Sistemas de Informação Distribuídos

Licenciaturas em Engenharia Informática e Informática e Gestão de Empresas

2019-2020, Segundo Semestre

Detecção de Intrusão e Incêndio em Museus

Auditoria e Migração

Identificação do grupo autor da especificação (Etapa A): **Grupo 20**

|  |  |  |
| --- | --- | --- |
| Número | Nome | Foto |
| 20687 | Pedro Santiago | C:\Users\pmsan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D03016C7.tmp |
| 82286 | Bin Guan | C:\Users\pmsan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\948FF36D.tmp |
| 79142 | Tomás Ferreira | C:\Users\pmsan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D9744503.tmp |
| 82652 | Alexandre Ferreira | C:\Users\pmsan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\952698FF.tmp |
| 82453 | José Pedro Fernandes | C:\Users\pmsan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B4500909.tmp |
| 82608 | Francisco Barros |  |
| Especificação: PHP Ficheiro  X | | |

Identificação do grupo autor da implementação (Etapas B e C): **Grupo 23**

|  |  |  |
| --- | --- | --- |
| Número | Nome | Foto |
| 82493 | Miguel Diaz Gonçalves |  |
| 83380 | Gonçalo Dias do Amaral |  |
| 82361 | André Freitas |  |
| 82946 | Pedro Jones |  |
| 74278 | Dmytro Astashov |  |
| 73788 | Vitor Manuel Figueira Canhão |  |
| Especificação: PHP Ficheiro  Implementação: PHP Ficheiro  X  X | | |

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 actualizado;
* Na folha de rosto (anterior) têm de constar toda a informação solicitada, nomeadamente todas as fotografias de todos os elementos dos dois grupos;
* 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 (Etapa B) 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 (*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 6](#_Toc36983108)

[1.1 Esquema relacional da base de Dados Mysql Origem 6](#_Toc36983109)

[1.2 Apreciação Crítica e esquema relacional implementado 9](#_Toc36983110)

[1.3 Esquema relacional da base de Dados Mysql Destino 10](#_Toc36983111)

[1.4 Apreciação Crítica e esquema relacional implementado 11](#_Toc36983112)

[1.5 Migração entre Bases de Dados 12](#_Toc36983113)

[1.5.1 Forma de Migração 12](#_Toc36983114)

[1.5.2 Apreciação Crítica à especificação da forma de migração 13](#_Toc36983115)

[1.6 Utilizadores Base de Dados de Origem 14](#_Toc36983116)

[1.7 Apreciação Crítica a Gestão de Utilizadores Base de Dados de Origem 16](#_Toc36983117)

[1.8 Utilizadores Base de Dados de Destino 17](#_Toc36983118)

[1.9 Apreciação Crítica a Gestão de Utilizadores Base de Dados de Destino 18](#_Toc36983119)

[1.10 Triggers de suporte à criação de logs e migração 19](#_Toc36983120)

[1.10.1 Apreciação Crítica de triggers para gestão de logs e migração 20](#_Toc36983121)

[1.10.1.1 Triggers Implementados para gestão de logs e migração 21](#_Toc36983122)

[1.11 Stored Procedures de suporte à criação de logs e migração 24](#_Toc36983123)

[1.11.1 Apreciação Crítica de Stored Procedures 25](#_Toc36983124)

[1.11.2 Stored Procedures Implementados 27](#_Toc36983125)

[1.12 Eventos de suporte à migração de dados 31](#_Toc36983126)

[1.12.1 Apreciação Crítica de Eventos 32](#_Toc36983127)

[1.12.2 Eventos Implementados 33](#_Toc36983128)

[1.13 PHP suporte à migração de dados (se relevante) 35](#_Toc36983129)

[1.13.1 Apreciação Crítica ao PHP especificado 36](#_Toc36983130)

[1.13.2 PHP Implementado 37](#_Toc36983131)

[1.14 Avaliação Global de especificações da Etapa A 38](#_Toc36983132)

[2 Etapa C (Especificação e Implementação do Próprio Grupo) 40](#_Toc36983133)

[2.1 Especificação do Próprio Grupo (versão compactada) 40](#_Toc36983134)

[2.1.1 Especificação do Esquema relacional da base de Dados Origem 40](#_Toc36983135)

[2.1.2 Especificação do Esquema relacional da base de Dados Destino 42](#_Toc36983136)

[2.1.3 Forma de Migração Especificada 42](#_Toc36983137)

[2.1.4 Especificação de Utilizadores 42](#_Toc36983138)

[2.1.5 Triggers de suporte à gestão de logs e migração 42](#_Toc36983139)

[2.1.6 Stored Procedures de suporte à gestão de logs e migração 43](#_Toc36983140)

[2.1.7 Eventos de suporte à migração de dados especificados 43](#_Toc36983141)

[2.1.8 PHP de suporte à migração de dados especificado 43](#_Toc36983142)

[2.2 Avaliação Global da Qualidade das Especificações do próprio grupo 44](#_Toc36983143)

[2.3 Implementação do Próprio Grupo 45](#_Toc36983144)

[2.3.1 Utilizadores implementados Base de Dados Origem 45](#_Toc36983145)

[1.1.1 Utilizadores implementados Base de Dados Destino 46](#_Toc36983146)

[2.3.2 Lista de Triggers 46](#_Toc36983147)

[2.3.3 Triggers Implementados 48](#_Toc36983148)

[2.3.4 Lista de Stored Procedures 56](#_Toc36983149)

[2.3.5 Stored Procedures Implementados 57](#_Toc36983150)

[2.3.6 Lista Eventos 60](#_Toc36983151)

[2.3.7 Eventos Implementados 61](#_Toc36983152)

[2.3.8 PHP Implementado 62](#_Toc36983153)

[3 Comparação de Implementações (ficheiro versos PHP) 64](#_Toc36983154)

[3.1 Eficiência de Migração 64](#_Toc36983155)

[3.2 Robustez 65](#_Toc36983156)

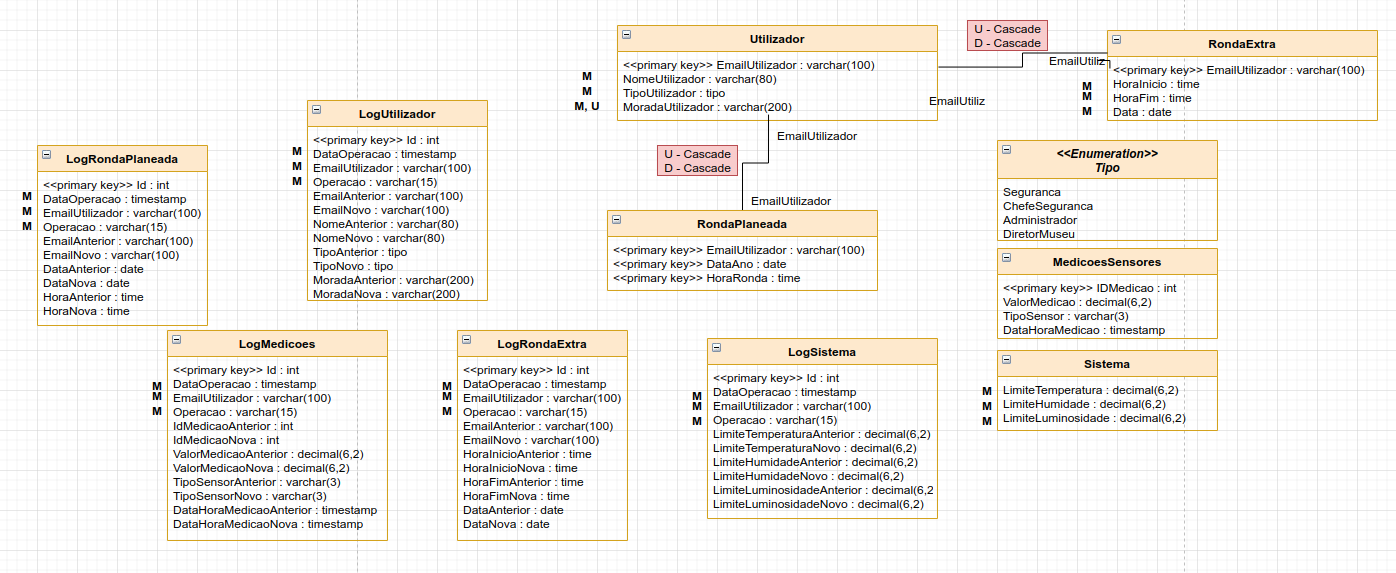
[3.3 Flexibilidade / Dependência 65](#_Toc36983157)

[3.4 Segurança 66](#_Toc36983158)

Detecção de Intrusão e Incêndio em Museus

# Etapa A e B

## Esquema relacional da base de Dados Mysql Origem



Retirámos o DiaSemana e inserimos os campos DataAno e HoraRonda diretamente em RondaPlaneada. A nossa base para esta decisão é o facto de DiaSemana ser algo demasiado vago (Há muitas segundas-feiras num ano), podendo ser compensado com um dia especifico do ano - date - que permite uma melhor localização temporal em caso de qualquer tipo de problema.

Exemplo: Se uma peça do museu tiver desaparecido e o último inventário onde essa peça apareceu tenha sido há 1 mês, podemos facilmente procurar as rondas dos dias imediamente aseguir ao dia que foi feito o inventário.

Decidimos pelo Cascade no que diz respeito a todas as operações de update, pois queremos que as alterações feitas numa tabela sejam devidamente actualizadas nas tabelas ligadas à mesma.

Relativamente à remoção de um Utilizador, esta operação deverá ter um comportamento Cascade, ou seja, a eliminação de um utilizador deverá ser seguida da eliminação das Rondas Planeadas e das Rondas Extras criadas para este utilizador. Os valores próprios para cada Ronda Planeada ou Ronda Extra terão sido devidamente registados nos respectivos Logs.

Em termos das tabelas de logs, estas deverão ser totalmente independentes pois queremos que todas as operações fiquem imaculadamente guardadas. Consideramos que a DataOperacao, EmailUtilizador e Operacao sejam Mandatory em todas as tabelas de Logs pois estes são dados absolutamente necessários para a posterior auditoria. Decidimos pôr os valores antes e depois da operação efectuada apenas para facilitar a auditoria.

Quanto ao utilizador, estará sempre na tabela Utilizador e ao mesmo tempo no sistema da BD - deve ser feito através de um SP para criar o utilizador e associar um user com uma senha.

Para termos uma base de dados mais segura, a password associada ao utilizador da BD será encriptada pelo sistema da BD e mantida internamente, pelo que nunca iremos ter nenhum campo ou tabela com as passwords de utilizadores. Assim, isolamos este tipo de informação sensível dos utilizadores da BD.

- Discussão sobre Tabela dos Utilizadores: saber se se deveria usar uma Primary Key como número de Empregado, para poder relacionar com outras tabelas e cada utilizador poder efectuar login com essa referência, e para poder saber se ainda era um empregado da Instituição (com um boolean para a propriedade “Empregado”). Na discussão de grupo, não foram mencionadas objecções para o atributo “Email” como Primary Key, pelo que foi mantido como tal.

- Discussão sobre Logs: principal divergência pelo grupo, por haver duas correntes distintas para a criação dos Logs do sistema: a primeira a ser vista como Logs por Grupos de Utilizadores, onde cada tabela gerada pelos triggers accionados seria actualizada em função do Tipo de Utilizador (foi convencionado que seriam 4 tipos de utilizador + 1 (auditor de dados)); a outra opinião era no modelo de haver Logs para cada uma das Tabelas dado o modelo relacional, assim como um Log que englobasse todos os triggers activados, tendo assim uma dupla certificação dos acontecimentos nos dados.

## Apreciação Crítica e esquema relacional implementado

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Razoável  Breve Justificação:  Não permite fazer um planeamento das rondas que devem ser efectuadas todas as semanas. O planeamento tem que ser feito para cada dia do ano.  A implementação não prevê a necessidade de evitar derivados de erros de leitura dos sensores. Também não prevê a detecção antecipada de problemas eminentes.  A utilização do enumerado para o tipo de utilizador não permite utilizar nomes mais complexos para os grupos (Chefe de segurança, Director do Museu, etc.).  Utilizando o email como primary key na tabela de utilizadores possibilita ataques por SQL Injection.  Foram feitas alterações? (Sim/Não): Não  **Novo Esquema (assinale e justifique as alterações)**  <Apenas preencher caso tenham procedido a alterações> |

## Esquema relacional da base de Dados Mysql Destino

A BD de auditoria (destino) apenas terá tabelas independentes com a informação de que operações foram efectuadas na BD origem. Como tal, não haverá regras de integridade relacional (Cascade ou Restrict). Cada tabela terá os valores antes e depois da operação para facilitar o trabalho do auditor e terá como primary key um ID que será incrementado a cada operação nova.

Consideramos que a DataOperacao, EmailUtilizador e Operacao sejam Mandatory em todas as tabelas de Logs pois estes são dados absolutamente necessários para a posterior auditoria.

## Apreciação Crítica e esquema relacional implementado

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Boa  Breve Justificação: A especificação da base de dados cumpre todos os requisitos do enunciado  Foram feitas alterações? (Sim/Não): Sim  **Novo Esquema (assinale e justifique as alterações)**  Não implementamos o autoincrement na primary key ID das tabelas de log uma vez que como os registos são copiados integralmente das tabelas de origem este já foi criado. |

## Migração entre Bases de Dados

### Forma de Migração

- Discussão sobre a forma de fazer a migração entre bases de dados: no tipo de ficheiro que deverá ser criado, pensamos que o melhor tipo de ficheiro será o CSV, pois será mais fácil de exportar a informação necessária para a migração dos dados. Para manter a coerência de dados durante a fase de migração, pensamos que deverá ser seguida uma sequência de eventos, desde a criação de um ficheiro, exportação dos dados da base de dados de origem, seguida da importação de dados na base de dados de destino, e, por fim, a eliminação do ficheiro. Caso este ficheiro não tenha sido criado, na fase de importação de dados será detectada a ausência do ficheiro e deverá voltar ao início do processo, mantendo a integridade dos dados e segurança na migração dos dados.

Informação a exportar:

- Se BD destino vazia: Cópia dos logs, réplica das Tabelas de Logs da BD origem.

- Actualização diária BD destino: importação de informação com base nos novos registos do último dia, utilizando os logs.

Periodicidade:

Todos os dias, à noite.

Quem toma iniciativa:

BD origem toma a iniciativa de exportar a informação para a BD destino.

### Apreciação Crítica à especificação da forma de migração

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Razoável  **Análise crítica (clareza, completude, rigor):**  A metodologia proposta para a migração não prevê a hipótese de haver um erro e não ser feita a cópia um ou mais dias. Nesse caso apenas é copiada a informação do último dia.  Não é claro também o que quer dizer informação do último dia. Pode ser o último dia existente na base de dados ou o dia actual da cópia.  A Base de dados de destino nunca poderá estar vazia uma vez que a sua implementação de acordo com o modelo proposto no Esquema relacional, faz parte dos requisitos enunciado. As tabelas essas sim podem, no início, estar vazias (sem registos) mas deveram estar criadas. |

## Utilizadores Base de Dados de Origem

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tipo de Utilizador** | | | |
| **Administrador** | **Director de Museu** | **Chefe de Segurança** | **Segurança** |
| **Utilizador** | **RE** | **-** | **-** | **-** |
| **Rondas Planeadas** | **REL** | **L** | **REL** | **L** |
| **Rondas Extra** | **RL** | **L** | **L** | **-** |
| **Sistema** | **REL** | **L** | **-** | **-** |
| **MediçõesSensores** | **L** | **L** | **-** | **-** |
| **Stored Proc.** |  |  |  |  |
| **Criação de Utilizador** | **X** | **-** | **-** | **-** |
| **Remoção de Utilizador** | **X** | **-** | **-** | **-** |
| **Alteração de Password** | **X** | **X** | **X** | **X** |
| **Alteração de Morada** | **X** | **X** | **X** | **X** |
| **Inserção de Ronda Planeada** | **X** | **-** | **X** | **-** |
| **Alteração de Ronda Planeada** | **X** | **-** | **X** | **-** |
| **Eliminação de Ronda Planeada** | **X** | **-** | **X** | **-** |
| **Inserção de Ronda Extra** | **X** | **-** | **X** | **X** |
| **Consultar tabela utilizadores** | **X** | **X** | **X** | **-** |
| **Exportação de Informação para Migração** | **-** | **-** | **-** | **-** |

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

A tabela Utilizador estará limitada ao máximo em termos de permissões, por forma a obrigar a utilização do Store Procedure (ConsultarUtilizadores) - assim, haverão logs sobre quem consultou esta tabela.

Apenas daremos permissões de leitura ao director do museu, todas as outras operações são irrelevantes para este tipo de utilizador, na nossa opinião.

O Administrador tem permissões de escrita, remoção e leitura em todas as tabelas excepto a Utilizadores (tem que utilizar o SP) e só de leitura na tabela de medições dos sensores e nos logs, pois queremos que estes dados sejam o mais correctos possível.

O chefe de segurança poderá monitorizar, alterar ou remover a tabela das rondas planeadas, bem como o poder de consultar e registar as rondas extraordinárias.

O segurança apenas terá permissão de leitura para consultar as rondas planeadas. Para registar uma ronda extra, o segurança terá usar o SP - Inserção de Ronda Extra.

O SP “Exportação de Informação para Migração” só deve ser executado pelo Sistema Operativo.

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

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Razoável  **Análise crítica (clareza, completude, rigor):**  Falta acrescentar a escrita ao “Administrador” na ronda extra como comprova a frase: “O Administrador tem permissões de escrita, remoção e leitura em todas as tabelas excepto a Utilizadores” O Chefe de segurança e o Segurança deviam também ter permissões de monitorizar as medições dos sensores.  **Solução Implementada:**  Implementado tal como foi especificado |

## Utilizadores Base de Dados de Destino

|  |  |
| --- | --- |
|  | **Tipo de Utilizador** |
| **Tabela** | **Auditor** |
| LogUtilizador | L |
| LogRondaExtra | L |
| LogRondaPlaneada | L |
| LogMedicoes | L |
| LogSistema | L |
| **Stored Proc.** |  |
| ConsultaUtilizador | X |
| ConsultaRondaExtra | X |
| ConsultaRondaPlaneada | X |
| ConsultaMedicoes | X |
| ConsultaSistema | X |

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

Para a BD de auditoria, apenas será necessário dar permissões de leitura sobre todas as tabelas, mais do que essa permissão é desnecessário e até consideramos que seria perigoso para a integridade dos logs.

Criámos alguns stored procedure, que não são mais do que comandos select sobre cada uma das tabelas. Apenas como forma de facilitar o trabalho do auditor.

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

|  |
| --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Boa  **Análise crítica (clareza, completude, rigor):**  A gestão de utilizadores faz sentido e cumpre com os requisitos.  **Solução Implementada:**  Implementado tal como foi especificado |

## Triggers de suporte à criação de logs e migração

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nome Trigger** | **Base de Dados** | **Tabela** | **Tipo de Operação (I,U,D)** | **Evento**  **(A, B)** | **Notas (apenas indicar aquilo que não seja óbvio)** |
| Inserção Utilizador | Origem | Utilizador | I | Inserção |  |
| Alteração Utilizador | Origem | Utilizador | U | Alteração |  |
| Remoção Utilizador | Origem | Utilizador | D | Remoção |  |
| Inserção RondaPlaneada | Origem | Ronda Planeada | I | Inserção |  |
| Alteração RondaPlaneada | Origem | Ronda Planeada | U | Alteração |  |
| Remoção RondaPlaneada | Origem | Ronda Planeada | D | Remoção |  |
| Inserção RondaExtra | Origem | Ronda Extra | I | Inserção |  |
| Remoção RondaExtra | Origem | Ronda Extra | D | Remoção | Só existe no caso de um utilizador ser eliminado, vão ser removidas as rondas e registadas essas remoções no log |

Todos os triggers de inserção têm que ter todas as colunas que se referem a campos anteriores a NULL.

Exemplo:

Inserção de um Utilizador na Tabela de Log Utilizador:

(id); “dia”; “Email@administrador.com”; I; NULL; “EmailNovo”; NULL; “NomeNovo”; NULL; “TipoUtilizadorNovo”; NULL; “MoradaNova”.

Todos os triggers de remoção têm que ter todas as colunas que se referem a campos novos a NULL.

Exemplo:

Remoção de um Utilizador na Tabela de Log Utilizador:

(id); “dia”; “Email@administrador.com”; D; “EmailAnterior”; NULL; “NomeAnterior”; NULL; “TipoUtilizadorAnterior”; NULL; “MoradaAnterior”; NULL.

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Razoável  Breve Justificação:  Não foram especificados os triggers para as tabelas, Medicoes e Sistema. Falta também um trigger para o update na tabela de RondaExtra. No caso do trigger Remoção RondaExtra a garantia que em caso de um utilizador ser apagado todas as suas rondas extra também o são é dada pela Foreign Key Option, On Delete Cascade.  **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) | | Inserção Utilizador | X |  |  |  | | Alteração Utilizador | X |  |  |  | | Remoção Utilizador | X |  |  |  | | Inserção RondaPlaneada | X |  |  |  | | Alteração RondaPlaneada | X |  |  |  | | Remoção RondaPlaneada | X |  |  |  | | Inserção RondaExtra | X |  |  |  | | Remoção RondaExtra |  | X |  |  | |

#### Triggers Implementados para gestão de logs e migração

|  |
| --- |
| 1. Nome Trigger: Inserção\_Utilizador  -- Inserção de Utilizador  CREATE TRIGGER ` Inserção\_Utilizador` AFTER INSERT ON `utilizador`  FOR EACH ROW  BEGIN  insert into log.logutilizador  (DataOperacao, EmailUtilizador, Operacao, EmailNovo, NomeNovo, TipoNovo, MoradaNova)  Values  (getdate(), user(), 'I', NEW.EmailUtilizador, NEW.NomeUtilizador, NEW.TipoUtilizador, NEW.MoradaUtilizador);  END  2. Nome Trigger: Alteracao\_Utilizador  -- Update no utilizador  CREATE TRIGGER `Alteracao\_Utilizador` AFTER UPDATE ON `utilizador`  FOR EACH ROW  BEGIN  insert into log.logutilizador  (DataOperacao, EmailUtilizador, Operacao, EmailNovo, EmailAnterior, NomeNovo, NomeAnterior, TipoNovo, TipoAnterior, MoradaNova, MoradaAnterior)  Values  (getdate(), user(), 'U', NEW.EmailUtilizador, OLD.EmailUtilizador, NEW.NomeUtilizador, OLD.NomeUtilizador, NEW.TipoUtilizador, OLD.TipoUtilizador, NEW.MoradaUtilizador, OLD.MoradaUtilizador);  END  3. Nome Trigger: Remocao\_Utilizador  -- Delete no utilizador  CREATE TRIGGER `Remocao\_Utilizador` AFTER DELETE ON `utilizador`  FOR EACH ROW  BEGIN  insert into log.logutilizador  (DataOperacao, EmailUtilizador, Operacao, EmailAnterior, NomeAnterior, TipoAnterior, MoradaAnterior)  Values  (getdate(), user(), 'D',OLD.EmailUtilizador, OLD.NomeUtilizador,OLD.TipoUtilizador,OLD.MoradaUtilizador);  END  4. Nome Trigger: Insercao\_RondaPlaneada  -- Insert no rondaplaneada  CREATE TRIGGER `Insercao\_RondaPlaneada` AFTER INSERT ON `rondaplaneada`  FOR EACH ROW  BEGIN  insert into log.logrondaplaneada  (DataOperacao, EmailUtilizador, Operacao, EmailNovo, DataNova, HoraNova)  Values  (getdate(), user(), 'I', NEW.EmailUtilizador, NEW.DataAno, NEW.HoraRonda);  END  5. Nome Trigger: Alteracao\_RondaPlaneada  -- Update no rondaplaneada  CREATE TRIGGER `Alteracao\_RondaPlaneada` AFTER UPDATE ON `rondaplaneada`  FOR EACH ROW  BEGIN  insert into log.logrondaplaneada  (DataOperacao, EmailUtilizador, Operacao, EmailNovo, EmailAnterior, DataNova, DataAnterior, HoraNova, HoraAnterior)  Values  (getdate(), user(), 'U', NEW.EmailUtilizador, OLD.EmailUtilizador, NEW.DataAno, OLD.DataAno, NEW.HoraRonda, OLD.HoraRonda);  END  6. Nome Trigger: Remoção\_RondaPlaneada  -- Delete no rondaplaneada  CREATE TRIGGER `Remoção\_RondaPlaneada` AFTER DELETE ON `rondaplaneada`  FOR EACH ROW  BEGIN  insert into log.logrondaplaneada  (DataOperacao, EmailUtilizador, Operacao, EmailAnterior, DataAnterior, HoraAnterior)  Values  (getdate(), user(), 'D',OLD.EmailUtilizador, OLD.DataAno, OLD.HoraRonda);  END  7. Nome Trigger: Insercao\_RondaExtra  -- Insert no rondaextra  CREATE TRIGGER `Insercao\_RondaExtra` AFTER INSERT ON `rondaextra`  FOR EACH ROW  BEGIN  insert into log.logrondaextra  (DataOperacao, EmailUtilizador, Operacao, EmailNovo, HoraInicioNova, HoraFimNova, DataNova)  Values  (getdate(), user(), 'I', NEW.EmailUtilizador, NEW.HoraInicio, NEW.HoraFim, NEW.Data);  END  8. Nome Trigger: Remocao\_RondaExtra  -- Delete no rondaextra  CREATE TRIGGER `Remocao\_RondaExtra` AFTER DELETE ON `rondaextra`  FOR EACH ROW  BEGIN  insert into log.logrondaextra  (DataOperacao, EmailUtilizador, Operacao, EmailAnterior, HoraInicioAnterior, HoraFimAnterior, DataAnterior)  Values  (getdate(), user(), 'D', OLD.EmailUtilizador, OLD.HoraInicio, OLD.HoraFim, OLD.Data);  END |

## Stored Procedures de suporte à criação de logs e migração

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Base de Dados | Nome Procedimento | Parâmetros Entrada | Parâmetros Saída | Muito breve descrição |
| Origem | **Criação de Utilizador** | Email, Nome, Tipo, Morada, Senha |  | Insere utilizador na tabela e no sistema da BD |
| Origem | **Remoção de Utilizador** | Email |  | Elimina utilizador do sistema BD e tabela Utilizador |
| Origem | **Alteração de Password** | NovaPassword |  | Altera a password na BD |
| Origem | **Alteração de Morada** | NovaMorada |  | Altera a morada na tabela Utilizador |
| Origem | **Inserção de Ronda Planeada** | Email(de quem fará a ronda), data, hora |  | Insere uma ronda na tabela RondaPlaneada |
| Origem | **Alteração de Ronda Planeada** | Email (de quem fará a ronda), data, hora |  | Altera a ronda na tabela RondaPlaneada |
| Origem | **Eliminação de Ronda Planeada** | Email, Data, Hora |  | Remove uma ronda previamente planeada |
| Origem | **Inserção de Ronda Extra** | Email, HoraInicio, HoraFim, Data |  | Regista uma ronda extraordinária |
| Origem | **Consultar tabela utilizadores** |  | Tabela Utilizador | Comando select sobre a tabela Utilizador |
| Destino | **Consulta Log Utilizador** |  | Tabela Log Utilizador | Consulta tabela LogUtilizador |
| Destino | **Consulta Log Ronda Extra** |  | Tabela Log Ronda Extra | Consulta tabela LogRondaExtra |
| Destino | **Consulta Log Ronda Planeada** |  | Tabela Log Ronda Planeada | Consulta tabela LogRondaPlaneada |
| Destino | **Consulta Log Medicoes** |  | Tabela Log Medições | Consulta tabela LogMedicoes |
| Destino | **Consulta Log Sistema** |  | Tabela Log Sistema | Consulta tabela LogSistema |
| Origem | **Exportação de Informação para Migração** |  | Dados de Logs | Copia informação dos logs do dia anterior para ficheiro |

### Apreciação Crítica de Stored Procedures

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Razoável  Breve Justificação:  Na nossa opinião foram especificados um número exagerado de Stored Procedures. Não é necessário fazer tudo por Stored Procedures.  Não implementamos o SP *Alteração de Ronda Planeada* uma vez que todos os parâmetros de entrada são as primary keys da tabela RondaPlaneada, não há dados para alterar.  **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) | | Criação de Utilizador | X |  |  |  | | Remoção de Utilizador | X |  |  |  | | Alteração de Password | X |  |  |  | | Alteração de Morada | X |  |  |  | | Inserção de Ronda Planeada | X |  |  |  | | Alteração de Ronda Planeada |  |  | X |  | | Eliminação de Ronda Planeada | X |  |  |  | | Inserção de Ronda Extra | X |  |  |  | | Consultar tabela utilizadores | X |  |  |  | | Consulta Log Utilizador | X |  |  |  | | Consulta Log Ronda Extra | X |  |  |  | | Consulta Log Ronda Planeada | X |  |  |  | | Consulta Log Medicoes | X |  |  |  | | Consulta Log Sistema | X |  |  |  | | Exportação de Informação para Migração | X |  |  |  | |

### Stored Procedures Implementados

|  |
| --- |
| 1. Nome SP: CriarUtilizador  -- Cria Utilizador na tabela utilizadores e no SQL  CREATE PROCEDURE `CriarUtilizador`(iEmailUtilizador varchar(100),iNomeUtilizador varchar(80) ,iTipoUtilizador enum('Seguranca','Chefe Seguranca','Administrador','DiretorMuseu') ,iMoradaUtilizador varchar(200) ,iSenhaUtilizador varchar(30))  BEGIN  SET @criarUtilizadorCMD = concat('CREATE USER ''', iEmailUtilizador, '''@''', 'localhost', ''' IDENTIFIED BY ''', iSenhaUtilizador, ''';');  PREPARE criarUtilizadorStatement FROM @criarUtilizadorCMD;  EXECUTE criarUtilizadorStatement;  DEALLOCATE PREPARE criarUtilizadorStatement;    SET @garantirUtilizadorCMD = concat('GRANT ''', iTipoUtilizador ,''' TO ''', iEmailUtilizador, '''@''', 'localhost', ''';');  PREPARE garantirUtilizadorStatement FROM @garantirUtilizadorCMD;  EXECUTE garantirUtilizadorStatement;  DEALLOCATE PREPARE garantirUtilizadorStatement;    INSERT into utilizador  Values(iEmailUtilizador,iNomeUtilizador,iTipoUtilizador,iMoradaUtilizador);  END  2. Nome SP: RemoverUtilizador  –- Apaga Utilizador tanto na base de dados como o utilizador SQL  CREATE PROCEDURE `RemoverUtilizador`(iEmailUtilizador varchar(100))  BEGIN  SET @apagarUtilizadorCMD = concat('DROP USER ''', iEmailUtilizador, '''@''', 'localhost', ''';');  PREPARE apagarUtilizadorStatement FROM @apagarUtilizadorCMD;  EXECUTE apagarUtilizadorStatement;  DEALLOCATE PREPARE apagarUtilizadorStatement;  DELETE FROM utilizador WHERE EmailUtilizador = iEmailUtilizador;  END  3. Nome SP: AlterarPassword  –- Altera Password do utilizador SQL  CREATE PROCEDURE `AlterarPassword`(iEmailUtilizador varchar(100) ,iSenhaUtilizador varchar(30))  BEGIN  SET @ AlterarPasswordCMD = concat('SET PASSWORD FOR ''', iEmailUtilizador, '''@''', 'localhost', ''' = PASSWORD(''', iSenhaUtilizador, ''');');  PREPARE AlterarPasswordStatement FROM @AlterarPasswordCMD;  EXECUTE AlterarPasswordStatement;  DEALLOCATE PREPARE AlterarPasswordStatement;  END  4. Nome SP: AlterarMorada  -- Altera a morada do utilizador na tabela utilizador  CREATE PROCEDURE `AlterarMorada`(iEmailUtilizador varchar(100), iMoradaUtilizador varchar(200))  BEGIN  UPDATE utilizador SET MoradaUtilizador = iMoradaUtilizador WHERE EmailUtilizador= iEmailUtilizador;  END  5. Nome SP: InsecaoRondaPlaneada  -- Cria Ronda Planeada  CREATE PROCEDURE `InsecaoRondaPlaneada`(iEmailUtilizador varchar(100), iDataAno date, iHoraRonda time)  BEGIN  INSERT into rondaplaneada  Values(iEmailUtilizador, iDataAno, iHoraRonda);  END  6. Nome SP: EliminacaoRondaPlaneada  -- Remove Ronda Planeada  CREATE PROCEDURE ` EliminacaoRondaPlaneada`(iEmailUtilizador varchar(100), iDataAno date, iHoraRonda time)  BEGIN  DELETE FROM rondaplaneada WHERE EmailUtilizador = iEmailUtilizador AND DataAno = iDataAno AND HoraRonda = iHoraRonda;  END  7. Nome SP: InsecaoRondaExtra  -- Cria Ronda Extra  CREATE PROCEDURE `InsecaoRondaExtra`(iEmailUtilizador varchar(100), iHoraInicio time, iHoraFim time, iData date)  BEGIN  INSERT into rondaextra  Values(iEmailUtilizador, iHoraInicio, iHoraFim, iData);  END  8. Nome SP: ConsultaUtilizador  -- Consulta Utilizadores e regista que foi acedido na tabela logutilizadores  CREATE PROCEDURE `ConsultaUtilizador`()  BEGIN  INSERT INTO logutilizador(DataOperacao,EmailUtilizador,Operacao) VALUES(GETDATE(), CURRENT\_USER(),'S')  SELECT \* FROM utilizador  END  9. Nome SP: ConsultarLogUtilizador  -- Consulta tabela logutilizador  CREATE PROCEDURE `ConsultaLogUtilizador`()  BEGIN  SELECT \* FROM logutilizador  END  10. Nome SP: ConsultaLogRondaExtra  -- Consulta tabela logrondaextra  CREATE PROCEDURE `ConsultaLogRondaExtra`()  BEGIN  SELECT \* FROM logrondaextra  END  11. Nome SP: ConsultaLogRondaPlaneada  -- Consulta tabela logrondaplaneada  CREATE PROCEDURE `ConsultaLogRondaPlaneada`()  BEGIN  SELECT \* FROM logrondaplaneada  END  12. Nome SP: ConsultaLogSistema  -- Consulta tabela logsistema  CREATE PROCEDURE `ConsultaLogSistema`()  BEGIN  SELECT \* FROM logsistema  END  13. Nome SP: ConsultaLogMedicoes  -- Consulta tabela logmedicoes  CREATE PROCEDURE `ConsultaLogMedicoes`()  BEGIN  SELECT \* FROM logmedicoes;  END  14. Nome SP: ExportarLogs  -- Exporta logs: um ficheiro para cada tabela de logs. A diretoria pode e deve ser configurada dependendo do ficheiro batch que depois importará tal ficheiro.  CREATE PROCEDURE `ExportarLogs`()  BEGIN  SELECT 'id','DataOperacao','Emailutilizador','Operacao','EmailAnterior','EmailNovo','NomeAnterior','NomeNovo','TipoAnterior','TipoNovo','MoradaAnterior','MoradaNova' UNION  SELECT \* FROM logutilizador  WHERE logutilizador.DataOperacao BETWEEN date\_sub(now(), interval 1 day) AND now()  INTO OUTFILE 'logutilizador.csv'  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n';  SELECT 'id','DataOperacao','Emailutilizador','Operacao','LimiteTemperaturaAnterior','LimiteTemperaturaNovo','LimiteHumidadeAnterior','LimiteHumidadeNovo','LimiteLuminosidadeAnterior','LimiteLuminosidadeNovo' UNION  SELECT \* FROM logsistema  WHERE logsistema.DataOperacao BETWEEN date\_sub(now(), interval 1 day) AND now()  INTO OUTFILE 'logsistema.csv'  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n';  SELECT 'id','DataOperacao','Emailutilizador','Operacao','EmailAnterior','EmailNovo','HoraInicioAnterior','HoraInicioNova','HoraFimAnterior','HoraFimNova','DataAnterior','DataNova' UNION  SELECT \* FROM logrondaextra  WHERE logrondaextra.DataOperacao BETWEEN date\_sub(now(), interval 1 day) AND now()  INTO OUTFILE 'logrondaextra.csv'  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n';  SELECT 'id','DataOperacao','Emailutilizador','Operacao','EmailAnterior','EmailNovo','DataAnterior','DataNova','HoraAnterior','HoraNova' UNION  SELECT \* FROM logrondaplaneada  INTO OUTFILE 'logrondaplaneada.csv'  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n';  SELECT 'id','DataOperacao','Emailutilizador','Operacao','idMedicaoAnterior','idMedicaoNova','ValorMedicaoAnterior','ValorMedicaoNova','TipoSensorAnterior','TipoSensorNovo','DataHoraMedicaoAnterior','DataHoraMedicaoNova' UNION  SELECT \* FROM logmedicoes  WHERE logmedicoes.DataOperacao BETWEEN date\_sub(now(), interval 1 day) AND now()  INTO OUTFILE 'logmedicoes.csv'  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n';  END |

## Eventos de suporte à migração de dados

|  |  |  |
| --- | --- | --- |
| Nome Evento | Local Execução  (Origem ou Destino, ou Sistema Operativo) | Muito breve descrição |
| Criação de Ficheiro de Migração | Sistema Operativo | Imperativamente deverá ser criado diariamente um ficheiro (sugerimos um ficheiro CSV), que conterá a informação do dia a reportar, num directório próprio. Sugerimos a criação de um ficheiro batch que correrá este evento, assim como o evento “Exportação de Informação”. |
| Exportação de Informação | BD Origem | Após a criação do ficheiro, deverão ser exportados todos os registos criados nas Tabelas de Logs do dia anterior até àquela hora.  Sugerimos que cada linha de registo indique de qual tabela retirou a informação.  Aqui também deverá indicar o número de registos que foram inseridos.  Depois de terminar a escrita dos registos, solicita o evento ”Importação de Informação”. |
| Importação de Informação | BD Destino | A Base de Dados de Destino deverá verificar a existência de um ficheiro num determinado caminho, de onde deverá importar os registos para cada uma das tabelas de Logs.  Se o ficheiro não existir, deverá solicitar o evento “Criação de Ficheiro”. Depois de terminar a escrita dos registos nas tabelas, solicita o evento ”Eliminação de Ficheiro”. |
| Eliminação de Ficheiro | Sistema Operativo | Deverá ser executado um programa que irá eliminar o ficheiro criado. Após este passo, poderá verificar se os registos estão iguais nas diferentes bases de dados. |

### Apreciação Crítica de Eventos

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Qualidade (Fraca, Razoável, Boa ou Muito Boa): Razoável  Breve Justificação:  A informação é contraditória, não se entende quem é que é responsável pela iniciação da migração, o sistema operativo, a base de dados de destino ou a base de dados de origem.  Como não encontramos maneira de importar um ficheiro com várias tabelas para várias tabelas no SQL, optamos por atribuir um ficheiro CSV por tabela  **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) | | Criação de Ficheiro de Migração |  |  | X |  | | Exportação de Informação |  | X |  |  | | Importação de Informação |  | X |  |  | | Eliminação de Ficheiro | X |  |  |  | |

### Eventos Implementados

|  |
| --- |
| 1. Nome Evento: Criação do Ficheiro, exportação e delete do mesmo  // Exporta os ficheiros e pede a base de dados do auditor para importar os ficheiros, caso os ficheiros não existirem, exportará outra vez e voltara a tentar importar. (Ficheiro .bat)  @echo off  :start  goto :export  :export  mysql -uroot -p -P 29999 "bdorigem" -e "CALL ExportarLogs();"  goto :fileassurance  :fileassurance  if exist logutilizador.csv (  if exist logrondaextra.csv (  if exist logrondaplaneada.csv (  if exist logsistema.csv (  if exist logmedicoes.csv (  goto :import  )else goto :export  )else goto :export  )else goto :export  )else goto :export  )else goto :export  :import  mysql -uroot -p -P 29999 --local-infile "bddestino" < importaux.sql  goto :delete  :delete  del \*.csv  goto :end  :end  echo Migration Complete  2. Nome Evento: Importação do ficheiro na base de dados do auditor  -- comando SQL para ser corrido na base de dados do auditor para importar. As diretorias dos ficheiros são para serem configuradas.  LOAD DATA INFILE 'C:\\xampp\\mysql\\bin\\logutilizador.csv' INTO  TABLE logutilizador  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n'  IGNORE 1 ROWS;  LOAD DATA INFILE 'C:\\xampp\\mysql\\bin\\logsistema.csv' INTO  TABLE logsistema  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n'  IGNORE 1 ROWS;  LOAD DATA INFILE 'C:\\xampp\\mysql\\bin\\logrondaextra.csv' INTO  TABLE logrondaextra  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n'  IGNORE 1 ROWS;  LOAD DATA INFILE 'C:\\xampp\\mysql\\bin\\logrondaplaneada.csv' INTO  TABLE logrondaplaneada  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n'  IGNORE 1 ROWS;  LOAD DATA INFILE 'C:\\xampp\\mysql\\bin\\logmedicoes.csv' INTO  TABLE logmedicoes  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n'  IGNORE 1 ROWS; |

## 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

A especificação teve um modelo relacional bastante completo e também uma definição de roles e gestão de utilizadores adequado.

Por outro lado, faltam triggers essenciais para os logs funcionarem de acordo com os requisitos; Há um excesso de stored procedures que não são necessários; A tabela de utilizadores utiliza o email do utilizador como primary key, facilitando ataques por injecção; Também não é previsto o facto de poder haver problemas com a migração e esta não ocorrer durante alguns dias. A migração é sempre feita diariamente e apenas dos dados do dia anterior. Finalmente é de referir alguma falta de clareza na especificação dos eventos da migração.

**Avaliação Global da Qualidade das Especificações recebidas**

|  |
| --- |
| Avaliação (A,B,C,D,E) : C  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**

|  |
| --- |
| O processo de migração é confuso e pouco resistente a falhas. |
| O excesso de Stored Procedures especificados. |
| Não é garantida a criação de logs para todas as tabelas. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |  |  | X |  | | Utilizadores Log |  |  | X |  | | BD Destino |  |  | X |  | | Forma Migração | X |  |  |  | | Triggers Migração |  |  |  |  | | SP Migração |  | X |  |  | | Eventos Migração | X |  |  |  | | Utilizadores Migração |  |  | X |  | | PHP Migração |  |  |  |  | |

# Etapa C (Especificação e Implementação do Próprio Grupo)

## Especificação do Próprio Grupo (versão compactada)

### Especificação do Esquema relacional da base de Dados Origem

- Na tabela **Grupo** foi removido o campo **ID** e a chave primária passou a ser o campo **nome**. Tal facilita na atribuição dos utilizadores aos grupos e garante que o grupo é único.

- Na tabela **RondaExtra** os campos **datainicio** e **dataFim** passam a ser DATETIME.

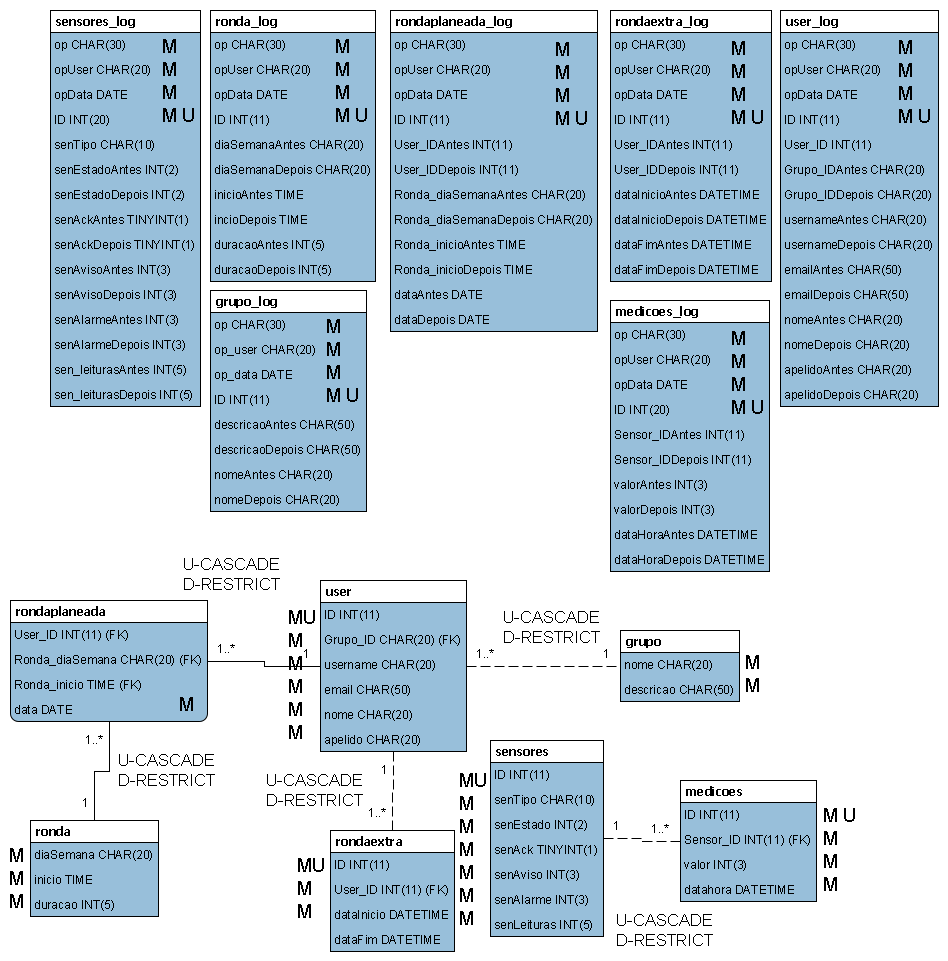
- Em todas as tabelas de Log o campo **opData** passa a ser DATETIME para permitir um registo mais preciso da data da operação.

- Decidimos fazer também registo das operações na tabela Grupo. Apesar de esta tabela não ser actualizada durante a operação normal da aplicação, monitorizamos as alterações não autorizadas.

- Na tabela **User\_logs** acrescentamos o campo **User\_ID** para registar os Selects a uma conta específica de um utilizador.

- Na tabela **Sensores\_Log** o campo **sem\_leitura** foi substituído por **sem\_leituraAntes** e **sem\_leituraDepois** para melhor registar alterações a este campo.

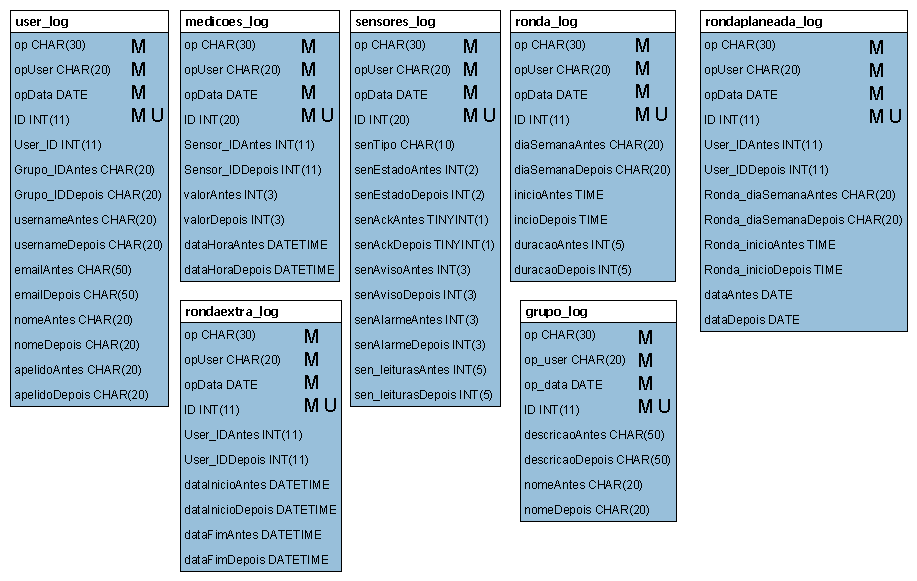
- As tabelas **Medicoes\_Log** e **RondaPlaneada\_Log** foram alteradas para possibilitar a monitorização de todos os campos das tabelas **Medicoes** e **RondaPlaneada.**



### Especificação do Esquema relacional da base de Dados Destino

- Em todas as tabelas a chave primária ID, deixou de ser autoincrement uma vez que o valor já é gerado na base de dados de origem.

- Todas as outras alterações reflectem o que foi alterado na BD de origem uma vez que estas são idênticas.



### Forma de Migração Especificada

A migração apenas deve copiar entradas das tabelas de log do servidor de origem com ID superior ao maior ID existente nas tabelas de log da base de dados do auditor. A migração deve apenas copiar os logs não os apagando da base de dados de origem. A migração dos logs para a base de dados do auditor deve ser realizada, quer automaticamente, com uma periodicidade configurável pelo administrador da aplicação, quer manualmente a pedido do auditor.

### Especificação de Utilizadores

A tabela **User** contem a informação pessoal dos utilizadores bem como o seu **username** SQL e o grupo a que este pertence. O utilizador é sempre validado com o seu username e password do servidor SQL. A tabela Grupo serve para ligar os grupos da aplicação (Seguranças, Administrador, Chefe de Segurança e Director do Museu) aos Roles de SQL que contem as permissões (Adm, Che, Dir, Seg e Aud). Não há permissões dadas directamente aos utilizadores, apenas a grupos.

Na implementação decidimos criar também o grupo Auditores com permissões de leitura apenas nas tabelas da BD de destino.

### Triggers de suporte à gestão de logs e migração

User\_insert, User \_update e User \_delete para monitorizar as operações na tabela User.

RondaExtra\_insert, RondaExtra\_delete para monitorizar as operações na tabela RondaExtra.

Ronda\_insert, Ronda\_update, Ronda\_delete para monitorizar as operações na tabela Ronda.

RondaPlaneada\_insert, RondaPlaneada\_update, RondaPlaneada\_delete para monitorizar as operações na tabela RondaPlaneada.

Sensores\_insert, Sensores\_update, Sensores\_delete para monitorizar as operações na tabela Sensores.

Na implementação decidimos criar também os triggers;

- **Grupo\_insert**, **Grupo\_update** e **Grupo\_delete** para monitorizar as operações na tabela **Grupo**.

- **RondaExtra\_update** para monitorizar também as alterações à tabela **RondaExtra**.

- **Medicoes\_insert**, **Medicoes\_update** e **Medicoes\_delete** para monitorizar as operações na tabela **Medicoes**.

### Stored Procedures de suporte à gestão de logs e migração

**RondaExtra** – Cria uma ronda extra

**Inserir\_User** – Cria um utilizador

**Apagar\_User** – Apaga um utilizador

**Editar\_User** – Permite alterar dados do utilizador; Nome, Apelido e email

**Mudar\_Password** – Alterar a password

**Aknowledge\_Sensor** – Permite marcar um sensor em alarme como Acknowledged

**Atualizar\_Sensor** – Permite alterar dados do sensor; Thresholds, número de leituras, etc.

Na implementação decidimos criar também os stored procedures;

- **select\_user** para registar todos os selects feitos à tabela de utilizadores e **select\_user\_id** para registar os Selects feitos a utilizadores específicos.

- **create\_user** para criar um utilizador no servidor de MySql e **grant\_user** para lhe atribuir um role e **delete\_user** para apagar um utilizador no servidor de MySql.

Decidimos não implementar o stored procedure **Migrar\_Logs** uma vez que a migração é iniciada pelo sistema operativo.

### Eventos de suporte à migração de dados especificados

- Migração manual e automática.

### PHP de suporte à migração de dados especificado

Programa **migração.php**. Para executar basta chamar; php.exe migração.php

## 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  Justificação:  <fazer um resumo dos principais pontos fracos e fortes.  Depois de ler esta secção o leitor deve ter uma visão sobre que secções estavam mais fracas> |

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

|  |
| --- |
| Não especificação dos SP para registo dos selects à tabela de utilizadores. |
| Não eram efectuados logs de todas as alterações às tabelas. |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |  |  | x |  | | Utilizadores Log | x |  |  |  | | BD Destino |  |  | x |  | | Forma Migração |  |  |  | x | | Triggers Migração |  |  |  | x | | SP Migração |  | x |  |  | | Eventos Migração |  |  | x |  | | Utilizadores Migração |  |  | x |  | | PHP Migração |  |  |  | x | |

## Implementação do Próprio Grupo

### Utilizadores implementados Base de Dados Origem

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Tabela** | **Tipo de Utilizador** | | | | | Administrador | Director do Museu | Chefe segurança | Segurança | | Ronda | - | - | E/L | L | | RondaPlaneada | - | - | E/L | L | | RondaExtra | - | - | L | - | | User | E/L | - | - | - | | Medicoes | - | L | L | - | | Sistema | E/L | - | - | - | | **Stored Proc.** |  |  |  |  | | RondaExtra | - | - | - | X | | Inserir\_User | X | - | - | - | | Create\_user | X | - | - | - | | Grant\_user | X | - | - | - | | Apagar\_User | X | - | - | - | | Delete\_User | X | - | - | - | | Editar\_User | X | X | X | X | | Mudar\_Password | X | X | X | X | | Aknowledge\_Sensor | - | - | X | X | | Atualizar\_Sensor | X | - | - | - | | Select\_user | X | X | X | X | | Select\_user\_id | X | X | X | X | |

### Utilizadores implementados Base de Dados Destino

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Tabela** | **Tipo de Utilizador** | | | | Auditor |  |  | | RondaExtra\_log | L |  |  | | User\_log | L |  |  | | Sistema\_log | L |  |  | | RondaPlaneada\_log | L |  |  | | Medicoes\_log | L |  |  | | Ronda\_log | L |  |  | | Grupo\_log | L |  |  | |

### Lista de Triggers

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Lista de Triggers (para cada trigger assinalar com x em célula correspondente)**     |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | Base de Dados (O/D) | Implementado de Acordo com Especificado | Implementado mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | Grupo\_insert | O |  |  |  | X | | Grupo\_update | O |  |  |  | X | | Grupo\_delete | O |  |  |  | X | | User\_insert | O | X |  |  |  | | User\_update | O | X |  |  |  | | User\_delete | O | X |  |  |  | | RondaExtra\_insert | O | X |  |  |  | | RondaExtra\_delete | O | X |  |  |  | | RondaExtra\_update |  |  |  |  | X | | Ronda\_insert | O | X |  |  |  | | Ronda\_update | O | X |  |  |  | | Ronda\_delete | O | X |  |  |  | | RondaPlaneada\_insert | O | X |  |  |  | | RondaPlaneada\_update | O | X |  |  |  | | RondaPlaneada\_update | O | X |  |  |  | | Sensores\_insert | O | X |  |  |  | | Sensores\_update | O | X |  |  |  | | Sensores\_delete | O | X |  |  |  | | Medicoes\_insert | O |  |  |  | X | | Medicoes\_update | O |  |  |  | X | | Medicoes\_delete | O |  |  |  | X | |

### Triggers Implementados

|  |
| --- |
| 1. Nome Trigger: User\_insert (Base de Dados:Origem )  //Cria registo na tabela User\_log apos criação de um utilizador  CREATE TRIGGER `User\_insert` AFTER INSERT ON `User`  FOR EACH ROW  BEGIN  INSERT INTO User\_log (  op,  opUser,  opData,  Grupo\_IDDepois,  usernameDepois,  emailDepois,  nomeDepois,  apelidoDepois  )  VALUES(  'insert',  current\_user(),  now(),  NEW.Grupo\_ID,  NEW.username,  NEW.email,  NEW.nome,  NEW.apelido  );  END  2. Nome Trigger: User\_update (Base de Dados: Origem)  //Cria registo na tabela User\_log apos alteração de um utilizador  CREATE TRIGGER `User\_update` AFTER UPDATE ON `User`  FOR EACH ROW  BEGIN  INSERT INTO User\_log (  op,  opUser,  opData,  Grupo\_IDAntes,  usernameAntes,  emailAntes,  nomeAntes,  apelidoAntes,  Grupo\_IDDepois,  usernameDepois,  emailDepois,  nomeDepois,  apelidoDepois  )  VALUES(  'update',  current\_user(),  now(),  OLD.Grupo\_ID,  OLD.username,  OLD.email,  OLD.nome,  OLD.apelido,  NEW.Grupo\_ID,  NEW.username,  NEW.email,  NEW.nome,  NEW.apelido  );  END  3. Nome Trigger: User\_delete (Base de Dados: Origem)  //Cria registo na tabela User\_log apos apagar um utilizador  CREATE TRIGGER `User\_delete` AFTER DELETE ON `User`  FOR EACH ROW  BEGIN  INSERT INTO User\_log (  op,  opUser,  opData,  Grupo\_IDAntes,  usernameAntes,  emailAntes,  nomeAntes,  apelidoAntes  )  VALUES(  'delete',  current\_user(),  now(),  OLD.Grupo\_ID,  OLD.username,  OLD.email,  OLD.nome,  OLD.apelido  );  END  4. Nome Trigger: Grupo\_insert (Base de Dados: Origem)  //Cria registo na tabela Grupo\_log apos criação de um Grupo  CREATE TRIGGER `Grupo\_insert` AFTER INSERT ON `Grupo`  FOR EACH ROW  BEGIN  INSERT INTO Grupo\_log (  op,  opUser,  opData,  nomeDepois,  descricaoDepois  )  VALUES(  'insert',  current\_user(),  now(),  NEW.nome,  NEW.descricao  );  END  5. Nome Trigger: Grupo\_update (Base de Dados: Origem)  //Cria registo na tabela Grupo\_log apos alteração de um Grupo  CREATE TRIGGER `Grupo\_update` AFTER UPDATE ON `Grupo`  FOR EACH ROW  BEGIN  INSERT INTO Grupo\_log (  op,  opUser,  opData,  nomeAntes,  descricaoAntes,  nomeDepois,  descricaoDepois  )  VALUES(  'update',  current\_user(),  now(),  OLD.nome,  OLD.descricao,  NEW.nome,  NEW.descricao  );  END  6. Nome Trigger: Grupo\_delete (Base de Dados: Origem)  //Cria registo na tabela Grupo\_log apos apagar um Grupo  CREATE TRIGGER `Grupo\_delete` AFTER DELETE ON `Grupo`  FOR EACH ROW  BEGIN  INSERT INTO Grupo\_log (  op,  opUser,  opData,  nomeAntes,  descricaoAntes  )  VALUES(  'delete',  current\_user(),  now(),  OLD.nome,  OLD.descricao  );  END$$  7. Nome Trigger: RondaExtra\_insert (Base de Dados: Origem)  //Cria registo na tabela RondaExtra\_log apos criação de uma RondaExtra  CREATE TRIGGER `RondaExtra\_insert` AFTER INSERT ON `RondaExtra`  FOR EACH ROW  BEGIN  INSERT INTO RondaExtra\_log (  op,  opUser,  opData,  User\_IDDepois,  dataInicioDepois,  dataFimDepois  )  VALUES(  'insert',  current\_user(),  now(),  NEW.User\_ID,  NEW.dataInicio,  NEW.dataFim  );  END$$  8. Nome Trigger: RondaExtra\_update (Base de Dados: Origem)  //Cria registo na tabela RondaExtra\_log apos alteração de uma RondaExtra  CREATE TRIGGER `RondaExtra\_update` AFTER UPDATE ON `RondaExtra`  FOR EACH ROW  BEGIN  INSERT INTO RondaExtra\_log (  op,  opUser,  opData,  User\_IDAntes,  dataInicioAntes,  dataFimAntes,  User\_IDDepois,  dataInicioDepois,  dataFimDepois  )  VALUES(  'update',  current\_user(),  now(),  OLD.User\_ID,  OLD.dataInicio,  OLD.dataFim,  NEW.User\_ID,  NEW.dataInicio,  NEW.dataFim  );  END  9. Nome Trigger: RondaExtra\_delete (Base de Dados: Origem)  //Cria registo na tabela RondaExtra\_log apos apagar uma RondaExtra  CREATE TRIGGER `RondaExtra\_delete` AFTER DELETE ON `RondaExtra`  FOR EACH ROW  BEGIN  INSERT INTO RondaExtra\_log (  op,  opUser,  opData,  User\_IDAntes,  dataInicioAntes,  dataFimAntes  )  VALUES(  'delete',  current\_user(),  now(),  OLD.User\_ID,  OLD.dataInicio,  OLD.dataFim  );  END    10. Nome Trigger: Ronda\_insert (Base de Dados: Origem)  //Cria registo na tabela Ronda\_log apos criação de uma Ronda  CREATE TRIGGER `Ronda\_insert` AFTER INSERT ON `Ronda`  FOR EACH ROW  BEGIN  INSERT INTO Ronda\_log (  op,  opUser,  opData,  diaSemanaDepois,  inicioDepois,  duracaoDepois  )  VALUES(  'insert',  current\_user(),  now(),  NEW.diaSemana,  NEW.inicio,  NEW.duracao  );  END  11. Nome Trigger: Ronda\_update (Base de Dados: Origem)  //Cria registo na tabela Ronda\_log apos alteração de uma Ronda  CREATE TRIGGER `Ronda\_update` AFTER UPDATE ON `Ronda`  FOR EACH ROW  BEGIN  INSERT INTO Ronda\_log (  op,  opUser,  opData,  diaSemanaAntes,  inicioAntes,  duracaoAntes,  diaSemanaDepois,  inicioDepois,  duracaoDepois  )  VALUES(  'update',  current\_user(),  now(),  OLD.diaSemana,  OLD.inicio,  OLD.duracao,  NEW.diaSemana,  NEW.inicio,  NEW.duracao  );  END  12. Nome Trigger: Ronda\_delete (Base de Dados: Origem)  //Cria registo na tabela Ronda\_log apos apagar ums Ronda  CREATE TRIGGER `Ronda\_delete` AFTER DELETE ON `Ronda`  FOR EACH ROW  BEGIN  INSERT INTO Ronda\_log (  op,  opUser,  opData,  diaSemanaAntes,  inicioAntes,  duracaoAntes  )  VALUES(  'delete',  current\_user(),  now(),  OLD.diaSemana,  OLD.inicio,  OLD.duracao  );  END  13. Nome Trigger: RondaPlaneada\_insert (Base de Dados: Origem)  //Cria registo na tabela RondaPlaneada\_log apos criação de uma RondaPlaneada  CREATE TRIGGER `RondaPlaneada\_insert` AFTER INSERT ON `RondaPlaneada`  FOR EACH ROW  BEGIN  INSERT INTO RondaPlaneada\_log (  op,  opUser,  opData,  User\_IDDepois,  Ronda\_diaSemanaDepois,  Ronda\_inicioDepois,  dataDepois  )  VALUES(  'insert',  current\_user(),  now(),  NEW.User\_ID,  NEW.Ronda\_diaSemana,  NEW.Ronda\_inicio,  NEW.data  );  END  14. Nome Trigger: RondaPlaneada\_update (Base de Dados: Origem)  //Cria registo na tabela RondaPlaneada\_log apos alteração de uma RondaPlaneada  CREATE TRIGGER `RondaPlaneada\_update` AFTER UPDATE ON `RondaPlaneada`  FOR EACH ROW  BEGIN  INSERT INTO RondaPlaneada\_log (  op,  opUser,  opData,  User\_IDAntes,  Ronda\_diaSemanaAntes,  Ronda\_inicioAntes,  dataAntes,  User\_IDDepois,  Ronda\_diaSemanaDepois,  Ronda\_inicioDepois,  dataDepois  )  VALUES(  'update',  current\_user(),  now(),  OLD.User\_ID,  OLD.Ronda\_diaSemana,  OLD.Ronda\_inicio,  OLD.data,  NEW.User\_ID,  NEW.Ronda\_diaSemana,  NEW.Ronda\_inicio,  NEW.data  );  END  15. Nome Trigger: RondaPlaneada\_delete (Base de Dados: Origem)  //Cria registo na tabela RondaPlaneada\_log apos apagar uma RondaPlaneada  CREATE TRIGGER `RondaPlaneada\_delete` AFTER DELETE ON `RondaPlaneada`  FOR EACH ROW  BEGIN  INSERT INTO RondaPlaneada\_log (  op,  opUser,  opData,  User\_IDAntes,  Ronda\_diaSemanaAntes,  Ronda\_inicioAntes,  dataAntes  )  VALUES(  'delete',  current\_user(),  now(),  OLD.User\_ID,  OLD.Ronda\_diaSemana,  OLD.Ronda\_inicio,  OLD.data  );  END    16. Nome Trigger: Sensores\_insert (Base de Dados: Origem)  //Cria registo na tabela Sensores\_log apos criação de um Sensor  CREATE TRIGGER `Sensores\_insert` AFTER INSERT ON `Sensores`  FOR EACH ROW  BEGIN  INSERT INTO Sensores\_log (  op,  opUser,  opData,  senTipoDepois,  senEstadoDepois,  senAckDepois,  senAvisoDepois,  senAlarmeDepois,  sen\_leiturasDepois  )  VALUES(  'insert',  current\_user(),  now(),  NEW.senTipo,  NEW.senEstado,  NEW.senAck,  NEW.senAviso,  NEW.senAlarme,  NEW.senLeituras  );  END  17. Nome Trigger: Sensores\_update (Base de Dados: Origem)  //Cria registo na tabela Sensores\_log apos alteração de um Sensor  CREATE TRIGGER `Sensores\_update` AFTER UPDATE ON `Sensores`  FOR EACH ROW  BEGIN  INSERT INTO Sensores\_log (  op,  opUser,  opData,  senTipoAntes,  senEstadoAntes,  senAckAntes,  senAvisoAntes,  senAlarmeAntes,  sen\_leiturasAntes,  senTipoDepois,  senEstadoDepois,  senAckDepois,  senAvisoDepois,  senAlarmeDepois,  sen\_leiturasDepois  )  VALUES(  'update',  current\_user(),  now(),  OLD.senTipo,  OLD.senEstado,  OLD.senAck,  OLD.senAviso,  OLD.senAlarme,  OLD.senLeituras,  NEW.senTipo,  NEW.senEstado,  NEW.senAck,  NEW.senAviso,  NEW.senAlarme,  NEW.senLeituras  );  END  18. Nome Trigger: Sensores\_delete (Base de Dados: Origem)  //Cria registo na tabela Sensores\_log apos apagar um Sensor  CREATE TRIGGER `Sensores\_delete` AFTER DELETE ON `Sensores`  FOR EACH ROW  BEGIN  INSERT INTO Sensores\_log (  op,  opUser,  opData,  senTipoAntes,  senEstadoAntes,  senAckAntes,  senAvisoAntes,  senAlarmeAntes,  sen\_leiturasAntes  )  VALUES(  'delete',  current\_user(),  now(),  OLD.senTipo,  OLD.senEstado,  OLD.senAck,  OLD.senAviso,  OLD.senAlarme,  OLD.senLeituras  );  END  19. Nome Trigger: Medicoes\_insert (Base de Dados: Origem)  //Cria registo na tabela Medicoes\_log apos criação de uma Medição  CREATE TRIGGER `Medicoes\_insert` AFTER INSERT ON `Medicoes`  FOR EACH ROW  BEGIN  INSERT INTO Medicoes\_log (  op,  opUser,  opData,  Sensor\_IDDepois,  valorDepois,  dataHoraDepois  )  VALUES(  'insert',  current\_user(),  now(),  NEW.Sensor\_ID,  NEW.valor,  NEW.dataHora  );  END  20. Nome Trigger: Medicoes\_update (Base de Dados: Origem)  //Cria registo na tabela Medicoes\_log apos alteração de uma Medição  CREATE TRIGGER `Medicoes\_update` AFTER UPDATE ON `Medicoes`  FOR EACH ROW  BEGIN  INSERT INTO Medicoes\_log (  op,  opUser,  opData,  Sensor\_IDAntes,  valorAntes,  dataHoraAntes,  Sensor\_IDDepois,  valorDepois,  dataHoraDepois  )  VALUES(  'update',  current\_user(),  now(),  OLD.Sensor\_ID,  OLD.valor,  OLD.dataHora,  NEW.Sensor\_ID,  NEW.valor,  NEW.dataHora  );  END  21. Nome Trigger: Medicoes\_delete (Base de Dados: Origem)  //Cria registo na tabela Medicoes\_log apos apagar uma Medição  CREATE TRIGGER `Medicoes\_delete` AFTER DELETE ON `Medicoes`  FOR EACH ROW  BEGIN  INSERT INTO Medicoes\_log (  op,  opUser,  opData,  Sensor\_IDAntes,  valorAntes,  dataHoraAntes  )  VALUES(  'delete',  current\_user(),  now(),  OLD.Sensor\_ID,  OLD.valor,  OLD.dataHora  );  END |

### Lista de Stored Procedures

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Lista de SP (para cada SP assinalar com x em célula correspondente)**     |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | Base de Dados (O/D) | Implementado de Acordo com Especificado | Implementado mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | select\_user | O |  |  |  | X | | select\_user\_id | O |  |  |  | X | | inserir\_user | O | X |  |  |  | | create\_user | O |  |  |  | X | | grant\_user | O |  |  |  | X | | apagar\_user | O | X |  |  |  | | delete\_user | O |  |  |  | X | | editar\_user | O | X |  |  |  | | mudar\_password | O | X |  |  |  | | ackowledge\_sensor | O | X |  |  |  | | actualizar\_sensor | O | X |  |  |  | | ronda\_extra | O | X |  |  |  | |

### Stored Procedures Implementados

|  |
| --- |
| 1. Nome SP: select\_user (Base de Dados: Origem)  //Registar todos os selects feitos à tabela de utilizadores  CREATE PROCEDURE `select\_user`()  BEGIN  INSERT INTO User\_log ( op, opUser, opData ) VALUES( 'select', current\_user(), now() );  SELECT ID,Grupo\_ID,username,email,nome,apelido from User;  END  2. Nome SP: select\_user\_id (Base de Dados: Origem)  //Registar todos os selects feitos à tabela de utilizadores a um ID específico.  CREATE PROCEDURE `select\_user\_id`(IN in\_ID Integer)  BEGIN  INSERT INTO User\_log ( op, opUser, opData, User\_Id ) VALUES( 'select', current\_user(), now(), in\_ID );  SELECT ID,Grupo\_ID,username,email,nome,apelido from User Where ID = in\_ID;  END  3. Nome SP: inserir\_user (Base de Dados: Origem)  //Criar um utilizador  CREATE PROCEDURE `inserir\_user`(  IN in\_Grupo\_ID char(20),  IN in\_username char(20),  IN in\_email char(50),  IN in\_nome char(20),  IN in\_apelido char(20),  IN in\_pwd varchar(20)  )  BEGIN  IF NOT EXISTS (SELECT \* FROM User WHERE username = in\_username) THEN  INSERT INTO User ( Grupo\_ID,username,email,nome,apelido ) VALUES( in\_Grupo\_ID,in\_username,in\_email,in\_nome,in\_apelido );  CALL create\_user(in\_username, in\_pwd);  CALL grant\_user(in\_username, in\_Grupo\_ID);  END IF;  END  4. Nome SP: create\_user (Base de Dados: Origem)  //Cria o utilizador no servidor MySQL  CREATE PROCEDURE `create\_user`(IN username varchar(100), IN pwd varchar(255))  BEGIN  SET @createUserCMD = concat('CREATE USER ''', username, '''@''', 'localhost', ''' IDENTIFIED BY ''', pwd, ''';');  PREPARE createUserStatement FROM @createUserCMD;  EXECUTE createUserStatement;  DEALLOCATE PREPARE createUserStatement;  END  5. Nome SP: grant\_user (Base de Dados: Origem)  //Atribui um Role ao utilizador  CREATE PROCEDURE `grant\_user`(IN username varchar(100), IN grupo char(20))  BEGIN  SET @grantUserCMD = concat('GRANT ''', grupo ,''' TO ''', username, '''@''', 'localhost', ''';');  PREPARE grantUserStatement FROM @grantUserCMD;  EXECUTE grantUserStatement;  DEALLOCATE PREPARE grantUserStatement;  END  6. Nome SP: apagar\_user (Base de Dados: Origem)  //Apaga um utilizador  CREATE PROCEDURE `apagar\_user`(  IN in\_ID Integer  )  BEGIN  IF EXISTS (SELECT \* FROM User WHERE ID = in\_ID) THEN  SELECT username INTO @login FROM User WHERE ID = in\_ID;  DELETE FROM User WHERE ID=in\_ID;  CALL delete\_user(@login);  END IF;  END  7. Nome SP: delete\_user (Base de Dados: Origem)  //Apaga um utilizador do servidor MySQL  CREATE PROCEDURE `delete\_user`(IN username varchar(100))  BEGIN  SET @deleteUserCMD = concat('DROP USER IF EXISTS ''', username, '''@''', 'localhost', ''';');  PREPARE deleteUserStatement FROM @deleteUserCMD;  EXECUTE deleteUserStatement;  DEALLOCATE PREPARE deleteUserStatement;  END  8. Nome SP: editar\_user (Base de Dados: Origem)  //Alterar a informação de perfil do utilizador  CREATE PROCEDURE `editar\_user`(  IN in\_ID Integer,  IN in\_email char(50),  IN in\_nome char(20),  IN in\_apelido char(20)  )  BEGIN  UPDATE User SET email=in\_email,nome=in\_nome,apelido=in\_apelido WHERE ID=in\_ID;  END  9. Nome SP: mudar\_password (Base de Dados: Origem)  //Alterar a password  CREATE PROCEDURE `mudar\_password`(  IN username varchar(100),  IN pwd varchar(255)  )  BEGIN  SET @passwordUserCMD = concat('SET PASSWORD FOR ''', username, '''@''', 'localhost', ''' = PASSWORD(''', pwd, ''');');  PREPARE passwordUserStatement FROM @passwordUserCMD;  EXECUTE passwordUserStatement;  DEALLOCATE PREPARE passwordUserStatement;  END  10. Nome SP: ackowledge\_sensor (Base de Dados: Origem)  // Permite marcar um sensor em alarme como Acknowledged  CREATE PROCEDURE `ackowledge\_sensor`(  IN in\_ID Integer  )  BEGIN  UPDATE Sensores SET senAck=true WHERE ID=in\_ID;  END  11. Nome SP: actualizar\_sensor (Base de Dados: Origem)  // Permite alterar dados do sensor; Thresholds, número de leituras, etc.  CREATE PROCEDURE `actualizar\_sensor`(  IN in\_ID Integer,  IN in\_senAviso Integer,  IN in\_senAlarme Integer,  IN in\_senLeituras Integer    )  BEGIN  UPDATE Sensores SET senAviso=in\_senAviso, senAlarme=in\_senAlarme, senLeituras=in\_senLeituras WHERE ID=in\_ID;  END  12. Nome SP: ronda\_extra (Base de Dados: Origem)  // Cria ou fecha uma ronda extra  CREATE PROCEDURE `ronda\_extra`(  IN in\_ID Integer  )  BEGIN  IF EXISTS (SELECT \* FROM RondaExtra WHERE dataFim IS NULL AND User\_ID = in\_ID) THEN  UPDATE RondaExtra SET dataFim = now() WHERE dataFim IS NULL AND User\_ID = in\_ID;  ELSE  INSERT INTO RondaExtra (User\_ID, dataInicio) VALUES (in\_ID, now());  END IF;  END |

### Lista Eventos

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Lista de Eventos (para cada evento assinalar com x em célula correspondente)**     |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | Base de Dados (O/D) | Implementado de Acordo com Especificado | Implementado mas diferente de Especificado | Não Implementado | Não Especificado (criado de novo) | | Migração automática |  |  |  | X |  | | Migração Manual |  |  |  | X |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |

### Eventos Implementados

|  |
| --- |
| 1. Nome Evento: \_\_\_\_\_ (Base de Dados: )  //*Breve Descrição*  *Código*  2. Nome Evento: \_\_\_\_\_ (Base de Dados: )  //*Breve Descrição*  *Código*  3. Nome Evento: \_\_\_\_\_ (Base de Dados: )  //*Breve Descrição*  *Código* |

### PHP Implementado

|  |
| --- |
| $host\_source = 'localhost';  $host\_target = 'localhost';  $db\_source = 'main';  $db\_target = 'log';  $user = 'system';  $password = 'password';  $GLOBALS['DEBUG']=true;  $conn\_source = db\_connect($host\_source, $user, $password, $db\_source);  $conn\_target = db\_connect($host\_target, $user, $password, $db\_target);  $tb\_list = table\_list($conn\_target);  foreach ($tb\_list as $table) {  if ($GLOBALS['DEBUG']) echo "Copy from $db\_source.$table to $db\_target.$table" . PHP\_EOL;  $max\_id = max\_id($conn\_target, $table);  if ($GLOBALS['DEBUG']) echo "Query : SELECT \* FROM $table WHERE ID > $max\_id" . PHP\_EOL;  $result = mysqli\_query($conn\_source, "SELECT \* FROM $table WHERE ID > $max\_id");  while ($row = mysqli\_fetch\_assoc($result)) {  $sql = "INSERT INTO $table (" . implode(", ", array\_keys($row)) . ") VALUES ('" . implode("', '", array\_values($row)) . "');";  if ($GLOBALS['DEBUG']) echo "Insert : $sql" . PHP\_EOL;  mysqli\_query($conn\_target, str\_replace("''", "null", $sql));  }  }  mysqli\_free\_result($result);  mysqli\_close($conn\_source);  mysqli\_close($conn\_target);  /\*\*  \*  \* Ligação a base de dados.  \*  \*/  function db\_connect($host, $user, $password, $database)  {  $connection = mysqli\_connect($host, $user, $password);  if (!$connection) {  die("Could not connect to $host: " . mysqli\_error($connection));  }  mysqli\_select\_db($connection, $database);  return $connection;  }  /\*\*  \*  \* Max ID.  \*  \* Retorna o ID do ultimo registo copiado  \*  \*/  function max\_id($connection, $table)  {  if (empty\_table($connection, $table)) {  $max\_id = 0;  } else {  $result = mysqli\_query($connection, "SELECT MAX(ID) AS max\_id FROM $table");  $row = mysqli\_fetch\_array($result);  $max\_id = $row['max\_id'];  mysqli\_free\_result($result);  }  return $max\_id;  }  /\*\*  \*  \* Empty table.  \*  \* Verifica se a tabela está vazia  \*  \*/  function empty\_table($connection, $table)  {  $result = mysqli\_query($connection, "SELECT \* FROM $table LIMIT 1");  if (!$result) {  die("Error: " . mysqli\_error($connection));  }  return (mysqli\_num\_rows($result) > 0) ? false : true;  }  /\*\*  \*  \* Table list.  \*  \* Gera uma lista das tabelas de log  \*  \*/  function table\_list($connection)  {  $table = array();  $result = mysqli\_query($connection, "SHOW TABLES");  if (!$result) {  die("Error: " . mysqli\_error($connection));  }  while ($tbl = mysqli\_fetch\_array($result)) {  $table[] = $tbl[0];  if ($GLOBALS['DEBUG']) echo "Table : $tbl[0]" . PHP\_EOL;  }  return ($table);  } |

# Comparação de Implementações (ficheiro versos PHP)

## Eficiência de Migração

Na migração por PHP a cópia é feita toda no mesmo comando pelo que não é possível separar a exportação da importação.

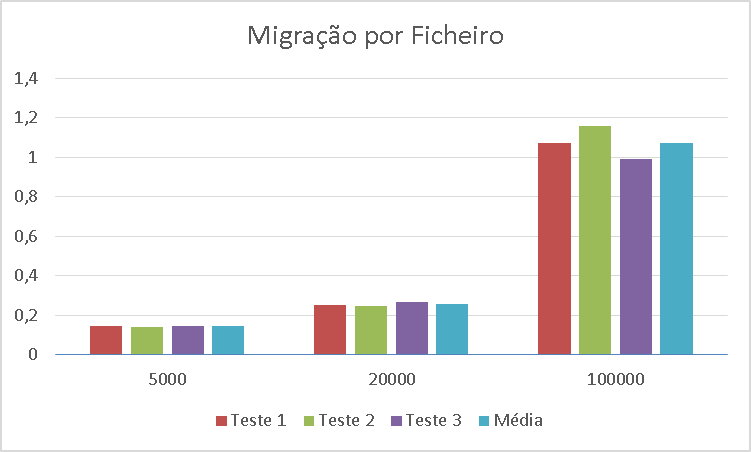
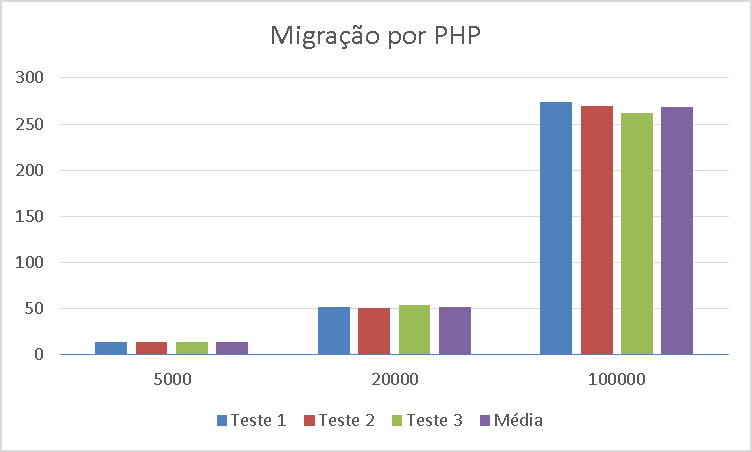
Por esse motivo apenas apresentamos os tempos para o processo completo.

Comparação entes os dois métodos de migração. Tempos em segundos.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 5K Registos | | 20K Registos | | 100K Registos | |
|  | PHP | CSV | PHP | CSV | PHP | CSV |
| Teste 1 | 13,84 | 0,14 | 51,43 | 0,25 | 274,14 | 1,07 |
| Teste 2 | 13,61 | 0,14 | 50,59 | 0,25 | 269,13 | 1,16 |
| Teste 3 | 13,83 | 0,15 | 53,90 | 0,27 | 262,21 | 0,99 |
| Média | 13,76 | 0,14 | 51,97 | 0,26 | 268,50 | 1,07 |

Concluímos que o método de migração por ficheiro é sempre mais rápido do que a migração por PHP. A performance da migração por ficheiro apresenta uma variação muito pequena em relação ao volume de dados a copiar. O mesmo não se passa com a migração por PHP que tem uma eficiência muito baixa com volumes de dados elevados.

Gráficos em segundos.



Finalmente é de referir que todos os testes foram efectuados no mesmo servidor pelo que não está reflectida performance da rede.

## Robustez

A migração por PHP é mais robusta pois caso esta falhe é sempre possível recomeçar e voltar a copiar os dados que ainda não tenham sido copiados. Em caso de corrupção dos dados nas tabelas do auditor é sempre possível apagar tudo e voltar a fazer a migração que copiará tudo de novo.

Por outro lado a migração por ficheiro é muito menos robusta pois apenas copia os dados do último dia. Caso falhe por alguns dias a cópia fica incompleta.

## Flexibilidade / Dependência

Ambos os métodos de migração dependem de nas duas bases de dados as tabelas de logs serem iguais. Caso não sejam a migração não pode ocorrer.

Quanto á flexibilidade, o método de migração por PHP é muito mais flexível pois mesmo que sejam feitas alterações nas tabelas de log ou mesmo acrescentadas novas tabelas o programa apenas necessita que as tabelas do auditor sejam iguais as da base de dados de origem. Já no método de migração por ficheiro as tabelas estão hardcoded o que impossibilita modificações sem adaptar o programa.

A alteração da periodicidade com o método por PHP é simples, basta alterar o agendamento da execução do programa. Como o método por ficheiro tal não é seque possível pois este método apenas copia os dados do dia anterior à sua execução o que obriga a que seja efectuado sempre diariamente.

## Segurança

A migração por PHP nunca expõe os dados sendo por isso mais segura. É no entanto de grande importância proteger o programa PHP pois este tem as credencias de acesso às bases de dados.

A segurança na migração por ficheiro e pior uma vez que grava os dados das tabelas no file system, abrindo mais possibilidades para ataques. Como no PHP é vital proteger o programa pois este contem as credenciais de acesso.