cognitiva de aprendizagem no curso teórico realizado na Universidade de Potsdam, na Alemanha, e têm levado a uma redução significativa das taxas de falha do curso. O objetivo era tornar as práticas do curso teórico de Ciência da Computação mais visíveis aos alunos e, portanto, mais fáceis de adotar. Reforçaram a modelagem introduzindo uma sessão tutorial, que consiste em apresentar as habilidades para manusear e trabalhar com o conhecimento que os alunos devem desenvolver no curso. Aplicaram o método *Scaffolding & Fading*, onde foi oferecido exercícios preparatórios específicos que devem ser resolvidos em conjunto durante a sessão de exercícios e servem como preparação para o dever de casa. E por fim, forneciam *feedback* para as submissões e trabalhos de casa realizados pelos alunos. Para avaliar a nova abordagem, foi realizado um exame final mantendo os mesmos requisitos das versões dos últimos 5 anos e obtiveram uma taxa de insucesso inferior a 10% em dois anos consecutivos que antes alcançava

de 30 a 60%.

Cukierman (2015) realizou estudos experimentais para determinar se novas atividades em sala de aula, como a instrução entre pares e a aprendizagem ativa, auxiliada pelo uso de sistemas de resposta do público (i-clickers) utilizado em palestras e o Programa de Aperfeiçoamento Acadêmico (AEP), que consiste na intervenção pró-ativa centrada nos estudantes desenvolvida e gerida pela *School of Computing Science and Student Learning Commons* na Universidade *Simon Fraser*, proporcionando oportunidades de auto-reflexão e exposição a estratégias de estudo melhoram em algumas medidas de resultado (tais como notas de exame final). Para realizar o experimento, foram definidas as seguintes variáveis:

- **AEP:** Participação em atividades da AEP. Este é um predictor dicotômico que indica se os alunos participaram (codificados como 1) ou não (codificados como 0) nas atividades da AEP.
 - **MID:** pontuações de meio termo. Este é um predictor contínuo, variando de 0 a 100.
 - **WIC:** Participação ponderada do I-clicker. Este é um predictor contínuo que mede a participação do i-clicker que varia de 0 a 100 e calculado acumulando pontos de todas as conferências quando i-clickers foram usados. Nessas palestras, os pontos foram baseados incluindo a participação e resposta correta de perguntas no i-clicker.
- item **FIN**:: Pontuações finais do exame. Este é o critério contínuo ou variável de resultado, que varia de 0 a 100.

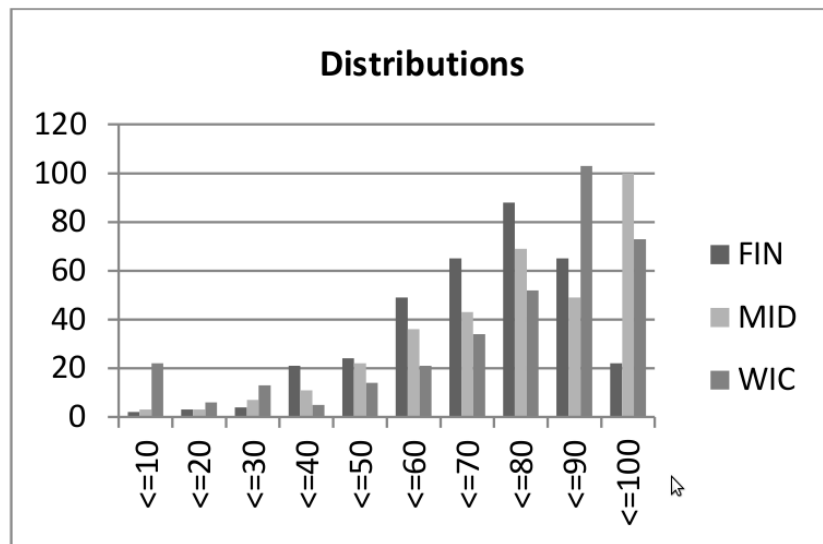


Figura 2.1. Distribuições de FIN, MID e WIC. O gráfico não inclui os alunos que se retiraram do curso, nem com FIN = 0. Fonte (Cukierman, 2015)

A Figura 2.1 apresenta o resultado obtido na pesquisa, sendo que o estudo não incluiu alunos que se retiraram do curso ou não vieram para o exame final. Intuitivamente, os alunos tiveram um desempenho muito bom no meio termo, com um grande número de alunos com

mais de 90%. Os alunos obtiveram valores muito bons nos pontos ponderados do i-clicker e no exame final pode-se observar que os resultados são mais distribuídos, embora negativamente inclinados.

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

Este capítulo apresenta o método utilizado para a elaboração deste trabalho. Assim, para avaliar o impacto do uso do mecanismo de dicas personalizadas no ensino de conceitos básicos de programação, primeiramente será realizado a implementação do iHint utilizando o *framework* de desenvolvimento Laravel¹. Subsequente, será efetuado um estudo com os dados gerados nos experimentos presenciais para realizar a avaliação do sistema. A Figura 3.1 ilustra cada etapa a ser realizada para o desenvolvimento da abordagem proposta.

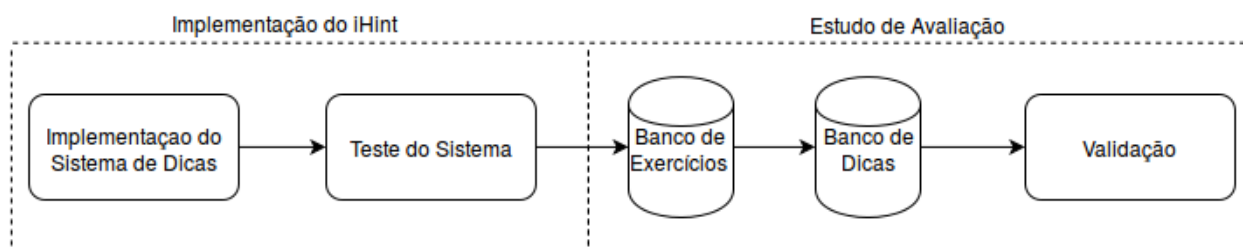


Figura 3.1. Visão geral do método proposto.

Implementação do iHint

Esta seção, descreve o sistema de dicas a ser desenvolvido. O caso de uso da Figura 3.2 apresenta as funcionalidades do iHint que serão modeladas.

¹ <https://laravel.com/>

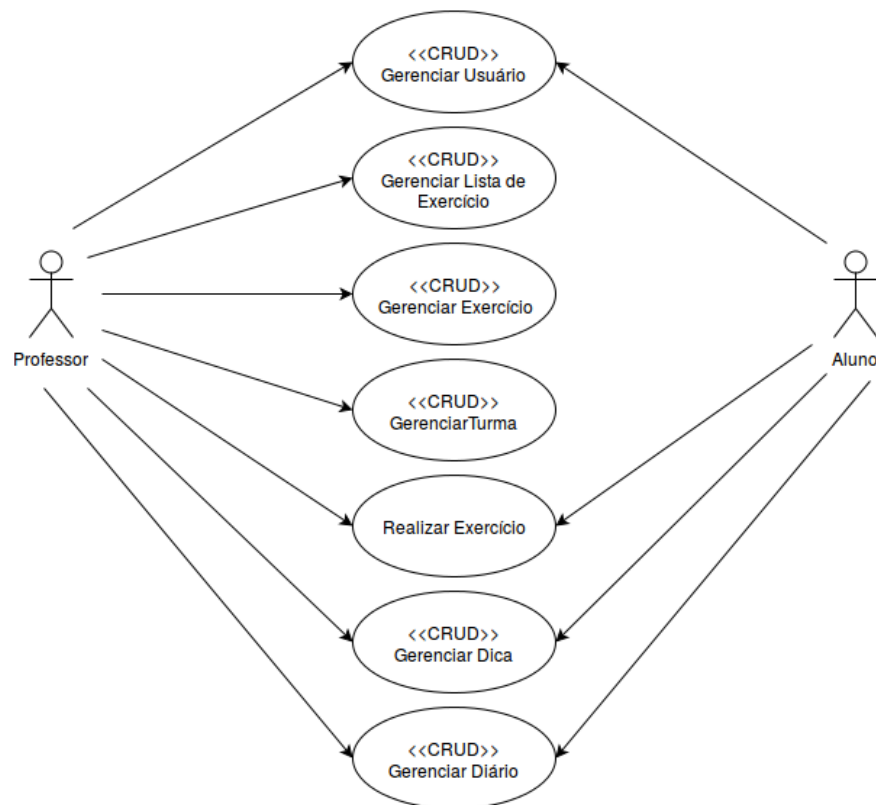


Figura 3.2. Visão geral do sistema de dicas.

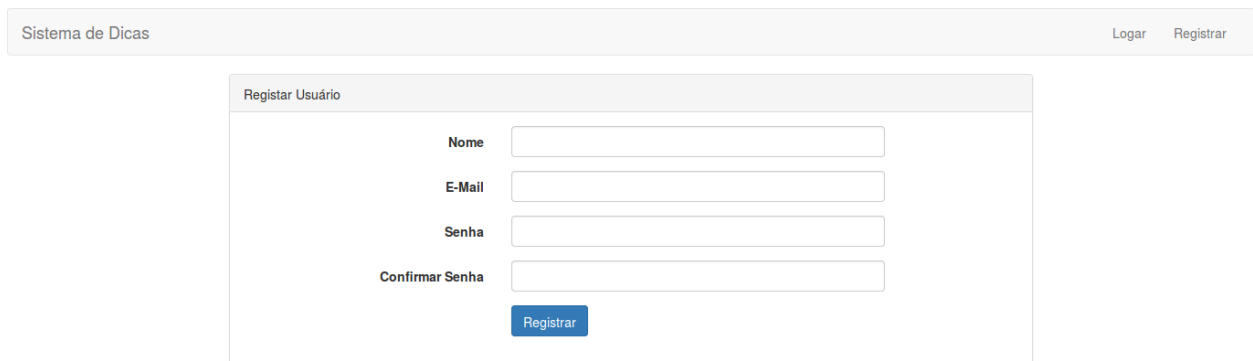
Gerenciar Usuário

Quando um usuário desejar realizar o cadastro no 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 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.

Primeiramente para o usuário se cadastrar no sistema, a primeira etapa é informar o nome, um email e a senha para o cadastro. O sistema irá verificar se o email do usuário está no formato "exemplo@exemplo.com", verificar se o email já está registrado no banco de dados e realizar a comparação do valor do campo de senha com o do campo confirmar senha. Esta funcionalidade já foi implementada conforme a tela do sistema representada na Figura 3.3.

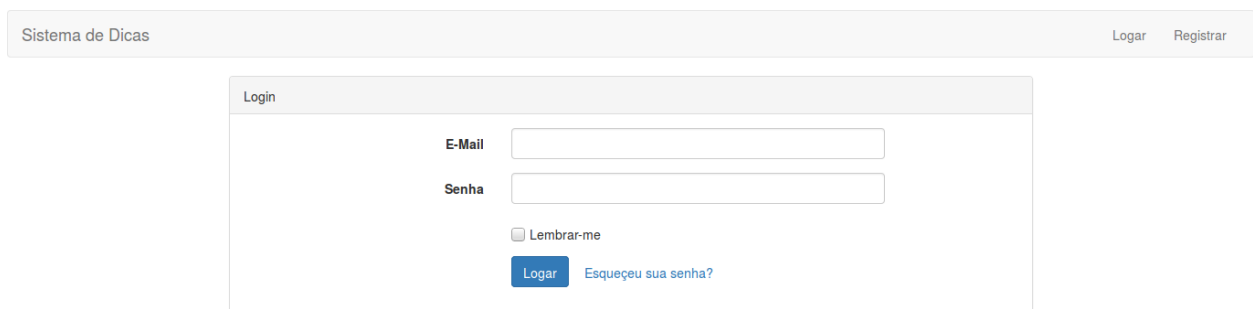
Após o usuário estar cadastrado no sistema, ele poderá realizar login através da tela representada na Figura 3.4 informando o email e a senha.

Se o usuário estiver logando pela primeira vez, ele terá que escolher se será um aluno ou professor e terá que preencher os campos apresentados nas Figuras 3.5 e 3.6



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 form titled "Registrar Usuário". The form contains four input fields: "Nome", "E-Mail", "Senha", and "Confirmar Senha". Each field is preceded by its label. Below the "Confirmar Senha" field, there is a blue button labeled "Registrar".

Figura 3.3. 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 form titled "Login". The form contains two input fields: "E-Mail" and "Senha". Each field is preceded by its label. Below the "Senha" field, there is a checkbox labeled "Lembrar-me". Below the checkbox, there is a blue button labeled "Logar" and a link labeled "Esqueceu sua senha?".

Figura 3.4. Tela de login no sistema.

Sistema de Dicas Home Respostas Alunos Professores teste ▾

Novo Professor

Professor:

Email:

Telefone:

Endereço:

Cidade:

Estado:

País:

Universidade:

CEP:

[Criar Professor](#)

Figura 3.5. 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 3.6. Tela de cadastro de aluno.

Gerenciar Exercício

Essa funcionalidade do sistema só poderá ser acessada se o usuário for um professor. Sendo assim, o professor terá que cadastrar um enunciado, a linguagem de programação que o aluno deve utilizar para resolver o exercício, o nível (fácil, médio ou difícil), o tipo do exercício (condicional ou laço de repetição), e 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.

Lista de Exercício

O professor poderá realizar o cadastro de uma lista de exercícios informando o nome da lista, os exercícios que deseja adicionar, o tempo máximo que um aluno deve realizar a lista. Os exercícios serão adicionados a partir da consulta no banco de dados de exercícios, para realizar a consulta o professor deverá informar o nível da dificuldade que deseja e o o tipo do exercício (condicional ou laço de repetição). Desta forma, o sistema irá retornar uma lista com os exercícios que satisfazem a consulta, assim o professor poderá escolher os exercícios que deseja adicionar na lista.

Gerenciar 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 ou gerar um código da sua turma através do sistema e disponibilizar para os alunos se cadastrarem.

Realizar Exercício

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 estando vinculado a uma turma, assim ele poderá resolver a lista de exercícios 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, data e dicas utilizadas.

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.

Boca (*BOCA Online Contest Administrator*)

Para realizar a compilação dos exercícios no sistema de dicas, será utilizado funções do BOCA, um *software* livre² desenvolvido por Campos e Ferreira (2004), o BOCA é um sistema de entrega de exercícios, com autenticação, controle de tempo e disponibilização de resultados, tudo em tempo real. Toda a programação do sistema foi feita na linguagem PHP, e assim é portátil para todo sistema onde tal linguagem esteja disponível. Para armazenamento dos dados e controle de concorrência é utilizado o banco de dados relacional PostgreSQL.

O Boca funciona através do envio das resoluções realizada por usuários que enviam um arquivo-fonte para a validação do sistema, o resultado é retornado para o usuário o mais breve possível. No momento em que o usuário submete seu arquivo-fonte, ele poderá escolher a linguagem que o programou e para qual problema ele é destinado. Através do ambiente de janelas do navegador do BOCA, o usuário escolhe o arquivo do disco que deseja submeter e o envia para correção, o BOCA irá compilar esse arquivo-fonte e irá enviar um resultado para o usuário.

² <https://github.com/cassiopc/boca/>

Gerenciar 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.

Gerenciar 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 deseje 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.

Teste do sistema

Após o desenvolvimento do sistema ser concluído, será necessário executar uma etapa de teste para encontrar possíveis erros no sistema. Para que essa etapa aconteça, será convocado um grupo de alunos da UTFPR do Curso de Bacharelado em Ciência da Computação de forma voluntária para utilizarem as funcionalidades do sistema, o estudo será realizado em um laboratório com a supervisão de um professor ou aluno. Os alunos deverão reportar os erros encontrados através de criação de *issues* no *GitHub* ³. Todos os erros serão corrigidos antes de realizar os estudos para avaliar o sistema.

Estudo de Avaliação

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

³ <https://github.com/gustavoCorreiaGonzalez/aplicacao-tcc>

Banco de Exercícios

Para que o estudo e a avaliação do sistema seja possível, será necessário criar um banco de dados de exercícios 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

Para realizar o estudo UTFPR com os voluntários, será preciso que o sistema já tenha algumas dicas cadastradas no banco de dados. Deste modo, o banco de dicas será provido após a conclusão do banco de exercícios, assim será realizada uma convocação por formulário *online* aos estudantes da UTFPR 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 UTFPR com um professor ou aluno monitorando as atividades e tirando as possíveis dúvidas dos participantes.

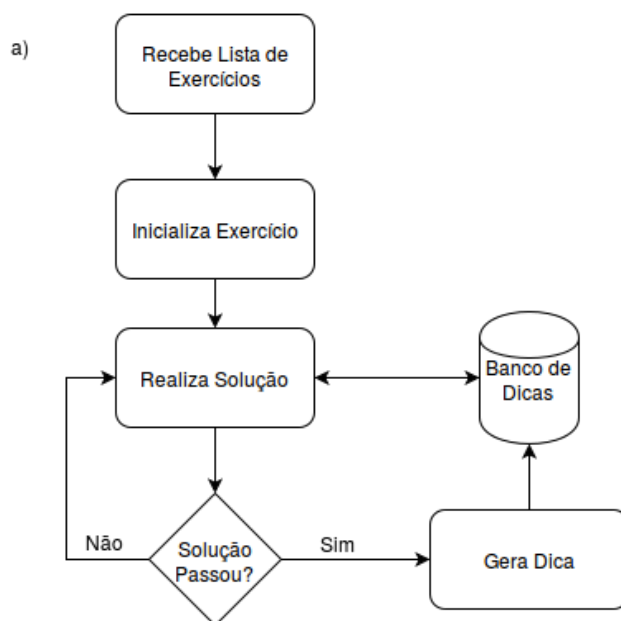


Figura 3.7. Fluxos de reflexão para criação de dicas.

A criação das dicas ocorrerá conforme dois fluxos de execução, sendo eles: fluxo de reflexão representado pela Figura 3.7 e fluxo de comparação representado pela Figura 3.8. Os

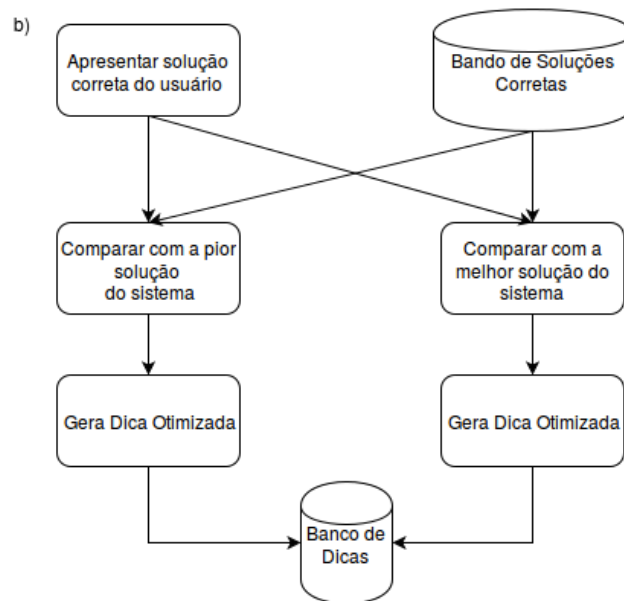


Figura 3.8. Fluxos de comparação para criação de dicas.

dois fluxos são iniciados quando o usuário recebe uma lista de exercícios para ser realizada. No momento em que 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, o aluno poderá requisitar ao sistema que necessita realizar uma consulta a algumas dicas. Deste modo, o sistema apresentará XXXX dicas para o aluno consultar e realizar sua solução.

Após a solução ser concluída, o usuário submete a solução para validação, o BOCO que 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. Neste fluxo, o sistema irá procurar no banco de dados soluções de outros usuários do exercício realizado. O sistema irá apresentar ao usuário uma das piores e a uma das melhores soluções do exercício para que ele crie uma dica para melhorar as soluções. Para o sistema conseguir distinguir uma boa solução de outra ruim, cada solução será classificada tendo em conta os dados da execução do exercício e as métricas retiradas do código. Na execução do exercício será avaliado o tempo para a resolução correta do exercício, quantidade de dicas utilizadas e quantidades de tentativas erradas. No entanto, as métricas do código serão medidas através do número de linhas entre as tentativas, complexidade ciclomática e tamanho da solução. Com esses dados analisados, o sistema poderá realizar um ranqueamento com as soluções e será considerado 25% das primeiras posições como o grupo das melhores

soluções e 25% das ultimas colocação como sendo o grupo das piores soluções. Assim, o sistema irá retornar para o aluno duas soluções que serão selecionadas de forma aleatória, sendo uma do grupo das melhores soluções e a outra do grupo das piores soluções.

A partir das soluções apresentadas ao aula pelo sistema, ele poderá analisar e gerar uma dica para melhorar as soluções. Essas dicas de otimização serão cadastradas no banco de dados de dicas para auxiliar outros usuários na resolução do exercício.

Avaliação do Sistema de Dicas

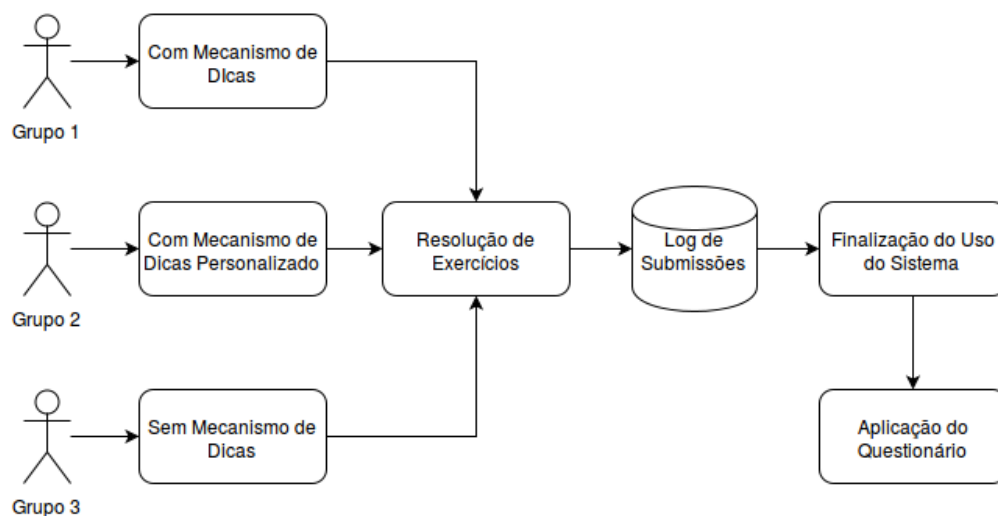


Figura 3.9. Visão geral da aplicação do estudo.

Nesta subseção será explicado as duas etapas que serão realizadas para responder as duas questões de pesquisa citadas abaixo.

- **QP₁:** *Existe impacto do uso do mecanismo de dicas para minimizar erros e melhorar soluções de exercícios?*
- **QP₂:** *Quais dicas personalizadas ajudam mais os alunos?*

Enquanto **QP₁** tem como objetivo investigar se o mecanismo de dicas irá ajudar os alunos a obterem melhores resultados na resolução de exercícios de programação. **QP₂** 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.

A primeira etapa consiste em aplicar um estudo controlado na UTFPR para gerar os dados de *log* de submissões das listas de exercícios. 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 questões de pesquisa.

Tabela 3.1. Formulário para aquisição de voluntários.

Perguntas
Nome?
Idade?
Telefone?
Email?
Curso?
Tempo que está cursando o curso?
Qual semestre você está?
Quantidade de vezes que cursou a disciplina de Algoritmos?
Teve experiência com programação antes da graduação?
Trabalha ou já trabalhou com programação?

A Figura 3.9 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 UTFPR, 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.

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 será respondida a partir da avaliação do desempenho dos grupos que utilizaram o sistema no estudo controlado. Será realizada uma comparação estatística para avaliar as hipóteses descritas abaixo.

Tabela 3.2. Questionário do estudo presencial.

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

- **Hipótese 1:** O grupo 1 desenvolverá soluções melhores com a utilização dos mecanismos de dicas em relação as soluções do grupo 3.
- **Hipótese 2:** O grupo 2 desenvolverá soluções melhores com a utilização dos mecanismos de dicas personalizadas em relação as soluções do grupo 3.
- **Hipótese 3:** O grupo 1 desenvolverá soluções melhores com a utilização dos mecanismos de dicas em relação as soluções do grupo 2.
- **Hipótese 4:** O grupo 2 desenvolverá soluções melhores com a utilização dos mecanismos de dicas personalizadas em relação as soluções do grupo 1.

A segunda questão será respondida com a análise estatística da comparação do desempenho do grupo 1 de voluntário com o grupo 2, será verificado se as dicas dos voluntários mais experientes possuíram melhor avaliação em relação as dicas dos voluntários menos experientes.

Cronograma

A Tabela 3.3 apresenta o cronograma das tarefas a serem realizadas futuramente.

Tabela 3.3. Cronograma

	Janeiro	Fevereiro	Março	Abril	Maiο	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>.

GLASSMAN, Elena L.; LIN, Aaron; CAI, Carrie J.; MILLER, Robert C. Learnersourcing personalized hints. In: *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, 2016. (CSCW '16), p. 1626–1636. ISBN 978-1-4503-3592-8. Disponível em: <http://doi.acm.org/10.1145/2818048.2820011>.

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>.

KNOBELSDORF, Maria; KREITZ, Christoph; BÖHNE, Sebastian. Teaching theoretical computer science using a cognitive apprenticeship approach. In: *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, 2014. (SIGCSE '14), p. 67–72. ISBN 978-1-4503-2605-6. Disponível em: <http://doi.acm.org/10.1145/2538862.2538944>.

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>.