

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. 18. 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 da pesquisa será realizado na Universidade Tecnológica Federal do Paraná, com alunos selecionados entre o primeiro e segundo ano das disciplinas de Algoritmos e Algoritmos e Estruturas de Dados 1.

**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:

- **QP<sub>1</sub>:** *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.

- **QP<sub>2</sub>:** *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

utilizará um sistema de dicas personalizadas aos alunos no momento da realização de exercícios. Para validar a eficiência do software será realizado um estudo controlado com três grupos de alunos distribuídos de forma homogênea em relação ao nível de conhecimento na 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.

**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:** Aplicação, 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

## Lista de tabelas

---

# Sumário

---

<b>1</b>	<b>Introdução</b>	<b>6</b>
<b>2</b>	<b>Referencial Teórico</b>	<b>9</b>
2.1	Reprovações em disciplinas de programação . . . . .	9
2.2	Mecanismo de dicas . . . . .	10
2.3	Abordagens de ensino gerazao . . . . .	11
<b>3</b>	<b>Proposta</b>	<b>12</b>
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 . . . . .	14
3.1.5	Cadastro de Dica . . . . .	15
3.1.6	Cadastro do Diário . . . . .	15
3.1.7	Boca . . . . .	15
3.2	Banco de Exercícios . . . . .	15
3.3	Banco de Dicas . . . . .	16
	<b>Referências</b>	<b>17</b>

# Introdução

- terceira pessoa do singular - sujeito oculto

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 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 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 o *learnersourcing*



trabalho prévio sobre *learnersourcing* apresentou uma abordagem para os alunos a se envolver em tarefas de computação humanas durante a tentativa de aprender uma nova habilidade. A nossa visão fundamental é que os alunos, através de sua própria experiência lutando com um problema particular, podem se tornar especialistas sobre as otimizações específicas que implementam ou bugs que resolver. Estes alunos podem então gerar sugestões para colegas com base em sua nova perícia. Nós apresentamos os fluxos de trabalho que colhem e organizar o conhecimento coletivo dos alunos e conselhos para ajudar os companheiros novatos através de problemas de design em engenharia. Sistemas que incorporam cada fluxo de trabalho foram avaliadas no contexto de um colégio de nível classe arquitetura de computador com uma matrícula de mais de duas centenas de estudantes a cada semestre. Mostramos que, dadas as nossas escolhas de design, os alunos podem criar dicas úteis para os seus pares que aumente ou até mesmo substituir 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 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(o que vou fazer para investigar)

No software serão implementadas funcionalidades que permitam o usuário resolver exercícios nas linguagens de programações C, C++ e Java, fornecer 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 na 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 por métrica x e y) e blabla. Por fim também serão avaliadas se as dicas blablabla.

– numero de linhas entre as tentativas – tamanho da solução – complexidade ciclomática  
ok

um grupo utilizará o software de dicas normal, outro irá dispor de dicas personalizado e o último utilizará o software sem o sistema de dicas personalizadas.

Os próximos capítulos estão organizados em três partes. O objetivo do capítulo de referencial teórico é apresentar e discutir os principais conceitos que envolvem este trabalho.

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, 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 durante os últimos 15 anos, o governo australiano realiza a pesquisa *Nacional Universidade Experience* (UES) que é usada desde 2012 e no Reino Unido utiliza 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 apresentam taxas mais baixas do que a média em muitos dos principais pontos de referência de engajamento.

## Mecanismo de dicas

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 funcionalidades 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 acabaram não tentando responder a pergunta

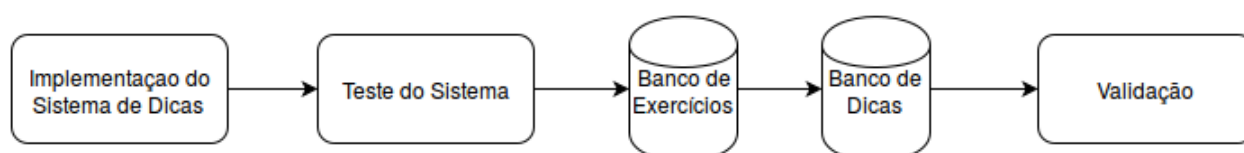
- fazer um resumo das abordagens de ensino e o engajamento dos estudantes

## **Abordagens de ensino geracao**

criar um capítulo de sumarização dos artigos relacionados e comentar o porque de usar o sistema de dicas que vou criar.

## 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.

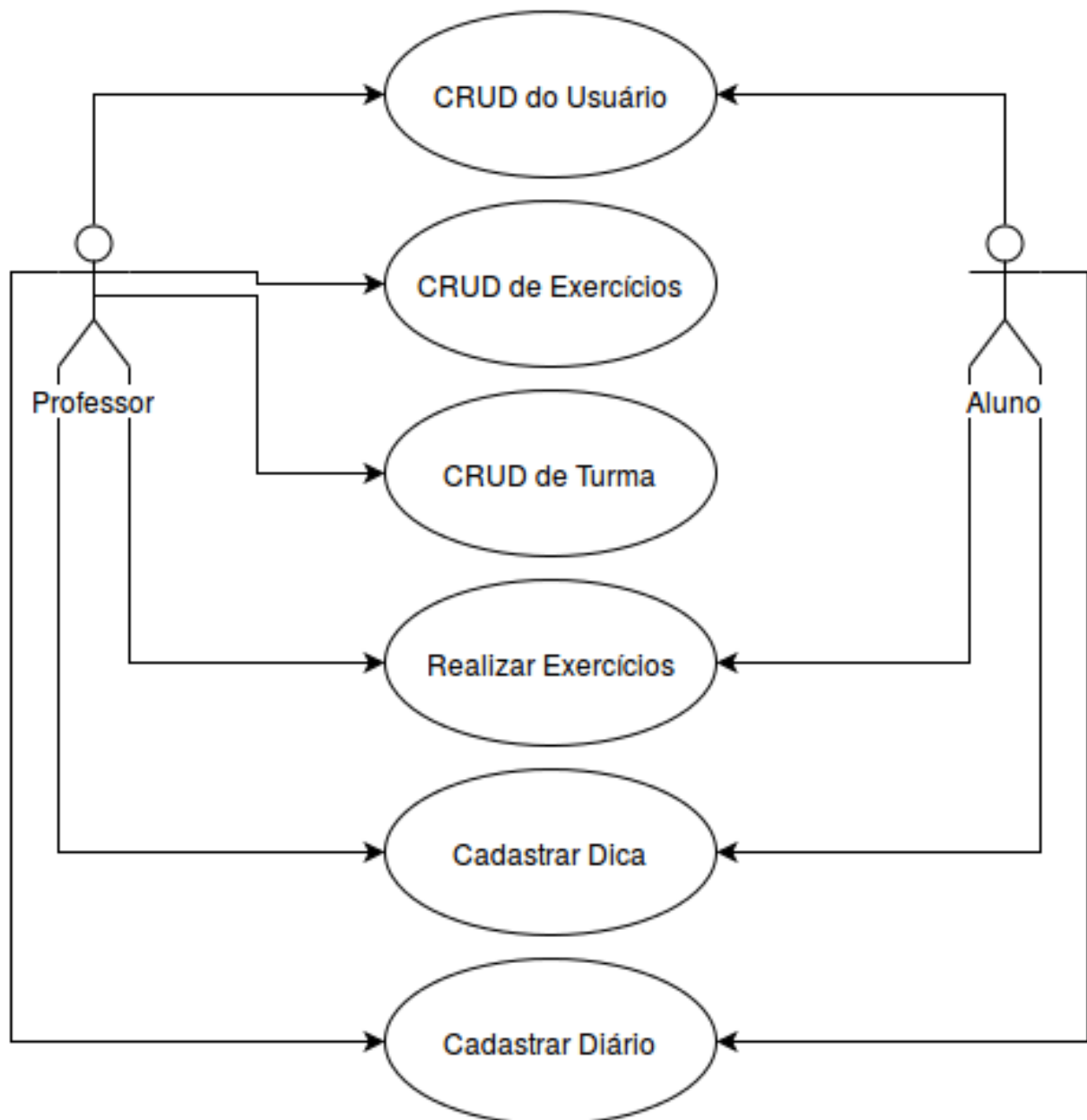
Para avaliar o impacto do uso do mecanismo de dicas personalizadas no ensino de conceitos básicos de programação, primeiramente será realizado a implementação do sistema de dicas utilizando o *framework* de desenvolvimento Laravel na versão 5.1.

## Implementação do Mecanismo de Dicas

Essa 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 usuário terá que cadastrar um enunciado, qual linguagem é para ser utilizada na resolução, o nível (fácil, médio ou difícil), o tipo do exercício (condicional, laço de repetição,

função, etc), a resposta que poderá ser escrita em um campo ou um *upload* do arquivo com as respostas. O professor poderá opinar por adicionar o exercício cadastrado em uma lista de 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 é encaminhado para a tela principal do sistema onde poderá realizar as atividades permitidas para cada tipo de usuário. Caso a opção seja realizar um exercício, o usuário poderá escolher qual tipo e nível de dificuldade do exercício deseja realizar, após preencher esses dados o sistema retornará uma lista com alguns exercícios que satisfazem os campos de busca. O usuário irá escolher um exercício, realiza-lo na linguagem de programação que deseja e submeter para a validação do sistema.

O exercício poderá estar correto ou não. Se estiver correto o sistema irá apresentar uma mensagem de parabenização e irá redirecionar o usuário para página de busca de exercícios, caso seja rejeitado o sistema retornará qual tipo de erro ocorreu na compilação do exercício, o usuário poderá realizar submissões até que consiga obter sucesso na submissão ou requisitar uma dica para o sistema. Cada submissão, tanto correta ou incorreta, é gravada no banco de dados na forma de *log*, sendo salvo o usuário que realizou o exercício, todas as submissões, os erros cometidos, o tempo demorado para obter sucesso no exercício.

## **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 sucesso, 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, para ocorrer essa atividade o usuário irá precisar acionar a opção de preenchimento do diário.

Após o usuário ter realizado um exercício e obter sucesso, o sistema perguntará se o usuário deseja realizar a escrita de um diário para o respectivo exercício, caso aceite, o sistema redirecionará o usuário para a página do diário onde o usuário informará os dados referentes as submissões dos exercícios.

## Boca

Para realizar a compilação dos exercícios no sistema de dicas, será utilizado funções do *software* BOCA 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).

## 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 validação 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 realizaremos 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. Cada participante terá que realizar todos os exercícios cadastrados na base de dados e cadastrar dicas para cada um.

# 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.

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>.