




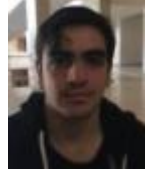


# Sistemas de Informação Distribuídos

Licenciaturas em Engenharia Informática e Informática e Gestão de Empresas  
2017-2018, Segundo Semestre




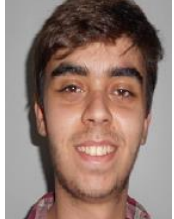

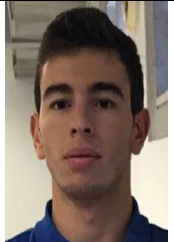
## Monitorização de Culturas em Laboratório

### Auditoria e Migração

Identificação do grupo autor da especificação (Etapa A): Grupo 13

Número	Nome	Foto
65598	André Almeida	
78463	Diogo Sarmento	
78001	Hugo Cruz	
78009	Luís Fernandes	
77577	Miguel Figueiredo	
77689	Rostislav Andreev	
Especificação: PHP		

Identificação do grupo autor da implementação (Etapas B e C): \_\_7\_\_

Número	Nome	Foto
73051	Diogo Juvandes	
73498	Alexandre Costa	
73528	Miguel Penedo	
78337	Marcos Teixeira	
79038	Diogo Toupa	
77983	Tiago Garção	
<p>Especificação: Ficheiro</p> <p>Implementação: PHP</p>		

# 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	10
1.2	Utilizadores Base de Dados de Origem	11
1.2.1	Apreciação Crítica a Gestão de Utilizadores Base de Dados de Origem	12
1.3	Gestão de Logs	13
1.3.1	Triggers de suporte à criação de logs Base de Dados de Origem	13
1.3.1.1	Apreciação Crítica de triggers para gestão de logs	14
1.3.1.2	Triggers Implementados para gestão de logs	15
1.3.2	Stored Procedures de suporte à criação de logs (24	
1.3.2.1	Apreciação Crítica de Stored Procedures de suporte à criação de logs	17
1.3.2.2	Stored Procedures Implementados de suporte à criação de logs	18
1.4	Migração entre Bases de Dados	19
1.4.1	Esquema relacional da base de Dados Mysql (destino)	19
1.4.1.1	Apreciação Crítica e esquema relacional implementado	20
1.4.2	Forma de Migração	21
1.4.2.1	Apreciação Crítica à especificação da forma de migração	22
1.4.3	Gestão de Utilizadores de Suporte à Migração (origem e/ou destino)	23
1.4.3.1	Apreciação Crítica à especificação da Gestão de Utilizadores	24
1.4.4	Triggers de suporte à migração de dados (origem e/ou destino) (35	
1.4.4.1	Apreciação Crítica de triggers de suporte à migração de dados	26
1.4.4.2	Triggers Implementados de suporte à migração de dados	27
1.4.5	Stored Procedures de suporte à migração de dados	28
1.4.5.1	Apreciação Crítica de Stored Procedures de suporte à migração de dados	29
1.4.5.2	Stored Procedures Implementados de suporte à migração de dados	30
1.4.6	Eventos de suporte à migração de dados	31
1.4.6.1	Apreciação Crítica de Eventos	32
1.4.6.2	Eventos Implementados	33
1.4.7	PHP suporte à migração de dados (se relevante)	34
1.4.7.1	Apreciação Crítica ao PHP especificado	35
1.4.7.2	PHP Implementado	36
	ISCTE / SID / 2018-2019	4

1.5	Avaliação Global de especificações da Etapa A	37
2	Etapa C (Especificação e Implementação do Próprio Grupo)	39
2.1	Especificação do Esquema relacional da base de Dados Origem	39
2.2	Especificação de Utilizadores	40
2.3	Especificação de Gestão de Logs	41
2.3.1	Triggers de suporte à gestão de logs	41
2.3.2	Stored Procedures de suporte à gestão de logs	42
2.4	Avaliação da especificação do próprio grupo Gestão de Logs	43
2.5	Implementação Gestão de Logs	44
2.5.1	Utilizadores implementados	44
2.5.2	Lista de Triggers	45
2.5.3	Triggers Implementados	46
2.5.4	Lista de Stored Procedures	47
2.5.5	Stored Procedures Implementados	48
2.6	Especificação de Migração entre Bases de Dados	49
2.6.1	Esquema relacional da base de Dados Mysql especificada (destino)	49
2.6.2	Forma de Migração Especificada	50
2.6.3	Utilizadores Especificados	51
2.6.4	Triggers de suporte à migração de dados especificados	52
2.6.5	Stored Procedures de suporte à migração de dados especificados	53
2.6.6	Eventos de suporte à migração de dados especificados	54
2.6.7	PHP de suporte à migração de dados especificado	55
2.7	Avaliação das especificações do próprio grupo Migração	56
2.8	Implementação da Migração de Dados	57
2.8.1	Utilizadores Implementado	57
2.8.2	Lista Triggers	58
2.8.3	Triggers Implementados	59
2.8.4	Lista de Stored Procedures	60
2.8.5	Stored Procedures Implementados	61
2.8.6	Lista Eventos	62
2.8.7	Eventos Implementados	63
2.8.8	PHP Implementado	64
	Avaliação Global da Qualidade das Especificações	65
2.9	Comparação de Implementações (ficheiro versus PHP)	66

2.9.1	Eficiência de Migração	67
2.9.2	Robustez	68
2.9.3	Flexibilidade / Dependência	69
2.9.4	Segurança	70
2.10	Auditoria de Dados (base de dados origem)	71

# Monitorização de Culturas em Laboratório

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

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

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

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

## Alertas

Existem dois tipos de alertas:

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

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

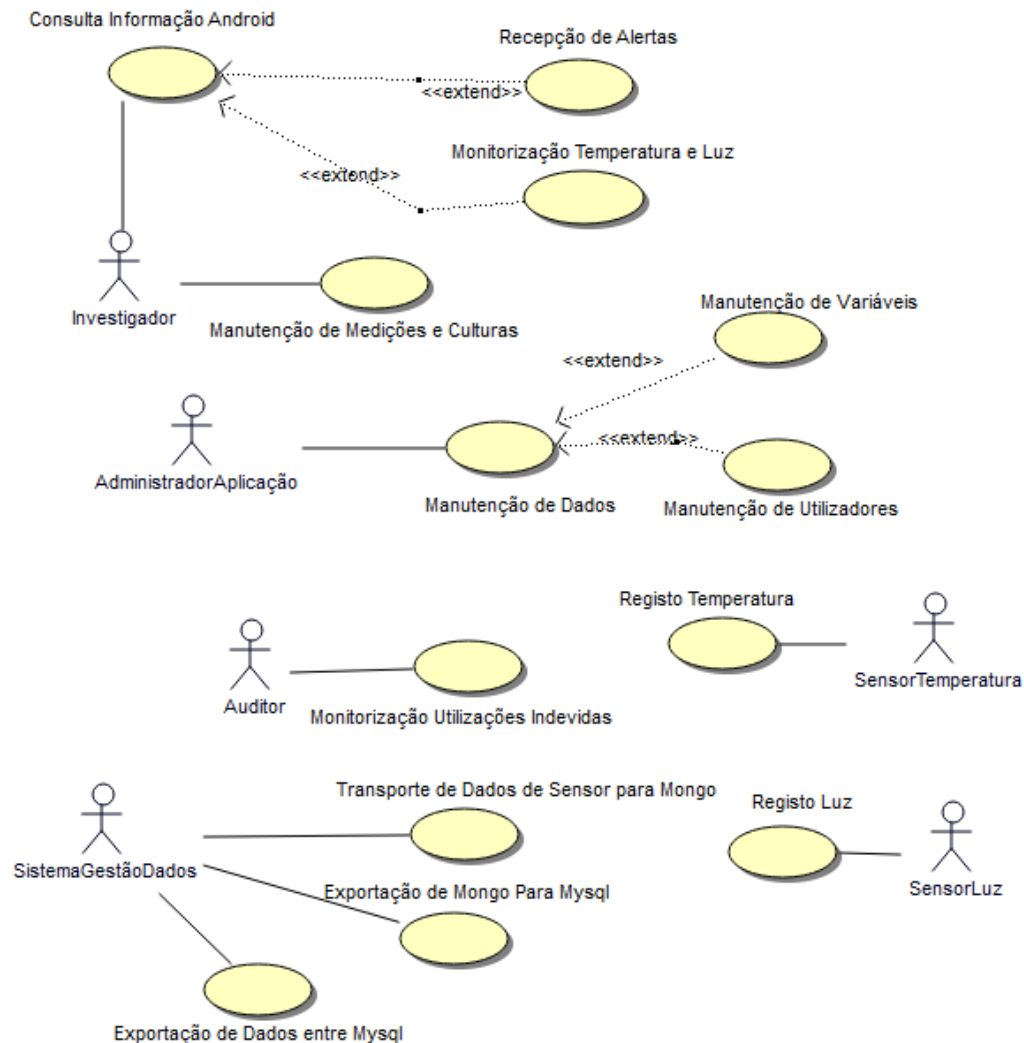
Cada investigador deverá ter a possibilidade de, através de um telemóvel, monitorizar a evolução da temperatura e luminosidade (não apenas a última leitura, mas a evolução na última hora ou horas) e receber os dois tipos de alertas.

## Registo de Acessos

É necessário guardar na base de dados (mysql) o registo de todas as operações de escrita sobre todas as tabelas (quais dados foram alterados/inseridos/apagados, quando e por quem) e o registo de operações de consulta apenas sobre a tabela Medições. Esse registo de alterações (*log*) é *exportado* incrementalmente (apenas informação nova) e periodicamente para uma base de dados autónoma (também mysql). Através dessa base de dados (apenas de

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

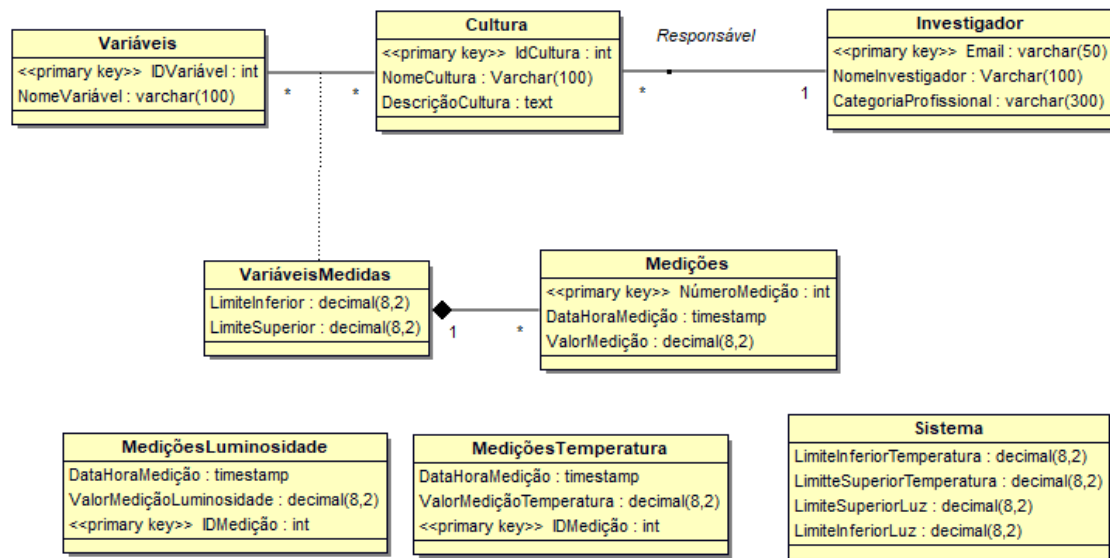
### Diagrama de Use Case Global



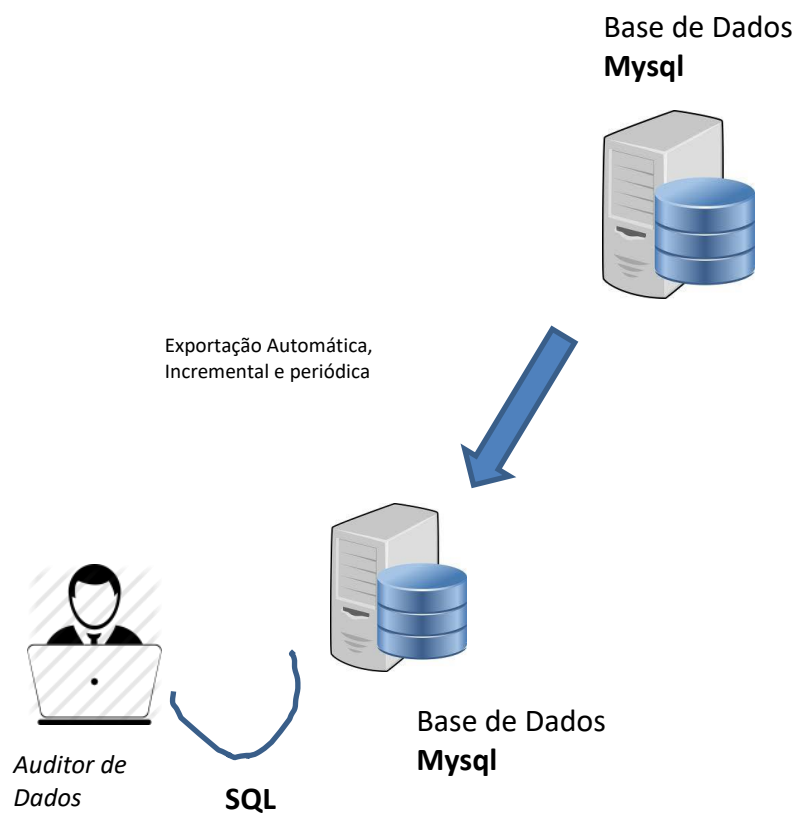
No presente relatório apenas são contemplados os use case “Exportação Dados entre Mysql”, “Monitorização de Utilizações Indevidas” e “Manutenção de Utilizadores” (apenas a componente Mysql/Privilégios/SP/Triggers)). A componente Java (manutenção de culturas, medições, variáveis e utilizadores) não é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



## Diagrama de Classes de Suporte à Base de Dados

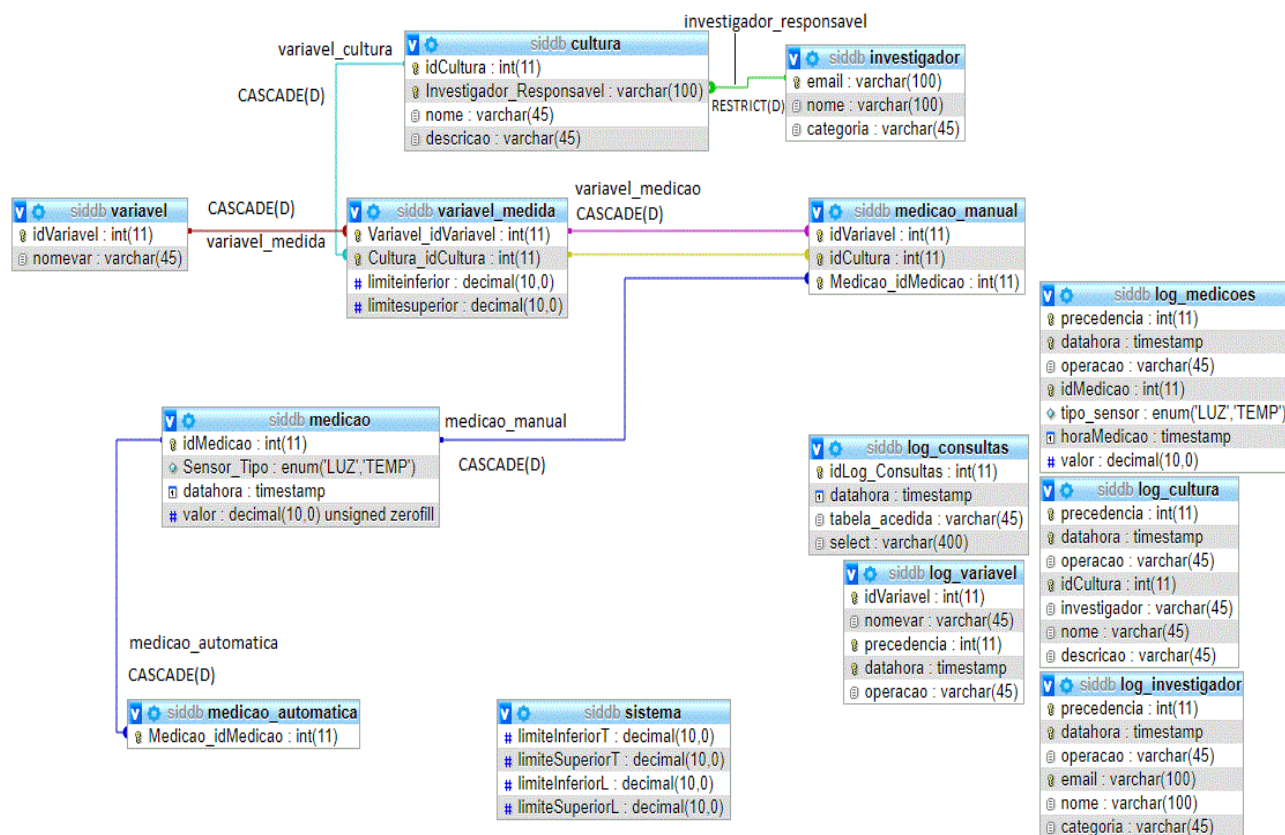


## Esquema de Migração



# 1 Etapa A e B

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



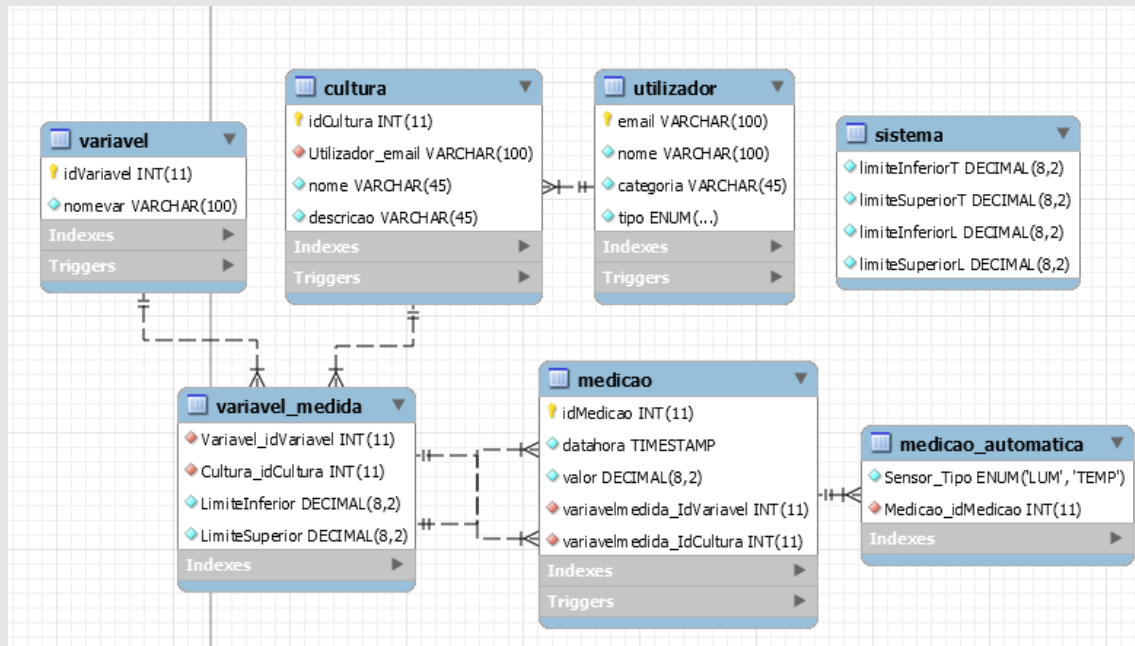
### 1.1.1 Apreciação Crítica e esquema relacional implementado

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

Esquema razoável, mas que não segue a estrutura inicial dada e não apresenta explicação para as alterações.

Foram feitas alterações? (Sim/Não): Sim

**Novo Esquema (assinale e justifique as alterações)**



Alteração do nome da tabela "investigador" para "utilizador" e criação do campo "tipo", com o intuito de poderem existir vários tipos de utilizador. Correção e simplificação das tabelas "medicao", "medicao\_automatica", "medicao\_manual" e respetivas relações.

<b>log_cultura</b> <ul style="list-style-type: none"> <li>precedencia INT(11)</li> <li>datahora TIMESTAMP</li> <li>operacao VARCHAR(1)</li> <li>idCultura INT(11)</li> <li>Utilizador_Email VARCHAR(45)</li> <li>nome VARCHAR(45)</li> <li>descricao VARCHAR(45)</li> </ul> <b>Indexes</b>	<b>log_medicoes</b> <ul style="list-style-type: none"> <li>precedencia INT(11)</li> <li>datahora TIMESTAMP</li> <li>operacao VARCHAR(1)</li> <li>idMedicao INT(11)</li> <li>horaMedicao TIMESTAMP(6)</li> <li>valor DECIMAL(8,2)</li> <li>variavelmedida_idVariavel INT(11)</li> <li>variavelmedida_idCultura INT(11)</li> </ul> <b>Indexes</b>
<b>log_utilizador</b> <ul style="list-style-type: none"> <li>precedencia INT(11)</li> <li>datahora TIMESTAMP</li> <li>operacao VARCHAR(1)</li> <li>email VARCHAR(100)</li> <li>nome VARCHAR(100)</li> <li>categoria VARCHAR(45)</li> <li>tipo ENUM(...)</li> </ul> <b>Indexes</b>	<b>log_variavel</b> <ul style="list-style-type: none"> <li>idVariavel INT(11)</li> <li>nomevar VARCHAR(100)</li> <li>precedencia INT(11)</li> <li>datahora TIMESTAMP</li> <li>operacao VARCHAR(1)</li> </ul> <b>Indexes</b>

Em relação aos logs, mantivemos a mesma estrutura, mas não incluímos a tabela log\_consultas uma vez que a sua utilização iria apenas ser realizada por um SP cuja utilização é questionável (select\_auditor).

## 1.2 Utilizadores Base de Dados de Origem

Tabelas	Tipo de Utilizador		
	Administrador	Investigador	Auditor Local
Investigador	E, L	-	-
Cultura	E, L	-	-
Variavel	E, L	-	-
Variavel_medida	E, L	-	-
Medicao	E, L	-	-
Medicao_manual	E, L	-	-
Medicao_automatica	E, L	-	-
Sistema	E, L	-	-
Log_consultas	E, L	-	L
Log_investigador	E, L	-	L
Log_cultura	E, L	-	L
Log_variavel	E, L	-	L
Log_medicoes	E, L	-	L
<b>Stored Proc.</b>			
Select_auditor	-	-	X
Cria_cultura	-	X	-
Mostra_culturas	-	X	-
Alterar_cultura	-	X	-
Apaga_cultura	-	X	-
Cria_variavel	-	X	-
Mostra_variaveis	-	X	-
Apaga_variavel	-	X	-
Cria_medicao	-	X	-

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

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

##### **Análise crítica (clareza, completude, rigor):**

Não existe (qualquer tipo de) auditor na base de dados de origem. No enunciado é referido que o administrador faz manutenção de variáveis e utilizadores e o investigador faz manutenção de culturas e medições, pelo que não conseguimos entender as decisões tomadas, tendo em conta a inexistência de qualquer justificação.

##### **Solução Implementada:**

Tabela	Administrador	Investigador
Cultura	-	L,E
Cultura_Log	-	L
Medições	-	L,E
Medições_Log	-	L
Sistema	L,E	L
Utilizador	L,E	L
Utilizador_Log	L	L
Variáveis	L,E	L
Variáveis_Log	L	L
VariáveisMedidas	L,E	L
Consultas_Log	-	L
MediçãoAutomática	L	L

Stored Procedures	Administrador	Investigador
select_all_medicoes	-	X
cria_utilizador	X	-
cria_medicao	-	X
cria_cultura	-	X
altera_cultura	-	X

## 1.3 Gestão de Logs

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

Nome Trigger	Tabela	Tipo de Operação (I,U,D)	Evento (A, B)	Notas (apenas indicar aquilo que não seja óbvio)
Log_investigador_I_A	Investigador	I	A	
Log_investigador_U_B	Investigador	U	B	
Log_investigador_U_A	Investigador	U	A	
Log_investigador_D_B	Investigador	D	B	
Log_cultura_I_A	Cultura	I	A	
Log_cultura_U_B	Cultura	U	B	
Log_cultura_U_A	Cultura	U	A	
Log_cultura_D_B	Cultura	D	B	
Log_medicao_I_A	Medicao	I	A	
Log_medicao_U_B	Medicao	U	B	
Log_medicao_U_A	Medicao	U	A	
Log_medicao_D_B	Medicao	D	B	
Log_variavel_I_A	Variavel	I	A	
Log_variavel_U_B	Variavel	U	B	
Log_variavel_U_A	Variavel	U	A	
Log_variavel_D_B	Variavel	D	B	

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

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

Em geral tem todos os triggers necessários que auxiliam ao bom funcionamento do projeto, no entanto não conseguimos entender a razão para especificar dois tipos de triggers de update (after e before update) pois para nós parecem-nos redundantes, o facto de também não possuir uma explicação para a sua adição foi a razão para os triggers 'before update' não serem implementados.

#### 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)
log_utilizador_I_A	X			
log_utilizador_U_A	X			
log_utilizador_D_B	X			
log_cultura_I_A	X			
log_cultura_U_A	X			
log_cultura_D_B	X			
log_medicao_I_A	X			
log_medicao_U_A	X			
log_medicao_D_B	X			
log_variavel_I_A	X			
log_variavel_U_A	X			
log_variavel_D_B	X			
log_utilizador_U_B			X	



log_cultura_U_B			X	
log_medicao_U_B			X	
log_variavel_U_B			X	

### 1.3.1.2 Triggers Implementados para gestão de logs

#### 1. Nome Trigger: log\_utilizador\_I\_A

Inserir na tabela log\_investigador uma linha com informações sobre o insert realizado.

*Código:*

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`log_utilizador_I_A`()
AFTER INSERT ON `utilizador`
FOR EACH ROW
BEGIN
insert into utilizador_log(Email, NomeUtilizador,
CategoriaProfissional, TipoUtilizador,
utilizador,data_operacao, operacao)
values (' ',' ',' ',' ',now(),' ');
END
```

#### 2. Nome Trigger: log\_utilizador\_U\_A

Inserir na tabela log\_investigador uma linha com informações sobre o update realizado.

*Código:*

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`sql_php`.`log_investigador_U_A` AFTER UPDATE ON
`utilizador` FOR EACH ROW
BEGIN
declare new_precedencia DOUBLE ;
select max(precedencia) + 1 into new_precedencia from
log_investigador;

insert into log_investigador (datahora, operacao, email,
```

```

nome, categoria)
values (now(), 'U', new.email, new.nome, new.categoria);

if(old.email<>new.email) or (old.email is Null and
new.email is NOT NULL) THEN
update log_investigador set email = new.email where
precedencia = new_precedencia;
end if;

if(old.nome<>new.nome) or (old.nome is Null and new.nome
is NOT NULL) THEN
update log_investigador set nome = new.nome where
precedencia = new_precedencia;
end if;

if(old.categoria<>new.categoria) or (old.categoria is
Null and new.categoria is NOT NULL) THEN
update log_investigador set categoria = new.categoria
where precedencia = new_precedencia;
end if;

END

```

### **3. Nome Trigger: log\_utilizador\_B\_D**

Inserir na tabela log\_investigador uma linha com informações sobre o delete realizado.

*Código:*

```

CREATE DEFINER='root'@'localhost' TRIGGER
`log_utilizador_B_D` BEFORE DELETE ON `cultura` FOR EACH
ROW BEGIN
insert into log_cultura
(datahora,operacao,idcultura,email,investigador,nome,desc
ricao) VALUES
(now(),'D',old.idcultura,
old.email,CURRENT_USER,,old.nome,old.descrição) ;
END

```

### **4. Nome Trigger: log\_cultura\_I\_A**

Inserir na tabela log\_cultura uma linha com informações sobre o insert realizado.

*Código:*

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`sql_php`.`log_cultura_I_A` AFTER INSERT ON `cultura` FOR
EACH ROW
BEGIN
insert into log_cultura(datahora, operacao, idCultura,
Investigador_Responsavel, nome, descricao)
values (now(), 'I', new.idCultura,
new.Investigador_Responsavel, new.nome, new.descricao);
END
```

## 5. Nome Trigger: log\_cultura\_U\_A

Inserir na tabela log\_cultura uma linha com informações sobre o update realizado.

*Código:*

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`sql_php`.`log_cultura_AFTER_UPDATE` AFTER UPDATE ON
`cultura` FOR EACH ROW
BEGIN

declare new_precedencia DOUBLE ;
select max(precedencia) + 1 into new_precedencia from
log_investigador;

insert into log_cultura(datahora, operacao, idCultura,
Investigador_Responsavel, nome, descricao)
values (now(), 'U', new.idCultura,
new.Investigador_Responsavel, new.nome, new.descricao);

if(old.nome<>new.nome) or (old.nome is Null and new.nome
is NOT NULL) THEN
update log_cultura set nome = new.nome where precedencia
= new_precedencia;
end if;
if(old.Investigador_Responsavel<>new.Investigador_Respons
avel) or (old.Investigador_Responsavel is Null and
new.Investigador_Responsavel is NOT NULL) THEN
update log_cultura set Investigador_Responsavel =
new.Investigador_Responsavel where precedencia =
new_precedencia;
end if;

if(old.descricao<>new.descricao) or (old.descricao is
Null and new.descricao is NOT NULL) THEN
update log_cultura set descricao = new.descricao where
precedencia = new_precedencia;
end if;
```

END

#### **6. Nome Trigger: log\_cultura\_B\_D**

Insere na tabela log\_cultura uma linha com informações sobre o delete realizado.

*Código:*

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`log_cultura_B_D` BEFORE DELETE ON `cultura` FOR EACH ROW
BEGIN
insert into insert into log_cultura(datahora, operacao,
idCultura, Investigador_Responsavel, nome, descricao)
values (now(), 'I', old.idCultura,
old.Investigador_Responsavel, old.nome, old.descricao);
END
```

#### **7. Nome Trigger: log\_medicao\_I\_A**

Insere na tabela log\_medicao uma linha com informações sobre o insert realizado.

*Código:*

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`sql_php`.`medicao_AFTER_INSERT` AFTER INSERT ON
`medicao` FOR EACH ROW

BEGIN

insert into log_medicoes(datahora, operacao, idMedicao,
Sensor_Tipo,valor)

values (now(), 'I', new.idMedicao, new.Sensor_Tipo,
new.valor);

END
```

#### **8. Nome Trigger: log\_medicao\_U\_A**

Insere na tabela log\_medicao uma linha com informações sobre o update realizado.

*Código:*

```

CREATE DEFINER=`root`@`localhost` TRIGGER
sql_php.`log_medicao_U_A` AFTER UPDATE ON medicao FOR
EACH ROW
BEGIN

declare new_precedencia DOUBLE ;
select max(precedencia) + 1 into new_precedencia from
log_investigador;

insert into log_medicoes(datahora, operacao, idMedicao,
Sensor_Tipo,valor)
values (now(), 'U', new.idMedicao, new.Sensor_Tipo,
new.valor);

if(old.Sensor_Tipo<>new.Sensor_Tipo) or (old.Sensor_Tipo
is Null and new.Sensor_Tipo is NOT NULL)
THEN
update log_medicoes set Sensor_Tipo = new.Sensor_Tipo
where precedencia = new_precedencia;
end if;

if(old.valor<>new.valor) or (old.valor is Null and
new.valor is NOT NULL)
THEN
update log_medicoes set valor = new.valor where
precedencia = new_precedencia;
end if;

END

```

#### **9. Nome Trigger: log\_medicao\_B\_D**

Inserir na tabela log\_medicao uma linha com informações sobre o delete realizado.

Código:

```

CREATE DEFINER=`root`@`localhost` TRIGGER
`log_medicao_B_D` BEFORE DELETE ON `medicao` FOR EACH ROW
BEGIN
insert into log_medicoes(datahora, operacao, idMedicao,
Sensor_Tipo,valor)
values (now(), 'I', old.idMedicao, old.Sensor_Tipo,
old.valor);

END

```

#### **10. Nome Trigger: log\_variavel\_I\_A**

Inserir na tabela log\_variavel uma linha com informações sobre o insert realizado.

Código:

```
CREATE DEFINER=`root`@`localhost` TRIGGER
sql_php.`log_variavel_I_A` AFTER INSERT ON variavel FOR
EACH ROW
BEGIN
insert into log_variavel(idVariavel, nomeVar, datahora,
operacao)
values (new.idVariavel, new.NomeVar,now(), 'I');
END
```

#### **11. Nome Trigger: log\_variavel\_U\_A**

Inserir na tabela log\_variavel uma linha com informações sobre o update realizado.

Código:

```
CREATE DEFINER=`root`@`localhost` TRIGGER
sql_php.`log_variavel_U_A` AFTER UPDATE ON variavel FOR
EACH ROW
BEGIN

declare new_precedencia DOUBLE ;
select max(precedencia) + 1 into new_precedencia from
log_investigador;

insert into log_variavel(idVariavel, nomeVar, datahora,
operacao)
values (new.idVariavel, new.NomeVar,now(), 'U');

if(old.nomeVar<>new.nomeVar) or (old.nomeVar is Null and
new.nomeVar is NOT NULL)
THEN
update nomeVar set nomeVar = new.nomeVar where
precedencia = new_precedencia;
end if;
END
```

#### **12. Nome Trigger: log\_variavel\_D\_B**

Inserir na tabela log\_variavel uma linha com informações sobre o delete realizado.

Código:

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`log_variavel_B_D` BEFORE DELETE ON `variavel` FOR EACH
ROW BEGIN
```

```
insert into log_variavel(idVariavel, nomeVar, datahora,  
operacao)  
values (old.idVariavel, old.NomeVar,now(), 'I');  
  
END
```

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

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	Muito breve descrição
Cria_cultura	Varchar(45) - nome, varchar(45) - descricao	Boolean-sucess	Cria uma nova cultura com o investigador que chama a procedure como responsável, devolve true se criada com sucesso.
Mostra_culturas	-	-	Realiza um select das culturas cujo o responsável seja o investigador que chama a procedure.
Alterar_cultura	Int-id_cultura, Varchar(45) - nome, varchar(45) - descricao	Boolean-sucess	Altera a cultura selecionada através do id_cultura, caso os parâmetros sejam vazios nada é alterado, devolve true se a alteração for realizada com sucesso.
Apaga_cultura	Int-id_cultura	Boolean-sucess	Apaga a cultura especificada, mas apenas se o investigador que chama a procedure for o seu responsável e devolve true se for apagada com sucesso.
Cria_variavel	Varchar(45) - nome, int-id_cultura	Boolean-sucess	Cria uma nova variável associada a cultura especificada falha caso o investigador que chama a procedure não seja o responsável da dada cultura.
Mostra_variaveis	Int-id_cultura	Boolean-sucess	Realiza um select das variáveis associadas, assim como as suas respectivas medições à cultura dada falha caso o investigador não seja o responsável
Apaga_variavel	Int-id_variavel	Boolean-sucess	Apaga a variavel especificada, mas apenas se o investigador que chama a procedure for o



			responsável da cultura associada.
Cria_medica ao	Int- id_variave l, ENUM('LUZ' , 'TEMP')- tipo,decim al-valor	Boolean- sucess	Cria uma medição manual associada a variável referida falha caso o investigador não seja responsável pela cultura associada a variável.
Select_aud itor	varchar(10 00)-select	-	Realiza o select indicado e regista um log relativo a esta acção na tabela log_consultas.

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

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

Os stored procedures criados são de qualidade fraca, pois apenas um deles (criar medição) é relevante.

Tivemos também de acrescentar dois store procedure que são obrigatórios para o bom funcionamento do projeto que são o criar utilizador(investigador) e select all medições.

#### 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)
altera cultura		X		
cria cultura		X		
cria utilizador				X
cria mediçao		X		
select_all_medições				X
mostra cultura			X	
apaga cultura			X	
cria variavel			X	
select auditor			X	
Mostra_variaveis			X	
apaga variavel			X	

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

#### 1. Nome SP: altera\_cultura

Altera uma coluna da cultura selecionada através do id da mesma (não devolve boolean).

Código:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`altera_cultura`(id int, nome1 varchar(100), descricao1
varchar(100))
BEGIN
if (id IS NOT NULL) AND (nome IS NOT NULL ) AND
(descricao IS NOT NULL) THEN
UPDATE `sql_php`.`cultura` SET `nome` = nome1,
`descricao` = descricao1 WHERE (`idCultura` = id);
END IF;

if (id IS NOT NULL) AND (nome IS NULL ) AND (descricao IS
NOT NULL) THEN
UPDATE `sql_php`.`cultura` SET `descricao` = descricao1
WHERE (`idCultura` = id);
END IF;

if (id IS NOT NULL) AND (nome IS NULL ) AND (descricao IS
NULL) THEN
UPDATE `sql_php`.`cultura` SET `nome` = nome1 WHERE
(`idCultura` = id);

END IF;
END
```

#### 2. Nome SP: cria\_cultura

Cria a cultura com o investigador que chama o SP como responsável, assume-se que o 'CURRENT\_USER' corresponde ao campo e-mail. (Não devolve boolean).

Código:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`cria_cultura`(in nome varchar(45), in descrição
varchar(45))
BEGIN
declare email1 varchar(100);

SET email1 = (select email from utilizador where
```

```
current_user()=utilizador.nome);
```

```
insert into cultura(Utilizador_email, nome, descricao)  
VALUES  
(email1, nome, descrição);
```

```
END
```

### 3. Nome SP: cria\_medicao

Cria uma coluna na tabela medicao com input do tipo da medição.

*Código:*

```
CREATE DEFINER=`root`@`localhost` PROCEDURE  
`cria_medicao`(in valor decimal )  
BEGIN
```

```
    insert into medicao(datahora,valor)  
        values (tipo,now(),valor);
```

```
END
```

### 4. Nome SP: cria\_utilizador

Cria um utilizador na tabela utilizador.

*Código:*

```
CREATE DEFINER=`root`@`localhost` PROCEDURE  
`cria_utilizador`(in email Varchar(50), in nome  
varchar(100), in categ varchar(50), in tipo  
ENUM( 'ADMIN', 'INVESTIGADOR')  
BEGIN  
INSERT INTO utilizador(email, nome, categ, tipo) VALUES  
(email, nome, categ,tipo);  
END
```

### 5. Nome SP: select\_all\_medicao

Executa um select completo da tabela medições e adiciona ao log das medições a operação 'S' com as informações relativas ao select.

*Código:*

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
```

```
`select_all_medicões`  
BEGIN  
SELECT*  
FROM medicamentos;  
END
```

## 1.4 Migração entre Bases de Dados

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

<b>siddb log_consultas</b> idLog_Consultas : int(11) datahora : timestamp tabela_acedida : varchar(45) select : varchar(400)	<b>siddb log_medicoes</b> precedencia : int(11) datahora : timestamp operacao : varchar(45) idMedicao : int(11) tipo_sensor : enum('LUZ','TEMP') horaMedicao : timestamp valor : decimal(10,0)
<b>siddb log_variavel</b> idVariavel : int(11) nomevar : varchar(45) precedencia : int(11) datahora : timestamp operacao : varchar(45)	<b>siddb log_cultura</b> precedencia : int(11) datahora : timestamp operacao : varchar(45) idCultura : int(11) investigador : varchar(45) nome : varchar(45) descricao : varchar(45)
	<b>siddb log_investigador</b> precedencia : int(11) datahora : timestamp operacao : varchar(45) email : varchar(100) nome : varchar(100) categoria : varchar(45)

#### 1.4.1.1 *Apreciação Crítica e esquema relacional implementado*

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

Apresenta todas as tabelas logs com registos manuais e criadas na base de dados origem. Com uma tabela log para cada tabela original, existe uma maior organização dos registos, sendo mais fácil a sua consulta.

Tal como na base de dados origem, não existem registos log da tabela Sistema. Esta decisão é aceitável mas não existe justificação para tal.

De acordo com o esquema da base de dados origem, as medições automáticas feitas pelos sensores ficam registadas na tabela Medições. Assim haverá logs destas medições automáticas na tabela log\_medicoes. Achamos isto desnecessário, pois oferece informação irrelevante ao Auditor. Este apenas tem interesse em analisar registos de ações feitas manualmente.

##### Foram feitas alterações? (Sim/Não): Sim

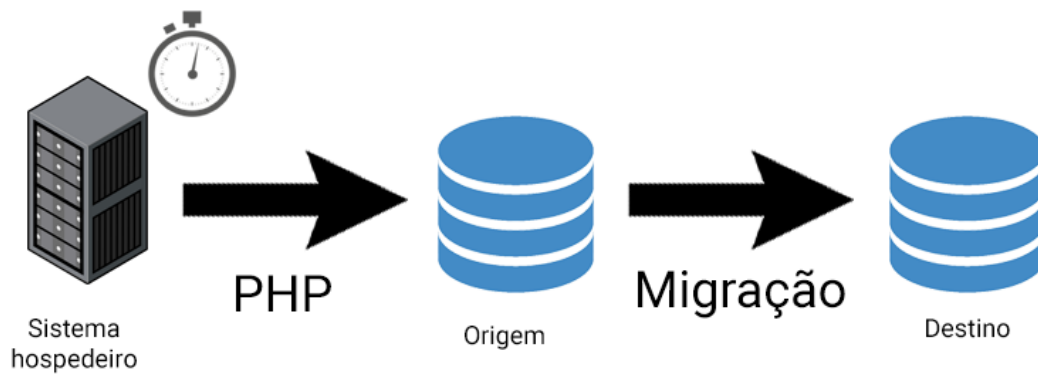


Tabela	Colunas e Tipos
log_cultura	precedencia INT(11) datahora TIMESTAMP operacao VARCHAR(1) idCultura INT(11) Utilizador_Email VARCHAR(45) nome VARCHAR(45) descricao VARCHAR(45)
log_medicoes	precedencia INT(11) datahora TIMESTAMP operacao VARCHAR(1) idMedicao INT(11) horaMedicao TIMESTAMP(6) valor DECIMAL(8,2) variavelmedida_idVariavel INT(11) variavelmedida_idCultura INT(11)
log_utilizador	precedencia INT(11) datahora TIMESTAMP operacao VARCHAR(1) email VARCHAR(100) nome VARCHAR(100) categoria VARCHAR(45) tipo ENUM(...)
log_variavel	idVariavel INT(11) nomevar VARCHAR(100) precedencia INT(11) datahora TIMESTAMP operacao VARCHAR(1)

Não incluímos a tabela log\_consultas uma vez que a sua utilização iria apenas ser realizada por um SP cuja utilização é questionável (select\_auditor).

### 1.4.2 Forma de Migração

A migração será feita automaticamente pelo sistema operativo hospedeiro com a ajuda do task scheduler, que irá executar um script escrito em PHP. A execução do script será feita periodicamente de uma em uma hora.



O script baseia-se na seleção dos dados da base de dados origem que foram adicionadas nas últimas 24 horas. De seguida verifica se dos dados seleccionados existem aqueles que ainda não foram migrados para a base de dados destino. Se for o caso então o script efetua a migração dos dados em falta.

A verificação da existência de dados na base de dados destino permite uma migração incremental, sem que haja nenhuma repetição de dados.



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

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

###### Pros:

- Garante que apenas dados novos sejam migrados, evitando repetição de registos.
- Utiliza o TaskScheduler do Windows para garantir a execução da migração automática e periodicamente, o qual preenche os requisitos do projeto.

###### Contras:

- O processo de migração consiste na verificação de quais são os registos presentes na BDO e não presentes na BDD, migrando aqueles que vão ao encontro destas duas condições. Este processo é feito a cada 1 hora. A verificação vai à procura dos registos feitos nas últimas 24h. Isto aparenta criar uma grande falta de eficiência e excesso de processamento, pois todas as horas, o programa terá de percorrer todos os registos log da BDO e todos os registos log da BDD feitos nas últimas 24 horas. Potencialmente, poderão existir demasiados registos para serem percorridos numa hora apenas. Isto também significa que cada registo será percorrido 24 vezes, para garantir se este foi migrado ou não. Para isto ser útil, um registo teria de falhar a migração 24 vezes. Esta verificação mais intensa poderia ser feita para garantir que possíveis falhas na migração nas últimas 24 horas fossem colmatadas. Mas não é apresentada justificação para tal.

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

Base de Dados (O/D)	Tabela	Tipo de Utilizador	
		Auditor Local	Auditor Remoto
O	Log_investigador	L	-
O	Log_cultura	L	-
O	Log_variavel	L	-
O	Log_medicoes	L	-
D	Log_investigador	-	L
D	Log_cultura	-	L
D	Log_variavel	-	L
D	Log_medicoes	-	L
	<b>Stored Proc.</b>		
	Migracao_auditor_local	X	-
	Migracao_auditor_remoto	-	X
	Select_remoto	-	X

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

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

Consideramos a utilização de dois tipos de Auditor desnecessária. Acesso de leitura aos logs na BDO poderia ser dado aos responsáveis desses mesmos dados (Investigador ou Administrador) para manutenção dos mesmos. Os registos logs são migrados para outra base de dados com o propósito de ter um Auditor (utilizador externo) a verificar possíveis falhas ou abusos nos registos. Não deveria ser concedido acesso à BDO a um utilizador do tipo Auditor. Se um Auditor pudesse verificar os logs na BDO, não seria necessário fazer migração.

##### Solução Implementada:

Tabela	Auditor
log_cultura	L
log_medicoes	L
log_utilizador	L
log_variavel	L

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

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

#### 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)**

	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
Nome Trigger (tal como especificado)				
Nome Trigger (tal como especificado)				
Nome Trigger (tal como especificado)				

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

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

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

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

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

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	BD (Origem ou Destino)	Muito breve descrição
Migracao_auditor_local	-	-	O	Faz um select das tabelas de logs relativamente as suas entradas nas últimas 24 horas.
Migracao_auditor_remoto	-	-	D	Faz um select das tabelas de logs relativamente as suas entradas nas últimas 24 horas.
Select_remoto	Varchar(100)-select	-	D	Faz o select indicado como parâmetro, registrando o mesmo na tabela de logs de consulta.

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

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

Os SPs criados para leitura dos logs selecionam todos os registos na BD de uma só vez, não sendo distinguível a que tabela cada registo pertence.

Mesmo que possível esta diferenciação, a agregação de todos os Logs num só SELECT vai contra a organização criada por ter uma tabela log para cada tabela original.

Um SP de leitura para cada tabela log seria mais lógico, tendo em conta o esquema da BDD.

##### **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)
Migracao_auditor_local			x	
Migracao_auditor_remoto		x		
Select_remoto			x	

#### ***1.4.5.2 Stored Procedures Implementados de suporte à migração de dados***

```
1. Nome SP: Migracao_auditor_remoto  
// Breve Descrição  
Código
```



#### 1.4.6 Eventos de suporte à migração de dados

Nome Evento	Local Execução (Origem ou Destino, ou Sistema Operativo)	Muito breve descrição
Execução do script PHP	Sistema Operativo	Executar o script da migração com ajuda do task scheduler

#### 1.4.6.1 *Apreciação Crítica de Eventos*

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

A criação do Evento no TaskScheduler para executar o script php responsável pela migração é o suficiente para que o processo ocorra. Nada a apontar.

##### **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)
Execução do script PHP	x			

#### 1.4.6.2 Eventos Implementados

1. Nome Evento: \_\_\_\_\_  
// Breve Descrição  
Código

2. Nome Evento: \_\_\_\_\_  
// Breve Descrição  
Código

3. Nome Evento: \_\_\_\_\_  
// Breve Descrição  
Código

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

```
$url = 'localhost';

$username = "root";

$conn = mysqli_connect($url, $username, $password);

if (!$conn) {

    die("ConnectionFailed: " . $conn->connect_error);

}

$sql = 'INSERT INTO auditordb.log_consultas

        SELECT * FROM siddb.log_consultas

        WHERE datahora >= NOW() - INTERVAL 1 DAY

        AND idLog_Consultas NOT IN (SELECT idLog_Consultas FROM

auditordb.log_consultas);

        INSERT INTO auditordb.log_cultura

        SELECT * FROM siddb.log_cultura

        WHERE datahora >= NOW() - INTERVAL 1 DAY

        AND idCultura NOT IN (SELECT idCultura FROM auditordb.log_cultura);

        INSERT INTO auditordb.log_investigador

        SELECT * FROM siddb.log_investigador

        WHERE datahora >= NOW() - INTERVAL 1 DAY

        AND email NOT IN (SELECT email FROM auditordb.log_investigador);

        INSERT INTO auditordb.log_medicoes

        SELECT * FROM siddb.log_medicoes

        WHERE datahora >= NOW() - INTERVAL 1 DAY

        AND idMedicao NOT IN (SELECT idMedicao FROM auditordb.log_medicoes);';

$result = mysqli_query($conn, $sql);

if($result) {

    echo "Dados migrados com sucesso";

} else {

    echo "Problema na migração";

}

mysqli_close($conn);
```

O script começa por estabelecer uma ligação com o host da base de dados. De seguida é executada uma query em SQL que vai buscar os dados à base de dados original que foram adicionadas nas últimas 24 horas. Caso existam dados que ainda não foram migrados, é feita a migração dos mesmos para a base de dados destino. Após a execução da query é verificada o sucesso da migração e é terminada a ligação com o host.

#### **1.4.7.1 Apreciação Crítica ao PHP especificado**

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

A lógica e o código apresentados seguem o mesmo raciocínio apresentado no ponto 1.4.2-"Forma de Migração", logo as críticas feitas nesse ponto mantêm-se. Ainda assim, o exemplo de código apresentado foi útil para a implementação da especificação apresentada.

#### 1.4.7.2 PHP Implementado

Código

```
<?php

$url = 'localhost';
$username = "root";
$password='';
$bdo="sql";
$bdd="bdd";

$conno = mysqli_connect($url, $username, $password,
$bdo);
$connd = mysqli_connect($url, $username, $password,
$bdd);
if (!$conno or !$connd ) {
    die("ConnectionFailed: " . $conno->connect_error);
    die("ConnectionFailed: " . $connd->connect_error);
}
else
{echo 'sucesso';}

$sql = 'INSERT INTO bdd.cultura_log
      SELECT * FROM sql.cultura_log
      WHERE data_operacao>= NOW() - INTERVAL 1 DAY
      AND id NOT IN (SELECT id FROM bdd.cultura_log)';

$sql1 = 'INSERT INTO bdd.utilizador_log
        SELECT * FROM sql.utilizador_log
        WHERE data_operacao>= NOW() - INTERVAL 1 DAY
        AND id NOT IN (SELECT id FROM
bdd.utilizador_log)';

$sql2 = 'INSERT INTO bdd.sistema_log
        SELECT * FROM sql.sistema_log
        WHERE data_operacao>= NOW() - INTERVAL 1 DAY
        AND id NOT IN (SELECT id FROM bdd.sistema_log)';

$sql3 = 'INSERT INTO bdd.variaveis_log
        SELECT * FROM sql.variaveis_log
        WHERE data_operacao>= NOW() - INTERVAL 1 DAY
        AND id NOT IN (SELECT id FROM
bdd.variaveis_log)';

$sql4 = 'INSERT INTO bdd.variaveismedidas_log
        SELECT * FROM sql.variaveismedidas_log
        WHERE data_operacao>= NOW() - INTERVAL 1 DAY
        AND id NOT IN (SELECT id FROM
```

```

bdd.variaveismedidas_log);';

$sql5 = 'INSERT INTO bdd.medicoes_log
        SELECT * FROM sql.medicoes_log
        WHERE data_operacao>= NOW() - INTERVAL 1 DAY
        AND id NOT IN (SELECT id FROM
bdd.medicoes_log);';

$result = mysqli_query($connd, $sql);

$result1 = mysqli_query($connd, $sql1);
$result2 = mysqli_query($connd, $sql2);
$result3 = mysqli_query($connd, $sql3);
$result4 = mysqli_query($connd, $sql4);
$result5 = mysqli_query($connd, $sql5);


if($result) {
    echo "Dados migrados com sucesso";
} else {
    echo "Problema na migração";
}

mysqli_close($conno);
mysqli_close($connd);

?>

```



### 1.5 Avaliação Global de especificações da Etapa A

<Texto avaliativo da qualidade e clareza das especificações recebidas. Referir a coerência, completude, nível de rigor e detalhe. Convém exemplificar afirmações>

A nível de clareza, completude e detalhe, podemos afirmar que o texto está fraco na medida em que na maior parte do trabalho não foi justificada qualquer decisão, pelo que impossibilitam uma possível interpretação e respetiva tentativa de implementação da nossa parte.

Uma vez que não existe informação contraditória ao longo do texto, podemos afirmar que apresenta boa coerência. A nível de rigor, constatamos que o grupo teve cuidado na apresentação do texto e tabelas em toda a extensão do trabalho, ainda que com alguns erros ortográficos e gralhas.

A nível global, qualificamos o trabalho como razoável, destacando como pontos positivos a especificação dos triggers\_log e dos eventos de migração e como pontos a melhorar uma base de dados disfuncional e a especificação dos SPs log e migração.

## Avaliação Global da Qualidade das Especificações recebidas

Avaliação (A,B,C,D,E) : C

Utilize a seguinte escala:

A: - 1 – 5 valores    B: 6 – 9 valores    C: 10 – 13 Valores D: 14 – 17 valores    E: 18 – 20 valores

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

As decisões não são devidamente justificadas, pelo que impossibilitam uma possível interpretação e respetiva tentativa de implementação da nossa parte.

Má interpretação e alteração de informação dada como certa pelo enunciado.

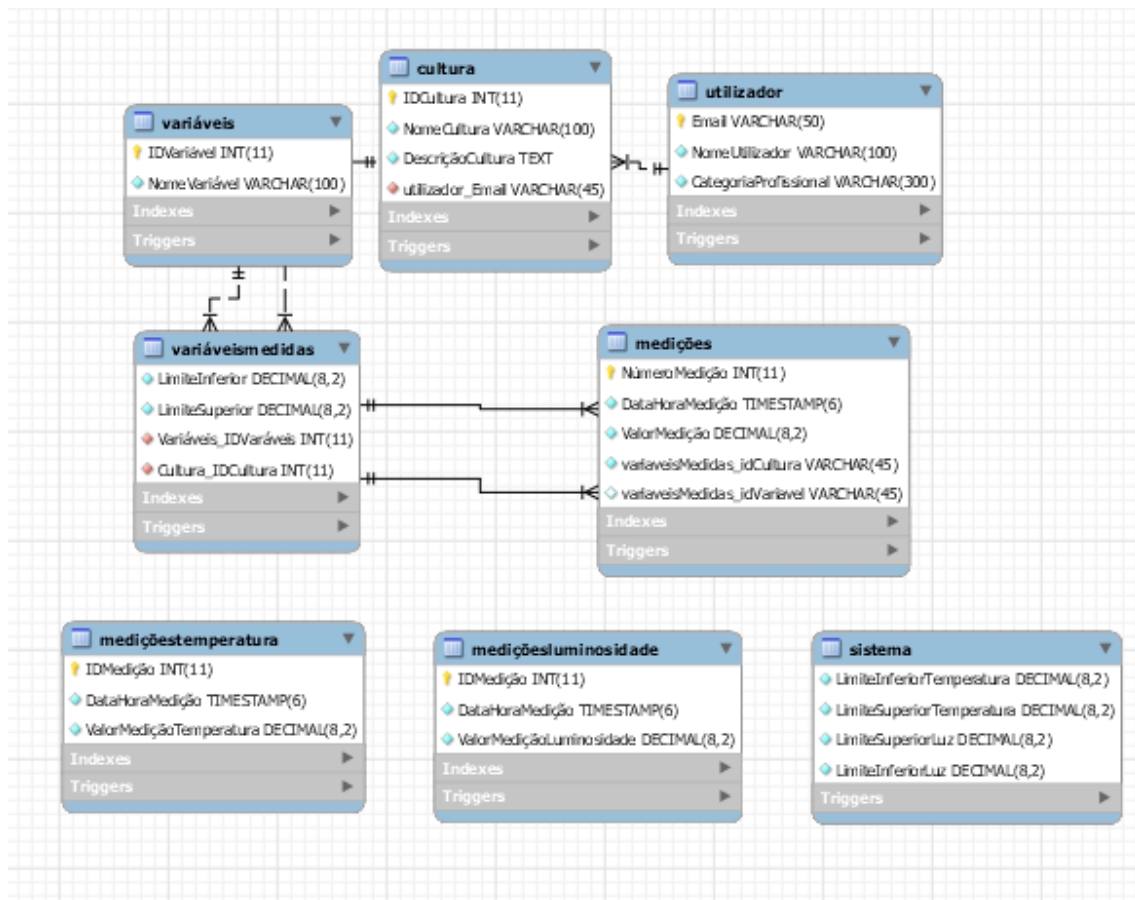
Base de dados disfuncional.

**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 Etapa C (Especificação e Implementação do Próprio Grupo)

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



Todos os atributos são not-null/mandatory (obrigatórios).

Para cada uma das chaves estrangeiras, vamos classificar a relação entre tabelas como cascade ou restrict: cascade - quando é feita uma alteração/delete na variável da tabela mãe as alterações/delete passam para a tabela filha;

restrict - alterações/delete feitas na tabela mãe não passam para a tabela filha).

Definimos a tabela Pai como a que oferece a chave primária e a tabela Filho como a que recebe a chave anterior como chave estrangeira.

No diagrama da Base de Dados Origem, foram identificadas 5 relações possíveis de classificar a sua integridade. Na relação entre as tabelas Utilizador - Cultura, onde Utilizador dá como chave estrangeira "Email",

se o valor da chave for alterado, é também alterado na tabela Cultura. Se for eliminado, a tabela Cultura mantém-se, ainda que o utilizador em questão já não conste na base de dados.

Consideramos que todos os Updates das chaves estrangeiras são definidos com Cascade para ser sempre mantida a relação de igualdade entre os valores das tabelas.

No caso do Delete, consideramos que os dados das tabelas Cultura e, por consequência, VariáveisMedidas têm um carácter mais relevante e central à base de dados. Sendo assim, a eliminação de dados de uma destas tabelas deve provocar a eliminação dos mesmos dados nas tabelas descendentes.

Isto não acontece com a eliminação de dados da tabela Variáveis, pois consideramos que mesmo após uma variável ser apagada, deve ser mantido o registo de uma medição associada a uma cultura sobre esta variável.

Tabela Pai	Tabela Filho	Chave Estrangeira	Update	Delete
Utilizador	Cultura	Email	Cascade	Restrict
Variável	VariávelMedida	ID_Variável	Cascade	Restrict
Cultura	VariávelMedida	ID_Cultura	Cascade	Cascade
VariávelMedida	Medição	ID_Variável	Cascade	Cascade
VariávelMedida	Medição	ID_Cultura	Cascade	Cascade

Table Name	Columns	Primary Key
<b>variáveis_log</b>	IDVariável INT(11), NomeVariável VARCHAR(100), utilizador VARCHAR(100), data_operacao TIMESTAMP(6), operacao VARCHAR(1), id INT(11)	id
<b>cultura_log</b>	IDCultura INT(11), NomeCultura VARCHAR(100), DescriçãoCultura TEXT, Utilizador_Email VARCHAR(50), utilizador VARCHAR(100), data_operacao TIMESTAMP(6), operacao VARCHAR(1), id INT(11)	id
<b>utilizador_log</b>	Email VARCHAR(50), NomeUtilizador VARCHAR(100), CategoriaProfissional VARCHAR(300), utilizador VARCHAR(100), data_operacao TIMESTAMP(6), operacao VARCHAR(1), id INT(11)	id
<b>variáveismedidas_log</b>	LimiteInferior DECIMAL(8,2), LimiteSuperior DECIMAL(8,2), Variáveis_IDVariáveis INT(11), Cultura_IDCultura INT(11), utilizador VARCHAR(100), data_operacao TIMESTAMP(6), operacao VARCHAR(1), id INT(11)	id
<b>medições_log</b>	NúmeroMedição INT(11), DataHoraMedição TIMESTAMP(6), ValorMedição DECIMAL(8,2), IDCultura VARCHAR(45), IDVariável VARCHAR(45), utilizador VARCHAR(100), data_operacao TIMESTAMP(6), operacao VARCHAR(1), id INT(11)	id
<b>sistema_log</b>	LimiteInferiorTemperatura DECIMAL(8,2), LimiteSuperiorTemperatura DECIMAL(8,2), LimiteSuperiorLuz DECIMAL(8,2), LimiteInferiorLuz DECIMAL(8,2), utilizador VARCHAR(100), data_operacao TIMESTAMP(6), operacao VARCHAR(1), id INT(11)	id

As tabelas log são preenchidas por triggers, através das tabelas de dados correspondentes.

A finalidade das tabelas log é registar toda a atividade das tabelas de dados correspondentes, logo, são um duplicado das tabelas originais, mas sem preenchimento obrigatório e com 5 novos campos extra:

ID: Identificador do registo;

Utilizador: Quem realiza a operação;

Operação: Qual a operação realizada (Insert, Update, Delete);

Data: Data em que a operação é realizada.

Todos os registos das tabelas log devem ser mantidos mesmo quando os dados correspondentes nas tabelas de dados são eliminados.

Dá-se a exceção dos logs da tabela Medições. Aqui será necessário guardar a informação de quem consultou esta tabela. Recorrendo a um Stored Procedure, será feita a leitura desta tabela e também o registo de quem executou este SP, guardando no campo `Operação` sobre o valor "S".

## 2.2 Especificação de Utilizadores

<b>Tabela</b>	<b>Administrador</b>	<b>Investigador</b>
<b>Cultura</b>	-	L,E
<b>Cultura_Log</b>	-	L
<b>Medições</b>	-	L,E
<b>Medições_Log</b>	-	L
<b>MediçõesLuminosidade</b>	L	L
<b>MediçõesTemperatura</b>	L	L
<b>Sistema</b>	L,E	L
<b>Sistema_Log</b>	L	L
<b>Utilizador</b>	L,E	L
<b>Utilizador_Log</b>	L	L
<b>Variáveis</b>	L,E	L
<b>Variáveis_Log</b>	L	L
<b>VariáveisMedidas</b>	L,E	L
<b>VariáveisMedidas_Log</b>	L	L

Stored Procedures	Administrador	Investigador
select_all_medições	-	X
cria_utilizador	X	-
apaga_utilizador	X	-
insere_medição	-	X

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

Na base de dados operacional teremos 2 tipos de utilizador: o Administrador e o Investigador.

O Administrador faz a manutenção de variáveis e utilizadores, pelo que tem acesso aos registos de todas as tabelas e permissão de escrita nas tabelas: “Sistema”, “Utilizador”, “Variáveis”, “VariáveisMedidas”. O Investigador faz a manutenção de culturas e medições, pelo que, também, tem acesso aos registos de todas as tabelas, mas permissão de escrita nas restantes: “Cultura”, “Medições”.

Nenhum dos dois tipos de utilizadores tem permissões de escrita nas tabelas: “MediçõesLuminosidade” e “MediçõesTemperatura” pois o preenchimento desta tabela é feito de forma automática, migrando os valores registados pelos sensores através do MongoDB. Não são dadas permissões de escrita nas tabelas de logs, pois isto iria comprometer a fiabilidade e legalidade dos registos que poderão ser lidos pelo Auditor.

No caso do acesso às tabelas Cultura e Medições, apenas o investigador responsável por uma certa cultura é que pode usufruir das permissões definidas.

Foram apenas implementados os stored procedures necessários para as tabelas “utilizador” e “medicao”. Os SPs “cria\_utilizador” e “apaga\_utilizador” só podem ser executados por administradores, pois são os responsáveis pela manutenção de utilizadores. O SP “insere\_medição” é apenas executável por Investigadores, pois são os responsáveis pelas medições das suas próprias culturas.

Foi criado uma tabela sistema\_log, pois, apesar de conter dados do sistema, a tabela Sistema tem permissões de escrita dadas ao Administrador, logo, contém dados que estão abertos a serem alterados. Assim, é relevante que exista uma tabela que guarde registos de alterações.

## 2.3 Especificação de Gestão de Logs

### 2.3.1 Triggers de suporte à gestão de logs

Nesta fase o grupo criou triggers after insert, after update e after delete para todas as tabelas cuja alteração fosse necessária registar em tabelas de logs.

As tabelas escolhidas foram a da cultura, medições, sistema, utilizador, variáveis e variáveis medidas. A lógica usada para cada uma das tabelas na construção dos triggers é igual para todas. Tomando como exemplo a tabela cultura, o trigger after insert garante que, após a inserção de dados na tabela cultura, será também inserido uma linha na tabela de logs com o devido registo e operação igual a "I" (insert).

Ao realizar-se um update este também é registado na tabela dos logs através do trigger after update, sendo inserido uma nova entrada com os novos valores e operação igual a "U" (update). Caso seja apagado alguma entrada da tabela cultura, o trigger after delete assegurará que na tabela dos logs estará uma nova entrada com a operação igual a "D" (delete).

Para além disto, cada trigger guarda na tabela dos logs a data a que ocorreu a ação, o responsável e um id.

Tendo isto em conta, o grupo não sentiu a necessidade da criação de triggers before visto que não trazia vantagens clara para este modelo.

Exemplo de Trigger After Update (Tabela Utilizador):

```
CREATE DEFINER=`root`@`localhost` TRIGGER `sql`.`utilizador_AFTER_UPDATE` AFTER UPDATE
ON `utilizador` FOR EACH ROW

BEGIN

declare new_id DOUBLE ;

select max(id) + 1 into new_id from mediçõesluminosidade_log;

insert into utilizador_log (Email, NomeUtilizador, CategoriaProfissional, utilizador,
data_operacao, operacao)

values (new.Email, new.NomeUtilizador, new.CategoriaProfissional, current_user, now(), "U");
```



```
if(old.Email<>new.Email) or (old.Email is Null and new.Email is NOT NULL) THEN

update utilizador_log set Email = new.Email where id = new_id;

end if;

if(old.NomeUtilizador<>new.NomeUtilizador) or (old.NomeUtilizador is Null and
new.NomeUtilizador is NOT NULL) THEN

update utilizador_log set NomeUtilizador= new.NomeUtilizador where id = new_id;

end if;

if(old.CategoriaProfissional<>new.CategoriaProfissional) or (old.CategoriaProfissional is Null
and new.CategoriaProfissional is NOT NULL) THEN

update utilizador_log set CategoriaProfissional = new.CategoriaProfissional where id = new_id;

end if;

END
```

Nome Trigger	Tabela	Tipo Operação	Evento
Cultura_AFTER_INSERT	Cultura	I	A
Cultura_AFTER_DELETE	Cultura	D	A
Cultura_AFTER_UPDATE	Cultura	U	A
medições_AFTER_INSERT	medições	I	A
medições_AFTER_DELETE	medições	D	A
medições_AFTER_UPDATE	medições	U	A
Sistema_AFTER_INSERT	Sistema	I	A
Sistema_AFTER_DELETE	Sistema	D	A
Sistema_AFTER_UPDATE	Sistema	U	A
Utilizador_AFTER_INSERT	Utilizador	I	A
Utilizador_AFTER_DELETE	Utilizador	D	A
Utilizador_AFTER_UPDATE	Utilizador	U	A
Variaveis_AFTER_INSERT	Variáveis	I	A
Variaveis_AFTER_DELETE	Variáveis	D	A
Variaveis_AFTER_UPDATE	Variáveis	U	A
VariaveisMedidas_AFTER_INSERT	Variáveis Medidas	I	A
VariaveisMedidas_AFTER_DELETE	Variáveis Medidas	D	A
VariaveisMedidas_AFTER_UPDATE	Variáveis Medidas	U	A

### 2.3.2 Stored Procedures de suporte à gestão de logs

Nesta fase, foram criados Stored Procedures de modo a poder ser possível visualizar nas tabelas de logs quem fez um select, funcionando por isso como um after select.

Foi realizada a criação de SPs que permitem o select de toda a tabela e o devido registo na tabela de log respetiva com a data da operação, o utilizador que realizou o select, id e operação igual a 'S'.

Em outros casos foram dados parâmetros de entrada, para que haja uma maior filtragem das linhas da tabela e uma pesquisa mais objetiva, resultando em parâmetros de saída mais específicos e que ajudam o utilizador a selecionar melhor a informação que deseja ver.

Exemplo de SP (criar\_medição):

```
CREATE DEFINER='root'@'localhost' PROCEDURE `cria_medicao` (in tipo enum('LUZ','TEMP'), in  
valor decimal )
```

```
BEGIN
```

```
insert into medicao(Sensor_tipo,datahora,valor)
```

```
values (tipo,now(),valor);
```

```
END
```

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	Breve descrição
select_all_medições	-	-	Seleciona todas as colunas da tabela medições
cria_utilizador	Email, NomeUtilizador, CategoriaProfissional, TipoUtilizador		Recebe como argumento dados, inseridos manualmente, de um utilizador e insere-os na tabela Utilizador
apaga_utilizador	Email		Recebe como argumento o Email de um utilizador e apaga os dados do próprio da tabela Utilizador
insere_medição	ValorMedição		Recebe como argumento o valor de uma medição inserida manualmente e insere-a na tabela Medições

## 2.4 Avaliação da especificação do próprio grupo Gestão de Logs

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

Justificação:

<fazer um resumo dos principais pontos fracos e fortes.  
Depois de ler esta secção o leitor deve ter uma visão  
sobre que secções estavam mais fracas (triggers? Base de  
dados?)>

## 2.5 Implementação Gestão de Logs

### 2.5.1 Utilizadores implementados

Tabela	Administrador	Investigador
Cultura	-	L,E
Cultura_Log	-	L
Medições	-	L,E
Medições_Log	-	L
MediçõesLuminosidade	L	L
MediçõesTemperatura	L	L
Sistema	L,E	L
Sistema_Log	L	L
Utilizador	L,E	L
Utilizador_Log	L	L
Variáveis	L,E	L
Variáveis_Log	L	L
VariáveisMedidas	L,E	L
VariáveisMedidas_Log	L	L

## 2.5.2 Lista de Triggers

<b>Lista de Triggers (para cada trigger assinalar com x em célula correspondente)</b>				
	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
Cultura_AFTER_INSERT	x			
Cultura_AFTER_DELETE	x			
Cultura_AFTER_UPDATE	x			
Medições_AFTER_INSERT	x			
Medições_AFTER_DELETE	x			
Medições_AFTER_UPDATE	x			
Sistema_AFTER_INSERT	x			
Sistema_AFTER_DELETE	x			
Sistema_AFTER_UPDATE	x			
Utilizador_AFTER_INSERT	x			
Utilizador_AFTER_DELETE	x			
Utilizador_AFTER_UPDATE	x			
Variaveis_AFTER_INSERT	x			
Variaveis_AFTER_DELETE	x			

Variaveis_ AFTER_UPDA TE	x			
VariaveisM edidas_AFT ER_INSERT	x			
VariaveisM edidas_AFT ER_DELETE	x			
VariaveisM edidas_AFT ER_UPDATE	x			



### 2.5.3 Triggers Implementados

1. Nome Trigger: Cultura\_AFTER\_INSERT

Insert na tabela dos logs.

Código

2. Nome Trigger: Cultura\_AFTER\_DELETE

Delete na tabela dos logs.

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`cultura_AFTER_DELETE` AFTER DELETE ON `cultura` FOR EACH
ROW BEGIN
insert into cultura_log (IDCultura, NomeCultura,
DescriçãoCultura, Utilizador_Email, utilizador,
data_operacao, operacao) VALUES
(old.IDCultura,
old.NomeCultura,old.DescriçãoCultura,old.Utilizador_Email
, CURRENT_USER, now() , 'D') ;
END
```

3. Nome Trigger: Cultura\_AFTER\_UPDATE

Update na tabela dos logs.

#### 2.5.4 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)
Select_all_medicoes	x			
Criar_utilizador	x			
Apagar_utilizador	x			
Insere_medicao	x			

### 2.5.5 Stored Procedures Implementados

1. Nome SP: `select_all_medicoes`

Seleciona todas as colunas da tabela `medicao`.

Código:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`select_all_medicoes`(in tipo enum('LUZ','TEMP'))
BEGIN
SELECT*
FROM medicões;
END
```

2. Nome SP: `criar_utilizador`

Cria um novo utilizador e insere-o na tabela `utilizador`.

Código:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`cria_utilizador`(in mail Varchar(50), in nome
varchar(100), in categp varchar(50), in tipo varchar(50))
BEGIN
INSERT INTO utilizador (Email, NomeUtilizador,
CategoriaProfissional, TipoUtilizador) VALUES
(mail, nome, categp,tipo);
END
```

3. Nome SP: `apagar_utilizador`

Apaga utilizador.

Código:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`apagar_utilizador`(in mail Varchar(50))
BEGIN
delete from utilizador
where mail=Email;
END
```

4. Nome SP: `insere_Medicao`

Insert na tabela `medicao`.

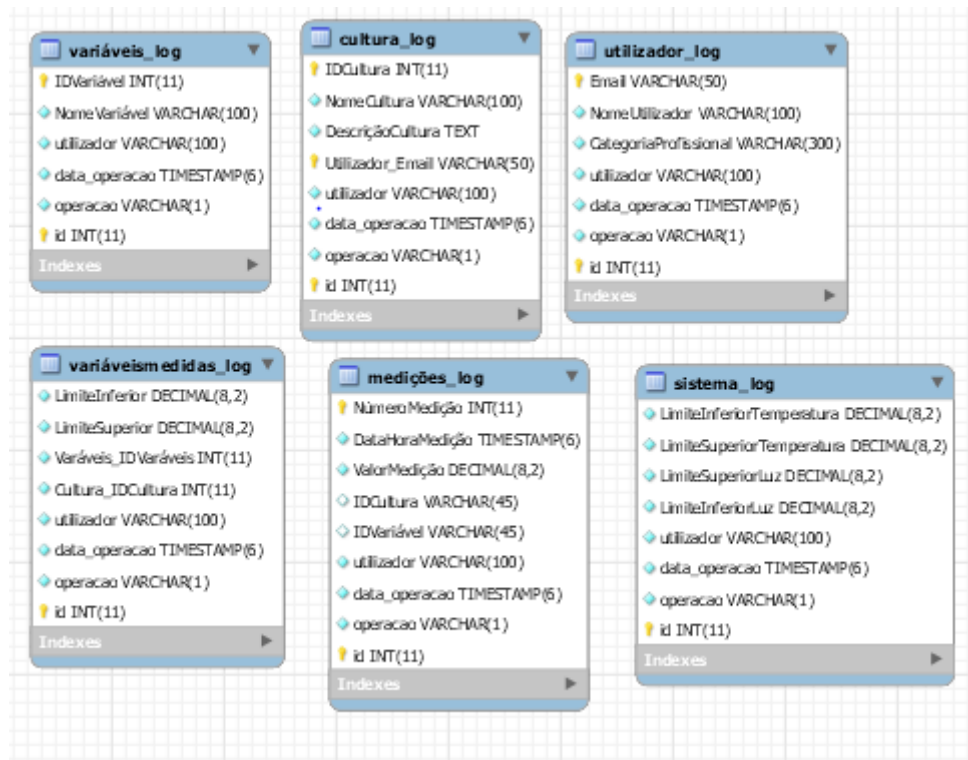
Código:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
```

```
`insere_medicao`(in valor decimal(8,2))  
BEGIN  
INSERT INTO medicoes (DataHoraMedicao, ValorMedicao)  
VALUES (now(), valor) ;  
END
```

## 2.6 Especificação de Migração entre Bases de Dados

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



## 2.6.2 Forma de Migração Especificada

### Requisitos para migração:

Para o processo de migração será necessária a instalação do programa Xampp, para o uso dos serviços Apache e MySQL do próprio. Isto criará um servidor de base de dados para o MySQL. Será também utilizado o Task Scheduler do Windows.

### Tipo de Ficheiro:

Foi considerada a utilização de dois tipos de ficheiros para este processo: xml ou csv.

As grandes vantagens do CSV sobre o XML é que o primeiro é legível e fácil de editar manualmente, o que por sua vez torna-se mais simples de implementar e analisar que o segundo.

Para além disso, o CSV ocupa menos espaço que o XML, o que o torna mais fácil de gerar e manter. Quanto ao XML, a sintaxe é redundante quando se trata de dados tabulares o que irá afetar a eficiência do processo.

Apesar do XML suportar Unicode (permite qualquer linguagem humana escrita seja comunicada), de poder representar estruturas de dados de ciência da computação (registos, listas e árvores) e de ter requisitos rigorosos de sintaxe, estas características não acrescentariam benefícios em relação ao CSV, tendo em conta as nossas necessidades.

XML contribuiria para uma leitura facilitada a nível do utilizador, mas considerando que será uma máquina a processar o ficheiro e a traduzi-lo para a base de dados, este fator também se torna irrelevante.

Considerando todos os fatos e necessidades, o csv torna-se mais benéfico e eficiente.

### Exportação:

A exportação será realizada sobre todas as tabelas de Logs: cultura\_log; medições\_log; sistema\_log; utilizador\_log; variaveis\_log; variaveismedidas\_log.

Sendo assim, existirá um Stored Procedure para cada tabela de logs, responsável pela exportação da própria tabela. Por exemplo: SP exporta\_cultura\_log() exportará apenas a informação na tabela cultura\_log.

Tendo isto em conta, cada SP irá gerar um ficheiro .csv para a tabela a ele atribuída. Por exemplo: SP export\_cultura\_log() gera o ficheiro "n\_cultura\_log.csv".

Um ficheiro .csv para cada tabela exportada.

No SP, estará declarada uma variável que guarda o último id exportado da tabela respetiva. Assim, apenas será exportada informação com id maior que o valor da variável, evitando envio de entradas previamente exportadas. Isto pode ser representado pelo exemplo de código:

```
declare ultimoid int default 0;
```

```
select * from cultura_log where id_log > @ultimoid
```

```
...
```

```
set @ultimoid := (select max(id_log) from cultura_log);
```

O procedimento começa com a verificação da existência de um id na tabela maior que o último id exportado. Isto para saber se existiu alguma inserção de dados desde a última exportação. Caso não existam dados novos, não há necessidade do procedimento continuar.

No caso de existirem dados novos:

- é selecionada toda a informação com um id maior que o guardado na variável;
- os dados selecionados são exportados para um ficheiro .csv;
- verifica-se o valor do id do último dado exportado e guarda-se este valor na variável.

O nome de cada ficheiro será criado de forma dinâmica - “n\_NomeTabela.csv” - em que “NomeTabela” representa a tabela de logs que está a ser exportada e “n” representa o id do ficheiro que será incrementado automaticamente. A importância deste id será explicada no ponto Importação.

A geração do nome pode ser feita da seguinte forma:

```
declare numero int;
```

```
set @filename= '_CulturaLog.csv';
```

```
set @comando = 'select * from `cultura_log` into outfile ""';
```

```
set @numero := @numero+1;
```

```
SET @statement =
```

```
CONCAT(@comando, @numero, @filename);
```

```
select @statement;
```



```
PREPARE s1 FROM @statement;
```

```
EXECUTE s1;
```

```
DROP PREPARE s1;
```

Este código consiste na criação de duas variáveis de texto - uma com o comando de exportação, outra com o nome do ficheiro - e outra variável INT, que guarda o numero associado ao ficheiro. A função CONCAT junta todas estas variáveis numa string. É necessária a utilização de Prepared Statements para executar o que estiver presente na string anteriormente criada.

É pretendido que todos estes passos sejam executados de forma periódica e automática. Logo, é necessário a criação de um evento MySQL. Aqui é declarado o período de tempo entre cada execução e é chamada a execução de cada SP. Foi decidido que um período de 24h é o mais indicado, tendo em conta que, desta forma, estaremos a exportar os dados recolhidos de cada dia de trabalho.

O evento pode ser criado com o seguinte exemplo de código:

```
create event export
```

```
on schedule every 24 hour
```

```
do
```

```
call_sp()
```

### Importação:

\_\_\_\_\_ Esta fase da migração será baseada na utilização de um ficheiro .bat e na criação de uma tarefa Windows no Task Scheduler, que auxiliará na execução automática e periódica deste ficheiro .bat. Neste ficheiro existem 3 tipos de comandos para a execução deste processo. Cada tipo de comando terá uma iteração para cada ficheiro importado. A execução seguirá a seguinte lógica (segundo o exemplo da importação do ficheiro “CulturaLog.csv”):

-O primeiro comando identifica todos os ficheiros com dados da tabela CulturaLog e junta-os num único ficheiro. Neste passo nota-se a importância de cada ficheiro ter um nome dinâmico, como descrito no ponto Exportação (e.g. “3\_CulturaLog.csv”). Para cada exportação, dá-se uma importação. Mas na eventualidade de existir uma falha na importação, o objetivo é preservar o ficheiro não importado, deixando-o disponível para importação na próxima tentativa. Imaginando o caso em que há uma falha a importar o ficheiro “3\_CulturaLog.csv”. A próxima exportação irá gerar o ficheiro “4\_CulturaLog.csv” na mesma diretoria que o anterior. Na próxima tentativa de importação, este comando irá juntar os dois ficheiros previamente mencionados, gerando o ficheiro CulturaLog.csv. Isto pode ser feito com o código exemplo (comando Windows):

```
c:\directory> copy *CulturaLog.csv CulturaLog.csv
```

-O segundo comando executará um script com linguagem MySQL com o simples propósito de importar os dados no ficheiro para a tabela presente na Base de dados Destino respetiva e a formatação necessária aos dados, tendo em conta que é um ficheiro csv. Um exemplo de código útil para esta implementação seria:

```
LOAD DATA INFILE("C:\directoria\CulturaLog.csv")
```

```
INTO TABLE cultura_log
```

```
FIELDS TERMINATED BY ','
```

```
ENCLOSED BY '\"
```

```
LINES TERMINATED BY '\n'
```

-Terceiro e último comando presente no ficheiro .bat servirá para eliminar os ficheiros já importados, completando assim o processo. Este passo é necessário por várias razões. Se estes não fossem apagados, a informação já importada seria enviada de novo, junta com os novos dados.

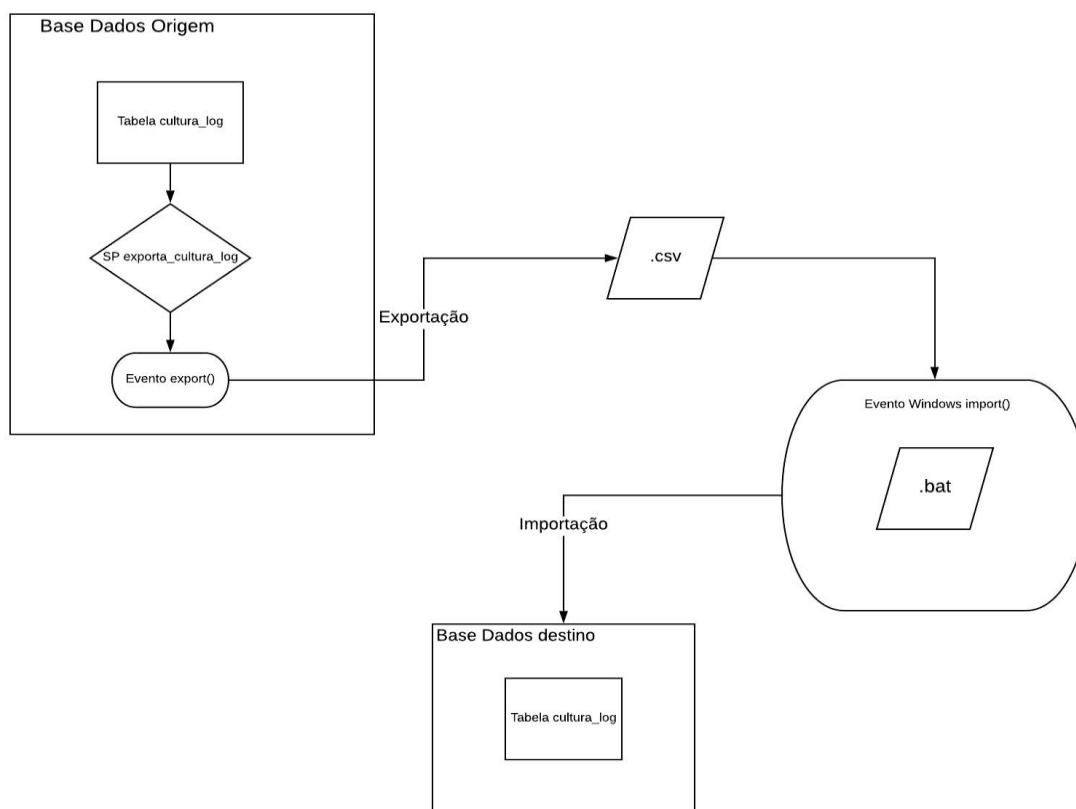
Isto também providencia uma maior segurança de dados, visto que seria perigoso existirem ficheiros com estes dados que poderiam ser acedidos sem qualquer tipo de restrições. Por último, isto também liberta espaço em disco, sendo que seria inútil manter os dados em ficheiro, sabendo que estes mesmos dados já se encontram onde se devem encontrar.

Será criada uma tarefa no Windows Task Sheduler. Esta tarefa executará o ficheiro .bat com a mesma periodicidade de 24h que tem o evento de exportação. De notar que esta tarefa só deve ser executada após o processo de exportação. Também reparar que para cada ficheiro importado, é aplicada a mesma lógica de comandos, como descrito no exemplo em cima.

Reparos gerais:

Todos estes passos garantem um processo automático, com um período bem definido. A eliminação de ficheiros após importação oferece uma maior segurança e privacidade de dados. Este processo é também eficiente, pois garante a chegada de todos os dados à BD destino, sempre enviando apenas dados novos.

Diagrama do processo de Migração (segundo o exemplo da tabela “cultura\_log”):



### 2.6.3 Utilizadores Especificados

Tabela	Auditor
Cultura_Log	L
Medições_Log	L
Sistema_Log	L
Utilizador_Log	L
Variáveis_Log	L
VariáveisMedidas_Log	L

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

A Gestão de utilizadores na base de dados destino, incide apenas sobre um tipo de utilizador, o Auditor.

O Auditor tem permissões para consultar qualquer registo em todas as tabelas log, de forma a certificar a conformidade dos dados.

#### 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	Descrição
exporta_cultura_log			Origem	Exporta dados da tabela `cultura_log` para o ficheiro respetivo
exporta_utilizador_log			Origem	Exporta dados da tabela `utilizador_log` para o ficheiro respetivo
exporta_variaveis_log			Origem	Exporta dados da tabela `variaveis_log` para o ficheiro respetivo
exporta_variaveismedidas_log			Origem	Exporta dados da tabela `variaveismedidas_log` para o ficheiro respetivo
exporta_medicoes_log			Origem	Exporta dados da tabela `medicoes_log` para o ficheiro respetivo
exporta_sistema_log			Origem	Exporta dados da tabela `sistema_log` para o ficheiro respetivo



### 2.6.6 Eventos de suporte à migração de dados especificados

Nome Evento	Local Execução	Descrição
exporta_geral	Origem	Executa todos os Stored Procedures responsáveis pela exportação de cada tabla
importa_geral	Sistema Operativo	Executa o ficheiro .bat responsável pela importação

### 2.6.7 PHP de suporte à migração de dados especificado

## 2.7 Avaliação das especificações do próprio grupo Migração

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

A qualidade da especificação da migração é razoável uma vez que garante a migração periódica incremental na ausência de falha de sistema ou eventos externos, mas tem problema de robustez e flexibilidade.

## 2.8 Implementação da Migração de Dados

### 2.8.1 Utilizadores Implementado

Tabela	Auditor
Cultura_Log	L
Medições_Log	L
Sistema_Log	L
Utilizador_Log	L
Variáveis_Log	L
VariáveisMedidas_Log	L

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	BD	Descrição
exporta_cultura_log			Origem	Exporta dados da tabela `cultura_log` para o ficheiro respetivo
exporta_utilizador_log			Origem	Exporta dados da tabela `utilizador_log` para o ficheiro respetivo
exporta_variaveis_log			Origem	Exporta dados da tabela `variaveis_log` para o ficheiro respetivo

exporta_variaveismedidas_log			Origem	Exporta dados da tabela `variaveismedidas_log` para o ficheiro respetivo
exporta_medicoes_log			Origem	Exporta dados da tabela `medicoes_log` para o ficheiro respetivo
exporta_sistema_log			Origem	Exporta dados da tabela `sistema_log` para o ficheiro respetivo

## 2.8.2 Lista Triggers

<b>Lista de Triggers (para cada trigger assinalar com x em célula correspondente)</b>				
	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
Nome Trigger (tal como especificado)				
Nome Trigger (tal como especificado)				
Nome Trigger (tal como especificado)				

### 2.8.3 Triggers Implementados

1. Nome Trigger: \_\_\_\_\_  
*// Breve Descrição*  
*Código*

2. Nome Trigger: \_\_\_\_\_  
*// Breve Descrição*  
*Código*

3. Nome Trigger: \_\_\_\_\_  
*// Breve Descrição*  
*Código*

#### 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)
Exporta_cultura_log	x			
Exporta_utilizador_log	x			
Exporta_variaveis_log	x			
Exporta_variaveismedidas_log	x			
Exporta_medicoes_log	x			
Exporta_sistema_log	x			



### 2.8.5 Stored Procedures Implementados

1. Nome SP: `exporta_cultura_log`

Exporta os dados da tabela `cultura_log` para um ficheiro.

Código:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`exporta_cultura_log`()
BEGIN
    set @ultimoid=0;
    if @numero is null then set @numero=0; end if;

    set @command1 = 'select * from cultura_log where id>';

    set @nome_ficheiro = '_CulturaLog.csv' FIELDS TERMINATED
    BY "," ENCLOSED BY "\"" LINES TERMINATED BY "\n" ;';
    set @conc = concat (@command1, @ultimoid, ' into outfile
    "',@numero, @nome_ficheiro);

    prepare s1 from @conc;
    execute s1;

    set @numero=@numero+1;
    set @ultimoid=(select max(id) from cultura_log);
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)
Exporta_geral	x			
Importa_geral	x			

### 2.8.7 Eventos Implementados

1. Nome Evento: `exporta_geral`

Chama todos os sps responsáveis pela migração.

*Código*

```
Delimiter //
create event exporta_geral
on schedule
  every 24 hour
  do

  begin

call exporta_cultura_log();
call exporta_utilizador_log();
call exporta_sistema_log();
call exporta_medicoes_log();
call exporta_variaveis_log();
call exporta_variaveismedidas_log();
end //
Delimiter ;
```

### 2.8.8 PHP Implementado

*Código*

## Avaliação Global da Qualidade das Especificações do próprio grupo

Avaliação (A,B,C,D,E) : C

Utilize a seguinte escala:

A: - 1 – 5 valores    B: 6 – 9 valores    C: 10 – 13 Valores    D: 14 – 17 valores    E: 18 – 20 valores

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

Ausência de auxílio na criação de Utilizadores no sistema.

SPs desnecessários ao funcionamento do projeto.

Ausência de definição dos privilégios aos Utilizadores no sistema.

**Resumo de Avaliações de Qualidade Anteriores (para cada linha assinalar com x em célula correspondente)**

	Fraco	Razoável	Bom	Muito Bom
BD Sybase				
Triggers Log		X		
SP Log		X		
Utilizadores Log			X	
BD Mysql			X	
Forma Migração		X		
Triggers Migração				
SP Migração			X	
Eventos Migração			X	
Utilizadores Migração				
PHP Migração				

## *2.9 Comparação de Implementações (ficheiro versos PHP)*

O processo de migração por php é mais facilmente implementável e compreensível. Apenas é necessário estabelecer uma ligação entre duas bases de dados e o resto do processo tem maior nível de autonomia, pois o acesso entre elas é recíproco. Uma desvantagem deste processo será a necessidade destas bases de dados se manterem online/disponíveis para ligação, apresentando dependência entre elas.

Migração por ficheiro requer maior elaboração e planeamento do processo, para permitir uma maior robustez e segurança. Uma vantagem deste processo é não necessitar de uma ligação estabelecida entre as BDs, não existindo dependência entre elas. Cada uma das BDs funciona de acordo com a própria configuração, existindo maior liberdade de estrutura do processo. Uma desvantagem será a complexidade necessária para que o processo definido ocorra de acordo com o desejado ou ideal (Maior robustez e flexibilidade).

### 2.9.1 Eficiência de Migração

De acordo com os nossos testes, obtivemos resultados muito idênticos entre as duas formas de migração.

A eficiência na migração por php está dependente da velocidade da ligação enquanto que a eficiência na migração por ficheiro pode estar dependente da máquina que executa o teste (velocidade de processamento, tipo de disco, etc). Uma diferença possivelmente identificada poderia ser causado por os fatores mencionados.



Testes php	
Registos	Tempo Total
50000	1,71
100000	3,85
250000	9,67
500000	18,38
1000000	33,44



**Testes Ficheiro**

Registos	Tempo Export	Tempo Import	Tempo Total
50000	0,719	0,9	1,619
100000	0,922	4,5	5,422
250000	2,6	7,296	9,896
500000	4	14,014	18,014
1000000	5,45	29,2	34,65



### 2.9.2 Robustez

No caso de php, a ligação apenas retomaria na próxima execução do evento de migração. O processo de migração apenas verifica os dados não enviados nas últimas 24h. Se uma BD fosse abaixo durante mais que esse período, haveria informação não migrada.

Na migração de ficheiro, a robustez da implementação encontra-se fraca, pois caso o ficheiro que contém os dados para migrar fosse apagado, este não seria recuperado e a informação nunca seria migrada. A implementação apenas resolve falhas na importação, e não desaparecimento do ficheiro. Desta forma, a especificação ficheiro será atualizada da seguinte resumida forma, para melhorar este parâmetro:

Alterações BDs:

-bdo:

- . Criar tabela "ids\_importados\_NomeTabela" (1 para cada log, ex: ids\_importados\_cultura).

- . Adicionar coluna "migração" a cada tabela log. Valor boolean.

-bdd:

- . Criar tabela "ultimo\_id", uma coluna para cada tabela log, guarda ultimo id confirmado.

- . Adicionar coluna "id\_log\_bdo" a cada tabela log. Guarda valor do id que registo tinha na bdo.

Alterações Forma de Migração:

-bdo:

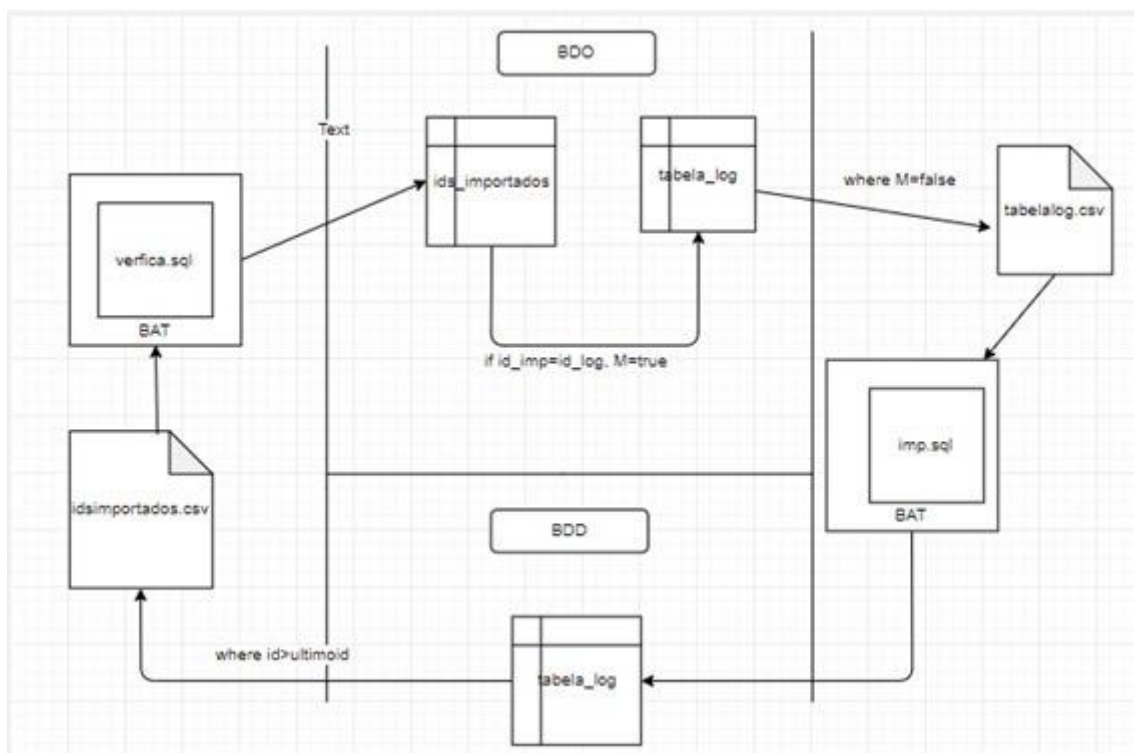
- . Sps de exportação- Exporta Migrações=false em vez de exportar desde ultimo id exportado.

- . Importação- Criar script "verifica.sql", que quando executado num ficheiro bat semelhante ao bat da migração original, importa os dados presentes nos ficheiros "ids\_importados\_NomeTabela.csv" para as tabelas "ids\_importados\_NomeTabela". Compara os ids recém chegados com os ids das tabelas Log respetivas e passa todos os valores Migração=False para True. Por fim apaga os registos

nas tabelas "ids\_importados\_NomeTabela" para estar pronto a receber novos ids importados e fazer novas verificações apenas com ids importados mais recentemente.

-bdd:

.Importação- Criar SP "exporta\_ids\_recebidos" que exporta os valores "id\_log\_bdo" de cada tabela log para um ficheiro "ids\_importados\_NomeTabela.csv", respetivamente. Reconhece o último id exportado, acedendo à tabela "ultimo\_id", de forma a verificar apenas novas entradas. Atualiza ultimo id respetivo. Criar Sp "Dups" que remove todas as entradas com ids duplicadas. Chamar estes dois novos SPs dentro do script de importação já existente.



### 2.9.3 Flexibilidade / Dependência

A nível da flexibilidade, ambas as implementações apresentam um baixo nível, não dando opções de mudança sobre certas acções.

A nível da dependência, a implementação ficheiro apresenta baixo nível ou mesmo independência, pois o comportamento da BDO não depende da BDD e vice-versa.

Com o php, existe dependência, pois na parte da migração, a ligação entre as duas depende da disponibilidade de ambas.

Uma das melhorias a fazer a nível da flexibilidade seria criar um SP que recebe como argumento um INT e uma medida de TEMPO para definir os parâmetros da periodicidade do evento de migração.

Para além de alterar a periodicidade, poderia ser criado um ficheiro bat que corre o processo de migração de imediato.

#### 2.9.4 Segurança

As fragilidades na migração por php surgem de possíveis intrusos nas BDs onde se encontram os dados. Este será o ponto mais crítico, pois são os únicos sítios onde estes se encontram. No caso da migração por ficheiro, os dados encontram-se nas BDs mas também nos ficheiros com os dados à espera de serem migrados. É possível que seja mais fácil aceder à máquina que tem estes ficheiros do que entrar, sem permissões, nas BDs. Logo, a informação está mais vulnerável a ser alterada, apagada ou mesmo roubada. Serão necessários mais contingências de segurança no caso de ficheiro do que no caso de php.

## *2.10 Auditoria de Dados (base de dados origem)*