Sistemas de Informação Distribuídos

Licenciaturas em Engenharia Informática e Informática e Gestão de Empresas

2017-2018, Segundo Semestre

Monitorização de Culturas em Laboratório

Auditoria e Migração

Identificação do grupo autor da especificação (Etapa A): 05

|  |  |  |
| --- | --- | --- |
| Número | Nome | Foto |
| 78530 | João Garcez |  |
| 78167 | Afonso Miao |  |
| 77974 | Francisco Arruda |  |
| 78107 | Miguel Gil |  |
| 78318 | Beatriz Patrício |  |
| 78760 | Pedro Norte |  |
| Especificação:  PHP  Ficheiro | | |

Identificação do grupo autor da implementação (Etapas B e C): \_\_\_19\_\_\_

|  |  |  |
| --- | --- | --- |
| Número | Nome | Foto |
| 73462 | Pedro Almeida |  |
| 77959 | Hugo Martins |  |
| 60343 | Raquel Fernandes |  |
| 68840 | Daniel Leal |  |
| 72777 | Daniel Rodrigues |  |
| 68934 | Manuel Sepúlveda |  |
| Especificação: PHP Ficheiro  Implementação: PHP Ficheiro | | |

Instruções

Estas instruções são de cumprimento obrigatório. Relatórios que não cumpram as indicações serão penalizados na nota final.

* Podem (e em várias situações será necessário) ser adicionadas novas páginas ao relatório, mas não podem ser removidas páginas. Se uma secção não for relevante, fica em branco, não pode ser removida;
* Todas as secções têm que iniciar-se no topo de página (colocar uma quebra de página antes);
* A paginação tem de ser sequencial e não ter falhas;
* O índice tem de estar atualizado;
* Na folha de rosto (anterior) têm de constar toda a informação solicitada, nomeadamente todas as fotografias de todos os elementos dos dois grupos. É obrigatório que caiba tudo numa única página;
* A formatação das “zonas” (umas sombreadas outras não sombreadas) não pode ser alterada;
* Nas etapas A e B (até secção 1.4 inclusive), o grupo que primeiro edita o documento (Etapa A) **apenas escreve nas zonas não sombreadas**, e o outro grupo apenas escreve nas zonas sombreadas;
* A etapa C é apenas preenchida pelo grupo que recebe o presente documento do outro grupo. Nas secções 2.1, 2.2, 2.3 e 2.6 deve colocar nas zonas não sombreadas a especificação que entregou ao outro grupo (sem alteração, *copy e paste*),
* As restantes secções são preenchidas normalmente pelo grupo que recebe o presente documento do outro grupo.

Índice

[1 Etapa A e B 1](#_1fob9te)1

[1.1 Esquema relacional da base de Dados Mysql (origem) 11](#_3znysh7)

[1.1.1 Apreciação Crítica e esquema relacional implementado 1](#_2et92p0)3

[1.2 Utilizadores Base de Dados de Origem 13](#_tyjcwt)

[1.2.1 Apreciação Crítica a Gestão de Utilizadores Base de Dados de Origem 1](#_3dy6vkm)6

[1.3 Gestão de Logs 1](#_1t3h5sf)7

[1.3.1 Triggers de suporte à criação de logs Base de Dados de Origem 1](#_4d34og8)7

[1.3.1.1 Apreciação Crítica de triggers para gestão de logs 1](#_17dp8vu)9

[1.3.1.2 Triggers Implementados para gestão de logs](#_3rdcrjn) 22

[1.3.2 Stored Procedures de suporte à criação de logs (](#_26in1rg)**se relevante**) 25

[1.3.2.1 Apreciação Crítica de Stored Procedures de suporte à criação de logs 2](#_lnxbz9)5

[1.3.2.2 Stored Procedures Implementados de suporte à criação de logs 22](#_35nkun2)

[1.4 Migração entre Bases de Dados 2](#_1ksv4uv)7

[1.4.1 Esquema relacional da base de Dados Mysql (destino) 2](#_44sinio)7

[1.4.1.1 Apreciação Crítica e esquema relacional implementado 2](#_2jxsxqh)8

[1.4.2 Forma de Migração 2](#_z337ya)9

[1.4.2.1 Apreciação Crítica à especificação da forma de migração 3](#_3j2qqm3)4

[1.4.3 Gestão de Utilizadores de Suporte à Migração (origem e/ou destino) 3](#_1y810tw)5

[1.4.3 Apreciação Crítica à especificação da Gestão de Utilizadores 3](#_4i7ojhp)6

[1.4.4 Triggers de suporte à migração de dados (origem e/ou destino) (](#_2xcytpi)**se relevante**) 38

[1.4.4.1 Apreciação Crítica de triggers de suporte à migração de dados 3](#_1ci93xb)9

[1.4.4.2 Triggers Implementados de suporte à migração de dados](#_3whwml4) 40

[1.4.5 Stored Procedures de suporte à migração de dados](#_2bn6wsx) 41

[1.4.5.1 Apreciação Crítica de Stored Procedures de suporte à migração de dados](#_qsh70q) 42

[1.4.5.2 Stored Procedures Implementados de suporte à migração de dados](#_3as4poj) 43

[1.4.6 Eventos de suporte à migração de dados 4](#_1pxezwc)4

[1.4.6.1 Apreciação Crítica de Eventos 4](#_49x2ik5)5

[1.4.6.2 Eventos Implementados 4](#_2p2csry)6

[1.4.7 PHP suporte à migração de dados (se relevante) 4](#_147n2zr)7

[1.4.7.1 Apreciação Crítica ao PHP especificado 4](#_3o7alnk)8

[1.4.7.2 PHP Implementado 4](#_23ckvvd)9

[1.5 Avaliação Global de especificações da Etapa A](#_ihv636) 50

[2 Etapa C (Especificação e Implementação do Próprio Grupo)](#_32hioqz) 52

[2.1 Especificação do Esquema relacional da base de Dados Origem](#_1hmsyys) 52

[2.2 Especificação de Utilizadores](#_41mghml) 61

[2.3 Especificação de Gestão de Logs](#_2grqrue) 62

[2.3.1 Triggers de suporte à gestão de logs](#_vx1227) 62

[2.3.2 Stored Procedures de suporte à gestão de logs](#_3fwokq0) 63

[2.4 Avaliação da especificação do próprio grupo Gestão de Logs](#_1v1yuxt) 68

[2.5 Implementação Gestão de Logs](#_4f1mdlm) 69

[2.5.1 Utilizadores implementados](#_2u6wntf) 69

[2.5.2 Lista de Triggers](#_19c6y18) 70

[2.5.3 Triggers Implementados](#_3tbugp1) 71

[2.5.4 Lista de Stored Procedures](#_28h4qwu) 85

[2.5.5 Stored Procedures Implementados](#_nmf14n) 85

[2.6 Especificação de Migração entre Bases de Dados](#_37m2jsg) 89

[2.6.1 Esquema relacional da base de Dados Mysql especificada (destino)](#_1mrcu09) 89

[2.6.2 Forma de Migração Especificada](#_46r0co2) 91

[2.6.3 Utilizadores Especificados](#_2lwamvv) 97

[2.6.4 Triggers de suporte à migração de dados especificados](#_111kx3o) 98

[2.6.5 Stored Procedures de suporte à migração de dados especificados](#_3l18frh) 98

[2.6.6 Eventos de suporte à migração de dados especificados](#_206ipza) 100

[2.6.7 PHP de suporte à migração de dados especificado](#_4k668n3) 101

[2.7 Avaliação das especificações do próprio grupo Migração](#_2zbgiuw) 103

[2.8 Implementação da Migração de Dados](#_1egqt2p) 104

[2.8.1 Utilizadores Implementado](#_3ygebqi) 104

[2.8.2 Lista Triggers](#_2dlolyb) 105

[2.8.3 Triggers Implementados](#_sqyw64) 105

[2.8.4 Lista de Stored Procedures](#_3cqmetx) 106

[2.8.5 Stored Procedures Implementados](#_1rvwp1q) 107

[2.8.6 Lista Eventos](#_4bvk7pj) 108

[2.8.7 Eventos Implementados](#_2r0uhxc) 112

[2.8.8 PHP Implementado](#_1664s55) 114

[Avaliação Global da Qualidade das Especificações](#_3q5sasy) 119

[2.9 Comparação de Implementações (ficheiro versos PHP)](#_25b2l0r) 121

[2.9.1 Eficiência de Migração](#_kgcv8k) 122

[2.9.2 Robustez](#_34g0dwd) 124

[2.9.3 Flexibilidade / Dependência](#_1jlao46) 125

[2.9.4 Segurança](#_43ky6rz) 126

[2.10 Auditoria de Dados (base de dados origem)](#_2iq8gzs) 127

Monitorização de Culturas em Laboratório

Um laboratório de investigação de um departamento de biologia necessita de um sistema para monitorizar a evolução de culturas. Mais concretamente, pretende acompanhar a temperatura e luz a que as culturas estão sujeitas, bem como detectar/antecipar potenciais problemas.

Numa estufa estão colocados dois sensores que medem a temperatura e quantidade de luz ambiente (que afecta todas as culturas existentes na estufa).

Periodicamente os investigadores dirigem-se à estufa para efectuarem manualmente várias medições de variáveis (humidade, ph, etc) e registá-las num computador que está localizado na estufa. Cada cultura tem um único investigador responsável e apenas ele pode criar, actualizar e consultar os dados de medições das suas culturas. Esta *protecção de dados* é um aspecto importante do sistema. Nem todas as variáveis necessitam serem lidas e registadas. Para cada cultura o investigador decide quais delas devem ser lidas, e regista no sistema qual o intervalo de valores que considera “normal” para o par variável/cultura.

Por exemplo, para as culturas hidropónicas de pimento e tomate, fazem-se medições do nível de concentração de mercúrio e chumbo. Mas numa cultura de bactérias onde se adicionaram antibióticos o que faz sentido medir é o índice de concentração das bactérias, não faz sentido medir o nível de concentração de mercúrio e chumbo.

**Alertas**

Existem dois tipos de alertas:

a) alertas resultantes das medições das variáveis. O investigador, quando insere manualmente um valor de uma medição, caso o valor ultrapasse os limites será alertado com um aviso (no próprio computador) e com uma mensagem para o telemóvel (por vezes o investigador pede a um colega para efectuar a medição, sendo por isso aconselhável que o alerta não apareça somente no monitor do computador).

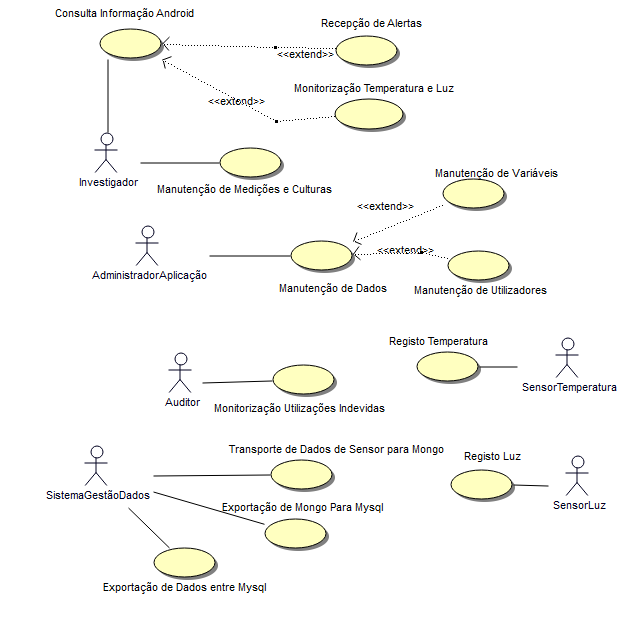
b) Alertas resultantes dos sensores de temperatura e luminosidade. O sistema sabe, para toda a estufa, o intervalo de valores de luminosidade e temperatura adequado (igual para todas as culturas). Se o sensor detectar que os valores vão ser ultrapassados deve notificar por telemóvel o investigador.

Cada investigador deverá ter a possibilidade de, através de um telemóvel, monitorizar a evolução da temperatura e luminosidade (não apenas a última leitura, mas a evolução na última hora ou horas) e receber os dois tipos de alertas.

**Registo de Acessos**

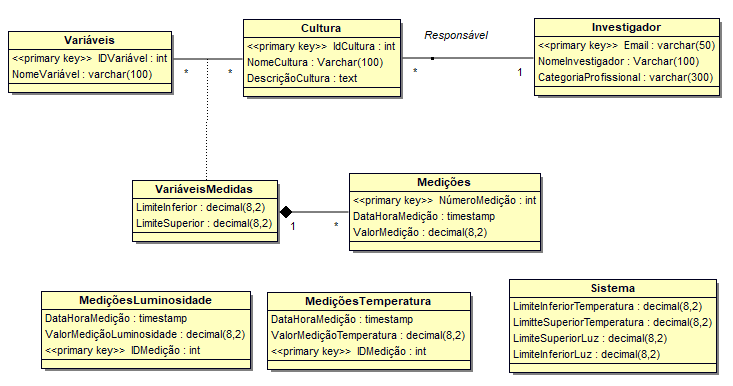
É necessário guardar na base de dados (mysql) o registo de todas as operações de escrita sobre todas as tabelas (quais dados foram alterados/inseridos/apagados, quando e por quem) e o registo de operações de consulta apenas sobre a tabela Medições. Esse registo de alterações (*log) é exportado* incrementalmente(apenas informação nova) e periodicamente para uma base de dados autónoma (também mysql). Através dessa base de dados (apenas de consulta) um auditor pode analisar se ocorreram utilizações abusivas dos dados (por exemplo, quem é que alterou limites de temperatura de uma cultura, etc.).

**Diagrama de Use Case Global**



No presente relatório apenas são contemplados os use case “Exportação Dados entre Mysql”, “Monitorização de Utilizações Indevidas” e “Manutenção de Utilizadores” (apenas a componente Mysql/Privilégios/SP/Triggers)). A componente Java (manutenção de culturas, medições, variáveis e utilizadores) não é especificada neste relatório (diz respeito à UC Eng. Prog II). Nenhum use case pressupõe a programação de formulários.

Diagrama de Classes de Suporte à Base de Dados



Esquema de Migração



# Etapa A e B

## Esquema relacional da base de Dados Mysql (origem)

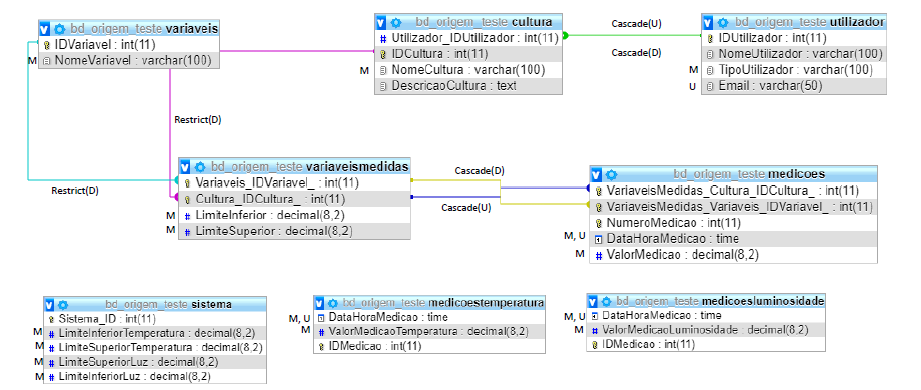
<Tem de incluir as tabelas e atributos que suportam a auditoria de dados. Deverá conter as regras de integridade das chaves estrangeiras e a indicação de atributos Unique e obrigatórios.

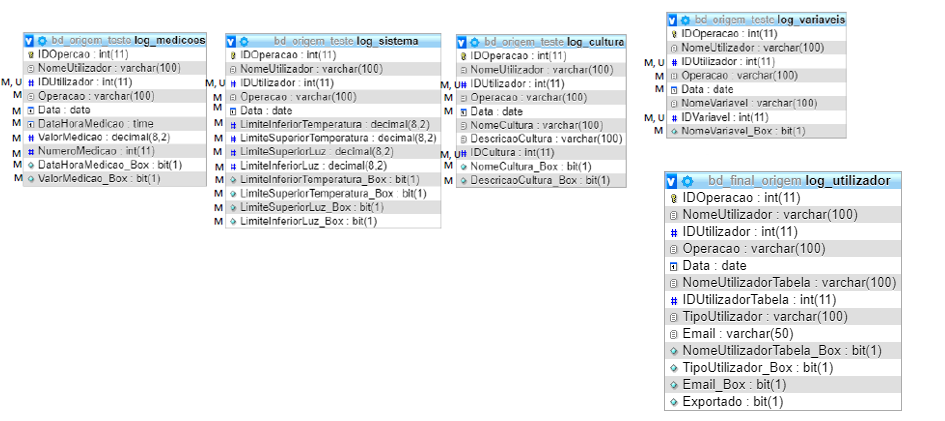
A base de dados deve respeitar o diagrama de classes (deverão ser criadas as tabelas para registo de logs). Excepcionalmente poderão ser adicionados, alterados ou removidos campos, desde que devidamente justificado no relatório. O esquema deverá ser apresentado exclusivamente através de um esquema semelhante ao aqui exemplificado:

Na ligação, junto a cada tabela, coloca-se o(s) atributo(s) que assegura a ligação entre as tabelas (sigla\_hotel em ambas as talelas, no exemplo) e a operação de integridade referencial escolhida para a operação de Delete (D) e Update (C).

À esquerda de cada campo de preenchimento obrigatório devem indicar M (de Mandatory), tal como exemplificado (não é necessário indicar nas chaves primárias). Os campos com valor único devem ser assinalados com a letra U.

Caso seja pertinente poderão ser adicionados comentários a justificar opções pouco óbvias.>





O nome da tabela “Investigador” foi trocado para “Utilizador”. Foi adicionado o campo “tipo utilizador” na tabela utilizador para podermos diferenciar os investigadores dos administradores.

### Apreciação Crítica e esquema relacional implementado

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Boa  Breve Justificação:  Boa caracterização da BD no geral.  Chave primária - confusa - “IDSistema” na tabela “Sistema”.  Chave composta criada na tabela Medições - não aplicável a nível gráfica.  Erro de especificação “Exportado” só está no log Utilizador o que contraria a especificação escrita pelos mesmos.  Fraca especificação de relações entre colunas - medicoes e variaveismedidas.  Uso de ON DELETE CASCADE na tabela utilizador irá eliminar a linha da tabela cultura associada a esse Investigador eliminado igualmente a cultura. Uma vez que não existe possibilidade de nenhum utilizador criar um tipo de cultura não irá existir mais essa cultura.  Possíveis problemas relacionados com tabelas Log - **1.4.1.1**  Foram feitas alterações? (Sim/Não): Não.  **Novo Esquema (assinale e justifique as alterações)**  <Apenas preencher caso tenham procedido a alterações> |

1.2 Utilizadores Base de Dados de Origem

<Nesta secção deverá ser explicado de que forma deverá ser feita a manutenção (e.g., a criação e privilégios) de utilizadores Mysql. Nomeadamente deverá ser indicado, para cada tipo de utilizador, que privilégios ele tem sobre que tabelas e Stored procedures (caso sejam referenciados na secção 1.3.2).

Sugestão de tabela

|  |  |  |
| --- | --- | --- |
| **Tabela** | **Tipo de Utilizador** | |
| Administrador | Investigador |
| Cultura | E | E, L |
| Log\_cultura | L | - |
| Log\_medicoes | L | - |
| Log\_sistema | L | - |
| Log\_utilizador | L | - |
| Log\_variaveis | L | - |
| Medições | L | E, L |
| MediçõesLuminosidade | L | L |
| MediçõesTemperatura | L | L |
| Sistema | L | L |
| Utilizador | E, L, E | - |
| Variaveis | E, L | L |
| VariaveisMedidas | L | L |
| **Stored Proc.** |  |  |
| Criar\_Utilizador | X | - |

Em que E=Escrita, L=Leitura, X=Executar e - = sem permissões

Caso seja pertinente poderão ser adicionados comentários explicar a estratégia a seguir.>

A estratégia a seguir vai ser criar duas roles: **grupo administradores** e **grupo investigadores**. Cada role vai ter as seguintes funcionalidades:

* **Grupo administradores:** Vai ter permissões create, read, update e delete (CRUD) sobre a tabela “utilizador”, ou seja, todos os utilizadores que pertencerem a este role vão ter permissões para criar, consultar, atualizar e apagar informações da tabela. Para além disso, também vão ter permissões create, read e delete (CRD) sobre a tabela “variáveis”.
* **Grupo investigadores:** Vai ter permissões create, read, update (CRU) sobre a tabela “cultura”. Ou seja, os utilizadores que pertencem a esta role apenas podem criar, consultar e atualizar a tabela “culturas”. E o grupo vai ter também permissões create, read, update, delete sobre a tabela medições

Quando o utilizador root criar um user vai ser preciso associar uma role ao user criado:

* GRANT grupoAdministradores to userTeste@localhost;
* GRANT grupoInvestigadores to userTeste@localhost;

Depois, quando entrarmos com a conta do novo user para conseguirmos usufruir das funcionalidades, dependendo do tipo de utilizador, é preciso ativar a role associada a esse user:

* SET ROLE grupoAdministradores;
* SET ROLE grupoInvestigadores;

Cada vez que se cria um utilizador com o comando “CREATE USER” é adicionado esse utilizador à tabela nativa do phpmyadmin “mysql.users” de seguida é necessário correr um stored procedure (criar\_utilizador) para adicionar o utilizador também à tabela “utilizador”. Após correr o stored procedure, é importante associar o utilizador a role que lhe é destinado.

O stored procedure “criar\_utilizador” vai ter os seguintes parâmetros de entrada:

* Nome do utilizador;
* Tipo de utilizador – administrador ou investigador;
* Email do utilizador.

### Apreciação Crítica a Gestão de Utilizadores Base de Dados de Origem

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Fraco  **Análise crítica (clareza, completude, rigor):**  No nosso ponto de ver o Admin não deveria ter permissões de leitura nas tabelas log, mas só de escrita nas tabelas log\_cultura e log\_utilizador.  Pensamos que esta seria uma boa opção:    (Não representamos a secção das SP’s, pois não averiguamos possíveis irregularidades)  Boa especificação das Roles dos Investigadores**.**  **Solução Implementada:**  A solução implementada foi exatamente igual à especificada pelo grupo. |

## Gestão de Logs

### Triggers de suporte à criação de logs Base de Dados de Origem

<Caso seja pertinente poderão ser adicionados comentários explicar a estratégia a seguir.>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome Trigger | Tabela | Tipo de Operação (I,U,D) | Evento  (A,B) | Notas (apenas indicar aquilo que não seja óbvio) |
| Trigger\_variaveis\_insert | Variáveis | I | A |  |
| Trigger\_variaveis\_Delete | Variáveis | D | A |  |
| Trigger\_variaveis\_update | Variáveis | U | A |  |
| Trigger\_cultura\_insert | Cultura | I | A |  |
| Trigger\_cultura\_delete | Cultura | D | A |  |
| Trigger\_cultura\_update | Cultura | U | A |  |
| Trigger\_medicoes\_insert | Medições | I | A |  |
| Trigger\_medicoes\_delete | Medições | D | A |  |
| Trigger\_medicoes\_update | Medições | U | A |  |
| Trigger\_sistema\_insert | Sistema | I | A |  |
| Trigger\_sistema\_delete | Sistema | D | A |  |
| Trigger\_sistema\_Update | Sistema | U | A |  |

Cada trigger vai introduzir a operação que ocorreu na sua respetiva tabela de log, por exemplo: se introduzirmos uma nova cultura na tabela cultura, o trigger de insert vai ter que escrever na tabela log cultura a operação que ocorreu.

As tabelas log devem conter obrigatoriamente:

* Nome do utilizador que fez a operação;
* ID do utilizador;
* O tipo de operação que foi feita;
* A data em que ocorreu a ação;
* Os campos da tabela original;
* De forma a saber quais os campos que foram alterados, sugerimos a cópia dos campos da tabela original, mas que vão ser booleanos. Sendo assim, se existir um campo X vai ter que haver um campo X\_Boolean. Ou seja, se o campo X for alterado então o campo X\_Boolean fica a true, caso contrário fica a false;
* Campo exportado (boolean) – este campo vai ser necessário para sabermos se um certo dado já foi exportado ou não, assim, conseguimos garantir que exportamos apenas os novos dados e não estamos a repetir informação.

#### Apreciação Crítica de triggers para gestão de logs

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): BOA  Breve Justificação: Falha na especificação dado que não houve comentários relativos à estratégia a seguir. Embora não seja especificado que o trigger seja AFTER a acção, vamos assumir que o é, dado que a sua ausência gera menor fiabilidade.  **Lista de Triggers (para cada trigger assinalar com x em célula correspondente)**     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Implementado de Acordo com Especificado | Implementado mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | Nome Trigger:  Trigger\_variaveis\_insert | x |  |  |  | | Nome Trigger:  Trigger\_variaveis\_Delete | x |  |  |  | | Nome Trigger:  Trigger\_variaveis\_update | x |  |  |  | | Nome Trigger:  Trigger\_cultura\_insert | x |  |  |  | | Nome Trigger:  Trigger\_cultura\_delete | x |  |  |  | | Nome Trigger:  Trigger\_cultura\_update | x |  |  |  | | Nome Trigger:  Trigger\_medicoes\_insert | x |  |  |  | | Nome Trigger:  Trigger\_medicoes\_delete | x |  |  |  | | Nome Trigger:  Trigger\_medicoes\_update | x |  |  |  | | Nome Trigger:  Trigger\_sistema\_insert | x |  |  |  | | Nome Trigger:  Trigger\_sistema\_delete | x |  |  |  | | Nome Trigger:  Trigger\_sistema\_update | x |  |  |  | |

#### Triggers Implementados para gestão de logs

|  |
| --- |
| 1. Nome Trigger: Trigger\_variaveis\_insert  Trigger para envio de informação, depois da Inserção, enviado para a tabela log correspondente.  **CREATE TRIGGER `Trigger\_variaveis\_insert` AFTER INSERT ON `variaveis` FOR EACH ROW insert into log\_variaveis Values (IDOperacao, NomeUtilizador, IDUtilizador, Operacao, now(), NomeVariavel, IDVariavel, NomeVariavel\_Box)**  2. Nome Trigger: Trigger\_variaveis\_Delete  Trigger para envio de informação, depois da Eliminação, enviado para a tabela log correspondente.  **CREATE TRIGGER `Trigger\_variaveis\_delete` AFTER DELETE ON `variaveis` FOR EACH ROW insert into log\_variaveis Values (IDOperacao, NomeUtilizador, IDUtilizador, Operacao, now(), NomeVariavel, IDVariavel, NomeVariavel\_Box)**  3. Nome Trigger: Trigger\_variaveis\_uptade  Trigger para envio de informação, depois da Actualização, enviado para a tabela log correspondente.  **CREATE TRIGGER `Trigger\_variaveis\_update` AFTER UPDATE ON `variaveis` FOR EACH ROW insert into log\_variaveis Values (IDOperacao, NomeUtilizador, IDUtilizador, Operacao, now(), NomeVariavel, IDVariavel, NomeVariavel\_Box)**  4. Nome Trigger: Trigger\_cultura\_insert  Trigger para envio de informação, depois da Inserção, enviado para a tabela log correspondente.  **CREATE TRIGGER `Trigger\_cultura\_insert` AFTER INSERT ON `cultura` FOR EACH ROW insert into log\_cultura VALUES(IDOperacao, NomeOperacao, IDUtilizador, Operacao, Now(), NomeCultura, DescricaoCultura, IDCultura, DescricaoCultura\_Box)**  5. Nome Trigger: Trigger\_cultura\_delete  Trigger para envio de informação, depois da Eliminação, enviado para a tabela log correspondente.  **CREATE TRIGGER `Trigger\_cultura\_delete` AFTER DELETE ON `cultura` FOR EACH ROW insert into log\_cultura VALUES(IDOperacao, NomeOperacao, IDUtilizador, Operacao, Now(), NomeCultura, DescricaoCultura, IDCultura, DescricaoCultura\_Box)**  6. Nome Trigger: Trigger\_cultura\_update  Trigger para envio de informação, depois da Actualização, enviado para a tabela log correspondente.  **CREATE TRIGGER `Trigger\_cultura\_update` AFTER UPDATE ON `cultura` FOR EACH ROW insert into log\_cultura VALUES(IDOperacao, NomeOperacao, IDUtilizador, Operacao, Now(), NomeCultura, DescricaoCultura, IDCultura, DescricaoCultura\_Box)**  7. Nome Trigger: Trigger\_medicoes\_insert  Trigger para envio de informação, depois da Inserçao, enviado para a tabela log correspondente.  **CREATE TRIGGER `Trigger\_medicoes\_insert` AFTER INSERT ON `medicoes` FOR EACH ROW insert into log\_medicoes Values (IDOperacao, NomeUtilizador, IDUtilizador, Operacao, now(), DataHoraMedicao, ValorMedicao, NumeroMedicao, DataHoraMedicao\_Box, ValorMedicao\_Box)**  8. Nome Trigger: Trigger\_medicoes\_delete  Trigger para envio de informação, depois da Eliminação, enviado para a tabela log correspondente.  **CREATE TRIGGER `Trigger\_medicoes\_delete` AFTER DELETE ON `medicoes` FOR EACH ROW insert into log\_medicoes Values (IDOperacao, NomeUtilizador, IDUtilizador, Operacao, now(), DataHoraMedicao, ValorMedicao, NumeroMedicao, DataHoraMedicao\_Box, ValorMedicao\_Box)**  9. Nome Trigger: Trigger\_medicoes\_update  Trigger para envio de informação, depois da Actualização, enviado para a tabela log correspondente.  **CREATE TRIGGER `trigger\_medicoes\_update` AFTER UPDATE ON `medicoes` FOR EACH ROW insert into log\_medicoes Values (IDOperacao, NomeUtilizador, IDUtilizador, Operacao, now(), DataHoraMedicao, ValorMedicao, NumeroMedicao, DataHoraMedicao\_Box, ValorMedicao\_Box)**  10. Nome Trigger: Trigger\_sistema\_insert  Trigger para envio de informação, depois da Inserçao, enviado para a tabela log correspondente.  **CREATE TRIGGER `Trigger\_sistema\_insert` AFTER INSERT ON `sistema` FOR EACH ROW insert into log\_sistema Values (IDOperacao, NomeUtilizador, IDUtilizador, Operacao, now(), LimiteInferiorTemperatura, LimiteSuperiorTemperatura, LimiteSuperiorLuz, LimiteInferiorLuz, LimiteInferiorTemperatura\_Box, LimiteSuperiorTemperatura\_Box, LimiteSuperiorLuz\_Box, LimiteInferiorLuz\_Box)**  11. Nome Trigger: Trigger\_sistema\_delete  Trigger para envio de informação, depois da Eliminação, enviado para a tabela log correspondente.  **CREATE TRIGGER `Trigger\_sistema\_delete` AFTER DELETE ON `sistema` FOR EACH ROW insert into log\_sistema Values (IDOperacao, NomeUtilizador, IDUtilizador, Operacao, now(), LimiteInferiorTemperatura, LimiteSuperiorTemperatura, LimiteSuperiorLuz, LimiteInferiorLuz, LimiteInferiorTemperatura\_Box, LimiteSuperiorTemperatura\_Box, LimiteSuperiorLuz\_Box, LimiteInferiorLuz\_Box)**  12. Nome Trigger: Trigger\_sistema\_update  Trigger para envio de informação, depois da Actualização, enviado para a tabela log correspondente.  **CREATE TRIGGER `Trigger\_sistema\_update` AFTER UPDATE ON `sistema` FOR EACH ROW insert into log\_sistema Values (IDOperacao, NomeUtilizador, IDUtilizador, Operacao, now(), LimiteInferiorTemperatura, LimiteSuperiorTemperatura, LimiteSuperiorLuz, LimiteInferiorLuz, LimiteInferiorTemperatura\_Box, LimiteSuperiorTemperatura\_Box, LimiteSuperiorLuz\_Box, LimiteInferiorLuz\_Box)** |

Stored Procedures de suporte à criação de logs (**se relevante**)

<Caso seja pertinente poderão ser adicionados comentários explicar a estratégia a seguir.>

|  |  |  |  |
| --- | --- | --- | --- |
| Nome Procedimento | Parâmetros Entrada | Parâmetros Saída | Muito breve descrição |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

#### Apreciação Crítica de Stored Procedures de suporte à criação de logs

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): \_\_\_\_\_\_\_\_\_\_\_\_  Breve Justificação:  **Lista de SP (para cada SP assinalar com x em célula correspondente)**     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Implementado de Acordo com Especificado | Implementado, mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | Nome SP  (tal como especificado) |  |  |  |  | | Nome SP  (tal como especificado) |  |  |  |  | | Nome SP  (tal como especificado) |  |  |  |  | |  |  |  |  |  | |

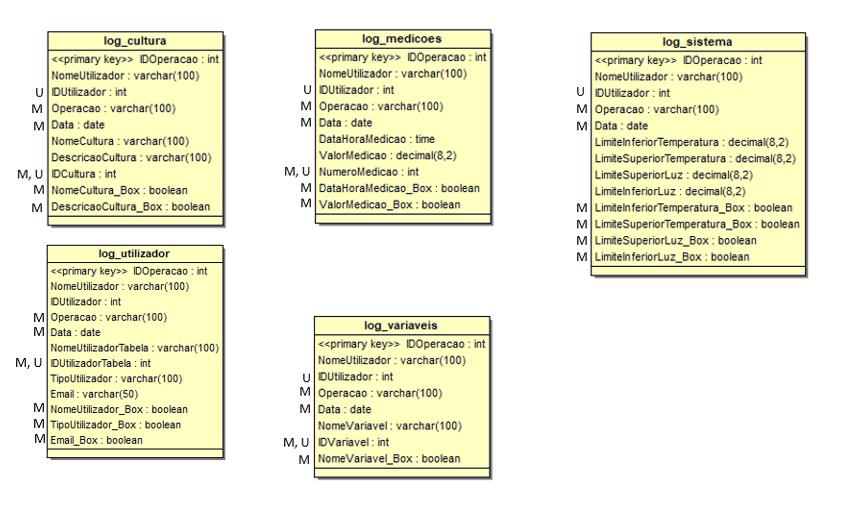
#### Stored Procedures Implementados de suporte à criação de logs

|  |
| --- |
| 1. Nome SP: \_\_\_\_\_  //*Breve Descrição*  *Código*  2. Nome SP: \_\_\_\_\_  //*Breve Descrição*  *Código*  3. Nome SP: \_\_\_\_\_  //*Breve Descrição*  *Código* |

## Migração entre Bases de Dados

### Esquema relacional da base de Dados Mysql (destino)

<Tem de incluir as tabelas e atributos que suportam a auditoria de dados. Seguir as mesmas indicações referidas na secção 1.1>



#### Apreciação Crítica e esquema relacional implementado

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Razoável  Breve Justificação:  Relativamente ao esquema relacional em si, não há nada a apontar de negativo. No entanto, receia-se que algumas decisões, no que diz respeito às tabelas, possam vir a comprometer o funcionamento como também a eficácia da BD. Dito isto, começa-se por relatar o possível problema de se ter para cada LOG o ID do utilizador assim como o Nome do mesmo. Numa situação futura, a possível alteração do Nome De Utilizador na tabela “Utilizador” não implica necessariamente que essa mudança seja realizada também nas tabelas LOG, o que levaria a uma situação indesejável.    Passando agora para os campos boolean, que servem para relatar se um campo fora alterado ou não, julga-se que trará mais desvantagens para o projeto, uma vez que para cada campo, sujeito a alteração, segue-se um outro com o mesmo nome + “\_Box”. Sendo assim, para cada campo da tabela original acomoda, na tabela LOG correspondente, esse mesmo campo mais outro só para reportar alguma alteração. Esta solução é pouco escalável, tendo como consequência uma BDs com tamanho excessivo. E pior será se, por alguma razão, as tabelas sofram alterações, nomeadamente um acréscimo de campos. Uma possível solução a este problema seria a de implementar uma lógica BitWise (apenas uma coluna nova com um inteiro, em que o valor armazenado nesta indicaria as colunas alteradas) que resolveria o problema da escalabilidade e de armazenamento da BD.    Na especificação, apenas o esquema relacional é pouco informativo e poderá levar à dificuldade na compreensão – poderiam ter aprofundado mais.  Foram feitas alterações? (Sim/Não): Não  **Novo Esquema (assinale e justifique as alterações)** |

### Forma de Migração

<Nesta secção deverá ser explicado a forma como a migração será efectuada. A informação apresentada deverá ser suficiente para que o grupo que a receba consiga implementar a migração.

Poderá ser importante incluir um diagrama que explique as várias etapas.

Deverão ser enumeradas as opções, comandos, utilitários, bibliotecas a utilizar. Caso relevante, indicar estratégias para nomes de ficheiros. Deverão ser abordadas questões como periodicidade, privacidade dos dados, eficiência, facilidade de manutenção.

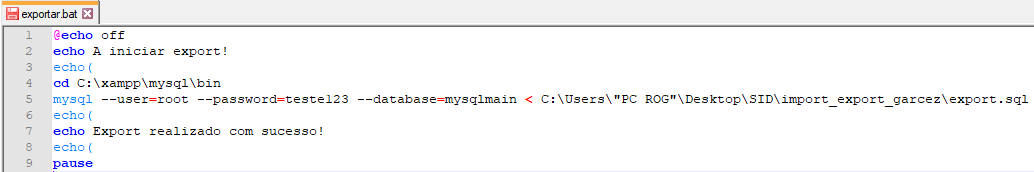
Quem especifica PHP não deve colocar o código, apenas a “lógica” subjacente.

Deverá estar claro no texto de que forma se assegura uma migração incremental (evitar que a informação chegue repetida à base de dados destino)>

1. Gerar as tabelas do esquema relacional da base de dados origem e base de dados destino. Para gerar essas tabelas é necessário gerar o código SQL, para isso podem usar o software que estão nos foram fornecido – BoUml (para gerar o esquema) e UmlToRel (passar do esquema para código SQL) ;
2. Introduzir os dados nas tabelas cultura, utilizador, variáveis, sistema, medições.
3. As colunas “*box*”, indicam qual ou quais as colunas alteradas/inseridas/removidas. Por exemplo se tiver *nome\_cultura\_box*=0 e *descrição\_cultura\_box*=1 significa que houve uma operação de *update* que apenas alterou a coluna “*descrição\_cultura\_box*”.
4. Os *triggers* serão responsáveis por gerar dados para as tabelas *log\_cultura, log\_utilizador, log\_variáveis, log\_sistema, log\_medições*. Todas estas tabelas tem uma coluna cujo o nome é “exportado” que inicialmente será colocado a “*false*” /0 pelos “*triggers after insert*”. É desta forma que é assegurada a migração incremental (evitar exportação de dados repetidos), isto é, só se exporta se o valor da coluna *exportado* for igual a “*false*” /0.
5. Correr o *batch file “exportar”.* O *batch file “exportar”* vai aceder ao ficheiro *exportar.sql* e vai mandar executar os comandos aí presentes guardando a informação da base de dados origem para um ficheiro *csv*. O *batch file* exporta apenas informação nova, esse objetivo é alcançado através da existência de um campo denominado *“exportado” – booleano* - que tem como finalidade saber se a linha já foi anteriormente exportada ou não;

Só serão exportados os dados em que o campo exportado estiver a false;

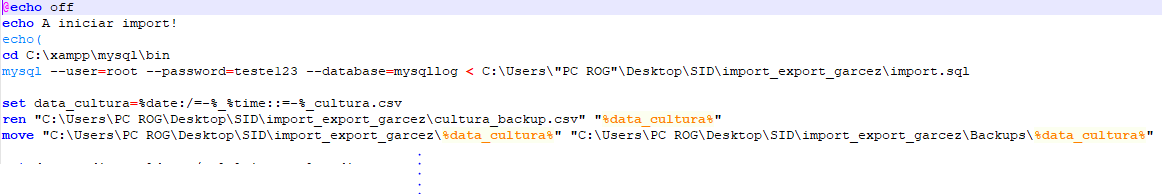
Após a exportação o campo “exportado” passa a *true* através do ficheiro *batch* que implementa código *sql* e modifica o campo para *true*, impedindo assim que o campo volte a ser exportado novamente.



**Ficheiro exportar.bat**

Linha 5: *teste123* é a password do utilizador *root, mysqlmain* é o nome dado à base de dados origem, *export.sql* é o nome do ficheiro que contém os comandos sql para exportar

1. Correr o *batch file* *importar.* O batch file *importar* por sua vez vai pegar no ficheiro criado e vai transferir essa informação para a base de dados destino, para a respetiva tabela;
2. As tabelas da base de dados destino deverão ter o mesmo nome das tabelas da base de dados origem a partir do qual o ficheiro que vai ser importado foi criado de modo a que o auditor tenha a perceção exata de quais as tabelas em que se procederam a modificações.
3. Deve criar uma pasta de backups pois depois de importados os ficheiros estes serão renomeados com a data e hora e serão movidos para essa pasta



**Ficheiro importar.bat**

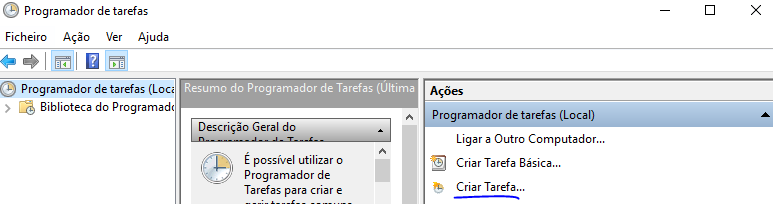
Linha 5: *teste123* é a password do utilizador *root, mysqllog* é o nome dado à base de dados destino, *import.sql* é o nome do ficheiro onde estão guardados os comandos *sql* para importar

Linha 7: define-se o novo nome com data e hora

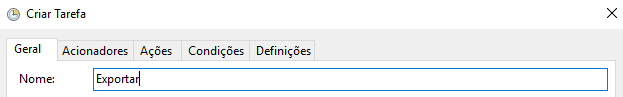
Linha 8: renomeia-se o ficheiro

Linha 9: move-se o ficheiro para a pasta backups

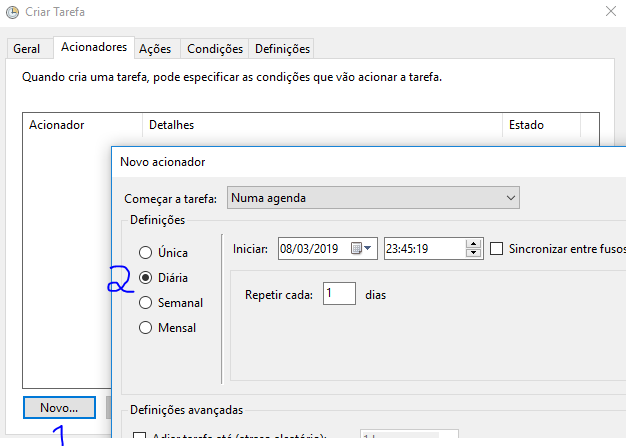
1. Depois de feitos os *batch files* exportar e importar deve faze-los correr diariamente. Para isso deve utilizar o “programador de tarefas do Windows” e aceder a “criar tarefa”

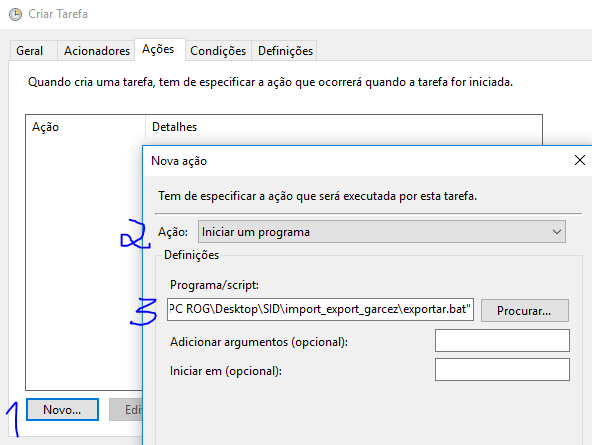


Depois, no separador “geral”, deve dar um nome à tarefa



De seguida, no separador “acionadores”, deve clicar em novo. Na janela que aparece deve alterar a periodicidade para diária e escolher a hora a que a tarefa ocorrerá



Por fim, no separador “ações”, deve carregar em “novo”. Na janela que aparece deve escolher a ação “Iniciar um programa” e deve escolher o ficheiro *batch* que quer correr



**Porquê *csv* em vez de *xml*?**

Visto que não interessa ver o conteúdo do ficheiro, mas sim usá-lo como um intermediário para mover a informação entre as bases de dados, optou-se por utilizar a exportação para ficheiro *csv* em vez de *xml*. Em *csv* basta exportar os dados separando-os por virgulas. Facilita também o processo de importação.

*CSV* é mais compacto em relação ao *xml*. Em *xml*, é necessário meter a tag inicial e tag final para cada coluna em cada linha. Ao contrário do *csv,* que apenas se escreve os cabeçalhos da coluna apenas uma vez.

Para além do *csv* ser um ficheiro de pequeno tamanho, é que o mais se assemelha a representação de uma tabela (colunas e linhas), sendo um ficheiro de manipulação fácil.

#### Apreciação Crítica à especificação da forma de migração

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Muito Boa  **Análise crítica (clareza, completude, rigor):**  Especificação em geral muito boa, detalhada e completa, apenas poderia ter uma especificação muito breve dos ficheiros exportar.sql e importar.sql. |

### Gestão de Utilizadores de Suporte à Migração (origem e/ou destino)

<Nesta secção deverá ser explicado de que forma deverá ser feita a manutenção (e.g., a criação e privilégios) de utilizadores (quer no destino, quer na origem, se necessário). Nomeadamente deverá ser indicado, para cada tipo de utilizador, que privilégios ele tem sobre que tabelas e Stored procedures (caso sejam referenciados na secção 1.4.5).

Sugestão de tabela

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Base de Dados (O/D)** | **Tabela** | **Tipo de Utilizador** | | |
| Administrador | Investigador | Auditor de dados |
| O | Cultura | E | E, L | - |
| O | Log\_cultura | L | - | - |
| O | Log\_medicoes | L | - | - |
| O | Log\_sistema | L | - | - |
| O | Log\_utilizador | L | - | - |
| O | Log\_variaveis | L | - | - |
| O | Medições | L | E, L | - |
| O | medicoesLuminosidade | L | L | - |
| O | MedicoesTemperatura | L | L | - |
| O | Sistema | L | L | - |
| O | Utilizador | E, L | - | - |
| O | Variáveis | E, L | L | - |
| O | VariaveisMedidas | L | L | - |
| D | Log\_cultura | - | - | L |
| D | Log\_medicoes | - | - | L |
| D | Log\_sistema | - | - | L |
| D | Log\_utilizador | - | - | L |
| D | Log\_variaveis | - | - | L |

Em que E=Escrita, L=Leitura, X=Executar e - = sem permissões

Caso seja pertinente poderão ser adicionados comentários explicar a estratégia a seguir.>

Quando se cria um user é necessário correr um stored procedure designado por *criar\_utilizador.* Esse stored procedure possui 3 parâmetros: NomeUtilizador, TipoUtilizador e Email. Este stored procedure vai guardar a informação dos parâmetros na tabela *utilizador* da base de dados de origem.

Vão existir três roles: grupo\_administradores, grupo\_investigadores e grupo\_auditores:

* **Grupo\_administradores:** Os utilizadores que pertencem a este grupo têm direito a ler as tabelas utilizador e variáveis da base de dados origem. Quem pertence a este role ainda tem direito a:
  + Inserir, alterar e apagar informação da tabela de utilizadores;
  + Inserir e apagar informação da tabela de variáveis.
* **Grupo\_investigadores:** Os utilizadores que pertencem a este grupo têm direito a:
  + Inserir, alterar e apagar informação da tabela de cultura;
  + Inserir, alterar e apagar informação da tabela de medições.
* **Grupo\_auditores:** Quem pertencer a esta role tem apenas direito a:
  + Leitura da tabela log\_cultura;
  + Leitura da tabela log\_medicoes;
  + Leitura da tabela log\_sistema;
  + Leitura da tabela log\_utilizador;
  + Leitura da tabela log\_variaveis.

#### Apreciação Crítica à especificação da Gestão de Utilizadores

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Boa  **Análise crítica (clareza, completude, rigor):**  Pouco rigor entre a informação na tabela e a descrição escrita que está por baixo, visto que, embora na tabela diga que os investigadores e os admins tenham permissões de leitura nas tabelas pelas quais são responsáveis, na descrição escrita não está explícito que vão ter esses direitos.  Para além disso, a especificação por baixo da tabela podia ser mais explicativa, como forma de melhorar a interpretação da implementação.  Em geral, a especificação está boa, só perdendo devido a estas incongruências.  **Solução Implementada:**  A solução implementada foi a especificada pelo grupo. |

### Triggers de suporte à migração de dados (origem e/ou destino) (**se relevante**)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nome Trigger | Tabela | Tipo de Operação (I,U,D) | Evento  (A,B) | BD  (Origem ou Destino) | Notas (apenas indicar aquilo que não será óbvio) |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

#### Apreciação Crítica de triggers de suporte à migração de dados

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): \_\_\_\_\_\_\_\_\_\_\_\_  Breve Justificação:  **Lista de Triggers (para cada trigger assinalar com x em célula correspondente)**     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Implementado de Acordo com Especificado | Implementado mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | Nome Trigger  (tal como especificado) |  |  |  |  | | Nome Trigger  (tal como especificado) |  |  |  |  | | Nome Trigger  (tal como especificado) |  |  |  |  | |  |  |  |  |  | |

#### Triggers Implementados de suporte à migração de dados

|  |
| --- |
| 1. Nome Trigger: \_\_\_\_\_  //*Breve Descrição*  *Código*  2. Nome Trigger: \_\_\_\_\_  //*Breve Descrição*  *Código*  3. Nome Trigger: \_\_\_\_\_  //*Breve Descrição*  *Código* |

### Stored Procedures de suporte à migração de dados

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome Procedimento | Parâmetros Entrada | Parâmetros Saída | BD  (Origem ou Destino) | Muito breve descrição |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

#### Apreciação Crítica de Stored Procedures de suporte à migração de dados

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): \_\_\_\_\_\_\_\_\_\_\_\_  Breve Justificação:  **Lista de SP (para cada SP assinalar com x em célula correspondente)**     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Implementado de Acordo com Especificado | Implementado mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | Nome SP  (tal como especificado) |  |  |  |  | | Nome SP  (tal como especificado) |  |  |  |  | | Nome SP  (tal como especificado) |  |  |  |  | |  |  |  |  |  | |

#### Storedd Procedures Implementados de suporte à migração de dados

|  |
| --- |
| 1. Nome SP: \_\_\_\_\_  //*Breve Descrição*  *Código*  2. Nome SP: \_\_\_\_\_  //*Breve Descrição*  *Código*  3. Nome SP: \_\_\_\_\_  //*Breve Descrição*  *Código* |

### Eventos de suporte à migração de dados

<Nesta secção deverá ser indicados os eventos relevantes para o processo de migração.>

|  |  |  |
| --- | --- | --- |
| Nome Evento | Local Execução  (Origem ou Destino, ou Sistema Operativo) | Muito breve descrição |
| Importar | Destino | O evento “importar” é um batch file que serve para migrar os dados de um ficheiro csv, que contém dados dos logs, para a base de dados destino. Após isso, vai mover o ficheiro csv para uma pasta backup. |
| Exportar | Origem | O evento “exportar” é um batch file que vai copiar apenas os dados das tabelas da base de dados origem, que não foram previamente exportados, para um ficheiro csv. |

#### Apreciação Crítica de Eventos

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Boa  Breve Justificação:  Todos os eventos apresentados pelo grupo 5 são os relevantes para a migração e encontram-se bem descritos na tabela acima representada.  **Lista de Eventos (para cada evento assinalar com x em célula correspondente)**     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Implementado de Acordo com Especificado | Implementado mas diferente do Especificado | Não Implementado | Não Especificado (criado de novo) | | Importar | X |  |  |  | | Exportar | X |  |  |  | |

#### Eventos Implementados

|  |
| --- |
| 1. Nome Evento: Importar  Implementado segundo o ponto apresentado pelo grupo 5 em 1.4.2.    2. Nome Evento: Exportar  Implementado segundo o ponto apresentado pelo grupo 5 em 1.4.2. |

### PHP suporte à migração de dados (se relevante)

<Nesta secção deverá especificar a lógica subjacente ao programa PHP de suporte à migração>

#### Apreciação Crítica ao PHP especificado

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): \_\_\_\_\_\_\_\_\_\_\_\_  Breve Justificação: |

#### PHP Implementado

|  |
| --- |
| *Código* |

## Avaliação Global de especificações da Etapa A

Boa especificação no geral que foi de fácil compreensão independemente de alguns pontos menos detalhados em outras secções.

Isto tudo levou-nos a uma implementação rápida e coerente ajudando também a ter outra perspectiva na abordagem ao tema.

Em comparação também com o nosso projeto, deu-nos uma visão externa que nos ajudou a melhorar a implementação via PHP.

**Avaliação Global da Qualidade das Especificações recebidas**

|  |
| --- |
| Avaliação (A,B,C,D,E) : D  Utilize a seguinte escala:  A: - 1 – 5 valores B: 6 – 9 valores C: 10 – 13 Valores D: 14 – 17 valores E: 18 – 20 valores |

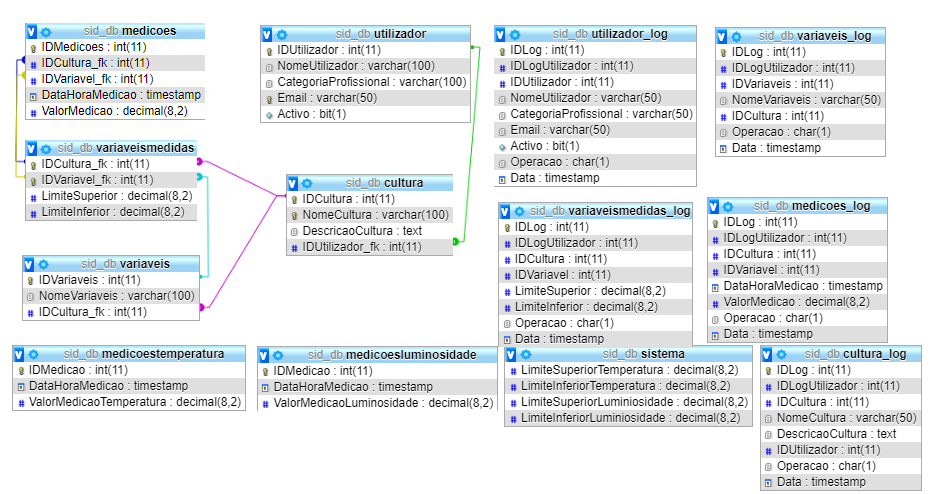
**Três principais deficiências de especificação que tiveram impacto mais negativo na qualidade da implementação**

|  |
| --- |
| Nome de Utilizador nas Tabelas Logs |
| Colunas desnecessárias nas tabelas logs para acompanhar as alterações de campos que foram efetuadas, no caso disto ser considerado necessário esta solução não seria a mais correcta. |
| As permissões dos utilizadores criavam fragilidades na BD, principalmente as permissões excessivas de consultas de tabelas por parte do Administrador. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Resumo de Avaliações de Qualidade Anteriores (para cada linha assinalar com x em célula correspondente)**   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Fraco | Razoável | Bom | Muito Bom | | BD Origem |  |  | x |  | | Triggers Log |  |  | x |  | | SP Log | - | - | - | - | | Utilizadores Log | x |  |  |  | | BD Destino |  | x |  |  | | Forma Migração |  |  |  | x | | Triggers Migração | - | - | - | - | | SP Migração | - | - | - | - | | Eventos Migração |  |  | x |  | | Utilizadores Migração |  |  | x |  | | PHP Migração |  |  | x |  | |

# Etapa C (Especificação e Implementação do Próprio Grupo)

## Especificação do Esquema relacional da base de Dados Origem



* Tabela **medicoes**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDMedicoes** | pk,Auto Increment | Chave primária da tabela **medicoes**, única para cada medição. Atributo Auto Increment que incrementa um valor IDMedicoes por ação à tabela **medicoes**. |
| **IDCultura** | fk | Chave estrangeira proveniente da tabela **variaveismedida**. |
| **IDVariavel** | fK | Chave estrangeira proveniente da tabela **variaveismedida**. |
| **DataHoraMedicao** | not null | Obrigatório completar com a data e hora da medição. |
| **ValorMedicao** | not null | Obrigatório completar com o valor da medição. |

* Tabela **variaveismedidas**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDCultura** | pk,fk | Chave primária da tabela **variaveismedidas**. Em conjunto com a chave primária IDVariavel.  Chave estrangeira proveniente da tabela **cultura**. |
| **IDVariavel** | pk,fk | Chave primária da tabela **variaveismedidas**. Em conjunto com a chave primária IDCultura.  Chave estrangeira proveniente da tabela **variavel**. |
| **LimiteSuperior** | not null | Obrigatório completar com o valor do limite máximo atingido. |
| **LimiteInferior** | not null | Obrigatório completar com o valor do limite mínimo atingido. |

* Tabela **variaveis**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDVariaveis** | pk,Auto Increment | Chave primária da tabela **variaveis**. Atributo Auto Increment que incrementa um valor IDVariaveis por ação à tabela **variaveis**. |
| **NomeVariaveis** | not null | Obrigatório completar com o nome da variável a medir. |
| **IDCultura** | fk | Chave estrangeira proveniente da tabela **cultura**. |

* Tabela **utilizador**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDUtilizador** | pk,Auto Increment | Chave primária da tabela **utilizador**. Atributo Auto Increment que incrementa um valor IDUtilizador por ação à tabela **utilizador**. |
| **NomeUtilizador** | not null | Obrigatório completar com o nome do utilizador. |
| **CategoriaProfissional** | not null | Obrigatório completar conforme a sua profissão (Investigador, Administrador ou Auditor). |
| **Email** | unique | Restrição que garante não existirem email de utilizadores iguais. |
| **Activo** | not null | Obrigatório completar com o valor 1 (true) para utilizador activo ou o valor 0 (false) para utilizador inactivo. |

* Tabela **cultura**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDCultura** | pk,Auto Increment | Chave primária da tabela **cultura**. Em conjunto com a chave primária NomeCultura. Atributo Auto Increment que incrementa um valor IDCultura por ação à tabela **cultura**. |
| **NomeCultura** | pk | Chave primária da tabela **cultura**. Em conjunto com a chave primária IDCultura. |
| **DescricaoCultura** | not null | Obrigatório completar com uma descrição da cultura. |
| **IDUtilizador** | fk | Chave estrangeira proveniente da tabela **utilizador**. |

* Tabela **sistema**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **LimiteSuperiorTemperatura** | not null | Obrigatório completar com o valor do limite máximo atingido na temperatura. |
| **LimiteInferiorTemperatura** | not null | Obrigatório completar com o valor do limite mínimo atingido na temperatura. |
| **LimiteSuperiorLuminosidade** | not null | Obrigatório completar com o valor do limite máximo atingido na luminosidade. |
| **LimiteInferiorLuminosidade** | not null | Obrigatório completar com o valor do limite mínimo atingido na luminosidade. |

* Tabela **medicoesluminosidade**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDMedicao** | pk,Auto Increment | Chave primária da tabela **medicoesluminosidade**. Atributo Auto Increment que incrementa um valor IDMedicao por ação à tabela **medicoesluminosidade**. |
| **DataHoraMedicao** | not null | Obrigatório completar com a data e hora da medição da luminosidade. |
| **ValorMedicaoLuminosidade** | not null | Obrigatório completar com o valor obtido na medição da luminosidade. |

* Tabela **medicoestemperatura**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDMedicao** | pk,Auto Increment | Chave primária da tabela **medicoestemperatura**. Atributo Auto Increment que incrementa um valor IDMedicao por ação à tabela **medicoestemperatura**. |
| **DataHoraMedicao** | not null | Obrigatório completar com a data e hora da medição da temperatura. |
| **ValorMedicaoTemperatura** | not null | Obrigatório completar com o valor obtido na medição da temperatura. |

* Tabela **utilizador\_log**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDLog** | pk,Auto Increment | Chave primária da tabela **utilizador\_log**. Designa o número da operação feita. Atributo Auto Increment que incrementa um valor IDLog por ação à tabela **utilizador\_log**. |
| **IDLogUtilizador** | not null | Obrigatório completar com o id de log do utilizador. |
| **IDUtilizador** | not null | Obrigatório completar com o id do utilizador. |
| **NomeUtilizador** | not null | Obrigatório completar com o nome do utilizador. |
| **CategoriaProfissional** | not null | Obrigatório completar com a categoria profissional do utilizador. |
| **Email** | not null | Obrigatório completar com o email do utilizador. |
| **Activo** | not null | Obrigatório completar com o valor 1 (true) para utilizador activo ou o valor 0 (false) para utilizador inactivo. |
| **Operacao** | not null | Obrigatório completar com o tipo de trigger que foi efectuado (I - Insert; U - Update; D - Delete. |
| **Data** | not null | Obrigatório completar com a data e hora do em que foi feito o log do utilizador. |

* Tabela **variaveis\_log**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDLog** | pk,Auto Increment | Chave primária da tabela **variaveis\_log**. Designa o número da operação feita. Atributo Auto Increment que incrementa um valor IDLog por ação à tabela **variaveis\_log**. |
| **IDLogUtilizador** | not null | Obrigatório completar com o id de log do utilizador. |
| **IDVariavel** | not null | Obrigatório completar com o id da variável. |
| **NomeVariaveis** | not null | Obrigatório completar com o nome da variável. |
| **IDCultura** | not null | Obrigatório completar com o id da cultura. |
| **Operacao** | not null | Obrigatório completar com o tipo de trigger que foi efectuado (I - Insert; U - Update; D - Delete. |
| **Data** | not null | Obrigatório completar com a data e hora do em que foi feito o log do variável. |

* Tabela **medicoes\_log**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDLog** | pk,Auto Increment | Chave primária da tabela **medicoes\_log**. Designa o número da operação feita. Atributo Auto Increment que incrementa um valor IDLog por ação à tabela **medicoes\_log**. |
| **IDLogUtilizador** | not null | Obrigatório completar com o id de log do utilizador. |
| **IDCultura** | not null | Obrigatório completar com o id da cultura. |
| **IDVariavel** | not null | Obrigatório completar com o id da variável. |
| **DataHoraMedicao** | not null | Obrigatório completar com a data e hora da medição. |
| **ValorMedicao** | not null | Obrigatório completar com o valor da medição. |
| **Operacao** | not null | Obrigatório completar com o tipo de trigger que foi efectuado (I - Insert; U - Update; D - Delete. |
| **Data** | not null | Obrigatório completar com a data e hora do em que foi feito o log do medição. |

* Tabela **cultura\_log**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDLog** | pk,Auto Increment | Chave primária da tabela **cultura\_log**. Designa o número da operação feita. Atributo Auto Increment que incrementa um valor IDLog por ação à tabela **cultura\_log**. |
| **IDLogUtilizador** | not null | Obrigatório completar com o id de log do utilizador. |
| **NomeCultura** | not null | Obrigatório completar com o nome da cultura. |
| **DescricaoCultura** | not null | Obrigatório completar com uma descrição da cultura. |
| **IDUtilizador** | not null | Obrigatório completar com o id do utilizador. |
| **Operacao** | not null | Obrigatório completar com o tipo de trigger que foi efectuado (I - Insert; U - Update; D - Delete. |
| **Data** | not null | Obrigatório completar com a data e hora do em que foi feito o log da cultura. |

* Tabela **variaveismedidas\_log**

|  |  |  |
| --- | --- | --- |
|  | **Atributos** | **Observações** |
| **IDLog** | pk,Auto Increment | Chave primária da tabela **variaveismedidas\_log**. Designa o número da operação feita. Atributo Auto Increment que incrementa um valor IDLog por ação à tabela **variaveismedidas\_log**. |
| **IDLogUtilizador** | not null | Obrigatório completar com o id de log do utilizador. |
| **IDCultura** | not null | Obrigatório completar com o id da cultura. |
| **IDVariavel** | not null | Obrigatório completar com o id da variável. |
| **LimiteSuperior** | not null | Obrigatório completar com o valor do limite máximo atingido. |
| **LimiteInferior** | not null | Obrigatório completar com o valor do limite mínimo atingido. |
| **Operacao** | not null | Obrigatório completar com o tipo de trigger que foi efectuado (I - Insert; U - Update; D - Delete. |
| **Data** | not null | Obrigatório completar com a data e hora do em que foi feito o log da variável medida. |

Para todas as tabelas apresentadas em cima que possuam chaves estrangeiras, estas têm como propriedade ON DELETE RESTRICT - Para que caso se queira apagar uma linha que que a pk seja referenciada como fk noutra tabela, surja um erro que nos impede de apagar sem conhecimento; E ON UPDATE RESTRICT - igual procedimento mas com a ação de update.

Foram acrescentadas tabelas log para além das tabelas já existentes no enunciado - utilizador\_log, variaveis\_log, medicoes\_log, variaveismedidas\_log e cultura\_log. Estas possuem os mesmos atributos das tabelas utilizador, variaveis, medicoes, variaveismedicoes e cultura, respectivamente; e também atributos novos:

* **IDLog** que designa o número da operação feita.
* **IDLogUtilizador** que designa o id do utilizador que executou a operação.
* **Operacao** que pode ter a letra I para INSERT, D para DELETE, U para UPDATE. Consoante o trigger que foi efectuado.
* **Data** em que foi realizado o log.

## Especificação de Utilizadores

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabela** | **Tipo de Utilizador** | | |
| Admin | Investigador | Auditor |
| Medicoes\_Log | - | E | - |
| Utilizador\_Log | E | - | - |
| Variáveis\_Log | - | E | - |
| Cultura\_Log | E | - | - |
| VariáveisMedidas\_Log | - | E | - |
| MedicoesLuminosidade | - | L | - |
| MedicoesTemperatura | - | L | - |
| Utilizador | E | - | - |
| Variaveis | - | E | - |
| Cultura | E | - | - |
| Medicoes | - | E | - |
| VariaveisMedidas | - | E | - |
| Sistema | - | L | - |
| **Stored Proc.** |  |  | - |
| CriarUtilizador | X | - | - |
| AlterarUtilizador | X | - | - |
| ApagarUtilizador | X | - | - |
| CriarMedicoes\_Log | - | X | - |
| CriarUtilizador\_Log | X | - | - |
| CriarVariaveis\_Log | - | - | - |
| CriarCultura\_Log | X | X | - |
| CriarVariaveisMedidas\_Log | - | X | - |
| CriarRoles | - | - | - |

Legenda:

E = Escrita

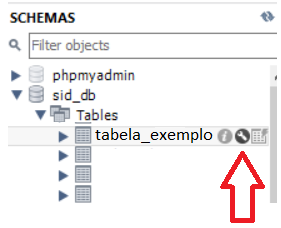
L = Leitura

X = Executar

- = sem permissões

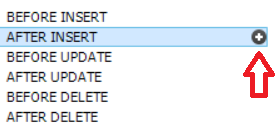
## Especificação de Gestão de Logs

### Criação Triggers de suporte à criação de logs Base de Dados de Origem - MySQL

Neste caso será demonstrada a criação de um log a partir do MySQL Workbench.  


Uma vez iniciado o MySQL Workbench iremos ao nosso Navigator[SCHEMAS], que se encontra do lado esquerdo, e já com uma BD criada acompanhada de pelo menos uma tabela podemos selecionar esta mesmo.

Clicamos no símbolo da “ferramenta” desta tabela e irá abrir uma nova janela. Esta janela será acompanhada na parte de baixo pelos seguintes atalhos onde será selecionado “Triggers”.  

Seguidamente teremos uma nova vista na mesma janela onde se disponibiliza vários tipos de Triggers para criação:

AFTER e BEFORE correspondentes ao INSERT, UPDATE e DELETE.

Já escolhido o tipo de Trigger irá surgir esta secção de edição para criar e definir este mesmo.  


Uma vez concluído a edição é necessária um  .

### Tipos de Triggers usados

* Insert

Será “disparado” assim que exista uma inserção de dados numa tabela com este tipo de Trigger.

* Update

Este será “acionado” logo que haja uma alteração em linhas já criadas numa tabela.

* Delete

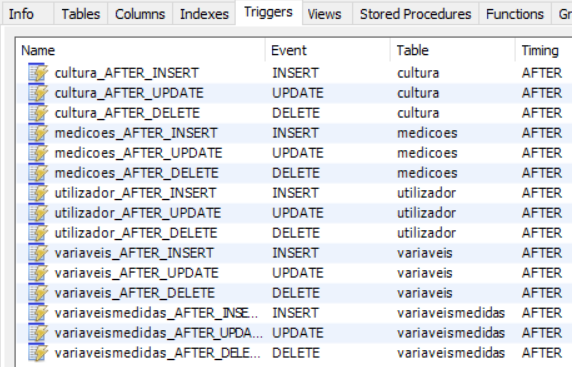
Este evento irá funcionar assim que seja removido uma linha da tabela

Todos os eventos nesta BD serão feitos em AFTER pelo motivo de poder registar nos Logs.  
Serão nestes triggers chamados os - Stored Procedures - 1.4.5 - correspondentes.

### Triggers de suporte à gestão de logs

Foram criados Triggers para se obter os “disparos” assim que haja uma alteração nas tabelas quer seja a nível de Inserção, Edição ou Eliminação com fim a ser executado num Stored Procedure para inserir na tabela de Logs.

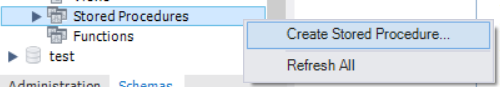
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome Trigger | Tabela | Tipo de Operação (I,U,D) | Evento  (A, B) | Notas |
| cultura\_AFTER\_INSERT | cultura | I | A | Depois de inserido efetua o registo em Cultura\_Log |
| cultura\_AFTER\_UPDATE | cultura | U | A | Depois de alterado efetua o registo em Cultura\_Log |
| cultura\_AFTER\_DELETE | cultura | D | A | Depois de apagado efetua o registo em Cultura\_Log |
| medicoes\_AFTER\_INSERT | medicoes | I | A | Depois de inserido efetua o registo em Medicoes\_Log |
| medicoes\_AFTER\_UPDATE | medicoes | U | A | Depois de alterado efetua o registo em Medicoes\_Log |
| medicoes\_AFTER\_DELETE | medicoes | D | A | Depois de apagado efetua o registo em Medicoes\_Log |
| utilizador\_AFTER\_INSERT | utilizador | I | A | Depois de inserido efetua o registo em Utilizador\_Log |
| utilizador\_AFTER\_UPDATE | utilizador | U | A | Depois de editado efetua o registo em Utilizador\_Log |
| utilizador\_AFTER\_DELETE | utilizador | D | A | Depois de apagado efetua o registo em Utilizador\_Log |
| variaveis\_AFTER\_INSERT | variaveis | I | A | Depois de inserido efetua o registo em Variaveis\_Log |
| variaveis\_AFTER\_UPDATE | variaveis | U | A | Depois de editado efetua o registo em Variaveis\_Log |
| variaveis\_AFTER\_DELETE | variaveis | D | A | Depois de apagado efetua o registo em Variaveis\_Log |
| variaveismedidas\_AFTER\_INSERT | variaveismedidas | I | A | Depois de inserido efetua o registo em VariaveisMedidas\_Log |
| variaveismedidas\_AFTER\_UPDATE | variaveismedidas | U | A | Depois de editado efetua o registo em VariaveisMedidas\_Log |
| variaveismedidas\_AFTER\_DELETE | variaveismedidas | D | A | Depois de apagado efetua o registo em VariaveisMedidas\_Log |



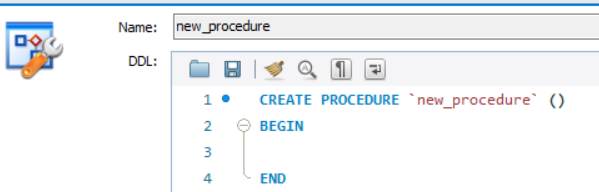
As tabelas Logs terão a seguinte construção: colunas para o ID do Log, para o ID do utilizador que fez a edição, todas as colunas da tabela a que este faz referência, a operação que foi feita e a data em que esta foi realizada.

### Criação de Stored Procedures - MySQL

Na secção à esquerda do MySQL Workbench podemos encontrar no Navigator os nossos Schemas a BD onde queremos criar SP’s para isso selecionamos com o lado direito do rato “Stored Procedures” e seguidamente na janela pop-up selecionar “Create Stored Procedure...”



Uma nova janela será aberta dentro da Workbench com uma base para podermos escrever o script da nossa SP.



Uma vez que o nosso script para a SP está completo confirmamos com .

Este processo será também referente às SP’s de suporte à migração - 1.4.5 .

### Stored Procedures de suporte à gestão de logs

Estas SPs são inicializadas com o intuito de criar os logs na base de dados origem, sendo estas SPs usadas também para o - suporte à migração das tabelas de log - 1.4.5.

|  |  |  |  |
| --- | --- | --- | --- |
| Nome Procedimento | Parâmetros Entrada | Parâmetros Saída | Muito breve descrição |
| CriarLogCultura | @IDLogUtilizador,  @IDCultura,  @NomeCultura,  @DescriçãoCultura,  @IDUtilizador,  @Operacao,  @Data |  | Popular tabela Cultura\_Log na BD origem |
| CriarLogUtilizador | @IDLogUtilizador,  @IDUtilizador,  @NomeUtilizador,  @CategoriaProfissional,  @Email,  @Activo,  @Operação,  @Char |  | Popular tabela Utilizador\_Log na BD origem |
| CriarLogMedicoes | @IDLogUtilizador,  @IDMedicoes,  @IDVariavel,  @DataHoraMedicao,  @ValorMedicao,  @Operacao,  @Data |  | Popular tabela Medicoes\_Log na BD origem |
| CriarLogVariaveis | @IDLogUtilizador,  @IDVariaveis,  @NomeVariaveis,  @IDCultura\_fk,  @Operacao,  @Data |  | Popular tabela Variaveis\_Log na BD origem |
| CriarLogVariaveisMedidas | @IDLogUtilizador,  @IDVariavel,  @LimiteSuperior,  @LimiteInferior,  @Operacao,  @Data |  | Popular tabela VariaveisMedidas\_Log na BD origem |

#### 

## Avaliação da especificação do próprio grupo Gestão de Logs

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Boa  Justificação:  Estavam a ser usados SP’s para o preenchimento das tabelas logs o que tirava performance quando estes eram chamados nos Triggers.  Alteração:  Foram retirados todos os SP’s de apoio a gestão de Logs. |

## Implementação Gestão de Logs

### Utilizadores implementados

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Tabela** | **Tipo de Utilizador** | | | | Admin | Investigador | Auditor | | utilizador\_log | E | - | L | | variaveis\_log | - | - | L | | variaveismedidas\_log | - | E | L | | cultura\_log | E | - | L | | medicoes\_log | - | E | L | | utilizador | E,L | - | L | | variaveis | - | E,L | - | | variaveismedidas | - | E,L | - | | cultura | E,L | L | - | | medicoes | - | E,L | - | | sistema | - | L | - | | medicoestemperatura | - | L | - | | medicoesluminiosidade | - | L | - | | **Stored Proc.** |  |  |  | | ApagarUtilizador | X | - | - | | CriarUtilizador | X | - | - | | EditarUtilizador | X | - | - | | LerMedicoes | - | X | - |   Foram usados SP’s de auxílio ao desenvolvimento da BD: popularBD e ApagarDadosLog. Estas tiveram uma ajuda crucial nos testes - 2.9. |

### 

### Lista de Triggers

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Lista de Triggers (para cada trigger assinalar com x em célula correspondente)**     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Implementado de Acordo com Especificado | Implementado mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | utilizador\_AFTER\_INSERT | x |  |  |  | | utilizador\_AFTER\_UPDATE | x |  |  |  | | cultura\_AFTER\_DELETE | x |  |  |  | | cultura\_AFTER\_UPDATE | x |  |  |  | | cultura\_AFTER\_DELETE | x |  |  |  | | medicoes\_AFTER\_INSERT | x |  |  |  | | medicoesr\_AFTER\_UPDATE | x |  |  |  | | medicoes\_AFTER\_DELETE | x |  |  |  | | medicoes\_BEFORE\_INSERT |  |  |  | x | | medicoes\_BEFORE\_UPDATE |  |  |  | x | | medicoes\_BEFORE\_DELETE |  |  |  | x | | variaveis\_AFTER\_INSERT | x |  |  |  | | variaveis\_AFTER\_UPDATE | x |  |  |  | | variaveis\_AFTER\_DELETE | x |  |  |  | | variaveis\_BEFORE\_INSERT |  |  |  | x | | variaveis\_BEFORE\_UPDATE |  |  |  | x | | variaveis\_BEFORE\_DELETE |  |  |  | x | | variaveismedidas\_AFTER\_INSERT | x |  |  |  | | variaveismedidas\_AFTER\_UPDATE | x |  |  |  | | variaveismedidas\_AFTER\_DELETE | x |  |  |  | | variaveismedidas\_BEFORE\_INSERT |  |  |  | x | | variaveismedidas\_BEFORE\_UPDATE |  |  |  | x | | variaveismedidas\_BEFORE\_DELETE |  |  |  | x |   O uso de triggers BEFORE serve de confirmação e segurança. É explicado mais à frente - 2.5.3. |

### Triggers Implementados

|  |
| --- |
| 1. Nome Trigger: cultura\_AFTER\_INSERT  Trigger para envio de informação, depois da Inserção, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `cultura\_AFTER\_INSERT` AFTER INSERT ON `cultura` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.cultura\_log*  *(IDLogUtilizador*  *, IDCultura*  *, NomeCultura*  *, DescricaoCultura*  *, IDUtilizador*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER),*  *NEW.IDCultura,*  *NEW.NomeCultura,*  *NEW.DescricaoCultura,*  *NEW.IDUtilizador\_fk,*  *'I',*  *NOW());*  *END*  2. Nome Trigger: cultura\_AFTER\_UPDATE  Trigger para envio de informação, depois da Actualização, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `cultura\_AFTER\_UPDATE` AFTER UPDATE ON `cultura` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.cultura\_log*  *(IDLogUtilizador*  *, IDCultura*  *, NomeCultura*  *, DescricaoCultura*  *, IDUtilizador*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER),*  *NEW.IDCultura,*  *NEW.NomeCultura,*  *NEW.DescricaoCultura,*  *NEW.IDUtilizador\_fk,*  *'U',*  *NOW());*  *END*  3. Nome Trigger: cultura\_AFTER\_DELETE  Trigger para envio de informação, depois da Eliminação, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `cultura\_AFTER\_DELETE` AFTER DELETE ON `cultura` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.cultura\_log*  *(IDLogUtilizador*  *, IDCultura*  *, NomeCultura*  *, DescricaoCultura*  *, IDUtilizador*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER),*  *OLD.IDCultura,*  *OLD.NomeCultura,*  *OLD.DescricaoCultura,*  *OLD.IDUtilizador\_fk,*  *'D',*  *NOW());*  *END*  4. Nome Trigger: utilizador\_AFTER\_INSERT  Trigger para envio de informação, depois da Inserção, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `utilizador\_AFTER\_INSERT` AFTER INSERT ON `utilizador` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.utilizador\_log*  *(IDLogUtilizador*  *, IDUtilizador*  *, NomeUtilizador*  *, CategoriaProfissional*  *, Email*  *, Activo*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, NEW.IDUtilizador*  *, NEW.NomeUtilizador*  *, NEW.CategoriaProfissional*  *, NEW.Email*  *, NEW.Activo*  *, 'I'*  *, NOW());*  *END*  5. Nome Trigger: utilizador\_AFTER\_UPDATE  Trigger para envio de informação, depois da Actualização, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `utilizador\_AFTER\_UPDATE` AFTER UPDATE ON `utilizador` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.utilizador\_log*  *(IDLogUtilizador*  *, IDUtilizador*  *, NomeUtilizador*  *, CategoriaProfissional*  *, Email*  *, Activo*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, NEW.IDUtilizador*  *, NEW.NomeUtilizador*  *, NEW.CategoriaProfissional*  *, NEW.Email*  *, NEW.Activo*  *, 'E'*  *, NOW());*  *END*  6. Nome Trigger: utilizador\_AFTER\_DELETE  Trigger para envio de informação, depois da Eliminação, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `utilizador\_AFTER\_DELETE` AFTER DELETE ON `utilizador` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.utilizador\_log*  *(IDLogUtilizador*  *, IDUtilizador*  *, NomeUtilizador*  *, CategoriaProfissional*  *, Email*  *, Activo*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, OLD.IDUtilizador*  *, OLD.NomeUtilizador*  *, OLD.CategoriaProfissional*  *, OLD.Email*  *, OLD.Activo*  *, 'D'*  *, NOW());*  *END*  7. Nome Trigger: medicoes\_AFTER\_INSERT  Trigger para envio de informação, depois da Inserção, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `medicoes\_AFTER\_INSERT` AFTER INSERT ON `medicoes` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.medicoes\_log*  *(IDLogUtilizador*  *, IDMedicoes*  *, IDCultura*  *, IDVariavel*  *, DataHoraMedicao*  *, ValorMedicao*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, NEW.IDMedicoes*  *, NEW.IDCultura\_fk*  *, NEW.IDVariavel\_fk*  *, NEW.DataHoraMedicao*  *, NEW.ValorMedicao*  *, 'I'*  *, NOW());*  *END*  8. Nome Trigger: medicoes\_AFTER\_UPDATE  Trigger para envio de informação, depois da Actualização, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `medicoes\_AFTER\_UPDATE` AFTER UPDATE ON `medicoes` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.medicoes\_log*  *(IDLogUtilizador*  *, IDMedicoes*  *, IDCultura*  *, IDVariavel*  *, DataHoraMedicao*  *, ValorMedicao*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, NEW.IDMedicoes*  *, NEW.IDCultura\_fk*  *, NEW.IDVariavel\_fk*  *, NEW.DataHoraMedicao*  *, NEW.ValorMedicao*  *, 'U'*  *, NOW());*  *END*  9. Nome Trigger: medicoes\_AFTER\_DELETE  Trigger para envio de informação, depois da Eliminação, enviado para a tabela log correspondente.  *, IDVariavel*  *, DataHoraMedicao*  *, ValorMedicao*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, OLD.IDMedicoes*  *, OLD.IDCultura\_fk*  *, OLD.IDVariavel\_fk*  *, OLD.DataHoraMedicao*  *, OLD.ValorMedicao*  *, 'D'*  *, NOW());*  *END*  *[*Implementado mas diferente do Especificado]  10. Nome Trigger: medicoes\_BEFORE\_INSERT  Trigger para verificação se o Investigador que insere a variável medida é o Investigador que está associado à cultura chamando um comando de alerta de erro caso isso não seja verdadeiro.  *CREATE DEFINER=`root`@`localhost` TRIGGER `medicoes\_BEFORE\_INSERT` BEFORE INSERT ON `medicoes` FOR EACH ROW BEGIN*  *SET @IDUtilizador = CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER);*    *IF NOT EXISTS (SELECT 1 FROM cultura c WHERE c.IDCultura = NEW.IDCultura\_fk AND c.IDUtilizador\_fk = @IDUtilizador) THEN*  *CALL cannot\_insert\_error;*  *END IF;*  *END*  *[*Implementado mas diferente do Especificado]  11. Nome Trigger: medicoes\_BEFORE\_UPDATE  Trigger para verificação se o Investigador que actualiza a variável medida é o Investigador que está associado à cultura chamando um comando de alerta de erro caso isso não seja verdadeiro.  *CREATE DEFINER=`root`@`localhost` TRIGGER `medicoes\_BEFORE\_UPDATE` BEFORE UPDATE ON `medicoes` FOR EACH ROW BEGIN*  *SET @IDUtilizador = CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER);*    *IF NOT EXISTS (SELECT 1 FROM cultura c WHERE c.IDCultura = NEW.IDCultura\_fk AND c.IDUtilizador\_fk = @IDUtilizador) THEN*  *CALL cannot\_update\_error;*  *END IF;*  *END*  *[*Implementado mas diferente do Especificado]  12. Nome Trigger: medicoes\_BEFORE\_DELETE  Trigger para verificação se o Investigador que elimina a variável medida é o Investigador que está associado à cultura chamando um comando de alerta de erro caso isso não seja verdadeiro.  *CREATE DEFINER=`root`@`localhost` TRIGGER `medicoes\_BEFORE\_DELETE` BEFORE DELETE ON `medicoes` FOR EACH ROW BEGIN*  *SET @IDUtilizador = CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER);*    *IF NOT EXISTS (SELECT 1 FROM cultura c WHERE c.IDCultura = OLD.IDCultura\_fk AND c.IDUtilizador\_fk = @IDUtilizador) THEN*  *CALL cannot\_delete\_error;*  *END IF;*  *END*  13. Nome Trigger: variaveis\_AFTER\_INSERT  Trigger para envio de informação, depois da Inserção, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveis\_AFTER\_INSERT` AFTER INSERT ON `variaveis` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.variaveis\_log*  *(IDLogUtilizador*  *, IDVariaveis*  *, NomeVariaveis*  *, IDCultura*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, NEW.IDVariaveis*  *, NEW.NomeVariaveis*  *, NEW.IDCultura\_fk*  *, 'I'*  *, NOW());*  *END*  14. Nome Trigger: variaveis\_AFTER\_UPDATE  Trigger para envio de informação, depois da Actualização, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveis\_AFTER\_UPDATE` AFTER UPDATE ON `variaveis` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.variaveis\_log*  *(IDLogUtilizador*  *, IDVariaveis*  *, NomeVariaveis*  *, IDCultura*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, NEW.IDVariaveis*  *, NEW.NomeVariaveis*  *, NEW.IDCultura\_fk*  *, 'U'*  *, NOW());*  *END*  15. Nome Trigger: variaveis\_AFTER\_DELETE  Trigger para envio de informação, depois da Eliminação, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveis\_AFTER\_DELETE` AFTER DELETE ON `variaveis` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.variaveis\_log*  *(IDLogUtilizador*  *, IDVariaveis*  *, NomeVariaveis*  *, IDCultura*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, OLD.IDVariaveis*  *, OLD.NomeVariaveis*  *, OLD.IDCultura\_fk*  *, 'D'*  *, NOW());*  *END*  *[*Implementado mas diferente do Especificado]  16. Nome Trigger: variaveis\_BEFORE\_INSERT  Trigger para verificação se o Investigador que insere a variável é o Investigador que está associado à cultura chamando um comando de alerta de erro caso isso não seja verdadeiro.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveis\_BEFORE\_INSERT` BEFORE INSERT ON `variaveis` FOR EACH ROW BEGIN*  *SET @IDUtilizador = CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER);*    *IF NOT EXISTS (SELECT 1 FROM cultura c WHERE c.IDCultura = NEW.IDCultura\_fk AND c.IDUtilizador\_fk = @IDUtilizador) THEN*  *CALL cannot\_insert\_error;*  *END IF;*  *END*  *[*Implementado mas diferente do Especificado]  17. Nome Trigger: variaveis\_BEFORE\_UPDATE  Trigger para verificação se o Investigador que actualiza a variável é o Investigador que está associado à cultura chamando um comando de alerta de erro caso isso não seja verdadeiro.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveis\_BEFORE\_UPDATE` BEFORE UPDATE ON `variaveis` FOR EACH ROW BEGIN*  *SET @IDUtilizador = CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER);*    *IF NOT EXISTS (SELECT 1 FROM cultura c WHERE c.IDCultura = NEW.IDCultura\_fk AND c.IDUtilizador\_fk = @IDUtilizador) THEN*  *CALL cannot\_update\_error;*  *END IF;*  *END*  *[*Implementado mas diferente do Especificado]  18. Nome Trigger: variaveis\_BEFORE\_DELETE  Trigger para verificação se o Investigador que elimina a variável é o Investigador que está associado à cultura chamando um comando de alerta de erro caso isso não seja verdadeiro.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveis\_BEFORE\_DELETE` BEFORE DELETE ON `variaveis` FOR EACH ROW BEGIN*  *SET @IDUtilizador = CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER);*    *IF NOT EXISTS (SELECT 1 FROM cultura c WHERE c.IDCultura = OLD.IDCultura\_fk AND c.IDUtilizador\_fk = @IDUtilizador) THEN*  *CALL cannot\_delete\_error;*  *END IF;*  *END*  19. Nome Trigger: variaveismedidas\_AFTER\_INSERT  Trigger para envio de informação, depois da Inserçao, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveismedidas\_AFTER\_INSERT` AFTER INSERT ON `variaveismedidas` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.variaveismedidas\_log*  *(IDLogUtilizador*  *, IDCultura*  *, IDVariavel*  *, LimiteSuperior*  *, LimiteInferior*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, NEW.IDCultura\_fk*  *, NEW.IDVariavel\_fk*  *, NEW.LimiteSuperior*  *, NEW.LimiteInferior*  *, 'I'*  *, NOW());*  *END*  20. Nome Trigger: variaveismedidas\_AFTER\_UPDATE  Trigger para envio de informação, depois da Actualização, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveismedidas\_AFTER\_UPDATE` AFTER UPDATE ON `variaveismedidas` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.variaveismedidas\_log*  *(IDLogUtilizador*  *, IDCultura*  *, IDVariavel*  *, LimiteSuperior*  *, LimiteInferior*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, NEW.IDCultura\_fk*  *, NEW.IDVariavel\_fk*  *, NEW.LimiteSuperior*  *, NEW.LimiteInferior*  *, 'U'*  *, NOW());*  *END*  21. Nome Trigger: variaveismedidas\_AFTER\_DELETE  Trigger para envio de informação, depois da Eliminação, enviado para a tabela log correspondente.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveismedidas\_AFTER\_DELETE` AFTER DELETE ON `variaveismedidas` FOR EACH ROW BEGIN*  *INSERT INTO sid\_bd\_php.variaveismedidas\_log*  *(IDLogUtilizador*  *, IDCultura*  *, IDVariavel*  *, LimiteSuperior*  *, LimiteInferior*  *, Operacao*  *, Data)*  *VALUES (CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER)*  *, OLD.IDCultura\_fk*  *, OLD.IDVariavel\_fk*  *, OLD.LimiteSuperior*  *, OLD.LimiteInferior*  *, 'D'*  *, NOW());*  *END*  *[*Implementado mas diferente do Especificado]  22. Nome Trigger: variaveismedidas\_BEFORE\_INSERT  Trigger para verificação se o Investigador que Insere a variável e a cultura é o Investigador que está associado à cultura chamando um comando de alerta de erro caso isso não seja verdadeiro.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveismedidas\_BEFORE\_INSERT` BEFORE INSERT ON `variaveismedidas` FOR EACH ROW BEGIN*  *SET @IDUtilizador = CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER);*    *IF NOT EXISTS (SELECT 1 FROM cultura c WHERE c.IDCultura = NEW.IDCultura\_fk AND c.IDUtilizador\_fk = @IDUtilizador) THEN*  *CALL cannot\_insert\_error;*  *END IF;*  *END*  *[*Implementado mas diferente do Especificado]  23. Nome Trigger: variaveismedidas\_BEFORE\_UPDATE  Trigger para verificação se o Investigador que Actualiza a variável e a cultura é o Investigador que está associado à cultura chamando um comando de alerta de erro caso isso não seja verdadeiro.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveismedidas\_BEFORE\_UPDATE` BEFORE UPDATE ON `variaveismedidas` FOR EACH ROW BEGIN*  *SET @IDUtilizador = CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER);*    *IF NOT EXISTS (SELECT 1 FROM cultura c WHERE c.IDCultura = NEW.IDCultura\_fk AND c.IDUtilizador\_fk = @IDUtilizador) THEN*  *CALL cannot\_update\_error;*  *END IF;*  *END*  *[*Implementado mas diferente do Especificado]  24. Nome Trigger: variaveismedidas\_BEFORE\_DELETE  Trigger para verificação se o Investigador que Elimina a variável e a cultura é o Investigador que está associado à cultura chamando um comando de alerta de erro caso isso não seja verdadeiro.  *CREATE DEFINER=`root`@`localhost` TRIGGER `variaveismedidas\_BEFORE\_DELETE` BEFORE DELETE ON `variaveismedidas` FOR EACH ROW BEGIN*  *SET @IDUtilizador = CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER);*    *IF NOT EXISTS (SELECT 1 FROM cultura c WHERE c.IDCultura = OLD.IDCultura\_fk AND c.IDUtilizador\_fk = @IDUtilizador) THEN*  *CALL cannot\_delete\_error;*  *END IF;*  *END* |

Lista de Stored Procedures

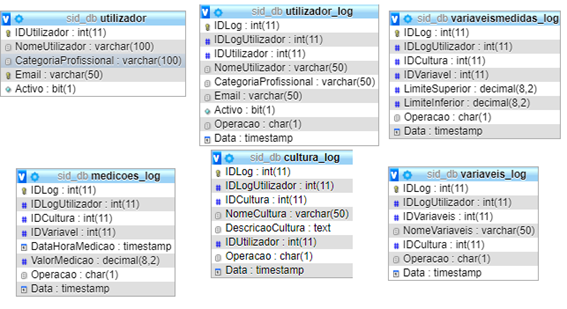
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Lista de SP (para cada SP assinalar com x em célula correspondente)**     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Implementado de Acordo com Especificado | Implementado mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | ApagarUtilizador |  |  |  | x | | CriarUtilizador | x |  |  |  | | EditarUtilizador |  |  |  | x | | LerMedicoes |  |  |  | x | |

### Stored Procedures Implementados

|  |
| --- |
| 1. Nome SP: ApagarUtilizador  Não elimina a linha correspondente ao utilizador, mas altera o seu estado de Activo para Inactivo colocando o Activo = False.  *CREATE DEFINER=`root`@`localhost` PROCEDURE `ApagarUtilizador`(`NewIDUtilizador` INT)*  *BEGIN*  *UPDATE sid\_bd\_php.utilizador SET Activo = False WHERE IDUtilizador = @NewIDUtilizador;*  *IF ROW\_COUNT() = 1 THEN*  *SET @User = CONCAT('"',@NewIDUtilizador,'"@"localhost"');*  *SET @query1 = CONCAT('DROP USER IF EXISTS ',@User,';');*  *PREPARE stmt FROM @query1; EXECUTE stmt; DEALLOCATE PREPARE stmt;*  *END IF;*  *END*  2. Nome SP: CriarUtilizador  A partir de querys permite a criação de um Utilizador (Investigador ou Auditor).  *CREATE DEFINER=`root`@`localhost` PROCEDURE `EditarUtilizador`(`NewIDUtilizador` INT, `NewNomeUtilizador` VARCHAR(100), `NewCategoriaProfissional` VARCHAR(100), `NewEmail` VARCHAR(50), `NewActivo` BIT(1), `NewPassUtilizador` VARCHAR(100))*  *BEGIN*  *IF @NewCategoriaProfissional = 'Investigador' OR @NewCategoriaProfissional = 'Auditor' OR @NewCategoriaProfissional = 'Administrador' THEN*  *UPDATE sid\_bd\_php.utilizador*  *SET Email = @NewEmail, CategoriaProfissional = @NewCategoriaProfissional, NomeUtilizador = @NewNomeUtilizador, Activo = @NewActivo*  *WHERE IDUtilizador = @NewIDUtilizador;*  *SET @User = CONCAT('"',@NewIDUtilizador,'"@"localhost"');*  *SET @UserActivo = (SELECT Activo FROM utilizador WHERE IDUtilizador = @NewIDUtilizador);*    *IF @UserActivo = false AND @Activo = true THEN*  *SET @query1 = CONCAT('CREATE USER IF NOT EXIST ',@User,' IDENTIFIED BY "',@NewPassUtilizador,'";');*  *PREPARE stmt FROM @query1; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query2 = CONCAT('GRANT ',@NewCategoriaProfissional,' TO ', @User,';');*  *PREPARE stmt FROM @query2; EXECUTE stmt; DEALLOCATE PREPARE stmt;*  *ELSE*  *SET @query1 = CONCAT('SET PASSWORD FOR ',@User,' = PASSWORD(',@NewPassUtilizador,');');*  *PREPARE stmt FROM @query1; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query2 = CONCAT('REVOKE ALL PRIVILEGES, GRANT OPTION FROM ',@NewIDUtilizador,';');*  *PREPARE stmt FROM @query2; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query3 = CONCAT('GRANT ',@NewCategoriaProfissional,' TO ', @User,';');*  *PREPARE stmt FROM @query3; EXECUTE stmt; DEALLOCATE PREPARE stmt;*  *END IF;*    *END IF;*  *END*  3. Nome SP: EditarUtilizador  A partir de querys irá fazer alterações na tabela de Utilizador no utilizador em questão, também permite a sua criação caso este não exista.  *CREATE DEFINER=`root`@`localhost` PROCEDURE `EditarUtilizador`(`NewIDUtilizador` INT, `NewNomeUtilizador` VARCHAR(100), `NewCategoriaProfissional` VARCHAR(100), `NewEmail` VARCHAR(50), `NewActivo` BIT(1), `NewPassUtilizador` VARCHAR(100))*  *BEGIN*  *IF @NewCategoriaProfissional = 'Investigador' OR @NewCategoriaProfissional = 'Auditor' OR @NewCategoriaProfissional = 'Administrador' THEN*  *UPDATE sid\_bd\_php.utilizador*  *SET Email = @NewEmail, CategoriaProfissional = @NewCategoriaProfissional, NomeUtilizador = @NewNomeUtilizador, Activo = @NewActivo*  *WHERE IDUtilizador = @NewIDUtilizador;*  *SET @User = CONCAT('"',@NewIDUtilizador,'"@"localhost"');*  *SET @UserActivo = (SELECT Activo FROM utilizador WHERE IDUtilizador = @NewIDUtilizador);*    *IF @UserActivo = false AND @Activo = true THEN*  *SET @query1 = CONCAT('CREATE USER IF NOT EXIST ',@User,' IDENTIFIED BY "',@NewPassUtilizador,'";');*  *PREPARE stmt FROM @query1; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query2 = CONCAT('GRANT ',@NewCategoriaProfissional,' TO ', @User,';');*  *PREPARE stmt FROM @query2; EXECUTE stmt; DEALLOCATE PREPARE stmt;*  *ELSE*  *SET @query1 = CONCAT('SET PASSWORD FOR ',@User,' = PASSWORD(',@NewPassUtilizador,');');*  *PREPARE stmt FROM @query1; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query2 = CONCAT('REVOKE ALL PRIVILEGES, GRANT OPTION FROM ',@NewIDUtilizador,';');*  *PREPARE stmt FROM @query2; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query3 = CONCAT('GRANT ',@NewCategoriaProfissional,' TO ', @User,';');*  *PREPARE stmt FROM @query3; EXECUTE stmt; DEALLOCATE PREPARE stmt;*  *END IF;*    *END IF;*  *END*  4. Nome SP: LerMedicoes  A partir de uma “DataInicio” e “DataFim” é possível fazer uma filtragem para melhor leitura das tabelas medicoes, medicoesluminiosidade e medicoestemperatura.  *CREATE DEFINER=`root`@`localhost` PROCEDURE `LerMedicoes`(`DataHoraMedicaoInicio` DATETIME, `DataHoraMedicaoFim` DATETIME)*  *BEGIN*  *SET @IDUtilizador = CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER);*  *SELECT \* FROM sid\_bd\_php.medicoes m*  *INNER JOIN cultura c ON c.IDCultura = m.IDCultura\_fk*  *AND c.IDUtilizador\_fk = @IDUtilizador*  *WHERE m.DataHoraMedicao >= @DataHoraMedicaoInicio*  *AND m.DataHoraMedicao <= @DataHoraMedicaoFim;*  *SELECT \* FROM sid\_bd\_php.medicoesluminosidade*  *WHERE DataHoraMedicao >= @DataHoraMedicaoInicio*  *AND DataHoraMedicao <= @DataHoraMedicaoFim;*  *SELECT \* FROM sid\_bd\_php.medicoestemperatura*  *WHERE DataHoraMedicao >= @DataHoraMedicaoInicio*  *AND DataHoraMedicao <= @DataHoraMedicaoFim;*  *END* |

## Especificação de Migração entre Bases de Dados

### Esquema relacional da base de Dados Mysql especificada (destino)



No que consta na Migração entre Base de Dados, a exportação deverá ser incremental e ser realizada ou por Ficheiro/Script ou através de PHP.

Nesta etapa será abordada a opção de se migrar dados utilizando PHP.

Conforme o esquema relacional da Base de Dados apresentado anteriormente, o intuito da base de dados destino será de guardar os registos de todas as operações de escrita, nomeadamente registos de inserção, alteração e de remoção de dados nas tabelas. Para melhor entendimento, as tabelas relativas à auditoria de dados serão designadas da seguinte maneira:

" NomeDaTabelaOrigem\_log ". Desta forma, apresenta-se uma solução coerente e de fácil gestão das mesmas e já que estas serão iguais às da base de dados origem, leva a um acompanhamento facilitado por parte do auditor na consulta e análise de acessos feitos.

Relativamente às relações entre as tabelas "\_log", estas serão inexistentes já que as tabelas guardarão informação independente.

Estas terão como chave primária o campo "IDLog", que terá comportamento de "Auto-Increment", querendo isto dizer que, futuramente, não haverão repetições. Esta abordagem leva a uma configuração mais simples como também uma fácil utilização das mesmas, bem como uma chave primária de tamanho reduzido.

Passando agora para os outros campos, "IDLogUtilizador" referir-se-á ao ID do utilizador que realizou a edição, seguido dos campos da tabela à qual o “\_log” se refere (neste caso tabela “Utilizador”). As restantes tabelas de “\_log” seguem o mesmo tipo de raciocínio.

Em relação ao campo "Operação", este guardará o tipo de alteração : "I" para uma inserção, "U" para uma alteração de um dado já existente e "D" caso se tratar de uma remoção.

Passando agora para a tabela “Utilizador”, a razão de esta se encontrar presente na BD destino é para servir de auxílio na análise e consulta das tabelas “\_log”, visto ser necessário e pertinente associar as edições/alterações da base de dados ao seu autor/editor.

Esta tabela terá opções de inserir, editar e remover dados, o que não acontece com os “\_log”s, já que, neste caso, só será oportuno inserir.

No caso da BD origem e destino se encontrarem em servidores diferentes, seria necessário ter na BD destino criar os “Users” do servidor de origem uma vez que os utilizadores criados no servidor da BD Origem não migrariam para o Servidor com BD destino.

Neste caso, não haverá essa preocupação, visto que as BDs estão alojadas no mesmo servidor. Sendo assim, a gestão dos Users do servidor é apenas efetuada na BD Origem.

### Forma de Migração Específicada

Esta migração de dados pode ser feita, na ótica de avaliação do projeto, de 2 formas:

* via ficheiro;
* via PHP.

Nesta fase do relatório, será apenas especificada a migração com recurso a código PHP.

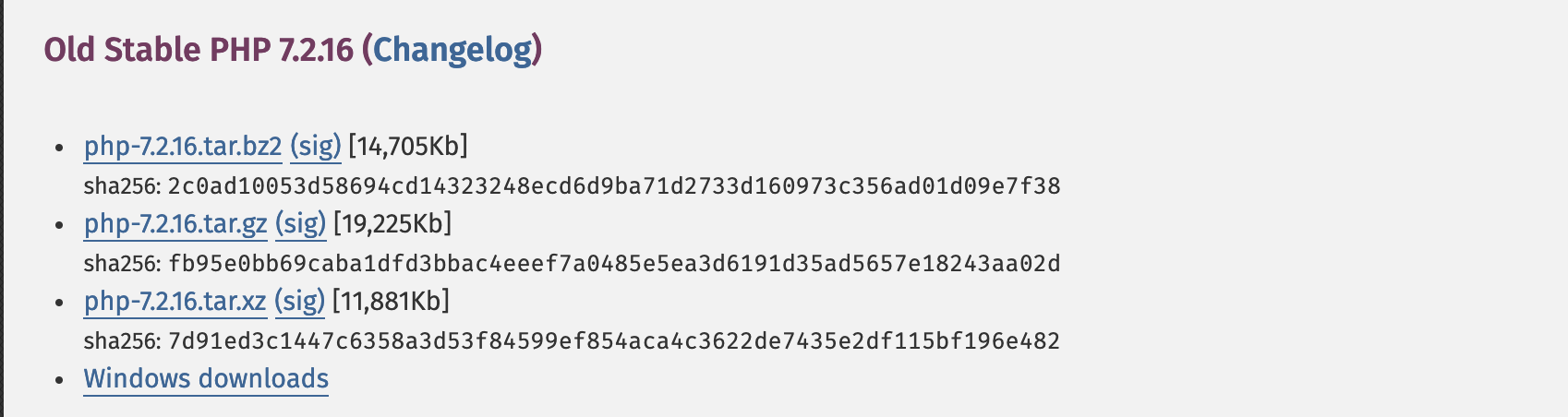
Para tal, é necessário a instalação do seguinte software:

* Xampp;
* MySQL;
* PHP;

INSTALAÇÃO PHP:

Relativamente à instalação do PHP, segue-se o link seguinte para o download da linguagem: <http://php.net/downloads.php>

Depois de ter acedido ao site de download referido anteriormente, depara-se com a página que oferece o serviço em várias versões e para sistemas operativos diferentes. Porém neste caso, para este projecto, será necessário a versão referida no anexo seguinte.



Segue pelo link que indica "Windows downloads" para realizar o download pretendido. Escolha a plataforma do PHP em relação ao seu computador, x86 ou x64 e tenha atenção, também, as versões "Thread Safe(TS) ou Non Thread Safe(NTS)". Neste caso, selecione a "Thread Safe(TS)".

Finalizado a transferência, descompacte o ficheiro ".zip" e renomeie a pasta para simplesmente "php". Após ter completado este passo, deverá movê-la para a partição principal da sua máquina, de forma a facilitar o processo de instalação (Normalmente designada por "C:").

Dependendo da versão do php transferida, poderá necessitar de fazer também o download do Microsoft Visual C++. A versão a utilizar, estará referida aqui :



Poderá realizar o download desta ferramenta por este link: <https://www.microsoft.com/en-us/download/details.aspx?id=48145>

Depois da instalação concluída, terá que ativar algumas configurações para o PHP funcionar corretamente. Dito isto, dentro da pasta do php (pasta editada nos passos anteriores) encontrará vários arquivos e executáveis. Por agora preste atenção ao ficheiro "php.ini-development". Renomeie este ficheiro para "php.ini", já que será este o ficheiro que o executável "php.exe" utilizará para carregar as configurações. Abra-o com o editor de texto que preferir e procure por  *extension\_dir = “ext”* e remova a " ; " para retirar de comentário.



Aproveite e faça o mesmo procedimento para a *extension=mysqli* conforme é exemplificado com a imagem seguinte:



De seguida precisará de configurar a PATH de forma permitir executar o php num terminal. Desta forma:

1- Em Pesquisar, procure e selecione: Sistema (Painel de Controlo);

2- Clique no link Configurações avançadas do sistema;

3- Clique em Variáveis de Ambiente. Na seção Variáveis do Sistema, localize a variável de ambiente PATH e selecione-a. Clique em Editar;

4- No campo de texto, adicione o caminho onde se encontra o PHP (C:\php).

Por fim, para verificar se o PHP ficou bem instalado e funcional, basta aceder à "cmd" do Windows e correr o seguinte comando : " php -v ".

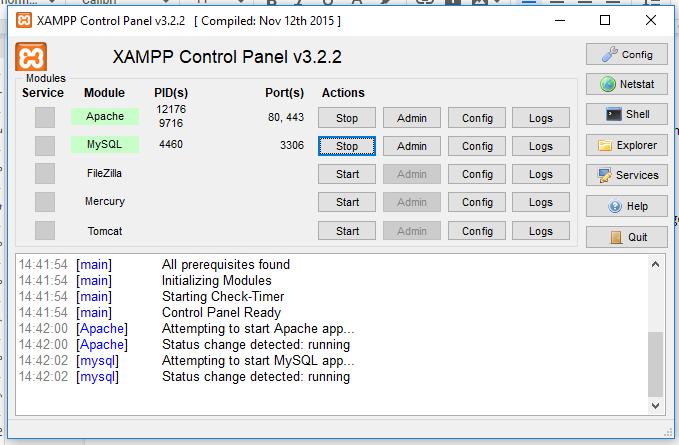
NOTA: Se conhecer um método mais eficaz e mais rápido que o descrito, por favor use-o. Apenas foi descrito uma das maneiras de instalar PHP.

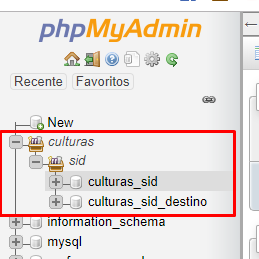
INSTALAÇÃO XAMPP E MYSQL:

A instalação de PHP é a mais polêmica, pois requer passos mais complicados para conseguir no mínimo compilar um ficheiro php.

Já as outras aplicações necessárias são mais simples e intuitivas, bastando apenas iniciar os executáveis respectivos, e à partida não será preciso mais nenhuma configuração.

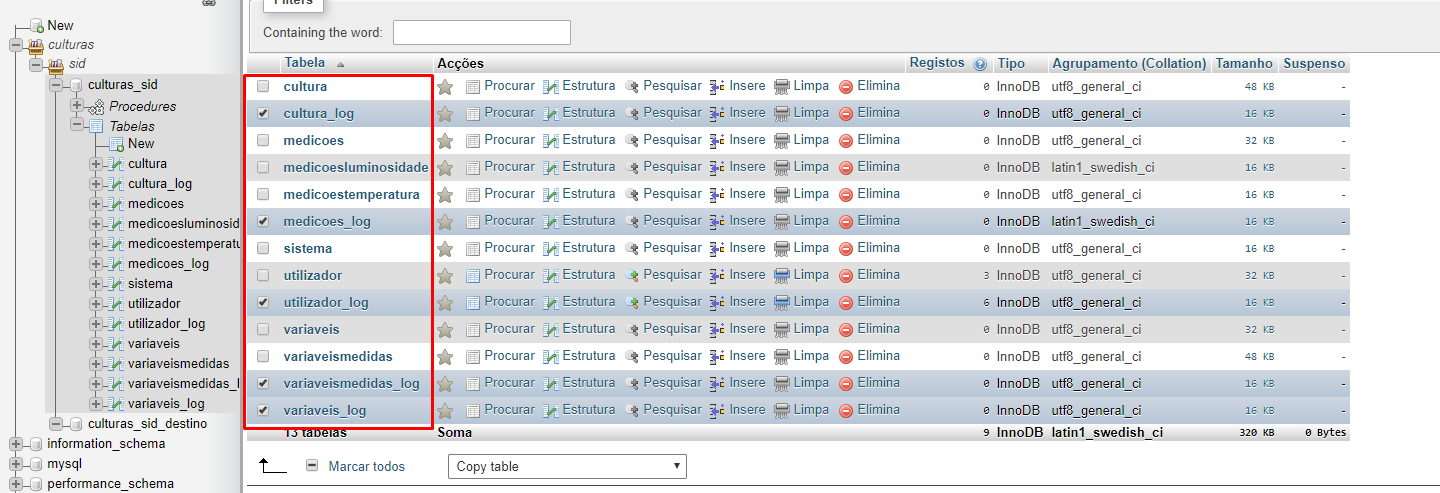
Assim, para começar, é necessário iniciar o XAMPP e ativar o serviço “Apache” e “MySQL”.



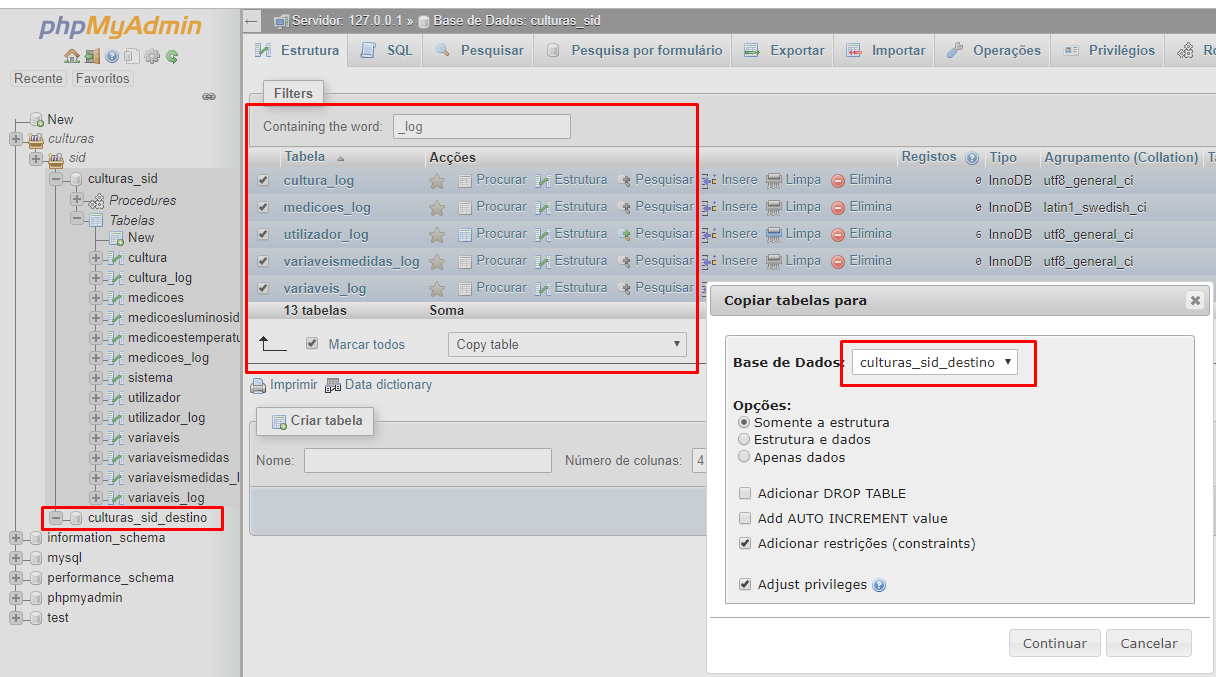
Tendo a BD origem pronta, agora é preciso começar a migração da origem para a destino, migrando apenas algumas das tabelas, ou seja as tabelas de auditoria.

Em primeiro lugar, é necessário criar a BD “Destino”. Na imagem podemos ver que temos 2 bases de dados, “culturas\_sid” que é a BD Origem. Depois temos a BD “culturas\_sid\_destino”, que para o caso é a BD que tem de ser criada nesta fase.

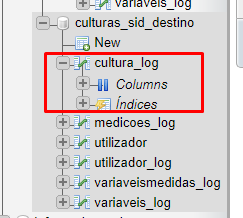
Quando as BD’s são criadas, estas vêm vazias. Portanto, para os fins que queremos, ou seja, para migrar os dados de auditoria da BD origem temos de criar as tabelas necessárias. Para tal, podemos copiar da origem, quando estas ainda estão vazias.



Por questão de coerência, e para facilitar a gestão, todas as tabelas de auditoria estão criadas com “\_log”. Para além das tabelas “\_log”, a base de destino vai precisar da tabela “Utilizador” para permitir ao auditor consultar quem fez o que, de forma a ter acesso aos dados do Utilizador que fez a atualização. Assim, para copiar a estrutura das tabelas de auditoria basta procurar “\_log”, selecionar as tabelas de interesse, e escolher “Copy Table”.



Feito isto, basicamente, a BD destino vai ficar com a estrutura de auditoria criada, ou seja, fica com as tabelas de auditoria apenas. Na imagem seguinte pode-se ver “culturas\_sid\_destino”, que não tinha tabelas, mas que agora já tem 6 tabelas vazias, prontas para receber a migração de dados.



Como podemos ver, a tabela “cultura\_log” está completamente vazia, tal como as outras, à espera de serem povoadas com a informação proveniente da origem.

Esta povoação, periódica e incremental, nesta etapa é feita com recurso a código PHP. Então, é necessário colocá-lo na pasta “~\xampp\htdocs” e corrido com recurso ao xampp para que dê início a primeira migração.

### Utilizadores Especificados

Nesta fase, apenas haverá um utilizador a operar na base de dados de destino, que será o auditor. Para além disso, o auditor nunca poderá alterar ou inserir dados nestas tabelas, apenas poderá consultar.

|  |  |
| --- | --- |
| **Tabela** | **Tipo de Utilizador** |
| Auditor |
| Medições\_Log | L |
| Utilizador\_Log | L |
| Variáveis\_Log | L |
| Cultura\_Log | L |
| VariáveisMedidas\_Log | L |
| Utilizador | L |
|  |  |
| **Stored Proc.** |  |
| CriarUtilizador | - |
| AlterarUtilizador | - |
| ApagarUtilizador | - |
| CriarMedições\_Log | - |
| CriarUtilizador\_Log | - |
| CriarVariáveis\_Log | - |
| CriarCultura\_Log | - |
| CriarVariáveisMedidas\_Log | - |

Legenda:

E = Escrita

L = Leitura

X = Executar

- = sem permissões

Então, quer isto dizer que o auditor vai ser o único com acesso a BD Destino. E embora seja o único, não vai ter acessos para além de leitura.

Para além disso, estas tabelas vão ter vários SPs, que vão funcionar independente do auditor, porque vão ser automáticos no seu preenchimento. Ou seja, vão preencher tabelas, tabelas estas que o auditor poderá depois consultar.

### Triggers de suporte à migração de dados especificados

(não relevante)

### Stored Procedures de suporte à migração de dados especificados

As SP’s de criação de Log’s são uma copia das SP’s de criação de Log’s da BD Origem, as SP’s de gestão dos utilizadores(CriarUtilizador, EditarUtilizador) estas alteram os dados da tabela utilizadores, foi decidido a utilização de SP’s para esta gestão de forma a facilitar o código de migração em PHP.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome Procedimento | Parâmetros Entrada | Parâmetros Saída | BD  (Origem ou Destino) | Muito breve descrição |
| CriarUtilizador | @Email, @CategoriaProfissional, @NomeUtilizador, @PassUtilizador, |  | Destino | Popular a tabela Utilizadores na BD destino |
| EditarUtilizador | @IDUtilizador,  @Email, @CategoriaProfissional, @NomeUtilizador, |  | Destino | Editar a tabela Utilizadores na BD destino |
| CriarLogCultura | @IDLogUtilizador,  @IDCultura,  @NomeCultura,  @DescriçãoCultura,  @IDUtilizador,  @Operacao,  @Data |  | Destino | Popular tabela Cultura\_Log na BD destino |
| CriarLogUtilizador | @IDLogUtilizador,  @IDUtilizador,  @NomeUtilizador,  @CategoriaProfissional,  @Email,  @Activo,  @Operação,  @Char |  | Destino | Popular tabela Utilizador\_Log na BD destino |
| CriarLogMedicoes | @IDLogUtilizador,  @IDMedicoes,  @IDVariavel,  @DataHoraMedicao,  @ValorMedicao,  @Operacao,  @Data |  | Destino | Popular tabela Medicoes\_Log na BD destino |
| CriarLogVariaveis | @IDLogUtilizador,  @IDVariaveis,  @NomeVariaveis,  @IDCultura\_fk,  @Operacao,  @Data |  | Destino | Popular tabela Variaveis\_Log na BD destino |
| CriarLogVariaveisMedidas | @IDLogUtilizador,  @IDVariavel,  @LimiteSuperior,  @LimiteInferior,  @Operacao,  @Data |  | Destino | Popular tabela VariaveisMedidas\_Log na BD destino |

### Eventos de suporte à migração de dados especificados

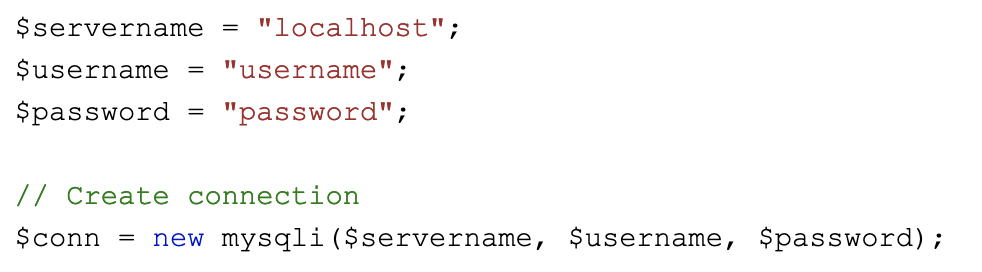
|  |  |  |
| --- | --- | --- |
| Nome Evento | Local Execução  (Origem ou Destino, ou Sistema Operativo) | Muito breve descrição |
| Trigger | Origem | Desvantagem em termos de segurança dado estar a criar uma relação com o computador(que não é bom), dependente também do número de vezes que se inserir um dado, pois será número de vezes que executar a migração o que também não é bom em termos de performance |
| MySQL | Origem | Desvantagem em termos de segurança por estar a ter uma relação externa com o computador pois ter-se-á de correr o ficheiro PHP, sendo que ter SQL a correr ficheiros também não é aconselhável |
| Serviço | Sistema Operativo | Define-se uma altura de x em x tempo em que se vai correr o serviço que irá abrir o ficheiro PHP; Corre um ficheiro bash e esse ficheiro vai correr o ficheiro PHP para fazer a migração, pode se criar mais do que um serviço e nesse caso como nós vamos ter os utilizadores, os utilizadores podem ser mais recorrentes que os logs. Os logs podem ser de 12 em 12 horas (relativo) e os utilizadores de hora em hora; Os logs que terão mais dados, são corridos menos vezes dado a sobrecarga do sistema, para eficiência e performance não serem afectados. |

### PHP de suporte à migração de dados especificado

Nesta secção será explicada a lógica base relativa à migração com suporte PHP.

Comecemos pelos pontos mais genéricos para facilitar o entendimento do processo em questão. Em primeiro lugar, para que tudo funcione corretamente, terá que se conectar à base de dados criada anteriormente (explicação na secção 1.4.2).

Este processo poderá e deverá ser desenvolvido em mais detalhe e aprimoramento, comparativamente ao exemplo que se segue:



Passando agora para o fecho da ligação à base de dados, que fique claro que todas as vezes que se realiza uma ligação prévia a uma BD e o script acabar de correr, a ligação será fechada automaticamente, porém poderá encerrá-la manualmente (se assim o decidir, aconselha-se a que realize a desconexão no final do script, para que não tenha que realizar uma ligação nova a cada "query" ou processo, e assim evitar "despesas operacionais" adicionais - overhead).

Relativamente a lógica por detrás da migração, foi decidido, para melhor compreensão, em dividir a explicação em duas partes : 1) Logs e 2) Utilizador.

1)Logs

Como as "\_logs" serão, supostamente, tabelas editadas com uma maior quantidade de dados, a disparidade entre a base de dados origem e destino, num intervalo de 12 em 12 horas, poderá ser grande. Tendo isto em conta, uma solução, para uma migração incremental e sem repetições, será a de comparar os "IDs" relativos às tabelas "\_log" origem e destino. Desta forma, facilitaria a forma de migrar dados entre BDs.

Seguindo este raciocínio, após o estabelecimento da ligação às base de dados (origem e destino), uma opção fiável seria a de consultar a tabela origem, consultar a tabela destino e por fim comparar os ID’s de forma a detetar os novos dados inseridos na tabela origem. Por fim inseri-los na tabela destino, recorrendo a um SP relativo à tabela em causa.

2)Utilizador

Este caso será mais trabalhoso, visto que nas tabelas "\_log", referidas previamente, os dados que se encontrariam nestas não deverão ser editados nem removidos, o que não acontece no caso das tabelas "Utilizador".

Tendo em conta a nossa forma de remoção do utilizador sendo esta: Inativação do utilizador(Coluna Activo = False), esta é efetuada na edição do utilizador.

Desta forma, deverá :

a) Inserir novos dados para a tabela "Utilizador" da BD destino, semelhante ao processo explicado para as tabelas "\_log" com a ajuda de um SP CriarUtilizador;

b) No caso de se tratar de uma operação de renovação/alteração de um dado qualquer sobre um utilizador já existente em ambas as BDs (origem e destino), utiliza-se o IDUtilizador como referência. Aconselha-se a editar todos os dados, mesmo que estes não tenham sido alterados, uma vez que o MySql não altera dados que sejam iguais. Recorremos a esta forma de edição uma vez que perderíamos performance se verificarmos todas as colunas em PHP. Por fim, correr SP EditarUtilizador;

## Avaliação das especificações do próprio grupo Migração

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Boa  Justificação:  Bem especificada e em detalhe. Foram cumpridos os requisitos mais importantes, nomeadamente ser uma exportação incremental e segura, já que a migração por PHP não utilizará recursos externos à BD de forma a migrar para a BD destino.  Porém considera-se um dos pontos mais frágeis os Eventos de suporte à migração de dados, já que não se entrou muito em detalhe sobre o assunto, apenas as vantagens e desvantagens. |

## Implementação da Migração de Dados

### Utilizadores Implementado

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Tabela** | **Tipo de Utilizador** | | | | UtilizadorMigracao | | | | cultura | - | | | | cultura\_log | E,L(Destino)/L(Origem) | | | | medicoes | - | | | | medicoes\_log | E,L(Destino)/L(Origem) | | | | medicoesluminosidades | - | | | | medicoestemperatura | - | | | | sistema | - | | | | utilizador | E,L(Destino)/L(Origem) | | | | utilizador\_log | E,L(Destino)/L(Origem) | | | | variaveis | - | | | | variaveis\_log | E,L(Destino)/L(Origem) | | | | variaveismedidas | - | | | | variaveismedidas\_log | E,L(Destino)/L(Origem) | | | | **Stored Proc.** |  | | | | CriarUtilizador | - | | | | EditarUtilizador | - | | | | ApagarUtilizador | - | | | | LerMedições | - | | | |

### Lista Triggers

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Lista de Triggers (para cada trigger assinalar com x em célula correspondente)**     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Implementado de Acordo com Especificado | Implementado mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | Nome Trigger  (tal como especificado) |  |  |  |  | | Nome Trigger  (tal como especificado) |  |  |  |  | | Nome Trigger  (tal como especificado) |  |  |  |  | |  |  |  |  |  | |

### Triggers Implementados

|  |
| --- |
| 1. Nome Trigger: \_\_\_\_\_  //*Breve Descrição*  *Código*  2. Nome Trigger: \_\_\_\_\_  //*Breve Descrição*  *Código*  3. Nome Trigger: \_\_\_\_\_  //*Breve Descrição*  *Código* |

### Lista de Stored Procedures

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Lista de SP (para cada SP assinalar com x em célula correspondente)**     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Implementado de Acordo com Especificado | Implementado mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | ApagarUtilizador |  |  |  | x | | CriarUtilizador | x |  |  |  | | EditarUtilizador |  |  |  | x | | LerMedicoes |  |  |  | x | |

### Stored Procedures Implementados

|  |
| --- |
| 1. Nome SP: ApagarUtilizador  Não elimina a linha correspondente ao utilizador, mas altera o seu estado de Activo para Inactivo colocando o Activo = False.  *CREATE DEFINER=`root`@`localhost` PROCEDURE `ApagarUtilizador`(`NewIDUtilizador` INT)*  *BEGIN*  *UPDATE sid\_bd\_php.utilizador SET Activo = False WHERE IDUtilizador = @NewIDUtilizador;*  *IF ROW\_COUNT() = 1 THEN*  *SET @User = CONCAT('"',@NewIDUtilizador,'"@"localhost"');*  *SET @query1 = CONCAT('DROP USER IF EXISTS ',@User,';');*  *PREPARE stmt FROM @query1; EXECUTE stmt; DEALLOCATE PREPARE stmt;*  *END IF;*  *END*  2. Nome SP: CriarUtilizador  A partir de querys permite a criação de um Utilizador (Investigador ou Auditor).  *CREATE DEFINER=`root`@`localhost` PROCEDURE `EditarUtilizador`(`NewIDUtilizador` INT, `NewNomeUtilizador` VARCHAR(100), `NewCategoriaProfissional` VARCHAR(100), `NewEmail` VARCHAR(50), `NewActivo` BIT(1), `NewPassUtilizador` VARCHAR(100))*  *BEGIN*  *IF @NewCategoriaProfissional = 'Investigador' OR @NewCategoriaProfissional = 'Auditor' OR @NewCategoriaProfissional = 'Administrador' THEN*  *UPDATE sid\_bd\_php.utilizador*  *SET Email = @NewEmail, CategoriaProfissional = @NewCategoriaProfissional, NomeUtilizador = @NewNomeUtilizador, Activo = @NewActivo*  *WHERE IDUtilizador = @NewIDUtilizador;*  *SET @User = CONCAT('"',@NewIDUtilizador,'"@"localhost"');*  *SET @UserActivo = (SELECT Activo FROM utilizador WHERE IDUtilizador = @NewIDUtilizador);*    *IF @UserActivo = false AND @Activo = true THEN*  *SET @query1 = CONCAT('CREATE USER IF NOT EXIST ',@User,' IDENTIFIED BY "',@NewPassUtilizador,'";');*  *PREPARE stmt FROM @query1; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query2 = CONCAT('GRANT ',@NewCategoriaProfissional,' TO ', @User,';');*  *PREPARE stmt FROM @query2; EXECUTE stmt; DEALLOCATE PREPARE stmt;*  *ELSE*  *SET @query1 = CONCAT('SET PASSWORD FOR ',@User,' = PASSWORD(',@NewPassUtilizador,');');*  *PREPARE stmt FROM @query1; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query2 = CONCAT('REVOKE ALL PRIVILEGES, GRANT OPTION FROM ',@NewIDUtilizador,';');*  *PREPARE stmt FROM @query2; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query3 = CONCAT('GRANT ',@NewCategoriaProfissional,' TO ', @User,';');*  *PREPARE stmt FROM @query3; EXECUTE stmt; DEALLOCATE PREPARE stmt;*  *END IF;*    *END IF;*  *END*  3. Nome SP: EditarUtilizador  A partir de querys irá fazer alterações na tabela de Utilizador no utilizador em questão, também permite a sua criação caso este não exista.  *CREATE DEFINER=`root`@`localhost` PROCEDURE `EditarUtilizador`(`NewIDUtilizador` INT, `NewNomeUtilizador` VARCHAR(100), `NewCategoriaProfissional` VARCHAR(100), `NewEmail` VARCHAR(50), `NewActivo` BIT(1), `NewPassUtilizador` VARCHAR(100))*  *BEGIN*  *IF @NewCategoriaProfissional = 'Investigador' OR @NewCategoriaProfissional = 'Auditor' OR @NewCategoriaProfissional = 'Administrador' THEN*  *UPDATE sid\_bd\_php.utilizador*  *SET Email = @NewEmail, CategoriaProfissional = @NewCategoriaProfissional, NomeUtilizador = @NewNomeUtilizador, Activo = @NewActivo*  *WHERE IDUtilizador = @NewIDUtilizador;*  *SET @User = CONCAT('"',@NewIDUtilizador,'"@"localhost"');*  *SET @UserActivo = (SELECT Activo FROM utilizador WHERE IDUtilizador = @NewIDUtilizador);*    *IF @UserActivo = false AND @Activo = true THEN*  *SET @query1 = CONCAT('CREATE USER IF NOT EXIST ',@User,' IDENTIFIED BY "',@NewPassUtilizador,'";');*  *PREPARE stmt FROM @query1; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query2 = CONCAT('GRANT ',@NewCategoriaProfissional,' TO ', @User,';');*  *PREPARE stmt FROM @query2; EXECUTE stmt; DEALLOCATE PREPARE stmt;*  *ELSE*  *SET @query1 = CONCAT('SET PASSWORD FOR ',@User,' = PASSWORD(',@NewPassUtilizador,');');*  *PREPARE stmt FROM @query1; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query2 = CONCAT('REVOKE ALL PRIVILEGES, GRANT OPTION FROM ',@NewIDUtilizador,';');*  *PREPARE stmt FROM @query2; EXECUTE stmt; DEALLOCATE PREPARE stmt;*    *SET @query3 = CONCAT('GRANT ',@NewCategoriaProfissional,' TO ', @User,';');*  *PREPARE stmt FROM @query3; EXECUTE stmt; DEALLOCATE PREPARE stmt;*  *END IF;*    *END IF;*  *END*  4. Nome SP: LerMedicoes  A partir de uma “DataInicio” e “DataFim” é possível fazer uma filtragem para melhor leitura das tabelas medicoes, medicoesluminiosidade e medicoestemperatura.  *CREATE DEFINER=`root`@`localhost` PROCEDURE `LerMedicoes`(`DataHoraMedicaoInicio` DATETIME, `DataHoraMedicaoFim` DATETIME)*  *BEGIN*  *SET @IDUtilizador = CONVERT(SUBSTRING\_INDEX(CURRENT\_USER(), '@', 1),UNSIGNED INTEGER);*  *SELECT \* FROM sid\_bd\_php.medicoes m*  *INNER JOIN cultura c ON c.IDCultura = m.IDCultura\_fk*  *AND c.IDUtilizador\_fk = @IDUtilizador*  *WHERE m.DataHoraMedicao >= @DataHoraMedicaoInicio*  *AND m.DataHoraMedicao <= @DataHoraMedicaoFim;*  *SELECT \* FROM sid\_bd\_php.medicoesluminosidade*  *WHERE DataHoraMedicao >= @DataHoraMedicaoInicio*  *AND DataHoraMedicao <= @DataHoraMedicaoFim;*  *SELECT \* FROM sid\_bd\_php.medicoestemperatura*  *WHERE DataHoraMedicao >= @DataHoraMedicaoInicio*  *AND DataHoraMedicao <= @DataHoraMedicaoFim;*  *END* |

### Lista Eventos

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Lista de Eventos (para cada evento assinalar com x em célula correspondente)**     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Implementado de Acordo com Especificado | Implementado mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | migracao\_php.bat |  |  |  | X | |

### Eventos Implementados

|  |
| --- |
| 1. Nome Evento: migraçao\_php.bat  Evento responsável pela migração periódica da db origem para a db destino. Este evento corre consoante o que lhe for pedido e estipulado.  @echo off  setlocal  set STARTTIME=%TIME%  echo A iniciar import!  echo(  php migracaoLOGS.php  echo(  echo Backup realizado com sucesso!  echo(  set ENDTIME=%TIME%  rem output as time  echo STARTTIME: %STARTTIME%  echo ENDTIME: %ENDTIME%  rem convert STARTTIME and ENDTIME to centiseconds  set /A STARTTIME=(1%STARTTIME:~0,2%-100)\*360000 + (1%STARTTIME:~3,2%-100)\*6000 + (1%STARTTIME:~6,2%-100)\*100 + (1%STARTTIME:~9,2%-100)  set /A ENDTIME=(1%ENDTIME:~0,2%-100)\*360000 + (1%ENDTIME:~3,2%-100)\*6000 + (1%ENDTIME:~6,2%-100)\*100 + (1%ENDTIME:~9,2%-100)  rem calculating the duratyion is easy  set /A DURATION=%ENDTIME%-%STARTTIME%  rem we might have measured the time inbetween days  if %ENDTIME% LSS %STARTTIME% set set /A DURATION=%STARTTIME%-%ENDTIME%  rem now break the centiseconds down to hors, minutes, seconds and the remaining centiseconds  set /A DURATIONH=%DURATION% / 360000  set /A DURATIONM=(%DURATION% - %DURATIONH%\*360000) / 6000  set /A DURATIONS=(%DURATION% - %DURATIONH%\*360000 - %DURATIONM%\*6000) / 100  set /A DURATIONHS=(%DURATION% - %DURATIONH%\*360000 - %DURATIONM%\*6000 - %DURATIONS%\*100)  rem some formatting  if %DURATIONH% LSS 10 set DURATIONH=0%DURATIONH%  if %DURATIONM% LSS 10 set DURATIONM=0%DURATIONM%  if %DURATIONS% LSS 10 set DURATIONS=0%DURATIONS%  if %DURATIONHS% LSS 10 set DURATIONHS=0%DURATIONHS%  rem outputing  echo DURATION: %DURATION% in centiseconds  echo %DURATIONH%:%DURATIONM%:%DURATIONS%,%DURATIONHS%  endlocal  pause |

### PHP Implementado

|  |
| --- |
| **<?php**  **$servername = "127.0.0.1";**  **$username = "userMigracao";**  **$password = "Migracao123\_";**  **$dbnameOrigem = "sid\_bd\_php"; //Mudar se for preciso**  **$dbnameDestino = "sid\_bd\_destino\_php"; //Mudar se for preciso**  **// Create connection**  **$connOrigem = new mysqli($servername, $username, $password, $dbnameOrigem);**  **$connDestino = new mysqli($servername, $username, $password, $dbnameDestino);**  **// Check connection**  **if ($connOrigem->connect\_error || $connDestino->connect\_error )**  **die("Connection failed: " . $connOrigem->connect\_error . $connDestino->connect\_error);**  **echo "Connected successfully\n";**  **ini\_set('memory\_limit', '-1');**  **echo "-------------------------Querys a construir:------------------------------------\n";**  **echo "-------------------------Querys a correr:------------------------------------\n";**  **$count=1;**  **#region CriarLogCultura**  **$tabela\_origem = get\_table\_data("cultura\_log", $connOrigem, $dbnameOrigem);**  **$tabela\_destino = get\_table\_data("cultura\_log", $connDestino, $dbnameDestino);**  **$sql="INSERT INTO sid\_bd\_destino\_php.cultura\_log**  **(IDLogUtilizador**  **, IDCultura**  **, NomeCultura**  **, DescricaoCultura**  **, IDUtilizador**  **, Operacao**  **, Data)**  **VALUES ";**  **$data\_to\_insert = data\_to\_insert($tabela\_origem, $tabela\_destino);**  **foreach($data\_to\_insert as $key => $data\_inserir) {**  **$sql.="({$data\_inserir["IDLogUtilizador"]}**  **,{$data\_inserir["IDCultura"]}**  **,'{$data\_inserir["NomeCultura"]}'**  **,'{$data\_inserir["DescricaoCultura"]}'**  **,{$data\_inserir["IDUtilizador"]}**  **,'{$data\_inserir["Operacao"]}'**  **,'{$data\_inserir["Data"]}')";**  **if ($key === array\_key\_last($data\_to\_insert))**  **$sql.=";\n";**  **else**  **$sql.=",\n";**  **echo $count++."\n";**  **}**  **$connDestino->query($sql);**  **#endregion**  **#region CriarLogMedicoes**  **$tabela\_origem = get\_table\_data("medicoes\_log", $connOrigem, $dbnameOrigem);**  **$tabela\_destino = get\_table\_data("medicoes\_log", $connDestino, $dbnameDestino);**  **$sql="INSERT INTO sid\_bd\_destino\_php.medicoes\_log**  **(IDLogUtilizador**  **, IDMedicoes**  **, IDCultura**  **, IDVariavel**  **, DataHoraMedicao**  **, ValorMedicao**  **, Operacao**  **, Data)**  **VALUES ";**  **$data\_to\_insert = data\_to\_insert($tabela\_origem, $tabela\_destino);**  **foreach($data\_to\_insert as $key => $data\_inserir) {**  **$sql.="({$data\_inserir["IDLogUtilizador"]}**  **,{$data\_inserir["IDMedicoes"]}**  **,{$data\_inserir["IDCultura"]}**  **,{$data\_inserir["IDVariavel"]}**  **,'{$data\_inserir["DataHoraMedicao"]}'**  **,{$data\_inserir["ValorMedicao"]}**  **,'{$data\_inserir["Operacao"]}'**  **,'{$data\_inserir["Data"]}')";**  **if ($key === array\_key\_last($data\_to\_insert))**  **$sql.=";\n";**  **else**  **$sql.=",\n";**  **echo $count++."\n";**  **}**  **$connDestino->query($sql);**  **#endregion**  **#region Utilizadores**  **$tabela\_origem = get\_table\_data("utilizador", $connOrigem, $dbnameOrigem);**  **$tabela\_destino = get\_table\_data("utilizador", $connDestino, $dbnameDestino);**  **$data\_to\_update = data\_to\_update($tabela\_origem, $tabela\_destino);**  **foreach($data\_to\_update as $data\_editar) {**  **$sql="UPDATE sid\_bd\_destino\_php.utilizador**  **SET Email = '{$data\_editar["Email"]}', CategoriaProfissional = '{$data\_editar["CategoriaProfissional"]}', NomeUtilizador = '{$data\_editar["NomeUtilizador"]}', Activo = {$data\_editar["Activo"]}**  **WHERE IDUtilizador = {$data\_editar["IDUtilizador"]};\n";**  **$connDestino->query($sql);**  **echo $count++."\n";**  **}**  **$sql="INSERT INTO sid\_bd\_destino\_php.utilizador**  **( NomeUtilizador**  **, CategoriaProfissional**  **, Email**  **, Activo)**  **VALUES ";**  **$data\_to\_insert = data\_to\_insert($tabela\_origem, $tabela\_destino);**  **foreach($data\_to\_insert as $key => $data\_inserir) {**  **$sql.="('{$data\_inserir["NomeUtilizador"]}'**  **, '{$data\_inserir["CategoriaProfissional"]}'**  **, '{$data\_inserir["Email"]}'**  **, {$data\_inserir["Activo"]})";**  **if ($key === array\_key\_last($data\_to\_insert))**  **$sql.=";\n";**  **else**  **$sql.=",\n";**  **echo $count++."\n";**  **}**  **$connDestino->query($sql);**  **#endregion**  **#region CriarLogUtilizador**  **$tabela\_origem = get\_table\_data("utilizador\_log", $connOrigem, $dbnameOrigem);**  **$tabela\_destino = get\_table\_data("utilizador\_log", $connDestino, $dbnameDestino);**  **$sql="INSERT INTO sid\_bd\_destino\_php.utilizador\_log**  **(IDLogUtilizador**  **, IDUtilizador**  **, NomeUtilizador**  **, CategoriaProfissional**  **, Email**  **, Activo**  **, Operacao**  **, Data)**  **VALUES ";**  **$data\_to\_insert = data\_to\_insert($tabela\_origem, $tabela\_destino);**  **foreach($data\_to\_insert as $key => $data\_inserir) {**  **$sql.="({$data\_inserir["IDLogUtilizador"]}**  **,{$data\_inserir["IDUtilizador"]}**  **,'{$data\_inserir["NomeUtilizador"]}'**  **,'{$data\_inserir["CategoriaProfissional"]}'**  **,'{$data\_inserir["Email"]}'**  **,{$data\_inserir["Activo"]}**  **,'{$data\_inserir["Operacao"]}'**  **,'{$data\_inserir["Data"]}')";**  **if ($key === array\_key\_last($data\_to\_insert))**  **$sql.=";\n";**  **else**  **$sql.=",\n";**  **echo $count++."\n";**  **}**  **$connDestino->query($sql);**  **#endregion**  **#region CriarLogVariaveis**  **$tabela\_origem = get\_table\_data("variaveis\_log", $connOrigem, $dbnameOrigem);**  **$tabela\_destino = get\_table\_data("variaveis\_log", $connDestino, $dbnameDestino);**  **$sql="INSERT INTO sid\_bd\_destino\_php.variaveis\_log**  **(IDLogUtilizador**  **, IDVariaveis**  **, NomeVariaveis**  **, IDCultura**  **, Operacao**  **, Data)**  **VALUES ";**  **$data\_to\_insert = data\_to\_insert($tabela\_origem, $tabela\_destino);**  **foreach($data\_to\_insert as $key => $data\_inserir) {**  **$sql="({$data\_inserir["IDLogUtilizador"]}**  **,{$data\_inserir["IDVariaveis"]}**  **,'{$data\_inserir["NomeVariaveis"]}'**  **,{$data\_inserir["IDCultura"]}**  **,'{$data\_inserir["Operacao"]}'**  **,'{$data\_inserir["Data"]}')";**  **if ($key === array\_key\_last($data\_to\_insert))**  **$sql.=";\n";**  **else**  **$sql.=",\n";**  **echo $count++."\n";**  **}**  **$connDestino->query($sql);**  **#endregion**  **#region CriarLogVariaveisMedidas**  **$tabela\_origem = get\_table\_data("variaveismedidas\_log", $connOrigem, $dbnameOrigem);**  **$tabela\_destino = get\_table\_data("variaveismedidas\_log", $connDestino, $dbnameDestino);**  **$sql="INSERT INTO sid\_bd\_destino\_php.variaveismedidas\_log**  **(IDLogUtilizador**  **, IDCultura**  **, IDVariavel**  **, LimiteSuperior**  **, LimiteInferior**  **, Operacao**  **, Data)**  **VALUES ";**  **$data\_to\_insert = data\_to\_insert($tabela\_origem, $tabela\_destino);**  **foreach($data\_to\_insert as $key => $data\_inserir) {**  **$sql.="({$data\_inserir["IDLogUtilizador"]}**  **,{$data\_inserir["IDCultura"]}**  **,{$data\_inserir["IDVariavel"]}**  **,{$data\_inserir["LimiteSuperior"]}**  **,{$data\_inserir["LimiteInferior"]}**  **,'{$data\_inserir["Operacao"]}'**  **,'{$data\_inserir["Data"]}')";**  **if ($key === array\_key\_last($data\_to\_insert))**  **$sql.=";\n";**  **else**  **$sql.=",\n";**  **echo $count++."\n";**  **}**  **$connDestino->query($sql);**  **#endregion**  **echo "--------------------Querys corridas com sucesso.------------------------------\n";**  **$connDestino->close();**  **$connOrigem->close();**  **function get\_table\_data($table\_name, $conn, $BD){**  **//Criacao de um array para obter as entradaas nas tabelas log da DB origem**  **$array = array();**  **$sqlLerTabela="SELECT \* FROM ".$BD.".".$table\_name.";";**  **$result = $conn->query($sqlLerTabela);**  **//Povoar o array\_origem com dados da BD origem**  **if ($result->num\_rows > 0)**  **// output de cada row**  **$array = mysqli\_fetch\_all($result, MYSQLI\_ASSOC);**  **return $array;**  **}**  **function data\_to\_insert($array\_origem, &$array\_destino)**  **{**  **if(!empty($array\_destino)){**  **$aux = array();**  **foreach ($array\_origem as $origem){**  **if(!in\_array($origem, $array\_destino)) {**  **array\_push($aux, $origem);**  **array\_push($array\_destino, $origem);**  **}**  **}**  **return $aux;**  **}else**  **return $array\_origem;**  **}**  **function data\_to\_update($array\_origem, &$array\_destino)**  **{**  **$aux = array();**  **for ($x = 0; $x < count($array\_origem); $x++){**  **for ($y = 0; $y < count($array\_destino); $y++){**  **if ($array\_origem[$x]["IDUtilizador"] == $array\_destino[$y]["IDUtilizador"]){**  **array\_push($aux, $array\_origem[$x]);**  **$array\_destino[$y] = $array\_origem[$x];**  **}**  **}**  **}**  **return $aux;**  **}** |

Avaliação Global da Qualidade das Especificações do próprio grupo

|  |
| --- |
| Avaliação (A,B,C,D,E) : D  Utilize a seguinte escala:  A: - 1 – 5 valores B: 6 – 9 valores C: 10 – 13 Valores D: 14 – 17 valores E: 18 – 20 valores |

**Três principais deficiências de especificação que tiveram impacto mais negativo na qualidade da implementação**

|  |
| --- |
| Eventos: O pouco detalhe e aprofundamento nesta secção acabou por influenciar negativamente a qualidade e facilidade de implementação do projecto. |
| Uso desnecessário de SP´s dentro dos triggers que influenciava, negativamente, a performance. Foi corrigido utilizando, apenas, script do trigger para chegar aos mesmos fins. |
| Mandatory e Unique não foram especificados no relatório e poderá ser um ponto de fraca compreensão de certos atributos. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Resumo de Avaliações de Qualidade Anteriores (para cada linha assinalar com x em célula correspondente)**   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Fraco | Razoável | Bom | Muito Bom | | BD MySQL Origem |  |  | x |  | | Triggers Log |  |  | x |  | | SP Log |  | x |  |  | | Utilizadores Log |  |  | x |  | | BD MySQL Destino |  | x |  |  | | Forma Migração |  |  | x |  | | Triggers Migração | - | - | - | - | | SP Migração |  |  | x |  | | Eventos Migração | x |  |  |  | | Utilizadores Migração |  |  | x |  | | PHP Migração |  |  | x |  | |

## Comparação de Implementações (ficheiro versos PHP)

As comparações que se seguem variam entre vários parâmetros que devem ser considerados na altura de escolher a abordagem tendo em conta os objetivos que temos do projeto.

**Ficheiro**

A implementação via ficheiro pode ter muitas vantagens, mas perde muito a nível de segurança porque, visto que são usados ficheiros intermediários, a informação contida nestes ficheiros irá sempre estar exposta algures fora do sistema da db. Ou seja, para realizar a migração, são utilizados ficheiros não pertencentes ao sistema de gestão de bases de dados, guardando algures na máquina hospedeira informação que deveria estar guardada apenas na db e esta exposição é uma fraqueza.

Não vendo esta característica apenas como uma fraqueza/desvantagem, podemos vê-la como uma vantagem, porque temos assim maneira de ter a informação migrada para um ficheiro para usar consoante as nossas necessidades.

Por outro lado, a implementação via ficheiro tem vantagens também interessantes. Uma delas é a capacidade de utilizar vários formatos de ficheiros para fazer esta migração (ex. csv, xml, etc), oferecendo assim um vasto leque de opções e mostrando várias portas a serem abertas consoante as necessidades.

**PHP**

Em relação a implementação por ficheiro, a implementação por PHP é mais segura, visto que nunca chega a armazenar fora da base de dados qualquer tipo de dados, dado que a migração é feita de forma mais direta e com recurso a menos “intermediários” e de forma mais centralizada.

Embora tenha a vantagem de ser mais seguro, este tipo de implementação depois traz consigo a questão de que para a sua utilização adiciona a necessidade de mais uma linguagem e do software adjacente necessário para a sua utilização.

### Eficiência de Migração

Em termos de eficiência, ao observar os quadros e tabelas que pudemos ver acima, e tendo em conta as especificidades da implementação em ficheiro que requer a importação e exportação de dados, vamos fixar atenções em primeiro lugar no “Tempo PHP” (Linha amarela) e no “Tempo Ficheiro Total”(Linha azul clara).  
Numa primeira análise, comprovamos logo que o tempo que a implementação PHP demora a migrar os dados é maior que a implementação com recurso a Ficheiro. PHP, com 1000 dados por tabela, totalizando 5000 dados a serem migrados, foi a única ocasião em que a implementação de ficheiro foi menos eficiente.

No restante dos casos, ou seja, quando os dados passaram a ser de 5000 dados por tabela, totalizando 25 mil dados ou mais, a implementação por ficheiro mostrou-se muito mais eficiente e rápida. Esta eficiência traduziu-se, quase/aproximadamente, no dobro do tempo em relação à implementação PHP.

Numa análise mais concreta vamos focar a atenção em 2 cenários: **Nr Total de Dados=5000** e **Nr Total de Dados=1000000**.

Com **Nr Total de Dados=5000:**

Neste caso, o tempo de migração em PHP=1seg e o tempo de migração por ficheiro=1.39s.

Com **Nr Total de Dados=1000000:**

Neste caso, o tempo de migração em PHP=52s e o tempo de migração por ficheiro=25s.

De forma mais concreta ao que foi dito anteriormente, é clara a diferença de tempos entre os 2 cenários.

Não quer esta análise dizer que a implementação em PHP não é viável, ou que a implementação de ficheiro é muito melhor. Podemos sim dizer que para cenários com menos dados a serem migrados, ou seja, neste caso se periodicamente são migrados 5000 dados, a migração PHP vai ser uma melhor escolha. Caso seja feita uma migração periódica mais extensa, a implementação por ficheiro mostra-se mais eficiente.

Em suma, a nivel de eficiencia, podemos afirmar que PHP é mais indicado para migrações de menos dados, mas que caso seja necessário migrar maior número de dados de cada vez, a implementação por meio de ficheiro é mais eficiente.

### Robustez

Para testar a robustez e poder comparar ambas as implementações em cenários de falhas idealizamos algumas situações mais comuns, não esquecendo obviamente que é necessário acautelar tudo, porque num cenário empresarial as bases de dados são uma parte fulcral.

Perda de ficheiro:

No caso do php vamos supor um cenário em que perdemos o ficheiro php que é responsável pela canalização de dados desde a db origem para a destino. Caso este ficheiro seja perdido, a migração fica totalmente debilitada, não há contacto nenhum entre as dbs e não vai ser possível migrar o que quer que seja sem aquele ficheiro perdido.

No cenário de perder-se um dos ficheiros intervenientes na migração por recurso a ficheiro, ou seja, o ficheiro de importação ou de exportar ou .csv, também o sistema vai ficar com várias dificuldades porque não terá nenhum local para onde canalizar os dados que quer migrar ou como o fazer. Aliás, caso o ficheiro não seja encontrado irá falhar e notificar que não consegue localizar o ficheiro em falta para concluir o processo.

Erro numa das dbs:

Já no caso de não ser a ausência de ficheiros, mas sim por qualquer razão uma das dbs tenha algum problema e deixe de funcionar já vão ser levantadas outras questões bastante importantes.

Vamos abordar em primeiro lugar o cenário em que uma das dbs da implementação por ficheiro falha. No caso em que é a db origem que falha, há falta de dados para serem migrados. Sendo este cenário grave, mais grave ainda seria a falha na db destino. Ou seja, a implementação ficheiro, com a coluna “exportado” que coloca a 1 quando o dado foi exportado e a 0 quando ainda está para ser exportado, pode ter uma desvantagem grave a nível de robustez.

Imaginemos o cenário que a db origem inicia o processo de migração dos dados, carrega-os para o ficheiro “.csv” exportar e coloca a 1 a coluna daquele dado. Ora, neste momento a db destino falha e não recebe os dados no ficheiro csv. Isto origina um grave problema porque a parcela de dados que estava para ser migrada foi dada como exportada com sucesso mas nunca chegou efetivamente a ser, originando perda de dados.

Em segundo lugar, na implementação em php, esta questão de perda de dados dados como entregues não acontece, visto que como os dados são canalizados de uma db para a outra de forma mais direta. Porém, caso uma delas falhe há um erro grave, e não há migração. É importante reforçar que é necessário preparar a db para que mais tarde os dados sejam migrados corretamente quando o sistema estiver de novo a 100%. A implementação php, neste cenário parece mais robusta, porque embora também grave, não vai assumir dados exportados sem que tenham sido mesmo exportados.

### Flexibilidade / Dependência

Em termos de flexibilidade, para fazer alterações, por exemplo, na periodicidade com que a db é migrada, a dificuldade vai ser exatamente igual, pois o método utilizado é o mesmo para ambas as implementações: ambas correm um ficheiro que coordena os eventos programáveis no Windows.

No caso de querer alterar o diretório onde são colocados os ficheiros csv também é necessário fazer uma alteração manual na implementação, e escolher um novo path.

Agora, em termos de dependência, há também problemas necessários de acautelar caso uma das dbs falhe, principalmente caso a db origem colapse. Ou seja, caso a db origem esteja em baixo, para além da aplicação ser altamente dependente da db e não funcionar corretamente sem esta, caso o sistema tente fazer a migração, não irá encontrar nada para migrar e notificará um erro.

Outra questão levanta-se em relação à dependência que a migração tem com php, visto que caso haja algum problema com o código php ou com o próprio software, a migração vai estar em causa. A migração via php é altamente dependente do correto funcionamento do software que sustenta esta funcionalidade. Por exemplo, caso se mude de máquina e a nova máquina não tenha php, nao vai ser possível fazer a migração.

A implementação via ficheiro também é dependente dos diretórios escolhidos, caso a aplicação não seja flexível a criar os ficheiros “intermediários”, ao mudar de máquina hospedeira sem os ficheiros, a migração também irá dar erro.

Em conclusão, ambas as implementações são dependentes, ou por causa de drivers ou por causa de diretórios escolhidos.

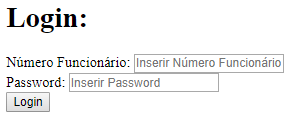
### Segurança

Como já foi dito acima, o método de migração que parece ser mais propenso a falhas na segurança é o método de migração através de ficheiro, visto que o processo gira em torno da criação de ficheiros em que armazena a informação da db algures num ficheiro. Esta informação fica portanto “materializada” algures fora da base de dados, disponível para consulta, etc.

Já o PHP é mais seguro porque a informação nunca chega a ser materializada fora da db, visto que quando a informação é migrada, esta informação é canalizada diretamente para a db destino.

## Auditoria de Dados (base de dados origem)

* Index



<?php

session\_start();

$\_SESSION = array();

session\_destroy();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<form action="ListaTabelas.php" method="POST">

<h1>Login:</h1>

<label for="username">Número Funcionário:</label>

<input type="text" placeholder="Inserir Número Funcionário" name="username" required>

<br/>

<label for="password">Password:</label>

<input type="password" placeholder="Inserir Password" name="password">

<br/>

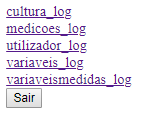
<button type="submit">Login</button>

</form>

</body>

</html>

* Lista Tabelas



<?php

/\*\*

\* Created by PhpStorm.

\* User: hugo

\* Date: 3/25/2019

\* Time: 12:02 AM

\*/

session\_start();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<?php

$servername = "127.0.0.1";

if(!isset($\_SESSION['username']))

$\_SESSION['username'] = $\_POST['username'];

if(!isset($\_SESSION['password']))

$\_SESSION['password'] = $\_POST['password'];

$bd = "dbphp"; //Mudar se for preciso

// Create connection

$conn = new mysqli($servername, $\_SESSION['username'], $\_SESSION['password'], $bd);

// Check connection

if ($conn->connect\_error) {

echo '<h1>Erro no Login</h1>';

die();

}else

{

echo '<a href="VerTabela.php?tabela=cultura\_log">cultura\_log</a><br/>';

echo '<a href="VerTabela.php?tabela=medicoes\_log">medicoes\_log</a><br/>';

echo '<a href="VerTabela.php?tabela=utilizador\_log">utilizador\_log</a><br/>';

echo '<a href="VerTabela.php?tabela=variaveis\_log">variaveis\_log</a><br/>';

echo '<a href="VerTabela.php?tabela=variaveismedidas\_log">variaveismedidas\_log</a><br/>';

}

?>

<form action="Index.php">

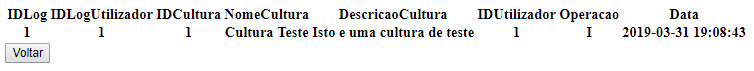
<input type="submit" value="Sair" />

</form>

</body>

</html>

* Ver Tabela



<?php

session\_start();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<?php

/\*\*

\* Created by PhpStorm.

\* User: hugo

\* Date: 3/25/2019

\* Time: 12:03 AM

\*/

$servername = "127.0.0.1";

$bd = "dbphp\_destino"; //Mudar se for preciso

// Create connection

$conn = new mysqli($servername, $\_SESSION['username'], $\_SESSION['password'], $bd);

// Check connection

if ($conn->connect\_error) {

echo '<h1>Erro a carregar a tabela</h1>';

die();

}else

{

$table\_data = get\_table\_data($\_GET['tabela'], $conn);

if(!isset($table\_data) || empty($table\_data)){

echo "Tabela não tem dados!";

}else {

echo "<table>";

echo "<head>";

//lista dados da tabela

foreach ($table\_data[0] as $name => $value)

echo "<th>" . $name . "</th>";

echo "</head>";

echo "<body>";

foreach ($table\_data as $data) {

echo "<tr>";

foreach ($data as $value)

echo "<th>" . $value . "</th>";

echo "<tr>";

}

echo "</body>";

echo "</table>";

}

}

function get\_table\_data($table\_name, $conn){

//Criacao de um array para obter as entradaas nas tabelas log da DB origem

$array = array();

$sql="SELECT \* from $table\_name";

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

//Povoar o array\_origem com dados da BD origem

if ($result->num\_rows > 0)

// output de cada row

$array = mysqli\_fetch\_all($result, MYSQLI\_ASSOC);

return $array;

}

?>

<form action="ListaTabelas.php">

<input type="submit" value="Voltar" />

</form>

</body>

</html>