






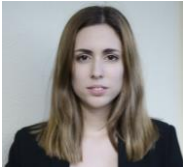
Sistemas de Informação Distribuídos

Licenciaturas em Engenharia Informática e Informática e Gestão de Empresas
2018-2019, Segundo Semestre

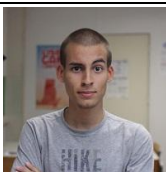



Monitorização de Culturas em Laboratório

Auditoria e Migração

Identificação do grupo autor da
especificação (Etapa A): 18

Número	Nome	Foto
60921	Wang Yang	
72783	André Carvalho	
73353	Bruno Colaço	
74455	Dinis Ferreira	
77667	Maria Dinis	
78014	Mariana Silva	
Especificação: PHP		Ficheiro
<input type="checkbox"/>		<input checked="" type="checkbox"/>

Identificação do grupo autor da
implementação (Etapas B e C): 10

Número	Nome	Foto
77981	André Silva	
73538	Gonçalo Fernandes	
77812	João Aparício	
72809	João Neto	
77561	João Saramago	
77778	Rita Costa	
Especificação: PHP		Ficheiro
<input checked="" type="checkbox"/>		<input type="checkbox"/>
Implementação: PHP		Ficheiro
<input type="checkbox"/>		<input checked="" type="checkbox"/>

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	9
1.1	Esquema relacional da base de Dados Mysql (origem)	9
1.1.1	Apreciação Crítica e esquema relacional implementado.....	11
1.2	Utilizadores Base de Dados de Origem	12
1.2.1	Apreciação Crítica a Gestão de Utilizadores Base de Dados de Origem	13
1.3	Gestão de Logs	14
1.3.1	Triggers de suporte à criação de logs Base de Dados de Origem	14
1.3.1.1	Apreciação Crítica de triggers para gestão de logs	15
1.3.1.2	Triggers Implementados para gestão de logs	17
1.3.2	Stored Procedures de suporte à criação de logs (se relevante)	22
	Apreciação Crítica de Stored Procedures de suporte à criação de logs.....	23
1.3.2.1	Stored Procedures Implementados de suporte à criação de logs	24
1.4	Migração entre Bases de Dados.....	27
1.4.1	Esquema relacional da base de Dados Mysql (destino).....	27
1.4.1.1	Apreciação Crítica e esquema relacional implementado.....	28
1.4.2	Forma de Migração	29
1.4.2.1	Apreciação Crítica à especificação da forma de migração.....	32
1.4.3	Gestão de Utilizadores de Suporte à Migração (origem e/ou destino)	33
1.4.3.1	Apreciação Crítica à especificação da Gestão de Utilizadores.....	34
1.4.4	Triggers de suporte à migração de dados (origem e/ou destino) (se relevante).....	35
1.4.4.1	Apreciação Crítica de triggers de suporte à migração de dados.....	36
1.4.4.2	Triggers Implementados de suporte à migração de dados.....	37
1.4.5	Stored Procedures de suporte à migração de dados	38
1.4.5.1	Apreciação Crítica de Stored Procedures de suporte à migração de dados... ..	40
1.4.5.2	Stored Procedures Implementados de suporte à migração de dados.....	41
	Eventos de suporte à migração de dados	45
1.4.5.3	Apreciação Crítica de Eventos.....	46
1.4.5.4	Eventos Implementados.....	47
1.4.6	PHP suporte à migração de dados (se relevante)	49
1.4.6.1	Apreciação Crítica ao PHP especificado	50
1.4.6.2	PHP Implementado	51

1.5	Avaliação Global de especificações da Etapa A.....	52
	Avaliação Global da Qualidade das Especificações recebidas	54
2	Etapa C (Especificação e Implementação do Próprio Grupo)	55
2.1	Especificação do Esquema relacional da base de Dados Origem	55
2.2	Especificação de Utilizadores	57
2.3	Especificação de Gestão de Logs.....	59
2.3.1	Triggers de suporte à gestão de logs.....	59
2.3.2	Stored Procedures de suporte à gestão de logs.....	62
2.4	Avaliação da especificação do próprio grupo Gestão de Logs	63
2.5	Implementação Gestão de Logs	64
2.5.1	Utilizadores implementados	64
2.5.2	Lista de Triggers.....	66
2.5.3	Triggers Implementados.....	67
2.5.4	Stored Procedures Implementados	75
2.6	Especificação de Migração entre Bases de Dados	81
2.6.1	Esquema relacional da base de Dados Mysql especificada (destino)	81
2.6.2	Forma de Migração Especificada	82
2.6.3	Utilizadores especificados	84
2.6.4	Triggers de suporte à migração de dados especificados.....	85
2.6.5	Stored Procedures de suporte à migração de dados especificados	86
2.6.6	Eventos de suporte à migração de dados especificados.....	87
2.6.7	PHP de suporte à migração de dados especificado	88
2.7	Avaliação das especificações do próprio grupo Migração	89
2.8	Implementação da Migração de Dados	90
2.8.1	Utilizadores Implementado.....	90
2.8.2	Lista Triggers.....	91
2.8.3	Triggers Implementados.....	92
2.8.4	Lista de Stored Procedures.....	93
2.8.5	Stored Procedures Implementados	94
2.8.6	Lista Eventos.....	95
2.8.7	Eventos Implementados.....	96
2.8.8	PHP Implementado	97
2.9	Avaliação Global da Qualidade das Especificações do próprio grupo	107
2.10	Comparação de Implementações (ficheiro versus PHP)	108

2.10.1	Eficiência de Migração	109
2.10.2	Robustez	111
2.10.3	Flexibilidade / Dependência	113
2.10.4	Segurança	115
2.11	Auditoria de Dados (base de dados origem)	116

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 afeta todas as culturas existentes na estufa).

Periodicamente os investigadores dirigem-se à estufa para efetuarem 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, atualizar e consultar os dados de medições das suas culturas. Esta *proteção de dados* é um aspeto 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.

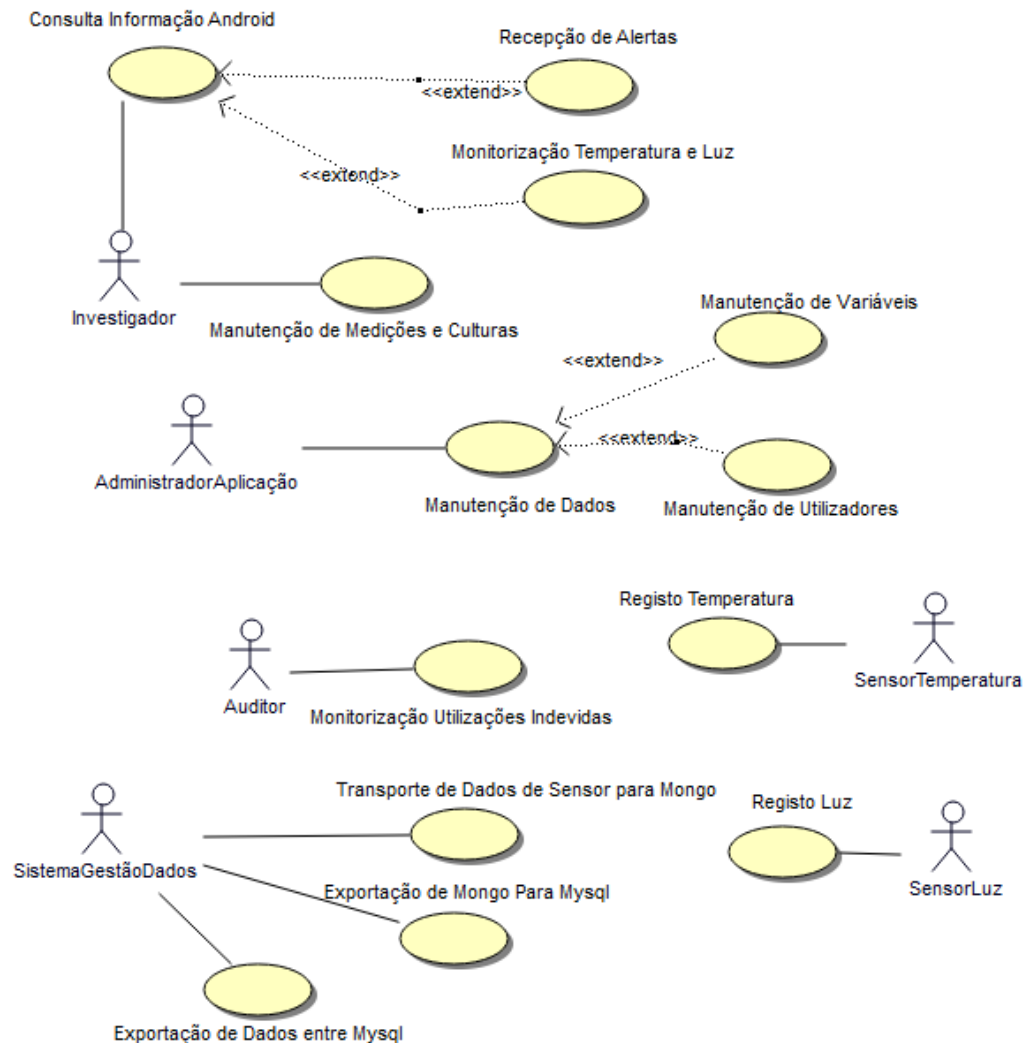
Cada investigador deverá ter a possibilidade de, através de um telemóvel, monitorizar a evolução da temperatura e luminosidade (não apenas a última leitura, mas a evolução na última hora ou horas) e receber os dois tipos de alertas.

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

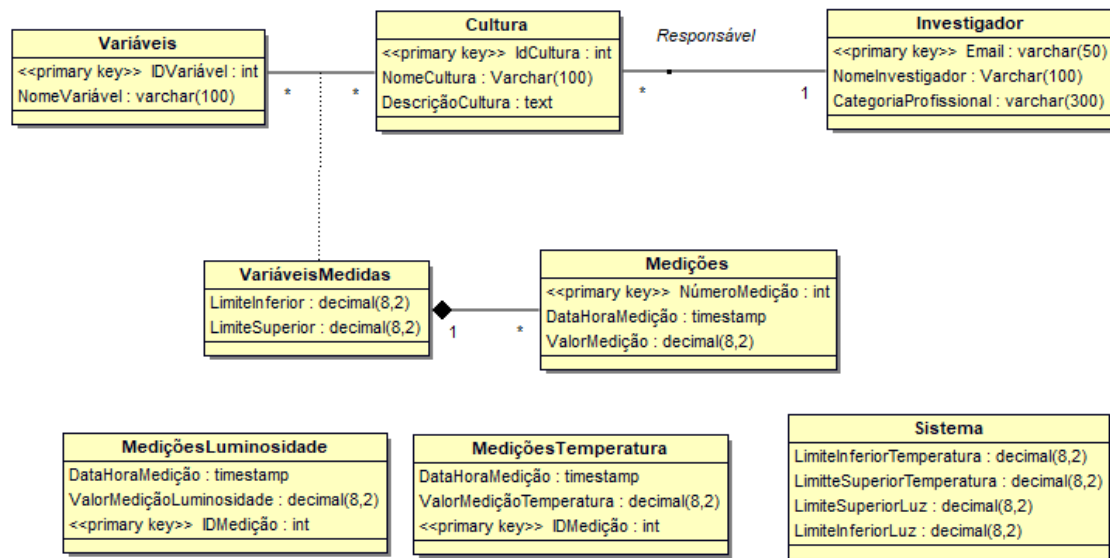
consulta) um auditor pode analisar se ocorreram utilizações abusivas dos dados (por exemplo, quem é que alterou limites de temperatura de uma cultura, etc.).

Diagrama de Use Case Global

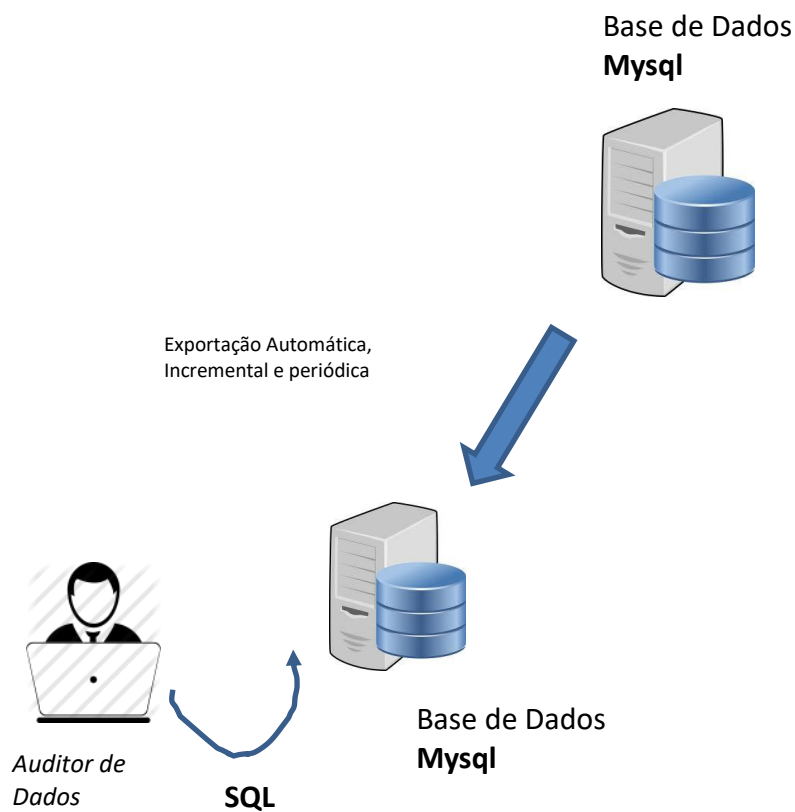


No presente relatório apenas são contemplados os use case “Exportação Dados entre Mysql”, “Monitorização de Utilizações Indevidas” e “Manutenção de Utilizadores” (apenas a componente Mysql/Privilégios/SP/Triggers)). A componente Java (manutenção de culturas, medições, variáveis e utilizadores) não é 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

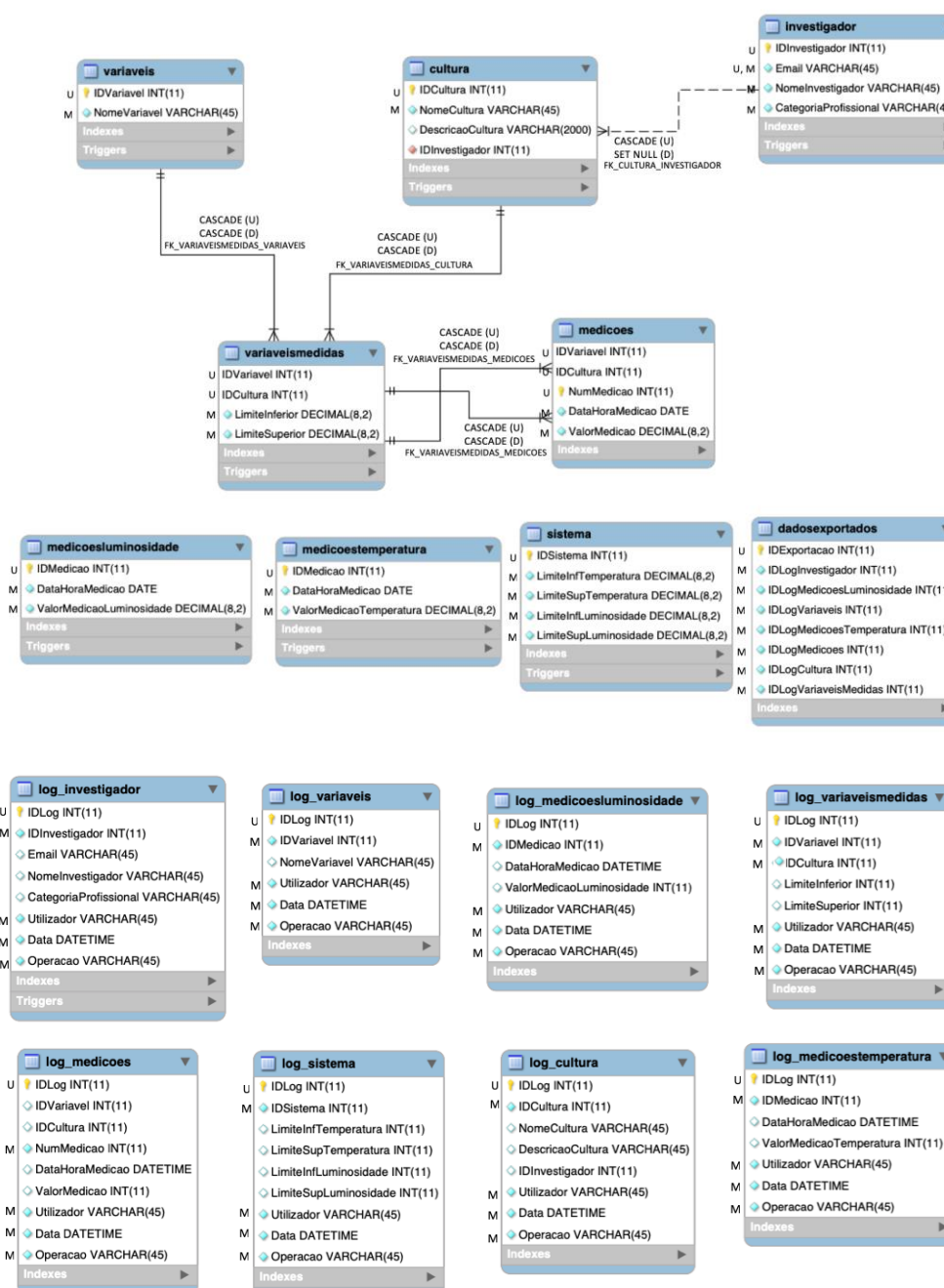


1 Etapa A e B

1.1 Esquema relacional da base de Dados Mysql (origem)

Na tabela relativa ao *Investigador*, não será considerado como chave primária o atributo *Email* e deve ser criado um atributo extra: *IDInvestigador*. A tabela terá, assim, uma chave primária de domínio numérico (por razões de eficiência).

Os atributos *mandatory* (ou seja, de preenchimento obrigatório) encontram-se assinalados com a letra M. Os atributos *unique* (ou seja, únicos) encontram-se assinalados com U.



Não devem ser utilizadas chaves estrangeiras nas tabelas *log* porque devem manter-se guardadas nestas tabelas linhas que foram anteriormente apagadas.

A operação *update* de integridade referencial das chaves estrangeiras deve estar sempre a *cascade*, de forma a permitir alterações, mas propagá-las, também, para as tabelas “filhas”.

A operação de *delete* deve ser *set null* no caso da chave estrangeira *IDInvestigador*, pois caso uma linha da tabela *Investigador* seja eliminada, o campo correspondente ao seu ID na tabela *Cultura* será colocado a NULL. Para as restantes chaves estrangeiras, deve optar-se pela operação *cascade* que, permite apagar a linha respetiva e a apaga, também, nas tabelas “filhas”.

Como forma de controlo de duplicados na importação dos dados das tabelas para as respetivas *log* deve optar-se pela criação de uma tabela extra *dadoexportados*; este tópico será desenvolvido posteriormente.

1.1.1 Apreciação Crítica e esquema relacional implementado

Qualidade (Fracas, Razoável, Boa ou Muito Boa): Boa

Breve Justificação:

Quantidade excessiva de tabelas de logs o que dificulta a pesquisa por data e por utilizador. Caso o auditor pretenda filtrar tudo o que um dado utilizador viu/alterou numa determinada data tem que recorrer a oito comandos select diferentes, dificultando por isso esta tarefa.

Consideramos desnecessário usar o IDUtilizador como primary key da tabela investigador, uma vez que por si só o email já é unique e a questão da eficiência não é relevante, ainda para mais com o baixo volume de dados da tabela em questão.

É desnecessária a existência de um id na tabela sistema, visto que os valores desta tabela devem apenas poder ser alterados e não acrescentados mais dados, pelo que não há necessidade de fazer essa distinção.

Incoerência nos tipos de variáveis. Nas tabelas medições, 'variaveismedidas', 'sistema', 'medicoesluminosidade' e 'medicoestemperatura' existem campos cujo tipo de valores é DECIMAL e esses mesmos campos encontram-se nas respetivas tabelas de logs com o tipo de valores INT. Esta situação tem como consequência o facto dos valores sofrerem um arredondamento às unidades quando chegam às tabelas de logs e, desta forma, perde-se a informação exata que foi registada, modificada ou eliminada da base de dados.

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

Na tabela abaixo são indicados os privilégios que cada utilizador tem sobre as tabelas e os *stored procedures* em que E=Escrita, L=Leitura, X=Executar e - = sem permissões.

Considera-se que os investigadores são responsáveis pela manutenção de culturas e medições e os administradores pela gestão de variáveis e utilizadores. Além disso devem ser dadas permissões de leitura ao Administrador sobre as tabelas *log*.

Não devem ser dadas quaisquer permissões de escrita sobre as tabelas *log* por questões de segurança visto que estas devem conter todas as alterações feitas sobre as restantes tabelas para que possam ser consultadas posteriormente pelo Auditor.

Tabela	Tipo de Utilizador	
	Administrador	Investigador
cultura	-	E, L
medicoes	-	E, L
medicoestemperatura	-	-
medicoesluminosidade	-	-
variaveis	E, L	L
variaveismedidas	E, L	L
investigador	E, L	L
sistema	E, L	L
dadosexportados	L	-
log_cultura	L	-
log_medicoes	L	-
log_medicoestemperatura	L	-
log_medicoesluminosidade	L	-
log_variaveis	L	-
log_variaveismedidas	L	-
log_investigador	L	-
log_sistema	L	-
Stored Procedures		
create_investigador	X	-
create_administrador	X	-
select_medicoes	-	X

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

Qualidade (Frac, Razoável, Boa ou Muito Boa): Boa

Análise crítica (clareza, completude, rigor):

Nenhum utilizador tem a possibilidade de fazer operações de leitura ou de escrita nas tabelas 'medicoestemperatura' e 'medicoesLuminosidade', pelo que as mesmas se tornam completamente inúteis pelo facto de ninguém ter acesso às informações das mesmas.

Na tabela 'dadosexportados' não existe nenhuma coluna para os logs da tabela sistema, pelo que não há forma de registar que dados foram migrados em relação à tabela 'log_sistema'.

Solução Implementada:

Respeitando a especificação fornecida uma vez que a mesma está funcional, apesar das lacunas anteriormente referidas.

1.3 Gestão de Logs

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

Para as ações de *insert* ou *update* deve ser guardada toda a linha na respetiva tabela *log*. Caso a operação realizada seja um *delete* deve ser guardado apenas o ID da linha apagada, deixando os restantes atributos a NULL (excepto os de preenchimento obrigatório).

Nome Trigger	Tabela	Tipo de Operação (I,U,D)	Evento (A, B)	Notas (apenas indicar aquilo que não seja óbvio)
investigador_insert	investigador	I	A	-
investigador_delete	investigador	D	A	-
investigador_update	investigador	U	A	-
cultura_insert	cultura	I	A	-
cultura_delete	cultura	D	A	-
cultura_update	cultura	U	A	-
variaveis_insert	variaveis	I	A	-
variaveis_delete	variaveis	D	A	-
variaveis_update	variaveis	U	A	-
variaveismedidas_insert	variaveismedidas	I	A	-
variaveismedidas_delete	variaveismedidas	D	A	-
variaveismedidas_update	variaveismedidas	U	A	-
medicoes_insert	medicoes	I	A	-
medicoes_delete	medicoes	D	A	-
medicoes_update	medicoes	U	A	-
medicoestemperatura_insert	medicoestemperatura	I	A	-
medicoestemperatura_delete	medicoestemperatura	D	A	-
medicoestemperatura_update	medicoestemperatura	U	A	-
medicoesluminosidade_insert	medicoesluminosidade	I	A	-
medicoesluminosidade_delete	medicoesluminosidade	D	A	-
medicoesluminosidade_update	medicoesluminosidade	U	A	-
sistema_insert	sistema	I	A	-
sistema_delete	sistema	D	A	-
sistema_update	sistema	U	A	-

1.3.1.1 Apreciação Crítica de triggers para gestão de logs

Qualidade (Frac, Razoável, Boa ou Muito Boa): Muito Boa

Breve Justificação:

Foram incluídos todos os tipos de operações sobre todas as respetivas tabelas que podiam despoletar logs.

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)
investigador_insert	X			
investigador_delete	X			
investigador_update	X			
cultura_insert	X			
cultura_delete	X			
cultura_update	X			
variaveis_insert	X			
variaveis_delete	X			
variaveis_update	X			
variaveismedidas_insert	X			
variaveismedidas_delete	X			
variaveismedidas_update	X			
medicoes_insert	X			
medicoes_delete	X			
medicoes_update	X			
medicoestemperatura_insert	X			
medicoestemperatura_delete	X			
medicoestemperatura_update	X			
medicoesluminosidade_insert	X			
medicoesluminosidade_delete	X			

medicoesluminosidade _update	X			
sistema_insert	X			
sistema_delete	X			
sistema_update	X			

1.3.1.2 Triggers Implementados para gestão de logs

1. Nome Trigger: **investigador_insert**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela investigador devido a uma operação de insert.

```
INSERT into log_investigador VALUES (null ,  
new.IDInvestigador, new.Email, new.NomeInvestigador,  
new.CategoriaProfissional, CURRENT_USER, CURRENT_TIME,  
'Insert')
```

2. Nome Trigger: **investigador_delete**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela investigador devido a uma operação de delete.

```
INSERT INTO log_investigador VALUES (null,  
old.IDInvestigador, null, null, null, CURRENT_USER, NOW(),  
"Delete")
```

3. Nome Trigger: **investigador_update**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela investigador devido a uma operação de update.

```
INSERT INTO log_investigador VALUES (null,  
old.IDInvestigador, old.Email, old.NomeInvestigador,  
old.CategoriaProfissional, CURRENT_USER, NOW(), "Update")
```

4. Nome Trigger: **cultura_insert**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela cultura devido a uma operação de insert.

```
log_cultura VALUES ( null ,new.IDCultura, new.NomeCultura,  
new.DescricaoCultura, new.IDInvestigador,CURRENT_USER,  
NOW(), 'Insert')
```

5. Nome Trigger: **cultura_delete**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela cultura devido a uma operação de delete.

```
INSERT INTO log_cultura VALUES (null, old.IDCultura, null,  
null, null, CURRENT_USER, NOW(), "Delete")
```

6. Nome Trigger: **cultura_update**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela cultura devido a uma operação de update.

```
INSERT INTO log_cultura VALUES (null, old.IDCultura,  
old.NomeCultura, old.DescricaoCultura, old.IDInvestigador,  
CURRENT_USER, NOW(), "Update")
```

7. Nome Trigger: **variaveis_insert**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela variáveis devido a uma operação de insert.

```
INSERT into log_variaveis VALUES ( null ,new.IDVariavel,  
new.NomeVariavel,CURRENT_USER, CURRENT_TIME, 'Insert')
```

8. Nome Trigger: **variaveis_delete**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela variáveis devido a uma operação de delete.

```
INSERT INTO log_variaveis VALUES (null, old.IDVariavel,  
null, CURRENT_USER, NOW(), "Delete")
```

9. Nome Trigger: **variaveis_update**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela variáveis devido a uma operação de update.

```
INSERT INTO log_variaveis VALUES (null, old.IDVariavel,  
old.NomeVariavel, CURRENT_USER, NOW(), "Update")
```

10. Nome Trigger: **variaveismedidas_insert**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela variáveis medidas devido a uma operação de insert.

```
INSERT into log_variaveismedidas VALUES  
( null ,new.IDVariavel, new.IDCultura, new.LimiteInferior,  
new.LimiteSuperior,CURRENT_USER, CURRENT_TIME, 'Insert')
```

11. Nome Trigger: **variaveismedidas_delete**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela variáveis medidas devido a uma operação de delete.

```
INSERT INTO log_variaveismedidas VALUES (null,  
old.IDVariavel, old.IDCultura, null, null, CURRENT_USER,  
NOW(), "Delete")
```

12. Nome Trigger: **variaveismedidas_update**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela variáveis medidas devido a uma operação de update.

```
INSERT INTO log_variaveismedidas VALUES (null,  
old.IDVariavel, old.IDCultura, old.LimiteInferior,  
old.LimiteSuperior, CURRENT_USER, NOW(), "Update")
```

13. Nome Trigger: **medicoes_insert**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela medições devido a uma operação de insert.

```
INSERT into log_medicoes VALUES ( null ,new.IDVariavel,  
new.IDCultura, new.NumMedicao, new.DataHoraMedicao,  
new.ValorMedicao,CURRENT_USER, CURRENT_TIME, 'Insert')
```

14. Nome Trigger: **medicoes_delete**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela medições devido a uma operação de delete.

```
INSERT INTO log_medicoes VALUES (null, null, null,  
old.NumMedicao, null, null, CURRENT_USER, NOW(), "Delete")
```

15. Nome Trigger: **medicoes_update**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela medições devido a uma operação de update.

```
INSERT INTO log_medicoes VALUES (null, old.IDVariavel,  
old.IDCultura, old.NumMedicao, old.DataHoraMedicao,  
old.ValorMedicao, CURRENT_USER, NOW(), "Update")
```

16. Nome Trigger: **medicoestemperatura_insert**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela medições de temperatura devido a uma operação de insert.

```
INSERT into log_medicoestemperatura VALUES  
( null ,new.IDMedicao, new.DataHoraMedicao,  
new.ValorMedicaoTemperatura, CURRENT_USER, CURRENT_TIME,  
'Insert')
```

17. Nome Trigger: **medicoestemperatura_delete**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela medições de temperatura devido a uma operação de delete.

```
INSERT INTO log_medicoestemperatura VALUES (null,  
old.IDMedicao, null, null, CURRENT_USER, NOW(), "Delete")
```

18. Nome Trigger: **medicoestemperatura_update**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela medições de temperatura devido a uma operação de update.

```
INSERT INTO log_medicoestemperatura VALUES (null,  
old.IDMedicao, old.DataHoraMedicao,  
old.ValorMedicaoTemperatura, CURRENT_USER, NOW(), "Update")
```

19. Nome Trigger: **medicoesluminosidade_insert**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela medições de luminosidade devido a uma operação de insert.

```
INSERT into log_medicoesluminosidade VALUES  
( null ,new.IDMedicao, new.DataHoraMedicao,  
new.ValorMedicaoLuminosidade, CURRENT_USER, CURRENT_TIME,  
'Insert')
```

20. Nome Trigger: **medicoesluminosidade_delete**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela medições de luminosidade devido a uma operação de delete.

```
INSERT INTO log_medicoesluminosidade VALUES (null,  
old.IDMedicao, null, null, CURRENT_USER, NOW(), "Delete")
```

21. Nome Trigger: **medicoesluminosidade_update**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela medições de luminosidade devido a uma operação de update.

```
INSERT INTO log_medicoesluminosidade VALUES (null,  
old.IDMedicao, old.DataHoraMedicao,  
old.ValorMedicaoLuminosidade, CURRENT_USER, NOW(),  
"Update")
```

22. Nome Trigger: **sistema_insert**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela sistema devido a uma operação de insert.

```
INSERT into log_sistema VALUES ( null ,new.IDSistema,  
new.LimiteInfTemperatura, new.LimiteSupTemperatura,  
new.LimiteInfLuminosidade,  
new.LimiteSupLuminosidade,CURRENT_USER, CURRENT_TIME,  
'Insert')
```

23. Nome Trigger: **sistema_delete**

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela sistema devido a uma operação de delete.

```
INSERT INTO log_sistema VALUES (null, old.IDSistema, null,  
null, null, null, CURRENT_USER, NOW(), "Delete")
```

24. Nome Trigger: sistema_update

O trigger provoca uma inserção de uma nova linha na tabela de logs da tabela sistema devido a uma operação de update.

```
INSERT INTO log_sistema VALUES (null, old.IDSistema,  
old.LimiteInfTemperatura, old.LimiteSupTemperatura,  
old.LimiteInfLuminosidade, old.LimiteSupLuminosidade,  
CURRENT_USER, NOW(), "Update")
```

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

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	Muito breve descrição
create_investigador	Email, NomeInvestigador, Password, CategoriaProfissional	-	Cria um investigador, adiciona-o ao grupo dos investigadores e regista na respetiva tabela.
create_administrador	Nickname, Password	-	Cria um administrador e adiciona-o ao grupo de administradores.
select_medicoes	Condicao*	IDVariavel, IDCultura, NumMedicao, DataHoraMedicao, ValorMedicao	Insere um ou mais registo(s) na tabela <i>log_medicoes</i> , relativa à consulta realizada pelo investigador, na tabela <i>medicoes</i> . Se o utilizador consultar mais do que uma linha, todas as linhas são registadas na tabela <i>log</i> .

* O parâmetro de entrada é a condição WHERE da *query* SELECT. Exemplo: SELECT * FROM *medicoes* WHERE **IDCultura=1**.

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Fraco

Breve Justificação:

Não é especificado onde devem ser guardados os utilizadores nem se o stored procedure "create_investigador" insere dados na tabela de "investigador" e na tabela de utilizadores (mysql.users). Se for o caso, não foi especificado se devem ser criados e inseridos os campos mandatory da tabela de investigadores na tabela mysql.users (exemplo: nome).

Lista de SP (para cada SP assinalar com x em célula correspondente)

	Implementa do de Acordo com Especifica do	Implementa do mas diferente de Especifica do	Não Implementa do	Não Especifica do (criado de novo)
create_investigador		X		
create_administrador		X		
select_medicoes	X			

1.3.2.1 *Stored Procedures Implementados de suporte à criação de logs*

1. Nome SP: **create_investigador**

O stored procedure `create_investigador` insere na tabela de investigadores e na tabela `mysql.user` os dados relativos ao novo investigador.

BEGIN

```
DROP USER IF EXISTS '@localhost;
```

```
ALTER TABLE mysql.user ADD COLUMN IF NOT EXISTS email  
varchar(100);
```

```
SET @sql := CONCAT('CREATE USER ', QUOTE(var_nome), '  
IDENTIFIED BY ', QUOTE(var_password));
```

```
PREPARE statement FROM @sql;  
EXECUTE statement;
```

```
SET @sql := CONCAT('GRANT investigador TO ',  
QUOTE(var_nome));
```

```
PREPARE statement FROM @sql;  
EXECUTE statement;
```

```
SET @sql := CONCAT('SET DEFAULT ROLE investigador FOR  
, QUOTE(var_nome));
```

```
PREPARE statement FROM @sql;  
EXECUTE statement;
```

```
SET @sql := CONCAT('UPDATE mysql.user SET email = ',  
QUOTE(var_email), ' WHERE User=', QUOTE(var_nome));
```

```
PREPARE statement FROM @sql;  
EXECUTE statement;
```

```
SET @sql := CONCAT('INSERT INTO investigador SET  
email=', QUOTE(var_email), ', nomeinvestigador=',  
QUOTE(var_nome), ', categoriaprofissional=',  
QUOTE(var_categoria_profissional));
```

```
PREPARE statement FROM @sql;  
EXECUTE statement;
```

```
DEALLOCATE PREPARE statement;  
FLUSH PRIVILEGES;
```

END

2. Nome SP: `create_administrador`

O stored procedure `create_administrador` insere na tabela `mysql.user` os dados relativos ao novo administrador.

```
BEGIN
```

```
    DROP USER IF EXISTS '@localhost;
```

```
    SET @sql := CONCAT('CREATE USER ', QUOTE(var_nome), '
IDENTIFIED BY ', QUOTE(var_password));
    PREPARE statement FROM @sql;
    EXECUTE statement;
```

```
    SET @sql := CONCAT('GRANT administrador TO ',
QUOTE(var_nome));
    PREPARE statement FROM @sql;
    EXECUTE statement;
```

```
    SET @sql := CONCAT('SET DEFAULT ROLE administrador FOR
', QUOTE(var_nome));
    PREPARE statement FROM @sql;
    EXECUTE statement;
```

```
    DEALLOCATE PREPARE statement;
    FLUSH PRIVILEGES;
```

```
END
```

3. Nome SP: `select_medicoes`

O stored procedure `select_medicoes` devolve dados relativos à tabela de medições com determinados filtros inseridos numa condição passada como parâmetro de entrada.

```
BEGIN
```

```
    IF var_condicao = "" THEN SET var_condicao = "1=1";
    END IF;
```

```
    SET @sql := CONCAT('INSERT log_medicoes (IDVariavel,
IDCultura, NumMedicao, DataHoraMedicao, ValorMedicao,
Utilizador, Data, Operacao) SELECT IDVariavel, IDCultura,
NumMedicao, DataHoraMedicao, ValorMedicao, CURRENT_USER,
NOW(), "Select" FROM medicoes WHERE ', var_condicao);
    PREPARE statement FROM @sql;
    EXECUTE statement;
```

```
    SET @sql := CONCAT('SELECT IDVariavel, IDCultura,
```

```
NumMedicao, DataHoraMedicao, ValorMedicao FROM medicoes  
WHERE ', var_condicao);  
    PREPARE statement FROM @sql;  
    EXECUTE statement;  
  
END
```

1.4 Migração entre Bases de Dados

1.4.1 Esquema relacional da base de Dados Mysql (destino)

Os atributos *mandatory* (ou seja, de preenchimento obrigatório) encontram-se assinalados com a letra M. Os atributos *unique* (ou seja, únicos) encontram-se assinalados com U.



1.4.1.1 Apreciação Crítica e esquema relacional implementado

Qualidade (Frac, Razoável, Boa ou Muito Boa): Boa

Breve Justificação:

Não está especificado ao que corresponde o atributo utilizador que surge em todas as tabelas.

Poderia ser utilizada apenas uma tabela de logs facilitando imenso a pesquisa por utilizador e por data.

Foram feitas alterações? (Sim/Não): Sim

Novo Esquema (assinale e justifique as alterações)

Alterar o nome do atributo 'Utilizador' para 'EmailUtilizador' em todas as tabelas.

auditoroutrogrupo_log_cultura IDLog : int(11) IDCultura : int(11) NomeCultura : varchar(45) DescricaoCultura : varchar(45) IDInvestigador : int(11) EmailUtilizador : varchar(45) Data : datetime Operacao : varchar(45)	auditoroutrogrupo_log_variaveis IDLog : int(11) IDVariavel : int(11) NomeVariavel : varchar(45) EmailUtilizador : varchar(45) Data : datetime Operacao : varchar(45)	auditoroutrogrupo_log_variaveismedidas IDLog : int(11) IDVariavel : int(11) IDCultura : int(11) LimiteInferior : int(11) LimiteSuperior : int(11) EmailUtilizador : varchar(45) Data : datetime Operacao : varchar(45)	auditoroutrogrupo_log_investigador IDLog : int(11) IDInvestigador : int(11) Email : varchar(45) NomeInvestigador : varchar(45) CategoriaProfissional : varchar(45) EmailUtilizador : varchar(45) Data : datetime Operacao : varchar(45)
auditoroutrogrupo_log_medicoes IDLog : int(11) IDVariavel : int(11) IDCultura : int(11) NumMedicao : int(11) DataHoraMedicao : datetime ValorMedicao : int(11) EmailUtilizador : varchar(45) Data : datetime Operacao : varchar(45)	auditoroutrogrupo_log_sistema IDLog : int(11) IDSistema : int(11) LimiteInfTemperatura : int(11) LimiteSupTemperatura : int(11) LimiteInfLuminosidade : int(11) LimiteSupLuminosidade : int(11) EmailUtilizador : varchar(45) Data : datetime Operacao : varchar(45)	auditoroutrogrupo_log_medicoesluminosidade IDLog : int(11) IDMedicao : int(11) DataHoraMedicao : datetime ValorMedicaoLuminosidade : int(11) EmailUtilizador : varchar(45) Data : datetime Operacao : varchar(45)	auditoroutrogrupo_log_medicoestemperatura IDLog : int(11) IDMedicao : int(11) DataHoraMedicao : datetime ValorMedicaoTemperatura : int(11) EmailUtilizador : varchar(45) Data : datetime Operacao : varchar(45)

1.4.2 Forma de Migração

A processo de migração de base de dados por ficheiro engloba duas etapas principais: a escrita para um ficheiro e a posterior leitura do mesmo. Todo este processo bem como as decisões que devem ser tomadas encontram-se descritos abaixo.

Ficheiro

Deve optar-se pela exportação através de um ficheiro .csv. O CSV (*Comma Separated Values*) é um formato de ficheiro no qual a informação surge organizada como um ficheiro de texto simples. Contem, normalmente, uma lista de valores divididos em linhas e colunas. As colunas encontram-se separadas por vírgulas e as linhas devem terminar com um *enter*.

A maior vantagem deste tipo de ficheiro é a portabilidade. Devido o facto de ser constituído por texto plano, o espaço necessário para guardar um ficheiro deste tipo é bastante reduzido.

Periodicidade

A migração deve ser realizada de 8 em 8 horas. A exportação tem início às 5h00 e ocorre posteriormente às 13h00 e, de seguida, às 21h00. Por sua vez, a importação, de modo a garantir a sincronização com o processo anterior ocorre com a mesma periodicidade, mas com um atraso de 10 minutos.

Deve optar-se por este horário para que não interfira com o horário laboral visto que a migração ocorre no início do dia, interrupção de almoço e, por fim, à noite.

Controlo de Duplicados

O controlo de duplicados é assegurado pela tabela *dadosexportados* da base de dados origem. Esta tabela deve ser constituída por uma única linha, em que cada atributo corresponde ao último ID exportado de cada uma das tabelas *log* da BD origem, conforme exemplo abaixo:

IDExportacao	IDLogInvestigador	IDLogMedicoesLuminosidade	IDLogVariaveis	IDLogMedicoesTemperatura	IDLogMedicoes	IDLogCultura	IDLogVariaveisMedidas
1	5	4	5	5	3	2	3

A cada exportação, os atributos são atualizados com o ID da última linha exportada (os IDs são incrementais). Cada *stored procedure* (de apoio à exportação) é responsável por atualizar o campo correspondente à tabela que está a exportar. Assim, é assegurada a migração incremental.

Exportação

Para a realização da exportação devem ser seguidos os seguintes passos:

1. Criar a tabela *dadosexportados* na base de dados origem (encontra-se explicado na secção anterior de que forma a tabela vai realizar o controlo de duplicados);
2. Criar *stored procedures* para cada uma das tabelas que serão exportadas (encontram-se especificados na secção 1.4.5.). Cada tabela tem um ficheiro respetivo (exemplo: a

- tabela *log_investigador* tem o ficheiro *log_investigador.csv*) e deve ser utilizada a função *OUTFILE* para a escrita no ficheiro *.csv*;
3. Criar um *stored procedure* para a realização de uma exportação manual (especificado na secção 1.4.5.) que invoca todos os SPs anteriores e que pode ser executado pelo administrador;
 4. Criar um evento *export_data* (especificado na secção 1.4.6.) que invoca o SP anterior e que define a periodicidade de 8 em 8 horas, com início às 5h00.

Importação

Para a realização da importação devem ser seguidos os seguintes passos:

1. Criar o script batch (*.bat*), para importar os dados dos ficheiros *.csv* para as tabelas da base de dados destino. De seguida os ficheiros devem ser apagados. O exemplo abaixo é um excerto de um código-tipo para o script, onde *nome_bd* deve corresponder ao nome da base de dados destino, *password* a palavra-passe para o utilizador *root* da BD, *file_path* o caminho para cada ficheiro *.csv* e *nome_tabela* o nome da tabela para a qual serão importados os dados;

```
cd C:\Program Files\MySQL\MySQL Server 8.0\bin\
mysql --user=root --password="password" --database="nome_bd" --execute="LOAD DATA INFILE ' "file_path" ' INTO TABLE "nome_tabela" FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES
TERMINATED BY '\r\n'
del "file_path"
```

2. Criar o evento de importação, *import_data* (especificado na secção 1.4.6.) através do sistema operativo *Windows*. Deverão ser seguidos os seguintes passos:
 - i. Abrir a *pesquisa* do *Windows*, escrever “*task*” e escolher o *Task Scheduler*;
 - ii. Clicar em *Create Task* no painel *Action*;
 - iii. No painel *General*, definir o nome *import_data* e, nas opções de segurança, escolher “*run whether user is logged on or not*”;
 - iv. No painel *Triggers*, clicar em *New* e, de seguida, na lista *drop-down* seleccionar “*on a schedule*”;
 - v. Escolher a opção “*one time*” e definir para ter início no dia seguinte às 5:10h;
 - vi. Em *Advanced Settings*, seleccionar “*repeat task every*” com os parâmetros “*8 hours*” e “*indefinitely*”;
 - vii. No painel *Actions*, clicar em *New* e, de seguida, escolher a ação “*start a program*”;
 - viii. Na lista *drop-down*, localizar e escolher o ficheiro *.bat* de *import*, criado anteriormente;
 - ix. Concluir a criação clicando em *OK*.

Privacidade dos Dados

Uma vez que após cada importação a informação guardada em cada ficheiro *.csv* é eliminada, não há necessidade de encriptar os dados visto que esta fica disponível apenas durante os 10 minutos de diferença entre o evento de exportação e importação.

Facilidade de Manutenção

Em caso de falha na migração, através da tabela *dadosexportados* é garantido que não existe perda de dados. Se a importação falhar, os ficheiros .csv não serão apagados, impedindo o *overwriting* de novos dados. Assim, na importação seguinte, os dados que previamente falharam são importados.

Se ocorrerem erros com a exportação, quer os ficheiros .csv quer a tabela *dadosexportados* não são atualizados, consequentemente nenhuma informação é importada para a base de dados destino. Na próxima exportação, quer os dados que falharam, quer os novos são exportados.

1.4.2.1 *Apreciação Crítica à especificação da forma de migração*

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Fraco

Análise crítica (clareza, completude, rigor):

A especificação relativa à forma de migração é apresentada de forma clara.

Quanto à portabilidade não é explicado o que torna este tipo de ficheiro mais portátil do que o XML ou JSON.

Na nossa perspetiva o tipo de ficheiro mais adequado seria o JSON, uma vez que o espaço ocupado nos dias de hoje não é um critério com grande relevância devido à redução do preço da memória e devido ao pequeno volume de dados associado a este caso concreto. O JSON é um formato que permite uma melhor e mais fácil organização hierárquica de dados e que tem um grande número de analisadores disponíveis para executar o 'parse' dos mesmos.

O facto de o ficheiro ficar disponível por dez minutos até que seja executada a task que copia a informação para a base de dados destino pode trazer problemas ao nível da segurança na medida em que este fica disponível para leitura por parte de qualquer pessoa, visto que não foi encriptado e pode dar-se o caso de alguém eliminar esse mesmo ficheiro acidentalmente ou propositadamente fazendo com que a migração não seja feita com sucesso e os dados sejam marcados como migrados na base de dados de origem levando à perda de informação. Devia ter sido repensada a necessidade de encriptar os dados devido ao grande período de vulnerabilidade a que os ficheiros estão sujeitos.

Este período de espera é inevitável caso se decida utilizar o ficheiro para fazer a migração, pelo que deveria ter sido especificada uma solução que verificasse qual era o maior id exportado relativo a cada uma das tabelas para assim se ter a certeza que os dados chegaram corretamente ao destino.

Isto podia ser feito através de uma ligação à base de dados destino para assim se proceder a um acerto da tabela 'dados migrados' da base de dados origem.

Posto isto, consideramos que a exportação feita através de PHP é bastante mais robusta e segura.

Adicionalmente, utilizando o stored procedure que altera a periodicidade de migração na base de dados origem, gera-se uma inconsistência, uma vez que esse stored procedure apenas altera a periodicidade na origem e não faz a alteração correspondente no evento de importação relativo à base de dados destino.

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

Na tabela abaixo são indicados os privilégios que cada utilizador tem sobre as tabelas e os *stored procedures* em que E=Escrita, L=Leitura, X=Executar e - = sem permissões.

Tal como na secção 1.2 deve considerar-se que o Administrador tem permissões de leitura sobre todas as tabelas indicadas. O Auditor tem como função a monitorização de utilizações indevidas, como tal, necessita apenas de permissões de leitura sobre as tabelas *log* da base de dados destino.

Base de Dados (O/D)	Tabela	Tipo de Utilizador	
		Administrador	Auditor
D	log_cultura	L	L
D	log_medicoes	L	L
D	log_medicoestemperatura	L	L
D	log_medicoesluminosidade	L	L
D	log_variaveis	L	L
D	log_variaveismedidas	L	L
D	log_investigador	L	L
D	log_sistema	L	L
	Stored Procedures		
O	export_log_investigador	X	-
O	export_log_variaveis	X	-
O	export_log_medicoesluminosidade	X	-
O	export_log_variaveismedidas	X	-
O	export_log_medicoes	X	-
O	export_log_sistema	X	-
O	export_log_cultura	X	-
O	export_log_medicoestemperatura	X	-
O	export_manual	X	-
O	mudar_horario_export	X	-

1.4.3.1 Apreciação Crítica à especificação da Gestão de Utilizadores

Qualidade (Frac, Razoável, Boa ou Muito Boa): Muito Boa

Análise crítica (clareza, completude, rigor):

A especificação apresentada é fácil de interpretar.
Todas as tabelas são contempladas pela solução, pelo que os stored procedures também estão corretamente especificados, sendo por isso uma solução sólida.

Solução Implementada:

Foi implementada uma solução que respeita a especificação fornecida.

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

Não é relevante a criação de *triggers* de suporte à migração de dados.

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

1.4.4.1 *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)

	Implementa do de Acordo com Especifica do	Implementa do mas diferente de Especifica do	Não Implementa do	Não Especifica do (criado de novo)
Nome Trigger (tal como especifica do)				

1.4.4.2 Triggers Implementados de suporte à migração de dados

```
1. Nome Trigger: _____  
// Breve Descrição  
Código
```

1.4.5 Stored Procedures de suporte à migração de dados

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	BD (Origem ou Destino)	Muito breve descrição
export_log_investigador	-	-	O	Seleciona as linhas da tabela <i>log_investigador</i> que não foram exportadas, cria o ficheiro <i>log_investigador.csv</i> e regista-as. Atualiza a tabela <i>dadosexportados</i> com o último ID exportado.
export_log_variaveis	-	-	O	Seleciona as linhas da tabela <i>log_variaveis</i> que não foram exportadas, cria o ficheiro <i>log_variaveis.csv</i> e regista-as. Atualiza a tabela <i>dadosexportados</i> com o último ID exportado.
export_log_medicoesluminosidade	-	-	O	Seleciona as linhas da tabela <i>log_medicoesluminosidade</i> que não foram exportadas, cria o ficheiro <i>log_medicoesluminosidade.csv</i> e regista-as. Atualiza a tabela <i>dadosexportados</i> com o último ID exportado.
export_log_variaveismedidas	-	-	O	Seleciona as linhas da tabela <i>log_variaveismedidas</i> que não foram exportadas, cria o ficheiro <i>log_variaveismedidas.csv</i> e regista-as. Atualiza a tabela <i>dadosexportados</i> com o último ID exportado.
export_log_medicoes	-	-	O	Seleciona as linhas da tabela <i>log_medicoes</i> que não foram exportadas, cria o ficheiro <i>log_medicoes.csv</i> e regista-as. Atualiza a tabela <i>dadosexportados</i> com o último ID exportado.
export_log_sistema	-	-	O	Seleciona as linhas da tabela <i>log_sistema</i> que não foram exportadas, cria o ficheiro

				<i>log_sistema.csv</i> e regista-as. Atualiza a tabela <i>dadosexportados</i> com o último ID exportado.
export_log_cultura	-	-	O	Seleciona as linhas da tabela <i>log_cultura</i> que não foram exportadas, cria o ficheiro <i>log_cultura.csv</i> e regista-as. Atualiza a tabela <i>dadosexportados</i> com o último ID exportado.
export_log_medicoestemperatura	-	-	O	Seleciona as linhas da tabela <i>log_medicoestemperatura</i> que não foram exportadas, cria o ficheiro <i>log_medicoestemperatura.csv</i> e regista-as. Atualiza a tabela <i>dadosexportados</i> com o último ID exportado.
export_manual	-	-	O	Possibilita um <i>export</i> manual e invoca todos os <i>stored procedures</i> anteriores.
mudar_horario_export	Horainicio, Periodicidade	-	O	Altera a hora de início do evento <i>export</i> e a periodicidade.

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

Qualidade (Fracas, Razoável, Boa ou Muito Boa): Boa

Breve Justificação:

Mediante a decisão de uma tabela de logs para cada tabela, esta solução revela-se bastante robusta e completa.

De referir que o stored procedure "mudar_horario_export" altera a periodicidade apenas na tabela de origem, o que pode causar problemas de incompatibilidade na importação para a base de dados destino.

Consideramos que criar uma nova linha na tabela de logs por cada linha na tabela visualizada no select, dificulta a consulta por parte do auditor porque quantos mais dados existirem mais tempo demorará a executar cada select.

Adicionalmente, caso o auditor queira ver toda a informação que foi visualizada com um único comando select pode não conseguir fazer uma query aos seus logs para escolher apenas as linhas referentes ao mesmo, caso o valor da data incremente durante os inserts nas tabelas log.

Lista de SP (para cada SP assinalar com x em célula correspondente)

	Impleme ntado de Acordo com Especif icado	Impleme ntado mas diferen te de Especif icado	Não Impleme ntado	Não Especif icado (criado de novo)
export_log_investiga dor	X			
export_log_variaveis	X			
export_log_medicoesl uminosidade	X			
export_log_variaveis medidas	X			
export_log_medicoes	X			
export_log_sistema	X			
export_log_cultura	X			
export_log_medicoest emperatura	X			
export_manual	X			
mudar_horario_export			X	

1.4.5.2 Stored Procedures Implementados de suporte à migração de dados

1. Nome SP: **export_log_investigador**

Este stored procedure exporta a tabela de logs relativa à tabela de investigadores para o ficheiro csv correspondente.

BEGIN

```
SELECT * INTO OUTFILE 'C:logs_investigador.csv '
  FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '"'
  LINES TERMINATED BY '\n'
  FROM log_investigador WHERE log_investigador.IDLog >
(SELECT IDLogInvestigador FROM `dadosexportados`) ;
```

```
UPDATE `dadosexportados` SET `IDLogInvestigador`= (SELECT
IDLog FROM log_investigador ORDER BY IDLog DESC LIMIT 1);
END
```

2. Nome SP: **export_log_variaveis**

Este stored procedure exporta a tabela de logs relativa à tabela de variáveis para o ficheiro csv correspondente.

BEGIN

```
SELECT * INTO OUTFILE 'C:logs_variaveis.csv '
  FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '"'
  LINES TERMINATED BY '\n'
  FROM log_variaveis WHERE log_variaveis.IDLog > (SELECT
IDLogVariaveis FROM `dadosexportados`) ;
```

```
UPDATE `dadosexportados` SET `IDLogVariaveis`= (SELECT
IDLog FROM log_variaveis ORDER BY IDLog DESC LIMIT 1);
```

END

3. Nome SP: **export_log_medicoesluminosidade**

Este stored procedure exporta a tabela de logs relativa à tabela de medições de luminosidade para o ficheiro csv correspondente.

BEGIN

```
SELECT * INTO OUTFILE 'C:logs_medicoesluminosidade.csv '
  FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '"'
  LINES TERMINATED BY '\n'
```

```

FROM log_medicoesluminosidade WHERE
log_medicoesluminosidade.IDLog > (SELECT
IDLogMedicoesLuminosidade FROM `dadosexportados`) ;

UPDATE `dadosexportados` SET `IDLogMedicoesLuminosidade`=
(SELECT IDLog FROM log_medicoesluminosidade ORDER BY IDLog
DESC LIMIT 1);

END

```

4. Nome SP: **export_log_variaveismedidas**

Este stored procedure exporta a tabela de logs relativa à tabela de variáveis medidas para o ficheiro csv correspondente.

```

BEGIN

SELECT * INTO OUTFILE 'C:logs_variaveismedidas.csv '
FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM log_variaveismedidas WHERE
log_variaveismedidas.IDLog > (SELECT IDLogVariaveisMedidas
FROM `dadosexportados`) ;

UPDATE `dadosexportados` SET `IDLogVariaveisMedidas`=
(SELECT IDLog FROM log_variaveismedidas ORDER BY IDLog DESC
LIMIT 1);

END

```

5. Nome SP: **export_log_medicoes**

Este stored procedure exporta a tabela de logs relativa à tabela de medições para o ficheiro csv correspondente.

```

BEGIN

SELECT * INTO OUTFILE 'C:logs_medicoes.csv '
FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM log_medicoes WHERE log_medicoes.IDLog > (SELECT
IDLogMedicoes FROM `dadosexportados`) ;

UPDATE `dadosexportados` SET `IDLogMedicoes`= (SELECT IDLog
FROM log_medicoes ORDER BY IDLog DESC LIMIT 1);

END

```

6. Nome SP: **export_log_sistema**

Este stored procedure exporta a tabela de logs relativa à tabela de sistema para o ficheiro csv correspondente.

BEGIN

```
SELECT * INTO OUTFILE 'C:logs_sistema.csv '  
  FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY ''''  
  LINES TERMINATED BY '\n'  
  FROM log_sistema WHERE log_sistema.IDLog > (SELECT  
IDLogSistema FROM `dadosexportados`);
```

```
UPDATE `dadosexportados` SET `IDLogSistema`= (SELECT IDLog  
FROM log_sistema ORDER BY IDLog DESC LIMIT 1);
```

END

7. Nome SP: **export_log_cultura**

Este stored procedure exporta a tabela de logs relativa à tabela de cultura para o ficheiro csv correspondente.

BEGIN

```
SELECT * INTO OUTFILE 'C:logs_cultura.csv '  
  FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY ''''  
  LINES TERMINATED BY '\n'  
  FROM log_cultura  
  WHERE log_cultura.IDLog > (SELECT IDLogCultura FROM  
`dadosexportados`);
```

```
UPDATE `dadosexportados` SET `IDLogCultura`= (SELECT IDLog  
FROM log_cultura ORDER BY IDLog DESC LIMIT 1);
```

END

8. Nome SP: **export_log_medicoestemperatura**

Este stored procedure exporta a tabela de logs relativa à tabela de medições de temperatura para o ficheiro csv correspondente.

BEGIN

```
SELECT * INTO OUTFILE 'C:logs_medicoestemperatura.csv '  
  FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY ''''  
  LINES TERMINATED BY '\n'  
  FROM log_medicoestemperatura WHERE  
log_medicoestemperatura.IDLog > (SELECT  
IDLogMedicoesTemperatura FROM `dadosexportados`);
```

```
UPDATE `dadosexportados` SET `IDLogMedicoesTemperatura`=  
(SELECT IDLog FROM log_medicoestemperatura ORDER BY IDLog  
DESC LIMIT 1);  
END
```

9. Nome SP: **export_manual**

Este stored procedure invoca todos os stored procedures de exportação e permite que a mesma seja feita de forma manual (a pedido).

```
BEGIN
```

```
CALL export_cultura;  
CALL export_investigador;  
CALL export_medicoes;  
CALL export_medicoesluminosidade;  
CALL export_medicoestemperatura;  
CALL export_sistema;  
CALL export_variaveis;  
CALL export_variaveismedidas;
```

```
END
```

10. Nome SP: **mudar_horario_export**

Este stored procedure não foi implementado uma vez que por questões de segurança, o phpMyAdmin não permite que a periodicidade dos eventos seja modificada através de stored procedures.

Eventos de suporte à migração de dados

Nome Evento	Local Execução (Origem ou Destino, ou Sistema Operativo)	Muito breve descrição
export_data	Origem	Este evento ocorre com uma periodicidade de 8 em 8 horas, com início às 5h00, invocando o <i>stored procedure export_manual</i> .
import_data	Sistema Operativo	Este evento ocorre com uma periodicidade de 8 em 8 horas, com início às 5h10, executando o <i>script</i> criado para a importação.

1.4.5.3 Apreciação Crítica de Eventos

Qualidade (Frac, Razoável, Boa ou Muito Boa): Razoável

Breve Justificação:

O período de 10 minutos de diferença entre os eventos faz com que surjam problemas de segurança na medida em que o ficheiro fica por 10 minutos disponível para consulta por parte de qualquer pessoa, modificação dos dados e também a possibilidade de eliminação dos mesmos. Neste caso a migração é considerada bem sucedida levando à perda de dados, pelo que consideramos que devia ser implementado um mecanismo de controlo que verificasse no fim da exportação se os dados realmente chegaram à base de dados destino. Para isso, devia existir uma conexão à base de dados do auditor a fim de averiguar o maior id relativo a cada uma das tabelas de logs, tentando assim perceber se a migração tinha sido bem sucedida.

Lista de Eventos (para cada evento assinalar com x em célula correspondente)

	Implementa do de Acordo com Especifica do	Implementa do mas diferente de Especifica do	Não Implementa do	Não Especifica do (criado de novo)
export_da ta	X			
import_da ta	X			

1.4.5.4 Eventos Implementados

1. Nome Evento: **export_data**

Este stored procedure chama a execução o stored procedure `export_manual`, despoletando assim a exportação de todas as tabelas para os respetivos ficheiros csv.

```
CREATE EVENT IF NOT EXISTS export_data
ON SCHEDULE
    EVERY 8 HOUR
    STARTS '2019-03-18 5:00:00'
DO
    CALL export_data
```

2. Nome Evento: **import_data**

Este stored procedure importa os dados presentes nos ficheiros csv para a base de dados de destino.

```
mysql --user=root --database=bdauditorgr18 --execute="LOAD
DATA LOCAL INFILE 'C:/xampp/mysql/data/logs_cultura.csv'
INTO TABLE log_cultura FIELDS TERMINATED BY ';' OPTIONALLY
ENCLOSED BY '"' LINES TERMINATED BY '\n';
LOAD DATA LOCAL INFILE
'C:/xampp/mysql/data/logs_investigador.csv' INTO TABLE
log_investigador FIELDS TERMINATED BY ';' OPTIONALLY
ENCLOSED BY '"' LINES TERMINATED BY '\n';
LOAD DATA LOCAL INFILE
'C:/xampp/mysql/data/logs_medicoes.csv' INTO TABLE
log_medicoes FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED
BY '"' LINES TERMINATED BY '\n';
LOAD DATA LOCAL INFILE
'C:/xampp/mysql/data/logs_medicoesluminosidade.csv' INTO
TABLE log_medicoesluminosidade FIELDS TERMINATED BY ';'
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
LOAD DATA LOCAL INFILE
'C:/xampp/mysql/data/logs_medicoestemperatura.csv' INTO
TABLE log_medicoestemperatura FIELDS TERMINATED BY ';'
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
LOAD DATA LOCAL INFILE
'C:/xampp/mysql/data/logs_sistema.csv' INTO TABLE
log_sistema FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY
'"' LINES TERMINATED BY '\n';
LOAD DATA LOCAL INFILE
'C:/xampp/mysql/data/logs_variaveis.csv' INTO TABLE
log_variaveis FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED
BY '"' LINES TERMINATED BY '\n';
LOAD DATA LOCAL INFILE
'C:/xampp/mysql/data/logs_variaveismedidas.csv' INTO TABLE
```

```
log_variaveismedidas FIELDS TERMINATED BY ';' OPTIONALLY  
ENCLOSED BY '"' LINES TERMINATED BY '\n'
```

```
del "C:\xampp\mysql\data\logs_cultura.csv"  
del "C:\xampp\mysql\data\logs_investigador.csv"  
del "C:\xampp\mysql\data\logs_medicoes.csv"  
del "C:\xampp\mysql\data\logs_medicoesluminosidade.csv"  
del "C:\xampp\mysql\data\logs_medicoestemperatura.csv"  
del "C:\xampp\mysql\data\logs_sistema.csv"  
del "C:\xampp\mysql\data\logs_variaveis.csv"  
del "C:\xampp\mysql\data\logs_variaveismedidas.csv"
```


1.4.6 PHP suporte à migração de dados (se relevante)

1.4.6.1 Apreciação Crítica ao PHP especificado

Qualidade (Frac, Razoável, Boa ou Muito Boa): _____

Breve Justificação:

1.4.6.2 PHP Implementado

Código

1.5 Avaliação Global de especificações da Etapa A

No esquema relacional da base de dados de origem não foi especificado o significado do atributo 'Utilizador' constante em todas as tabelas de logs, pelo que não se sabe qual o critério utilizado para a identificação dos mesmos.

Não ficou claro onde devem ser registados os utilizadores nem qual é o stored procedure que deve ser utilizado para esse fim.

Relativamente às migrações surge um problema grave de segurança decorrente do período de 10 minutos que existe entre a exportação para o ficheiro '.csv' e a respetiva importação para a base de dados destino. Este problema pode provocar a perda dos dados devido à eliminação do ficheiro antes da importação dos dados existentes no mesmo, sem que seja detetado o problema numa posterior migração, uma vez que na tabela que verifica os dados que foram migrados, estes surgem como tendo sido corretamente migrados. Por esta razão devia ter sido especificado um mecanismo de controlo que verificasse no fim do período previsto para a migração e recorrendo a uma conexão à base de dados destino qual tinha sido o maior id relativo a cada uma das tabelas de logs efetivamente migrado. Desta forma era garantido que esta vulnerabilidade existente, que é inerente à migração por ficheiro era minimizada.

Adicionalmente, qualquer pessoa pode ler o ficheiro durante o período de dez minutos e até mesmo modificá-lo, fazendo com que o auditor tenha acesso a dados corrompidos, sendo por isso a estratégia de migração por php mais robusta. Esta questão podia ter sido minimizada com uma encriptação do ficheiro.

Em relação à periodicidade, quando a mesma é alterada na base de dados de origem, essa mesma mudança não é refletida no evento do sistema operativo. Esta situação provoca problemas de coerência na migração, visto que quando ocorre o evento do sistema operativo que importa os dados para a base de dados do auditor pode dar-se um dos seguintes casos:

- A exportação para o ficheiro pode ter sido adiada em mais de dez minutos (período de espera entre exportação e

importação especificado) e quando se tenta fazer a importação ainda não existe nenhum ficheiro a importar.

- O evento responsável pela importação pode ter sido antecipado em mais de dez minutos fazendo com que a importação não ocorra pela falta de um ficheiro de onde importar os dados.

De referir que não foram especificadas as permissões de leitura sobre a tabela de medições luminosidade e medições de temperatura o que faz com que as mesmas se tornem inúteis.

Adicionalmente, não existe nenhuma coluna na tabela de 'dadosexportados' destinada à tabela 'sistema', pelo que não há forma de registar os logs dessa mesma tabela.

Em geral consideramos que a especificação fornecida é razoável na medida em que é funcional, apesar de ter alguns problemas ao nível da segurança e não ser completa em alguns aspetos como o tipo de informação que o atributo 'Utilizador' representa ou como deve ser representado o tipo operação nas tabelas de logs.

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

-Não foi especificado o local onde devem ser registados os utilizadores, nem qual o stored procedure que deve ser utilizado para o efeito.

- Intervalo entre exportação e importação demasiado longo, o que leva a problemas ao nível da segurança devido à opção de não encriptar o ficheiro e possíveis perdas de dados sem que haja um mecanismo de controlo na base de dados destino que verifique os dados efetivamente migrados.

- Foram esquecidas as permissões de leitura e escrita sobre as tabelas 'medições temperatura' e ' medições luminosidade', o que faz com que as mesmas percam toda a sua utilidade.

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	-	-	-	-

2 Etapa C (Especificação e Implementação do Próprio Grupo)

2.1 Especificação do Esquema relacional da base de Dados Origem

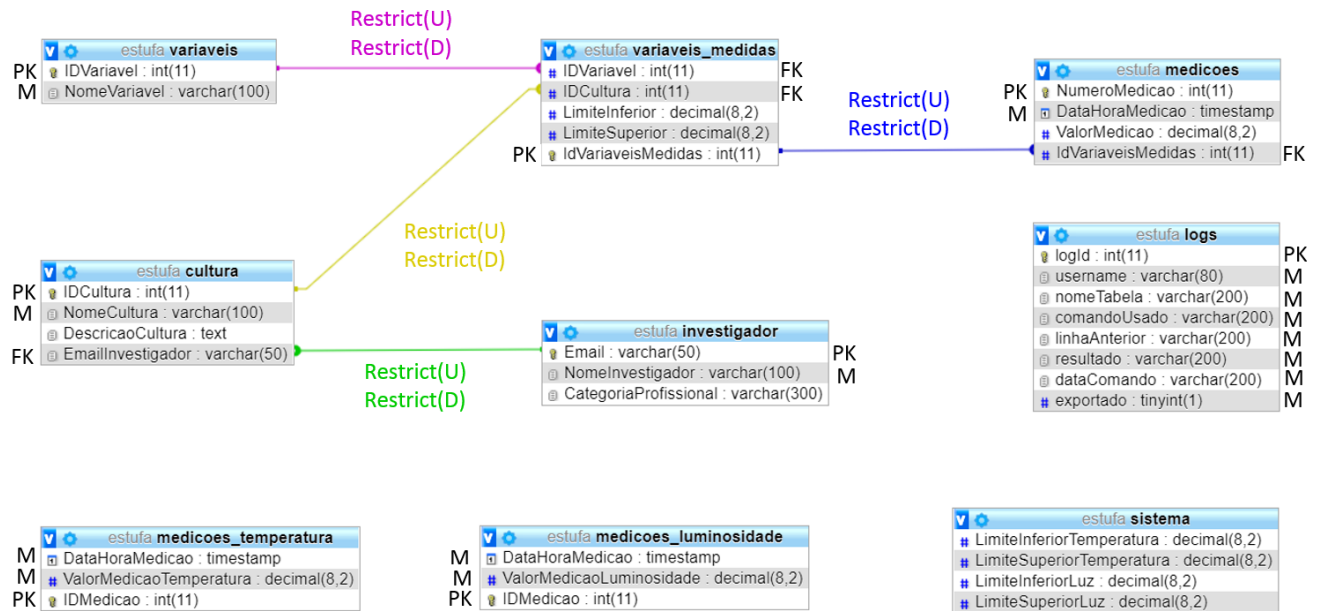


Tabela	Tipo de Utilizador				
	Investigador	Administrador ou Aplicação	Sensor Temperatura	Sensor Luz	phpUser
sistema	L	E, L	-	-	-
variaveis	E, L	E, L	-	-	-
investigador	E, L	E, L	-	-	-
medicoes	E, L	E, L	-	-	-
variaveis_medidas	E, L	E, L	-	-	-
medicao_luminosidade	L	E, L	-	E	-
medicao_temperatura	L	E, L	E	-	-
cultura	E, L	E, L	-	-	-
logs	-	-	-	-	-
mysql.users	-	E, L	-	-	-
Stored Proc.					
add_User	-	X	-	-	-
init	-	X	-	-	-
selectMedicoes	X	X	-	-	-
alterarLimites Temperatura	-	X	-	-	-
alterarLimites Luz	-	X	-	-	-
deleteUser	-	X	-	-	-
alterarValorMedido	X	X	-	-	-
apagarMedicao	X	X	-	-	-

apagarCultura	X	X	-	-	-
selectDadosNaoMigrados	-	-	-	-	X
updateMigrados	-	-	-	-	X

2.2 Especificação de Utilizadores

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

O investigador relativamente às tabelas de medição de luminosidade e medição de temperatura tem apenas permissões de leitura, uma vez que apenas os sensores correspondentes podem inserir os resultados das medições. Quanto à tabela sistema o investigador pode apenas realizar operações de leitura visto que os valores limites de temperatura e luminosidade afetam toda a estufa e consequentemente afetam as culturas de outros investigadores, pelo que terá de ser o administrador da aplicação a fazer a gestão desses valores mediante um prévio acordo com os vários investigadores.

O role 'phpUser' serve apenas para executar os Stored Procedures 'selectDadosNaoMigrados' e 'updateMigrados' que servem respetivamente para devolver uma lista de todos os dados que ainda não foram migrados e marcar como migrados os dados que já foram corretamente migrados.

Por sua vez, os sensores apenas podem efetuar operações de escrita nas tabelas correspondentes.

O SP 'add_User' deve receber como parâmetros o nome do utilizador, a password, o email e o role do utilizador a inserir e deve registá-lo na tabela mysql. users de acordo com a informação fornecida pelo administrador da aplicação, que deve ser o único a poder realizar esta operação.

Por sua vez o SP 'init' deve criar os roles de investigador, auditor, administrador da aplicação, sensor luminosidade, sensor temperatura e phpUser, bem como atribuir os respetivos privilégios de acordo com o especificado na tabela anterior.

O SP 'selectMedicoes' para além de retornar a tabela de medições de acordo com a condição recebida como parâmetro deve também acionar um trigger que deve criar uma nova entrada na tabela de logs, onde é inserido na coluna correspondente o tipo de comando executado e a condição especificada no campo resultado. Tanto o administrador da aplicação como o investigador devem poder executar este stored procedure, uma vez que são as únicas pessoas que têm permissões de leitura sobre a tabela medições e sobre as quais há necessidade de guardar informação de logs.

Os SP's 'alterarLimitesLuz' e 'alterarLimitesTemp' devem receber os limites inferior e superior da luminosidade/temperatura como parâmetros e alterar nos campos correspondentes da tabela sistema.

O SP 'deleteUser' deve permitir eliminar um determinado utilizador dado o seu email como parâmetro.

O SP 'alterarValorMedido' deve implementar a funcionalidade de dado o id da medição e o novo valor da mesma, fazer a respetiva alteração na tabela medições.

2.3 Especificação de Gestão de Logs

2.3.1 Triggers de suporte à gestão de logs

Relativamente à gestão de logs optámos por ter uma única tabela onde devem ser guardados o email de quem executou o comando, o nome da tabela afetada, o tipo de comando executado (insert, update, delete, select), a linha correspondente antes da execução do comando (não aplicável para o insert), a nova linha/ resultado obtido (não aplicável para o delete), a data de execução, um identificador único e um 'tinyint' designado por 'exportado'.

A opção de uma única tabela facilita a consulta principalmente em informações relativas a datas ou utilizadores em específico e diminui drasticamente a possibilidade de ocorrência de erros na migração, uma vez que é migrada apenas uma tabela.

Para resolver a questão da exportação incremental deve ser usado o campo "exportado" da tabela de logs, que deve estar a zero (por default) quando uma nova linha é inserida e assim que essa mesma linha é migrada com sucesso deve ser alterado o valor para '1', garantindo assim que não há perda de informações nem duplicados.

Optámos pela seguinte metodologia relativamente às seguintes operações:

- **Update** - Deve ser guardada a linha a alterar, antes e depois da alteração efetuada para permitir uma mais fácil identificação dos valores alterados. Esta opção também tem a vantagem de permitir a identificação de uso abusivo de permissões por parte de determinados utilizadores.
- **Delete** - Deve ser guardado o valor anterior, isto é, a linha que foi apagada.
- **Insert** - Deve ser guardada a nova linha para se saber que tipo de informações foram inseridas na base de dados.
- **Select** - Deve ser executado sob a forma de um stored procedure que guarda o resultado obtido, para monitorizar as informações visualizadas por uma determinada pessoa.

Nome Trigger	Tabela	Tipo de Operação (I,U,D)	Evento (A, B)	Notas (apenas indicar aquilo que não seja óbvio)
insertCultura	cultura	I	A	
insertVariaveis	variaveis	I	A	
insertVariaveisMedidas	variaveis_medidas	I	A	
insertMedicoes	medicoes	I	A	
insertInvestigador	investigador	I	A	
insertMedicoesLuminosidade	medicoes_luminosidade	I	A	
insertMedicoesTemperatura	medicoes_temperatura	I	A	
insertSistema	sistema	I	A	
updateCultura	cultura	U	A	
updateVariaveis	variáveis	U	A	
updateVariaveisMedidas	variaveis_medidas	U	A	
updateMedicoes	medicoes	U	A	
updateInvestigador	investigador	U	A	
updateSistema	sistema	U	A	
updateMedicoesTemperatura	medicoes_temperatura	U	A	
updateMedicoesLuminosidade	medicoes_luminosidade	U	A	
deleteCultura	cultura	D	A	

deleteVariaveis	variaveis	D	A	
deleteVariaveisMedidas	variaveis_medidas	D	A	
deleteMedicoes	medicoes	D	A	
deleteInvestigador	investigador	D	A	
deleteSistema	sistema	D	A	
deleteMedicoesTemperatura	medicoes_temperatura	D	A	
deleteMedicoesLuminosidade	medicoes_luminosidade	D	A	

2.3.2 Stored Procedures de suporte à gestão de logs

Nome Procedimento	Parâmetr os Entrada	Parâmetro s Saída	Muito breve descrição
selectMedicoes	Condição	Tabela Medições conforme condição	Este stored procedure devolve a tabela de medições filtrada de acordo com a condição recebida como parâmetro.

2.4 Avaliação da especificação do próprio grupo Gestão de Logs

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Boa

Justificação:

Foram contemplados todos os tipos de utilizadores que poderiam ter papéis de escrita ou leitura nas respetivas bases de dados e foram-lhe atribuídas as permissões de acordo com o pedido no enunciado.

Contudo faltou mencionar na especificação que no stored procedure 'add_User' devia ser contemplado o caso de o novo utilizador a inserir ser do tipo investigador, para se proceder ao seu registo na tabela de investigadores.

É uma solução robusta na medida em que assegura que a questão da segurança e integridade dos dados é assegurada pelos mecanismos de autenticação e privilégios dos utilizadores.

2.5 Implementação Gestão de Logs

2.5.1 Utilizadores implementados

Tabela	Tipo de Utilizador				
	Investigador	Administrador Aplicação	Sensor Temperatura	Sensor Luz	phpUser
sistema	L	E, L	-	-	-
variaveis	E, L	E, L	-	-	-
investigador	E, L	E, L	-	-	-
medicoes	E, L	E, L	-	-	-
variaveis_medidas	E, L	E, L	-	-	-
medicao_luminosidade	L	E, L	-	E	-
medicao_temperatura	L	E, L	E	-	-
cultura	E, L	E, L	-	-	-
logs	-	-	-	-	-
mysql.users	-	E, L	-	-	-
Stored Proc.					
add_User	-	X	-	-	-
init	-	X	-	-	-
selectMedicoes	X	X	-	-	-
alterarLimitesTemperatura	-	X	-	-	-
alterarLimitesLuz	-	X	-	-	-

deleteUser	-	X	-	-	-
alterarValorMedi do	X	X	-	-	-
apagarMedicao	X	X	-	-	-
apagarCultura	X	X	-	-	-
selectDadosNaoMi grados	-	-	-	-	X
updateMigrados	-	-	-	-	X
criarRoles	-	X	-	-	-
criarPrivilegios	-	X	-	-	-
criarPrivilegios Execute	-	X	-	-	-
criarPrivilegios	-	X	-	-	-

2.5.2 Lista de Triggers

Lista de Triggers (para cada trigger assinalar com x em célula correspondente)

	Implemen tado de Acordo com Especifi cado	Implemen tado mas diferent e de Especifi cado	Não Implemen tado	Não Especifi cado (criado de novo)
insertCultura	X			
insertVariaveis	X			
insertVariaveisMed idas	X			
insertMedicoes	X			
insertInvestigador	X			
insertMedicoesLumi nosidade	X			
insertSistema	X			
updateCultura	X			
updateVariaveis	X			
updateVariaveisMed idas	X			
updateMedicoes	X			
updateInvestigador	X			
updateSistema	X			
updateMedicoesTemp eratura	X			
updateMedicoesLumi nosidade	X			
deleteCultura	X			
deleteVariaveis	X			
deleteVariaveisMed idas	X			
deleteMedicoes	X			
deleteInvestigador	X			
deleteSistema	X			
deleteMedicoesTemp eratura	X			
deleteMedicoesLumi nosidade	X			

2.5.3 Triggers Implementados

1. Nome Trigger: **insertCultura**

Este stored procedure insere na tabela de logs uma linha relativa à operação de insert feita sobre a tabela cultura.

```
INSERT into logs VALUES (null, CURRENT_USER, "cultura",  
"INSERT", "Não Aplicável", CONCAT("IdCultura", ":", "  
new.IdCultura, " NomeCultura", ":", " new.NomeCultura, "  
DescricaoCultura", ":", " new.DescricaoCultura, "  
EmailInvestigador", ":", " new.EmailInvestigador), NOW(),0)
```

2. Nome Trigger: **insertVariaveis**

Este stored procedure insere na tabela de logs uma linha relativa à operação de insert feita sobre a tabela variáveis.

```
INSERT into logs VALUES (null, CURRENT_USER, "variaveis",  
"INSERT", "Não Aplicável", CONCAT("IDVariavel", ":", "  
new.IDVariavel, " NomeVariavel", ":", " new.NomeVariavel),  
NOW(),0)
```

3. Nome Trigger: **insertVariaveisMedidas**

Este stored procedure insere na tabela de logs uma linha relativa à operação de insert feita sobre a tabela variáveis medidas.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"variaveis_medidas", "INSERT", "Não Aplicável",  
CONCAT("IDVariavel", ":", " new.IDVariavel, " IDCultura",  
":", " new.IDCultura, " LimiteInferior", ":", "  
new.LimiteInferior, " LimiteSuperior", ":", "  
new.LimiteSuperior, " IdVariaveisMedidas", ":", "  
new.IdVariaveisMedidas), NOW(),0)
```

4. Nome Trigger: **insertMedicoes**

Este stored procedure insere na tabela de logs uma linha relativa à operação de insert feita sobre a tabela medições.

```
INSERT into logs VALUES (null, CURRENT_USER, "medicoes",  
"INSERT", "Não Aplicável", CONCAT("NumeroMedicao", ": ",  
new.NumeroMedicao, " DataHoraMedicao", ": ",  
new.DataHoraMedicao, " ValorMedicao", ": ",  
new.ValorMedicao, " IdVariaveisMedidas", ": ",  
new.IdVariaveisMedidas), NOW(),0)
```

5. Nome Trigger: **insertInvestigador**

Este stored procedure insere na tabela de logs uma linha relativa à operação de insert feita sobre a tabela investigador.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"investigador", "INSERT", "Não Aplicável", CONCAT("Email",  
": ", new.Email, " NomeInvestigador", ": ",  
new.NomeInvestigador, " CategoriaProfissional", ": ",  
new.CategoriaProfissional), NOW(),0)
```

6. Nome Trigger: **insertMedicoesLuminosidade**

Este stored procedure insere na tabela de logs uma linha relativa à operação de insert feita sobre a tabela medições luminosidade.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"medicoes_luminosidade", "INSERT", "Não Aplicável",  
CONCAT("DataHoraMedicao", ": ", new.DataHoraMedicao, "  
ValorMedicaoLuminosidade", ": ",  
new.ValorMedicaoLuminosidade, " IDMedicao", ": ",  
new.IDMedicao), NOW(),0)
```

7. Nome Trigger: **insertMedicoesTemperatura**

Este stored procedure insere na tabela de logs uma linha relativa à operação de insert feita sobre a tabela medições temperatura.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"medicoes_temperatura", "INSERT", "Não Aplicável",  
CONCAT("DataHoraMedicao", ": ", new.DataHoraMedicao, "  
ValorMedicaoTemperatura", ": ", new.ValorMedicaoTemperatura  
, " IDMedicao", ": ", new.IDMedicao), NOW(),0)
```

8. Nome Trigger: **insertSistema**

Este stored procedure insere na tabela de logs uma linha relativa à operação de insert feita sobre a tabela sistema.

```
INSERT into logs VALUES (null, CURRENT_USER, "sistema",
"INSERT", "Não Aplicável",
CONCAT("LimiteInferiorTemperatura", ": ",
new.LimiteInferiorTemperatura, "
LimiteSuperiorTemperatura", ": ",
new.LimiteSuperiorTemperatura, " LimiteInferiorLuz", ": ",
", new.LimiteInferiorLuz, " LimiteSuperiorLuz", ": ",
new.LimiteSuperiorLuz), NOW(), 0)
```

9. Nome Trigger: **updateCultura**

Este stored procedure insere na tabela de logs uma linha relativa à operação de update feita sobre a tabela cultura.

```
INSERT into logs VALUES (null, CURRENT_USER, "cultura",
"UPDATE", CONCAT("IdCultura", ": ", old.IdCultura, "
NomeCultura", ": ", old.NomeCultura, " DescricaoCultura",
": ", old.DescricaoCultura, " EmailInvestigador", ": ",
old.EmailInvestigador), CONCAT("IdCultura", ": ",
new.IdCultura, " NomeCultura", ": ", new.NomeCultura, "
DescricaoCultura", ": ", new.DescricaoCultura, "
EmailInvestigador", ": ", new.EmailInvestigador), NOW(), 0)
```

10. Nome Trigger: **updateVariaveis**

Este stored procedure insere na tabela de logs uma linha relativa à operação de update feita sobre a tabela variáveis.

```
INSERT into logs VALUES (null, CURRENT_USER, "variaveis",
"UPDATE", CONCAT("IDVariavel", ": ", old.IDVariavel, "
NomeVariavel", ": ", old.NomeVariavel),
CONCAT("IDVariavel", ": ", new.IDVariavel, "
NomeVariavel", ": ", new.NomeVariavel), NOW(), 0)
```

11. Nome Trigger: **updateVariaveisMedidas**

Este stored procedure insere na tabela de logs uma linha relativa à operação de update feita sobre a tabela variáveis medidas.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"variaveis_medidas", "UPDATE", CONCAT("IDVariavel", ":",  
old.IDVariavel, " IDCultura", ":", old.IDCultura, "  
LimiteInferior", ":", old.LimiteInferior, "  
LimiteSuperior", ":", old.LimiteSuperior, "  
IdVariaveisMedidas", ":", old.IdVariaveisMedidas),  
CONCAT("IDVariavel", ":", new.IDVariavel, " IDCultura",  
":", new.IDCultura, " LimiteInferior", ":",  
new.LimiteInferior, " LimiteSuperior", ":",  
new.LimiteSuperior, " IdVariaveisMedidas", ":",  
new.IdVariaveisMedidas), NOW(), 0)
```

12. Nome Trigger: **updateMedicoes**

Este stored procedure insere na tabela de logs uma linha relativa à operação de update feita sobre a tabela medições.

```
INSERT into logs VALUES (null, CURRENT_USER, "medicoes",  
"UPDATE", CONCAT("NumeroMedicao", ":", old.NumeroMedicao,  
" DataHoraMedicao", ":", old.DataHoraMedicao, "  
ValorMedicao", ":", old.ValorMedicao, "  
IdVariaveisMedidas", ":", old.IdVariaveisMedidas),  
CONCAT("NumeroMedicao", ":", new.NumeroMedicao, "  
DataHoraMedicao", ":", new.DataHoraMedicao, "  
ValorMedicao", ":", new.ValorMedicao, "  
IdVariaveisMedidas", ":", new.IdVariaveisMedidas), NOW(),  
0)
```

13. Nome Trigger: **updateInvestigador**

Este stored procedure insere na tabela de logs uma linha relativa à operação de update feita sobre a tabela investigador.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"investigador", "UPDATE", CONCAT("Email", ":", old.Email,  
" NomeInvestigador", ":", old.NomeInvestigador, "  
CategoriaProfissional", ":", old.CategoriaProfissional),  
CONCAT("Email", ":", new.Email, " NomeInvestigador", ":",  
", new.NomeInvestigador, " CategoriaProfissional", ":",  
new.CategoriaProfissional), NOW(), 0)
```

14. Nome Trigger: **updateSistema**

Este stored procedure insere na tabela de logs uma linha relativa à operação de update feita sobre a tabela sistema.

```
INSERT into logs VALUES (null, CURRENT_USER, "sistema",  
"UPDATE", CONCAT("LimiteInferiorTemperatura", ": ",  
old.LimiteInferiorTemperatura, "  
LimiteSuperiorTemperatura", ": ",  
old.LimiteSuperiorTemperatura, " LimiteInferiorLuz", ": ",  
old.LimiteInferiorLuz, " LimiteSuperiorLuz", ": ",  
old.LimiteSuperiorLuz), CONCAT("LimiteInferiorTemperatura",  
": ", new.LimiteInferiorTemperatura, "  
LimiteSuperiorTemperatura", ": ",  
new.LimiteSuperiorTemperatura, " LimiteInferiorLuz", ": ",  
new.LimiteInferiorLuz, " LimiteSuperiorLuz", ": ",  
new.LimiteSuperiorLuz), NOW(), 0)
```

15. Nome Trigger: **updateMedicoesTemperatura**

Este stored procedure insere na tabela de logs uma linha relativa à operação de update feita sobre a tabela medições temperatura.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"medicoes_temperatura", "UPDATE", CONCAT("DataHoraMedicao",  
": ", old.DataHoraMedicao, " ValorMedicaoTemperatura", ":  
", old.ValorMedicaoTemperatura, " IDMedicao", ": ",  
old.IDMedicao), CONCAT("DataHoraMedicao", ": ",  
new.DataHoraMedicao, " ValorMedicaoTemperatura", ": ",  
new.ValorMedicaoTemperatura, " IDMedicao", ": ",  
new.IDMedicao), NOW(), 0)
```

16. Nome Trigger: **updateMedicoesLuminosidade**

Este stored procedure insere na tabela de logs uma linha relativa à operação de update feita sobre a tabela medições luminosidade.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"medicoes_luminosidade", "UPDATE",  
CONCAT("DataHoraMedicao", ": ", old.DataHoraMedicao, "  
ValorMedicaoLuminosidade", ": ",  
old.ValorMedicaoLuminosidade, " IDMedicao", ": ",  
old.IDMedicao), CONCAT("DataHoraMedicao", ": ",  
new.DataHoraMedicao, " ValorMedicaoLuminosidade", ": ",  
new.ValorMedicaoLuminosidade, " IDMedicao", ": ",  
new.IDMedicao), NOW(), 0)
```

17. Nome Trigger: **deleteCultura**

Este stored procedure insere na tabela de logs uma linha relativa à operação de delete feita sobre a tabela cultura.

```
INSERT into logs VALUES (null, CURRENT_USER, "cultura",  
"DELETE", CONCAT("IdCultura", ": ", old.IdCultura, "  
NomeCultura", ": ", old.NomeCultura, " DescricaoCultura",  
": ", old.DescricaoCultura, " EmailInvestigador", ": ",  
old.EmailInvestigador), "Linha Eliminada", NOW(), 0)
```

18. Nome Trigger: **deleteVariaveis**

Este stored procedure insere na tabela de logs uma linha relativa à operação de delete feita sobre a tabela variáveis.

```
INSERT into logs VALUES (null, CURRENT_USER, "variaveis",  
"DELETE", CONCAT("IDVariavel", ": ", old.IDVariavel, "  
NomeVariavel", ": ", old.NomeVariavel), "Linha Eliminada",  
NOW(), 0)
```

19. Nome Trigger: **deleteVariaveisMedidas**

Este stored procedure insere na tabela de logs uma linha relativa à operação de delete feita sobre a tabela variáveis medidas.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"variaveis_medidas", "DELETE", CONCAT("IDVariavel", ": ",  
old.IDVariavel, " IDCultura", ": ", old.IDCultura, "  
LimiteInferior", ": ", old.LimiteInferior, "  
LimiteSuperior", ": ", old.LimiteSuperior, "  
IdVariaveisMedidas", ": ", old.IdVariaveisMedidas), "Linha  
Eliminada", NOW(), 0)
```

20. Nome Trigger: **deleteMedicoes**

Este stored procedure insere na tabela de logs uma linha relativa à operação de delete feita sobre a tabela medições.

```
INSERT into logs VALUES (null, CURRENT_USER, "medicoes",  
"DELETE", CONCAT("NumeroMedicao", ": ", old.NumeroMedicao,  
" DataHoraMedicao", ": ", old.DataHoraMedicao, "  
ValorMedicao", ": ", old.ValorMedicao, "  
IdVariaveisMedidas", ": ", old.IdVariaveisMedidas), "Linha  
Eliminada", NOW(), 0)
```


21. Nome Trigger: **deleteInvestigador**

Este stored procedure insere na tabela de logs uma linha relativa à operação de delete feita sobre a tabela investigador.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"investigador", "DELETE", CONCAT("Email", ": ", old.Email,  
" NomeInvestigador", ": ", old.NomeInvestigador, "  
CategoriaProfissional", ": ", old.CategoriaProfissional),  
"Linha Eliminada", NOW(), 0)
```

22. Nome Trigger: **deleteSistema**

Este stored procedure insere na tabela de logs uma linha relativa à operação de delete feita sobre a tabela sistema.

```
INSERT into logs VALUES (null, CURRENT_USER, "sistema",  
"DELETE", CONCAT("LimiteInferiorTemperatura", ": ",  
old.LimiteInferiorTemperatura, "  
LimiteSuperiorTemperatura", ": ",  
old.LimiteSuperiorTemperatura, " LimiteInferiorLuz", ": ",  
old.LimiteInferiorLuz, " LimiteSuperiorLuz", ": ",  
old.LimiteSuperiorLuz), "Linha Eliminada", NOW(), 0)
```

23. Nome Trigger: **deleteMedicoesTemperatura**

Este stored procedure insere na tabela de logs uma linha relativa à operação de delete feita sobre a tabela medições temperatura.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"medicoes_temperatura", "DELETE", CONCAT("DataHoraMedicao",  
": ", old.DataHoraMedicao, " ValorMedicaoTemperatura", ":  
", old.ValorMedicaoTemperatura, " IDMedicao", ": ",  
old.IDMedicao), "Linha Eliminada", NOW(), 0)
```

24. Nome Trigger: **deleteMedicoesLuminosidade**

Este stored procedure insere na tabela de logs uma linha relativa à operação de delete feita sobre a tabela medições luminosidade.

```
INSERT into logs VALUES (null, CURRENT_USER,  
"medicoes_luminosidade", "DELETE",  
CONCAT("DataHoraMedicao", ": ", old.DataHoraMedicao, "  
ValorMedicaoLuminosidade", ": ",  
old.ValorMedicaoLuminosidade, " IDMedicao", ": ",  
old.IDMedicao), "Linha Eliminada", NOW(), 0)
```

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)
add User	X			
init		X		
selectMedicoes	X			
alterarLimitesTemp	X			
alterarLimitesLuz	X			
deleteUser	X			
alterarValorMedido	X			
apagarMedicao	X			
apagarCultura	X			
selectDadosNaoMigrados	X			
updateMigrados	X			
criarRoles	X			
criarPrivilegios	X			
criarPrivilegios Execute		X		
createPhpUser	X			

2.5.4 Stored Procedures Implementados

1. Nome SP: **add_User**

Este stored procedure permite criar um novo utilizador, inserindo-o na tabela mysql.user e na tabela investigador, caso este seja um investigador.

BEGIN

```
IF (SELECT EXISTS( SELECT * FROM mysql.user WHERE  
`email` = var_email))=0 THEN
```

```
DROP USER IF EXISTS '@localhost;
```

```
SET @sql := CONCAT('CREATE USER ', QUOTE(var_nome),  
' IDENTIFIED BY ', QUOTE(var_password));
```

```
PREPARE statement FROM @sql;  
EXECUTE statement;
```

```
SET @sql := CONCAT('GRANT ', var_role, ' TO ',  
QUOTE(var_nome));
```

```
PREPARE statement FROM @sql;  
EXECUTE statement;
```

```
SET @sql := CONCAT('SET DEFAULT ROLE ', var_role ,'  
FOR ', QUOTE(var_nome));
```

```
PREPARE statement FROM @sql;  
EXECUTE statement;
```

```
SET @sql := CONCAT('UPDATE mysql.user SET email =  
, QUOTE(var_email), ' WHERE User=', QUOTE(var_nome));
```

```
PREPARE statement FROM @sql;  
EXECUTE statement;
```

```
SET @sql := CONCAT('INSERT INTO investigador SET  
email=', QUOTE(var_email), ', nomeinvestigador=',  
QUOTE(var_nome), ', categoriaprofissional=',  
QUOTE(var_categoria_profissional));
```

```
PREPARE statement FROM @sql;  
EXECUTE statement;
```

```
DEALLOCATE PREPARE statement;  
FLUSH PRIVILEGES;
```

```
END IF;
```

END

2. Nome SP: **init**

Este stored procedure chama a execução o stored procedure `criarRoles`, `criarPrivilegios` e `criarPrivilegiosExecute`.

```
BEGIN
```

```
    CALL criarRoles;
```

```
    CALL criarPrivilegios;
```

```
    CALL criarPrivilegiosExecute;
```

```
    ALTER TABLE mysql.user ADD COLUMN IF NOT EXISTS email  
    varchar(100) UNIQUE;
```

```
END
```

3. Nome SP: **selectMedicoes**

Este stored procedure devolve a tabela de medições filtrada de acordo com a condição recebida como parâmetro.

```
BEGIN
```

```
    IF var_condicao = "" THEN  
        SET var_condicao = "1=1";  
        SELECT "Dentro IF";  
    END IF;
```

```
    SET @sql := CONCAT('INSERT INTO estufa.logs VALUES  
(null, CURRENT_USER, "medicoes", "SELECT", "Não Aplicável",  
"', var_condicao, '", NOW(), 0)');  
    PREPARE statement FROM @sql;  
    EXECUTE statement;
```

```
    SET @sql := CONCAT('SELECT * FROM estufa.medicoes  
WHERE ', var_condicao);  
    PREPARE statement FROM @sql;  
    EXECUTE statement;
```

```
END
```

4. Nome SP: **alterarLimitesTemp**

Este stored procedure altera os valores superior e inferior de temperatura da tabela `sistema` dado como parâmetro de entrada o `'limiteInferior'` e o `'limiteSuperior'`.

```
IF EXISTS(select * from sistema) THEN  
    UPDATE sistema SET sistema.LimiteInferiorTemperatura =
```

```

limiteInferior, sistema.LimiteSuperiorTemperatura =
limiteSuperior;
ELSE
    INSERT INTO sistema VALUES (limiteInferior,
limiteSuperior, 0, 0);
END IF

```

5. Nome SP: **alterarLimitesLuz**

Este stored procedure altera os valores superior e inferior de luminosidade da tabela sistema dado como parâmetro de entrada o 'limiteInferior' e o 'limiteSuperior'.

```

IF EXISTS(select * from sistema) THEN
    UPDATE sistema SET LimiteInferiorLuz = limiteInferior,
LimiteSuperiorLuz = limiteSuperior WHERE 1;
ELSE
    INSERT INTO sistema VALUES (0, 0, limiteInferior,
limiteSuperior);
END IF

```

6. Nome SP: **deleteUser**

Este stored procedure permite apagar um utilizador dado o seu email.

```

BEGIN

SET @sql := CONCAT('SELECT User INTO @user FROM mysql.user
WHERE email = ', QUOTE(var_email));
PREPARE statement FROM @sql;
EXECUTE statement;

SET @sql := CONCAT('DROP USER IF EXISTS ', @user);
PREPARE statement FROM @sql;
EXECUTE statement;

END

```

7. Nome SP: **alterarValorMedido**

Este stored procedure permite alterar o valor de uma dada medição, sendo que deve ser passado como parâmetro o 'IdVar' que representa o id da medição a alterar e o 'novoValor' da mesma.

```

UPDATE medicoes SET medicoes.ValorMedicao = novoValor WHERE
medicoes.NumeroMedicao = IdVar

```

8. Nome SP: **apagarMedicao**

Este stored procedure permite apagar uma medição dado o seu id.

```
DELETE FROM `medicoes` WHERE medicoes.NumeroMedicao=id
```

9. Nome SP: **apagarCultura**

Este stored procedure permite apagar uma cultura dado o seu id.

```
DELETE FROM `cultura` WHERE cultura.IDCultura=id
```

10. Nome SP: **updateMigrados**

Este stored procedure marca como migradas as linhas da tabela de logs cujo id é igual ou inferior ao id dado como parâmetro.

```
UPDATE logs SET logs.exportado=1 WHERE logs.logId<=endID
```

11. Nome SP: **selectDadosNaoMigrados**

Este stored procedure devolve uma tabela com as linhas da tabela de logs que ainda não foram migradas.

```
SELECT * FROM logs WHERE logs.exportado=0
```

12. Nome SP: **criarRoles**

Este stored procedure cria os roles investigador, administrador da aplicação, sensor luminosidade, sensor temperatura e phpUser.

```
CREATE ROLE IF NOT EXISTS investigador, administrador,  
sensorLuminosidade, sensorTemperatura, phpUser
```

13. Nome SP: **criarPrivilegios**

Este stored procedure cria os privilégios de leitura e escrita sobre as tabelas da base de dados.

```
BEGIN
```

```
GRANT SELECT ON estufa.logs TO phpUser;
```

```
GRANT EXECUTE ON PROCEDURE estufa.selectDadosNaoMigrados TO  
phpUser;
```

```
GRANT EXECUTE ON PROCEDURE estufa.updateMigrados TO  
phpUser;
```

```
GRANT SELECT ON estufa.medicoes_luminosidade TO  
investigador;
```

```
GRANT SELECT ON estufa.medicoes_temperatura TO  
investigador;
```

```

GRANT      INSERT      ON      estufa.medicoes_temperatura      TO
sensorTemperatura;
GRANT      INSERT      ON      estufa.medicoes_luminosidade      TO
sensorLuminosidade;

GRANT SELECT,INSERT, UPDATE, DELETE ON estufa.sistema TO
administrador,investigador;
GRANT SELECT,INSERT, UPDATE, DELETE ON estufa.variaveis TO
administrador,investigador;
GRANT SELECT,INSERT, UPDATE, DELETE ON estufa.investigador
TO administrador,investigador;
GRANT SELECT,INSERT, UPDATE, DELETE ON estufa.medicoes TO
administrador,investigador;
GRANT SELECT,INSERT, UPDATE, DELETE ON estufa.sistema TO
administrador,investigador;
GRANT SELECT,INSERT, UPDATE, DELETE ON estufa.cultura TO
administrador,investigador;
GRANT SELECT,INSERT, UPDATE, DELETE ON estufa.sistema TO
administrador,investigador;

GRANT      SELECT,INSERT,      UPDATE,      DELETE      ON
estufa.medicoes_temperatura TO administrador;
GRANT      SELECT,INSERT,      UPDATE,      DELETE      ON
estufa.medicoes_luminosidade TO administrador;

END

```

14. Nome SP: **criarPrivilegiosExecute**

Este stored procedure cria os privilégios de execução sobre os stored procedures especificados.

```
BEGIN
```

```

GRANT EXECUTE ON PROCEDURE estufa.apagarCultura TO
administrador,investigador;
GRANT EXECUTE ON PROCEDURE estufa.apagarMedicao TO
administrador,investigador;
GRANT EXECUTE ON PROCEDURE estufa.addUser TO administrador;
GRANT EXECUTE ON PROCEDURE estufa.alterarLimitesLuz TO
administrador;
GRANT EXECUTE ON PROCEDURE estufa.alterarLimitesTemperatura
TO administrador;
GRANT EXECUTE ON PROCEDURE estufa.alterarValorMedido TO
administrador,investigador;
GRANT EXECUTE ON PROCEDURE estufa.createPhpUser TO
administrador;
GRANT EXECUTE ON PROCEDURE estufa.criarPrivilegios TO

```

```

administrador;
GRANT EXECUTE ON PROCEDURE estufa.criarRoles TO
administrador;
GRANT EXECUTE ON PROCEDURE estufa.deleteUser TO
administrador;
GRANT EXECUTE ON PROCEDURE estufa.init TO administrador;

END

```

15. Nome SP: **createPhpUser**

Este stored procedure cria um utilizador com o role de phpUser para realizar as operações sobre a base de dados relacionadas com a migração.

```

BEGIN
DROP USER IF EXISTS '@localhost;
SET @sql := CONCAT('CREATE USER ', 'php', ' IDENTIFIED BY
', QUOTE('php'));
    PREPARE statement FROM @sql;
    EXECUTE statement;

    SET @sql := CONCAT('GRANT ', 'phpUser', ' TO ',
QUOTE('php'));
    PREPARE statement FROM @sql;
    EXECUTE statement;

    SET @sql := CONCAT('SET DEFAULT ROLE ', 'phpUser' ,'
FOR ', QUOTE('php'));
    PREPARE statement FROM @sql;
    EXECUTE statement;

    DEALLOCATE PREPARE statement;
    FLUSH PRIVILEGES;
END

```


2.6 Especificação de Migração entre Bases de Dados

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

auditor logs	
logId : int(11)	PK
username : varchar(80)	M
nomeTabela : varchar(200)	M
comandoUsado : varchar(200)	M
linhaAnterior : varchar(200)	M
resultado : varchar(200)	M
dataComando : varchar(200)	M

Na base de dados destino deve ser criado o role auditor, ao qual devem ser concedidas permissões de leitura (Select), e o role php que deverá ter permissões de escrita (Insert) na tabela de logs.

As operações de escrita devem ser feitas utilizando um Stored Procedure e as operações de leitura utilizando uma interface dedicada a esse fim.

2.6.2 Forma de Migração Especificada

A migração dos logs é feita recorrendo a ficheiros PHP que têm como função exportar a tabela de logs para a base de dados do auditor de forma incremental e periódica. De modo a cumprir o primeiro requisito, optámos por utilizar a coluna "exportado" da tabela de logs, sendo que sempre que é inserida uma nova informação na tabela o 'tinyint' desta coluna deve ficar a zero (por default). Assim, sempre que for feita uma nova migração, quer seja a pedido ou automática, a tabela de logs deve ser consultada e serão apenas preparados para exportar os logs que tenham o campo "exportado" a zero. Isto não garante a não inserção de dados duplicados por si só. Para isto comparamos os id's dos dados previamente preparados com os id's da base de dados de destino, sendo que todos os dados preparados que tenham o seu id já presente na base de dados são ignorados (não são inseridos novamente). Os restantes dados são inseridos.

Uma vez terminada a migração deve ser feita uma nova conexão à base de dados de origem com o objetivo de marcar com um '1' todos os dados migrados.

Esta solução é bastante robusta, na medida em que salvaguarda a integridade da informação em caso de ocorrência de falhas em cada uma das seguintes situações:

- Caso haja uma falha no pedido das migrações ou na migração automática, nenhum dado é automaticamente marcado como enviado pelo que, mal surja um novo pedido todos os dados serão migrados sem que haja perdas.

- Caso haja uma falha ao inserir os dados na base de dados destino, os dados nunca são marcados como enviados na base de dados de origem pelo que na próxima migração há o conhecimento que faltam migrar esses mesmos dados.

- No caso de haver falhas na marcação dos dados enviados, quando ocorrer uma nova migração esta falha será detetada

devido ao id da base de dados origem já estar na base de dados destino e esses mesmos dados serão marcados como migrados sem que haja por isso qualquer tipo de duplicação de informação.

Para que o auditor possa consultar os dados e interagir com a sua base de dados deve ser criado um outro ficheiro `guiAuditor.php` que facilita a interação com a base de dados e evita a necessidade de utilização de ferramentas mais complexas como o `'phpmyadmin'`.

A migração deve ser feita todos os dias às 2:00, sendo que esta periodicidade deve ser assegurada recorrendo ao Windows task scheduler. Nesta ferramenta do Windows, deve ser criada uma nova `'task'` que é executada com a periodicidade anteriormente referida e que tem como função executar um ficheiro `'.bat'`, que por sua vez corre o ficheiro `makeMigrations.php`.

Relativamente à questão da privacidade dos dados, é assegurada através do `'POST'`, que ao contrário do `'GET'` não mostra os dados no URL.

2.6.3 Utilizadores especificados

Tabela	Tipo de Utilizador	
	Auditor	PhpUser
logs	L	E

2.6.4 Triggers de suporte à migração de dados especificados

2.6.5 Stored Procedures de suporte à migração de dados especificados

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	BD (Origem ou Destino)	Muito breve descrição
selectDadosNaoMigrados	-	-	Origem	Este SP deve retornar uma tabela de todas as linhas que ainda não foram migradas.
updateMigrados	Inteiro - startID	-	Origem	Este SP deve marcar como migradas todas as linhas que tenham id igual ou superior ao dado como parâmetro.
insertLog	-	-	Destino	Este SP serve para inserir novos logs na base de dados do auditor.
selectId	-	-	Destino	Este SP devolve o maior ID da tabela logs.

2.6.6 Eventos de suporte à migração de dados especificados

Nome Evento	Local Execução (Origem ou Destino, ou Sistema Operativo)	Muito breve descrição
migrarResultados	Sistema Operativo	Este evento deve ocorrer diariamente às 2:00, ou caso a periodicidade seja alterada, deve ser alterado de acordo com o especificado aquando dessa alteração.

2.6.7 PHP de suporte à migração de dados especificado

Devem existir ficheiros PHP que quando executados assegurem a migração.

Esta é composta pelas seguintes fases:

- 1- Ir buscar os dados à base de dados origem, recorrendo ao SP apropriado e em seguida retorná-los em JSON.
- 2- Caso a fase anterior tenha retornado alguns dados, devemos colocá-los na base de dados do auditor/destino.
- 3- Devem ser marcados como migrados os dados na base de dados origem, recorrendo ao SP apropriado, como especificado nas secções anteriores.

De forma a assegurar a migração periódica deverá ser criado um ficheiro ``.bat'` cuja função será executar os ficheiros PHP responsáveis pelas fases anteriormente descritas com a periodicidade anteriormente definida. Este mesmo ficheiro será executado através de uma task do Windows Task Scheduler.

Por parte do administrador da aplicação deve ser possível executar as migrações a pedido recorrendo diretamente aos ficheiros PHP através de uma interface.

Adicionalmente é necessário implementar uma interface que permita ao auditor correr as migrações a pedido e executar comandos do tipo select sobre a sua base de dados.

2.7 Avaliação das especificações do próprio grupo Migração

Qualidade (Frac, Razoável, Boa ou Muito Boa): Muito Boa

Justificação:

A gestão de logs é feita de uma forma simples recorrendo a uma única tabela, o que facilita imenso as operações de consulta por parte do auditor.

Os possíveis problemas decorrentes da migração foram previamente identificados, sendo que os mesmos são ultrapassados seguindo a seguinte metodologia:

- Caso haja uma falha no pedido das migrações ou na migração automática, nenhum dado é automaticamente marcado como enviado pelo que, mal surja um novo pedido todos os dados serão migrados sem que haja perdas.

- Caso haja uma falha ao inserir os dados na base de dados destino, os dados nunca são marcados como enviados na base de dados de origem pelo que na próxima migração há o conhecimento que faltam migrar esses mesmos dados.

- No caso de haver falhas na marcação dos dados enviados, quando ocorrer uma nova migração esta falha será detetada devido ao id da base de dados origem já estar na base de dados destino e esses mesmos dados serão marcados como migrados sem que haja por isso qualquer tipo de duplicação de informação.

2.8 Implementação da Migração de Dados

2.8.1 Utilizadores Implementado

Tabela	Tipo de Utilizador	
	Auditor	PhpUser
logs	L	E

2.8.2 Lista Triggers

Lista de Triggers (para cada trigger assinalar com x em célula correspondente)				
	Implementa do de Acordo com Especifica do	Implementa do mas diferente de Especifica do	Não Implementa do	Não Especifica do (criado de novo)
Nome Trigger (tal como especifica do)				

2.8.3 Triggers Implementados

<p>1. Nome Trigger: _____ // <i>Breve Descrição</i> <i>Código</i></p>

2.8.4 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)
selectMedicoes		X		

2.8.5 Stored Procedures Implementados

1. Nome SP: **selectMedicoes**

Este stored procedure devolve a tabela de medições filtrada de acordo com a condição recebida como parâmetro.

```
BEGIN
```

```
    IF var_condicao = "" THEN  
        SET var_condicao = "1=1";  
        SELECT "Dentro IF";  
    END IF;
```

```
    SET @sql := CONCAT('INSERT INTO estufa.logs VALUES  
(null, CURRENT_USER, "medicoes", "SELECT", "Não Aplicável",  
'', var_condicao, '', NOW(), 0)');  
    PREPARE statement FROM @sql;  
    EXECUTE statement;
```

```
    SET @sql := CONCAT('SELECT * FROM estufa.medicoes  
WHERE ', var_condicao);  
    PREPARE statement FROM @sql;  
    EXECUTE statement;
```

```
END
```

2.8.6 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)
makeMigrations	X			

2.8.7 Eventos Implementados

1. Nome Evento: **makeMigrations**

Este evento faz com que os dados sejam importados para a base de dados destino com a periodicidade desejada, chamando a execução a seguinte linha de código.

```
c:\xampp\php\php.exe -f c:\xampp\htdocs\makeMigrations.php
```


2.8.8 PHP Implementado

apiGet.php

```
<?php
header("Content-Type: application/json;charset=utf-8");

    $data = get_data();
    if(empty($data)){
        echo 'No unmigrated data found';
    } else{
        echo $data;
    }

function get_data() {
    // Database credentials
    $url="localhost";
    $database="estufa";
    $username="php";
    $password="php";

    $conn = mysqli_connect($url, $username, $password,
    $database); // Connects to the mysql database. If
    unsuccessful $conn is an object that is false.
    /* change character set to utf8 */
    if (!$conn->set_charset("utf8")) {
        printf("Error loading character set utf8: %s\n",
    $conn->error);
        exit();
    }

    if (!$conn){
        die ("Connection Failed: ".$conn->connect_error);
    } // Prints a message and exits the
    current script

    $sql = "call selectDadosNaoMigrados"; // sql query
    $result = mysqli_query($conn, $sql); // Performs
    the query on the database
    $rows = array(); // Creates an empty array
    if ($result) {
        if (mysqli_num_rows($result)>0){ //
        Returns number of rows in the result set
            while($r=mysqli_fetch_assoc($result)){ //
            Returns an array that corresponds to the fetched row
                array_push($rows, $r);
            // Pushes one or more elements onto the end of array
        }
    }
}
```

```

    }
}
$result->close();
$conn->next_result();
$conn->close();           // Closes connection
return json_encode($rows); // Returns a string
                           containing the JSON representation of the supplied value.
}

```

apiPut.php

```

<?php
header("Content-Type: application/json;charset=utf-8");

//Make sure that it is a POST request.
if(strcasecmp($_SERVER['REQUEST_METHOD'], 'POST') != 0){
    throw new Exception('Request method must be POST!');
}

//Make sure that the content type of the POST request has
been set to application/json
$contentType = isset($_SERVER["CONTENT_TYPE"]) ?
trim($_SERVER["CONTENT_TYPE"]) : '';
if(strcasecmp($contentType, 'application/json') != 0){
    throw new Exception('Content type must be:
application/json');
}

$data = trim(file_get_contents("php://input"));
$result = put_data($data);
echo $result;

function put_data($data){
    $logs = json_decode($data);

    if(!is_array($logs)){
        throw new Exception('Received content contained
invalid JSON!');
    }

    // Database credentials
    $url="localhost";
    $database="auditor";
    $username="root";
    $password="";

    $conn = new mysqli($url, $username, $password,
    $database);
    /* change character set to utf8 */
    if (!$conn->set_charset("utf8")) {

```

```

        exit();
    }

    if ($conn){
        $result = mysqli_query($conn, "call selectID;");

        $rows = array();
        if ($result) {
            if (mysqli_num_rows($result)>0){
                while($r=mysqli_fetch_assoc($result)){
                    array_push($rows, $r);
                }
                $next_record_id =
intval($rows[0]["Maximo"]);
            }
            } else {
                $next_record_id = 1;
            }
        }
        $result->close();
        $conn->next_result();

        foreach ($logs as $log) {
            if ($log->logId >= $next_record_id) {

                $logId = $log->logId;
                $username = $log->username;
                $nomeTabela= $log->nomeTabela;
                $comandoUsado = $log->comandoUsado;
                $linhaAnterior = $log->linhaAnterior;
                $resultado = $log->resultado;
                $dataComando = $log->dataComando;

                $insertQuery = "INSERT INTO logs
                                VALUES ('$logId',
'$username', '$nomeTabela',
'$comandoUsado','$linhaAnterior', '$resultado',
'$dataComando')";
                $insert = mysqli_query($conn,
$insertQuery);

                $next_record_id++;
            }
        }

        mysqli_close ($conn);
    }

    return $next_record_id;
}

```

makeMigrations.php

```
<?php
    // Get data

    echo "<h1>Migrations</h1>";
    echo "<p>-----</p>";

    printCurrentTimestamp("Started migrations");

    $url = "http://localhost/apiGet.php";
    $client = curl_init($url);
    curl_setopt( $client, CURLOPT_POST, 1);
    curl_setopt( $client, CURLOPT_FOLLOWLOCATION, 1);
    curl_setopt( $client, CURLOPT_HEADER, 0);
    curl_setopt( $client, CURLOPT_RETURNTRANSFER, 1);
    $data = curl_exec($client);

    if (curl_errno($client)) { // Check if getting data
was sucessful
        die('Couldn\'t send request: ' .
curl_error($client));
    } else {
        $resultStatus = curl_getinfo($client,
CURLINFO_HTTP_CODE); // check the HTTP status code of the
request
        if ($resultStatus != 200) { // the request did
not complete as expected.
            die('Request failed: HTTP status code: ' .
$resultStatus);
        }
    }

    printCurrentTimestamp("Finished getting data. Started
posting.");

    curl_close($client);

    if(!empty(json_decode($data, true))) {
        // Put data
        $url = "http://localhost/apiPut.php";
        $put = curl_init($url);
        curl_setopt( $put, CURLOPT_POST, 1);
        curl_setopt( $put, CURLOPT_POSTFIELDS, $data);
        curl_setopt( $put, CURLOPT_HTTPHEADER,
array('Content-Type: application/json'));
        curl_setopt( $put, CURLOPT_RETURNTRANSFER, 1);
        $last_updated = curl_exec($put);
        curl_close($put);
    }
}
```

```

        printCurrentTimestamp("Finished putting data.");

        markDataAsMigrated($last_updated);

        printCurrentTimestamp("Finished migrations");

    } else {
        echo "<p>No new or unmigrated data returned from
server.</p>";
    }

    echo "-----<br> Finished
migrations...";

function markDataAsMigrated($endID) {
    printf("<p>Next updated ID: ".$endID);

    // Database credentials
    $url="localhost";
    $username="php";
    $password="php";
    $database="estufa";

    $conn = new mysqli($url, $username, $password,
$database); // Connects to the mysql database. If
unsuccessful $conn is an object that is false.

    if (!$conn){
        die ("Connection to original database
failed: Couldn't mark data as migrated. Will retry on next
migration."); // Prints a message and exits the
current script
    }

    // Change character set to utf8
    if (!$conn->set_charset("utf8")) {
        printf("<p>Error loading character set
utf8: %s\n", $conn->error);
        exit();
    }

    $sql = "call updateMigrados(".$endID.)"; //
sql string constructed. '.' concats strings
    $result = mysqli_query($conn, $sql); //
Performs the query on the database
    if($result) {
        echo "<p> Marking data as Migrated:
Success</p>";
    } else {
        printf("<p>Unable to mark data as

```

```

migrated");
    }

    $conn->next_result();
}

function printCurrentTimestamp($string) {
    $time = microtime(true);
    $dFormat = "l jS F, Y - H:i:s";
    $mSecs = $time - floor($time);
    $mSecs = substr($mSecs,1);

    echo '<br />';
    echo '<br />'.$string.' '.sprintf('%s%s',
date($dFormat), $mSecs);
}

?>

```

guiAuditor.php

```

<!DOCTYPE html>
<html>
<body>

<style>
html, body {
    font-family: sans-serif;
    margin:0;
    padding: 0;
    color: #404040;
}

h1 {
    margin: 30px 10px;
    font-size: 50px;
    text-align: center;
}

h2 {
    margin-top: 0;
    margin-bottom: 20px;
}

.row {
    display: flex;
}

button {
    background-color: #629fde;
    border: none;
}

```

```

    color: white;
    padding: 15px 32px;
    text-align: center;
    font-size: 16px;
    display: block;
    border-radius: 7px;
    box-shadow: 0px 3px 9px rgba(0,0,0,0.4);
}

input[type=text] {
    border: 1px solid grey;
    border-radius: 5px;
    padding: 10px;
}

.floating-box, pre {
    box-shadow: 0px 3px 10px rgba(0,0,0,0.4);
    margin: 0 0 0 15px;
    border-radius: 10px;
    padding: 30px;
}

.floating-box.last {
    flex: 1;
    margin-right: 15px;
    padding-right: 10px;
}

.credentials-box, .query-box {
    display: inline-block;
}

.credentials-box label, .credentials-box input {
    display: block;
}

.credentials-box label, .query-box label {
    margin-bottom: 10px;
}

.credentials-box input.first, .query-box input {
    margin-bottom: 15px;
}

.query-box {
    flex: 1;
    margin-left: 35px;
    margin-right: 20px;
}

.query-box label, .query-box input {

```

```

        display: block;
    }

    .query-box input {
        width: calc(100% - 20px);
    }

    pre {
        background-color: white;
        margin-right: 15px;
        margin-top: 15px
    }
</style>

    <h1> Auditoria de Dados </h1>
    <div class="row">
        <div class="floating-box">
            <h2> Ferramentas </h2>

            <form action="makeMigrations.php"
method="get">
                <button type="submit">Migrar
manualmente</button>
            </form>
        </div>

        <div class="floating-box last">
            <h2> Consultar logs </h2>

            <form name="form" action="" method="get">

                <div class="row">
                    <div class="credentials-box">
                        <label>Nome de utilizador:</label>
                        <input class="first" name="user"
type="text"/>

                        <label>Palavra-passe:</label>
                        <input name="password"
type="text"/>
                    </div>

                    <div class="query-box">
                        <label>Comando SQL:</label>
                        <input name="query" type="text"/>

                        <div>
                            <button style="float: right;
margin-top: 20px" type="submit"
name="execute">Executar</button>
                        </div>
                    </div>
                </div>
            </form>
        </div>
    </div>

```



```

        </div>
    </div>

    </form>
</div>
</div>

<?php
    if($_GET){
        if(!empty($_GET['user'])
        && !empty($_GET['query'])){
            $url="localhost";
            $username = $_GET['user'];
            $password = "".$_GET['password'];
            $database="auditor";
            $sql = $_GET['query'];

            $conn = mysqli_connect($url, $username,
            $password, $database); // Connects to the mysql database.
            If unsuccessful $conn is an object that is false.
            if (!$conn){
                die ("Connection Failed: Check your
            username and password, and try again."); // Prints
            a message and exits the current script
            }
            $result = mysqli_query($conn, $sql); //
            Performs the query on the database
            $rows = array(); // Creates an empty
            array
            if ($result) {
                if (mysqli_num_rows($result)>0){ //
            Returns number of rows in the result set

                while($r=mysqli_fetch_assoc($result)){ // Returns an
            array that corresponds to the fetched row
                    array_push($rows, $r); //
            Pushes one or more elements onto the end of array
                }
            }
            }
            mysqli_close ($conn); // Closes
            connection
            echo '<pre>', json_encode($rows,
            JSON_PRETTY_PRINT), '</pre>'; // Echoes a string containing
            a pretty representation of the supplied value.

            } else {
                echo "<pre> Please insert values into all
            fields in order to execute the query. </pre>";

```

```
    }  
}
```

```
?>  
</body>  
</html>
```

2.9 Avaliação Global da Qualidade das Especificações do próprio grupo

Avaliação (A,B,C,D,E) : E

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

-Não ficou claro que o stored procedure 'add_User' deve para além de registar o utilizador na tabela mysql.users registar também o investigador na tabela de investigador.

-Não existe um administrador na base de dados do auditor.

-O stored procedure 'selectMedicoes' que foi especificado tem um funcionamento muito limitado e rudimentar, visto que apenas permitia visualizar o conteúdo da tabela na integra. Por esta razão, o mesmo foi repensado na implementação, tendo sido corrigida a especificação.

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				X

2.10 Comparação de Implementações (ficheiro versus PHP)

As duas abordagens diferem maioritariamente quanto à necessidade da existência de ficheiros na exportação feita por CSV (um por cada tabela no caso da especificação fornecida pelo outro grupo) e na necessidade de da existência de dois eventos, um para a exportação e outro para a importação. Esta abordagem acarreta a necessidade de existir um período de espera entre a exportação para o ficheiro e a importação para a base de dados o que faz com que os dados estejam expostos durante esse mesmo período o que se pode revelar um problema de segurança.

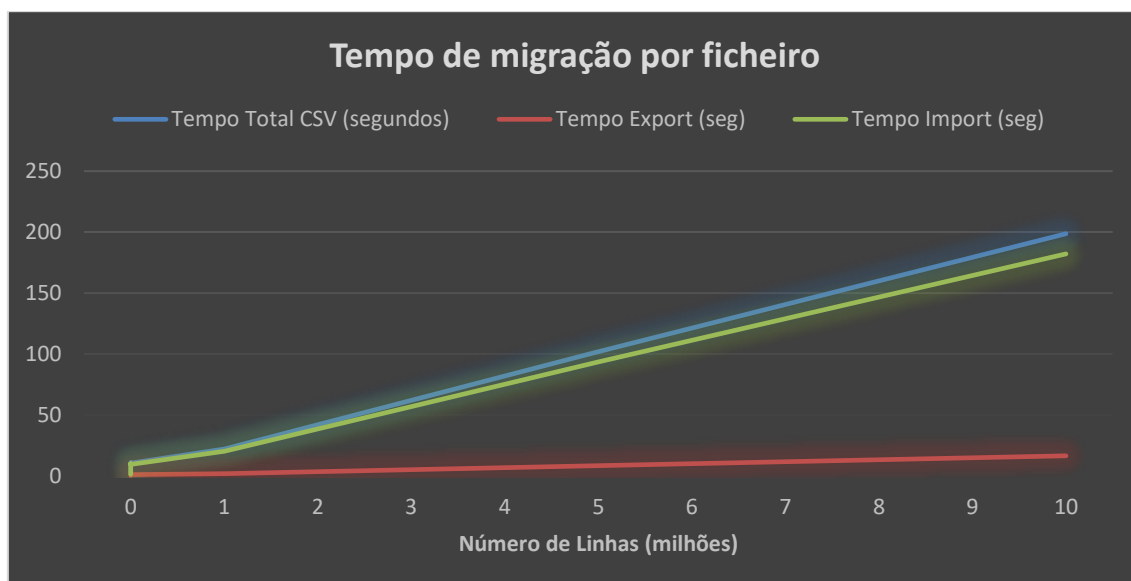
No caso do php apenas há necessidade da existência de um evento para a importação e não há necessidade de recorrer a ficheiros, pelo que esta abordagem é mais segura.

Quanto à rapidez de exportação, concluímos através dos testes efetuados que a abordagem de ficheiro se revela mais rápida.

Por sua vez, a robustez da migração recorrendo a PHP é mais robusta na medida em que consegue lidar com problemas decorrentes de falhas de energia ou nas bases de dados e adicionalmente não podem surgir problemas com a eliminação/alteração dos ficheiros como acontece com o CSV, como explicado na secção 2.10.2.

Em relação à flexibilidade o PHP revela-se mais versátil, uma vez que apenas tem um evento associado à migração, contrariamente ao que acontece com a abordagem baseada em CSV que tem dois eventos. Isto faz com que uma possível alteração da periodicidade da migração se revele mais problemática no CSV em comparação com o PHP.

2.10.1 Eficiência de Migração

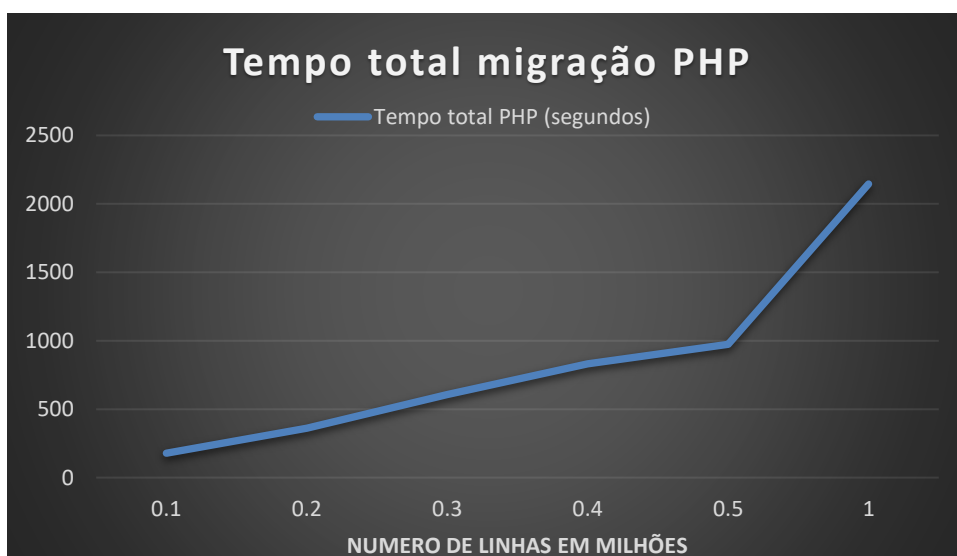


Relativamente à migração por ficheiro temos duas fases distintas, a exportação para os ficheiros e a importação dos dados presentes nos ficheiros para a base de dados destino.

Como se pode observar no gráfico, a fase mais demorada é a importação dos dados para a base de dados destino, que se encontra representada a verde, sendo que a exportação para os ficheiros é quase nula.

Deste modo, o tempo total de migração segue um crescimento aproximadamente linear.

Por sua vez, a migração feita recorrendo ao PHP revelou-se mais demorada.

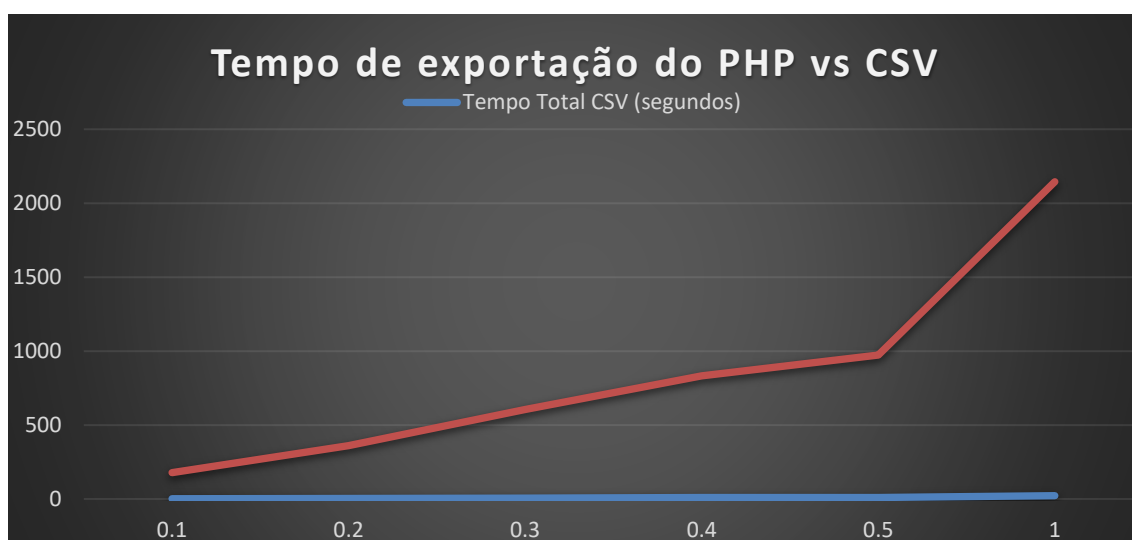


Com o gráfico seguinte podemos ver as diferenças dos tempos de migração dos dados relativos às estratégias anteriormente referidas, nas quais o CSV se revela cerca de 9 vezes mais rápido que o PHP.

Esta diferença justifica-se com o facto de a nossa implementação recorrer a um formato JSON para representar os dados. O PHP implementado começa por codificar os dados recebidos da base de dados, de seguida de modo a verificar se há dados para migrar, descodificamos os mesmos. Seguidamente os dados não descodificados são usados noutro script PHP, onde são novamente descodificados para serem inseridos na base de dados do auditor. Todo este processo de codificações e descodificações é bastante demorado, o que aumenta bastante o tempo total de migração.

Uma forma de melhorar este desempenho seria verificando apenas se haviam dados para inserir na base de dados destino, unicamente no script responsável pela inserção dos dados.

Desta forma, descodificamos a variável JSON uma única vez com o intuito de verificar se existem dados a inserir e no caso de existirem usam-se os dados já descodificados, reduzindo assim o número de descodificações e consequentemente o tempo total de migração.



De referir que o gráfico apresentado não se encontra à escala.

2.10.2 Robustez

Relativamente à especificação apresentada que é baseada em ficheiro, a mesma revela-se robusta no caso de ocorrer uma falha de energia, contudo o mesmo não acontece caso ocorra um erro de software.

Caso ocorra uma falha de energia os ficheiros não são eliminados e a task responsável por executar a importação dos mesmos é chamada a execução quando houver uma reposição da energia. Contudo, se ocorrer um erro de software a task pode não ser chamada novamente a execução fazendo com que a exportação não ocorra ainda que os dados tenham sido marcados como migrados na base de dados de origem.

O período de espera entre a exportação e a migração é inevitável caso se decida utilizar o ficheiro para fazer a migração, pelo que deveria ter sido especificada uma solução que verificasse qual era o maior id exportado relativo a cada uma das tabelas para assim se ter a certeza que os dados chegaram corretamente ao destino.

Isto podia ser feito através de uma ligação à base de dados destino para assim se proceder a um acerto da tabela 'dados migrados' da base de dados origem.

Relativamente à migração feita através de PHP, consideramos que a mesma é bastante mais robusta, nomeadamente em relação a falhas de energia e de software uma vez que salvaguarda as seguintes situações:

- Caso haja uma falha de qualquer um dos tipos anteriormente referidos no pedido das migrações ou na migração automática, nenhum dado é automaticamente marcado como enviado pelo que, mal surja um novo pedido todos os dados serão migrados sem que haja perdas.

- Caso haja uma falha de energia ou de software ao inserir os dados na base de dados destino, os dados nunca são marcados como enviados na base de dados de origem pelo que na próxima migração há o conhecimento que faltam migrar esses mesmos dados.

- No caso de haver falhas de energia ou de software durante a marcação dos dados enviados, quando ocorrer uma nova migração os erros decorrentes das referidas falhas serão detetados devido ao id da base de dados origem já estar na base de dados destino e esses mesmos dados serão marcados como migrados sem que haja por isso qualquer tipo de duplicação de informação.

2.10.3 Flexibilidade / Dependência

A especificação e consequente implementação recorrendo a PHP revela-se bastante mais versátil em relação à alteração da periodicidade.

Tal acontece porque na especificação que usa o PHP apenas há um evento do sistema operativo que efetua a migração com a periodicidade desejada, pelo que se quisermos alterar essa periodicidade basta alterar apenas esse mesmo evento sem que haja qualquer tipo de problema decorrente dessa operação.

Por outro lado, recorrendo a um ficheiro para executar a migração há a necessidade de ter dois eventos, um na base de dados origem para migrar dos dados para um ficheiro e outro no sistema operativo para importar os dados do ficheiro para a base de dados destino. Assim, se for alterada a periodicidade do evento na base de dados origem é necessário alterar na mesma medida o evento da base de dados destino. Caso isto não seja feito podem surgir problemas de incoerência nos seguintes casos:

- se for antecipada a periodicidade do evento de importação em mais de dez minutos (período especificado de diferença entre exportação e importação) a migração não será efetuada pois ainda não irá existir nenhum ficheiro csv de onde importar os dados (no caso de os ficheiros antigos serem eliminados ao fim da importação) ou irá ocorrer uma duplicação de dados (no caso de os ficheiros não serem apagados após a importação).

- se for adiada em mais de 10 minutos a periodicidade do evento de exportação vai ocorrer exatamente o mesmo problema referido no ponto anterior.

Devido aos factos anteriormente referidos concluímos que o PHP é mais versátil em relação à alteração da periodicidade de exportação dos dados.

Quanto à dependência de uma base de dados em relação à outra concluímos que o PHP apresenta uma menor relação de

dependência comparando com a exportação recorrendo a ficheiros csv, uma vez que na solução apresentada a exportação incremental por csv é feita recorrendo a uma tabela de controlo denominada por 'dadosexportados'. Se houver uma falha na base de dados origem ou na tabela anteriormente referida a exportação fica comprometida.

2.10.4 Segurança

A exportação através de ficheiro cria um período de suscetibilidade, que corresponde ao intervalo de tempo entre a exportação para o ficheiro e a importação dos dados.

Durante esta janela temporal qualquer pessoa pode ler, apagar ou modificar os dados o que se revela um problema bastante grave ao nível da segurança, sendo que quanto maior for esse intervalo de tempo maior é a possibilidade de ocorrência dos casos anteriormente referidos.

Por sua vez, o php não tem esta vulnerabilidade, uma vez que a migração é feita de forma direta entre as duas bases de dados.

Posto isto chegamos à conclusão que a solução implementada recorrendo ao php é mais segura.

2.11 Auditoria de Dados (base de dados origem)

```
<!DOCTYPE html>

<html>

<body>


<style>

html, body {

    font-family: sans-serif;

    margin:0;

    padding: 0;

    color: #404040;

}


h1 {

    margin: 30px 10px;

    font-size: 50px;

    text-align: center;

}


h2 {

    margin-top: 0;

    margin-bottom: 20px;

}


.row {

    display: flex;
```

```
}
```

```
button {  
    background-color: #629fde;  
    border: none;  
    color: white;  
    padding: 15px 32px;  
    text-align: center;  
    font-size: 16px;  
    display: block;  
    border-radius: 7px;  
    box-shadow: 0px 3px 9px rgba(0,0,0,0.4);  
}
```

```
input[type=text] {  
    border: 1px solid grey;  
    border-radius: 5px;  
    padding: 10px;  
}
```

```
.floating-box, pre {  
    box-shadow: 0px 3px 10px rgba(0,0,0,0.4);  
    margin: 0 0 0 15px;  
    border-radius: 10px;  
    padding: 30px;  
}
```

```

.floating-box.last {
    flex: 1;
    margin-right: 15px;
    padding-right: 10px;
}

.credentials-box, .query-box {
    display: inline-block;
}

.credentials-box label, .credentials-box input {
    display: block;
}

.credentials-box label, .query-box label {
    margin-bottom: 10px;
}

.credentials-box input.first, .query-box input {
    margin-bottom: 15px;
}

.query-box {
    flex: 1;
    margin-left: 35px;
    margin-right: 20px;
}

```

```
.query-box label, .query-box input {
    display: block;
}
```

```
.query-box input {
    width: calc(100% - 20px);
}
```

```
pre {
    background-color: white;
    margin-right: 15px;
    margin-top: 15px
}
```

```
</style>
```

```
<h1> Auditoria de Dados </h1>

<div class="row">
    <div class="floating-box">
        <h2> Ferramentas </h2>

        <form                action="makeMigrations.php"
method="get">

            <button                type="submit">Migrar
manualmente</button>

        </form>

    </div>
```

```

<div class="floating-box last">

    <h2> Consultar logs </h2>

    <form name="form" action="" method="get">

        <div class="row">

            <div class="credentials-box">

                <label>Nome de utilizador:</label>

                <input class="first" name="user"
type="text"/>

                <label>Palavra-passe:</label>

                <input name="password"
type="text"/>

            </div>

            <div class="query-box">

                <label>Comando SQL:</label>

                <input name="query" type="text"/>

                <div>

                    <button style="float: right;
margin-top: 20px" type="submit"
name="execute">Executar</button>

                </div>

            </div>

        </div>

    </form>

```


</div>

</div>

<?php

```
    if($_GET){
        if(!empty($_GET['user'])
        && !empty($_GET['query'])) {
            $url="localhost";
            $username = $_GET['user'];
            $password = "".$_GET['password'];
            $database="auditor";
            $sql = $_GET['query'];

            $conn = mysqli_connect($url, $username,
            $password, $database); // Connects to the mysql database.
            If unsuccessful $conn is an object that is false.

            if (!$conn){
                die ("Connection Failled: Check your
                username and password, and try again."); // Prints
                a message and exits the current script
            }

            $result = mysqli_query($conn, $sql); //
            Performs the query on the database

            $rows = array(); // Creates an empty
            array

            if ($result) {
                if (mysqli_num_rows($result)>0){ //
                Returns number of rows in the result set
```

```

        while($r=mysqli_fetch_assoc($result)){ // Returns an
array that corresponds to the fetched row

                                array_push($rows, $r); //
Pushes one or more elements onto the end of array

                                }

                                }

                                }

                                mysqli_close ($conn); // Closes
connection

                                echo '<pre>', json_encode($rows,
JSON_PRETTY_PRINT), '</pre>'; // Echoes a string containing
a pretty representation of the supplied value.

                                } else {

                                echo "<pre> Please insert values into all
fields in order to execute the query. </pre>";

                                }

                                }

?>

</body>

</html>

```