


# 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): \_\_7\_\_

Número	Nome	Foto
73051	Diogo Juvandes	
73498	Alexandre Costa	
73528	Miguel Penedo	
78337	Marcos Teixeira	
79038	Diogo Toupa	

77983	Tiago Garção	
Especificação: PHP <input type="checkbox"/> Ficheiro <input checked="" type="checkbox"/>		

Identificação do grupo autor da implementação (Etapas B e C): 13

Número	Nome	Foto
65598	André Almeida	
78463	Diogo Sarmento	
78001	Hugo Cruz	
78009	Luís Fernandes	
77577	Miguel Figueiredo	

77689	Rostislav Andreev	
<p>             Especificação: PHP <input checked="" type="checkbox"/> Ficheiro <input type="checkbox"/> </p> <p>             Implementação: PHP <input checked="" type="checkbox"/> Ficheiro <input type="checkbox"/> </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 actualizado;
- 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

## Conteúdo

1	Etapa A e B	11
1.1	Esquema relacional da base de Dados Mysql (origem)	11
1.1.1	Apreciação Crítica e esquema relacional implementado	15
1.2	Utilizadores Base de Dados de Origem	16
1.2.1	Apreciação Crítica a Gestão de Utilizadores Base de Dados de Origem	19
1.3	Gestão de Logs	20
1.3.1	Triggers de suporte à criação de logs Base de Dados de Origem	20
1.3.2	Stored Procedures de suporte à criação de logs ( <b>se relevante</b> )	26
1.4	Migração entre Bases de Dados	31
1.4.1	Esquema relacional da base de Dados Mysql (destino)	31
1.4.2	Forma de Migração	33
1.4.3	Gestão de Utilizadores de Suporte à Migração (origem e/ou destino)	42
1.4.4	Triggers de suporte à migração de dados ( <b>se relevante</b> )	44
1.4.5	Stored Procedures de suporte à migração de dados	47
1.4.6	Eventos de suporte à migração de dados	50
1.4.7	PHP suporte à migração de dados (se relevante)	53
1.5	Avaliação Global de especificações da Etapa A	56
2	Etapa C (Especificação e Implementação do Próprio Grupo)	58
2.1	Especificação do Esquema relacional da base de Dados Origem	58
2.2	Especificação de Utilizadores	59
2.3	Especificação de Gestão de Logs	60
2.3.1	Triggers de suporte à gestão de logs	60
2.3.2	Stored Procedures de suporte à gestão de logs	61
2.4	Avaliação da especificação do próprio grupo Gestão de Logs	62
2.5	Implementação Gestão de Logs	63
2.5.1	Utilizadores implementados	63
2.5.2	Lista de Triggers	64
2.5.3	Triggers Implementados	65
2.5.4	Lista de Stored Procedures	66
2.5.5	Stored Procedures Implementados	67
2.6	Especificação de Migração entre Bases de Dados	68
2.6.1	Esquema relacional da base de Dados Mysql especificada (destino)	68
2.6.2	Forma de Migração Especificada	69
2.6.3	Utilizadores Especificados	70
2.6.4	Triggers de suporte à migração de dados especificados	71
2.6.5	Stored Procedures de suporte à migração de dados especificados	72
2.6.6	Eventos de suporte à migração de dados especificados	73
2.6.7	PHP de suporte à migração de dados especificado	74

2.7	Avaliação das especificações do próprio grupo Migração	75
2.8	Implementação da Migração de Dados	76
2.8.1	Utilizadores Implementado	76
2.8.2	Lista Triggers	77
2.8.3	Triggers Implementados	78
2.8.4	Lista de Stored Procedures	79
2.8.5	Stored Procedures Implementados	80
2.8.6	Lista Eventos	81
2.8.7	Eventos Implementados	82
2.8.8	PHP Implementado	83
2.9	Comparação de Implementações (ficheiro versos PHP)	85
2.9.1	Eficiência de Migração	86
2.9.2	Robustez	87
2.9.3	Flexibilidade / Dependência	88
2.9.4	Segurança	89
2.10	Auditoria de Dados (base de dados origem)	90

## 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.

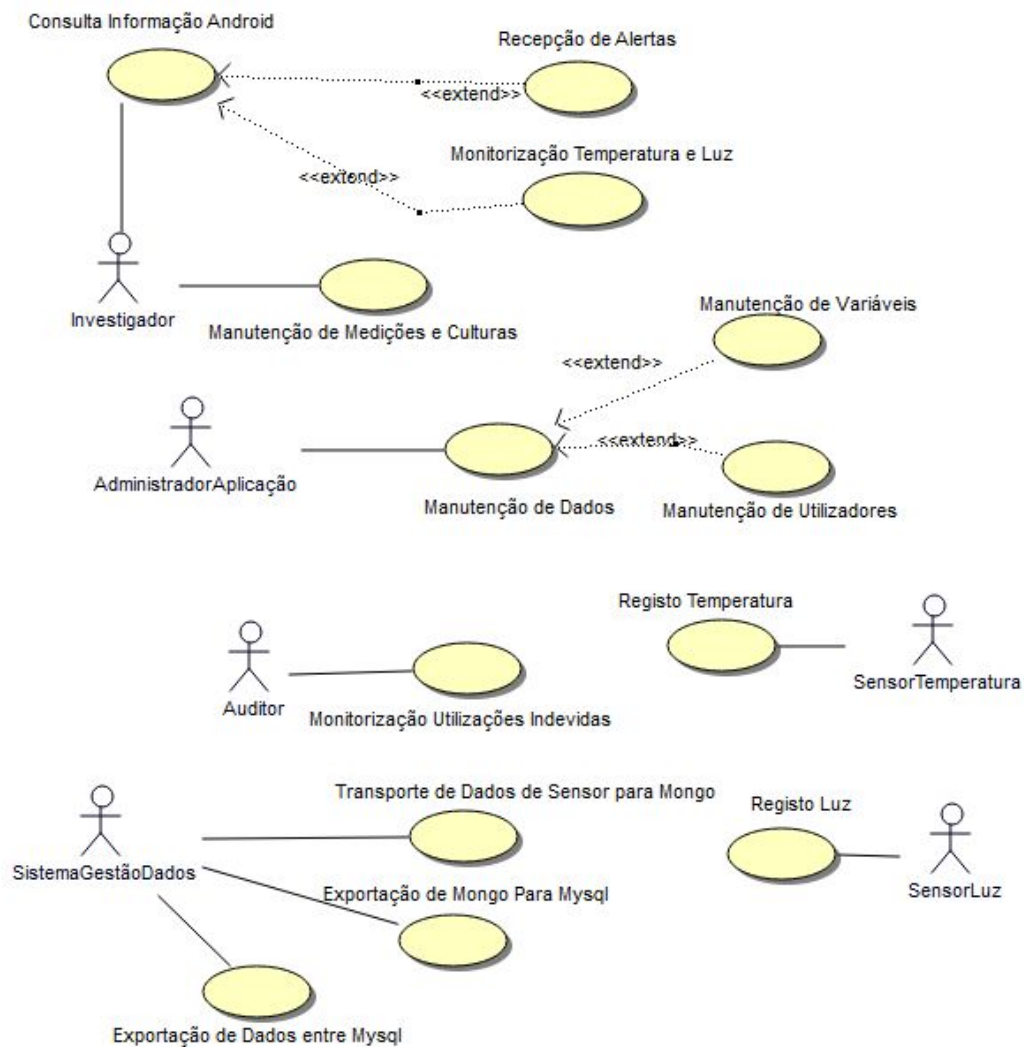
Cada investigador deverá ter a possibilidade de, através de um telemóvel, monitorizar a evolução da temperatura e luminosidade (não apenas a última leitura, mas a evolução na última hora ou horas) e receber os dois tipos de alertas.

### **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.).

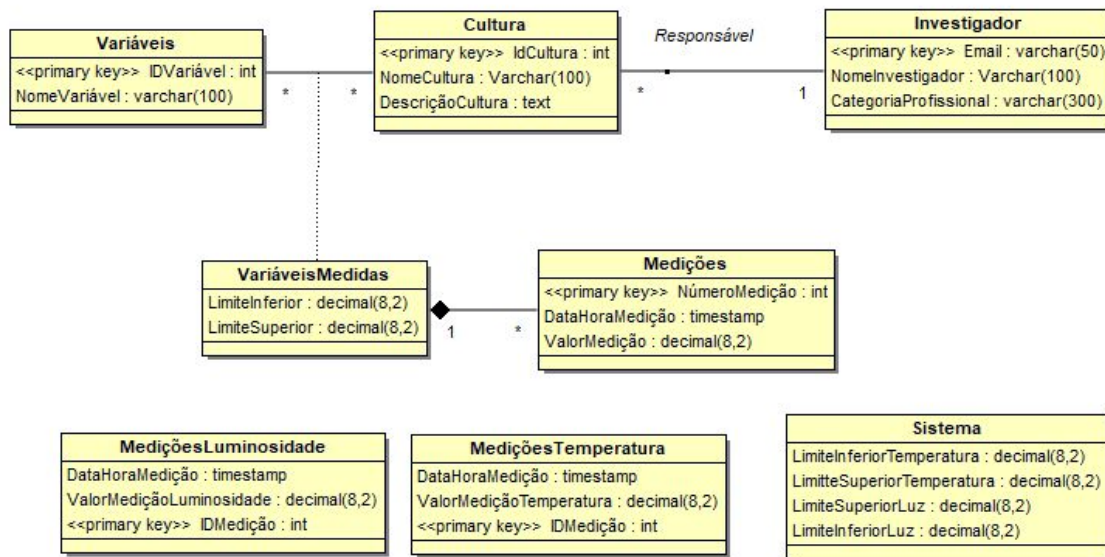
### **Diagrama de Use Case Global**



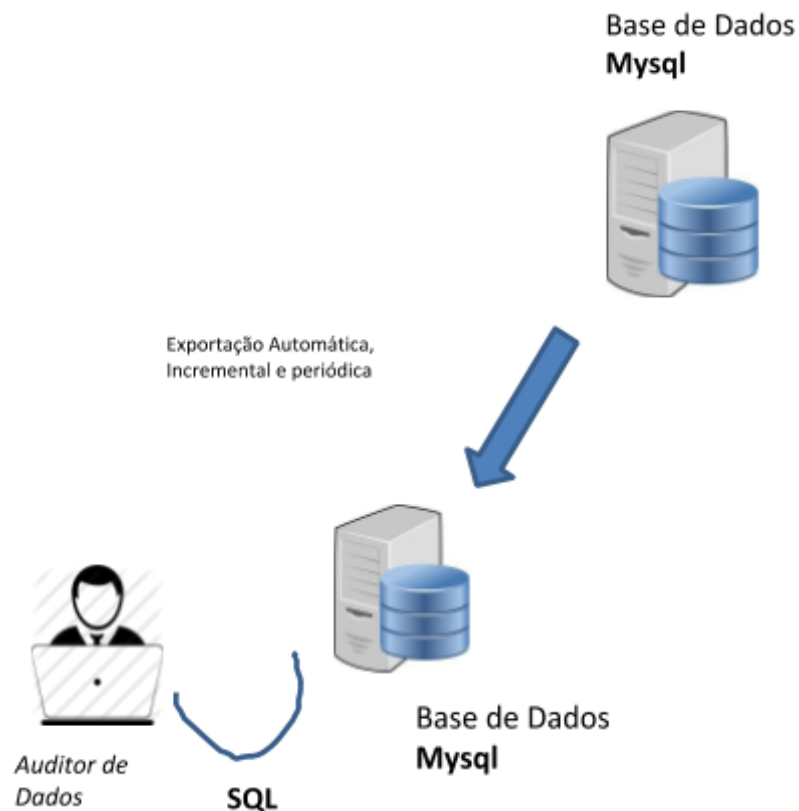


No presente relatório apenas são contemplados os use case "Exportação Dados entre Mysql", "Monitorização de Utilizações Indevidas" e "Manutenção de Utilizadores" (apenas a componente Mysql/Privilégios/SP/Triggers)). A componente Java (manutenção de culturas, medições, variáveis e utilizadores) não é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

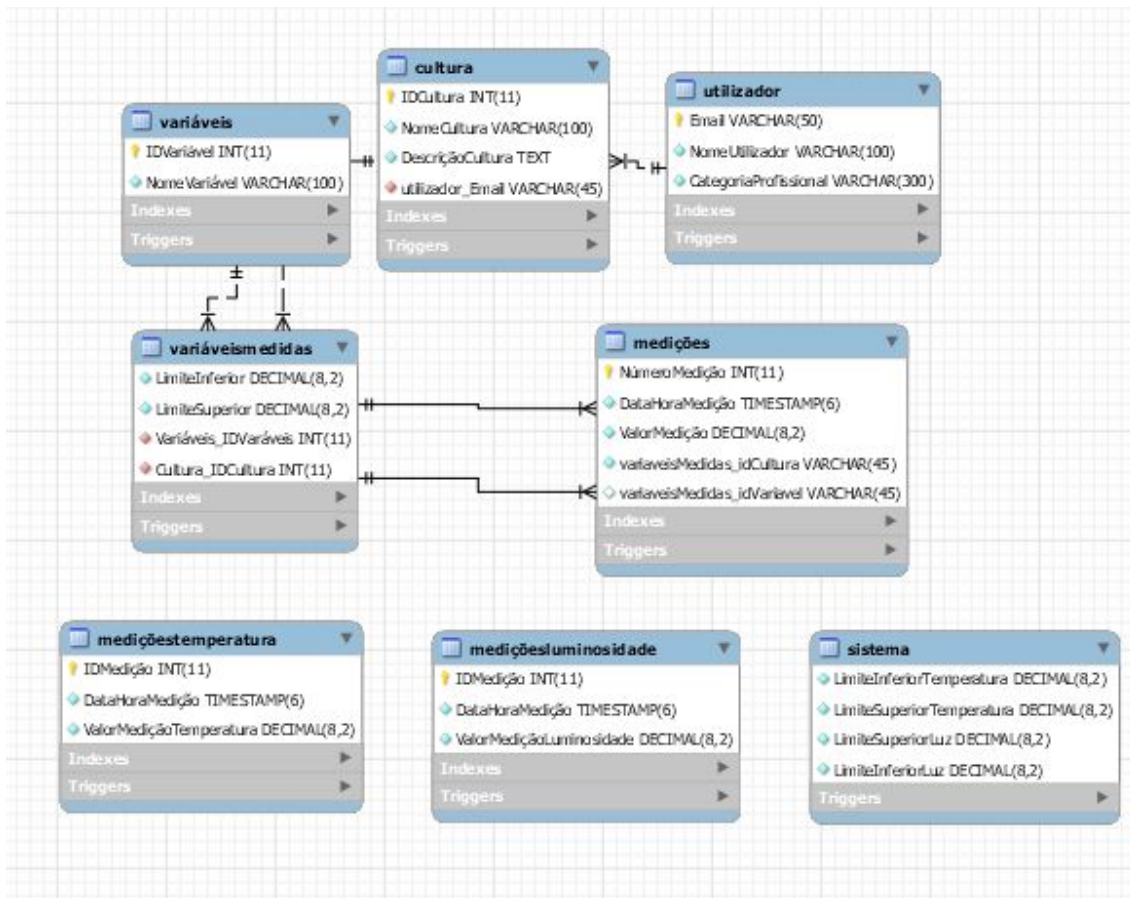


## Esquema de Migração



# 1 Etapa A e B

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



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, VariaveisMedidas 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	Restriçt
Variável	VariávelMedida	ID_Variável	Cascade	Restriçt

Cultura	VariávelMedida	ID_Cultura	Cascade	Cascade
VariávelMedida	Medição	ID_Variável	Cascade	Cascade
VariávelMedida	Medição	ID_Cultura	Cascade	Cascade

Tabela	Campo	Tipo	PK
variáveis_log	IDVariável	INT(11)	Sim
	NomeVariável	VARCHAR(100)	Não
	utilizador	VARCHAR(100)	Não
	data_operacao	TIMESTAMP(6)	Não
	operacao	VARCHAR(1)	Não
	id	INT(11)	Não
cultura_log	IDCultura	INT(11)	Sim
	NomeCultura	VARCHAR(100)	Não
	DescriçãoCultura	TEXT	Não
	Utilizador_Email	VARCHAR(50)	Não
	utilizador	VARCHAR(100)	Não
	data_operacao	TIMESTAMP(6)	Não
	operacao	VARCHAR(1)	Não
utilizador_log	Email	VARCHAR(50)	Sim
	NomeUtilizador	VARCHAR(100)	Não
	CategoriaProfissional	VARCHAR(300)	Não
	utilizador	VARCHAR(100)	Não
	data_operacao	TIMESTAMP(6)	Não
	operacao	VARCHAR(1)	Não
variáveismedidas_log	LimiteInferior	DECIMAL(8,2)	Não
	LimiteSuperior	DECIMAL(8,2)	Não
	Varáveis_IDVaráveis	INT(11)	Não
	Cultura_IDCultura	INT(11)	Não
	utilizador	VARCHAR(100)	Não
	data_operacao	TIMESTAMP(6)	Não
	operacao	VARCHAR(1)	Não
	id	INT(11)	Sim
medições_log	NúmeroMedição	INT(11)	Sim
	DataHoraMedição	TIMESTAMP(6)	Não
	ValorMedição	DECIMAL(8,2)	Não
	IDCultura	VARCHAR(45)	Não
	IDVariável	VARCHAR(45)	Não
	utilizador	VARCHAR(100)	Não
	data_operacao	TIMESTAMP(6)	Não
	operacao	VARCHAR(1)	Não
sistema_log	LimiteInferiorTemperatura	DECIMAL(8,2)	Não
	LimiteSuperiorTemperatura	DECIMAL(8,2)	Não
	LimiteSuperiorLuz	DECIMAL(8,2)	Não
	LimiteInferiorLuz	DECIMAL(8,2)	Não
	utilizador	VARCHAR(100)	Não
	data_operacao	TIMESTAMP(6)	Não
sistema_log	operacao	VARCHAR(1)	Não
	id	INT(11)	Sim

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".

### 1.1.1 Apreciação Crítica e esquema relacional implementado

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

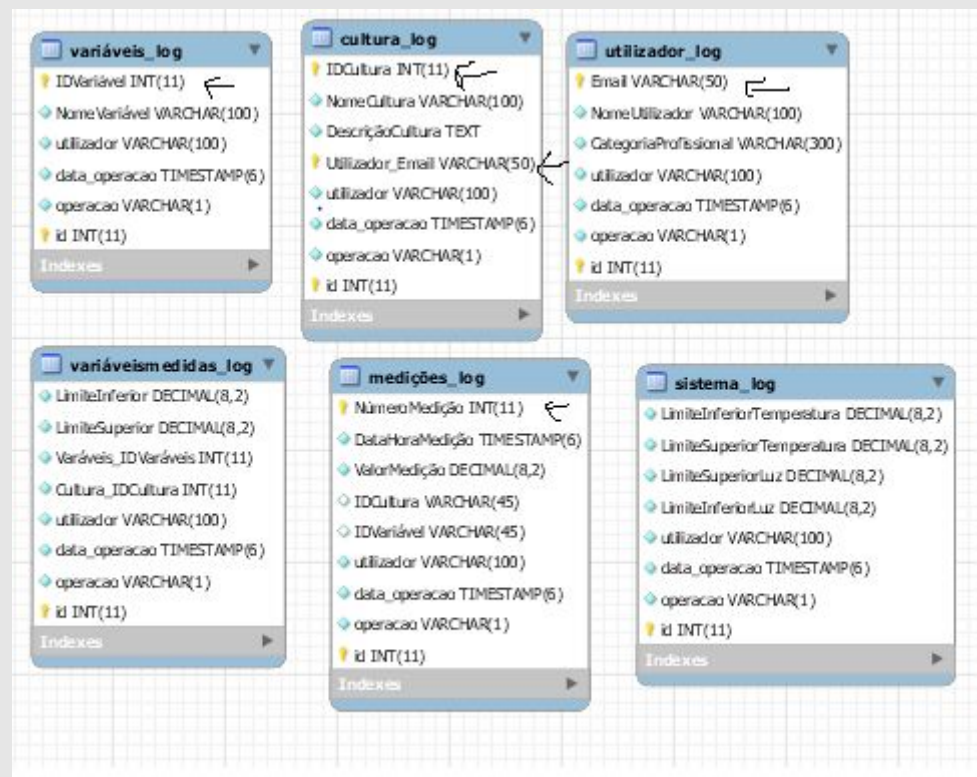
Breve Justificação: A especificação é boa para o propósito pois que todas as necessidades sejam cumpridas sendo possível, a única falha foi a existência de múltiplas chaves primarias em algumas das tabelas de logs que são desnecessária tendo em conta que todas elas se encontram identificadas por um id.

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

As colunas indicadas deixaram de ser chaves primarias pois não tinham impacto na estrutura da base de dados uma vez que o id será sempre único.

As colunas id de todas as tabelas de logs foram alteradas de forma a utilizarem o sistema de Auto Increment facilitando a gestão da base de dados.

Foi também criada uma tabela extra com o nome estado\_migracao para possibilitar um sistema de exportação sem duplicados.





## 1.2 Utilizadores Base de Dados de Origem

<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
<b>select_all_medições</b>	-	X
<b>consulta_por_dia</b>	X	X
<b>consulta_Email_por_CategoriaProfissional</b>	X	X
<b>consulta_utilizador_email_por_NomeCultura</b>	X	X
<b>cria_utilizador</b>	X	-
<b>apaga_utilizador</b>	X	-
<b>insere_medição</b>	-	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.

A maioria dos Stored Procedures definidos têm função de leitura. São cedidas permissões de leitura a todos os utilizadores para todas as tabelas exceto para a tabela Cultura e Medições, pois apenas investigadores têm acesso a estas.

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.

### 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):**

A especificação é bastante clara em relação ao modo como devem ser implementados os utilizadores, no entanto não resolve alguns problemas relativamente à restrição de acesso dos investigadores a culturas e medições alheias, sendo permitido a qualquer investigador ler e escrever todas as culturas e medições. Considerando que esta especificação foi escrita com base na ideia de que o investigador apenas tem acesso as stored procedures criadas para este pensamos que estas apresentam capacidades só permitindo queries muito específicas. É uma boa decisão não haver permissões sobre as tabelas de logs deste modo estas não estão sujeitas a adulteração.

##### **Solução Implementada:**

Tabela	Administrador	Investigador
Cultura	-	-
Cultura_Log	-	-
Medições	-	-
Medições_Log	-	-
MediçõesLuminosidade	L	-
MediçõesTemperatura	L	-
Sistema	L, E	-
Sistema_Log	L	-

Utilizador	L,E	-
Utilizador_Log	L	-
Variáveis	L,E	L
Variáveis_Log	L	-
VariáveisMedidas	L,E	-
VariáveisMedidas_Log	L	-

Stored Procedures	Administrador	Investigador
select_all_medições	-	X
consulta_por_dia	X	X
consulta_Email_por_CategoriaProfissional	X	X
consulta_utilizador_email_por_NomeCultura	X	X
cria_utilizador	X	-
apaga_utilizador	X	-
insere_medição	-	X



## 1.3 Gestão de Logs

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

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  
medições\_luminosidade\_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

Variaveis_AFTER_INSERT	Variáveis Medidas	I	A
Variaveis_AFTER_DELETE	Variáveis Medidas	D	A
Variaveis_AFTER_UPDATE	Variáveis Medidas	U	A

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

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

Breve Justificação: A especificação é boa o suficiente para cobrir todas as necessidades de logs, no entanto não com o código indicado pois este não funciona de acordo com o pretendido e falha se a tabela de logs se encontra vazia.

**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)
Cultura_AFT ER_INSERT		x		
Cultura_AFT ER_DELETE		x		

Cultura_AFT ER_UPDATE		x		
medições_AF TER_INSERT		x		
medições_AF TER_DELETE		x		
medições_AF TER_UPDATE		x		
Sistema_AFT ER_INSERT		X		
Sistema_AFT ER_DELETE		X		
Sistema_AFT ER_UPDATE		X		
Utilizador_ AFTER_INSERT T		X		
Utilizador_ AFTER_DELETE E		X		
Utilizador_ AFTER_UPDATE E		X		
Variaveis_A FTER_INSERT		X		

Variaveis_A FTER_DELETE		X		
Variaveis_A FTER_UPDATE		X		
Variaveis_A FTER_INSERT		X		
Variaveis_A FTER_DELETE		X		
Variaveis_A FTER_UPDATE		x		

### *Triggers Implementados para gestão de logs*

Serão apenas descritos 3 triggers pois o código replica-se para todas as tabelas que tem uma tabela de logs associada.

1. Nome Trigger: **cultura\_AFTER\_DELETE**

Cria um log após ser apagado um registo.

Código:

```
BEGIN
insert into cultura_log (IDCultura, NomeCultura,
DescricaoCultura, Utilizador_Email, utilizador,
data_operacao, operacao, id)
values
(old.IDCultura,old.NomeCultura,old.DescricaoCultura,old.utilizador_Email, current_user(),now(),"D",null);
END
```

2. Nome Trigger: **cultura\_AFTER\_INSERT**

Cria um log após ser inserido um registo.

Código:

```
BEGIN
insert into cultura_log (IDCultura, NomeCultura,
DescricaoCultura, Utilizador_Email, utilizador,
data_operacao, operacao, id)
values
(new.IDCultura,new.NomeCultura,new.DescricaoCultura,new.utilizador_Email, current_user(),now(),"I",null);
END
```

3. Nome Trigger: **cultura\_AFTER\_UPDATE**

Cria um log após ser alterado um registo.

Código:

```
BEGIN
insert into cultura_log (IDCultura, NomeCultura,
DescricaoCultura, Utilizador_Email, utilizador,
data_operacao, operacao, id)
values
(new.IDCultura,new.NomeCultura,new.DescricaoCultura,new.utilizador_Email, current_user(),now(),"U",null);
```

*END*

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

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 seleccionar melhor a informação que deseja ver.

Exemplo de SP(select\_Email\_by\_CategoriaProfissional):

```
CREATE PROCEDURE `select_Email_by_CategoriaProfissional`(in  
categp varchar(50), out mail varchar(100))
```

```
BEGIN
```

```
    INSERT INTO utilizador_log (Email, NomeUtilizador,  
CategoriaProfissional, utilizador, data_operacao, operacao)
```

```
VALUES
```

```
('','','', CURRENT_USER, now() , 'S');
```

```
    Select Email into mail
```

```
    from utilizador
```

```
    where categp = CategoriaProfissional;
```

```
END
```

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	Breve descrição
select_all_medições	-	-	Seleciona todas as colunas da tabela medições
consulta_por_dia	DataHoraMedição	-	Seleciona todas as colunas da tabela medições que tenham como data de medição o valor dado
consulta_Email_por_CategoriaProfissional	CategoriaProfissional	Email	Seleciona coluna Email da tabela utilizador devolvendo as linhas que tenham como CategoriaProfissional o valor dado
consulta_utilizador_email_por_NomeCultura	NomeCultura	Investigador_Email	Seleciona coluna Utilizador_Email da tabela cultura devolvendo as linhas que



			tenham com NomeCultura valor dado
cria_utilizador	Email, NomeUtilizador, CategoriaProfissio nal, TipoUtilizador		Recebe com argumento dados, inseridos manualmente, d um utilizador insere-os n tabela Utilizador
apaga_utilizador	Email		Recebe com argumento Email de u utilizador apaga os dado do próprio o tabela Utilizador
insere_medição	ValorMedição		Recebe com argumento valor de um medição inserida manualmente insere-a n tabela Medição



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

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

Breve Justificação: A especificação foi bem indicada no entanto não a procedure cria\_utilizador e insere\_medicao não têm os parâmetros necessários para realizar a sua função especificada.

#### 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)
select_all_ medições	x			
consulta_po r_dia	x			
consulta_Em ail_por_Cat egoriaProfi ssional	x			
consulta_ut ilizador_em ail_por_Nom eCultura	x			
cria_utiliz ador		x		

apaga_utili zador	x			
insere_medi ção		x		

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

```
1. Nome SP: apaga_utilizador
CREATE DEFINER=`root`@`localhost` PROCEDURE
`apaga_utilizador` (`E` VARCHAR(50)) BEGIN
    select substring_index(E,'@',1) as username;
    drop user username@'%';
    delete from utilizador
    where E=utilizador.Email;

2. Nome SP: consulta_Email_por_CategoriaProfissional
CREATE DEFINER=`root`@`localhost` PROCEDURE
`consulta_Email_por_CategoriaProfissional` (`CP`
    VARCHAR(300)) BEGIN
    select email from utilizador
    where utilizador.CategoriaProfissional=CP;
    insert into utilizador_log
values(null,null,CP,current_user(),now(),'S',null);

3. Nome SP: consulta_por_dia
CREATE DEFINER=`root`@`localhost` PROCEDURE
`consulta_por_dia` (IN `datahora` TIMESTAMP(6)) BEGIN
    select * from medicao
    where cast(datahora as date) =
    cast(medicao.datahoramedicao as date) ;
    insert into medicao_log
values(null,DataHoraMedicao,null,null,null,current_user(),n
        ow(),"S",null);

4. Nome SP: consulta_utilizador_email_por_NomeCultura
CREATE DEFINER=`root`@`localhost` PROCEDURE
`consulta_utilizador_email_por_NomeCultura` (`NC`
    VARCHAR(100)) BEGIN
    select utilizador_email from cultura
    where cultura.NomeCultura=NC;
    insert into cultura_log
values(null,NC,null,null,current_user(),now(),'S',null);

5. Nome SP: cria_utilizador
CREATE DEFINER=`root`@`localhost` PROCEDURE
`cria_utilizador` (IN `Email` VARCHAR(50), IN `Nome`
```

```

VARCHAR(100), IN `CProf` VARCHAR(300), IN `p_Name`
  VARCHAR(100), IN `p_Passw` VARCHAR(100)) BEGIN
DECLARE `_HOST` CHAR(14) DEFAULT '@'localhost'';

SET `p_Name` := CONCAT(''', REPLACE(TRIM(`p_Name`),
  CHAR(39), CONCAT(CHAR(92), CHAR(39))), '''),
`p_Passw` := CONCAT(''', REPLACE(`p_Passw`, CHAR(39),
  CONCAT(CHAR(92), CHAR(39))), '');
SET @`sql` := CONCAT('CREATE USER ', `p_Name`, `_HOST`,
  ' IDENTIFIED BY ', `p_Passw`);
  PREPARE `stmt` FROM @`sql`;
  EXECUTE `stmt`;
SET @`sql` := CONCAT('GRANT ALL PRIVILEGES ON *.* TO ',
  `p_Name`, `_HOST`);
  PREPARE `stmt` FROM @`sql`;
  EXECUTE `stmt`;
  DEALLOCATE PREPARE `stmt`;
  FLUSH PRIVILEGES;
  insert into utilizador
    values (Email, Nome, CProf);

```

```

6. Nome SP: exporta_cultura_log
CREATE DEFINER=`root`@`localhost` PROCEDURE
  `exporta_cultura_log` () NO SQL
  BEGIN
    set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='cultura_log');
    if (@lastmigrated is null) THEN
      INSERT into estado_migracao VALUES
        ('cultura_log', 0, null);
    set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='cultura_log');
    end if;

    if exists (SELECT * from cultura_log) then
      set @max = (select max(id) from cultura_log);
    ELSE
      set @max=0;
    end if;
    set @filename= '_CulturaLog.csv';
    set @comando1 = 'select * from `cultura_log` where
      id>';
    set @comando2 = CONCAT('FIELDS TERMINATED BY
      ', CHAR(39), char(44), char(39), 'ENCLOSED BY
      ', char(39), CHAR(92), char(39), CHAR(39), 'LINES TERMINATED BY
      ', char(39), char(92), char(110), char(39));
    SET @stmt = CONCAT(@comando1, @lastmigrated, ' into
      outfile ', char(39), @max, @filename, char(39), @comando2);

```

```

        select @stmt;
        PREPARE s1 FROM @stmt;
        EXECUTE s1;
        DROP PREPARE s1;
UPDATE estado_migracao set ultima_migracao=@max where
        tabela='cultura_log';

```

```

7.Nome SP: exporta_medicao_log
CREATE DEFINER=`root`@`localhost` PROCEDURE
`exporta_medicao_log` () NO SQL
        BEGIN
        set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='medicao_log');
        if (@lastmigrated is null) THEN
        INSERT into estado_migracao VALUES
        ('medicao_log',0,null);
        set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='medicao_log');
        end if;

        if exists (SELECT * from medicao_log) then
        set @max = (select max(id) from medicao_log);
        ELSE
        set @max=0;
        end if;

        set @filename= '_medicaoLog.csv';
set @comando1 = 'select * from `medicao_log` where
        id>';
        set @comando2 = CONCAT('FIELDS TERMINATED BY
        ',CHAR(39),char(44),char(39),'ENCLOSED BY
        ',char(39),CHAR(92),char(39),CHAR(39),'LINES TERMINATED BY
        ',char(39),char(92),char(110),char(39));
        SET @stmt = CONCAT(@comando1,@lastmigrated,' into
        outfile ',char(39),@max,@filename,char(39),@comando2);
        select @stmt;
        PREPARE s1 FROM @stmt;
        EXECUTE s1;
        DROP PREPARE s1;
UPDATE estado_migracao set ultima_migracao=@max where
        tabela='medicao_log';

```

```

8. Nome SP: exporta_sistema_log
CREATE DEFINER=`root`@`localhost` PROCEDURE
`exporta_sistema_log` () NO SQL
        BEGIN
        set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='sistema_log');

```

```

        if (@lastmigrated is null) THEN
            INSERT into estado_migracao VALUES
                ('sistema_log',0,null);
            set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='sistema_log');
            end if;

            if exists (SELECT * from sistema_log) then
                set @max = (select max(id) from sistema_log);
            ELSE
                set @max=0;
            end if;

            set @filename= '_sistemaLog.csv';
            set @comando1 = 'select * from `sistema_log` where
                id>';
            set @comando2 = CONCAT('FIELDS TERMINATED BY
                ',CHAR(39),char(44),char(39),'ENCLOSED BY
                ',char(39),CHAR(92),char(39),CHAR(39),'LINES TERMINATED BY
                ',char(39),char(92),char(110),char(39));
            SET @stmt = CONCAT(@comando1,@lastmigrated,' into
outfile ',char(39),@max,@filename,char(39),@comando2);
            select @stmt;
            PREPARE s1 FROM @stmt;
            EXECUTE s1;
            DROP PREPARE s1;
            UPDATE estado_migracao set ultima_migracao=@max where
                tabela='sistema_log';

9. Nome SP: exporta_utilizador_log
CREATE DEFINER=`root`@`localhost` PROCEDURE
    `exporta_utilizador_log` () BEGIN
    set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='utilizador_log');
        if (@lastmigrated is null) THEN
            INSERT into estado_migracao VALUES
                ('utilizador_log',0,null);
            set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='utilizador_log');
            end if;

            if exists (SELECT * from utilizador_log) then
                set @max = (select max(id) from utilizador_log);
            ELSE
                set @max=0;
            end if;

            set @filename= '_utilizadorLog.csv';
            set @comando1 = 'select * from `utilizador_log` where
                id>';

```



```

        set @comando2 = CONCAT('FIELDS TERMINATED BY
        ',CHAR(39),char(44),char(39),'ENCLOSED BY
        ',char(39),CHAR(92),char(39),CHAR(39),'LINES TERMINATED BY
        ',char(39),char(92),char(110),char(39));
        SET @stmt = CONCAT(@comando1,@lastmigrated,' into
        outfile ',char(39),@max,@filename,char(39),@comando2);
        select @stmt;
        PREPARE s1 FROM @stmt;
        EXECUTE s1;
        DROP PREPARE s1;
        UPDATE estado_migracao set ultima_migracao=@max where
        tabela='utilizador_log';

```

```

10. Nome SP: exporta_variaveis_medidas_log
CREATE DEFINER=`root`@`localhost` PROCEDURE
`exporta_variaveis_medidas_log` () NO SQL
BEGIN
    set @lastmigrated = (select ultima_migracao from
    estado_migracao where tabela='variaveis_medidas_log');
    if (@lastmigrated is null) THEN
        INSERT into estado_migracao VALUES
        ('variaveis_medidas_log',0,null);
    set @lastmigrated = (select ultima_migracao from
    estado_migracao where tabela='variaveis_medidas_log');
    end if;

    if exists (SELECT * from variaveis_medidas_log) then
        set @max = (select max(id) from
        variaveis_medidas_log);
        ELSE
            set @max=0;
        end if;

        set @filename= '_variaveismedidasLog.csv';
        set @comando1 = 'select * from `variaveis_medidas_log`
        where id>';
        set @comando2 = CONCAT('FIELDS TERMINATED BY
        ',CHAR(39),char(44),char(39),'ENCLOSED BY
        ',char(39),CHAR(92),char(39),CHAR(39),'LINES TERMINATED BY
        ',char(39),char(92),char(110),char(39));
        SET @stmt = CONCAT(@comando1,@lastmigrated,' into
        outfile ',char(39),@max,@filename,char(39),@comando2);
        select @stmt;
        PREPARE s1 FROM @stmt;
        EXECUTE s1;
        DROP PREPARE s1;
        UPDATE estado_migracao set ultima_migracao=@max where
        tabela='variaveis_medidas_log';

```

```

11. Nome SP: exporta_variavel_log
CREATE DEFINER=`root`@`localhost` PROCEDURE
`exporta_variavel_log` () NO SQL
BEGIN
set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='variavel_log');
if (@lastmigrated is null) THEN
INSERT into estado_migracao VALUES
('variavel_log',0,null);
set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='variavel_log');
end if;

if exists (SELECT * from variavel_log) then
set @max = (select max(id) from variavel_log);
ELSE
set @max=0;
end if;

set @filename= '_variavelLog.csv';
set @comando1 = 'select * from `variavel_log` where
id>';
set @comando2 = CONCAT('FIELDS TERMINATED BY
',CHAR(39),char(44),char(39),'ENCLOSED BY
',char(39),CHAR(92),char(39),CHAR(39),'LINES TERMINATED BY
',char(39),char(92),char(110),char(39));
SET @stmt = CONCAT(@comando1,@lastmigrated,' into
outfile ',char(39),@max,@filename,char(39),@comando2);
select @stmt;
PREPARE s1 FROM @stmt;
EXECUTE s1;
DROP PREPARE s1;
UPDATE estado_migracao set ultima_migracao=@max where
tabela='variavel_log';

12. Nome SP: insere_medicao
CREATE DEFINER=`root`@`localhost` PROCEDURE
`insere_medicao` (`valor` DECIMAL(8,2), `idCultura` INT,
`idVariavel` INT) BEGIN

insert into medicao
values(null,now(),valor,idCultura,idVariavel);

12. Nome SP: select_all_medicoes
CREATE DEFINER=`root`@`localhost` PROCEDURE
`select_all_medicoes` () BEGIN
select * from medicao;
insert into medicao_log

```

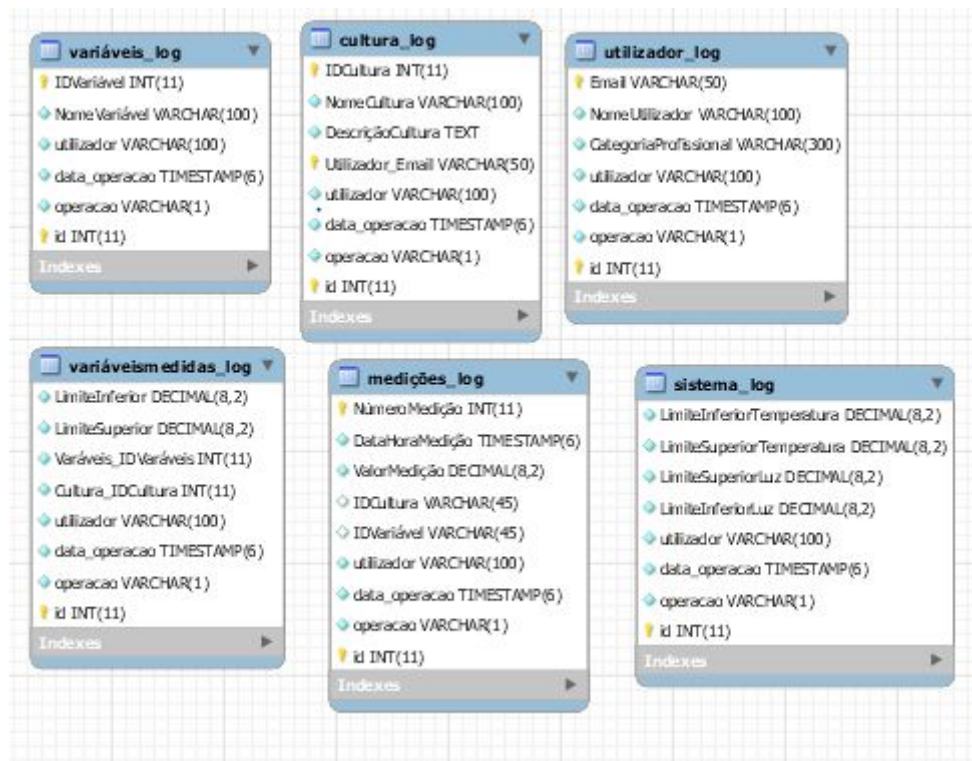
```
values(null,null,null,null,null,current_user(),now(),"S",null);
```

```
13. Nome SP: select_all_medicoes  
CREATE DEFINER=`root`@`localhost` PROCEDURE  
`select_all_medicoes`() BEGIN  
    select * from medicao;  
    insert into medicao_log
```

```
values(null,null,null,null,null,current_user(),now(),"S",null);
```

## 1.4 Migração entre Bases de Dados

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



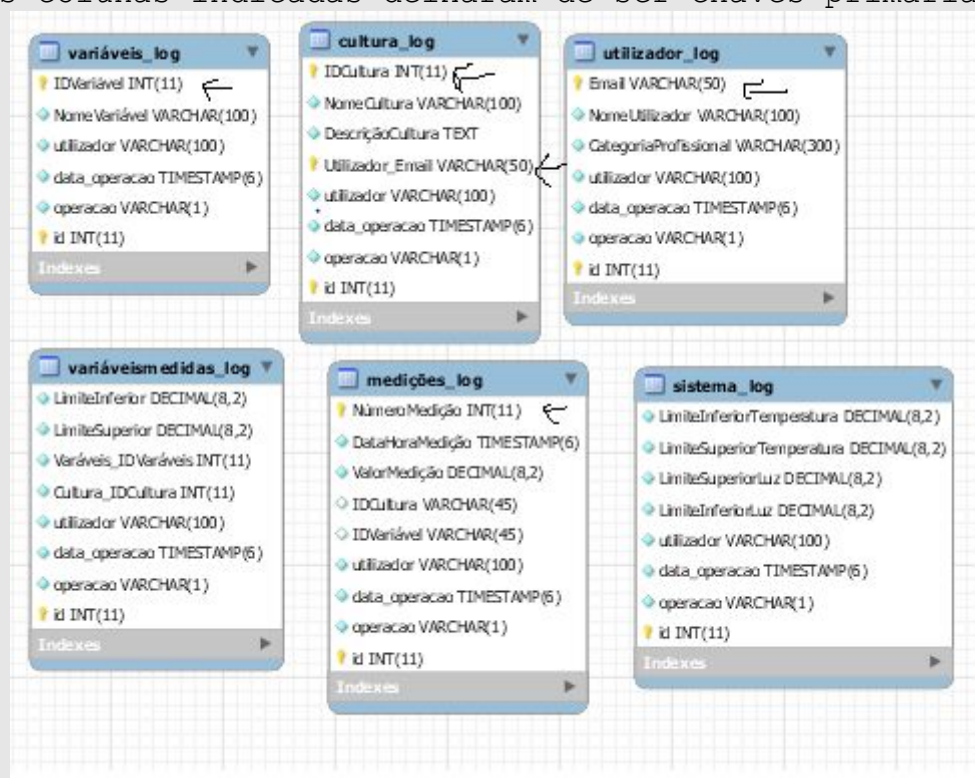
## Apreciação Crítica e esquema relacional implementado

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

Breve Justificação: A especificação é boa para o seu propósito mas mais uma vez falha por apresentar chaves primarias desnecessárias em algumas das tabelas que não têm qualquer impacto na funcionalidade da base de dados.

Foram feitas alterações? (Sim/Não): Sim

As colunas indicadas deixaram de ser chaves primarias.





### 1.4.2 Forma de Migração

#### 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 (seguindo 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.

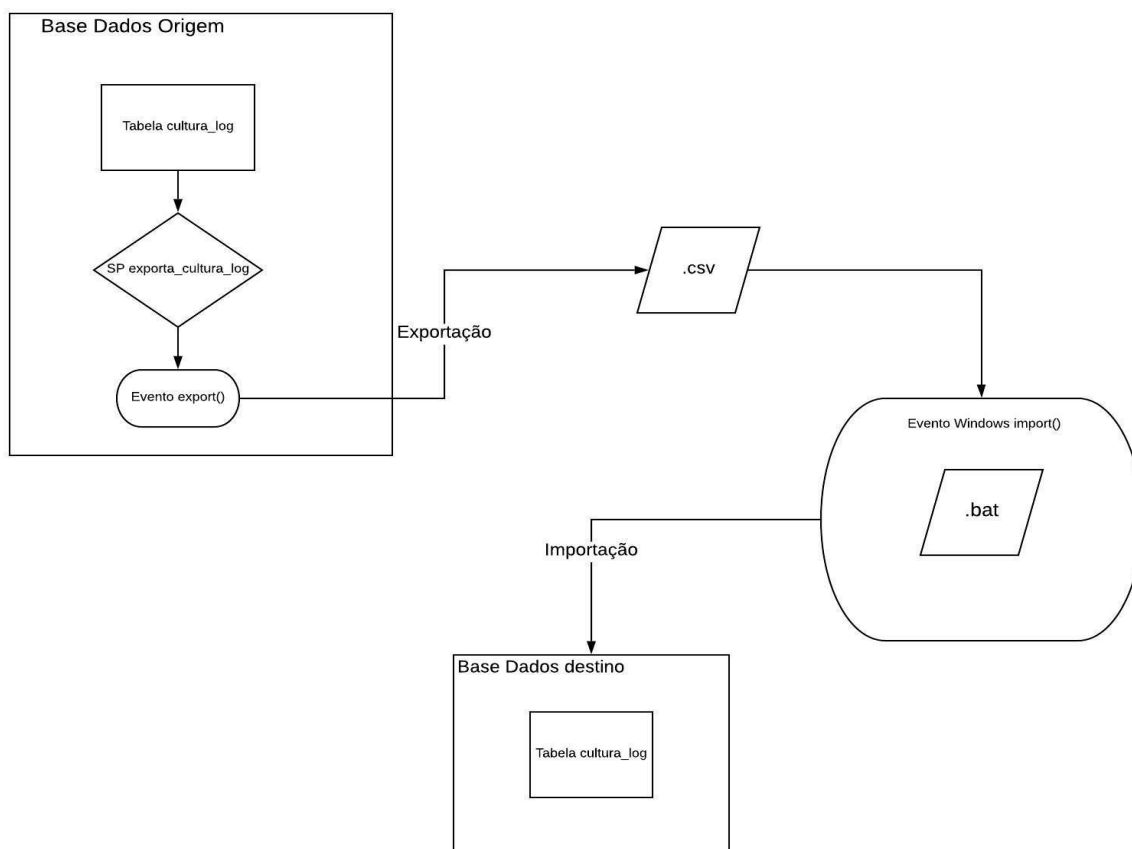
Isto também providencia uma maior segurança de dados, visto que seria perigoso existirem ficheiros com estes dados que poderiam ser acedidos sem qualquer tipo de restrições. Por último, isto também liberta espaço em disco, sendo que seria inútil manter os dados em ficheiro, sabendo que estes mesmos dados 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 (seguindo o exemplo da tabela "cultura\_log"):



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

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

A especificação estava muito bem detalhada e explicava todos os passos para realizar uma exportação resistente a imprevistos no entanto contem código e interações que não são possíveis, uma vez que é guardado o ultimo id migrado numa variável em sql, estas não são permanentes e são apagadas no final da execução da stored procedure. Para solucionar este problema foi criada uma nova tabela na base de dados principal para manter a informação do ultimo id exportado. Outro problema nesta especificação e que ao executar a exportação através do event do mysql e a importação através do scheduler do Windows não é possível garantir que o ficheiro esteja pronto para ser importado quando ocorre a execução do ficheiro .bat assim seria preferível que ambos fossem executados pelo scheduler do Windows onde poderiam ser configurados para ocorrer em sucessão garantindo o seu funcionamento.

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

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.



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

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

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

A especificação esta boa e simples, apenas é necessário um utilizador visto que a base de dados de auditoria tem apenas o propósito de leitura.

**Solução Implementada:**

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

#### 1.4.4 Triggers de suporte à migração de dados (se relevante)

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

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

Qualidade (Frac, 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 especificad o)				
Nome Trigger (tal como especificad o)				
Nome Trigger (tal como especificad o)				

### *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	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
---------------------	--	--	--------	---

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

**Qualidade (Fracá, Razoável, Boa ou Muito Boa): Muito Boa**

Breve Justificação: Bem especificadas e eficazes, só são necessárias estas 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_medições_log	X			
exporta_sistema_log	x			

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

Todas as stored procedures apresentam estruturas semelhantes.

1. Nome SP: `exporta_cultura_log`

Exporta para um ficheiro .csv a informação na tabela `cultura_log`.

Código:

```
BEGIN

set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='cultura_log');
if (@lastmigrated is null) THEN
INSERT into estado_migracao VALUES
('cultura_log',0,null);
set @lastmigrated = (select ultima_migracao from
estado_migracao where tabela='cultura_log');
end if;

if exists (SELECT * from cultura_log) then
set @max = (select max(id) from cultura_log);
ELSE
set @max=0;
end if;

set @filename= '_CulturaLog.csv';
set @comando1 = 'select * from `cultura_log` where
id>';
set @comando2 = CONCAT('FIELDS TERMINATED BY
',CHAR(39),char(44),char(39),'ENCLOSED BY
',char(39),CHAR(92),char(39),CHAR(39),'LINES TERMINATED BY
',char(39),char(92),char(110),char(39));
SET @stmt = CONCAT(@comando1,@lastmigrated,' into
outfile ',char(39),@max,@filename,char(39),@comando2);
select @stmt;
PREPARE s1 FROM @stmt;
EXECUTE s1;
DROP PREPARE s1;
UPDATE estado_migracao set ultima_migracao=@max where
tabela='cultura_log';

END
```



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

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

### *Apreciação Crítica de Eventos*

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

Breve Justificação: Estes eventos estão bem definidos no entanto dão aso a um problema anteriormente referido que é a sincronização da exportação e a importação.

**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)
exporta_ger al		x		
importa_ger al	x			

## Eventos Implementados

1. Nome Evento: exporta\_geral  
Exporta todos as tabelas para ficheiros.

```
@echo off
C:\xampp\mysql\bin\mysql.exe -u root -h localhost
filemaindb -e "call exporta_cultura_log();"
C:\xampp\mysql\bin\mysql.exe -u root -h localhost
filemaindb -e "call exporta_medicao_log();"
C:\xampp\mysql\bin\mysql.exe -u root -h localhost
filemaindb -e "call exporta_sistema_log();"
C:\xampp\mysql\bin\mysql.exe -u root -h localhost
filemaindb -e "call exporta_utilizador_log();"
C:\xampp\mysql\bin\mysql.exe -u root -h localhost
filemaindb -e "call exporta_variaveis_medidas_log();"
C:\xampp\mysql\bin\mysql.exe -u root -h localhost
filemaindb -e "call exporta_variavel_log();"
```

2. Nome Evento: importa\_geral  
Importa o ficheiro replica-se para todas as tabelas.

```
@ECHO off
for /r "C:\xampp\mysql\data\filemaindb\" %%A in (*) do (
    for /f "tokens=1,2 delims=_ " %%i in ("%%~nxA") do (
        if "%%j"=="UtilizadorLog.csv" (
            type %%A >>
            C:\xampp\mysql\data\filemaindb\agregatorUtilizador.csv
        )
    )
    if exist
"C:\xampp\mysql\data\filemaindb\agregatorUtilizador.csv" (
        C:\xampp\mysql\bin\mysql.exe -u root -h localhost
        fileremotedb -e "load data infile
'C:/xampp/mysql/data/filemaindb/agregatorUtilizador.csv'
into table utilizador_log fields terminated by ',' enclosed
by '\"' lines terminated by '\n'"
        del
        C:\xampp\mysql\data\filemaindb\agregatorUtilizador.csv
        for /r "C:\xampp\mysql\data\filemaindb\" %%A in (*) do
        (
            for /f "tokens=1,2 delims=_ " %%i in ("%%~nxA") do
            (
```

```
if "%%j"=="UtilizadorLog.csv" (  
    del %%A  
)  
)  
)
```

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

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

*Apreciação Crítica ao PHP especificado*

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

Breve Justificação:

## *PHP Implementado*

*Código*

### *1.5 Avaliação Global de especificações da Etapa A*

A especificação esta completa e bem detalhada, no entanto falha por vezes na coerência como na escolha do tipo de ficheiro é referenciado que o csv e fácil de editar manualmente como uma das suas vantagens e logo a seguir nos motivos pelos quais se poderia considerar o xml esta também é uma das vantagens. Existem várias ocasiões principalmente na especificação relativa a migração em que o código e a ideia apresentadas nas são possíveis, como já foi detalhado que não é possível manter variáveis de entre execuções de uma stored procedure sendo impossível utilizar a especificação como esta se encontra. Outra das alterações necessárias foi a utilização de id com auto increment nas tabelas de logs uma vez que a abordagem de todas as inserções fazerem um select do valor máximo da coluna id da tabela e adicionar mais um falha quando a tabela esta vazia, complica o código e reduz a eficiência. Os utilizadores da base de dados principal estavam defenidos incorretamente pois tinham premissões que não deveriam ter sendo possível a um utilizador comum realizar selects de todas as tabelas da base de dados.

Ultrapassando estas dificuldades não existe muito mais a apontar negativamente, o restante esta bem especificado e permite uma fácil implementação para além de servir todos os propósitos pedidos no enunciado.



### **Avaliação Global da Qualidade das Especificações recebidas**

**Avaliação (A,B,C,D,E) : C**

**Utilize a seguinte escala:**

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

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

O sistema de controlo da migração não era funcional e os eventos não apresentavam uma especificação muito detalhada.

Algumas stored procedures não estavam especificadas de forma adequada.

Os utilizadores da base de dados de origem tinham alguns problemas em relação as suas permissões.

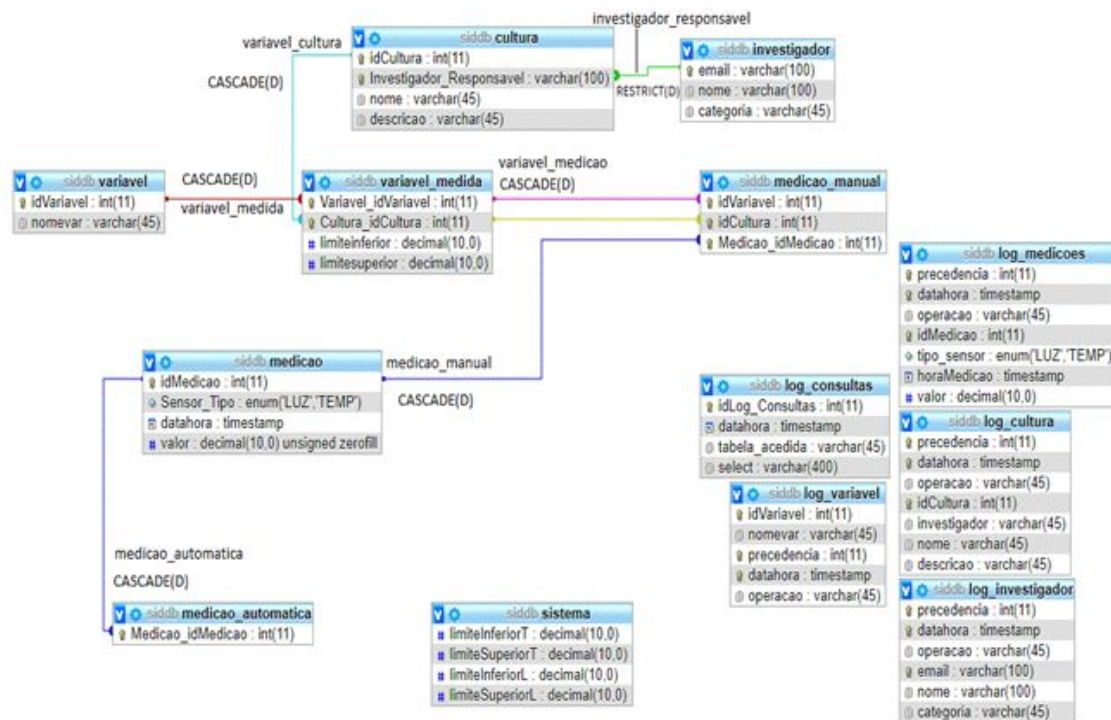
### **Resumo de Avaliações de Qualidade Anteriores (para cada linha assinalar com x em célula correspondente)**

	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



## 2.2 Especificação de Utilizadores

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_veriavel	-	X	-
Cria_medicao	-	X	-

## 2.3 Especificação de Gestão de Logs

### 2.3.1 Triggers de suporte à gestão de logs

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	-

cao_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	-

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

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	Muito breve descrição
Cria_cultura	Varchar(45)-nome, varchar(45)-descricao	Boolean-sucesso	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-sucesso	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-s uccess	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-s uccess	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-s uccess	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-s uccess	Apaga a variavel especificada, mas apenas se o investigador que chama a procedure for o responsável da cultura associada.
Cria_medicao	Int-id_variavel , ENUM('LUZ', 'TEMP')-tipo,de cimal-valor	Boolean-s uccess	Cria uma medição manual associada a variável referida falha caso o investigador não seja responsável pela cultura associada a variável.
Select_auditor	varchar(1000)-s elect	-	Realiza o select indicado e regista um log relativo a esta acção na tabela log_consultas .

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

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

Justificação:

Os triggers e as stored procedures indicadas seriam suficiente para catalogar todos os logs necessários no entanto a especificação estava muito incompleta em termos de explicações e não seguia corretamente as necessidades do enunciado, assim como a estrutura da base de dados e os utilizadores criados não faziam sentido em relação ao enunciado.

## 2.5 Implementação Gestão de Logs

### 2.5.1 Utilizadores implementados

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

Stored Procedures	Administrador	Investigador
select_medicoes	-	X

<b>cria_utilizador</b>	X	-
<b>Apaga_utilizador</b>	X	-
<b>insere_medicao</b>	-	X
<b>apaga_medicao</b>	-	X
<b>cria_cultura</b>	-	X
<b>apaga_cultura</b>	-	X
<b>Altera_cultura</b>	-	X
<b>Select_culturas</b>	-	X

Existem apenas dois tipos de utilizadores para a base de dados principal:

-Administrador que tem privilégios de leitura e escrita nas tabelas todas exceto nas tabelas de logs que são apenas para leitura para que não possa haver nenhuma adulteração e nas tabelas cultura e medições cuja manutenção e cargo dos investigadores, as capacidades de escrita atribuídas ao administrador devem se ao facto de assumir que este percebe o suficiente para utilizar a base de dados corretamente no entanto para seu auxilio foi criada uma stored procedure para ajudar na manutenção de users.

-Investigador só tem permissões de leitura em tabelas cujo o seu acesso é irrelevante em termos de impactos negativos de privacidade no entanto contém informações que podem ser uteis ao investigador, o investigador deve manter as suas culturas e as respetivas medições associadas as mesmas de forma a restringir o acesso do investigador para que este não possa interferir com as culturas de outros investigadores, o investigador apenas pode fazer esta manutenção através das stored procedures definidas uma vez que nas stored procedures podem ser executadas ações pelo investigador mesmo que este não tenha permissões de escrita ou leitura uma vez que estas são executadas com as permissões do definer da SP que neste caso será o utilizador root.



## 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)
Log_investigador_I_A		X		
Log_investigador_D_B		X		
Log_investigador_U_A		X		
Log_investigador_D_B			x	
Log_cultura_I_A		X		
Log_cultura_U_B			x	
Log_cultura_U_A		X		
Log_cultura_D_B		X		
Log_medicao_I_A		X		
Log_medicao_U_B			x	
Log_medicao_U_A		X		
Log_medicao_D_B		X		

Log_variave l_I_A		X		
Log_variave l_U_B			x	
Log_variave l_U_A		X		
Log_variave l_D_B		X		
sistema_AFT ER_INSERT				X
sistema_AFT ER_UPDATE				X
sistema_AFT ER_DELETE				X
variaveis_m edidas_AFTE R_INSERT				X
variaveis_m edidas_AFTE R_UPDATE				X
variaveis_m edidas_AFTE R_DELETE				X



### 2.5.3 Triggers Implementados

```
1. Nome Trigger: cultura_AFTER_INSERT
create TRIGGER `cultura_AFTER_INSERT` AFTER INSERT ON
`cultura` FOR EACH ROW BEGIN
    insert into cultura_log
        values
(null,1,now(),'I',current_user(),new.nomecultura,new.descri
caocultura,new.utilizador_email);
END
```

```
2. Nome Trigger: cultura_AFTER_UPDATE
create TRIGGER `cultura_AFTER_Update` AFTER update ON
`cultura` FOR EACH ROW BEGIN
    insert into cultura_log
        values
(null,1,now(),'U',current_user(),old.nomecultura,old.descri
caocultura,old.utilizador_email);
    insert into cultura_log
        values
(null,2,now(),'U',current_user(),new.nomecultura,new.descri
caocultura,new.utilizador_email);
END
```

```
3. Nome Trigger: cultura_AFTER_DELETE
create TRIGGER `cultura_AFTER_DELETE` AFTER DELETE ON
`cultura` FOR EACH ROW BEGIN
    insert into cultura_log
        values
(null,1,now(),'D',current_user(),old.nomecultura,old.descri
caocultura,old.utilizador_email);
END
```

4. Nome Trigger: variaveis\_medidas\_AFTER\_INSERT  
Semelhante ao cultura\_AFTER\_INSERT
5. Nome Trigger: variaveis\_medidas\_AFTER\_UPDATE  
Semelhante ao cultura\_AFTER\_UPDATE
6. Nome Trigger: variaveis\_medidas\_AFTER\_DELETE  
Semelhante ao cultura\_AFTER\_DELETE
7. Nome Trigger: sistema\_AFTER\_INSERT  
Semelhante ao cultura\_AFTER\_INSERT
8. Nome Trigger: sistema\_AFTER\_UPDATE  
Semelhante ao cultura\_AFTER\_UPDATE
9. Nome Trigger: sistema\_AFTER\_DELETE  
Semelhante ao cultura\_AFTER\_DELETE
10. Nome Trigger: medicao\_AFTER\_INSERT  
Semelhante ao cultura\_AFTER\_INSERT
11. Nome Trigger: medicao\_AFTER\_UPDATE  
Semelhante ao cultura\_AFTER\_UPDATE
12. Nome Trigger: medicao\_AFTER\_DELETE  
Semelhante ao cultura\_AFTER\_DELETE
13. Nome Trigger: variavel\_AFTER\_INSERT  
Semelhante ao cultura\_AFTER\_INSERT
14. Nome Trigger: variavel\_AFTER\_UPDATE  
Semelhante ao cultura\_AFTER\_UPDATE
15. Nome Trigger: variavel\_AFTER\_DELETE  
Semelhante ao cultura\_AFTER\_DELETE
16. Nome Trigger: utilizador\_AFTER\_INSERT  
Semelhante ao cultura\_AFTER\_INSERT
17. Nome Trigger: utilizador\_AFTER\_DELETE  
Semelhante ao cultura\_AFTER\_DELETE

## 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_auditor			x	
Cria_cultura	x			
Mostra_culturas		x		
Alterar_cultura			x	
Apaga_cultura	x			
Cria_variavel			x	
Mostra_variaveis			x	
Apaga_variavel		x		

Cria_medica cao		x		
select_medicoes				x
cria_utilizador				x
apaga_utilizador				x
insere_medicao				x
apaga_medicao				x
select_cultura				x

### 2.5.5 Stored Procedures Implementados

#### 1. Nome SP: select\_medicoes

Recebe como argumento o id de uma cultura e devolve as medições associadas a esta caso o utilizador que chama a SP seja dono da mesma.

*Código:*

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

Cria um utilizador e insere na tabela utilizador

*Código:*

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`cria_utilizador` (IN `Email` VARCHAR(50), IN `Nome`
VARCHAR(100), IN `CProf` VARCHAR(300), IN `p_Name`
VARCHAR(100), IN `p_Passw` VARCHAR(100)) BEGIN
DECLARE `_HOST` CHAR(14) DEFAULT '@''localhost'';

SET `p_Name` := CONCAT(''', REPLACE(TRIM(`p_Name`),
CHAR(39), CONCAT(CHAR(92), CHAR(39))), ''),
`p_Passw` := CONCAT(''', REPLACE(`p_Passw`, CHAR(39),
CONCAT(CHAR(92), CHAR(39))), '');
SET @`sql` := CONCAT('CREATE USER ', `p_Name`, `_HOST`,
' IDENTIFIED BY ', `p_Passw`);
PREPARE `stmt` FROM @`sql`;
EXECUTE `stmt`;

SET @`sql` := CONCAT("GRANT 'investigador' TO ",
`p_Name`, `_HOST`);
PREPARE `stmt` FROM @`sql`;
EXECUTE `stmt`;
DEALLOCATE PREPARE `stmt`;
FLUSH PRIVILEGES;
insert into utilizador
values (Email,Nome,CProf,concat (p_Name, `_HOST`));

END
```

### 3. Nome SP: apaga\_utilizador

Apaga um utilizador e remove da tabela utilizador.

*Código:*

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`apaga_utilizador` (`nomeutilizador` VARCHAR(50)) BEGIN
    drop user if exists nomeutilizador;
    delete from utilizador
    where nomeutilizador=utilizador.username;

    END
```

### 4. Nome SP: insere\_medicao

Insere uma medição para uma cultura caso o utilizador seja seu dono.

Insere uma medicao

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`insere_medicao` (`valor` DECIMAL(8,2), `idCultura` INT,
    `idVariavel` INT) BEGIN
    if idcultura in(select idcultura from cultura
    where current_user()=cultura.username) then
        insert into medicao
        values(null,now(),valor,idCultura,idVariavel);
    end if;

    END
```

### 5. Nome SP: apaga\_medicao

Apaga uma medição se esta pertencer ao utilizador.

```
CREATE PROCEDURE `apaga_medicao` (id int(11),idcultura
    int(11))
    BEGIN
    if idcultura in(select idcultura from cultura
    where current_user()=cultura.username) then
        delete from medicao
        where id=medicao.numeromedicao;
    end if;

    END
```

### 6. Nome SP: cria\_cultura

Cria uma cultura associada ao utilizador.

```
CREATE PROCEDURE `cria_cultura` (nome varchar(100),
    descricao text)
    BEGIN
    insert into cultura
    values(null,nome,descricao,currentuser());

    END
```

### 7. Nome SP: apaga\_cultura

```

Apaga a cultura se o utilizador for seu dono.
CREATE PROCEDURE `apaga_cultura` (id int(11))
    BEGIN
        if id in(select idcultura from cultura
        where current_user()=cultura.username) then
            delete from cultura
            where cultura.idcultura=id;
        end if;
    END

```

8.Nome SP: altera\_cultura

Altera a descrição de uma cultura se esta pertencer ao utilizador.

```

CREATE PROCEDURE `altera_cultura` (id int(11),des text)
    BEGIN
        if id in(select idcultura from cultura
        where current_user()=cultura.username) then
            update cultura
            set cultura.descricaoocultura=des
            where cultura.idcultura=id;
        end if;
    END

```

9. Nome SP: select\_culturas

```

CREATE PROCEDURE `select_culturas` ()
    BEGIN
        select * from cultura
        where current_user()=cultura.username;
    END

```

## 2.6 Especificação de Migração entre Bases de Dados

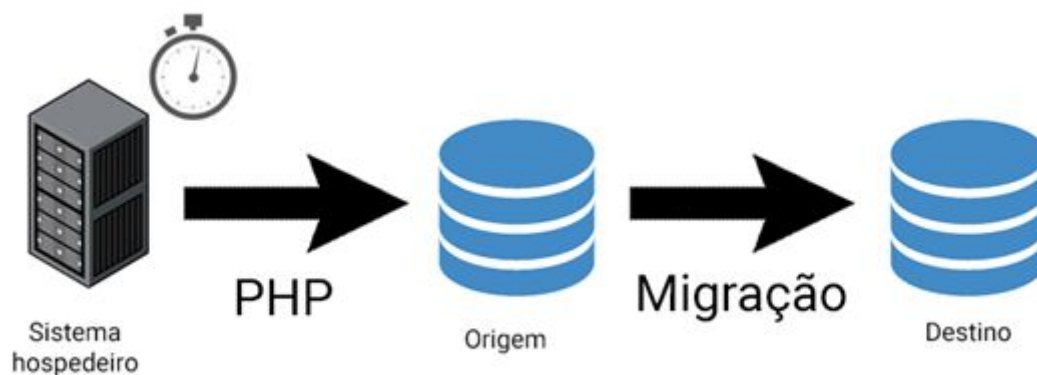
### 2.6.1 Esquema relacional da base de Dados Mysql especificada (destino)

<b>siddb log_consultas</b> <ul style="list-style-type: none"><li>idLog_Consultas : int(11)</li><li>datahora : timestamp</li><li>tabela_acedida : varchar(45)</li><li>select : varchar(400)</li></ul>	<b>siddb log_medicoes</b> <ul style="list-style-type: none"><li>precedencia : int(11)</li><li>datahora : timestamp</li><li>operacao : varchar(45)</li><li>idMedicao : int(11)</li><li>tipo_sensor : enum('LUZ','TEMP')</li><li>horaMedicao : timestamp</li><li>valor : decimal(10,0)</li></ul>
<b>siddb log_variavel</b> <ul style="list-style-type: none"><li>idVariavel : int(11)</li><li>nomevar : varchar(45)</li><li>precedencia : int(11)</li><li>datahora : timestamp</li><li>operacao : varchar(45)</li></ul>	<b>siddb log_cultura</b> <ul style="list-style-type: none"><li>precedencia : int(11)</li><li>datahora : timestamp</li><li>operacao : varchar(45)</li><li>idCultura : int(11)</li><li>investigador : varchar(45)</li><li>nome : varchar(45)</li><li>descricao : varchar(45)</li></ul>
	<b>siddb log_investigador</b> <ul style="list-style-type: none"><li>precedencia : int(11)</li><li>datahora : timestamp</li><li>operacao : varchar(45)</li><li>email : varchar(100)</li><li>nome : varchar(100)</li><li>categoria : varchar(45)</li></ul>



## 2.6.2 Forma de Migração Especificada

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.

### 2.6.3 Utilizadores Especificados

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

#### 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
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(1000)-select	-	D	Faz o select indicado como parâmetro, registando o mesmo na tabela de logs de consulta.
---------------	----------------------	---	---	---

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

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

## 2.6.7 PHP de suporte à migração de dados especificado

```
$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.*

## *2.7 Avaliação das especificações do próprio grupo Migração*

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

Justificação:

O método de migração era funcional no entanto estava muito mal especificado e para além da sua má especificação a forma de migração era inadequada uma vez que migrava baseado em períodos de 24 horas o que não garantia as características de robustez e flexibilidade desejadas. Os utilizadores não tinham sido compreendidos a este ponto uma vez que foram especificados utilizadores e stored procedures desnecessários.



## 2.8 Implementação da Migração de Dados

### 2.8.1 Utilizadores Implementado

Tabela	Auditor
Cultura_Log	L
Medicao_Log	L
Sistema_Log	L
Utilizador_Log	L
Variavel_Log	L
Variaveis_medidas_Log	L

## 2.8.2 Lista Triggers

**NÃO FORAM IMPLEMENTADOS OU ESPECIFICADOS TRIGGERS NESTA PARTE.**

### 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)
Migracao_auditor_local			x	
Migracao_auditor_remot o			x	
select_remo to			x	

### 2.8.5 Stored Procedures Implementados

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

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

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

### 2.8.6 Lista Eventos

<b>Lista de Eventos (para cada evento assinalar com x em célula correspondente)</b>				
	Implementa do de Acordo com Especifica do	Implementa do mas diferente de Especifica do	Não Implementa do	Não Especifica do (criado de novo)
Execução do script PHP	x			

### 2.8.7 Eventos Implementados

1. Nome Evento:Execução do script PHP  
O código está especificado na parte do PHP

## 2.8.8 PHP Implementado

```
<?php

$url = 'localhost';
$username = "root";
$password = "";
$conn = mysqli_connect($url, $username, $password);

if (!$conn) {
    die("ConnectionFailed: " . $conn->connect_error);
}

ini_set('max_execution_time', 300);

$sql1 = "INSERT INTO auditordb.variaveis_medidas_log
        SELECT * FROM siddb.variaveis_medidas_log
        WHERE id >
            (SELECT IF(MAX(id) != '', MAX(id), 0) FROM
auditordb.variaveis_medidas_log);";

$sql2 = "INSERT INTO auditordb.cultura_log
        SELECT * FROM siddb.cultura_log
        WHERE id >
            (SELECT IF(MAX(id) != '', MAX(id), 0) FROM
auditordb.cultura_log);";

$sql3 = "INSERT INTO auditordb.utilizador_log
        SELECT * FROM siddb.utilizador_log
        WHERE id >
            (SELECT IF(MAX(id) != '', MAX(id), 0) FROM
auditordb.utilizador_log);";

$sql4 = "INSERT INTO auditordb.medicao_log
        SELECT * FROM siddb.medicao_log
        WHERE id >
            (SELECT IF(MAX(id) != '', MAX(id), 0) FROM
auditordb.medicao_log);";

$sql5 = "INSERT INTO auditordb.sistema_log
        SELECT * FROM siddb.sistema_log
```



```

        WHERE id >
            (SELECT IF(MAX(id) != '', MAX(id), 0) FROM
auditoradb.sistema_log);";

$sql6 = "INSERT INTO auditoradb.variavel_log
        SELECT * FROM siddb.variavel_log
        WHERE id >
            (SELECT IF(MAX(id) != '', MAX(id), 0) FROM
auditoradb.variavel_log);";

echo "Starting migrations.<br>";

$start_time = strtotime("now");
$result1 = mysqli_query($conn, $sql1);
$result2 = mysqli_query($conn, $sql2);
$result3 = mysqli_query($conn, $sql3);
$result4 = mysqli_query($conn, $sql4);
$result5 = mysqli_query($conn, $sql5);
$result6 = mysqli_query($conn, $sql6);
$end_time = strtotime("now");

if ($result1) {
    echo "SUCCESS - variaveis_medidas_log<br>";
} else {
    echo "FAILED - variaveis_medidas_log<br>";
}

if ($result2) {
    echo "SUCCESS - cultura_log<br>";
} else {
    echo "FAILED - cultura_log<br>";
}

if ($result3) {
    echo "SUCCESS - utilizador_log<br>";
} else {
    echo "FAILED - utilizador_log<br>";
}

if ($result4) {
    echo "SUCCESS - medicao_log<br>";
}

```

```
} else {
    echo "FAILED - medicao_log<br>";
}

if ($result5) {
    echo "SUCCESS - sistema_log<br>";
} else {
    echo "FAILED - sistema_log<br>";
}

if ($result6) {
    echo "SUCCESS - variavel_log<br>";
} else {
    echo "FAILED - variavel_log<br>";
}

echo "Migrations completed.<br>";
echo "Time elapsed: ", $end_time - $start_time, " seconds.<br>";

mysqli_close($conn);

?>
```

## Avaliação Global da Qualidade das Especificações do próprio grupo

Avaliação (A,B,C,D,E) : B

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

A base de dados origem não estava de acordo com o enunciado assim como os seus utilizadores, triggers e stored procedures.

O método de migração não era o melhor não garantindo qualquer robustez ou flexibilidade.

Ao longo de toda a especificação nada está justificado impedindo qualquer tipo de interpretação.

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

	Fraco	Razoável	Bom	Muito Bom
BD Sybase		x		
Triggers Log		x		
SP Log	x			
Utilizadores Log	x			
BD Mysql			x	
Forma Migração		x		

Triggers Migração	-	-	-	-
SP Migração	-	-	-	-
Eventos Migração			x	
Utilizadores Migração	x			
PHP Migração		x		

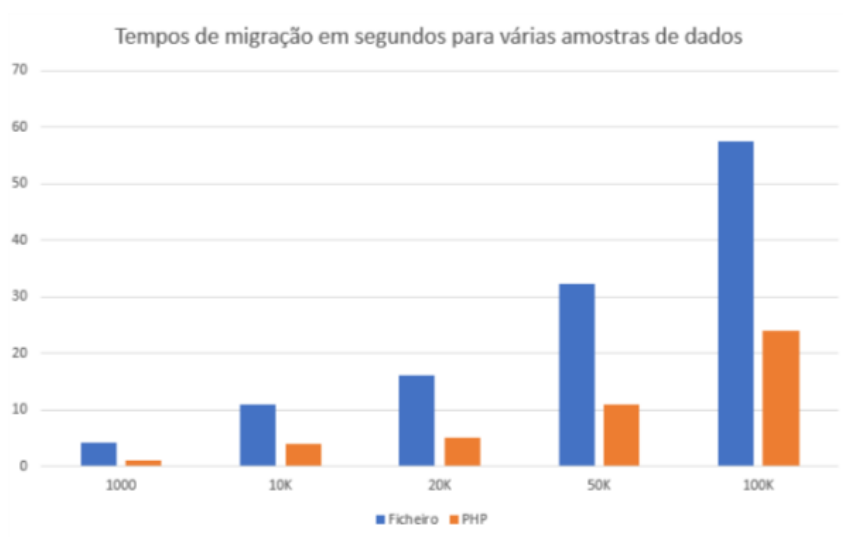
## *2.9 Comparação de Implementações (ficheiro versos PHP)*

Comparando as duas soluções migração por ficheiro e por PHP, reparamos que o ficheiro permite uma maior flexibilidade uma vez que o ficheiro pode ser transportado de qualquer modo, pode ser alterado e pode ainda ser guardado para uma importação posterior eliminando a dependência de ligação direta entre as bases de dados. No entanto esta solução tem a limitação no que toca à segurança e robustez uma vez que o ficheiro encontra-se acessível e pode ser apagado, roubado ou alterado caso seja interceptado entre a exportação e a importação. Neste aspeto o PHP é muito mais seguro uma vez que a comunicação será feita diretamente entre as duas bases de dados, não sendo possível a adulteração dos dados, no entanto mais uma vez isto impõe a limitação de ligação direta que limita a flexibilidade da solução.

### 2.9.1 Eficiência de Migração

Segundo os resultados obtidos experimentalmente, a migração efetuada com PHP foi mais rápida que a migração por ficheiro.

Amostra de dados	Ficheiro (segundos)	PHP (segundos)
1000	4.21	1
10 000	10.9	4
20 000	16.1	5
50 000	32.4	11
100 000	57.5	24



### 2.9.2 Robustez

Ambas as especificações demonstram robustez, pois nos casos de falhas do sistema a migração pode ser retomada sem grandes complicações e garantir que todos os dados sejam migrados com sucesso.

No entanto, PHP é mais robusto pois faz a verificação dos dados em falta imediatamente antes da migração. Enquanto a outra especificação depende da integridade dos ficheiros com dados a migrar, que por sua vez podem ser sujeitos à eliminação por fontes externas.

### 2.9.3 Flexibilidade / Dependência

Ambas as opções têm uma flexibilidade similar pois para alterar a periodicidade de migração basta alterar a frequência com que os eventos no task scheduler do windows são lançados sendo assim isto não é uma limitação. No entanto o PHP apresenta uma limitação neste aspeto uma vez que ambas as bases de dados tem de estar em funcionamento em simultâneo caso contrário é impossível fazer a migração. Por isso no aspecto da dependência podemos dizer que no PHP as bases de dados estão diretamente dependentes uma da outra enquanto que no caso do Ficheiro mesmo que uma esteja a falhar é possível esperar para que a migração seja feita mais tarde.



### 2.9.4 Segurança

Considerando a segurança das duas soluções temos que o php será a opção mais segura uma vez que a transferência é feita diretamente da base de dados origem à base de dados destino, fazendo com que, ao contrário da migração por ficheiro, não deixe informacao para trás em caso de falha, o que é o caso da implementação de migração baseada no ficheiro, uma vez que caso esta falhe fica um ficheiro por apagar para que possa ser inserido na próxima vez que seja executado o processo de migração.

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

auditor.html

```
<html lang="en">

<head>
    <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Auditor</title>
</head>

<body>
    <form action="view.php">
        <input type="submit" value="sistema_log" name="table">
        <input type="submit" value="cultura_log" name="table">
        <input type="submit" value="variaveis_medidas_log"
name="table">
        <input type="submit" value="utilizador_log"
name="table">
        <input type="submit" value="medicao_log" name="table">
        <input type="submit" value="variavel_log" name="table">
    </form>
</body>

</html>
```

view.php

```
<?php

$url = 'localhost';
$username = "root";
$password = "123";
$conn = mysqli_connect($url, $username, $password);

if (!$conn) {
    die("ConnectionFailed: " . $conn->connect_error);
}
```

```

}

$sql = "SELECT * FROM auditordb.".$_GET["table"];
$result = mysqli_query($conn, $sql);
$rows = array();

if ($result) {
    if (mysqli_num_rows($result) > 0) {
        while ($r = mysqli_fetch_assoc($result)) {
            array_push($rows, $r);
        }
    }
}

mysqli_close($conn);

echo "<form      action='auditor.html'><input      type='submit'
value='Voltar'></form>";
echo "<h1>Tabela: ", $_GET["table"], "</h1>";
echo "<table border='1'>";
foreach ($rows as $value) {
    echo "<tr>";
    foreach ($value as $k => $v) {
        echo "<th>$k</th>";
    }
    echo "</tr>";
    break;
}
foreach ($rows as $value) {
    echo "<tr>";
    foreach ($value as $k => $v) {
        echo "<td>$v</td>";
    }
    echo "</tr>";
}
echo "</table>";

?>

```

