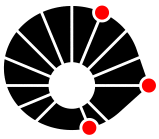
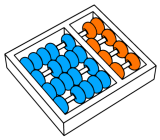




Aprimoramento de cursos de CS1 utilizando o Computer Science Concept Inventory e Learning Analytics

G. Feitosa *R. Caceffo*

Relatório Técnico - IC-PFG-19-39
Projeto Final de Graduação
2019 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Aprimoramento de cursos de CS1 utilizando o Computer Science Concept Inventory e Learning Analytics

Gabriel Feitosa Ricardo Edgard Caceffo

2019-12-07

Resumo

Concept Inventory (CI), é um repositório de questões de múltipla escolha que pode ser usado para avaliar a compreensão dos estudantes em determinado tópico, bem como ser aplicado no início e final de um curso para medir a curva de aprendizagem dos estudantes. Este é um relatório final de graduação de um projeto de aprimoramento do sistema *online* do *Computer Science Concept Inventory (CSCI)*, um conjunto de CIs voltado para cursos introdutórios de programação (CS1, cuja equivalência seria MC102 no contexto do IC/Unicamp) nas linguagens C, Python e Java. Em trabalhos anteriores foi implementado um sistema de geração de relatórios de desempenho dos alunos no CSCI para linguagem C. Neste trabalho realizou-se um *refactoring* da parte visual desse sistema e de parte de seu código, tornando-o mais próximo à expectativa dos usuários. Foram adicionadas ao sistema informações sobre as *misconceptions* identificadas por trabalhos prévios de documentação de erros frequentes de alunos. O relatório de desempenho nos questionários, chamado *Misconception Report*, detalha em diversas granularidades os setores do conhecimento básico de programação em C onde os alunos mais tiveram dificuldades, auxiliando o docente a ministrar suas aulas de acordo com as necessidades da turma.

1 Introdução

O *Peer Instruction (PI)* é uma metodologia de Aprendizado Ativo, elaborada para o contexto das aulas de ensino de física, que prevê a intercalação de exercícios práticos (usualmente em forma de questões de múltipla escolha) durante as aulas teóricas. Caso os alunos atinjam um desempenho inferior ao esperado (75% de respostas corretas) o instrutor solicita que os alunos discutam o problema entre si e submetam novamente as respostas. A tecnologia utilizada para suportar o PI usualmente envolve dispositivos móveis (e.g. smartphones) ou clickers, pequenos aparelhos similares a controles remotos.

Uma das metodologias adotadas em conjunto com o PI é o *Concept Inventory (CI)*, um repositório de questões de múltipla escolha que pode ser usado para avaliar a compreensão dos estudantes em determinado tópico em dado momento do curso, bem como serem aplicados no início e final do curso para medir a curva de aprendizagem dos estudantes [7]. Um CI usualmente leva em conta os problemas de compreensão dos alunos (*misconceptions*) mais comuns que os estudantes possuem em determinado tópico, permitindo ao professor

identificar problemas de aprendizagem. As alternativas disponíveis em cada questão, além da alternativa correta, são denominadas distratores (*distractors*). Os distractors devem ser cuidadosamente escolhidos e definidos para mapear de forma precisa os principais e mais comuns misconceptions dos alunos, reproduzindo as mais diversas (e incorretas) linhas de raciocínio que os estudantes podem ter ao resolver as questões [1].

Os CIs podem ser aplicados como questionários pré e pós-curso, avaliando assim o ganho de aprendizado dos alunos durante a disciplina. Essa abordagem pode ser utilizada para avaliar o impacto de metodologias educacionais de áreas como Ubiquitous Learning [13, 14], Ubiquitous Computing [15] e Active Learning [16] e em relação a este último, a técnicas como Aprendizado Cooperativo e Colaborativo [17], Aprendizado Baseado em Problemas (PBL) [18], Peer Instruction [19, 20] e Just in Time Teaching [21].

Métodos de ensino ativo, como o PI, são comprovadamente mais eficientes que o ensino tradicional, e o curso de graduação da Unicamp peca em não utilizá-las em mais disciplinas. Assim, o projeto nasce com o intuito de incentivar, sistematizar e analisar os impactos da aplicação do CI pré e pós-curso no contexto de disciplinas introdutórias de programação (MC102) ministradas numa versão adaptada da metodologia de *Peer Instruction* para Ciência da Computação (CSPI) [3, 22]

Os CIs desenvolvidos no contexto do projeto, para disciplinas introdutórias de programação (CS1), são chamados de CSCI (Computer Science Concept Inventory). Até o momento o CSCI possui versões nas linguagens C [5, 4, 2, 1] e Python [9], sendo que o CI em Java [10, 11, 12] está em elaboração. Com o intuito de tornar mais simples e efetiva a administração do CSCI tanto na Unicamp como em outras instituições, foi desenvolvido um sistema de gerenciamento de questionários [8], que permite o cadastro, *login*, envio de questionários e visualização de *reports* das respostas (disponível em `http://edu.ic.unicamp.br/limesurvey/`).

Especificamente, este Projeto Final de Graduação apresenta melhorias na implementação desse sistema, focando-se tanto em sua usabilidade como em novas funcionalidades, com destaque para o módulo de *Misconception Report* de dados elaborado para a versão do CSCI na linguagem C.

2 Metodologia

O planejamento de execução do projeto previa 4 etapas:

1. **Estudo de material bibliográfico**

   Para adequadamente realizar as mudanças no sistema, era necessário colocar-se a par do aparato teórico e das terminologias que regiam o projeto. Considerável tempo foi alocado nesta etapa no planejamento, pois o escopo inicial deste trabalho era ampliado, e envolvia classificar o *Concept Inventory* em termos de *taxonomia de Bloom*, um outro aparato teórico que acabou sendo despriorizado em prol de uma melhor implementação das ferramentas de *learning analytics*.

2. **Compreensão do histórico do projeto e do sistema vigente**

O sistema online de suporte ao aprendizado consistia de uma implementação adaptada do *Lime Survey*, de acordo com o especificado em [8]. Assim, foi crucial entender a arquitetura da solução, a modelagem do banco de dados e fazer o *setup* do ambiente de trabalho de forma que minhas alterações nunca comprometessem o funcionamento do site em produção.

O maior desafio seria operar sobre o sistema já construído fazendo o mínimo de alterações possíveis, objetivo este que tornaria-se difícil devido a um excesso de arquivos "mortos"na pasta do projeto e, também, devido a estranha esquematização de banco de dados que o software *Lime Survey* utiliza por padrão, não sendo possível alterá-la.

3. **Implementação do *Misconception Report***

   Esta etapa pretendia criar uma versão "completa"em termos de dados que seriam apresentados ao usuário do *Misconception Report*. Pretendia-se, portanto, extrair do banco de dados do *Lime Survey* os dados necessários para processar, definir objetos PHP para manipulá-los com facilidade e definir um esqueleto de interface para apresentar esses objetos. Sem dúvida, seria a etapa mais custosa do projeto em termos de esforço.

4. **Iteração sobre usabilidade do *Misconception Report*, aprimoramentos front-end e refactoring de código**

   O principal requisito não-funcional era o de usabilidade extremamente intuitiva e simples, pois os professores que utilizarão o sistema provavelmente não terão familiaridade com as teorias e termos do projeto e de CSPI. Assim, uma vez que na etapa anterior já havia sido realizada a coleta dos dados, restaria o trabalho de iterar no design da página para cumprir o requisito não-funcional chave. Testar várias versões e pegar feedback sobre elas seria uma tarefa não tão custosa em termos de esforço, mas a necessidade de interface com o professor-orientador tornaria esta etapa um avanço progressivo ao longo do semestre.

   De forma concorrente e complementar à estas iterações, a adaptação do sistema para paradigmas mais modernos de front-end possibilitaria uma mais rápida iteração de usabilidade e possivelmente uma customização visual posterior. Um imperativo para que esta adaptação front-end obtesse êxito seria um refactoring do código em geral, organizando-o e removendo trechos repetidos, módulos não-utilizados, corrigindo identação e comentando-o.

3 Resultados

3.1 *Misconception Report*: Relatório de Desempenho

Para melhor compreender o *Misconception Report*, convém explicitar os 3 segmentos que o compõem:

1. O cabeçalho contém informações sobre a quantidade de questionários preenchidos e como acurácia geral das questões que não foram deixadas em branco (Figura 2).

2. O *breakdown* de misconceptions por grupo de questões deve ser o foco dos professores, pois consegue de forma sucinta identificar os pontos fracos da turma e quais misconceptions devem ser clarificadas aos alunos durante as aulas. Cada misconception aqui listada é um link. A Figura 3 ilustra o repositório, sendo que a Figura 4 as explica.

3. O detalhamento de questões separa as questões por grupo. Após clicar no grupo desejado, pode-se ver questão a questão o desempenho dos alunos em termos de acertos e erros (Figura 5). Ao clicar no número de erros, rastreia-se em quais *distractors* os alunos caíram, mapeando-os em misconceptions (Figura 6).

Para fins de documentação e possiveis futuros aprimoramentos, explico o fluxo de código *post-refactoring* do arquivo *sign_in_script.php* que gera o relatório:

- Há um loop que itera sobre todos os questionários da pagina. No loop, há um fluxo de condicionais que define qual linguagem está tratando-se no *survey*. Para este projeto, foi implementado o relatório para a linguagem C.

- Realiza-se apenas uma *query* SQL para manter o sistema ágil e evitar interagir com o modelo de banco de dados confuso do *Lime Survey*. Todo o resultado desta *query* é armazenado em objetos PHP. A variável relevante para isto é a *$relatorio* (Figura 9) que encapsula todo o resultado das *queries* SQL em um único objeto contendo as respostas do questionário.

- Após a obtenção do objeto que codifica as respostas da *survey*, o relatório divide-se em blocos responsáveis por gerar os 3 segmentos do *Misconception Report* (Figura 10).

- A variável chave para gerar o relatório é a *$report*, uma string que conterá o HTML do relatório. Além desta, há alguns objetos que podem ou não conter *hard-numbers* que foram inseridos no código e só se aplicam para gerar o relatório em C. Estes *hard-numbers* estão indicados por comentários no código (Figura 11).

Por fim, explicito que os relatórios estão totalmente carregados nas páginas e estão apenas ocultos. Os botões não geram os relatórios em tempo real, mas apenas mostram-os usando estilos em CSS (Figura 12) e uma função Javascript *client-side* (Figura 13).

3.2 Aprimoramentos Visuais e Refactoring

Alguns aprimoramentos foram realizados com o intuito de tornar o sistema mais palatável aos usuários (Figuras 1 e 7), que acessam a internet via *browsers, tablets* e *smartphones*, em diversas resoluções e com interfaces de usuário diferentes. Além disso, o sistema possuia seu *front-end* de forma estática em *Hypertext Markup Language (HTML)*, sem utilizar *Cascading Style Sheets (CSS)*, tornando alterações visuais custosas de serem realizadas. Assim, as seguintes atividades foram realizadas:

1. **Refactoring do código para utilizar CSS e tornar trivial realizar alterações visuais**

Objetivou-se, além de melhorar a interface do sistema, tornar mais prática suas alterações posteriores. Cada grupo de elementos visuais no site teve seu estilo definido em classes CSS (Figura 14), e não em suas próprias *tags* de HTML. Com isto, uma única alteração de estilo no arquivo *main.css*, que pode ser encontrado na pasta raíz do projeto, muda visualmente o site inteiro sem necessidade de alterar individualmente as tags HTML. Essa implementação tornou factível realizar augmentações como a de responsividade e, futuramente, com o crescimento do sistema, facilita a migração para algum *framework* moderno de front-end.

2. **Implementação de responsividade no sistema**

   Objetivou-se tornar o sistema responsivo ao acesso em diversas resoluções de tela comuns no mundo moderno. Em vez da utilização de tabelas HTML e de estilização individual dos elementos, o arquivo CSS dita que as páginas serão flexíveis e utilizarão resoluções adaptáveis ao tamanho da tela do usuário.

3. **Refactoring de formulários para adequar-se aos *templates* modernos de cadastro**

   Objetivou-se melhorar a usabilidade do sistema, permitindo que navegadores modernos usem funcionalidades de *autocomplete* (Figura 8), inferindo o que significa cada campo. Além disso, a tela foi despoluída de elementos distratores.

4. **Inclusão no sistema de páginas explicativas para cada misconception do CSCI**

   O projeto, como um todo, carecia de um repositório de fácil acesso para as misconceptions que não fosse um artigo publicado, acessível e navegável apenas como PDF. Isso dificultaria professores e alunos que desejassem entender pontos de desempenho críticos apontados no *Misconception Report*. Assim, foram criadas páginas (Figura 4) contendo todas as misconceptions identificadas para linguagem C, separadas por grupo e inseridas em um *template* padrão.

4 Conclusão

O sistema foi satisfatoriamente desenvolvido e os resultados listados previamente devem tornar melhor a experiência dos usuários (professores de cursos de CS1), visto que poderão analisar o desempenho de sua turma em termos do aparato teórico suportado pelo *Concept Inventory*. As misconceptions identificadas podem ser facilmente explanadas no próprio sistema e, possivelmente, até utilizadas como fonte para preparo de aulas e uma assertiva explicação de erros aos alunos.

Não houve tempo hábil para realizar todas as implementações desejadas pois a complexidade inicial do sistema era ampla. Agora, com a *curiosa* arquitetura do *Lime Survey* domada, informações podem ser manipuladas em objetos PHP e isso deve facilitar a atividades futuras. Além disto, o *refactoring* do código permitiu organizá-lo deixando já lacunas (Figura 9) e comentários (Figura 10) guiando o programador que desejar realizar alterações futuras. Para dar continuidade ao trabalho aqui executado, sugiro:

- **Adequação do sistema para CIs Python e Java**

  Estas linguagens extremamente comuns em cursos de CS1 são também relevantes em disciplinas mais avançadas, bem como no mercado de trabalho. Há um aparato teórico [9, 10, 11, 12] que suporte essa expansão do sistema, e o código foi preparado de forma que está sinalizado o local onde alterações são necessarias para incluir novas linguagens. O trabalho deve centrar-se no *Misconception Report* e nas páginas de repositório de misconceptions.

- **Alterações visuais a gosto do *product owner* e usuários finais**

  Com o front-end do site devidamente compartimentabilizado e classificado, o CSS pode ser customizado para melhor encaixar-se no padrão esperado. Visto que este não era o foco do projeto, não foi priorizado realizar definições estéticas que não impactassem na usabilidade. Por exemplo, seria possível definir estilos diferentes para cada tipo de botão no sistema, pois as classes estão bem categorizadas e unificadas no arquivo *main.css* como mostra (Figura 14). Deve ser realizado um trabalho de entender o usuário final, principalmente professores praticantes de CSPI, e o sistema precisa adequar-se às estas expectativas.

Referências

[1] CACEFFO,R.; WOLFMAN,S.; BOOTH, K. *Developing a Computer Science Concept Inventory for Introductory Programming.* In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16). ACM, New York, NY, USA, 364-369 (2016).

[2] CACEFFO, R. FRANÇA, B.; GAMA, G.; BENATTI, R.; APARECIDA, T.; CALDAS, T.; AZEVEDO, R. *An Antipattern Documentation about Misconceptions related to an Introductory Programming Course in C.* In Technical Report 17-15, Institute of Computing, University of Campinas, SP, Brasil. 42 pages. (October 2017).

[3] CACEFFO, R.; GAMA, G.; AZEVEDO, R. *Exploring Active Learning Approaches to Computer Science Classes.* In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18). ACM, New York, NY, USA, 922-927 (2018).

[4] CACEFFO, R.; GAMA, G.; BENATTI, R.; APARECIDA, T.; CALDAS, T.; AZEVEDO, R. *A Concept Inventory for CS1 Introductory Programming Courses in C.* In Technical Report 18-06, Institute of Computing, University of Campinas, SP, Brasil. 107 pages. (March 2018)

[5] CACEFFO, R.; WOLFMAN, S.; BOOTH, K.; GAMA, G.; GARCIA, I.; CALDAS, T.; AZEVEDO, R. *An exploratory questionnaire to support the identification and assessment of misconceptions in CS1 courses based on C programming language.* In Technical Report 18-16, Institute of Computing, University of Campinas, SP, Brasil. 41 pages. (October 2018).

[6] BLOOM, T.; LUXTON-REILLY, A.; WHALLEY, J.; HU, M.; ROBBINS, P. *Bloom's taxonomy for CS assessment*. In Proceedings of the tenth conference on Australasian computing education - Volume 78 (ACE '08), Simon Hamilton and Margaret Hamilton (Eds.), Vol. 78. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 155-161 (2008)

[7] HAKE, R. *Interactive-Engagement Versus Traditional Methods: A Six-Thousand-Student Survey of Mechanics Test Data for Introductory Physics Courses*. Am. J. Phys. 66, 1 (January 1998), 64–74. (1998)

[8] BENATTI, R.; CACEFFO, R.; AZEVEDO, R. *Criação de uma Ferramenta Web para Gerenciamento de Inventários Conceituais no Contexto de Cursos Introdutórios de Programação (CS1)*. In Technical Report 18-17, Institute of Computing, University of Campinas, SP, Brasil. 42 pages. (November, 2018).

[9] GAMA, G; CACEFFO, R.; SOUZA, R.; BENATTI, R.; APARECIDA, T.; GARCIA, I.; AZEVEDO, R. *An Antipattern Documentation about Misconceptions related to an Introductory Programming Course in Python*. In Technical Report 18-19, Institute of Computing, University of Campinas, SP, Brasil. 106 pages. (November, 2018).

[10] SOUZA, R.; CACEFFO, R.; FRANK-BOLTON, P; AZEVEDO, R.*An Antipattern Documentation about Possible Misconceptions related to Introductory Programming Courses (CS1) in Java*. In Technical Report 18-20, Institute of Computing, University of Campinas, SP, Brasil. 42 pages. (December, 2018).

[11] CACEFFO, R.; Frank-Bolton, P.; Souza, R.; Azevedo, R. *Identifying and Validating Java Misconceptions – Complementary Material*. In Technical Report 19-05, Institute of Computing, University of Campinas, SP, Brasil. 49 pages. (April, 2019).

[12] CACEFFO, R.; Frank-Bolton, P.; Souza, R.; Azevedo, *Identifying and Validating Java Misconceptions Toward a CS1 Concept Inventory*. In Innovation and Technology in Computer Science Education (ITiCSE '19), Aberdeen, Scotland Uk. ACM, New York, NY, USA, 7 pages (July, 2019)

[13] CACEFFO, R.; ROCHA, H. *Ubiquitous Classroom Response System: An Innovative Approach to Support the Active Learning Model*. In Ubiquitous Learning: An International Journal, v. 3, p. 43-55. (2011)

[14] CACEFFO, R.; ROCHA, H. *Design and Model of a Ubiquitous Classroom Response System Through Context Factors*. In Ubiquitous Learning an International Journal, v. 4, p. 61-77. (2012)

[15] CACEFFO, R.; AZEVEDO, R. *LSQuiz: A Collaborative Classroom Response System to Support Active Learning through Ubiquitous Computing*. In 11th International Conference on Cognition and Exploratory Learning in Digital Age (CELDA 2014), Porto, Portugal, October 2014. pp 63-71. (2014).

[16] CACEFFO, R.; ROCHA, H.; AZEVEDO, R. *A Ubiquitous, Pen-Based and Touch Classroom Response System Supported by Learning Styles. In The Impact of Pen and Touch Technology on Education*. Hammond, T.; Valentine, S.; Adler, A.; Payton, M. (Eds). Human-Computer Interaction Series. pp 373-381. ISBN 878-3-319-15593-7. ISBN 978-3-319-15594-4 (eBook). ISSN 1571-5035. Springer International Publishing. Switzerland. (2015).

[17] SCHNEIDER, B. *Designing Tabletop Activities for Inquiry-based Learning: Lessons from Phylogenetics, Neuroscience and Logistics*. In Proc. 2012 ACM Intl. Conf. on Interactive Tabletops and Surfaces (ITS '12). ACM, New York, USA, 289–294. (2012).

[18] MIAO, Y.; Wang, D. et al. *Towards a Web-Based Adaptive Problem-Based Learning Application*. In Proc. 2015 IEEE 15th Intl. Conf.on Advanced Learning Technologies (ICALT '15). IEEE Computer Society,Washington, DC, USA, 44–48. (2015).

[19] ZINGARO, D.; PORTER, L. *Peer Instruction in computing: The value of instructor intervention*. Computers & Education 71 (2014), 87 – 96. (2014).

[20] CROUCH, C.; MAZUR, E.; *Peer Instruction: Ten years of experience and results*. In American Journal of Physics 69, 9, 970-977 (2001).

[21] NOVAK, G.M.; *Just-in-time Teaching: Blending Active Learning with Web Technology*. Prentice Hall. (2011).

[22] CACEFFO, R..; GAMA, G.; MOREIRA, M.; GARCIA, I.; AZEVEDO, R. *Usability and Reliability Data Relative to the Use of Clickers to Support the Computer Science Peer Instruction (CSPI) Approach.* In Technical Report 18-08, Institute of Computing, University of Campinas, SP, Brasil. 40 pages. June, 2018 (2018)

ANEXO I

Neste anexo são apresentados os detalhes do processo de implementação e *refactoring* descritos no texto. A menos que explicitado o contrário, o arquivo atrelado com a informação da imagem é o *sign_in_script.php*.

1. A Figura 1 apresenta a tela inicial do sistema, após o login. A tabela cinza se repetirá para cada *survey* da conta logada.

2. A Figura 2 mostra uma visão geral do *Misconception Report*, mostrando seus 3 segmentos: cabeçalho, misconceptions por grupo e detalhamento de questões.

3. A Figura 3 mostra o segundo segmento do *Misconception Report*, com um retângulo vermelho indicando duas misconceptions com alta ocorrência. O professor usuário poderá clicar nos links "A4"e "A5"para ser direcionado ao repositório que detalha o que significam aquelas misconceptions.

4. A Figura 4 mostra o repositório de misconceptions citado anteriormente. Cada grupo de misconceptions possui um arquivo HTML. O grupo A mostrado na imagem encontra-se em *misconceptions/A.php*.

5. A Figura 5 mostra o terceiro segmento do *Misconception Report* expandido para mostrar questões do grupo A, QA1 até QA5. Foi tomada uma decisão de UX de esconder a complexidade do relatório inicialmente, mas deixando dicas que era possível entrar em mais detalhes. Para isso, as linhas com as misconceptions aparecem na primeira coluna e o botão de incorretos está destacado em vermelho, incentivando o usuário a investigar e clicá-lo, gerando a próxima imagem.

6. A Figura 6 mostra o que ocorre ao clicar nos botões vermelhos da figura anterior: Os erros de cada questão são mapeados em misconceptions. No exemplo, dos 3 erros da questão QA2, 1 foi a misconception A3 e 2 foram a misconception A6.

7. A Figura 7 mostra a *landing page* do sistema. Cada objeto está "conteinerizado"em CSS, sendo trivial fazer uma alteração estética no sistema de forma controlada. O arquivo atrelado é o *main.html* e o *main.css*.

8. A Figura 8 mostra a nova tela de cadastro, incluindo um menu de radio para possíveis expansões do sistema para incluir CIs de outras linguagens. O arquivo atrelado é o *sign_up_script.php*.

9. As figuras 9, 10, 11, 12 e 13, mostram o *sign_in_script.php* com alguns *highlights* coloridos, melhor explicados em suas legendas.

10. Por último, a figura 14 mostra como está estruturado o CSS da página, consolidado integralmente no arquivo *main.css*. Cada *container* da interface está mapeado em uma classe neste arquivo, sendo trivial alterar, por exemplo apenas o botão do terceiro segmento do *Misconception Report*.

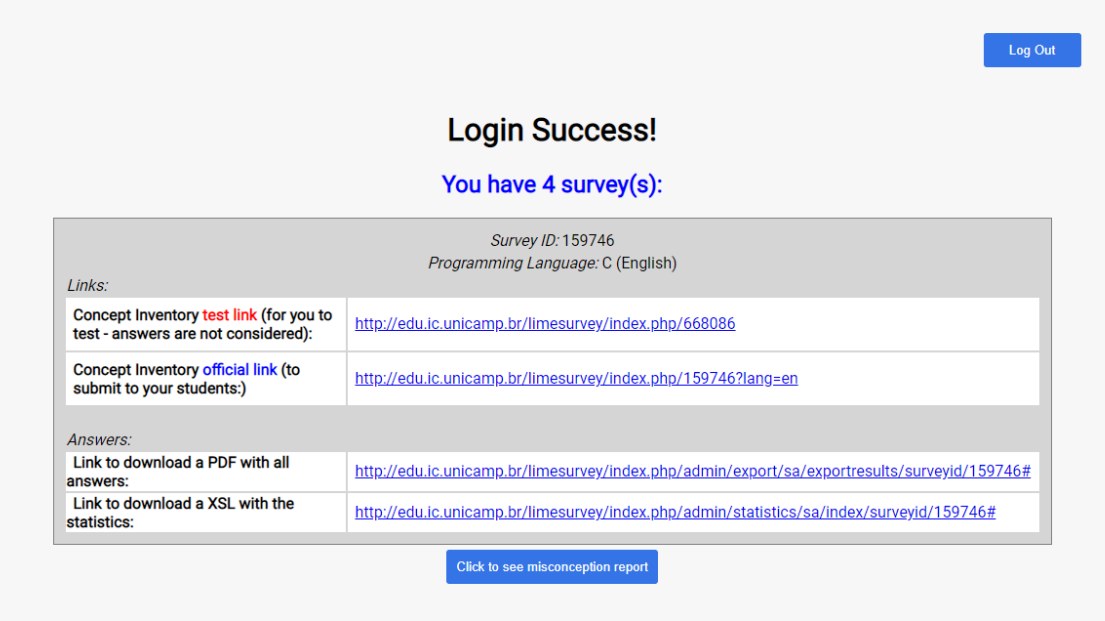


Figura 1: Nova interface de survey

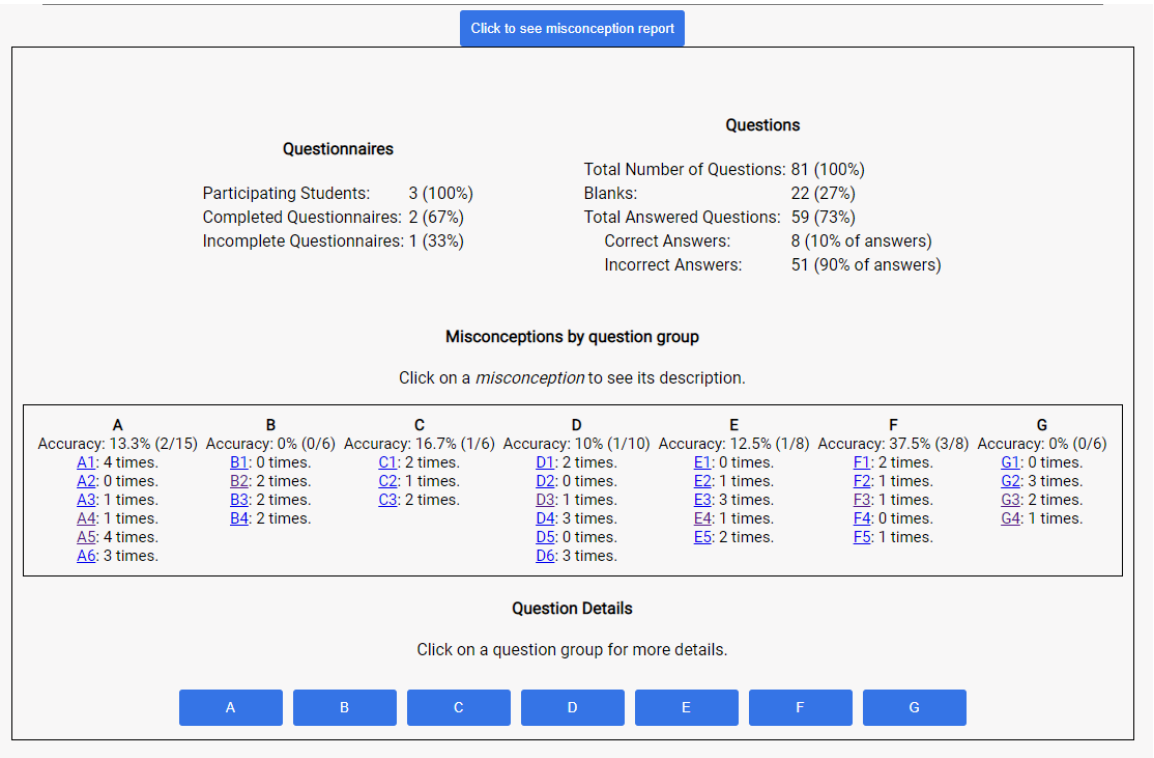


Figura 2: Misconception Report

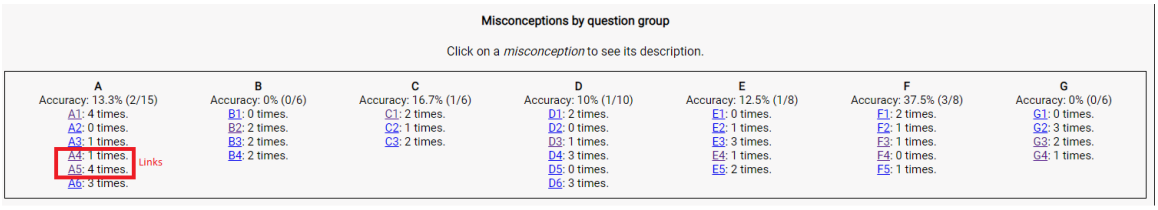
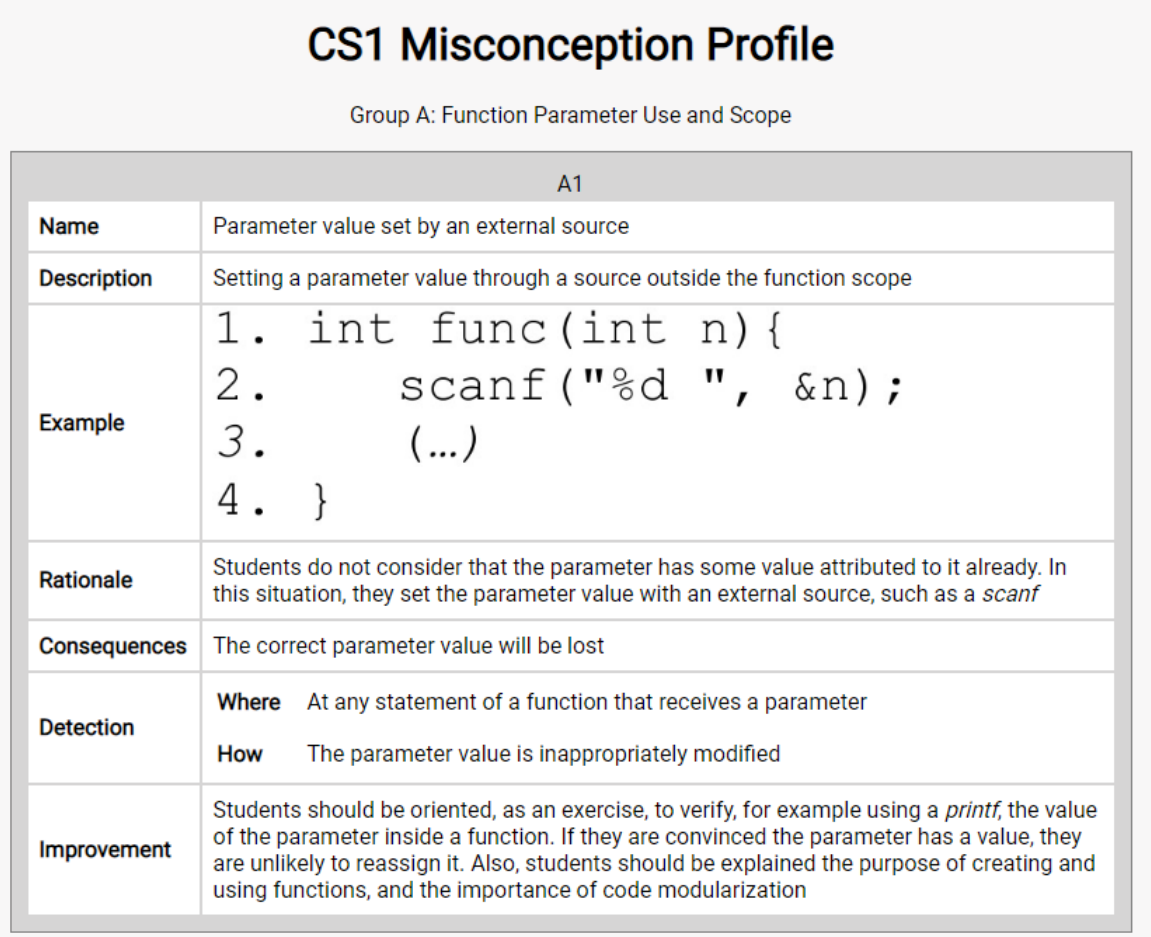


Figura 3: Detalhamento de misconceptions e links para repositório explicativo



CS1 Misconception Profile

Group A: Function Parameter Use and Scope

A1	
Name	Parameter value set by an external source
Description	Setting a parameter value through a source outside the function scope
Example	1. int func(int n){ 2. scanf("%d ", &n); 3. (...) 4. }
Rationale	Students do not consider that the parameter has some value attributed to it already. In this situation, they set the parameter value with an external source, such as a *scanf*
Consequences	The correct parameter value will be lost
Detection	**Where** At any statement of a function that receives a parameter **How** The parameter value is inappropriately modified
Improvement	Students should be oriented, as an exercise, to verify, for example using a *printf*, the value of the parameter inside a function. If they are convinced the parameter has a value, they are unlikely to reassign it. Also, students should be explained the purpose of creating and using functions, and the importance of code modularization

Figura 4: Visão do repositório de misconceptions, categorizadas por grupos de conteúdo

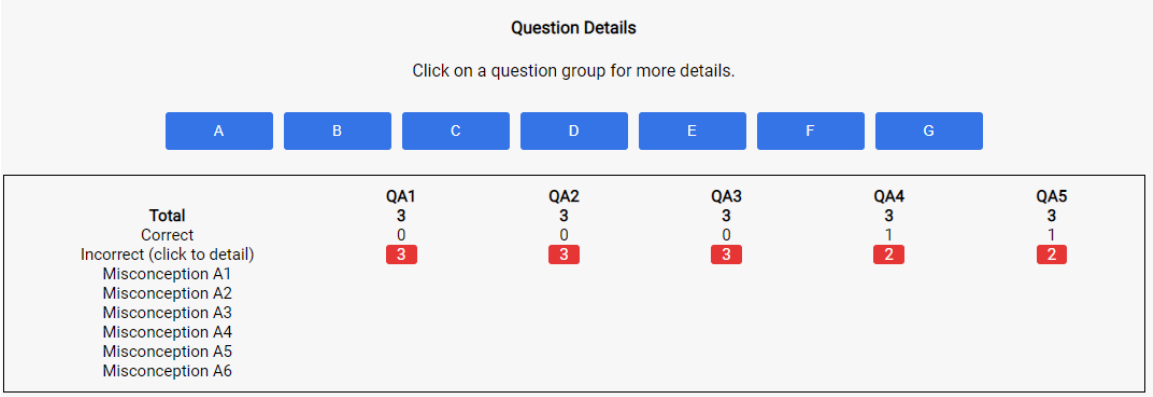


Figura 5: Detalhamento de Questões

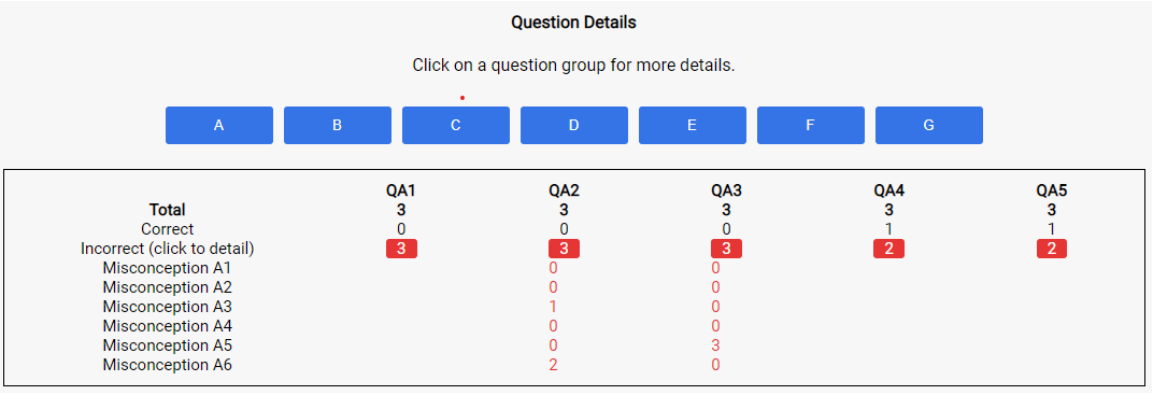


Figura 6: Detalhamento de Questões - Rastreio de Misconceptions

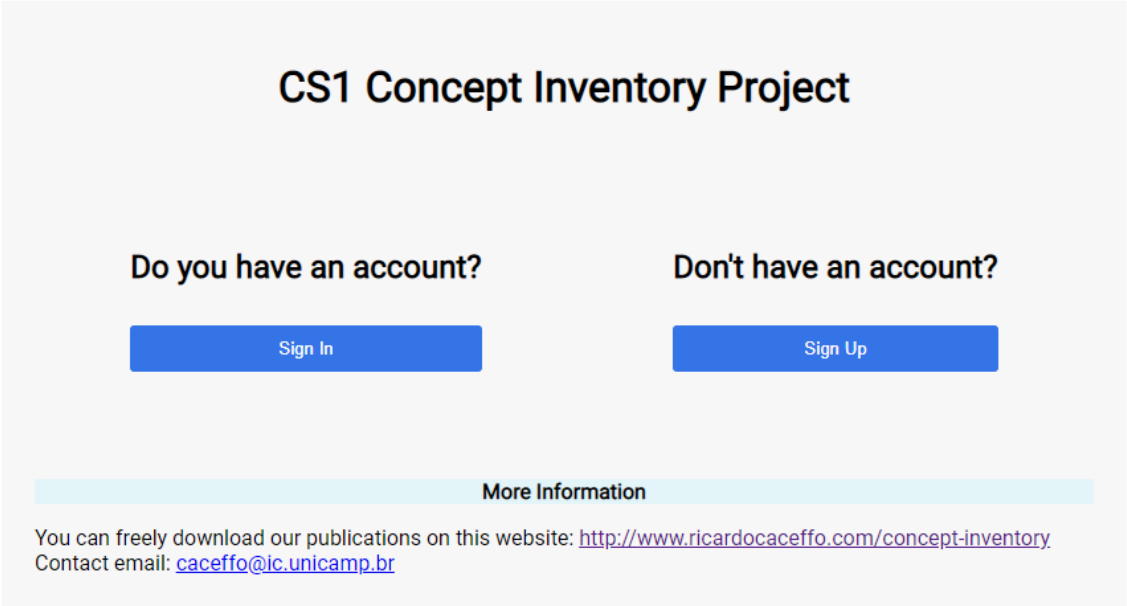


Figura 7: Nova landing page

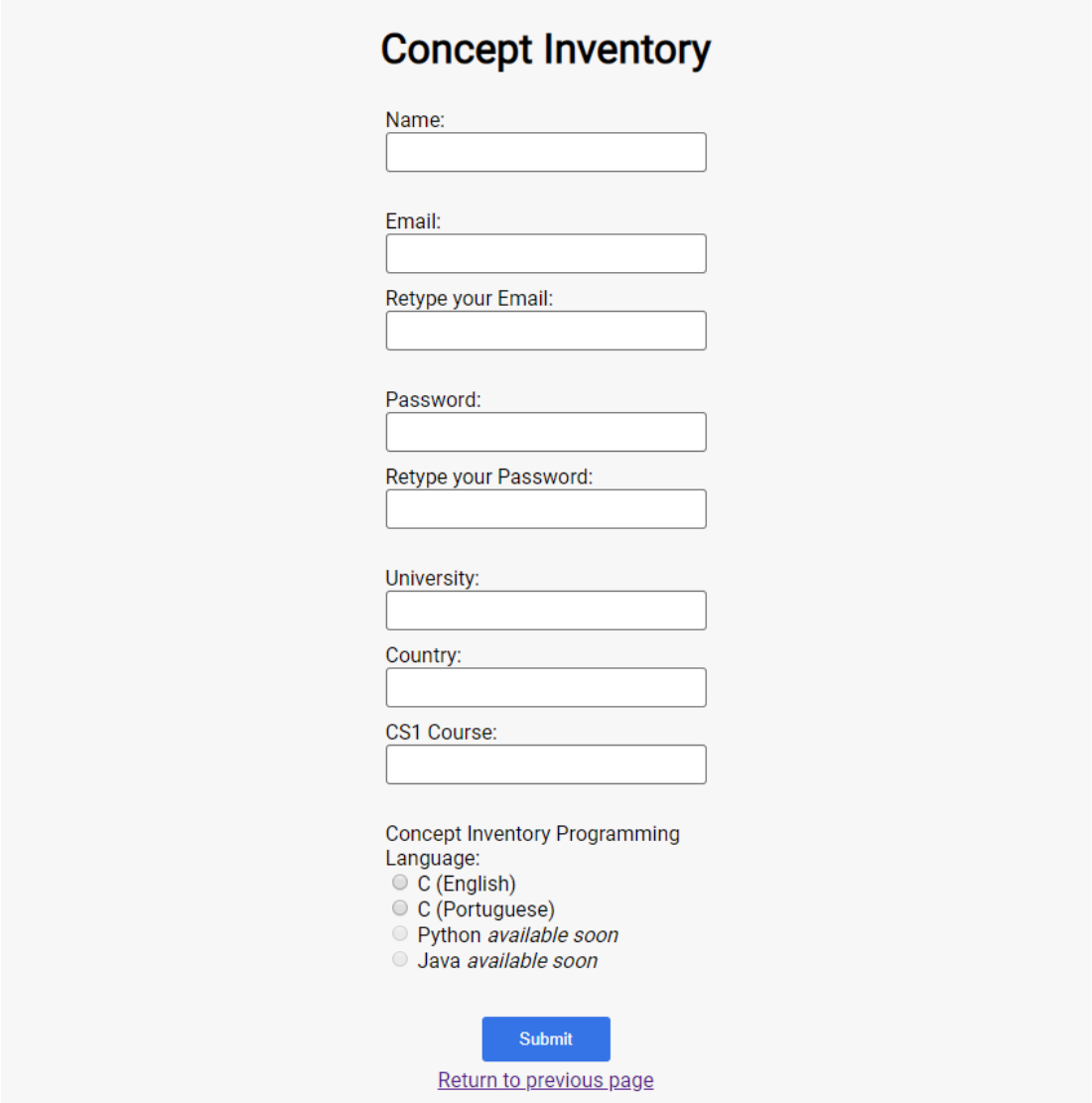
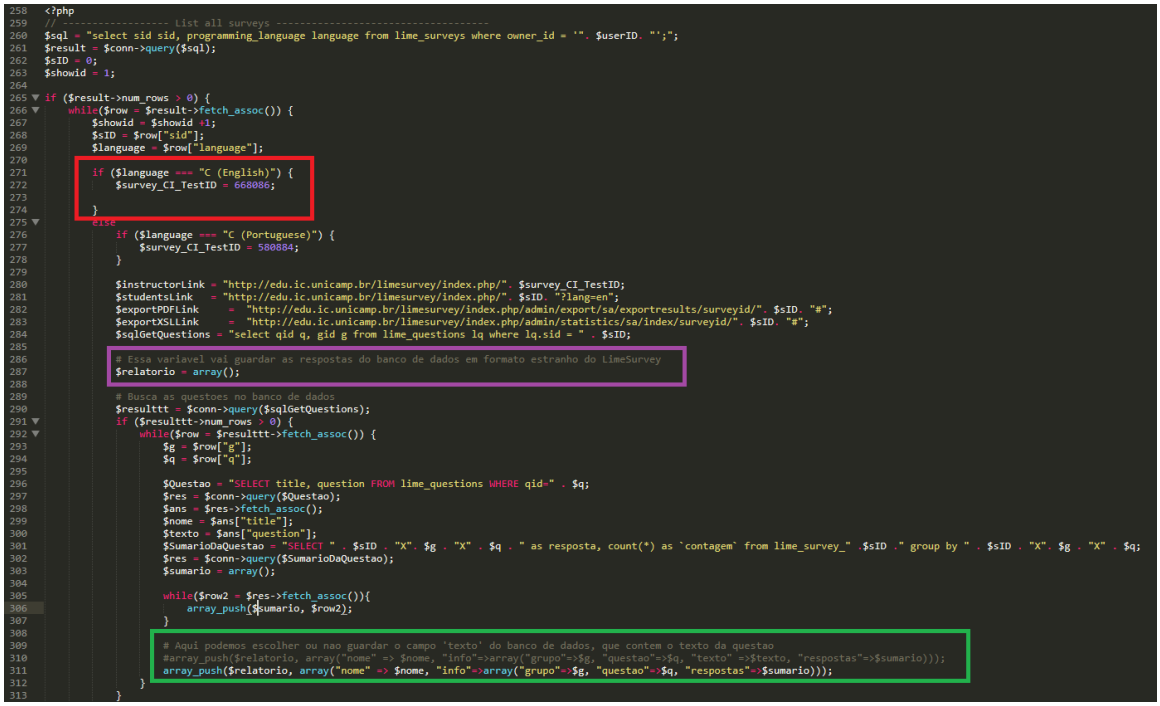
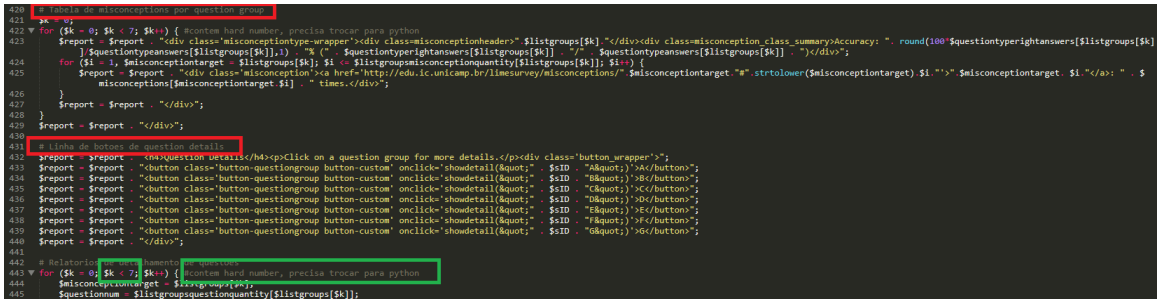


Figura 8: Padrão de novos formulários



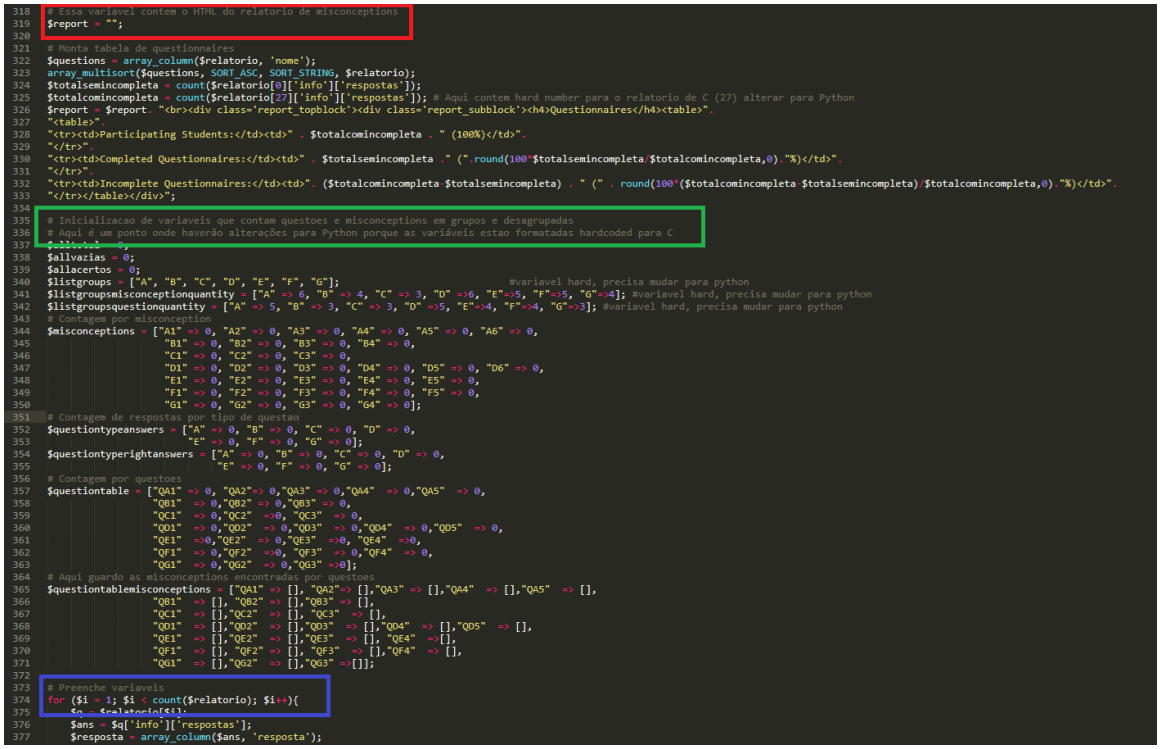
```php
<?php
// ---------------- List all surveys ----------------------------------
$sql = "select sid sid, programming_language language from lime_surveys where owner_id = '". $userID. "';";
$result = $conn->query($sql);
$sID = 0;
$showid = 1;

if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        $showid = $showid +1;
        $sID = $row["sid"];
        $language = $row["language"];

        if ($language === "C (English)") {
            $survey_CI_TestID = 668086;


        }
        else
            if ($language === "C (Portuguese)") {
                $survey_CI_TestID = 580884;
            }

            $instructorLink = "http://edu.ic.unicamp.br/limesurvey/index.php/". $survey_CI_TestID;
            $studentsLink   = "http://edu.ic.unicamp.br/limesurvey/index.php/". $sID. "?lang=en";
            $exportPDFLink     =  "http://edu.ic.unicamp.br/limesurvey/index.php/admin/export/sa/exportresults/surveyid/". $sID. "#";
            $exportXSLLink     =  "http://edu.ic.unicamp.br/limesurvey/index.php/admin/statistics/sa/index/surveyid/". $sID. "#";
            $sqlGetQuestions = "select qid q, gid g from lime_questions lq where lq.sid = " . $sID;

            # Essa variavel vai guardar as respostas do banco de dados em formato estranho do LimeSurvey
            $relatorio = array();

            # Busca as questoes no banco de dados
            $resulttt = $conn->query($sqlGetQuestions);
            if ($resulttt->num_rows > 0) {
                while($row = $resulttt->fetch_assoc()) {
                    $g = $row["g"];
                    $q = $row["q"];

                    $Questao = "SELECT title, question FROM lime_questions WHERE qid=" . $q;
                    $res = $conn->query($Questao);
                    $ans = $res->fetch_assoc();
                    $nome = $ans["title"];
                    $texto = $ans["question"];
                    $SumarioDaQuestao = "SELECT " . $sID . "X". $g . "X" . $q . " as resposta, count(*) as `contagem` from lime_survey_" .$sID ." group by " . $sID . "X". $g . "X" . $q;
                    $res = $conn->query($SumarioDaQuestao);
                    $sumario = array();

                    while($row2 = $res->fetch_assoc()){
                        array_push($sumario, $row2);
                    }

                    # Aqui podemos escolher ou nao guardar o campo 'texto' do banco de dados, que contem o texto da questao
                    #array_push($relatorio, array("nome" => $nome, "info"=>array("grupo"=>$g, "questao"=>$q, "texto" =>$texto, "respostas"=>$sumario)));
                    array_push($relatorio, array("nome" => $nome, "info"=>array("grupo"=>$g, "questao"=>$q, "respostas"=>$sumario)));
                }
            }
```

Figura 9: Exemplo de pontos de alteração. Em vermelho, onde adicionar código para gerar o *Misconception Report* em outras linguagens. Em roxo, oa variável *$relatorio*, objeto PHP que guarda as respostas do *survey*. Em verde, um dos pontos de escolha de como tratar variáveis no código. Neste caso, é a decisão de guardar ou não no objeto o texto das questões.



```php
# Tabela de misconceptions por question group
$k = 0;
for ($k = 0; $k < 7; $k++) { #contem hard number, precisa trocar para python
    $report = $report . "<div class='misconceptiontype-wrapper'><div class=misconceptionheader>".$listgroups[$k]."</div><div class=misconception_class_summary>Accuracy: ". round(100*$questiontyperightanswers[$listgroups[$k]
        ]/$questiontypeanswers[$listgroups[$k]],1) . "% (" . $questiontyperightanswers[$listgroups[$k]] . "/" . $questiontypeanswers[$listgroups[$k]] . ")</div>";
    for ($i = 1, $misconceptiontarget = $listgroups[$k]; $i <= $listgroupsmisconceptionquantity[$listgroups[$k]]; $i++) {
        $report = $report . "<div class='misconception'><a href='http://edu.ic.unicamp.br/limesurvey/misconceptions/".$misconceptiontarget."#".strtolower($misconceptiontarget).$i."'>".$misconceptiontarget. $i."</a>: " . $
            misconceptions[$misconceptiontarget.$i] . " times.</div>";
    }
    $report = $report . "</div>";
}
$report = $report . "</div>";

# Linha de botoes de question details
$report = $report . "<h4>Question details</h4><p>Click on a question group for more details.</p><div class='button_wrapper'>";
$report = $report . "<button class='button-questiongroup button-custom' onclick='showdetail(&quot;" . $sID . "A&quot;)'>A</button>";
$report = $report . "<button class='button-questiongroup button-custom' onclick='showdetail(&quot;" . $sID . "B&quot;)'>B</button>";
$report = $report . "<button class='button-questiongroup button-custom' onclick='showdetail(&quot;" . $sID . "C&quot;)'>C</button>";
$report = $report . "<button class='button-questiongroup button-custom' onclick='showdetail(&quot;" . $sID . "D&quot;)'>D</button>";
$report = $report . "<button class='button-questiongroup button-custom' onclick='showdetail(&quot;" . $sID . "E&quot;)'>E</button>";
$report = $report . "<button class='button-questiongroup button-custom' onclick='showdetail(&quot;" . $sID . "F&quot;)'>F</button>";
$report = $report . "<button class='button-questiongroup button-custom' onclick='showdetail(&quot;" . $sID . "G&quot;)'>G</button>";
$report = $report . "</div>";

# Relatorio de detalhamento de questões
for ($k = 0; $k < 7; $k++) { //contem hard number, precisa trocar para python
    $misconceptiontarget = $listgroups[$k];
    $questionnum = $listgroupsquestionquantity[$listgroups[$k]];
```

Figura 10: Em vermelho, exemplo de comentários delimitando análises que geram blocos de interface. Em verde, comentário exemplo de pontos onde há "hard numbers"que precisam ser alterados para implementar outras linguagens



```
# Essa variavel contem o HTML do relatorio de misconceptions
$report = "";

# Monta tabela de questionnaires
$questions = array_column($relatorio, 'nome');
array_multisort($questions, SORT_ASC, SORT_STRING, $relatorio);
$totalsemincompleta = count($relatorio[0]['info']['respostas']);
$totalcomincompleta = count($relatorio[27]['info']['respostas']); # Aqui contem hard number para o relatorio de C (27) alterar para Python
$report = $report. "<br><div class='report_topblock'><div class='report_subblock'><h4>Questionnaires</h4><table>".
"<table>".
"<tr><td>Participating Students:</td><td>" . $totalcomincompleta . " (100%)</td>".
"</tr>"
"<tr><td>Completed Questionnaires:</td><td>" . $totalsemincompleta ." (".round(100*$totalsemincompleta/$totalcomincompleta,0)."%)</td>".
"</tr>".
"<tr><td>Incomplete Questionnaires:</td><td>". ($totalcomincompleta-$totalsemincompleta) . " (" . round(100*($totalcomincompleta-$totalsemincompleta)/$totalcomincompleta,0)."%)</td>".
"</tr></table></div>";

# Inicializacao de variaveis que contam questoes e misconceptions em grupos e desagrupadas
# Aqui é um ponto onde haverão alterações para Python porque as variáveis estao formatadas hardcoded para C
$allvazias = 0;
$allacertos = 0;
$listgroups = ["A", "B", "C", "D", "E", "F", "G"];                              #variavel hard, precisa mudar para python
$listgroupsmisconceptionquantity = ["A" => 6, "B" => 4, "C" => 3, "D" =>6, "E"=>5, "F"=>5, "G"=>4]; #variavel hard, precisa mudar para python
$listgroupsquestionquantity = ["A" => 5, "B" => 3, "C" => 3, "D" =>5, "E"=>4, "F"=>4, "G"=>3]; #variavel hard, precisa mudar para python
# Contagem por misconception
$misconceptions = ["A1" => 0, "A2" => 0, "A3" => 0, "A4" => 0, "A5" => 0, "A6" => 0,
                   "B1" => 0, "B2" => 0, "B3" => 0, "B4" => 0,
                   "C1" => 0, "C2" => 0, "C3" => 0,
                   "D1" => 0, "D2" => 0, "D3" => 0, "D4" => 0, "D5" => 0, "D6" => 0,
                   "E1" => 0, "E2" => 0, "E3" => 0, "E4" => 0, "E5" => 0,
                   "F1" => 0, "F2" => 0, "F3" => 0, "F4" => 0, "F5" => 0,
                   "G1" => 0, "G2" => 0, "G3" => 0, "G4" => 0];
# Contagem de respostas por tipo de questao
$questiontypeanswers = ["A" => 0, "B" => 0, "C" => 0, "D" => 0,
                        "E" => 0, "F" => 0, "G" => 0];
$questiontyperightanswers = ["A" => 0, "B" => 0, "C" => 0, "D" => 0,
                             "E" => 0, "F" => 0, "G" => 0];
# Contagem por questoes
$questiontable = ["QA1" => 0, "QA2"=> 0,"QA3" => 0,"QA4"  => 0,"QA5"  => 0,
                  "QB1"  => 0,"QB2" => 0,"QB3" => 0,
                  "QC1"  => 0,"QC2"  =>0, "QC3"  => 0,
                  "QD1"  => 0,"QD2"  => 0,"QD3"  => 0,"QD4"  => 0,"QD5"  => 0,
                  "QE1"  =>0,"QE2"  => 0,"QE3"  =>0, "QE4"  =>0,
                  "QF1"  => 0,"QF2"  =>0, "QF3"  => 0,"QF4"  => 0,
                  "QG1"  => 0,"QG2"  => 0,"QG3" =>0];
# Aqui guardo as misconceptions encontradas por questoes
$questiontablemisconceptions = ["QA1" => [], "QA2"=> [],"QA3" => [],"QA4"  => [],"QA5"  => [],
                  "QB1"  => [], "QB2" => [],"QB3" => [],
                  "QC1"  => [],"QC2"  => [], "QC3"  => [],
                  "QD1"  => [],"QD2"  => [],"QD3"  => [],"QD4"  => [],"QD5"  => [],
                  "QE1"  => [],"QE2"  => [],"QE3"  => [], "QE4"  =>[],
                  "QF1"  => [], "QF2" => [], "QF3"  => [],"QF4"  => [],
                  "QG1"  => [],"QG2"  => [],"QG3" =>[]];

# Preenche variaveis
for ($i = 1; $i < count($relatorio); $i++){
    $q = $relatorio[$i];
    $ans = $q['info']['respostas'];
    $resposta = array_column($ans, 'resposta');
```

Figura 11: Em vermelho, exemplo de comentário explicando variáveis. A variável destacada é a principal, pois é a *string* que contém o HTML a ser enviado ao usuário. Em verde, comentário explicando variáveis que contém "hard-numbers"e devem ser mudadas para implementar novos CIs. Em roxo, o bloco de código que preenche essas variáveis auxiliares.

```
<div align="center">
    <button align="center"class="button-misconception button-custom" onclick="show(<?php echo $showid?>)"> Click to see misconception report</button>
</div>
<div align="center" class="misconceptionreport" style="display: none;" id = "<?php echo $showid?>"> <?php echo $report ?></div>
<br><br>
```

Figura 12: Os relatórios estão todos carregados na página, constam apenas como o estilo "display: none", este que é alterado com na interação com o botão para "display: flex"usando código JS

```
function showdetail(report) {
    var div = document.getElementById(report.toUpperCase().trim());
    if (div.style.display !== 'none') {
        div.style.display = 'none';
    }
    else {
        div.style.display = 'flex';
    }
}
```

Figura 13: Exemplo de um dos códigos JS que controla o que está visível ao usuário

main.css

```
body {
    font-family: "Roboto", sans-serif;
}

.button-custom,
.button-misconception,
.button-questiongroup {
    padding: 10px;
    min-width: 100px;
    background: #3574e6;
    color: white;
    border-radius: 3px;
    outline: none;
    border: none;
    margin: 5px 0;
    margin-top: 10px;
    cursor: pointer;
}

.report_topblock {
    display: inline-flex;
    margin: 10px;
    padding: 15px;
}
.report_subblock {
    align-self: center;
    margin: 0 50px;
}
.button-misconception {
    margin: auto auto;
}
```

Figura 14: Arquivo main.css, que contém todas as classes usadas no projeto