Sistemas de Informação Distribuídos

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

Monitorização de Culturas em Laboratório

Mongo DB e Android

Identificação do grupo autor da especificação (Etapa A):

Número	Nome	Foto
62109	André Vieira	
16482	Paulo Vieira	
69565	Rodolfo Arnaldo	
73553	Rui Tomé	
65345	Tiago Rodrigues	

Identificação do grupo autor da implementação (Etapas B):

Número	Nome	Foto

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 falha;
- 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;
- O grupo que primeiro edita o documento (Etapa A) apenas escreve na secção 1.1, e o outro grupo apenas em todas as outras secções.

Índice

Monitorização de Culturas em Laboratório

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

Cada cultura tem um único investigador responsável e apenas ele pode actualizar e consultar os dados de medições das suas culturas. Esta *protecção de dados* é um aspecto importante do sistema.

Sobre cada cultura são regularmente efectuadas (manualmente) medições com base num conjunto de variáveis que variam consoante a cultura. Para cada cultura o sistema conhece o intervalo de valores normal para cada variável, logo, o sistema poderá emitir alertas caso surja um valor anormal.

Por exemplo, para as culturas hidropónicas de pimento e tomate, fazem-se medições do nível de concentração de mercúrio e chumbo. Se, por exemplo, a concentração de chumbo no pimento reduzir significativamente – menos de 25 mg/litro – significa que a planta ajuda a absorver os metais indesejáveis. (*Culturas = pimento e tomate (hidropónico), variáveis = mercúrio, chumbo.)*

Outro exemplo. Numa solução onde convivem bactérias e antibióticos, se o número de bactérias cresce pouco então é porque são sensíveis ao antibiótico (logo, sabemos como as matar se forem prejudiciais). Se o número de colónias de bactérias *Bacillus subtilis,* colocadas junto de antibiótico penicilina, aumentar em mais de 30% em 2 horas é porque o antibiótico não é eficaz. *(Cultura = Bacillus subtilis, variável = penicilina.)*

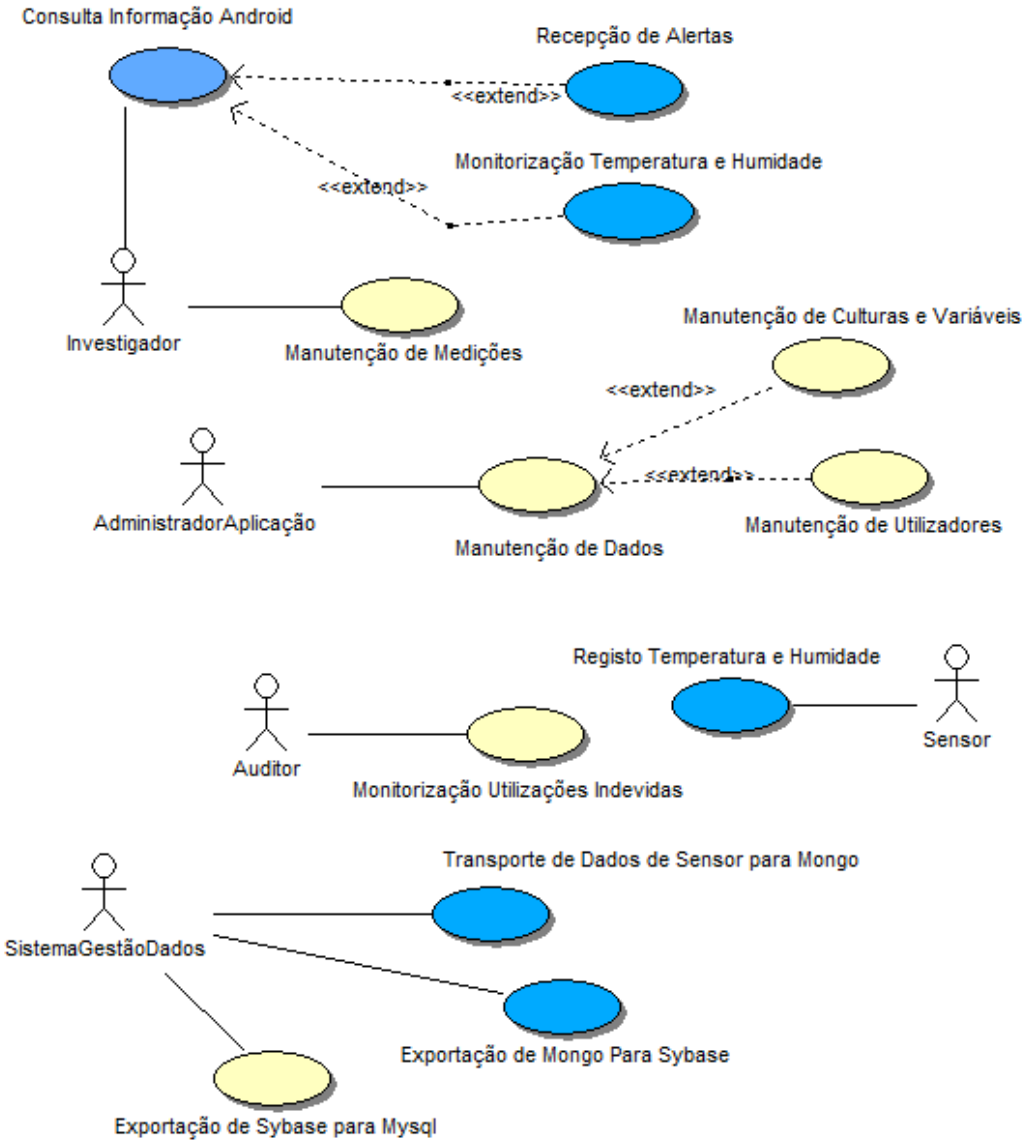
Existe um sensor que periodicamente *lê* a temperatura e humidade no laboratório. Os dados são registados na base de dados (classe HumidadeTemperatura), e pretende-se que sejam utilizados para emitir alertas (o sistema sabe o intervalo de valores de humidade e temperatura ideal para cada cultura) e para tentar *explicar* eventuais valores anómalos de variáveis (por exemplo, “detecta-se que sempre que a temperatura desce bruscamente – mais do que 5 graus em menos de uma hora – a concentração de ferro no pimento apresenta valores anormalmente baixos”).

Cada investigador deverá ter a possibilidade de, através de um telemóvel, monitorizar a evolução da temperatura e humidade (não apenas a última leitura, mas a evolução da última hora ou horas) e receber alertas relativos a variações bruscas nós valores das variáveis das suas culturas.

É necessário guardar no sybase o registo de todas as operações de escrita sobre todas as tabelas (qua dados foram alterados/inseridos/apagados, quando e por quem) e registo de operações de consulta sobre a tabela Medições. Esse registo de alterações (*log) é exportado* incrementalmente (apenas informação nova) e periodicamente para uma base de dados autónoma (mysql). Através dessa base de dados (apenas de consulta) um auditor pode analisar se ocorreram utilizações abusivas dos dados (por exemplo, verificar se um investigador tentou

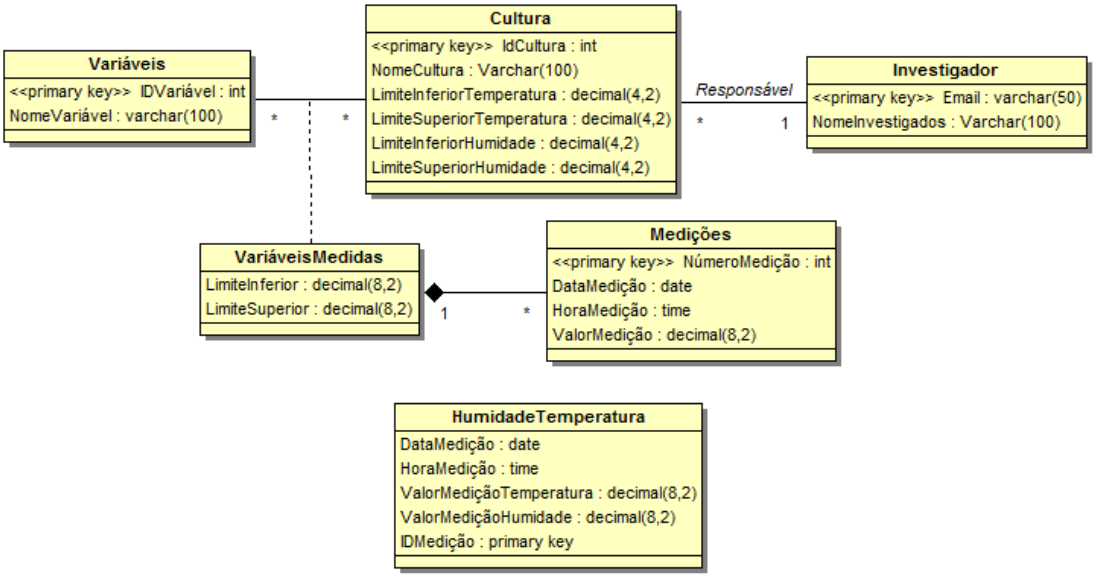
ler medições de culturas que não as suas, quem é que alterou limites de Temperatura de uma cultura, etc.).

Diagrama de Use Case Global



.

Diagrama de Classes de Suporte à Base de Dados Sql Anywhere

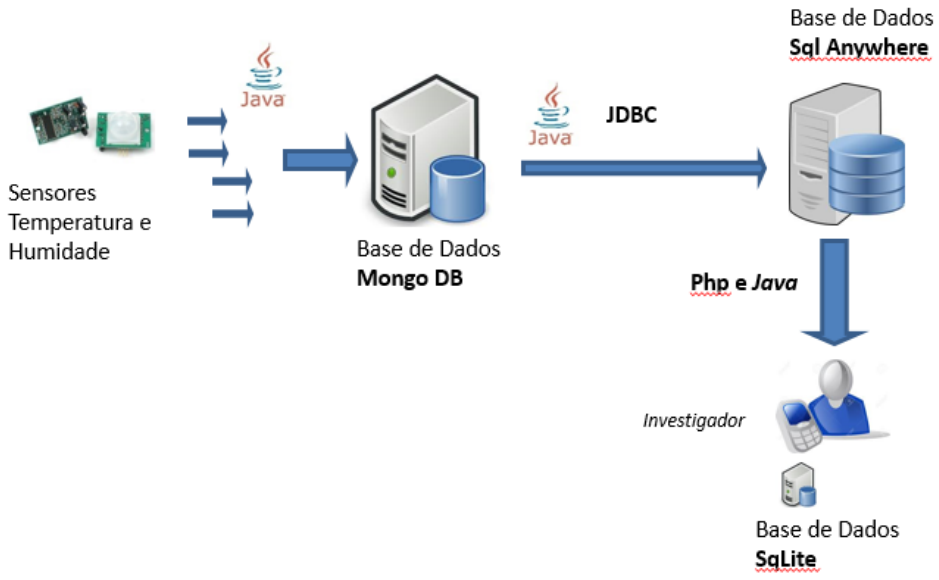


Sensor

Exemplo Mensagens

{"sensor":"1"," datapassagem ":"2016/12/12"," horapassagem ":"18:45:24"}

Esquema de Importação e Migração



1 Mongo DB

1.1 Descrição Geral do Procedimento

1.1.1 Objetivo

A organização do processo de migração será de forma a que dados enviados pelos sensores sejam armazenados na base de dados MongoDB que posteriormente transitam para a base de dados Sybase. Será a partir ao Sybase que os aparelhos móveis dos investigadores solicitarão as medições aí armazenadas e ainda não visualizadas.

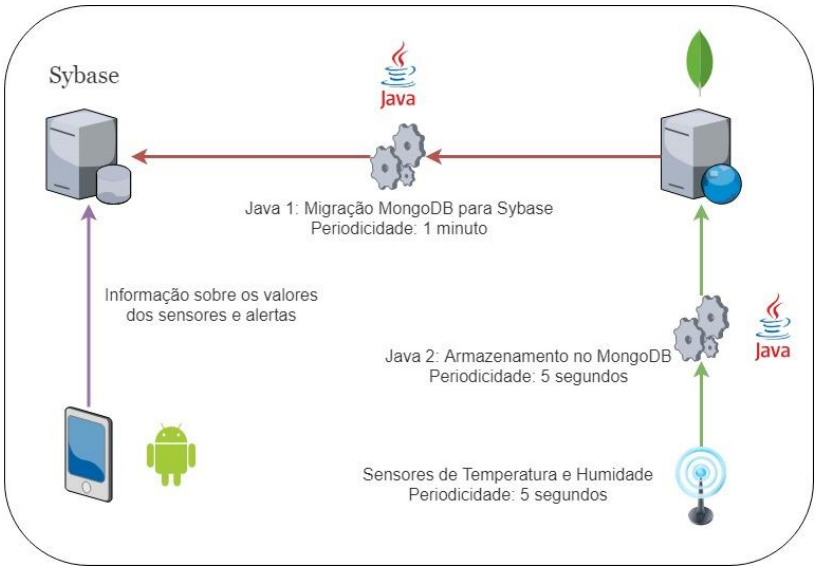


Figura 1 - esquema de processos e agentes desde a sensorização até à leitura

1.1.2 Java

A componente Java da implementação fará a integração entre os diferentes elementos: Sensores, MongoDB e Sybase, funcionando como intermediário.

Esta componente será estruturada de forma a que sejam executados dois processos distintos, que serão lançados através de dois .jar's, um para tratar da comunicação entre o MongoDB e o Sybase (J1) e outra que se encarregará de interligar a informação dos sensores com o Mongo DB (J2).

Os dois processos terão as seguintes funções:

- J1 - Este processo será responsável pela migração dos dados do Mongo DB para o Sybase, com uma periodicidade pré-definida e explicada na secção 1.3. Fará também a "tradução" dos dados provenientes do MongoDB e preparará os comandos de inserção no Sybase.
- J2 - Este processo ficará à escuta de dados por parte do sensor e escreverá o respetivo conteúdo no MongoDB, também respeitando uma periodicidade pré-definida.

1.1.2.1 Configuração dos processos

Para a configuração dos processos (J1 e J2) é necessário que exista um ficheiro em JSON, para cada processo, com as configurações globais, que será carregado sempre que o programa inicia e traduzido para variáveis do próprio processo. Desta forma, sempre que o processo seja corrido, não será necessário editar código, mas sim apenas o ficheiro de configuração. Para efeitos de simplificação, o ficheiro deverá constar na mesma pasta que o processo (ficheiro jar). Quando à nomenclatura, deverá ser usado os nomes j1.conf, e j2.conf para o processo J1 e J2 respetivamente.

A estrutura a seguir para cada ficheiro de configuração é demonstrada de seguida.

Para o processo J1 (que liga o MongoDB ao Sybase) são necessários os seguintes parâmetros:

- IP do servidor Sybase;
- Porto do servidor Sybase;
- Utilizador Sybase;
- Password do utilizador Sybase;
- IP do servidor Mongo;
- Porto do servidor Mongo;
- Utilizador Mongo;
- Password do utilizador Mongo;
- Periodicidade com que o processo retira os dados do mongoDB e envia para o Sybase.

Para o processo J2 (que liga o sensor ao MongoDB) são necessários os seguintes parâmetros:

- IP do servidor Mongo;
- Porto do servidor Mongo;
- Utilizador Mongo;
- Password do utilizador Mongo.

Opcionalmente, poderão ser adicionados também parâmetros de configuração para o sensor, nomeadamente o tópico e o QOS.

Os ficheiros j1.conf e j2.conf, deverão seguir a seguinte estrutura:

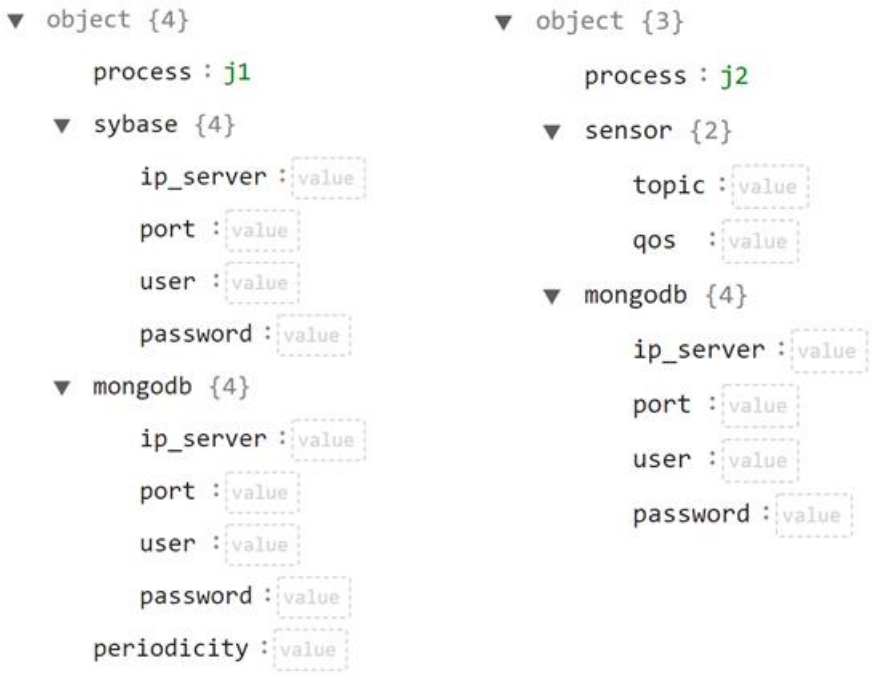


Figura 2 - Estrutura do ficheiro j1.conf (à esquerda) e do ficheiro j2.conf (à direita).

1.1.2.2 Processo J1

O processo J1, como referido anteriormente, será responsável pela migração dos dados no MongoDB para o Sybase, fazendo ainda o registo num ficheiro .log da ocorrência de erros aquando da sua operação e outros registos que se considerem relevantes.

A migração será efetuada respeitando o seguinte procedimento:

1. Selecionar o último membro existente no MongoDB num dado momento;
2. Guardar em memória o identificador(id) correspondente ao *timestamp* desse membro, que será usado como chave de pesquisa;
3. Selecionar todos os registos, presentes no MongoDB, com o id (*timestamp*) igual ou inferior ao do membro selecionado (chave de pesquisa);
4. Preparar uma *string* com os dados para inserção, no formato adequado à execução do comando INSERT para múltiplas linhas – formato:

```
INSERT INTO HumidadeTemperatura
(dataHoraMedicao, valorMedicaoTemperatura, valorMedicaoHumidade)
VALUES      (dataHora#1, temperatura#1, humidade#1),
            (dataHora#2, temperatura#2, humidade#2),
            ...
            (dataHora#n, temperatura#n, humidade#n);
```

5. Executar o comando INSERT;
6. Testar o SQLSTATE[1] à saída da execução do INSERT;
    - Se SQLSTATE <> 0 (falha durante a operação), os dados irão ser completamente descartados no Sybase pois caso o INSERT falhe, este não introduz qualquer dado, visto a operação INSERT ser uma operação atómica (procedimento ROLLBACK). Os dados não enviados irão permanecer no Mongo, sendo enviados no próximo envio em conjunto com os novos dados provenientes dos sensores;
    - Se SQLSTATE = 0 (operação executada com sucesso), eliminar todos os membros na base de dados MongoDB com o id (*timestamp*) igual ou inferior ao do membro selecionado (chave de pesquisa);

NOTA: É assumido que a eliminação no MongoDB será sempre bem-sucedida, não havendo lugar a erros durante esta operação.

1.1.2.3 Processo J2

O processo J2, estará encarregue de popular a base de dados Mongo com a informação que recebe do sensor. Para isso, o seu funcionamento respeitará os seguintes passos:

1. Abrir ligação com sensor e MongoDB;
2. Ficar "à escuta" de dados provenientes do sensor;
3. Receção de dados do sensor e interpretação dos diferentes campos recebidos;
4. Envio dos dados para o MongoDB.

NOTA 1: O processo J2 deverá enviar para o MongoDB os dados, assim que estes são recebidos do sensor, não fazendo qualquer buffer de dados.

NOTA 2: Assume-se que o MongoDB se encontra permanentemente disponível, não sendo possível a ocorrência de falhas pelo processo J2 não se conseguir conectar ao MongoDB.

1.1.2.4 2 Processos Vs 2 Threads

A opção de se criarem dois projetos Java, cada um correspondendo a um processo, ao em vez da utilização de duas threads num único processo, foi ponderada tendo em conta a dicotomia performance/robustez.

Com a utilização de dois processos em separado iremos consumir mais recursos, uma vez que cada um terá as suas próprias variáveis, stack e alocação de memória. No caso da utilização de threads, estas iriam partilhar o mesmo espaço de memória e as mesmas variáveis globais. Mediante a opção tomada, a implementação será mais "pesada" do ponto de vista a utilização de recursos, diminuindo a sua performance.

[1] O SQLSTATE trata-se de uma variável de estado devolvida sempre que um comando é executado.

Todavia, a perda de performance é contrabalançada pelo ganho em robustez e fiabilidade do nosso sistema. Com a implementação especificada, mesmo que um dos processos termine de forma abrupta o outro continuará em execução. Um de três necessários poderá ocorrer:

- Falha no J1 - Se houver uma falha no processo responsável pela migração dos dados do MongoDB para o Sybase, garantimos que os dados do sensor continuam a ser armazenados no MongoDB, visto ser da responsabilidade de outro processo. Esta falha será detectada pela falta de novos registos do lado do Sybase.
- Falha no J2 - Se a falha ocorrer na migração dos dados do sensor para o MongoDB, o J1 pode continuar a migrar dados do MongoDB para o Sybase e detectará a falha quando tentar migrar mais dados e perceber que no MongoDB não se encontra nenhuma informação para migrar, algo que será impossível, uma vez que a periodicidade de migração para o Sybase é inferior à periodicidade de registos de dados no MongoDB. Desta forma, saberíamos que o problema se encontra ou no sensor ou no processo J2.
- Falha em ambos os processos - No caso de falha dos dois processos, nenhuma informação é migrada e os dados enviados pelo sensor serão irremediavelmente perdidos.

1.1.3 MongoDB

O Mongo sendo uma base de dados não relacional, não utiliza tabelas ou relações entre elas, organizando a sua informação em Coleções, cuja constituição se encontra especificada no capítulo 1.2.

Este tipo de base de dados é o ideal para armazenar grandes volumes de dados, garantindo uma escalabilidade e flexibilidade superior às bases de dados relacionais. Desta forma, representa a opção correta para acomodar a informação proveniente do nosso sensor, que num futuro próximo se pode tornar em inúmeros sensores que transmitem dados não necessariamente similares.

A base de dados a utilizar será do tipo “Document Database” em que os documentos são estruturas de dados JSON agrupadas em coleções. Com este tipo de base de dados é possível associar a uma “key” estruturas de dados mais complexas, mantendo a acessibilidade e eficiência nas consultas.

Considera-se que o MongoDB não está sujeito a falhas, encontrando-se sempre disponível para todas as operações sobre esta base de dados. Esta garantia pode ser dada através da implementação de um Cluster de servidores Mongo (funcionalidade que não é de implementação obrigatória).

1.1.4 Sybase

Como se poderá verificar no capítulo 1.4, somente se considera uma alteração relativamente à estrutura da base de dados Sybase anteriormente especificada.

Face à existência do requisito de geração de alertas para o investigador relativamente aos valores de temperatura e humidade da sala, torna-se necessário criar uma tabela para guardar os registos de alertas. Como os alertas são gerados a partir de valores únicos de leitura ou através da comparação com registos anteriores, o seu processamento tem de ser feito ao nível do Sybase, sendo necessário incorporar os triggers e procedimentos necessários para essa função (capítulo 1.6).

O ciclo de verificação e geração de alertas deverá ser executado sempre após receção de novos registos na tabela HumidadeTemperatura.

1.2 Estrutura da Base de Dados Mongo

A base de dados no MongoDB denomina-se "**Leituras**".

Existe apenas uma *collection*, denominada "**leitura_sensor**", cujos membros (*documents*) estão definidos como objetos sujeitos à indexação automática por omissão (com índice único - *unique index*), os quais têm na sua constituição 3 propriedades, todas obrigatórias:

- **dataHora**, com o tipo de dados *String*, capaz de alojar dados no formato correspondente a um *timestamp*;
- **temperatura**, com o tipo de dados *Double*, capaz de acomodar um valor numérico com 2 casas decimais;
- **humidade**, com o tipo de dados *Double*, capaz de acomodar um valor numérico com 2 casas decimais.

Os tipos de dados e os formatos escolhidos para o armazenamento na *collection* têm em vista o seu destino final - a tabela HumidadeTemperatura existente na base de dados do Sybase SQL Anywhere.

Existe ainda um índice único suplementar, sobre a propriedade dataHora, tendo como objetivo suprimir qualquer possibilidade de registo de valores diferentes enquanto resultado de leitura do mesmo sensor num dado instante.

A título de exemplo, os registos/documentos podem ser inseridos com o comando db.collection.insert() do seguinte modo:

> **db.leitura_sensor.insert({ dataHora: "2018-01-01 12:11:30.000", temperatura: 17.00, humidade: 74.00 })**

A figura seguinte apresenta o resultado da execução do comando *db.leitura_sensor.find().pretty()*, após a inserção de alguns registos.

```
> db.leitura_sensor.find().pretty()
{
        "_id" : ObjectId("5af7822421607b794878fd14"),
        "dataHora" : "2018-01-01 12:06:20.000",
        "temperatura" : 16.5,
        "humidade" : 70
}
{
        "_id" : ObjectId("5af7824121607b794878fd15"),
        "dataHora" : "2018-01-01 12:07:30.000",
        "temperatura" : 16.6,
        "humidade" : 71
}
{
        "_id" : ObjectId("5af7825a21607b794878fd16"),
        "dataHora" : "2018-01-01 12:08:30.000",
        "temperatura" : 17,
        "humidade" : 71
}
```

Figura 3 - exemplo da execução do comando find() no mongoDB.

1.3 Periodicidade de Leitura de Sensores e Escrita no Mongo

Como definido no projeto, o processo J2 recebe dados provenientes do sensor, de 5 em 5 segundos. Por motivos de fiabilidade (caso o processo J2 vá a baixo), os dados são enviados para o MongoDB sempre que recebidos, não havendo assim qualquer buffer do lado do processo java. Ou seja, sempre que o java recebe um dado do sensor, insere imediatamente no java, e por isso, a periodicidade acaba automaticamente, por ser de 5 em 5 segundos também.

1.4 Estrutura da Base de Dados Sybase

1.4.1 Estrutura Geral

A estrutura da base de dados Sybase permanece de acordo com o especificado anteriormente, sendo composta pelas tabelas relacionais de dados e as respetivas tabelas de logs, com a função de migração de dados para a base de dados do auditor (MySQL), como se pode ver na Figura 1.

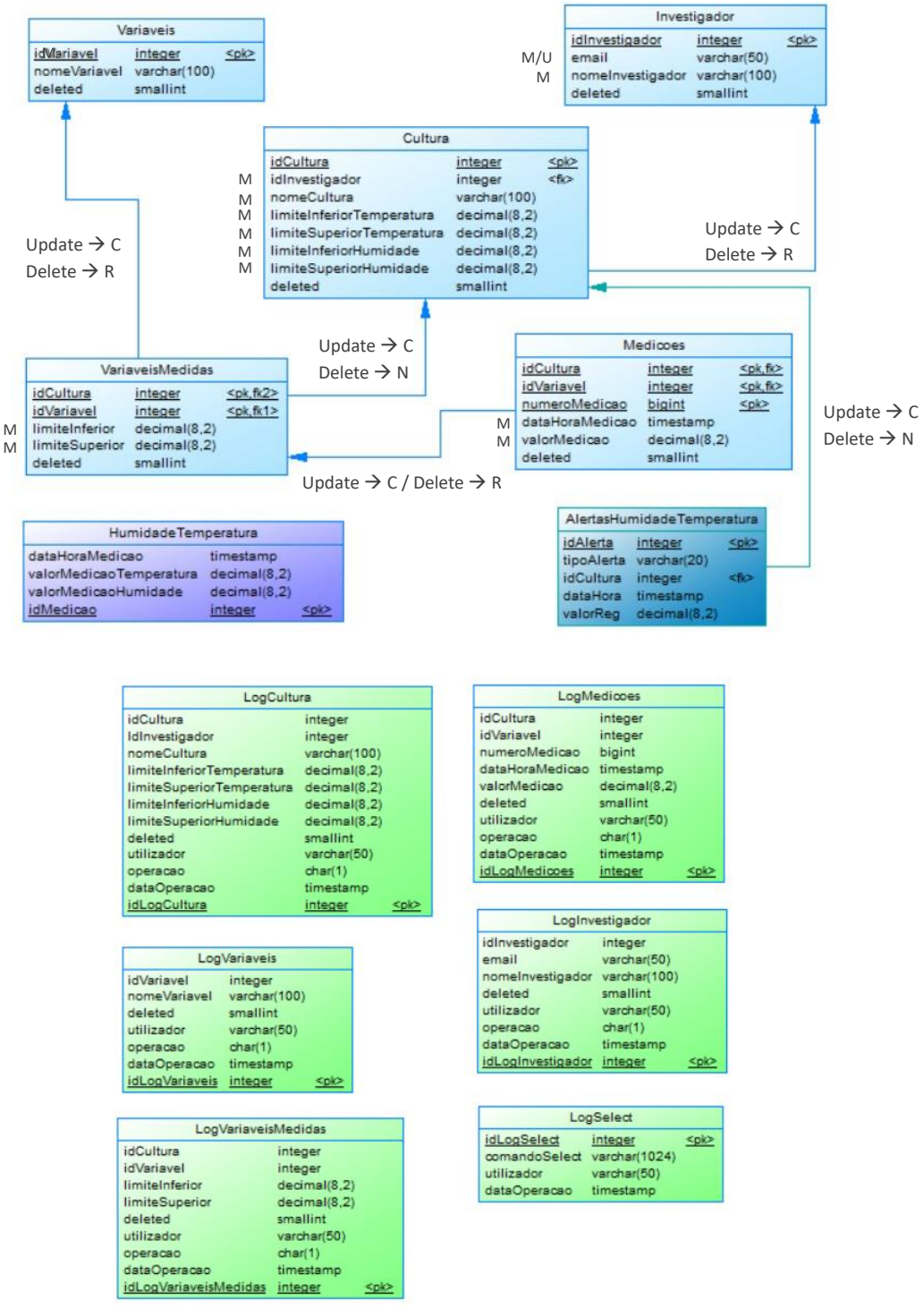


Figura 4 - Diagrama físico da BD Sybase

Devido à compatibilidade total dos dados provenientes do mongoDB, não se verificou qualquer necessidade de alterar as tabelas e os atributos, assim como o tipo de relação entre as mesmas e as respetivas restrições.

Relativamente a adições, foi necessário acrescentar uma tabela e funções de controlo, devido à necessidade de emissão de alertas para os investigadores. Esse acrescento encontra-se especificado abaixo e atualizado na figura acima.

1.4.2 Criação da tabela de Alertas

Face à necessidade de criação de alertas para os investigadores relativamente aos valores de temperatura e humidade registados pelos sensores, procedeu-se à alteração da estrutura da base de dados Sybase através da criação da tabela AlertasHumidadeTemperatura.

A tabela terá os seguintes atributos:

Nome	**Tipo**	**Descrição**	**FK**	**Mandatório**
idAlerta	integer (auto-inc)	Identificação única do alerta	Não	Sim
tipoAlerta	varchar(20)	Tipo de alerta ativado	Não	Sim
idCultura	integer	Referência à cultura se necessário	Sim	Não
dataHora	dateTime	Data e hora de ativação do alerta	Não	Sim
valorReg	decimal(8,2)	Valor de temperatura ou humidade	Não	Não

Nota 1: O atributo valorReg faz referência ao valor de temperatura ou humidade registados no momento da deteção do alerta, podendo ser retirado do registo de medição que despoletou o mesmo. No caso de alerta de falha de sensores ou sensor, o valor desse campo será Null.

1.4.3 Tipos de Alertas possíveis de detetar

Os tipos de alertas possíveis de serem detetados e registados são os seguintes:

NoDataAlert (geral) - é registado quando, num ciclo de envio de leituras do mongoDb, não se verificaram presença de registos dos sensores.

NoTempAlert (geral) - é registado quando, em três registos consecutivos, não se verificam valores de temperatura.

NoHumiAlert (geral) - é registado quando, em três registos consecutivos, não se verificam valores de humidade.

HighTempAlert (relativo a uma cultura) - é registado quando o valor da temperatura de um registo se encontra com uma diferença de 3 graus do limite máximo da cultura.

HighHumiAlert (relativo a uma cultura) - é registado quando o valor da humidade de um registo se encontra com uma diferença de 5 unidades do limite máximo da cultura.

LowTempAlert (relativo a uma cultura) - é registado quando o valor da temperatura de um registo se encontra com uma diferença de 3 graus do limite mínimo da cultura.

LowHumiAlert (relativo a uma cultura) - é registado quando o valor da humidade de um registo se encontra com uma diferença de 5 unidades do limite mínimo da cultura.

IncTempAlert (geral) - é registado quando, nos últimos 60 registos consecutivos, se verifica uma subida de temperatura de 1 grau.

DecTempAlert (geral) - é registado quando, nos últimos 60 registos consecutivos, se verifica uma descida de temperatura de 1 grau.

IncHumiAlert (geral) - é registado quando, nos últimos 60 registos consecutivos, se verifica uma subida de humidade de 2 valores.

DecHumiAlert (geral) - é registado quando, nos últimos 60 registos consecutivos, se verifica uma descida de humidade de 2 valores.

ReadTempErrorAlert (geral) – é registado quando, no caso de temperatura, se verifique uma variação de 20 graus face ao registo anterior. Este alerta previne erros pontuais de leitura.

ReadHumiErrorAlert (geral) – é registado quando, no caso de humidade, se verifique uma variação de 20 valores face ao registo anterior. Este alerta previne erros pontuais de leitura.

A título de exemplo, os registos de alertas na tabela AlertasHumidadeTemperatura deverão ser os seguintes:

idAlerta	tipoAlerta	idCultura	dataHora	valorReg
1	HighTempAlert	3	2018-05-11 12:11:32.000	25.6
2	DecHumiAlert	Null	2018-05-12 10:21:17.000	68.2
3	NoDataAlert	Null	2018-05-13 18:27:41.000	Null

Nota: Por forma a organizar e separar os novos alertas dos anteriores, será criado um sistema do lado do SQLlite que permite saber se um dado alerta já foi dado como lido por parte do investigador. Esse sistema foi colocado do lado do investigador por duas principais razões:

1. Assim, o investigador somente precisa de ter permissões de leitura da tabela de alertas, não podendo de forma alguma alterar o seu conteúdo. Caso o controlo dos alertas fosse do lado do Sybase, o investigador teria de dar a informação à base de dados, aumentando a complexidade das permissões e proteções;
2. Os alertas gerais poderão ser dados como lidos por investigador. No Sybase não seria fácil implementar uma solução que permitisse ao sistema saber que investigador já leu o alerta geral. Também se poderia optar por nunca colocar o alerta geral como lido, sendo ineficaz e pouco escalável para uma consulta eficiente dos alertas.

1.5 Periodicidade de Leitura de Mongo e Escrita no Sybase

O processo J1, é responsável, como indicado anteriormente, pelo processo de retirar os dados do MongoDB e inserir no Sybase. Para este caso decidimos que o processo J1, deverá retirar os dados do MongoDB de minuto em minuto, exportando assim para o sybase (no melhor caso possível), 12 registos de cada vez. Desta forma, garantimos que os dados estão sempre bastante atualizados, mas com a garantia também que não são efetuadas demasiadas querys ao sybase de forma a gerar um aumento significativo da memória disponível.

1.6 Triggers, SP e Views no Sybase

1.6.1 Triggers

Por forma a otimizar o processo de deteção e criação de alertas, optou-se por definir um trigger na tabela HumidadeTemperatura que faz uma verificação geral das medições em busca de possíveis alertas.

Nome Trigger	Tabela	Tipo de Operação	After / Before	Notas
tr_ins_HumiTemp	HumidadeTemperatura	Insert	After	

O trigger deverá, de forma hierárquica, chamar os respetivos stored procedures para deteção dos alertas e registo na tabela AlertasHumidadeTemperatura.

1.6.2 Stored Procedures

Como referido no capítulo dos triggers, os stored procedures aqui referidos serão utilizados para encapsular os vários mecanismos de deteção dos alertas. Cada SP será responsável por verificar uma série de alertas e, em caso positivo, registá-los na tabela de Alertas. A ação de distribuir a verificação dos vários alertas por mais do que um SP (ou trigger) traz vantagens, quer em termos de organização do código, quer em termos de chamadas hierárquicas dos próprios alertas (por exemplo, não permitir registar um alerta de temperatura alta [HighTempAlert], quando registar para o mesmo registo um erro de leitura [ReadErrorAlert], pois a informação é redundante).

A lista de stored procedures é a seguinte:

Nome SP	Parâmetro Entrada	Parâmetro Saída	Descrição
sp_NoDataAlert	Registo da tabela HumidadeTemperatura	Resultado da deteção de alertas: 0 – Com valores 1 – Sem temperatura 2 – Sem humidade 3 – Sem valores	Deteta e regista se existe ausência de leitura de temperatura e/ou humidade
sp_ReadErrorAlert	Registo da tabela HumidadeTemperatura e output do sp_NoDataAlert	Resultado da deteção de alerta: 0 – Valores sem erro 1 – Erro na temperatura 2 – Erro na humidade 3 – Erro nos dois	Deteta e regista se o valor dos sensores é inválido (variação > 20 na humidade e temperatura)
sp_TempAlert	Registo da tabela HumidadeTemperatura	Resultado da deteção de alerta: 0 – Sem alertas 1 – Com alertas	Deteta e regista valores limite ou variações de temperatura
sp_HumiAlert	Registo da tabela HumidadeTemperatura	Resultado da deteção de alerta: 0 – Sem alertas 1 – Com alertas	Deteta e regista valores limite ou variações de humidade

Os diversos tipos de alertas serão tratados pelos respetivos SP's conforme a tabela abaixo:

Tipo de Alerta	Stored Procedure responsável
NoDataAlert	sp_NoDataAlert
NoTempAlert	sp_NoDataAlert
NoHumiAlert	sp_NoDataAlert
ReadTempErrorAlert	sp_ReadErrorAlert
ReadHumiErrorAlert	sp_ReadErrorAlert
HighTempAlert	sp_TempAlert
LowTempAlert	sp_TempAlert
IncTempAlert	sp_TempAlert
DecTempAlert	sp_TempAlert
HighHumiAlert	sp_HumiAlert
LowHumiAlert	sp_HumiAlert
IncHumiAlert	sp_HumiAlert
DecHumiAlert	sp_HumiAlert

1.6.3 Hierarquia de Store Procedures

Visto existirem alertas que, pela natureza dos mesmos, são considerados eliminatórios para outros, existe a necessidade de criar uma hierarquia de chamada das store procedures por parte do trigger tr_ins_HumiTemp. Por exemplo, caso não existam valores de temperatura num registo (por motivo de falha), não vale a pena chamar o sp_TempAlert. Nesse caso, a hierarquia das chamadas dos SPs pelo trigger é a seguinte:

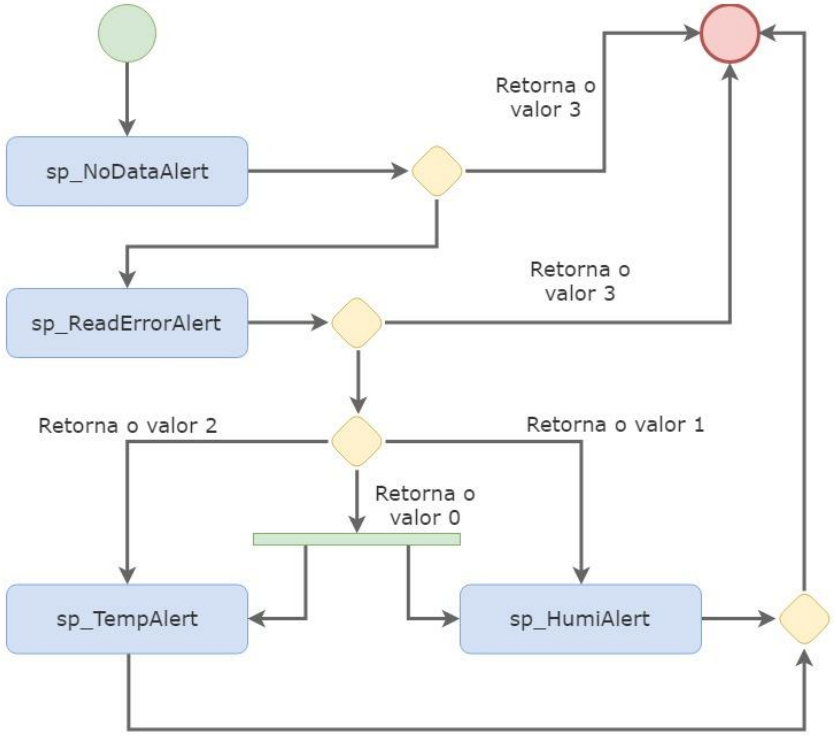


Figura 5 - Hierarquia de chamada dos SP's pelo trigger tr_ins_HumiTemp.

1.6.4 Views

Por forma a garantir que um investigador não consulta os alertas de um outro investigador, à semelhança da situação verificada na tabela Medicoes anteriormente especificada, optou-se por criar uma View que mostra somente os alertas gerais ou específicos de culturas pertencentes ao investigador em causa. Dessa forma o utilizador apenas terá permissões para executar a view v_AlertasPorInvestigador.

Nome View	**Tabela respeitante**	**Notas**
v_AlertasPorInvestigador	AlertasHumidadeTemperatura	

1.7 Utilizadores relevantes no Sybase e respetivos privilégios

Como referido no relatório anterior, existirá um utilizador que terá a função de escrever dados na tabela "HumidadeTemperatura", sendo este designado como MongoDB, que será utilizado pelo processo Java 1 (ligação entre MongoDB e Sybase), para transporte de dados. O mesmo, como especificado no relatório anterior, irá precisar de apenas a permissão de INSERT na tabela" HumidadeTemperatura".

No caso da tabela AlertasHumidadeTemperatura, por forma a cumprir com a política de proteção de dados anteriormente especificada, somente existirá permissão de select por parte dos administradores e select/delete por parte do administrador principal. Os investigadores somente poderão ter acesso à informação da tabela de alertas através da view v_AlertasPorInvestigador.

Visto os stored procedures serem somente chamados pelo trigger tr_ins_HumiTemp, estes não têm permissões de call por parte de qualquer utilizador.

Permissões	**Administrador Principal**	**Administradores**	**Investigadores**	**MongoDB**
Tabelas				
HumidadeTemperatura	-	-	-	Insert
AlertaHumidadeTemperatura	Select, HardDelete	Select	-	-
Views				
v_AlertasPorInvestigador	-	-	Select	-
Store Procedures dos Alertas	-	-	-	-

1.8 Qualidade de Transmissão dos Dados (QoS)

O protocolo MQTT - *Message Queue Telemetry Transport* - oferece três níveis de Qualidade de Serviço (QoS) para entregar mensagens entre clientes e servidores: QoS0 (ou "no máximo uma vez"), QoS1 ("pelo menos uma vez") e QoS2 ("exatamente uma vez")..

A QoS é um atributo de cada mensagem publicada, sendo que:

No nível **QoS0** a mensagem é entregue no máximo uma vez e poderá nem ser entregue, um vez que não é armazenada, nomeadamente no caso de o cliente *subscriber* estar desligado. Este nível de QoS é obviamente mais rápido e computacionalmente menos dispendioso.

No nível **QoS1** a mensagem é sempre entregue pelo menos uma vez. Isto significa que o remetente armazenará a mensagem até obter confirmação de que a mensagem foi publicada pelo destinatário; se não a obtiver, tentará entregá-la de novo.

Com o nível **QoS2**, de modo semelhante ao que acontece com o nível QoS1, a mensagem é armazenada localmente no remetente, até que este receba a confirmação de que a mensagem foi publicada pelo destinatário, garantido além disso que não ocorre nenhuma duplicação de mensagens. O QoS2 é o modo de transferência mais seguro, mas é também o mais lento e o que tem maiores custos.

Para o processo que se pretende implementar com esta especificação, afigura-se que não há necessidade de prevenir a duplicação de mensagens recorrendo ao mecanismo implementado com o nível QoS2, uma vez que a simples criação de um índice único por dataHora na *collection* definida no MongoDB elimina essa necessidade, além de que o sistema previsivelmente toleraria um certo número de repetições de valores das medições, sem que nada de substantivo se alterasse. Por outro lado, também se julga aceitável que o sistema perca um certo número de mensagens, no caso de ficar desligado, dado que o sistema se destina essencialmente à monitorização em tempo quase real das culturas. Ora, se o sistema recetor das mensagens estiver desligado, não providenciará esse serviço de monitorização e, se o não estiver, a perda de algumas mensagens não se afigura crítica, tendo em conta a frequência esperada do envio das mesmas.

Tudo ponderado, considera-se aceitável optar pelo nível QoS0, por ser mais rápido, computacionalmente menos oneroso e não ter como implicação nenhuma desvantagem operacional relevante.

1.9 Gestão de Logs nos Processos J1 e J2

Para um melhor funcionamento dos processos, e para que seja facilitada as ações de debugging, é aconselhado a acrescentar mais um ficheiro, junto aos explicados anteriormente. Os ficheiros deverão ser ficheiros de texto, com extensão .log. Estes ficheiros deverão ter uma linha para cada ação de import ou export dos dados, erros ou outros casos importantes, como inicio e fim do programa, de ambas as bases de dados. Para o processo j1, deve ser criado o ficheiro j1.log, e para o processo 2 deve ser criado o ficheiro j2.log.

A estrutura que deve ser seguida para cada linha dos ficheiros de logs é a seguinte:

dd-mm-aaaa hh:mm:ss || TEXTO_PARA_O_EVENTO

De seguida mostra-se um diagrama relativo aos ficheiros necessários para um sistema completo:

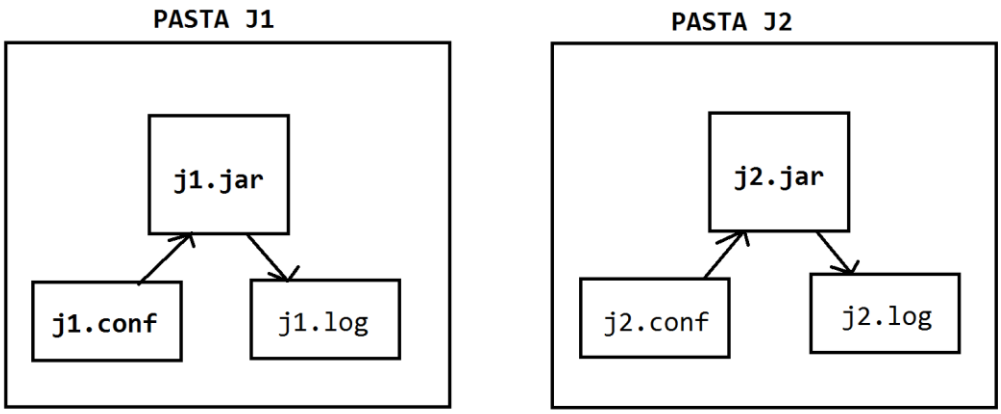


Figura 6 - estrutura de ficheiros dos processos J1 e J2.

1.10 Avaliação Global da Qualidade das Especificações do próprio grupo

Avaliação (A,B,C,D,E) : _____

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

Análise crítica (clareza, completude, rigor):

1.11 Implementação

1.11.1 Código Mongo Implementado (dentro do java)

<Listar todo o código Mongo utilizado no processo, quer para importar, quer para exportar. O código tem de ser comentado para que se torne legível para quem sabe uns rudimentos de MongoDB. Fragmentos de código java apenas serão mostrados para dar algum contexto >

1.11.2 Divergências face ao especificado

<Indicar as divergências relevantes (ignorar pequenos detalhes de implementação) face à especificação recebida, nomeadamente as que consideram que permitiu chegar a uma solução melhor.>

1.11.3 Código SQL Implementado

<Listar todo o código SQL utilizado no processo de colocação de inserção nas tabelas SQL Anywhere. O código tem de ser comentado para que se torne legível para quem sabe SQL. Os comentários não podem ser redundantes, colocar apenas o essencial. Indicar triggers ou eventos no lado Sql Anywhere, se existirem>

1.11.4 Divergências face ao especificado

<Indicar as divergências relevantes (ignorar pequenos detalhes de implementação) face ao especificado, nomeadamente as que consideram que permitiu chegar a uma solução melhor.>

2 Android e Php

2.1 Esquema da BD Lite Geral

<Modelo relacional implementado no Android, tabelas e atributos>

2.2 Layout Implementado no Android

<PrintScreen de um exemplo de interacção>