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. Marco Aurélio Graciotto Silva

Coorientador: Prof. Dr. Igor Scaliante Wiese

CAMPO MOURÃO

2016

Resumo

Gonzalez, Gustavo. Um estudo sobre o uso do mecanismo de dicas personalizadas no ensino de conceitos básicos de programação. 2016. 20. 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 várias 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 do 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 dos exercícios. Também será avaliado qual tipo de usuário (iniciantes, experientes ou professores) ofereceu as dicas que mais ajudaram os alunos.
Método: Será desenvolvido um *software* para auxiliar o aprendizado de programação baseado no sistema *Dear Beta* desenvolvido por Glassman et al. (2016), no qual foi utilizado um mecanismo de dicas para auxiliar os 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. Os voluntários realizarão 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 um, 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 desistências do curso de Bacharelado em Ciência da Computação (CBCC).
Palavras-chaves: Ensino de Algoritmo, Dicas, Dicas Personalizadas, Learnersourcing

Lista de figuras

Lista de tabelas

Sumário

Capítulo 1

Introdução

O ensino de linguagens de programação visa propiciar aos alunos o desenvolvimento de 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). 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 permite ao professor enviar uma dica personalizada em tempo real após verificar que o aluno realizou várias tentativas para resolver um exercício. Entretanto, com o aumento do número de alunos utilizando o sistema, os professores não conseguem oferecer suporte a todos eles. 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 usuários, suas avaliações de explicações e elícita a geração de novas explicações para 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, possam criar dicas úteis através de suas implementações, gerando sugestões para colegas com base em sua própria experiência.

Capítulo 2

Delimitação

Este estudo utiliza as áreas de Engenharia de Software, Análise Qualitativa e Quantitativa de dados. A Engenharia de Software é utilizada no desenvolvimento do sistema, onde é implementado padrões de projetos na codificação do sistema. Após a implementação e o experimento, será realizada a validação do sistema por meio da análise dos dados. Antes do experimento final as dicas passarão por uma análise qualitativa, para excluir dicas que não sejam conveniente para o sistema. Assim quando ocorrer o experimento final com os alunos do primeiro ano de Ciência da Computação da UTFPR, será realizada uma análise quantitativa dos dados gerados na utilização do sistema feita pelos alunos.

Capítulo 3

Problemas e premissas

O problema do insucesso é evidenciado por Lahtinen, Ala-Mutka e Järvinen (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. Portanto, são essas 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 desistência dos alunos nas disciplinas de programação, por isso os pesquisadores estão utilizando várias metodologias e *softwares* para minimizar esse problema.

Capítulo 4

Objetivos

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, será aplicado um experimento controlado com três grupos distintos de alunos. O primeiro grupo utilizará o mecanismo de dicas 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 ao usuário resolver exercícios nas linguagens de programações C, C++ e Java, além de fornecer dicas para a solução de exercícios e um diário para relatar as dificuldades enfrentadas durante a resolução.

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 experientes são melhores ou não em relação as dicas dos alunos com menos experiência.

Capítulo 5

Proposta

Este capítulo apresentará 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 Laravel[1]. Subsequente, será efetuado um estudo com os dados gerados nos experimentos presenciais para realizar a avaliação do sistema. A Figura 5.1 ilustra cada etapa a ser realizada para o desenvolvimento da abordagem proposta.

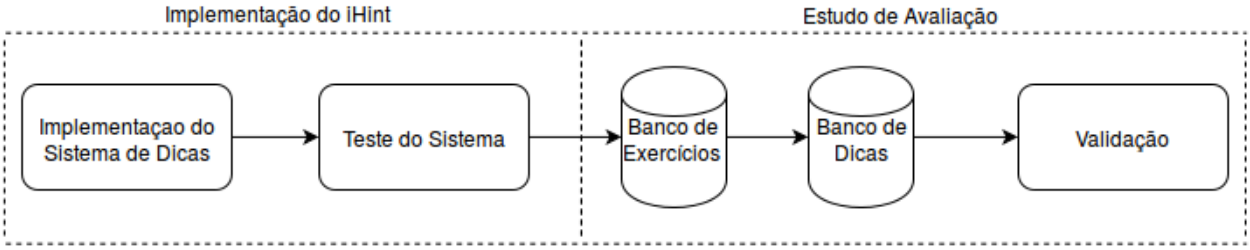

Figura 5.1. Visão geral do método proposto.

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 (CBCC) 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* [2]. Todos os erros serão corrigidos antes de realizar os estudos para avaliar o sistema.

[1] <https://laravel.com/>
[2] <https://github.com/gustavoCorreiaGonzalez/aplicacao-tcc >

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.

Banco de Exercícios

Para que o estudo e a avaliação do sistema sejam possíveis, 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 dois grupos, o primeiro abordará estruturas condicionais e o segundo 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 CBCC 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.

A criação das dicas ocorrerá conforme dois fluxos de execução, sendo eles: fluxo de reflexão representado pela Figura 5.2 e fluxo de comparação representado pela Figura 5.3. Os 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 5 dicas para o aluno consultar e realizar sua solução.

Após a solução ser concluída, o usuário submete a solução para validação, o BOCA 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 5 dicas 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.

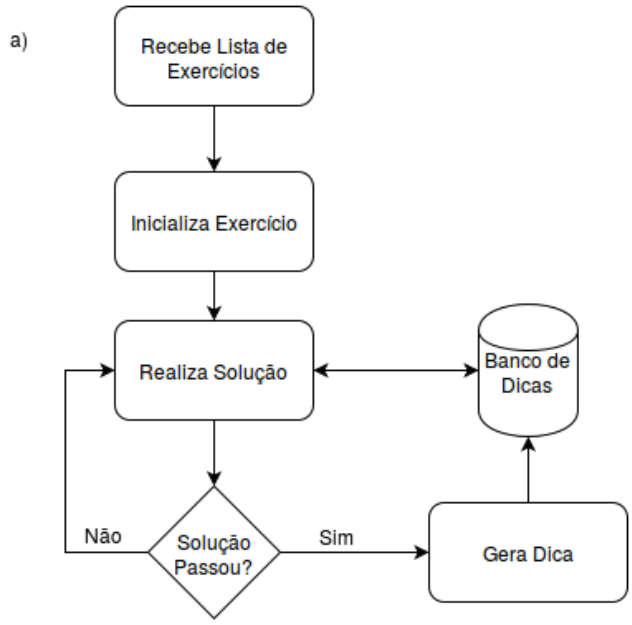

Figura 5.2. Fluxo de reflexão para criação de dicas.
Fonte: adaptada de Glassman et al. (2016)

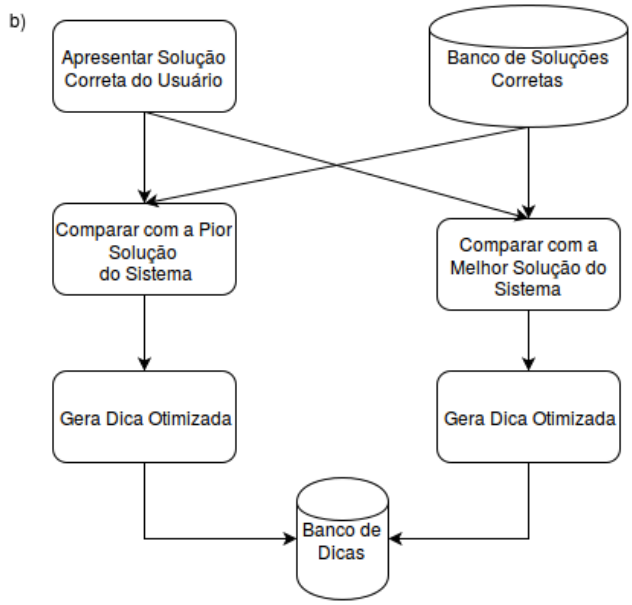

Figura 5.3. Fluxo de comparação para criação de dicas.
Fonte: adaptada de Glassman et al. (2016)

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 realizadas por outros usuários para que ele crie uma dica para melhorar as soluções. Para o sistema conseguir distinguir uma boa solução de uma 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 *ranking* com as soluções e será considerado 25% das primeiras posições como o grupo das melhores soluções e 25% das últimas colocações 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 aluno 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

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

- $\mathbf{QP}_1$: *Existe impacto do uso do mecanismo de dicas para minimizar erros e melhorar soluções de exercícios?*
  $\mathbf{QP}_2$: *Quais dicas personalizadas ajudam mais os alunos?*

Enquanto $\mathbf{QP}_1$ tem como objetivo investigar se o mecanismo de dicas irá ajudar os alunos a obterem melhores resultados na resolução de exercícios de programação. $\mathbf{QP}_2$ avaliará qual tipo de dica é melhor, podendo ser aleatorizadas em função do tipo do exercício ou geradas a partir de erros cometidos em exercícios passados. Levando em consideração a fonte da dica gerada, podendo ser por alunos iniciantes, alunos experientes ou professores.

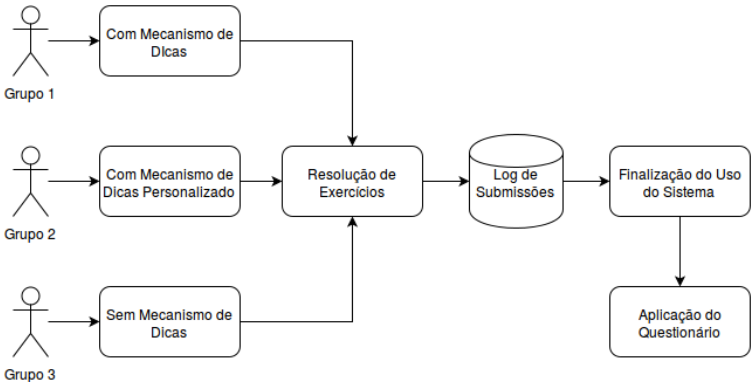

Figura 5.4. Visão geral da aplicação do estudo.

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 5.1. Formulário para aquisição de voluntários.

Perguntas
Nome?
Idade?
Telefone?
E-mail?
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 5.4 representa a primeira etapa da avaliação do sistema, onde será necessário a colaboração de voluntários do CBCC para o experimento presencial que será realizado na UTFPR, para isso será criado um formulário no *Google Forms* representado na Tabela 5.1 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 o sistema de três formas diferentes:

- **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: o 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 para os voluntários que utilizaram o mecanismo de dicas representado na Tabela 5.2 e outro questionário para os voluntários que não utilizaram o mecanismo de dicas representado na Tabela 5.3 para avaliar a usabilidade do sistema e a experiência da utilização de um sistema *web* para solucionar exercícios de programação. As respostas dos questionários será realizada na escala Likert (1932) de 5 pontos (1: Não concordo totalmente, 2: Não concordo parcialmente, 3: Indiferente, 4: Concordo parcialmente, 5: Concordo totalmente).

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

Tabela 5.2. Questionário do estudo presencial para os voluntários que utilizaram o mecanismo de dicas.

Questionário
Foi fácil utilizar o *iHint*?
Precisou do apoio de uma pessoa para usar o sistema?
A interface do sistema é agradável?
O sistema oferece todas as informações necessárias para completar as tarefas?
Você recomendaria este sistema para outras pessoas?
O objetivo do sistema foi alcançado?
O mecanismo de dica ajudou na resolução dos exercícios?
Foi fácil acessar as dicas pelo sistema?
As dicas foram apresentadas de forma clara e objetiva

Tabela 5.3. Questionário do estudo presencial para os voluntários que não utilizaram o mecanismo de dicas.

Questionário
Foi fácil utilizar o *iHint*?
Precisou do apoio de uma pessoa para usar o sistema?
A interface do sistema é agradável?
O sistema oferece todas as informações necessárias para completar as tarefas?
Você recomendaria este sistema para outras pessoas?
O objetivo do sistema foi alcançado?

o sistema no estudo controlado. Será realizada uma comparação estatística para avaliar as hipóteses descritas abaixo.

- **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.

Utilizaremos o método estatístico *Mann-Whitney-Wilcoxon Test*[3] para analisar as métricas: número de linhas entre as tentativas, complexidade ciclomática e tamanho da solução. Como cada grupo de voluntários contém alunos diferentes, então os dados gerados de cada grupo não afetaram uns aos outros, portanto são independentes. Assim, sem assumir que os dados tem distribuição normal, decidimos em 0,05 o nível de significância se os dados do número de linhas entre as tentativas, complexidade ciclomática e tamanho da solução tem distribuição de dados idênticos.

[3] <http://www.r-tutor.com/elementary-statistics/non-parametric-methods/mann-whitney-wilcoxon-test>

Deste modo, a hipótese nula é quando os valores que são parte da amostra de cada grupo vem de populações idênticas. Para testar as hipóteses, será aplicado o teste estatístico para comparar as amostras e analisar o valor de p que será retornado, este valor será comparado com o nível de significância e caso o valor de p seja menor, rejeitamos a hipótese nula, tornando assim a hipótese verdadeira. Caso o valor de p seja maior, aceitamos a hipótese nula e a hipótese será falsa.

Também será aplicado o teste *Effect Size* (MACBETH; RAZUMIEJCZYK; LEDESMA, 2011), para verificar a significância dos dados gerados por cada grupo de voluntários, será verificado se os dados possuem eventuais diferenças entre as médias ou variâncias. Mas como o estudo presencial não foi realizado, não temos os dados para análise e não foi possível escolher qual método de média ou variância será utilizado.

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. Os usuários serão divididos em mais experiente e menos experiente através dos dados informados no cadastro, sendo os dados cadastrais: o tempo que está cursando o CBCC, o semestre que está cursando, a quantidade de vezes que cursou a disciplina de Algoritmos, se o usuário teve alguma experiência com programação antes da graduação, se trabalha ou já trabalhou com programação e se é um aluno ou professor. Assim, a partir desses dados será possível classificar os usuários de acordo com sua experiência na faculdade e fora dela. Após ser realizada a classificação dos usuários, será classificada as dicas a partir da avaliação feita pelos voluntários no estudo presencial, assim será analisado que tipo de usuário contribuiu para as dicas que mais ajudaram os voluntários no estudo presencial.

Capítulo 6

Demonstrativo da aplicação dos recursos

Para realizar o estudo, o sistema deve estar hospedado em um servidor para que todos os voluntários tenham acesso em qualquer computador com Internet. Esse servidor necessita de bom desempenho para atender a demanda de acessos que serão realizados durante os experimentos. Para hospedar o sistema de dicas, foi escolhido o Digital Ocean que provê o serviço de *Cloud computing*. Inicialmente para o teste do sistema, foi contratado o serviço básico que disponibiliza um servidor, com o custo de US$ 5 (cinco dólares) por mês, com as seguintes configurações:

- 512MB de memória RAM
- 1 Core de processador
- 20GB de armazenamento de disco em SSD
- 1TB de transferência de rede

Se houver sobrecarga na utilização do sistema, será necessário contratar um plano com mais recursos. O segundo plano, custando US$ 10 (dez dólares) por mês, oferece as seguintes configurações:

- 1GB de memória RAM
- 1 Core de processador
- 30GB de armazenamento de disco em SSD
- 2TB de transferência de rede

Capítulo 7

Cronograma

A Tabela 7.1 apresenta o cronograma das tarefas a serem realizadas futuramente.

Tabela 7.1. Cronograma

	Janeiro	Fevereiro	Março	Abril	Maio	Junho
Implementação do Sistema	X	X				
Escrita do Trabalho		X	X	X	X	X
Teste do Sistema			X			
Banco de Exercícios			X			
Banco de Dicas				X		
Avaliação do Sistema de Dicas				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*. Recife, PB, Brasil: WEI, 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*. Salvador, BA, Brasil: 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*. New York, NY, USA: ACM, 2015. (ITiCSE '15), p. 248–253. ISBN 978-1-4503-3440-2. Disponível em: <http://doi.acm.org/10.1145/2729094.2742623>.

CUMMINS, Stephen et al. Investigating the use of hints in online problem solving. In: *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*. New York, NY, USA: ACM, 2016. (L@S '16), p. 105–108. ISBN 978-1-4503-3726-7. Disponível em: <http://doi.acm.org/10.1145/2876034.2893379>.

EDWARDS, Stephen H.; MARTIN, Joshua; SHAFFER, Clfford A. Examining classroom interventions to reduce procrastination. In: *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: ACM, 2015. (ITiCSE '15), p. 254–259. ISBN 978-1-4503-3440-2. Disponível em: <http://doi.acm.org/10.1145/2729094.2742632>.

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*. New York, NY, USA: ACM, 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. et al. Learnersourcing personalized hints. In: *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. New York, NY, USA: ACM, 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*. New York, NY, USA: ACM, 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*. New York, NY, USA: ACM, 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*. New York, NY, USA: ACM, 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*. New York, NY, USA, 2005. v. 37, n. 3, p. 14–18.

LIKERT, Rensis. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

MACBETH, Guillermo; RAZUMIEJCZYK, Eugenia; LEDESMA, Rubén Daniel. Cliff's delta calculator: A non-parametric effect size program for two groups of observations. *Universitas Psychologica*, Pontificia Universidad Javeriana, v. 10, n. 2, p. 545–555, 2011.

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

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