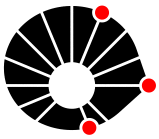
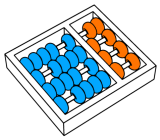




Criação de uma Ferramenta Web para Gerenciamento de Inventários Conceituais no Contexto de Cursos Introdutórios de Programação (CS1)

Raysa Benatti *Ricardo Caceffo* *Rodolfo Azevedo*

Technical Report - IC-18-17 - Relatório Técnico
November - 2018 - Novembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Criação de uma Ferramenta Web para Gerenciamento de Inventários Conceituais no Contexto de Cursos Introdutórios de Programação (CS1)

Raysa Benatti, Ricardo Caceffo, Rodolfo Azevedo

Instituto de Computação Universidade Estadual de Campinas (UNICAMP), Caixa Postal 6176
13083-970 Campinas-SP, Brasil

raysa.benatti@students.ic.unicamp.br; caceffo@ic.unicamp.br, rodolfo@ic.unicamp.br

Resumo. O projeto Inventário Conceitual, desenvolvido por alunos, professores e pesquisadores do Instituto de Computação da Unicamp, tem como proposta inicial a identificação de problemas de compreensão (*misconceptions*) de alunos de cursos introdutórios de programação, como MC102. A partir dessa identificação foram criados Inventários Conceituais (CIs), questionários de múltipla-escolha em que cada alternativa incorreta está mapeada a um desses *misconceptions*. Um dos objetivos do projeto é permitir que professores das mais variadas instituições do mundo possam, de forma automatizada, criar, gerenciar, submeter aos seus alunos, e visualizar relatórios da aplicação de CIs. Deste modo, uma das necessidades do projeto é o uso de uma ferramenta que ao mesmo tempo permita: a) a automatização do processo de criação, gerenciamento e submissão de questionários e; b) a possibilidade de customização dos dados registrados no sistema, como por exemplo a extração de informações pontuais e padrões de resposta dos questionários respondidos. Este Relatório Técnico apresenta a implementação de uma ferramenta web que atende a essas necessidades.

Palavras-chave: inventário conceitual; banco de dados; aprendizado ativo; ferramenta web

1. Introdução

O presente relatório tem por objetivo documentar o processo de criação e as funcionalidades de uma ferramenta web utilizada para gerenciamento das informações do projeto Inventário Conceitual. A organização do documento ocorre do seguinte modo: na Seção 2 é apresentado o **contexto da pesquisa** da pesquisa, onde o projeto Inventário Conceitual é brevemente descrito e os aspectos técnicos do projeto são apresentados em alto nível; na Seção 3 é apresentado o ***frontend*** das principais funcionalidades do *website*; e na Seção 4 é apresentado o ***backend*** das principais funcionalidades do *website*. Ainda, ao final do documento, está disponível um apêndice (Apêndice 1) onde são apresentados os *scripts*, elaborados em PHP, utilizados para implementação das funcionalidades descritas neste relatório.

2. Contexto da Pesquisa

2.1. Projeto Inventário Conceitual

O projeto Inventário Conceitual [1, 2, 10] vem sendo desenvolvido por alunos, professores e pesquisadores do Instituto de Computação da Universidade Estadual de Campinas. A proposta central do grupo é a criação de um Inventário Conceitual (*Concept Inventory*/CI) – um questionário de múltipla escolha onde cada alternativa incorreta está mapeada a um *misconception* – para cursos Introdutórios de Programação (CS1). As etapas já concluídas da pesquisa são:

1. Identificação dos problemas de compreensão dos alunos (*misconceptions*) na linguagem C [1].
2. Criação do Inventário Conceitual em C [2, 10].

O projeto Inventário Conceitual está inserido em uma iniciativa mais ampla de *Computer Science Education* [1, 2, 3, 5], focada no *redesign* de disciplinas introdutórias de programação (CS1), como MC102[1].

Este Relatório Técnico apresenta as atividades do projeto de Iniciação Científica realizado por Raysa Benatti e orientado por Ricardo Caceffo e Rodolfo Azevedo. **A motivação deste projeto** reside no fato de que, para a validação estatística dos CIs criados no projeto, é necessário um número grande de alunos participantes. Assim, é necessária a criação de uma ferramenta online que permita: a) a automatização do processo de criação, gerenciamento e submissão de questionários e; b) a possibilidade de customização dos dados registrados no sistema, como por exemplo a extração de informações pontuais e padrões de resposta dos questionários respondidos. Pretende-se, com isso, utilizar a ferramenta como forma de disseminação do projeto, convidando pesquisadores e instituições nacionais (a começar pela Unicamp, com a disciplina MC102) e internacionais a aplicarem os CIs aos seus alunos.

Este projeto de iniciação científica descreve a criação de uma ferramenta web que atenda a essas necessidades.

2.2. Materiais e Métodos

O *website* foi criado tendo como base a ferramenta de software livre LimeSurvey (versão 2.64.7+170404) [6] para aplicação e gerenciamento dos questionários, integrado ao SGBD MySQL Oracle (versão 5.7.22) [7] para persistência dos dados.

[1]Disponível em: https://www.ic.unicamp.br/~mc102/ Acesso em: setembro de 2018.

O LimeSurvey [6] permite a criação e gerenciamento de questionários (*surveys*) **por um único *manager***, ou seja, o administrador do sistema precisa logar em sua conta no LimeSurvey e criar um questionário, obtendo um *link* a ser enviado a todos os possíveis respondentes.

No projeto Inventário Conceitual, o objetivo é que esse processo ocorra de maneira **transparente**, ou seja, o professor ou pesquisador que acessa o sistema deve receber automaticamente o *link* do CI a ser enviado aos seus alunos, bem como um *link* para visualização dos relatórios. Deste modo, **todo o processo de criação do CI deve ocorrer de maneira automatizada**, no *background* do sistema. Assim, os requisitos do *website* são:

1. Cadastro dos pesquisadores/professores;
2. Login dos usuários e visualização dos CIs associados a cada um deles;
3. Criação automática de um CI, a partir de um *template* já definido no sistema:
    a. Disponibilização do *link* de divulgação do CI;
    b. Disponibilização do *link* de acesso ao relatório do CI.

As Seções 3 e 4, a seguir, apresentam os detalhes de implementação do *website*. Na Seção 3 é apresentado o *frontend*, com detalhes sobre as funcionalidades, telas e exemplos de uso. Por sua vez, na Seção 4 é apresentado o *backend* do website, composto por um **conjunto de *scripts* em PHP** criados para automatização das funcionalidades.

O link para acesso ao website é: http://edu.ic.unicamp.br/limesurvey/

3. Principais funcionalidades do website *(frontend)*

3.1. Cadastro dos pesquisadores/professores

Inicialmente os pesquisadores ou professores interessados em criar e administrar CIs devem acessar a página inicial do projeto, disponível em http://edu.ic.unicamp.br/limesurvey, conforme indicado na Figura 1 a seguir.

Figura 1. Tela inicial do *website "CS1 Concept Inventory Project"*.
Disponível em: http://edu.ic.unicamp.br/limesurvey.

Na tela inicial do *website* (Figura 1), os pesquisadores ou professores interessados em criar e gerenciar CIs devem inicialmente fazer um cadastro. Para tanto, devem clicar na opção *"Don't have an account?"*. Após isso o usuário é direcionado para a tela de cadastro, conforme indicado na Figura 2.

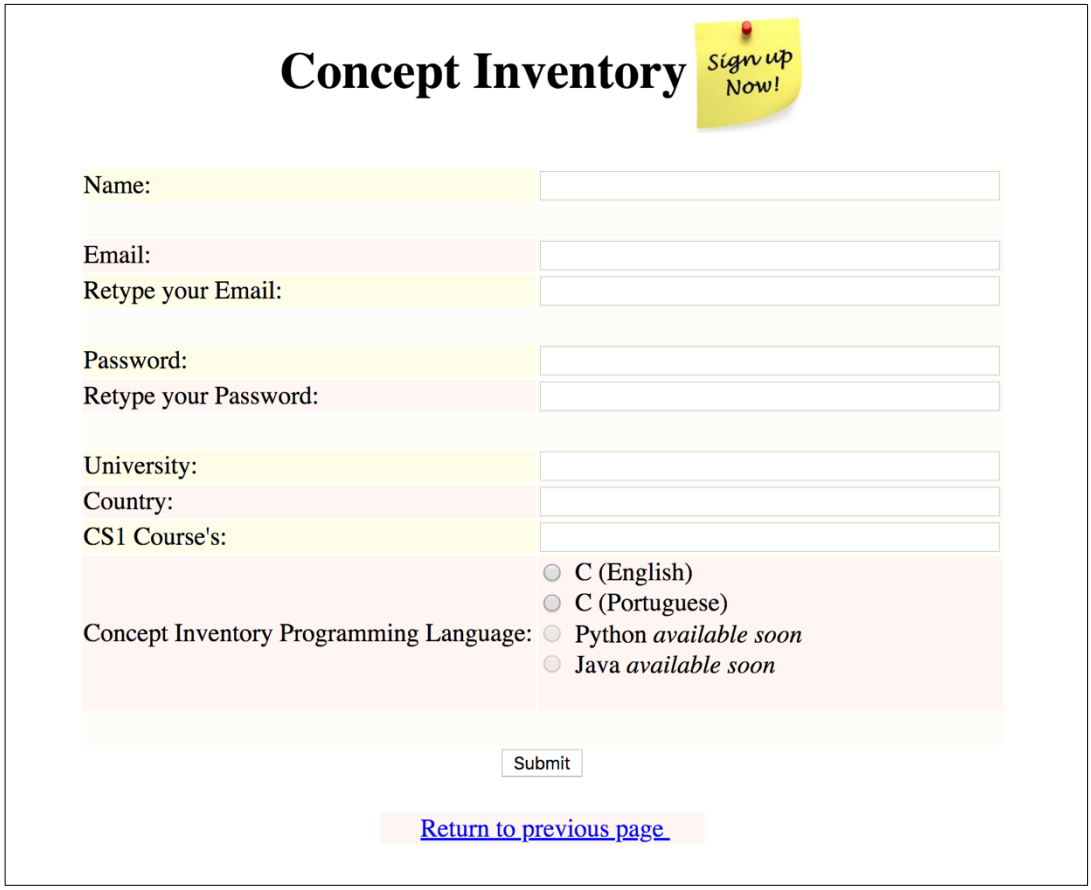


Figura 2. Tela de cadastro do *website*. Nesta tela, o professor ou pesquisador interessado deve informar os seguintes dados: nome, *email*, universidade, país, curso de CS1 e qual *Concept Inventory* deseja criar. No momento, existem dois questionários disponíveis na linguagem C (Português e Inglês).

Considerando que todos os campos são obrigatórios, é feita uma análise em *javascript* para verificar se o usuário digitou corretamente todas as informações. Ainda, é obrigatório que o usuário indique qual CI deve ser criado e associado à sua conta. No momento, existem dois questionários disponíveis na linguagem C (Português e Inglês). A Figura 3, a seguir, indica um exemplo de preenchimento do questionário, onde é indicada a criação de um CI na linguagem C e no idioma inglês:

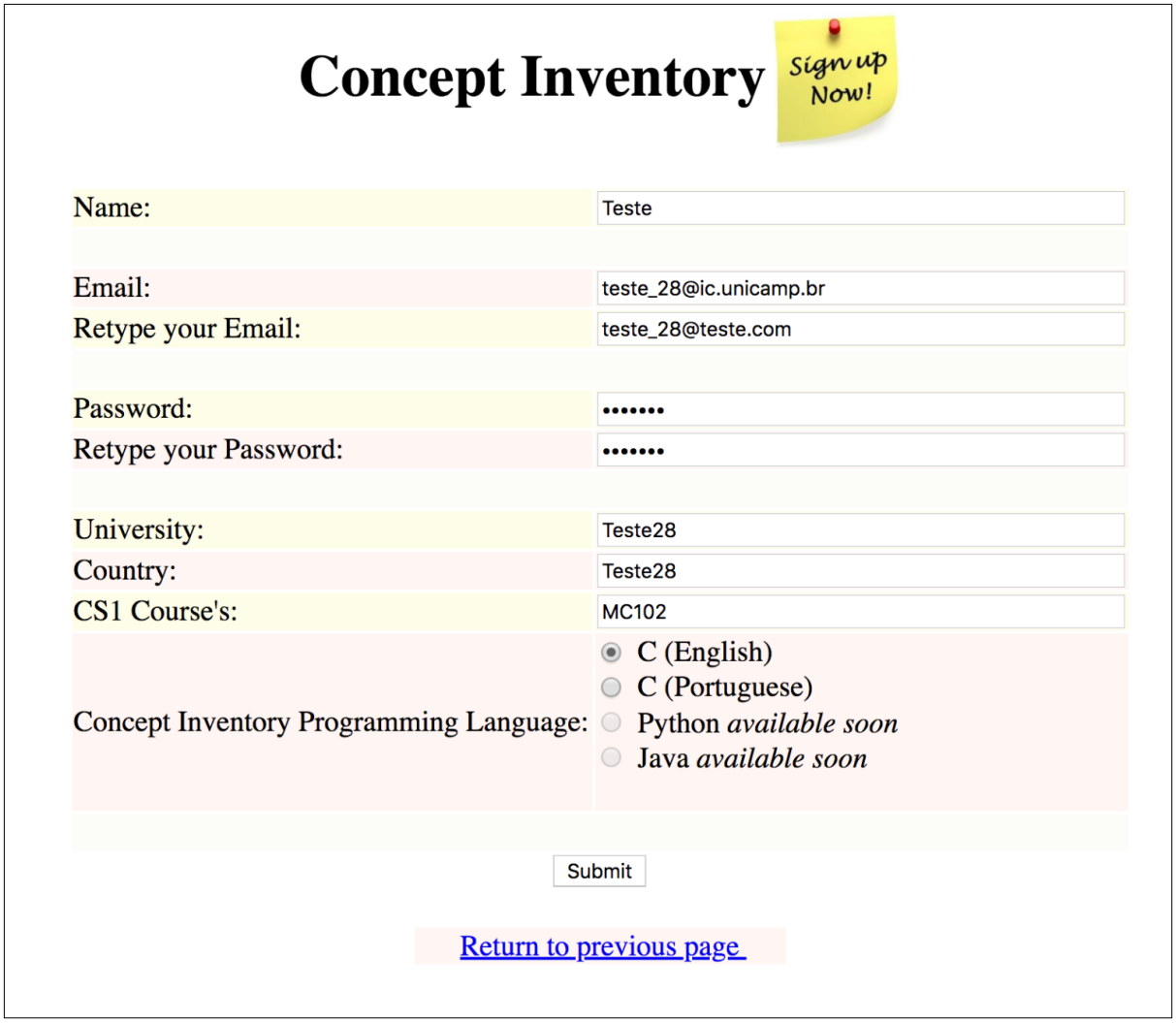


Figura 3. Exemplo de preenchimento da tela de cadastro do *website*. Neste exemplo, o usuário selecionou a opção de criação de um CI na linguagem C e no idioma inglês.

Após a criação do cadastro, o usuário recebe a confirmação dos seus dados de *login* (*email* e senha), bem como os *links* referentes ao CI criado, conforme ilustrado na Figura 4. O processo de criação e associação do CI ao usuário ocorre no *backend*, através da execução de uma série de *scripts* desenvolvidos em **PHP**, conforme descrito na Seção 4 e detalhado no Apêndice 1.

Figura 4. Tela de confirmação de cadastro de usuário. Neste exemplo o usuário teste_28@ic.unicamp.br foi criado, tendo a senha "teste28". Foram ainda fornecidos os *links* tanto para divulgação do CI (a ser enviado aos alunos) como para acesso às respostas (a ser consultado pelo professor).

3.2. *Login* dos usuários e visualização dos CIs associados

Uma vez realizado o cadastro, o usuário pode acessar o *website* e: a) consultar os CIs associados à sua conta e; b) criar novos CIs (até um total de 10). A tela de *login* é acessada através da opção "*Do you have an account?*", conforme apresentado na Figura 1. Por sua vez, a Figura 5, a seguir, apresenta um exemplo da tela de login:

Figura 5. Exemplo da tela de *login*. O usuário deve fornecer seu *email* e senha. Caso tenha esquecido a senha, pode acionar a opção "*Forgot your password?*", fazendo com que o sistema envie para seu *email* a senha cadastrada.

Caso o usuário informe erroneamente sua senha, uma página de erro é exibida, conforme ilustrado na Figura 6, a seguir:

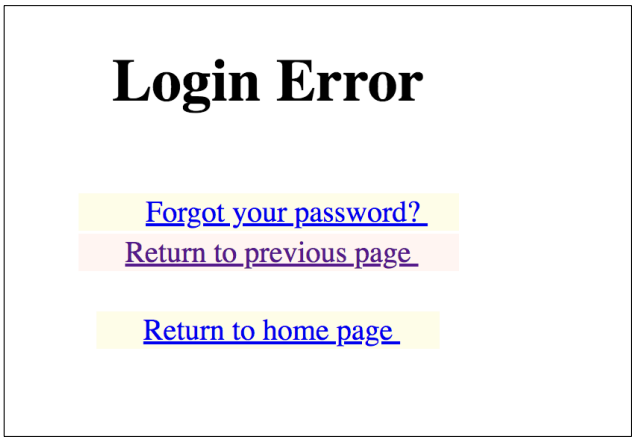


Figura 6. Exemplo da tela de erro de *login*. O usuário pode acionar a opção "*Forgot your password?*", fazendo com que o sistema envie para seu *email* a senha cadastrada.

Uma vez que o *login* ocorra de maneira correta, o usuário visualiza os dados associados ao seus CIs, conforme ilustrado na Figura 7.

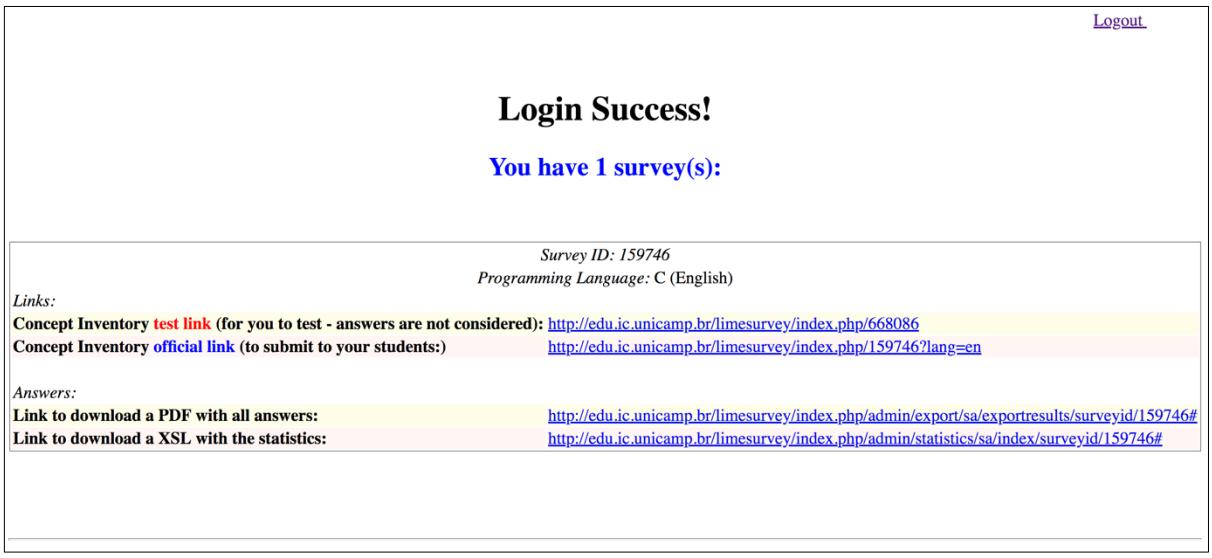


Figura 7. Tela de listagem de CIs associados a um determinado usuário. Neste exemplo, o usuário possui apenas 1 CI associado a ele.

3.3 Criação automática de novos CIs

Ainda na mesma tela de visualização dos CIs (Figura 7), o usuário pode criar novos CIs (até um total de 10), conforme apresentado na Figura 8, a seguir:

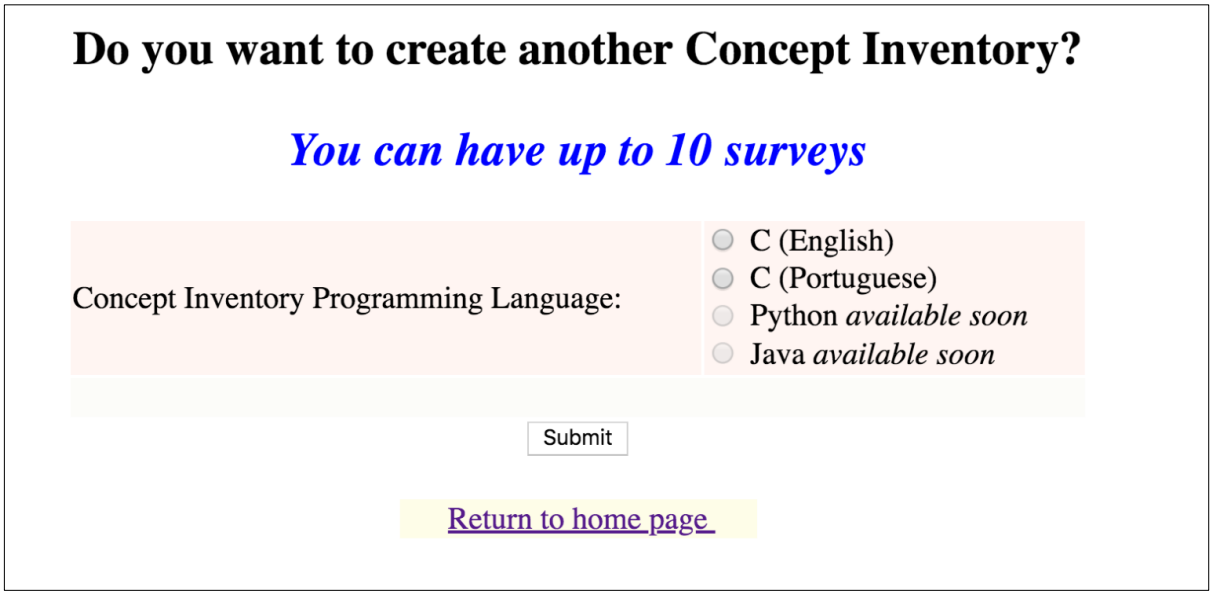


Figura 8. A tela de listagem de CIs associados a um determinado usuário apresenta ainda a opção de criação de novos CIs para o usuário.

No exemplo da Figura 8, caso o usuário selecione a opção "C (Portuguese)", um novo CI será criado, conforme ilustrado na Figura 9.

Figura 9. Tela de confirmação de criação de um CI. Neste exemplo, foi criado um CI na linguagem C e no idioma português para o usuário teste_28@ic.unicamp.br.

No exemplo da Figura 9, caso o usuário selecione a opção "*Show all your Concept Inventories*", ele será direcionado novamente para a tela de listagem de CIs (similar à descrita na Figura 7), sendo que o novo CI passará a ser listado, conforme indicado na Figura 10:

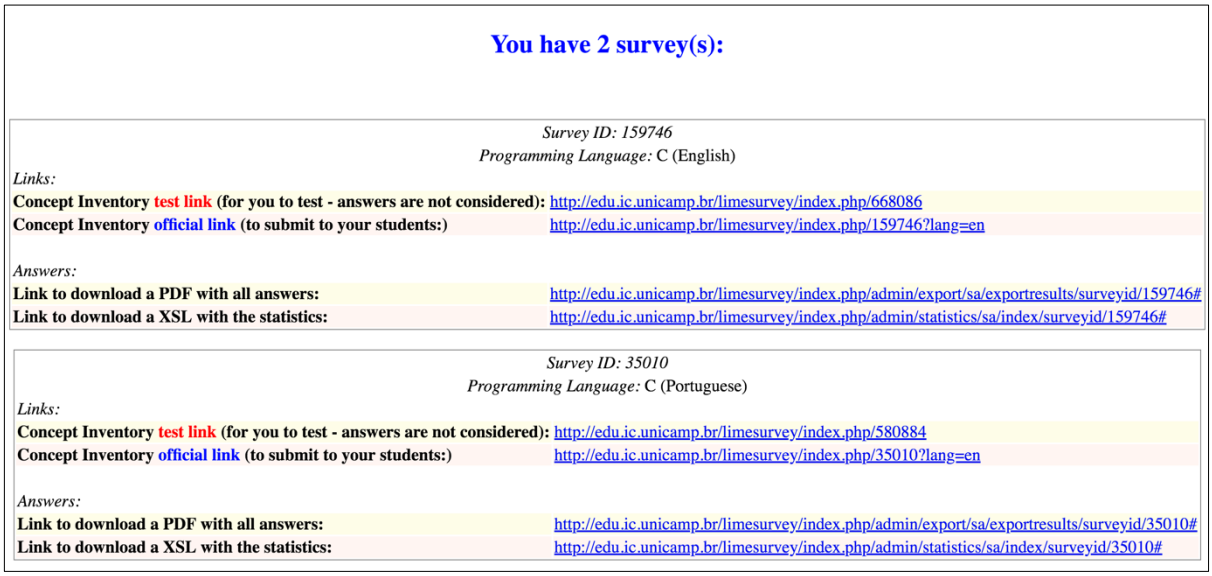


Figura 10. Tela de listagem de CIs associados a um determinado usuário. Neste exemplo, o usuário possui 2 CIs associados a ele.

4. Principais funcionalidades do *website (backend)*

Nesta seção são apresentados os *scripts* elaborados em PHP para suportar as funcionalidades descritas nas seções anteriores. Basicamente, os *scripts* em PHP agem como uma camada superior em relação ao LimeSurvey, manipulando assim diretamente os dados no SGBD. Por exemplo, o *script* responsável pela criação de um usuário na página de cadastro faz essa criação "manualmente", ou seja, **ele diretamente acessa as tabelas relacionadas** (lime_users e lime_permissions), fazendo a inserção dos dados diretamente no SGBD.

Considerando que se está em um nível "inferior" (do banco de dados), esta seção trata os CIs criados/manipulados pela sua nomenclatura no SGBD: questionários, ou *surveys*.

4.1 Criação de usuário

A Figura 11 exibe as principais entidades envolvidas na criação de um novo usuário que se registra através do *website*:

Figura 11. Relações entre as entidades **lime_users** e **lime_permissions.** (Há atributos de cada uma dessas entidades que não são exibidos na figura.) Criada com o *software* Dia Diagram Editor 0.97.3.

A entidade **lime_users** guarda informações sobre os usuários que se cadastram no *website*. Além da chave primária **uid**, gerada automaticamente pelo *script* correspondente (detalhado na Tabela 1), ela contém os demais atributos preenchidos pelo usuário no momento do cadastro: nome de usuário (chave do tipo *unique*; neste banco de dados, o nome de usuário é seu *email*), nome completo, senha (que, no banco, é exibida após criptografia), *email*, universidade, país de origem.

O campo referente ao nome do curso de introdução à programação ministrado (*cs1_course*) não existe de forma nativa no LimeSurvey, tendo sido adicionado manualmente. O *script* usado foi:

```
ALTER TABLE lime_users
ADD cs1_course varchar(254);
```

O Apêndice 1, Tabela 1, contém o *script* com a operação de criação de um usuário. O *script* em questão verifica se o *email* inserido pelo usuário já existe no banco de dados – pois, nesse caso, não pode ser utilizado novamente, e uma mensagem de erro é exibida. Caso o *email* seja inédito, são executados os passos necessários para inserção do usuário no banco: geração de uma nova chave **uid**; preenchimento, no *script*, dos campos correspondentes definidos pelo usuário, a partir dos dados inseridos por ele no formulário (*email*, nome completo, senha); preenchimento dos demais campos com valores padrão ou, quando for o caso, da data e hora correntes; inserção desses campos na tabela **lime_users**.

A entidade **lime_permissions** guarda informações sobre as permissões concedidas a um usuário cadastrado. Além da chave primária **id**, gerada automaticamente pelo *script* correspondente (ver Apêndice 1, Tabela 2, que contém a operação de adição de permissões a um usuário), ela contém a chave estrangeira **uid**, que referencia a entidade **lime_users**. Também guarda as informações **entity** (tipo de permissão), **entity_id** (chave do tipo *multiple,* que referencia a chave primária **sid** da entidade **lime_surveys**, explicada adiante) e o tipo de permissão concedida.

O *script* descrito no Apêndice 1, Tabela 2 adiciona três permissões a um usuário recentemente criado: (1) de *login*, (2) de acesso a *templates* e (3) de acesso a *surveys*. Para tanto, são inseridas três novas tuplas na tabela **lime_permissions** (cada uma com sua chave

id), nas quais o campo **uid** será preenchido com o mesmo **uid** do usuário em questão, e os demais campos serão ajustados de modo a acrescentar as respectivas permissões.

As Figuras 12 e 13 ilustram as modificações geradas na tabela **lime_users** ao ser introduzido um novo usuário no banco de dados:

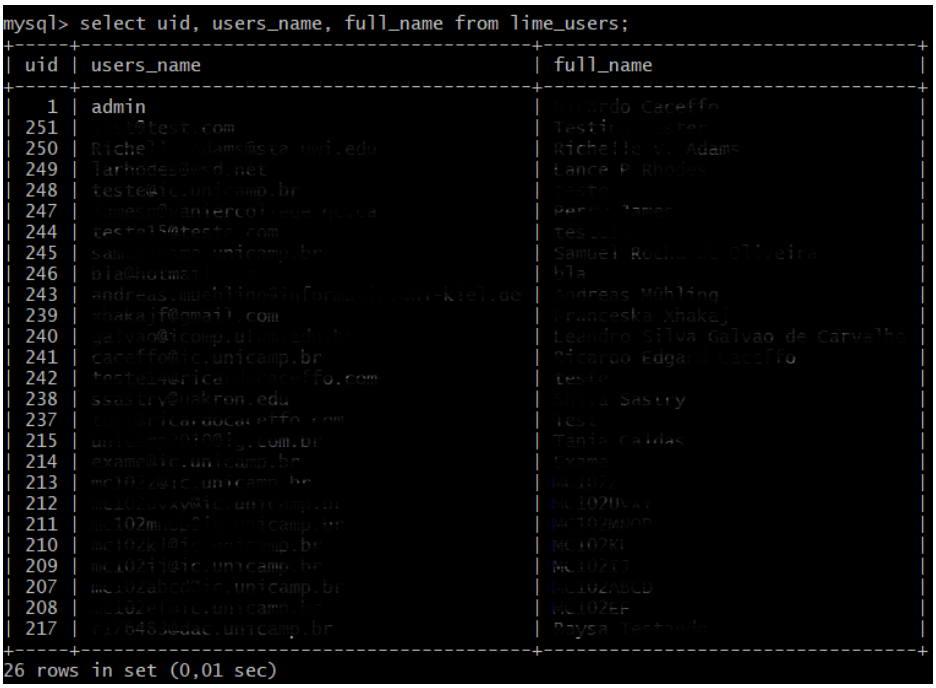


Figura 12. Resultado de consulta sobre a tabela **lime_users**, antes da criação de um novo usuário ilustrativo. A consulta resulta em 26 tuplas. A maior parte dos nomes foi ocultada por razões de segurança.

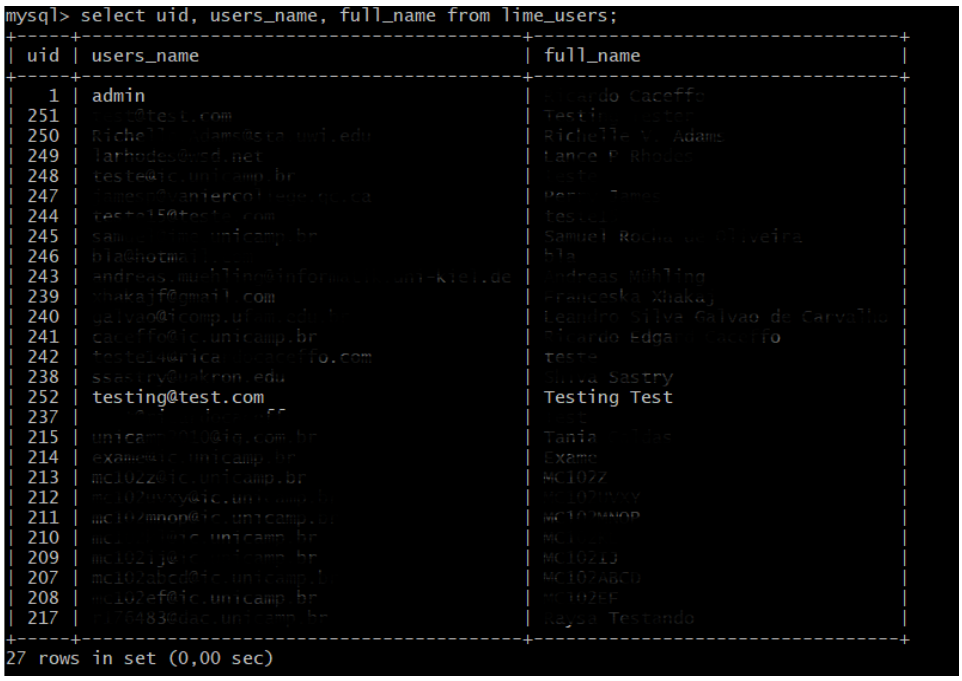


Figura 13. Resultado de consulta sobre a tabela **lime_users**, após a criação de um novo usuário fictício, Testing Test, de *email* testing@test.com. A tupla correspondente ao novo usuário foi incluída na tabela, resultando em 27 linhas no total. O identificador 252 foi atribuído ao novo usuário.

A Figura 14, por sua vez, ilustra as alterações em **lime_permissions** após introdução de um usuário:

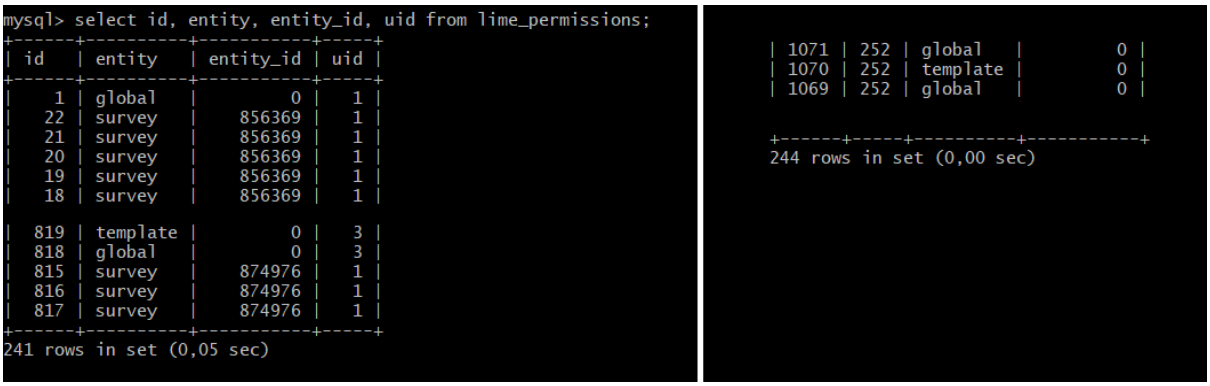


Figura 14. Alteração na tabela **lime_permissions** após criação do usuário fictício de **id** 252. À esquerda, consulta original sobre a tabela, resultando em 241 tuplas (a maior parte das tuplas foi omitida na imagem). À direita, parte do resultado da mesma consulta após a introdução de Testing Test, resultando em 244 tuplas, com destaque para as 3 novas permissões adicionadas.

4.2 Criação de um *survey*

As principais entidades envolvidas na criação de um novo *survey* (CI) estão exibidas na Figura 15:

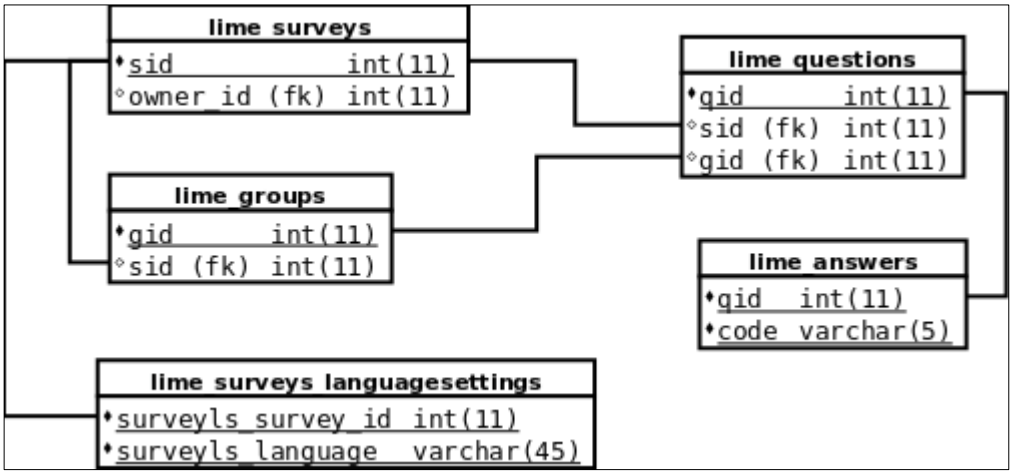


Figura 15. Relações entre as entidades **lime_surveys, lime_questions, lime_groups, lime_surveys_languagesettings** e **lime_answers. (**Há atributos de cada uma dessas entidades que não são exibidos na figura.) Criada com o *software* Dia Diagram Editor 0.97.3.

A entidade **lime_surveys** guarda informações sobre os questionários existentes (*surveys*). Além da chave primária **sid**, cujo conteúdo é idêntico ao de **entity_id** relacionada ao usuário que criou determinado *template* (*owner*), ela contém a chave estrangeira **owner_id**, que referencia a entidade **lime_users** (o conteúdo de **owner_id** é o **id** do usuário *owner*). Também guarda outras informações sobre o *template* criado, como a data de início, o *status* (ativo ou não), a linguagem de programação e idioma utilizados, dentre outros. Ela passa a ser utilizada a partir do momento em que um usuário cadastrado cria um novo questionário.

A entidade **lime_questions** guarda informações sobre as questões disponíveis em um *survey*. Além da chave primária **qid**, gerada automaticamente, ela contém as chaves estrangeiras **sid**, que referencia a entidade **lime_surveys**, e **gid**, que referencia a entidade **lime_groups**. Ou seja: cada questão está associada a um *survey* e a um grupo de questões dentro desse *survey*. Também guarda outras informações sobre cada questão, como seu título, o texto da pergunta, a obrigatoriedade de ser respondida, o idioma (atributo que também compõe a chave primária), dentre outros.

A entidade **lime_groups** guarda informações sobre grupos de questões existentes em um *survey*. Além da chave primária **gid**, gerada automaticamente, ela contém a chave estrangeira **sid**, que referencia a entidade **lime_surveys**. Ou seja: cada grupo está associado a um *survey*. Também guarda outras informações sobre cada grupo, como seu nome, descrição e idioma (atributo que também compõe a chave primária), dentre outros.

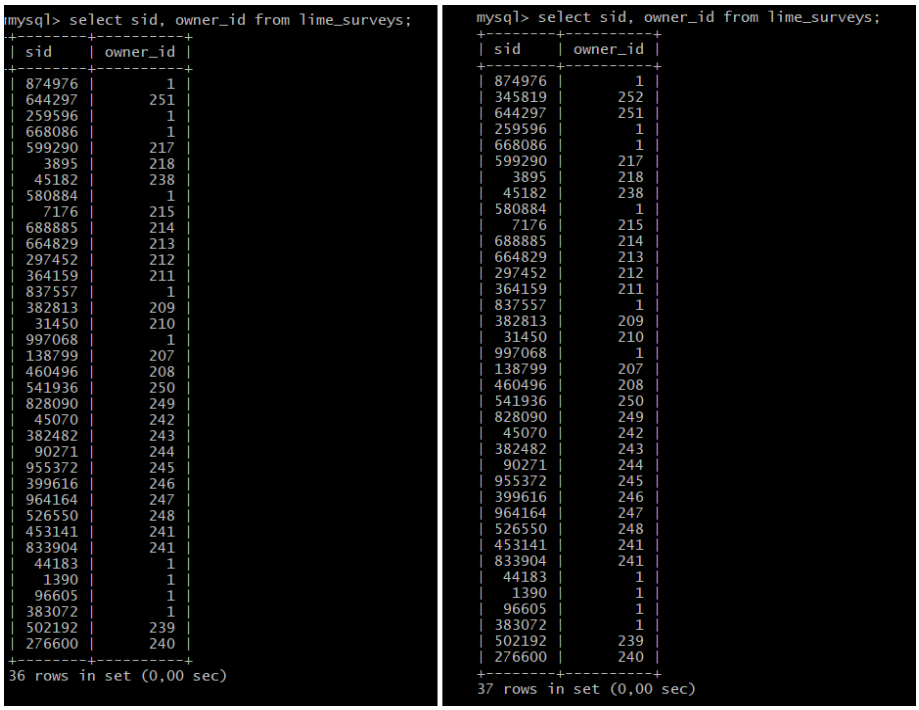
A entidade **lime_surveys_languagesettings** guarda informações sobre configurações específicas de um *survey*. Além da chave primária **surveyls_survey_id** (cujo conteúdo é idêntico à chave **sid** do *survey* ao qual uma tupla está associada), ela contém informações de configuração como idioma (atributo que também compõe a chave primária), título e descrição do *survey*, dentre outros.

A entidade **lime_answers** guarda informações sobre as respostas disponíveis em um *survey*. Além da chave primária **qid** (cujo conteúdo é idêntico à chave **qid** da questão à qual uma tupla está associada), ela contém informações sobre cada resposta, como seu código, idioma (atributos que também compõem a chave primária) e texto da resposta, dentre outros. O código é o atributo que guarda a informação de corretude de uma resposta: se seu valor é **R**, ela é a resposta correta de uma questão; qualquer outro valor equivale ao *misconception* ao qual a resposta é mapeada, conforme o código previamente determinado no trabalho de identificação de *misconceptions*. No caso do *survey*-base (*template*) da linguagem C em inglês, por exemplo, os códigos correspondem àqueles determinados em [1] – como **A1**, "*Parameter value set by an external source*", ou **G3**, "*Arithmetic expression instead of Boolean expression*".

Quando um novo *survey* é criado por um usuário, utiliza-se, como base, um *template* pré-existente no *site*. No banco de dados, os *templates* são *surveys* comuns, inseridos como registros na entidade **lime_surveys** com **owner_id** representando a autoria do administrador (por exemplo, pode-se determinar que o **owner_id** do administrador será sempre 1). A partir da escolha de *template* feita pelo usuário no formulário disponibilizado no *site*, copia-se o conteúdo do *template* em questão para o novo *survey*, que terá, então, **id** e registro próprios no banco. Também se cria um *survey* de teste para o usuário, espelhado do novo *survey,* para que se possa realizar verificações e tentativas sem "sujar" o *survey* principal recentemente criado.

Atualmente, são dois os *templates* de *surveys* disponíveis: um em inglês (de **id** 509059) e um em português (de **id** 837557), ambos em linguagem de programação C. Esses *templates* e seus **ids** estão sujeitos a alterações – que devem ser inseridas, manualmente, no *script* correspondente. O *script* do Apêndice 1, Tabela 3 mostra a captura e preservação da escolha feita pelo usuário para criação de um novo *survey*. O Apêndice 1, Tabela 4, por sua vez, contém o *script* utilizado para efetivamente criar, no banco de dados, um novo registro em **lime_surveys**, com o novo identificador. Tal *script* verifica se o id gerado já existe e, em caso positivo, gera um novo; em seguida, insere, na tabela **lime_surveys**, um novo registro.

A Figura 16 ilustra a modificação ocorrida em **lime_surveys** a partir do momento em que o usuário recém-introduzido no banco, Testing Test, cria um novo *survey*:



```
mysql> select sid, owner_id from lime_surveys;
+--------+----------+
| sid    | owner_id |
+--------+----------+
| 874976 |        1 |
| 644297 |      251 |
| 259596 |        1 |
| 668086 |        1 |
| 599290 |      217 |
|   3895 |      218 |
|  45182 |      238 |
| 580884 |        1 |
|   7176 |      215 |
| 688885 |      214 |
| 664829 |      213 |
| 297452 |      212 |
| 364159 |      211 |
| 837557 |        1 |
| 382813 |      209 |
|  31450 |      210 |
| 997068 |        1 |
| 138799 |      207 |
| 460496 |      208 |
| 541936 |      250 |
| 828090 |      249 |
|  45070 |      242 |
| 382482 |      243 |
|  90271 |      244 |
| 955372 |      245 |
| 399616 |      246 |
| 964164 |      247 |
| 526550 |      248 |
| 453141 |      241 |
| 833904 |      241 |
|  44183 |        1 |
|   1390 |        1 |
|  96605 |        1 |
| 383072 |        1 |
| 502192 |      239 |
| 276600 |      240 |
+--------+----------+
36 rows in set (0,00 sec)
```

```
mysql> select sid, owner_id from lime_surveys;
+--------+----------+
| sid    | owner_id |
+--------+----------+
| 874976 |        1 |
| 345819 |      252 |
| 644297 |      251 |
| 259596 |        1 |
| 668086 |        1 |
| 599290 |      217 |
|   3895 |      218 |
|  45182 |      238 |
| 580884 |        1 |
|   7176 |      215 |
| 688885 |      214 |
| 664829 |      213 |
| 297452 |      212 |
| 364159 |      211 |
| 837557 |        1 |
| 382813 |      209 |
|  31450 |      210 |
| 997068 |        1 |
| 138799 |      207 |
| 460496 |      208 |
| 541936 |      250 |
| 828090 |      249 |
|  45070 |      242 |
| 382482 |      243 |
|  90271 |      244 |
| 955372 |      245 |
| 399616 |      246 |
| 964164 |      247 |
| 526550 |      248 |
| 453141 |      241 |
| 833904 |      241 |
|  44183 |        1 |
|   1390 |        1 |
|  96605 |        1 |
| 383072 |        1 |
| 502192 |      239 |
| 276600 |      240 |
+--------+----------+
37 rows in set (0,00 sec)
```

Figura 16. À esquerda, consulta original sobre **lime_surveys**, resultando em 36 tuplas. À direita, mesma consulta após introdução de um novo *survey* pelo usuário de **id** 252, resultando em 37 tuplas. A nova tupla tem **sid** 345819. Não é criada uma nova tupla para o *survey* teste, apenas para o principal.

A inserção das informações copiadas em um novo registro também é feita na tabela **lime_surveys_languagesettings**, conforme pode ser constatado no *script* do Apêndice 1, Tabela 5. As Figuras 17 e 18 ilustram as modificações ocorridas em **lime_surveys_languagesettings**:

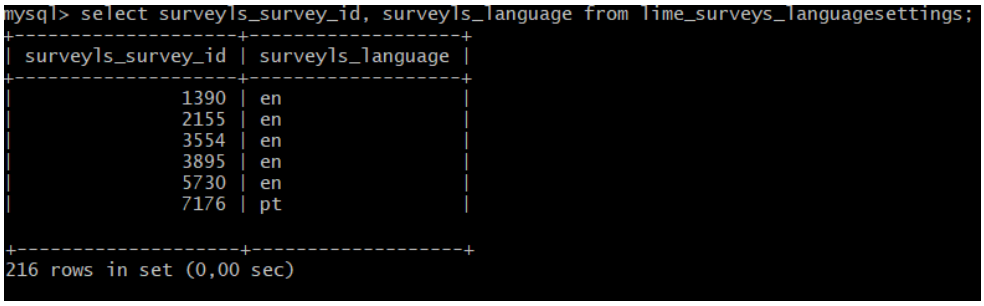
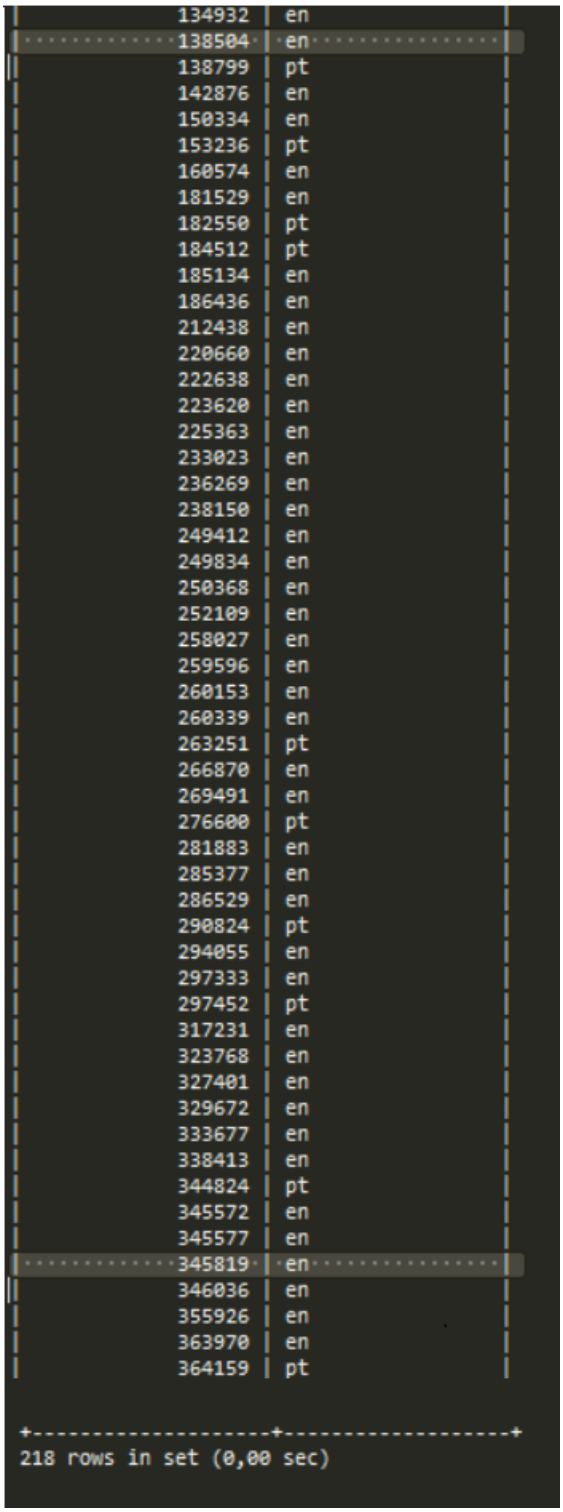


Figura 17. Consulta original sobre a tabela **lime_surveys_languagesettings**, resultando em 216 tuplas (a maioria das quais foi omitida na imagem).



134932	en
138504	en
138799	pt
142876	en
150334	en
153236	pt
160574	en
181529	en
182550	pt
184512	pt
185134	en
186436	en
212438	en
220660	en
222638	en
223620	en
225363	en
233023	en
236269	en
238150	en
249412	en
249834	en
250368	en
252109	en
258027	en
259596	en
260153	en
260339	en
263251	pt
266870	en
269491	en
276600	pt
281883	en
285377	en
286529	en
290824	pt
294055	en
297333	en
297452	pt
317231	en
323768	en
327401	en
329672	en
333677	en
338413	en
344824	pt
345572	en
345577	en
345819	en
346036	en
355926	en
363970	en
364159	pt

218 rows in set (0,00 sec)

Figura 18. Parte do resultado da mesma consulta após a criação de um novo *survey,* com destaque para as duas tuplas a mais que figuram no resultado: uma de **id** 345819 e outra de **id** 138504, sendo a última referente ao *survey* teste.

Em seguida, são copiadas as informações sobre questões e grupos para o novo questionário criado. O *script* do Apêndice 1, Tabela 6 traz a descrição e implementação do algoritmo utilizado para realizar a cópia. O último elemento a ser copiado são as respostas (ver

Apêndice 1, Tabela 7). As Figuras 19, 20 e 21 ilustram as modificações geradas nas tabelas **lime_groups, lime_questions** e **lime_answers** após o usuário Testing Test criar um novo *survey*:

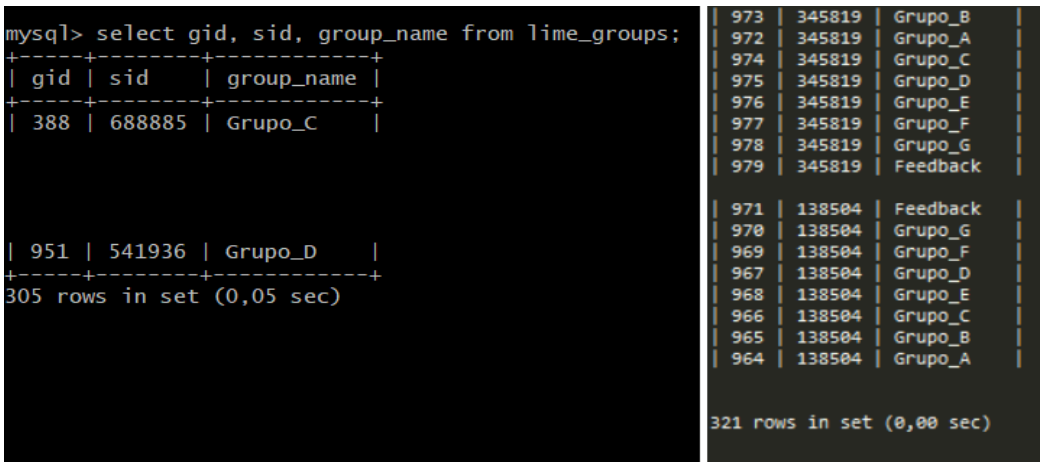


Figura 19. À esquerda, consulta original sobre a tabela **lime_groups**, resultando em 305 tuplas (a maioria das quais foi omitida na imagem). À direita, parte do resultado da mesma consulta após a criação de um novo *survey,* resultando em 16 tuplas a mais: oito para o novo *survey* principal, de **id** 345819, e oito para o novo *survey* teste, de **id** 138504. Cada novo *survey* passa a ter, portanto, informações sobre os grupos de questões previamente definidos (no exemplo em tela, 8 grupos no total: 7 convencionados conforme a referência [1] e 1 dedicado a abarcar uma questão final de *feedback*).

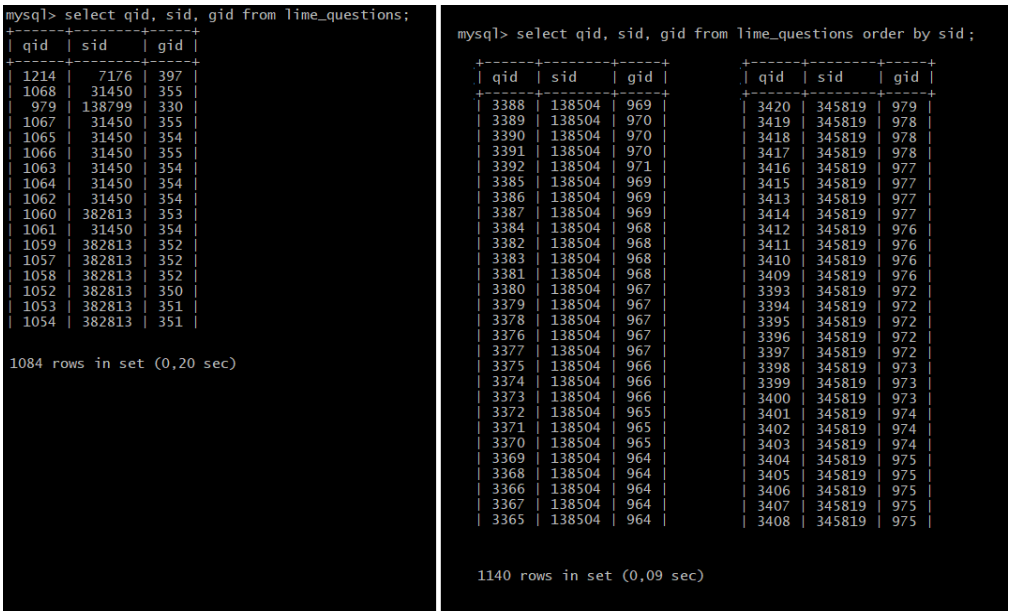


Figura 20. À esquerda, consulta original sobre a tabela **lime_questions**, resultando em 1084 tuplas (a maioria das quais foi omitida na imagem). À direita, parte do resultado da mesma consulta após a criação de um novo *survey,* resultando em 56 tuplas a mais: 28 para o novo *survey* principal, de **id** 345819, e 28 para o novo *survey* teste, de **id** 138504. Cada novo *survey* passa a ter, portanto, informações sobre as questões previamente definidas (no exemplo em tela, 28 questões no total, 27 das quais determinadas na referência [2], sendo a 28ª uma questão final de *feedback*).

```
mysql> select qid, code from lime_answers;
+------+------+
| qid  | code |
+------+------+
|  482 | A1   |
|  482 | A2   |
|  482 | A3   |
|  482 | R    |
|  485 | A3   |
|  485 | A4   |
|  485 | A5   |
|  485 | A6   |
|  485 | R    |
+------+------+
6434 rows in set (0,14 sec)
```

```
| 3419 | G2   |
| 3419 | G3   |
| 3419 | G4   |
| 3419 | R    |
+------+------+
6686 rows in set (0,02 sec)
```

Figura 21. À esquerda, consulta original sobre a tabela **lime_answers**, resultando em 6434 tuplas (a maioria das quais foi omitida na imagem). À direita, parte do resultado da mesma consulta após a criação de um novo *survey,* resultando em 252 tuplas a mais: 126 para o novo *survey* principal, de **id** 345819, e 126 para o novo *survey* teste, de **id** 138504. Cada novo *survey* passa a ter, portanto, informações sobre as opções de respostas disponibilizadas, conforme previamente definido para cada questão (no exemplo em tela, as possibilidades de respostas foram determinadas na referência [2]).

Finalmente, criam-se duas novas tabela no banco (uma para o *survey* principal e uma para o *survey* teste), do tipo **lime_survey_X**, em que X é o **id** do questionário recém-criado (**sid**) (ver Apêndice 1, Tabela 8). A Figura 22 ilustra a criação de uma das novas tabelas, mostrando seus atributos:

```
mysql> desc lime_survey_345819;
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| id              | int(11)     | NO   | PRI | NULL    | auto_increment |
| token           | varchar(35) | YES  | MUL | NULL    |                |
| submitdate      | datetime    | YES  |     | NULL    |                |
| lastpage        | int(11)     | YES  |     | NULL    |                |
| startlanguage   | varchar(20) | NO   |     | NULL    |                |
| 345819X979X3420 | varchar(5)  | YES  |     | NULL    |                |
| 345819X978X3419 | varchar(5)  | YES  |     | NULL    |                |
| 345819X978X3418 | varchar(5)  | YES  |     | NULL    |                |
| 345819X978X3417 | varchar(5)  | YES  |     | NULL    |                |
| 345819X977X3416 | varchar(5)  | YES  |     | NULL    |                |
| 345819X977X3415 | varchar(5)  | YES  |     | NULL    |                |
| 345819X977X3413 | varchar(5)  | YES  |     | NULL    |                |
| 345819X977X3414 | varchar(5)  | YES  |     | NULL    |                |
| 345819X976X3412 | varchar(5)  | YES  |     | NULL    |                |
| 345819X976X3411 | varchar(5)  | YES  |     | NULL    |                |
| 345819X976X3410 | varchar(5)  | YES  |     | NULL    |                |
| 345819X976X3409 | varchar(5)  | YES  |     | NULL    |                |
| 345819X972X3393 | varchar(5)  | YES  |     | NULL    |                |
| 345819X972X3394 | varchar(5)  | YES  |     | NULL    |                |
| 345819X972X3395 | varchar(5)  | YES  |     | NULL    |                |
| 345819X972X3396 | varchar(5)  | YES  |     | NULL    |                |
| 345819X972X3397 | varchar(5)  | YES  |     | NULL    |                |
| 345819X973X3398 | varchar(5)  | YES  |     | NULL    |                |
| 345819X973X3399 | varchar(5)  | YES  |     | NULL    |                |
| 345819X973X3400 | varchar(5)  | YES  |     | NULL    |                |
| 345819X974X3401 | varchar(5)  | YES  |     | NULL    |                |
| 345819X974X3402 | varchar(5)  | YES  |     | NULL    |                |
| 345819X974X3403 | varchar(5)  | YES  |     | NULL    |                |
| 345819X975X3404 | varchar(5)  | YES  |     | NULL    |                |
| 345819X975X3405 | varchar(5)  | YES  |     | NULL    |                |
| 345819X975X3406 | varchar(5)  | YES  |     | NULL    |                |
| 345819X975X3407 | varchar(5)  | YES  |     | NULL    |                |
| 345819X975X3408 | varchar(5)  | YES  |     | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
33 rows in set (0,09 sec)
```

Figura 22. Nova tabela criada para o *survey* principal, de **id** 345819. Uma tabela semelhante é criada para o *survey* teste.

Conforme explicado no *script* do Apêndice 1, Tabela 8, é nas tabelas do tipo **lime_survey_X** que o banco de dados armazena informações sobre as respostas fornecidas por pessoas que respondem aos *surveys*. Assim, essas tabelas possuem, além da chave primária **id**, atributos que gravam, para cada respondente, informações sobre a submissão, como data e hora. Também é criado um atributo para cada uma das questões, obedecendo ao formato *12345X22X333nome,* em que *12345* é o **id** do survey, *22* é o **id** do grupo, *333* é o **id** da questão e *nome* é o código da resposta (mostrado apenas para alguns tipos de questão).

A Figura 23 mostra que, enquanto não houver respondentes, os atributos referentes a cada uma das questões permanecem vazios:

```
mysql> select 345819X979X3420 from lime_survey_345819;
Empty set (0,00 sec)
```

Figura 23. Ao tentar selecionar o atributo referente a uma das questões do *survey* recém-criado (portanto, não respondido) de **id** 345819, o banco retorna um resultado vazio.

4.3 Respostas dos *surveys* e estatísticas

Conhecendo o modo como o banco de dados armazena as informações de respostas submetidas nos *surveys*, é possível elaborar *scripts* que permitam retornar, ao professor autor de um questionário, um relatório sobre o desempenho de seus alunos. Esta seção descreve as características dessas respostas no banco de dados e fornece opções para esses *scripts,* escritos e testados em linguagem de consulta (MySQL), para que possam ser eventualmente integrados ao *website*.

O objetivo é que, por exemplo, se um *survey* teve 100 respondentes, seu autor deve receber uma mensagem informando que 100 questionários foram respondidos; além disso, ele deve ser informado sobre quantas questões respondidas esse número representa no total, bem como quantas dessas questões pertencem a cada tópico. Por exemplo, se o *template* utilizado para a elaboração desse questionário foi conforme descrito em [2], essas 100 respostas representam um total de 2800 questões respondidas – 500 das quais referem-se ao tópico A (“Uso e Escopo de Parâmetros de Funções”). Se, dessas 500, metade foi marcada com a resposta correta (ou seja, 250), o relatório deve indicar, por exemplo, que “o aproveitamento no tópico A foi de 50%, o que indica que os respondentes têm conhecimento intermediário neste tópico”.

A quantidade de respondentes que finalizaram um questionário, inclusive submetendo-o, pode ser obtida a partir de um *script* que siga a forma*:*

(1) `select count(345819X979X3420) from lime_survey_345819;`

Neste caso, `lime_survey_345819` é a tabela do tipo lime_survey_X criada pelo banco de dados para o questionário criado pelo usuário Testing Test, usado nos exemplos anteriores. O atributo `345819X979X3420` faz referência à questão de **id** 3420, pertencente ao grupo de **id** 979 - que, no caso em tela, é a questão de submissão do questionário. O número do grupo que contém a questão de submissão só aparece uma vez na lista de atributos de uma tabela desse tipo.

Quando um respondente submete o questionário, após responder todas as questões, uma nova tupla, de conteúdo vazio, é adicionada a esse atributo. Usuários que respondam ao questionário apenas parcialmente, sem submetê-lo, geram uma nova tupla de conteúdo "NULL" neste atributo; neste caso, as colunas de todas as questões também ganharão uma nova tupla, cujo conteúdo será o código da resposta (no caso das respostas marcadas antes de fechar o questionário) ou "NULL" (no caso das respostas não marcadas antes de o questionário ser fechado). Essas novas tuplas somente deixam de ser gravadas no banco de dados se o respondente seleciona a opção "*exit and clear survey*", optando, assim, por abandonar o questionário sem registrar nenhuma resposta deixada até então.

A Figura 24 ilustra o conteúdo da coluna `345819X979X3420` da tabela `lime_survey_345819` após a realização de um teste com quatro respondentes, em que: 1 deles respondeu a todas as perguntas e submeteu o questionário; 1 deles respondeu a todas as perguntas e não submeteu o questionário, fechando a tela abruptamente sem submeter; 1 deles respondeu a algumas perguntas e fechou o questionário antes de terminá-lo, fechando a tela abruptamente, sem submeter; e 1 deles respondeu a algumas perguntas e fechou o questionário antes de terminá-lo, clicando em "*exit and clear survey*":

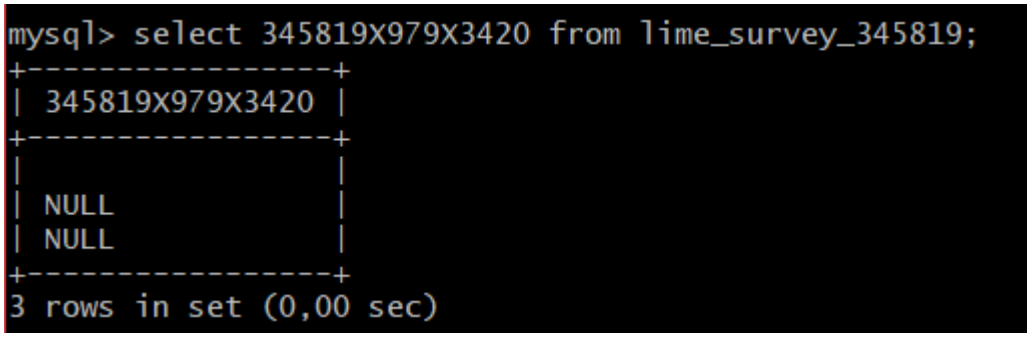


Figura 24. Após os testes, o atributo da questão de submissão registra três novas tuplas, duas das quais com conteúdo "NULL" (em razão de usuários que responderam parcialmente ao questionário, sem submetê-lo, e abandonando-o apenas fechando a tela) e uma com conteúdo vazio (em razão do usuário que submeteu

corretamente o questionário). O usuário que abandonou o questionário clicando em "*exit and clear survey*" não gerou uma nova tupla em nenhuma das colunas da tabela **lime_survey_345819**.

Como a função *count* ignora as tuplas com conteúdo "NULL" exceto quando explicitado o contrário, o *script* (1) devolverá a quantidade de respondentes que submeteram o questionário corretamente – no caso do exemplo, devolverá a quantidade 1.

Caso se deseje permitir ao autor de um questionário que saiba quantos questionários foram respondidos parcialmente, abandonados abruptamente (sem clicar em "*exit and clear survey*"), é possível obter essa informação a partir de um *script* que siga a forma:

(2) `select count(id) from lime_survey_345819 where 345819X979X3420 is null;`

Neste caso, conta-se quantos **ids** foram registrados (o que é feito automaticamente pelo banco de dados quando o questionário tem respondentes, ainda que eles sejam anônimos) e destacam-se somente aqueles que geraram, na coluna da questão de submissão, o conteúdo "NULL". No caso do exemplo, o *script* devolverá o resultado 2.

Para obter informações sobre as respostas de uma dada questão, é preciso analisar a coluna referente àquela questão. Vejamos, por exemplo, na Figura 25, o conteúdo do atributo `345819X972X3393` da tabela `lime_survey_345819`, que se refere à questão de **id** 3393 (A4), do grupo de **id** 972 (grupo A):

```
mysql> select 345819X972X3393 from lime_survey_345819;
+-----------------+
| 345819X972X3393 |
+-----------------+
| A4              |
| NULL            |
| A4              |
+-----------------+
3 rows in set (0,00 sec)
```

Figura 25. Conteúdo da coluna que mostra as respostas para a questão A4 do questionário de **id** 345819. Dois dos três respondentes marcaram, para essa questão, a mesma resposta, que corresponde a um *misconception* mapeado com o código A4; portanto, nenhum dos dois acertou. O outro respondente não marcou nenhuma resposta, o que indica que abandonou o questionário antes de atingir essa questão. Note que, neste caso, a questão (A4) e o código das respostas dadas (A4) são os mesmos, o que se trata de uma coincidência apenas.

A quantidade de pessoas que responderam a uma dada questão, portanto, pode ser obtida a partir de um *script* que siga a forma:

(3) `select count(345819X972X3393) from lime_survey_345819;`

Já a quantidade de respondentes que acertaram a uma dada questão pode ser obtida a partir de um *script* que siga a forma*:*

(4) `select count(345819X972X3393) from lime_survey_345819 where 345819X972X3393 = 'R';`

Nos exemplos (3) e (4), o atributo `345819X972X3393` faz referência à questão de **id** 3393 (A4), pertencente ao grupo de **id** 972 (grupo A). Note que, em (4), foi incluída uma condição: o conteúdo da coluna da questão deve ser igual ao código 'R', indicando que a questão foi respondida corretamente. Para o exemplo apresentado anteriormente, o *script* (3) devolveria o resultado 2, e o *script* (4) devolveria o resultado 0, pois as duas respostas válidas (não "NULL") registradas estão incorretas.

Assim, caso se queira obter a proporção de pessoas que acertaram uma dada questão, o *script* a ser usado como base pode ter a forma:

(5)

```
select
(select count(345819X972X3393) from lime_survey_345819 where 345819X972X3393 = 'R')
/
(select count(345819X972X3393) from lime_survey_345819)
as AcertosA4;
```

Neste caso, divide-se o número encontrado em (4) (pessoas que acertaram a questão) pelo número encontrado em (3) (pessoas que responderam à questão).

Para obter informações sobre um grupo inteiro, de modo a auferir a compreensão de um conjunto de respondentes em um dado assunto, o caminho é similar; neste caso, porém, é preciso coletar as informações de todas as colunas de questões de um mesmo grupo, isto é, colunas cujos dígitos intermediários (referentes ao **id** do grupo da questão) sejam iguais. Vejamos, por exemplo, na Figura 26, o conteúdo de todos os atributos de questões do grupo A, isto é, com **id** de grupo 972 da tabela `lime_survey_345819`:

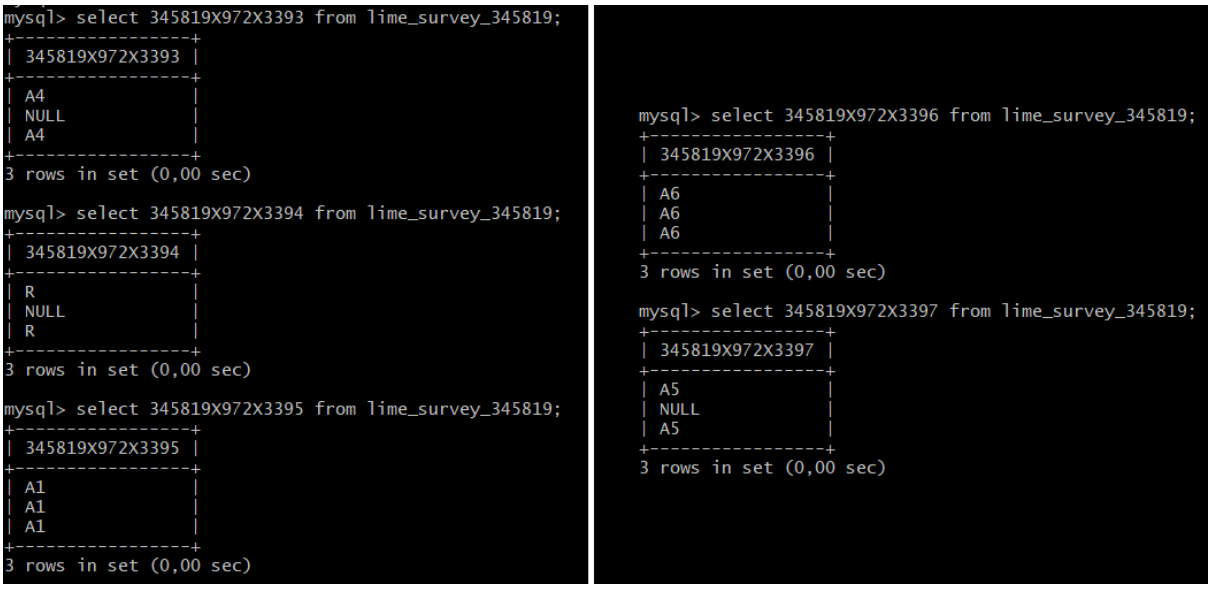


Figura 26. Conteúdos de todas as respostas de questões do grupo A. As questões têm **ids** 3393 (A4), 3394 (A5), 3395 (A1), 3396 (A2) e 3397 (A3). Uma tupla com conteúdo "NULL" indica que um respondente abandonou o questionário abruptamente antes de chegar àquela questão. Note que o LimeSurvey não organiza os **ids** das questões na mesma ordem em que elas são apresentadas para os respondentes.

A quantidade de pessoas que responderam às questões de um dado grupo, portanto, pode ser obtida a partir de um *script* que siga a forma:

(6)

```
select
((select count(345819X972X3393) from lime_survey_345819)+
(select count(345819X972X3394) from lime_survey_345819) +
(select count(345819X972X3395) from lime_survey_345819) +
(select count(345819X972X3396) from lime_survey_345819) +
(select count(345819X972X3397) from lime_survey_345819))
as RespostasValidasA;
```

Já a quantidade de respondentes que acertaram a questões de um dado grupo pode ser obtida a partir de um *script* que siga a forma:

(7)

```
select
((select count(345819X972X3393) from lime_survey_345819
where 345819X972X3393 = 'R') +
(select count(345819X972X3394) from lime_survey_345819
where 345819X972X3394 = 'R') +
(select count(345819X972X3395) from lime_survey_345819
where 345819X972X3395 = 'R') +
(select count(345819X972X3396) from lime_survey_345819
where 345819X972X3396 = 'R') +
(select count(345819X972X3397) from lime_survey_345819
where 345819X972X3397 = 'R'))
as AcertosA;
```

Nos exemplos (6) e (7), somam-se os resultados de todas as questões pertencentes a um mesmo grupo – no caso, o grupo A, que tem **id** 972. Para o exemplo apresentado conforme a Figura 25, o *script* (6) devolveria o resultado 12, e o *script* (7) devolveria o resultado 2.

Assim, caso se queira obter a proporção de pessoas que acertaram questões de um dado grupo, o *script* a ser usado como base poderia ter a forma:

(8)

```
select
(select
((select count(345819X972X3393) from lime_survey_345819
where 345819X972X3393 = 'R') +
(select count(345819X972X3394) from lime_survey_345819
where 345819X972X3394 = 'R') +
(select count(345819X972X3395) from lime_survey_345819
where 345819X972X3395 = 'R') +
(select count(345819X972X3396) from lime_survey_345819
where 345819X972X3396 = 'R') +
(select count(345819X972X3397) from lime_survey_345819
where 345819X972X3397 = 'R'))
as AcertosA)
```

```
      /
       (select
      ((select count(345819X972X3393) from lime_survey_345819)+
      (select count(345819X972X3394) from lime_survey_345819) +
      (select count(345819X972X3395) from lime_survey_345819) +
      (select count(345819X972X3396) from lime_survey_345819) +
      (select count(345819X972X3397) from lime_survey_345819))
as RespostasValidasA)
       as ResultadoGrupoA;
```

Neste caso, divide-se o número encontrado em (7) (pessoas que acertaram questões desse grupo) pelo número encontrado em (6) (pessoas que responderam a questões desse grupo). No exemplo mostrado anteriormente, o *script* (8) devolveria o resultado 0,1667, isto é, 1/6; essa é a proporção de sucesso que os respondentes tiveram no grupo A, indicando, caso se tratasse de uma turma real de programação, um conhecimento baixo no tópico correspondente ao código desse grupo (*Function Parameter Use and Scope*, segundo [1]).

É importante notar, ainda, como o LimeSurvey estabelece a ordem dos **ids** de grupos. Os **ids** de grupo são incrementados a partir do último **id** de grupo utilizado, por ordem cronológica de criação de questionários. (Para esse fim, considera-se que um questionário-teste é sempre criado antes de seu questionário principal correspondente.) Dentro de um mesmo questionário, os grupos de A a G terão **ids** atribuídos em ordem crescente, equivalente à ordem alfabética; por fim, o grupo dedicado à questão de submissão terá o maior **id** de todos. Assim, por exemplo, se um questionário de **id** 950, criado agora, tem grupos com **ids** que vão de 1 a 8 (A-G + grupo para questão de submissão), um questionário criado daqui a 10 minutos, de **id** 500, terá grupos com **ids** que vão de 9 a 16 (A-G + questão de submissão). Observe que, para a ordem de atribuição de identificadores de grupos, o **id** dos questionários é irrelevante.

5. Agradecimentos

O presente relatório é parte do projeto de iniciação científica “Criação de um *website* para aplicação do modelo de Inventário Conceitual no contexto do aprendizado em programação”, da aluna Raysa Benatti, orientada por Ricardo Caceffo e Rodolfo Azevedo, realizado entre agosto de 2017 e julho de 2018. O projeto foi apoiado pelas seguintes instituições: CAPES, CNPq, Instituto de Computação da Unicamp, Pró-reitoria de Pesquisa da Unicamp, Unicamp e FAPESP – processo nº 2014/07502-4, Fundação de Amparo à Pesquisa

do Estado de São Paulo (FAPESP). Agradecemos aos apoiadores e aos professores Islene Garcia, Breno de França e Marco Aurélio Gerosa pelo suporte no desenvolvimento desta pesquisa.

Referências Bibliográficas

[1] 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. Technical Report IC-17-15. Campinas: Instituto de Computação da Unicamp, outubro de 2017. Disponível em: <http://www.ic.unicamp.br/~reltech/2017/17-15.pdf>.

[2] CACEFFO, R.; GAMA, G.; BENATTI, R.; APARECIDA, T.; CALDAS, T.; AZEVEDO, R. A Concept Inventory for CS1 Introductory Programming Courses in C. Technical Report IC-18-06. Campinas: Instituto de Computação da Unicamp, março de 2018. Disponível em: <http://www.ic.unicamp.br/~reltech/2018/18-06.pdf>.

[3] CACEFFO, R.; GAMA, G.; AZEVEDO, R. (2018). 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. DOI: <https://doi.org/10.1145/3159450.3159585>.

[4] ANDRADE, I.; CACEFFO, R.; SOUZA, R.; AZEVEDO, R.; BRAGA, D. (2017). Aprender a língua espanhola pelo aplicativo Vecindario: soluções para as necessidades contemporâneas. Em XXVIII SBIE – Simpósio Brasileiro de Informática na Educação. 8 páginas. 2017, UFPE, Recife, Brasil.

[5] CACEFFO, R..; GAMA, G.; MOREIRA, M.; GARCIA, I.; AZEVEDO, R. (2018) 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.

[6] LimeSurvey. Disponível em <https://www.limesurvey.org/>. Acesso em junho de 2018.

[7] MySQL Oracle. Disponível em <https://www.oracle.com/br/mysql>. Acesso em junho de 2018.

[8] ELMASRI, R.; NAVATHE, S. B.. Fundamentals of Database Systems. Sixth edition. Pearson, 2010.

[9] LimeSurvey 2.5 database layout documentation. Disponível em <https://manual.limesurvey.org/LimeSurvey_2.5_database_layout_documentation>. Acesso em julho de 2018.

[10] CACEFFO,R.; WOLFMAN,S.; BOOTH, K. 2016. 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. DOI=http://dx.doi.org/10.1145/2839509.2844559

Apêndice 1

As Tabelas 1 a 8 deste Apêndice apresentam os *scripts*, **elaborados em PHP**, utilizados no *website* desenvolvido para implementação das funcionalidades descritas neste relatório.

```
// Check if the email already exists in the userName field
// Operation: CreateUserOperation

      if ($formOperation == "CreateUser") {

          if ($debug == 1) {
              echo "<br> <b> <br> 2 - Checking if username already
      exists (only when creating an user) <br> </b> <hr> ";
          }

          $sql = "select count(*) usersFound from lime_users where
      users_name = '". $formEmail. "';";
          $result = $conn->query($sql);
          $count = 0;

          if ($result->num_rows > 0) {
              while($row = $result->fetch_assoc()) {
                  $count = $row["usersFound"];
              }

              if ($count >=1) {
                  $error = 1; // user already exists
                  if ($debug == 1) {
                      echo "2.1) Username is NOT valid: it already
      exists.";
                      echo "<br> sql = ". $sql;
                  }
              } else
                  if ($debug == 1) {
                      echo "2.1) Username is valid: it does not
      exist.";
                  }
              }
          }
      }


// 3) Create an User
// Operation: CreateUser

      if ($formOperation == "CreateNewSurvey") {
          $userId = $_POST["userID"];
      } else
          if ($formOperation == "CreateUser") {
              if ($error == 0) {
                if ($debug == 1) {
                     echo "<b> <br> <br> <br> <br>3 - Creating new user
      <br></b> <hr>";
                 }
              }
          }
      }
```

```
// 3.1) Discover the next UID on lime_users_table

      $sql = "select max(uid) max from lime_users;";
      $result = $conn->query($sql);
      $count = 0;

      if ($result->num_rows > 0) {
          while($row = $result->fetch_assoc()) {
              $count = $row["max"];
          }
      } else {
          if ($debug == 1) {
              echo "0 results";
          }
      }

      $parentId = $count;
      $userId = $parentId +1;

      if ($debug == 1) {
          echo "<br> 3.1 - parentId = " . $parentId;
          echo "<br> 3.2 - userId = "   . $userId;
      }

      $password = hash('sha256',$formPassword);

      if ($debug == 1) {
          echo "<br> 3.3 - Password = " . $password;
      }

// 3.2) Insert a new User

      if ($error == 0) {
          $username       = $formEmail; // userName is the same as the
      email!

          if ($debug == 1) {
              echo "<br> 3.3 - Username = " . $username;
          }

          $fullName       = $formName;
          $lang           = "auto";
          $email          = $formEmail;
          $htmlEditorMode = "default";
          $templateEditorMode   = "default";
          $questionSelectorMode = "default";
          $oneTimePW  = "NULL";
          $dateformat = "1";
          $created   = date('Y-m-d H:i:s');
          $modified  = date('Y-m-d H:i:s');

          $sql = "INSERT INTO lime_users (users_name, password,
      full_name, parent_id, lang, email, htmleditormode,
      templateeditormode, questionselectormode, one_time_pw,
      dateformat, created, modified, university, country, cs1_course)";
          $sql = $sql . " VALUES ('" . $username . "','" . $password .
      "','" . $fullName . "','" . $parentId . "','" . $lang . "','" .
      $email . "','" . $htmlEditorMode . "','" . $templateEditorMode .
```

```
"','";
    $sql = $sql .  $questionSelectorMode . "','" . $oneTimePW .
"','" . $dateformat . "','" . $created . "','" . $modified .
"','" . $formUniversity .  "','" . $formCountry . "','" .
$formCourse ."');";

    if ($conn->query($sql) === TRUE) {
        if ($debug == 1) {
            echo "<br> <br> 3.4 - New record in lime_users
created successfully";
        }
    } else {
        $error = 2;
        echo "<br> <b> Error: </b> <hr>";
        echo "<br> 3.4 - Error in lime_users: <br>" .
$conn->error;         }

    if ($debug == 1) {
        echo "<br> [" . $sql . "] <br>";
    }
}
```

Tabela 1. *Script* com operação de criação de um usuário.

```
// Add user permissions

if ($error == 0) {

// Permission 1 - Permissao de login

$sql1 = "INSERT INTO lime_permissions (entity, entity_id, uid,
permission, create_p, read_p, update_p, delete_p, import_p,
export_p)";

$sql1 = $sql1 . " VALUES ('global','0','" . $userId .
"','auth_db','0','1','0','0','0','0');";

if ($conn->query($sql1) === TRUE) {
    if ($debug == 1) {
        echo "<br> 3.5 - New record created in lime_permissions
successfully";
    }
} else {
    if ($debug == 1) {
        echo "<br> 3.5 - Error in lime_permissions: " . "<br>" .
$conn->error;
    }
}

if ($debug == 1) {
    echo "<br> [" . $sql1 . "]";
}


// Permission 2 - Acesso a templates
```

```
$sql2 = "INSERT INTO lime_permissions (entity, entity_id, uid,
permission, create_p, read_p, update_p, delete_p, import_p,
export_p)";

$sql2 = $sql2 . " VALUES ('template','0','" . $userId .
"','default','0','0','0','0','0','0');";

if ($conn->query($sql2) === TRUE) {
    if ($debug == 1) {
        echo "<br> <br> 3.6 - New record created in
lime_permissions successfully";
    }
} else {
    if ($debug == 1) {
        echo "<br> 3.6 - Error in lime_permissions: " . "<br>" .
$conn->error;
    }
}

if ($debug == 1) {
    echo "<br> [" . $sql2 . "]";
}
```

```
// Permission 3 - Acesso a surveys
```

```
$sql3 = "INSERT INTO lime_permissions (entity, entity_id, uid,
permission, create_p, read_p, update_p, delete_p, import_p,
export_p)";

$sql3 = $sql3 . " VALUES ('global','0','" . $userId .
"','surveys','0','0','0','0','0','1');";

if ($conn->query($sql3) === TRUE) {
    if ($debug == 1) {
        echo "<br> <br> 3.7 - New record created in
lime_permissions successfully";
    }
} else {
    if ($debug == 1) {
        echo "<br> 3.7 - Error in lime_permissions: " . "<br>" .
$conn->error;
    }
}

if ($debug == 1) {
    echo "<br> [" . $sql3 . "]";
}
```

```
}
```

Tabela 2. *Script* com operação de adição de permissões a um usuário.

```
$formName          = $_POST["name"];
$formEmail         = $_POST["email"];
$formPassword      = $_POST["password"];
$formUniversity    = $_POST["university"];
```

```
$formCountry       = $_POST["country"];
$formCourse        = $_POST["course"];
$formLanguage      = $_POST["language"];
$formOperation     = $_POST["operation"];

if ($debug == 1) {
      echo "<b> <br> 0 - Form Handling <br> </b> <hr> ";
      echo "<br> 0.1 - name = "          . $formName;
      echo "<br> 0.1 - email = "         . $formEmail;
      echo "<br> 0.2 - password = "      . $formPassword;
      echo "<br> 0.3 - university = "    . $formUniversity;
      echo "<br> 0.4 - country = "       . $formCountry;
      echo "<br> 0.5 - course = "        . $formCourse;
      echo "<br> 0.6 - language = "      . $formLanguage;
      echo "<br> 0.7 - operation = "     . $formOperation;
      echo "<br> 0.8 - survey_baseCI_ID = "     .
$survey_baseCI_ID;            echo "<br> 0.9 - survey_CI_TestID
= "     . $survey_CI_TestID;  }
```

```
// The survey IDs (base and test) relate to the $formLanguage as:
// 0: CI-EN
//    - base: 509059
//    - test: 533684
// 1: CI-PT
//    - base: 837557
//    - test: 580884
```

```
if ($formLanguage === "C (English)") {
    $survey_baseCI_ID     = 1390;     // BASE Survey. The new
survey will be a copy of it!
    $survey_CI_TestID     = 383072;     // Survey to be
used/tested by instructors
}
else
if ($formLanguage === "C (Portuguese)") {
    $survey_baseCI_ID     = 96605;     // BASE Survey. The new
survey will be a copy of it!
    $survey_CI_TestID     = 44183 ;    // Survey to be
used/tested by instructors
}
else {
   #error = 1;
}
```

Tabela 3. Identificação do *survey*-base a partir da escolha feita pelo usuário no formulário do *site*.

```
// 4) Copy survey
// Operation: All operations
```

```
 if ($debug == 1) {
     echo "<br> <b> <br> 4 - Copying Survey <br> </b> <hr> ";
 }
```

```
// 4.1) Find a correct (not yet used) SID
```

```
// While a valid SID is not found

while ($count != 0) {

    $newSID = rand(1,999999);

    $sqlCheck = "select count(sid) surveysFound from
lime_surveys where sid = ". $newSID;

    $result = $conn->query($sqlCheck);

    if ($result->num_rows > 0) {
        while($row = $result->fetch_assoc()) {
            $count = $row["surveysFound"];
        }

        if ($count >=1) {
        // newSID already exists

            if ($debug == 1) {
                echo "<br> 4.0) newSID is NOT valid: it already
exists.";
                echo "<br> sql = ". $sqlCheck;
            }
        }
        else {
            if ($debug == 1) {
            // newSID does not exist
                echo "<br> 4.0) newSID is valid: it does not
exist. newSID = ". $newSID;
                echo "<br> sql = ". $sqlCheck;
            }
        }
    }
}
```

```
// 4.2) Copy register in the lime_surveys table

   /* SQL command:
   insert into your_table (c1, c2, ...)
   select c1, c2, ...
   from your_table
   where id = 1
   */

// New survey is created with the current date (otherwise it
would have the date from the base survey!)
$created   = date('Y-m-d');

$sql4 = "INSERT INTO lime_surveys (sid, owner_id, admin, active,
expires, startdate, adminemail, anonymized, faxto, format,
savetimings, template, language, additional_languages,
datestamp, usecookie, allowregister, allowsave,
autonumber_start, autoredirect, allowprev, printanswers, ipaddr,
refurl, datecreated, publicstatistics, publicgraphs, listpublic,
htmlemail, sendconfirmation, tokenanswerspersistence,
assessments, usecaptcha, usetokens, bounce_email,
attributedescriptions, emailresponseto, emailnotificationto,
```

```
tokenlength, showxquestions, showgroupinfo, shownoanswer,
showqnumcode, bouncetime, bounceprocessing, bounceaccounttype,
bounceaccounthost, bounceaccountpass, bounceaccountencryption,
bounceaccountuser, showwelcome, showprogress, questionindex,
navigationdelay, nokeyboard,alloweditaftercompletion,
googleanalyticsstyle, googleanalyticsapikey,
programming_language)";

$sql4 = $sql4 . "  select ". $newSID. "," .$userId. ", admin,
active, expires, startdate, adminemail, anonymized, faxto,
format, savetimings, template, language, additional_languages,
datestamp, usecookie, allowregister, allowsave,
autonumber_start, autoredirect, allowprev, printanswers, ipaddr,
refurl, '". $created . "', publicstatistics, publicgraphs,
listpublic, htmlemail, sendconfirmation,
tokenanswerspersistence,  assessments, usecaptcha, usetokens,
bounce_email, attributedescriptions, emailresponseto,
emailnotificationto, tokenlength, showxquestions, showgroupinfo,
shownoanswer, showqnumcode, bouncetime, bounceprocessing,
bounceaccounttype,  bounceaccounthost, bounceaccountpass,
bounceaccountencryption, bounceaccountuser, showwelcome,
showprogress, questionindex, navigationdelay,
nokeyboard,alloweditaftercompletion, googleanalyticsstyle,
googleanalyticsapikey, '" .$formLanguage. "' ";

$sql4 = $sql4 . "  from lime_surveys ";

$sql4 = $sql4 . "  where sid = ". $survey_baseCI_ID;

if ($conn->query($sql4) === TRUE) {
    if ($debug == 1) {
        echo "<br> <br> 4.1 - New record created in lime_surveys
successfully";
    }
} else {
    if ($debug == 1) {
        echo "<br> 4.1 - Error in lime_surveys: " . "<br>" .
$conn->error;
    }
}

if ($debug == 1) {
    echo "<br> [" . $sql4 . "]";
}
```

Tabela 4. Criação, no banco de dados, um novo registro em **lime_surveys**, com o novo identificador.

```
// 4.2) Copy register in the lime_surveys_languagesettings

$sql5 = "INSERT INTO lime_surveys_languagesettings
(surveyls_survey_id, surveyls_language, surveyls_title,
surveyls_description, surveyls_welcometext,   surveyls_endtext,
surveyls_url, surveyls_urldescription,
```

```
surveyls_email_invite_subj, surveyls_email_invite,
surveyls_email_remind_subj, surveyls_email_remind,
surveyls_email_register_subj, surveyls_email_register,
surveyls_email_confirm_subj, surveyls_email_confirm,
surveyls_dateformat, surveyls_attributecaptions,
email_admin_notification_subj,email_admin_notification,
email_admin_responses_subj,  email_admin_responses,
surveyls_numberformat,  attachments)";

$sql5 = $sql5 . "  select ". $newSID.", surveyls_language,
surveyls_title, surveyls_description, surveyls_welcometext,
surveyls_endtext, surveyls_url, surveyls_urldescription,
surveyls_email_invite_subj, surveyls_email_invite,
surveyls_email_remind_subj, surveyls_email_remind,
surveyls_email_register_subj, surveyls_email_register,
surveyls_email_confirm_subj, surveyls_email_confirm,
surveyls_dateformat, surveyls_attributecaptions,
email_admin_notification_subj,email_admin_notification,
email_admin_responses_subj,  email_admin_responses,
surveyls_numberformat,  attachments ";

$sql5 = $sql5 . "  from lime_surveys_languagesettings ";

$sql5 = $sql5 . "  where surveyls_survey_id =  ".
$survey_baseCI_ID;

if ($conn->query($sql5) === TRUE) {
    if ($debug == 1) {
        echo "<br> <br> 4.2a - New record created in
lime_surveys_languagesettings: successfully";
    }
} else {
    if ($debug == 1) {
        echo "<br> 4.2a - Error in
lime_surveys_languagesettings: " . "<br>" . $conn->error;
    }
}

if ($debug == 1) {
    echo "<br> [" . $sql5 . "]";
}
```

Tabela 5. Inserção das informações copiadas em um novo registro.

```
// Algoritmo para cópia das questões e grupos.

// 4.3) Liste todas as questões do Lime_Survey de origem (sid =
$survey_baseCI_ID), ordenadas por GID, e as coloque em um vetor.

// 4.4) Para cada questão no vetor:
//        4.4a) Na primeira questão de um grupo, faça a cópia desse grupo
```

```
na tabela correspondente.
//        4.4b) Obtenha o id deste novo grupo.
//        4.4c) Update nos gids da tabela lime_assessment. Registros ja
existem, mas se referem ao survey de origem.
//        4.4d) Faça a cópia da questão na tabela correspondente,
informando o id do novo grupo e o id do novo survey.
```

```
// 4.3) Liste todas as questões do Lime_Survey de origem,
ordenadas por GID, e as coloque em um vetor.

$sql_questions = " select qid, parent_qid, sid, gid, type,
title, question, preg, help, other,
      mandatory, question_order, language, scale_id,
same_default, relevance, modulename
      from lime_questions where sid = ".$survey_baseCI_ID. "
order by gid ";

if ($debug == 1) {
    echo "<br> <br> 4.3) sql = ". $sql_questions;
}

$result_sql_questions = $conn->query($sql_questions);
$current_group = -1; // current group id (related to the
original survey)
$old_group     = -1; // old group id  (related to the original
survey)
$newGroup      =  0; // boolean: was a new group found?
$newGid = -1;        // newGid created (related to the new
survey)
$count = 0;

if ($debug == 1) {
    echo "<br> <br> 4.3) result_sql_questions->num_rows = ".
$result_sql_questions->num_rows;
}
```

```
if ($result_sql_questions->num_rows > 0) {

// 4.4) Para cada questão no vetor
```

```
//     4.4a) Na primeira questão de um grupo, faça a cópia
desse grupo na tabela correspondente.

while($row = $result_sql_questions->fetch_assoc()) {

    // Recupere o grupo atual e verifique se é um novo grupo
    $current_group = $row["gid"];
    $current_questionID = $row["qid"];

    // Se é o primeiro grupo, trata-se de um novo grupo!
    if ($old_group == -1) {
        $old_group = $current_group;
        $newGroup  = 1;
    }
    else {
    // Não é o primeiro grupo.
    // Será um novo grupo de $current_group for diferente de
$old_group
        if ($current_group != $old_group) {
```

```
            // Novo grupo detectado
            $newGroup = 1;
            // Atualize $oldGroup
            $old_group = $current_group;

        }
        else {
            // Mesmo grupo
            $newGroup  = 0;
        }
    }

    // Se o grupo atual for um novo grupo, devemos inseri-lo na
tabela lime_groups e atualizar a variável $newGid.
    // Além disso, devemos atualizar o gid de lime_assessments
para o novo grupo criado.


    if ($newGroup == 1) {
        $sql6 = "INSERT INTO lime_groups (sid, group_name,
group_order, description, language, randomization_group,
grelevance)";
        $sql6 = $sql6 . "  select ". $newSID.", group_name,
group_order, description, language, randomization_group,
grelevance ";
        $sql6 = $sql6 . "  from lime_groups ";
        $sql6 = $sql6 . "  where gid = ". $current_group;
        if ($conn->query($sql6) === TRUE) {
            if ($debug == 1) {
                echo "<br> <br> 4.5a - New record created in
lime_groups successfully";
            }
        } else {
            if ($debug == 1) {
                echo "<br> 4.5a - Error in lime_groups: " .
"<br>" . $conn->error;
            }
        }

    if ($debug == 1) {
        echo "<br> [" . $sql6 . "]";
    }
```

```
//     4.4b) Obtenha o id deste novo grupo.


// Descobre id do novo grupo criado ($newGid)
$sql7 = "select max(gid) m from lime_groups;";
$result_sql7 = $conn→query($sql7);

if ($result_sql7->num_rows > 0) {
    while($row = $result_sql7->fetch_assoc()) {
        $newGid = $row["m"]; // updates the group
    }
}
else
    if ($debug == 1) {
        echo "<br> 4.5b -  0 results";
    }
```

```
    if ($debug == 1) {
        echo "<br> 4.5b - newGid = " . $newGid;
        echo "<br> [" . $sql7 . "]";
    }
}
```

```
//        4.4c) Update nos gids da tabela lime_assessment.
Registros ja existem, mas se referem ao survey de origem.

$sql_assessments = "UPDATE lime_assessments  SET gid = ".
$newGid. " WHERE sid = ". $newSID. " and gid =
".$current_group;

if ($conn->query($sql_assessments) === TRUE) {
    if ($debug == 1) {
        echo "<br> <br> 4.2c - New records created in
lime_surveys_assessments: successfully";
    }
} else
    if ($debug == 1) {
        echo "<br> 4.2c - Error in lime_surveys_assessments:
" . "<br>" . $conn->error;
    }
    if ($debug == 1) {
        echo "<br> [" . $sql_assessments . "]";
    }
}
// Fim: novo grupo foi inserido. Novo gid está na variável
$newGid.
```

```
//        4.4d) Faça a cópia da questão na tabela
correspondente, informando o id do novo grupo e o id do novo
survey.

$sql8 = "INSERT INTO lime_questions (parent_qid, sid, gid,
type, title, question, preg, help, other, mandatory,
question_order, language, scale_id, same_default, relevance,
modulename, real_parent_qid)";

$sql8 = $sql8 . "  select lq.parent_qid, ". $newSID.",".
$newGid. ", lq.type, lq.title, lq.question, lq.preg, lq.help,
lq.other, lq.mandatory, lq.question_order, lq.language,
lq.scale_id, lq.same_default, lq.relevance, lq.modulename,
lq.qid ";

$sql8 = $sql8 . "  from lime_questions lq ";

$sql8 = $sql8 . "  where lq.qid = ". $current_questionID;

if ($conn->query($sql8) === TRUE) {
    if ($debug == 1) {
        echo "<br> <br> 4.5d - New record created in
lime_questions successfully";
    }
} else {
    if ($debug == 1) {
```

```
            echo "<br> 4.5d - Error in lime_questions: " .
        "<br>" . $conn->error;
            }
        }

        if ($debug == 1) {
            echo "<br> [" . $sql8 . "]";
        }

        } // fim while
    } // fim if "Para cada questão no vetor"
    else
        if ($debug == 1) {
            echo "<br> <br> 4.4) Not found any questions for survey ".
$newSID;
        }
    }
```

Tabela 6. Algoritmo e *script* de cópia das informações sobre questões e grupos para o novo questionário criado.

```
// 4.5) Copy lime_answers (question's choices)
    /*
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| qid              | int(11)     | NO   | PRI | 0       |       |
| code             | varchar(5)  | NO   | PRI |         |       |
| answer           | text        | NO   |     | NULL    |       |
| sortorder        | int(11)     | NO   | MUL | NULL    |       |
| assessment_value | int(11)     | NO   |     | 0       |       |
| language         | varchar(20) | NO   | PRI | en      |       |
| scale_id         | int(11)     | NO   | PRI | 0       |       |
+------------------+-------------+------+-----+---------+-------+
*/

        $sql9 = "INSERT INTO lime_answers (qid, code, answer,
        sortorder, assessment_value, language, scale_id)";
        $sql9 = $sql9 . "  select lq.qid, la.code, la.answer,
        la.sortorder, la.assessment_value, la.language, la.scale_id ";
        $sql9 = $sql9 . "  from lime_answers la, lime_questions lq ";
        $sql9 = $sql9 . "  where la.qid = lq.real_parent_qid AND
        lq.sid = " .$newSID;

        if ($conn->query($sql9) === TRUE) {
            if ($debug == 1) {
                echo "<br> <br> 4.6 - New record created in
        lime_answers successfully";
            }
        } else {
            if ($debug == 1) {
                echo "<br> 4.6 - Error in lime_answers: " . "<br>" .
        $conn->error;
            }
        }

        if ($debug == 1) {
            echo "<br> [" . $sql9 . "]";
        }
```

Tabela 7. Cópia das respostas.

```
// 4.6) To active the Survey, we have to create a table
lime_survey_XXXXXX, where
// XXXXXX is the survey id  (sid).

/* The fields are:
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| id            | int(11)     | NO   | PRI | NULL    | auto_increment |
| token         | varchar(35) | YES  | MUL | NULL    |                |
| submitdate    | datetime    | YES  |     | NULL    |                |
| lastpage      | int(11)     | YES  |     | NULL    |                |
| startlanguage | varchar(20) | NO   |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+


There also one extra field for each question related to the survey
(to keep the answers related to those questions)

* Field is like "12345X22X333name":
* 12345 = surveyID
* 22 = groupID
* 333 = questionID
* name = answer code (only shown for certain question types)


Example: CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
      */
```

```
$sql10 = "CREATE TABLE lime_survey_".$newSID . " ( ";
$sql10 = $sql10 . "  id            int(11) NOT NULL
AUTO_INCREMENT,";
$sql10 = $sql10 . "  token         varchar(35),";
$sql10 = $sql10 . "  submitdate    datetime,";
$sql10 = $sql10 . "  lastpage      int(11),";
$sql10 = $sql10 . "  startlanguage varchar(20) NOT NULL ";

// Insert one new line for each question related to this
Survey
$sql_GID_SID = " select gid g, qid q from lime_questions lq
where lq.sid = ". $newSID;
$result = $conn->query($sql_GID_SID);
$g = 0;
$q = 0;
$count = 0;

if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        $g = $row["g"];
        $q = $row["q"];
        $count++;
```

```
        $sql10 = $sql10 . ", ". $newSID."X".$g."X".$q."
varchar(5) ";
    }
    $sql10 = $sql10. ",  PRIMARY KEY (ID), KEY (token) );";
}
else  {
    if ($debug == 1) {
        echo "4.7) Not found any questions for survey ".
$newSID;
    }
}

if ($conn->query($sql10) === TRUE) {
    if ($debug == 1) {
        echo "<br> <br> 4.7 - New table created:
lime_survey_".$newSID;
    }
} else {
    if ($debug == 1) {
        echo "<br> 4.7 - Error creating table:
lime_survey_".$newSID. " <br>". $conn->error;
    }
}

if ($debug == 1) {
    echo "<br> [" . $sql10 . "]";
}
```

Tabela 8. Criação de uma nova tabela no banco, **lime_survey_X**.