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 |
| 72898 | Diana Lopes |  |
| 73529 | Kevin Batista |  |
| 72973 | Ricardo Santos |  |
| 73036 | Gustavo Morais |  |

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

[1 Mongo DB 7](#_Toc514013867)

[1.1 Descrição Geral do Procedimento 7](#_Toc514013868)

[1.2 Estrutura da Base de Dados Mongo 9](#_Toc514013869)

[1.3 Periodicidade de Leitura de Sensores e Escrita no Mongo 10](#_Toc514013870)

[1.4 Estrutura da Base de Dados Sybase 11](#_Toc514013871)

[1.5 Periodicidade de Leitura de Mongo e Escrita no Sybase 12](#_Toc514013872)

[1.6 Triggers, SP ou eventos no Sybase (caso relevante) 13](#_Toc514013873)

[1.7 Utilizadores relevantes no Sybase e respectivos privilégios 14](#_Toc514013874)

[1.8 Avaliação Global da Qualidade das Especificações do próprio grupo 15](#_Toc514013875)

[1.9 Implementação 16](#_Toc514013876)

[1.9.1 Código Mongo Implementado (dentro do java) 16](#_Toc514013877)

[1.9.2 Divergências face ao especificado 17](#_Toc514013878)

[1.9.3 Código SQL Implementado 18](#_Toc514013879)

[1.9.4 Divergências face ao especificado 19](#_Toc514013880)

[2 Android e Php 20](#_Toc514013881)

[2.1 Esquema da BD Lite Geral 20](#_Toc514013882)

[2.2 Layout Implementado no Android 21](#_Toc514013883)

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

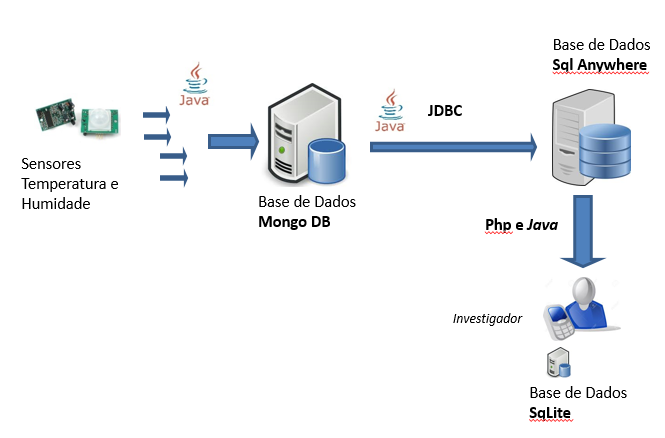


Sensor

*Exemplo Mensagens*

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

Esquema de Importação e Migração



# Mongo DB

## Descrição Geral do Procedimento

No main serão instanciadas 5 classes:

* 3 classes que tratam de cada conexão: ao Sensor, Mongo e Sybase (estas últimas requerem autenticação);
* 1 classe que inicia a Thread que envia dados para o sybase;
* 1 classe que lida com a sincronização das Threads, para não haver colisão na escrita e leitura no mongo.

Na classe onde se estabelece a conexão com o Paho, as informações do sensor irão ser recebidas através do messageReceived(), que é um método da interface MqttCallback.

O messageReceived() recebe a mensagem e chama um método, presente na classe de sincronização de threads, que irá passar as informações ao mongo (este método recebe a mensagem como argumento).

Estando estabelecidas todas as ligações, basta criar uma thread que obtém dados do mongo (através de um método na classe onde ocorre a sincronização de Threads) e envia-os para o Sybase. Não é necessário criar uma thread para passar mensagens do sensor para o mongo porque o messageReceived() é um listener com a sua própria thread.

Para que a exportação seja incremental, acrescenta-se no documento o campo ‘migrated’ com o valor default false (cada vez que se insere um documento vindo do sensor no mongo). Só serão exportadas para o Sybase as linhas do mongo DB cujo valor do campo ‘migrated’ for false. Cada linha, depois de ser exportada, irá ser atualizada, passando o valor do campo ‘migrated’ a true.

Para além das bibliotecas que estão indicadas no e-learning: **org.eclipse.paho.client.mqttv3-1.1.0.jar** e **mongo-java-driver-2.11.3.jar**, foi instalada a biblioteca **sajdbc4.jar** (driver que estabelece a conexão JDBC ao Sybase).

Haverá também um ficheiro de configuração do sistema com as seguintes informações:

* Nome da DB Mongo e da sua coleção;
* Nome da DB Sybase e da sua tabela;
* Nome de utilizador e password de acesso ao:
  + Sybase (JDBC user);
  + Mongo;
* Tópico de subscrição;
* Periodicidade de exportação do Mongo para o Sybase;
* Mongo client URI (informação para a conexão ao mongo);
* Mqqt client link (informação para a conexão ao sensor);
* Path para a DB Sybase (ver imagem abaixo)

Fica ao critério de quem implementar juntar o nome da DB sysbase com path do último ponto em vez de dar os dois valores em separado.

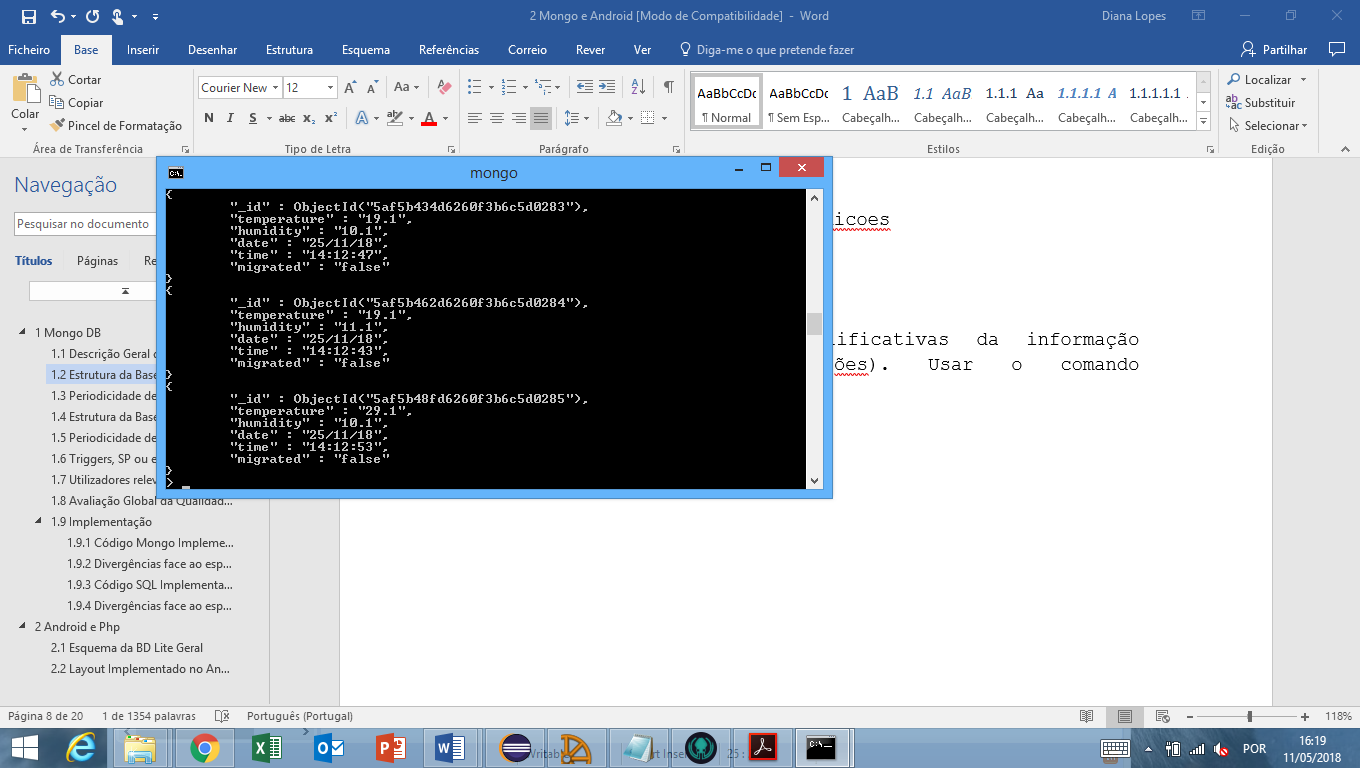


Este ficheiro é lido quando se inicia a aplicação.

## Estrutura da Base de Dados Mongo

Nome da base de dados: SensorMedicoes

Nome da coleção: HumidadeTemperatura



## Periodicidade de Leitura de Sensores e Escrita no Mongo

A cada 5s o sensor envia dados para o Java. O Java recebe as mensagens do sensor através do método messageArrived(), que está na classe que estabelece a conexão com o Paho.

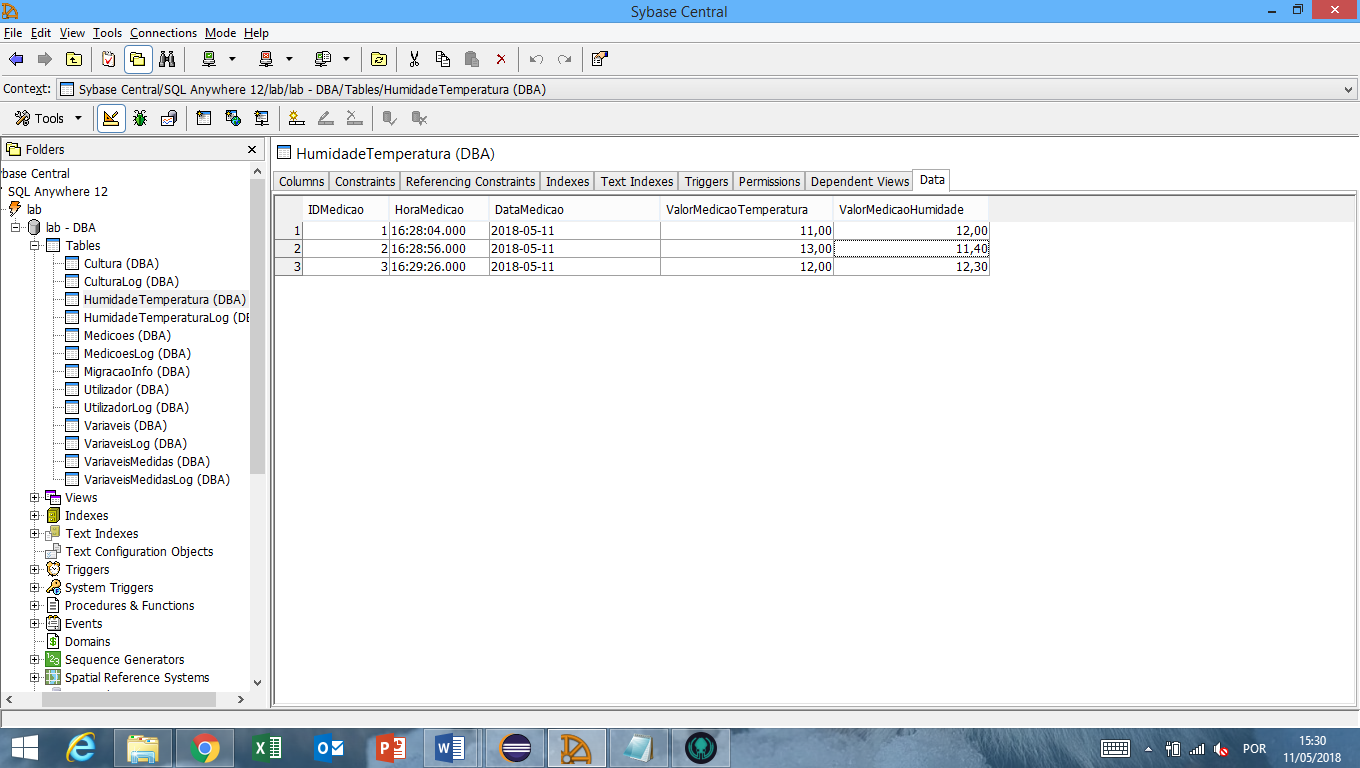
Será necessário tratar dos valores da mensagem, verificando potenciais exceções, isto é, dados mal inseridos que iriam dar erro no Sybase, fazendo o programa parar.

Exemplos de algumas validações a considerar:

* Analisar a estrutura do objeto JSON para ver se corresponde ao esperado;
* Verificar se cada par: key value tem o tipo de dados correto (por exemplo, double na temperatura);
* Verificar se a data está num espaço de tempo que faça sentido (não demasiado velha ou no futuro) ou estar a null;
* Verificar se a hora da medição é válida, não podendo ser null e o campo das horas tem estar entre 0 e 24 e o dos minutos e segundos estar entre 0 e 59.
* Verificar se os valores da temperatura e da humidade são valores dentro de um padrão minimamente normal (podem ser negativos).

Os valores são tratados e posteriormente exportados para o mongo, como referido no ponto 1.1, na classe que lida com as Threads, fazendo com que não ocorra colisão nas operações de escrita e leitura no mongo.

## Estrutura da Base de Dados Sybase

Os valores do sensor são inseridos através de JDBC na tabela HumidadeTemperatura:

## Periodicidade de Leitura de Mongo e Escrita no Sybase

O Mongo exporta os dados para o Sybase a cada 20s, através de JDBC. Este valor foi escolhido para reduzir a quantidade de vezes que se enviam os dados, mas mantendo o utilizador ciente de que estão a ocorrer atualizações.

A Thread que envia os dados para o sybase irá chamar, a cada 20s, um método da classe que lida com a sincronização das Threads.

Esse método irá buscar ao mongo os dados que ainda não foram migrados.

Como explicado no ponto 1.1, cada linha que está na coleção do mongo é exportada se tiver o valor do campo ‘migrated’ a false.

Este método devolve os dados que serão enviados para o Sybase. Depois de exportados, cada linha migrada irá ser atualizada, passando o campo ‘migrated’ para true.

A tabela onde serão inseridos os dados provenientes do sensor é a HumidadeTemperatura.

## Triggers, SP ou eventos no Sybase (caso relevante)

<Especificar que trigger ou SO pretendem que sejam implementados >

## Utilizadores relevantes no Sybase e respectivos privilégios

|  |  |  |
| --- | --- | --- |
| **Tabela** | **Tipo de Utilizador** | |
| Investigador | JDBC user |
| Cultura | - | - |
| Investigador | - | - |
| HumidadeTem-peratura | L | E |
| **Stored Proc.** |  |  |

Normalmente haveria um SP para permitir aos investigadores fazer leituras e inserções nas tabelas que a eles não têm acesso direto, mas como essa funcionalidade não é relevante para esta parte do trabalho, sendo apenas relevante no contexto da antiga especificação, esse SP não é especificado.

O JDBC user é utilizado pela aplicação java para escrever dados no Sybase.

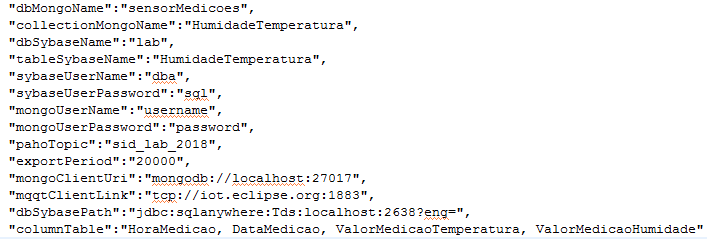
## Avaliação Global da Qualidade das Especificações do próprio grupo

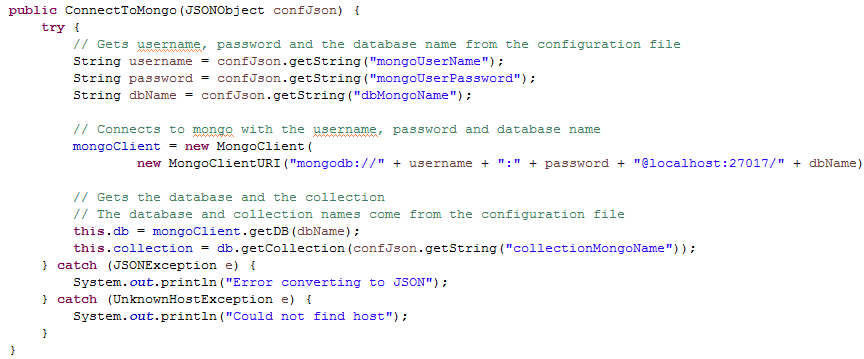
|  |
| --- |
| Avaliação (A,B,C,D,E) : E  Utilize a seguinte escala:  A: - 1 – 5 valores B: 6 – 9 valores C: 10 – 13 Valores D: 14 – 17 valores E: 18 – 20 valores  **Análise crítica (clareza, completude, rigor):** |

## Implementação

### Código Mongo Implementado (dentro do java)

Ficheiro de configuração, que irá ser referido ao longo do relatório:



Na classe ConnectToMongo, tal como foi especificado, a conexão ao mongo é feita com autenticação (como se pode ver pelo código dentro do retângulo vermelho).

Para tal, teve de ser criado um utilizador no mongo:



Dentro da BD sensorMedicoes, é criado um utilizador com o nome “username” e password “password”. As suas permissões são apenas de escrita e leitura.

Para haver esta segurança de acesso à BD, é necessário que o servidor corra com o modo de autenticação:

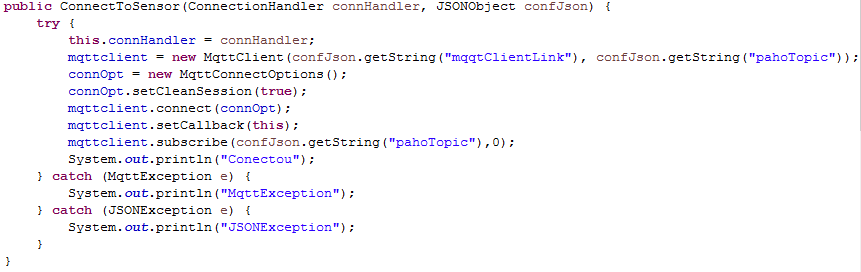


Na classe que trata da ligação do Java com o paho: ConnectToSensor, onde o que é importante salientar, é que o tópico pode ser mudado, visto que está definido no ficheiro de configuração e o QoS está a 0.

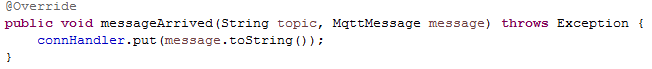
Com o QoS a 0, cada mensagem é enviada uma vez, no máximo, e se o cliente não estiver conectado ao paho aquando o envio da mensagem pelo sensor, perde-a, visto que não será confirmada pelo destinatário nem será armazenada ou devolvida pelo remetente.

Com o QoS a 0, tem-se um modo de transferência de mensagens mais rápido que o QoS a 1 ou a 2.

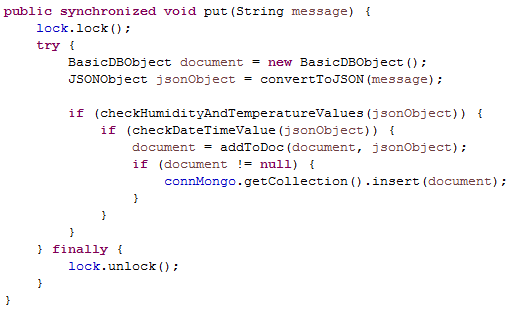
Adotou-se este mecanismo, visto o principal objetivo é que a mensagem chegue atualizada e não, que ela seja efetivamente entregue. Além disso, o envio do sensor é a cada 5s, não sendo muito prejudicial a perda de algumas mensagens.



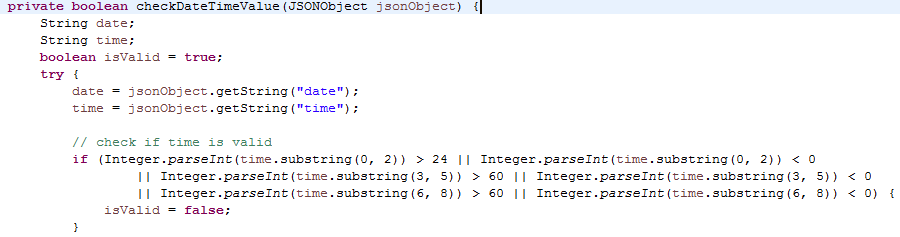
No messageArrived(), que é um método da interface MqttCallback, é chamado o método da classe que lida com as threads (ConnectionHandler) que irá tratar dos valores provenientes do paho e inseri-los no mongo:



Na classe ConnectionHandler existe este método sincronizado, de modo a que a escrita e a leitura no mongo não colidam. Chama 2 métodos de verificação de valores: verificação da humidade e temperatura – checkHumidityAndTemperatureValues e verificação da data e hora – checkDateTimeValues. Se estes 4 valores estiverem válidos, são adicionados a um BasicDBObject, e se este não estiver nulo é inserido na coleção do mongo definida anteriormente.

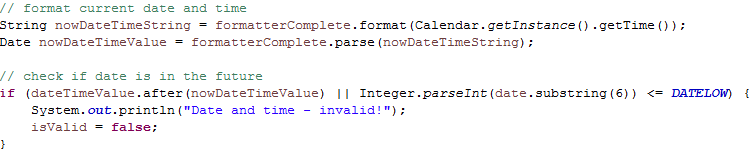


Verificação da data e hora:

Primeiramente, é verificado se as horas, minutos e segundos têm valores validos (horas só podem ter valores entre 0 e 24 e os minutos e segundos só podem ter valores entre 0 e 60).

Posteriormente, o valor da data e hora são convertidos para o seu respetivo formato, de modo a verificar se são data e horas válidas.

A data/hora que chega do sensor é comparada com a atual, e se for superior que a atual, isto é, a data for do futuro, é descartada. Se a data for muito antiga, também é descartada. O valor DATELOW é 1970, visto que é a partir deste ano que as datas são aceites no Sybase.

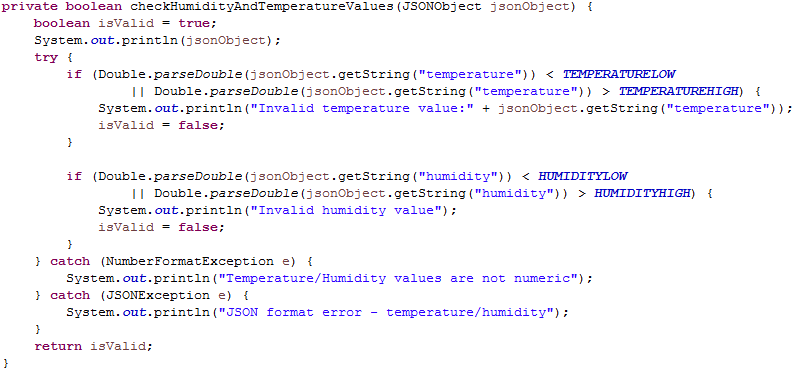


Se todas as verificações forem concretizadas, o método retorna true.

Verificação dos valores da humidade e temperatura:

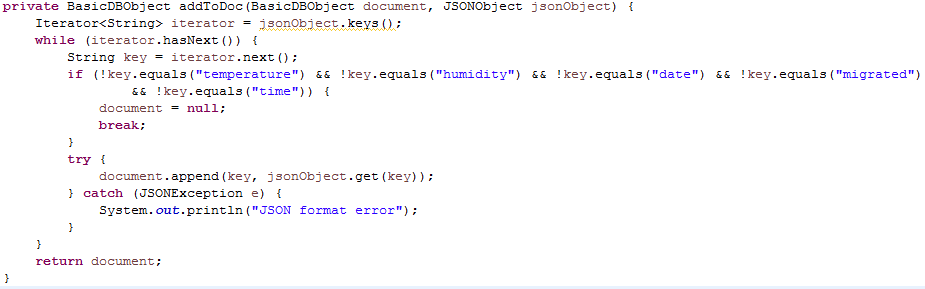
Os valores da humidade e temperatura são convertidos em double, verificando-se assim se estão no formato correto.

Os valores da humidade que não estiverem entre 0 e 100 são descartados, visto que a humidade é expressa em percentagem.

Os valores da humidade que não estiverem entre -273 (zero absoluto) e 150 são descartados, visto que quase nenhuma cultura sobrevive a temperaturas tão extremas:

Se todas as verificações forem concretizadas, o método retorna true.

Se os dois métodos anteriores retornarem true, então é adicionado ao BasicBDObject os valores da mensagem. É verificado se a estrutura JSON é a correta (não é permitindo que haja chaves no JSON que não sejam as definidas: temperature, humidity, date, time e migrated):



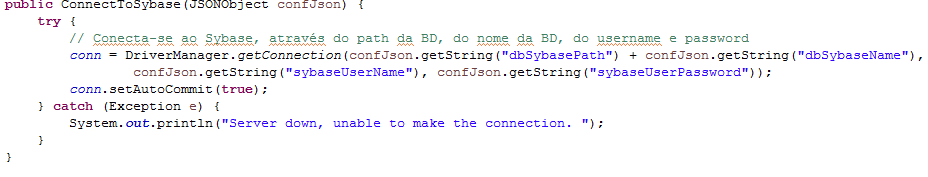
Caso o BasicDBObject não seja nulo, a mensagem é inserida no mongo.

### Divergências face ao especificado

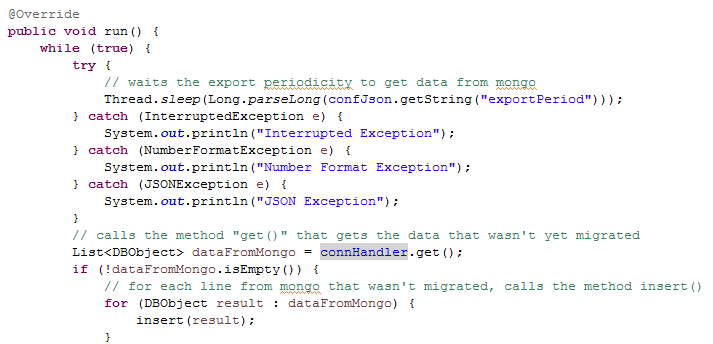
A especificação usada para a implementação foi a nossa, e não a recebida, visto que achamos a nossa implementação melhor que a deles, pelas razões referidas a seguir:

1. Para a migração incremental, o outro grupo move os dados de uma coleção para a outra, enquanto que nós apenas alteramos o valor de um campo, sendo a nossa operação mais rápida e gasta menos recursos (só temos uma coleção e não usamos stacks)
2. Apesar de não termos especificado o QoS, implementamo-lo com o valor de 0, e não 2, como o outro grupo especificou. Isto porque, com o QoS a 0, a transferência de mensagem é mais rápida (não existe armazenamento ou devolução pelo remetente nem confirmação pelo destinatário), sendo a contrapartida a perda de mensagens, caso o Java não esteja conectado ao sensor. Como o principal objetivo é que os dados cheguem atualizados ao Sybase, o Qos = 0 é o ideal, visto que os dados irão chegar mais rapidamente que o QoS a 1 ou 2 e não é preocupante a perda de mensagens, visto que é preferível do que chegarem dados desatualizados ao Sybase.

### Código SQL Implementado

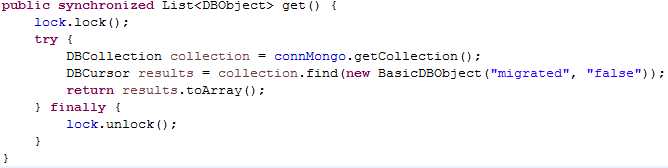
Na classe ConnectToSybase, tal como foi especificado, a conexão ao sybase é feita com autenticação:

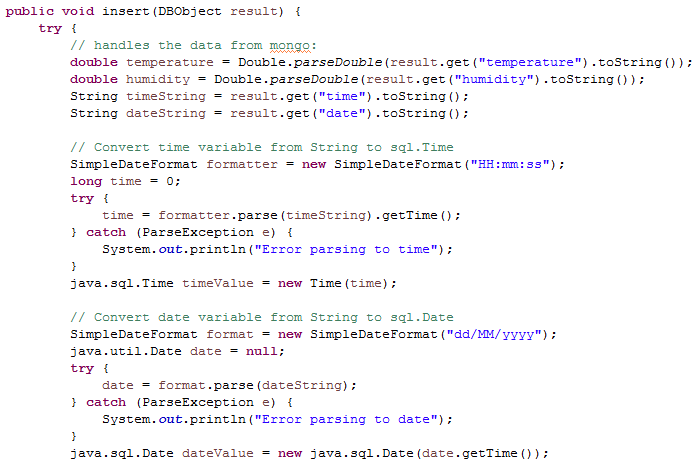
Na classe SendToSybase, no método run (override da implementação de um Runnable), a cada 20s (periodicidade definida no ficheiro de configuração), é chamado o método que está na classe ConnectionHandler, de modo a ir buscar os valores do mongo que ainda não foram exportados. Só se houver valores para serem exportados, é que é chamado o método para inseri-los no Sybase.



O método get() está na classe ConnectionHandler e é um método sincronizado, de modo a não haver colisão de threads.

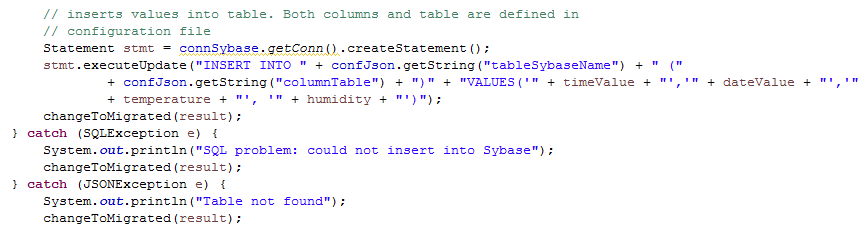
Este método vai buscar ao mongo os valores que ainda não foram exportados, isto é, têm o campo migrated a false e retorna uma lista de BasicDBObject:



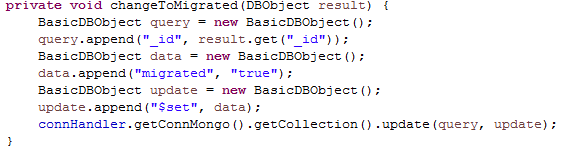
No método insert, da classe SendToSybase, os valores são convertidos de String para date, time ou double, dependendo do tipo de dados a tratar:

Os dados anteriormente tratados irão ser inseridos no Sybase. A tabela onde irão ser inseridos é definida no ficheiro de configuração e as colunas também, de modo a tornar esta solução flexível para diferentes tabelas, e consequentemente para diferentes colunas.

Depois da mensagem ser inserida no Sybase, o campo migrated passa a true, de modo a assegurar a exportação incremental dos dados, não chegando dados repetidos ao Sybase.

Se por algum motivo, houver erro na inserção no Sybase, a mensagem é alterada na mesma (campo migrated passa a true), visto que não vale a pena ficar à espera de ser migrada, pois quando o fosse, provavelmente seria uma mensagem desatualizada:

O método changeToMigrated, atualiza a mensagem, mudando o campo migrated para true:



Trigger implementado:

O trigger 'Limites' é utilizado para verificar as medições de Humidade e Temperatura perante os limites definidos em cada cultura.

Configurou-se o trigger para 'AFTER INSERT' porque compara a última medição inserida com as medições realizadas anteriormente dentro de um intervalo de 30 segundos. O intervalo é de 30s, visto que a periodicidade com que o mongo exporta para o sybase é 20s e estamos a dar uma margem de 10s, caso tenha ocorrido algum atraso.

É criado um cursor para a tabela 'HumidadeTemperatura' que faz o select dos valores de medição de temperatura e humidade realizados 30 segundos antes do novo campo adicionado, tal como explicado anteriormente. A tabela é ordenada por ordem ascendente da hora. Inicialmente é comparado se o ano e mês é igual, depois o dia, hora e minutos são convertidos para segundos de modo a calcular a diferença (é calculado desta forma devido o facto que poderá ser realizado uma medição ás 00:00:05, por exemplo, o que causaria erros de cálculo devido a mudança de dia).

Após a criação do cursor é verificado se existe crescimento ou redução do perigo nas medições, comparando o novo campo adicionado com todas as temperaturas e humidades presentes no cursor 'HumidadeTemperatura'. A tabela 'Cultura' é percorrida para analisar todos os limites individualmente.

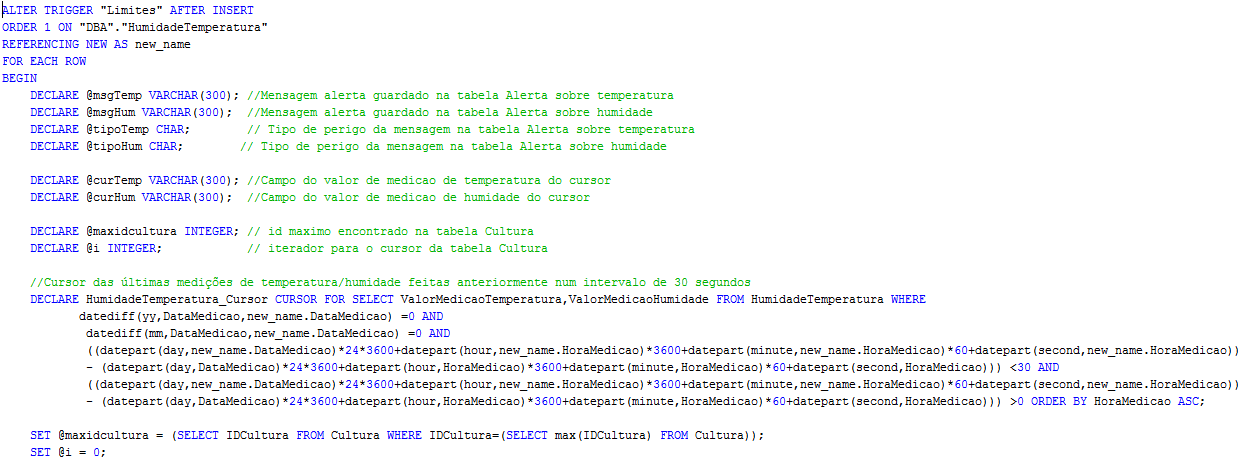
Se é verificada alguma alteração, é registada na tabela 'Alertas' esta mudança.

No campo ‘Alerta’ da tabela 'Alertas' é inserida a última alteração de temperatura ou humidade verificada dentro do cursor que ultrapassou o limite ou está perto (75% do limite superior ou menos de 25% do limite inferior) de algum limite definido em alguma cultura. No campo ‘Tipo’ é inserido o tipo de alerta:

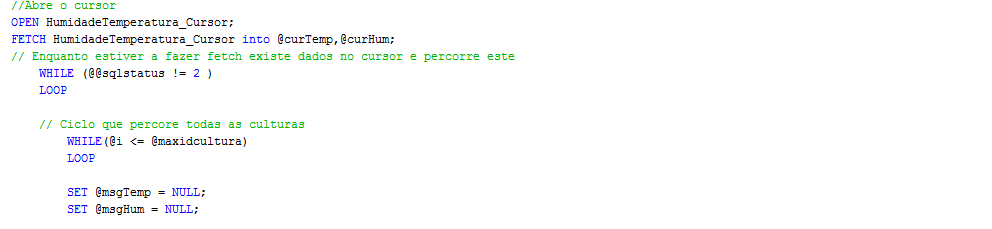
- R (Red): para quando a temperatura ou humidade passam dos limites (superior ou inferior)

- O (Orange): para quando a temperatura ou humidade estão perto dos limites (superior ou inferior)

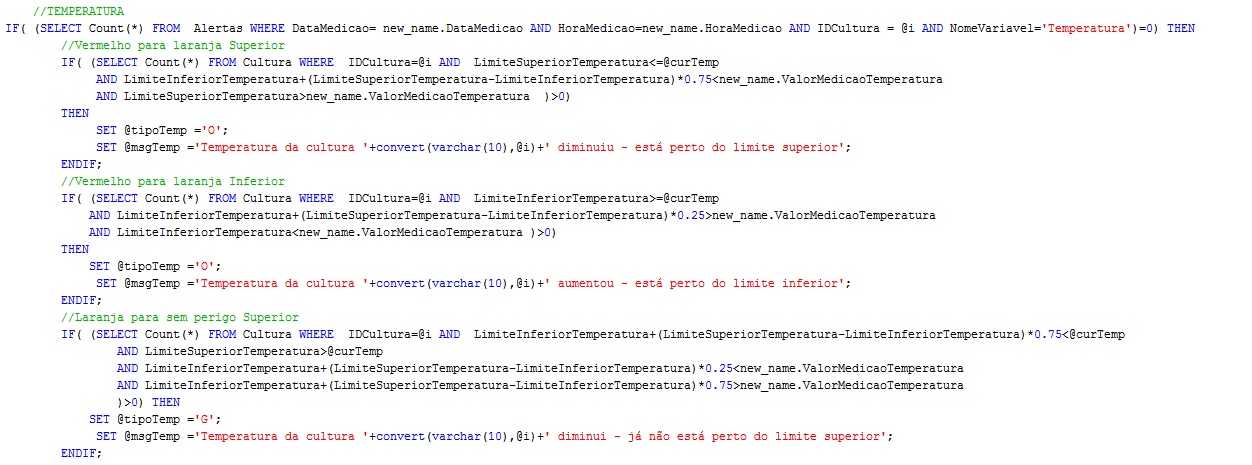
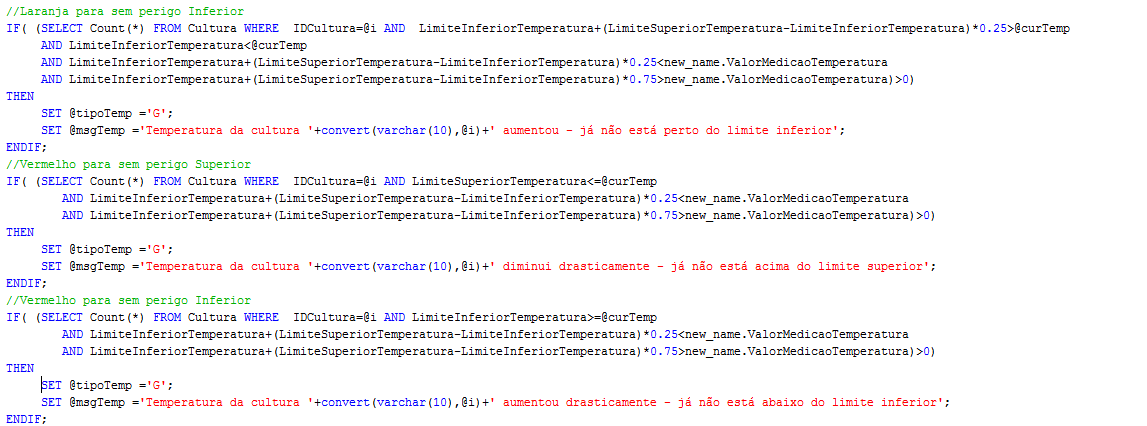
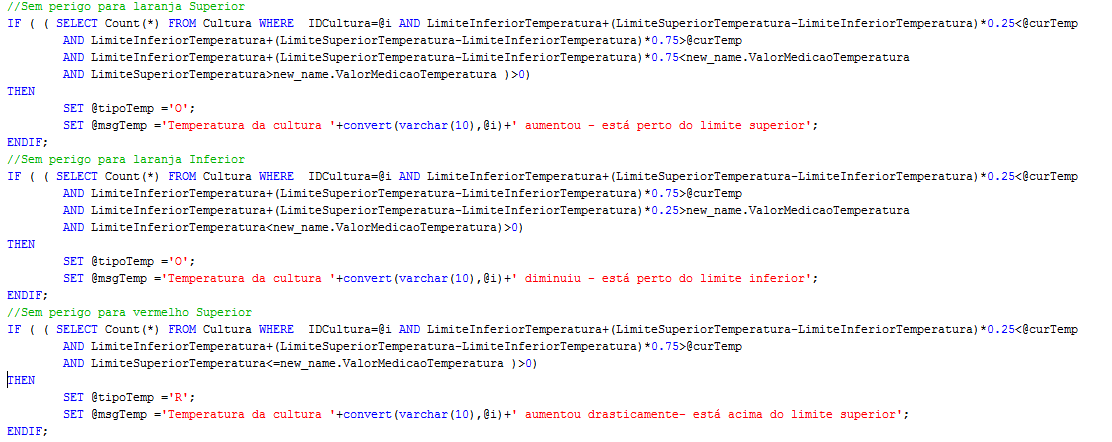
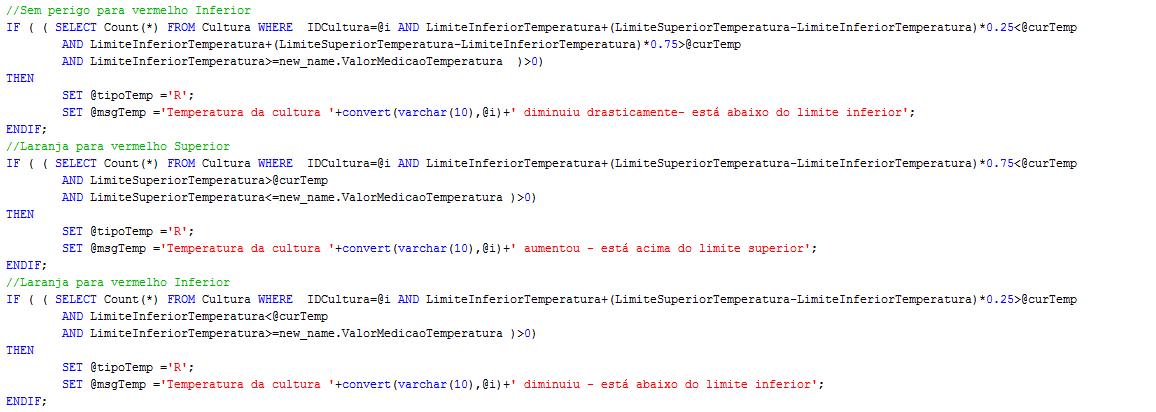
- G (Green): para quando a temperatura ou humidade deixam de ficar em perigo (passa de ‘perto do limite’ para dentro de limites sem perigo ou passa drasticamente de ‘limite ultrapassado’ para dentro de limites sem perigo)

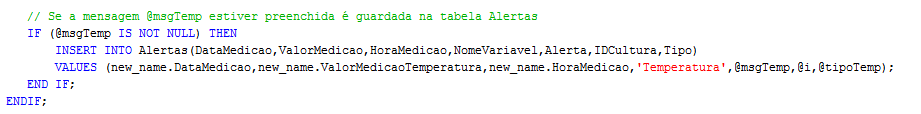


Loop pelos dados no cursor e dentro desse, existe um loop dentro:

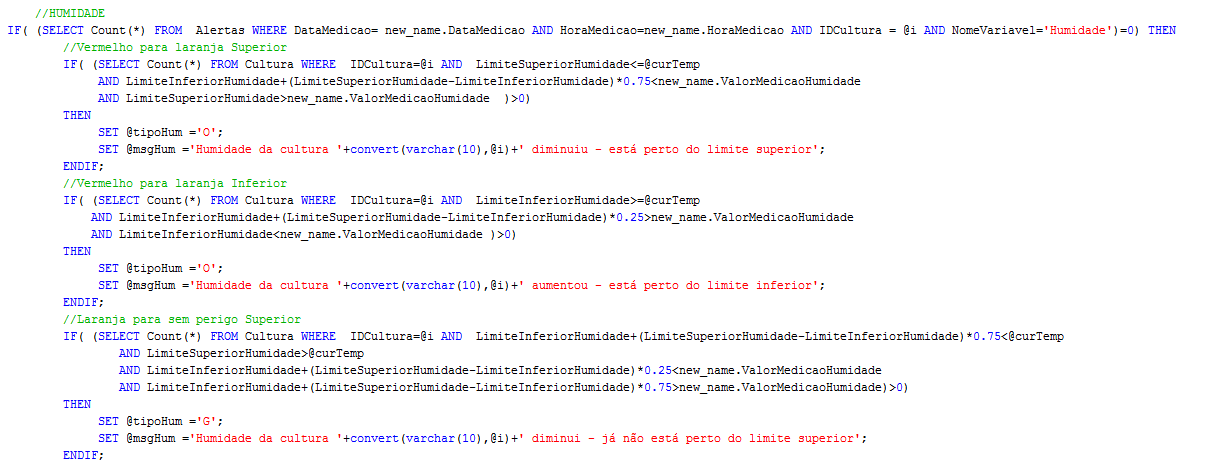
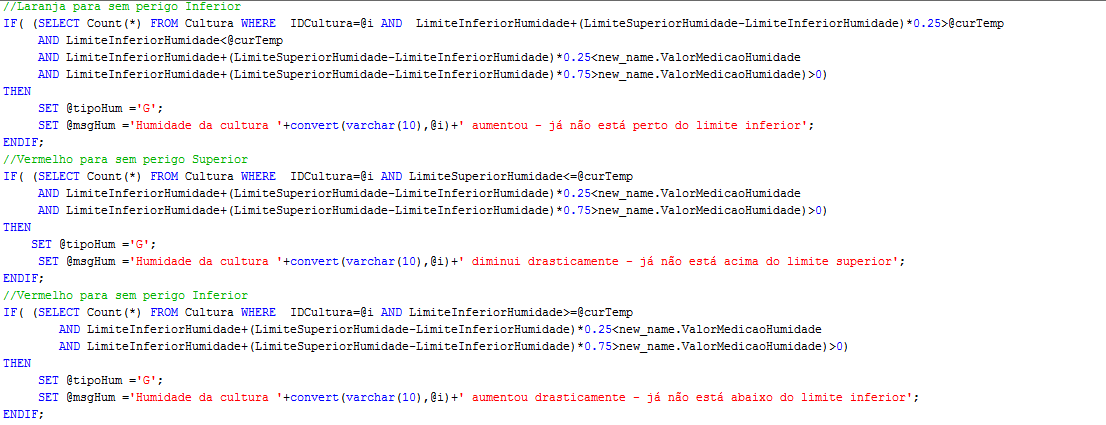
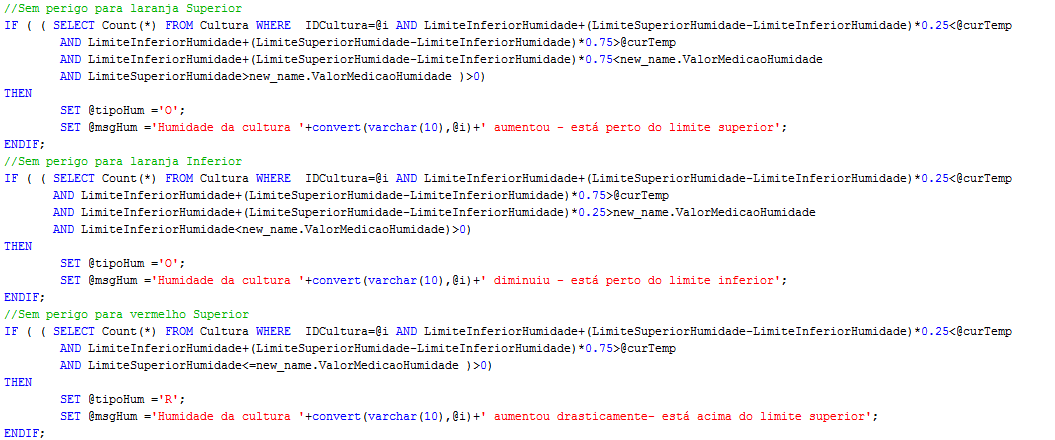


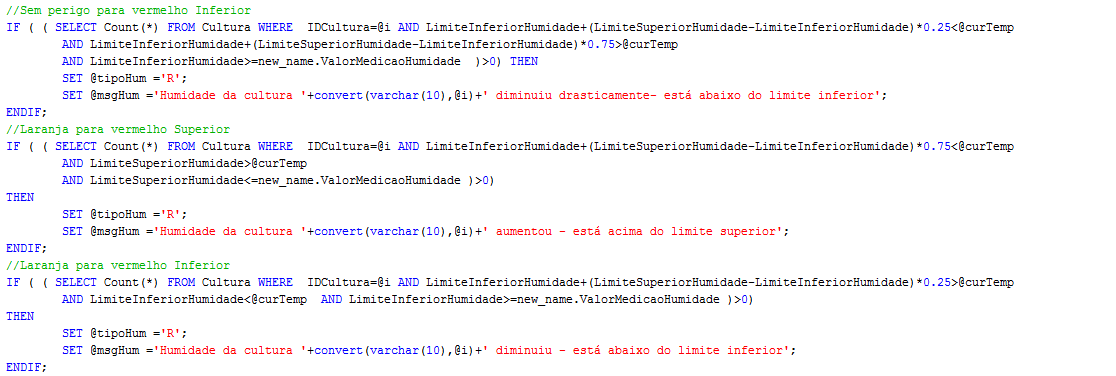
Verificação de mudanças na temperatura, se sim, adiciona-se o tipo de alerta na tabela (O – Orange, R - Red e G - Green) e a respetiva mensagem de alerta no campo ‘Alerta’.

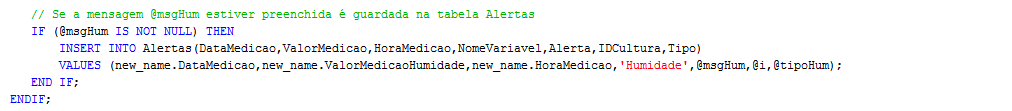
Preenchimento da tabela ‘Alertas’:

Verificação de mudanças na humidade, se sim, adiciona-se o tipo de alerta na tabela (O – Orange, R - Red e G - Green) e a respetiva mensagem de alerta no campo ‘Alerta’.



Preenchimento da tabela ‘Alertas’:

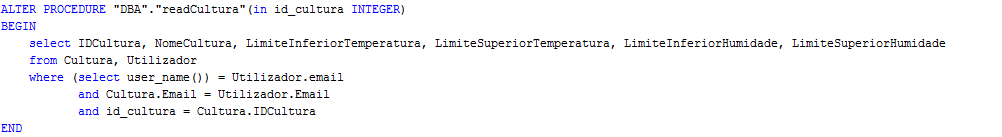


Fecho do cursor e removê-lo da memória:

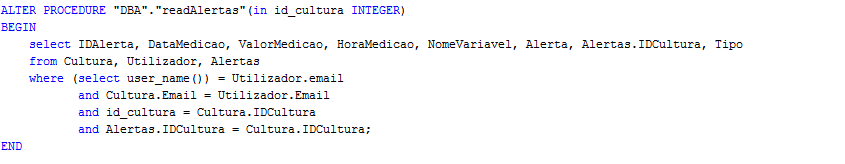


Stored Procedures implementados:

- readCultura: SP que é executado pelo utilizador de modo a que este tenha acesso aos valores da sua cultura.



- readAlertas: SP que é executado pelo utilizador de modo a que este tenha acesso aos alertas da sua cultura.



### Divergências face ao especificado

A especificação recebida tem uma periodicidade de exportação do mongo para o Sybase de 10s.

Como o sensor a cada 5s envia dados (e se não ocorrer nenhum tipo de erro nem atrasos), com a especificação do outro grupo, a cada 2 mensagens recebidas, estas iam ser enviadas para o Sybase, enquanto que na nossa especificação (20s), a cada 4 mensagens recebidas, estas seriam enviadas.

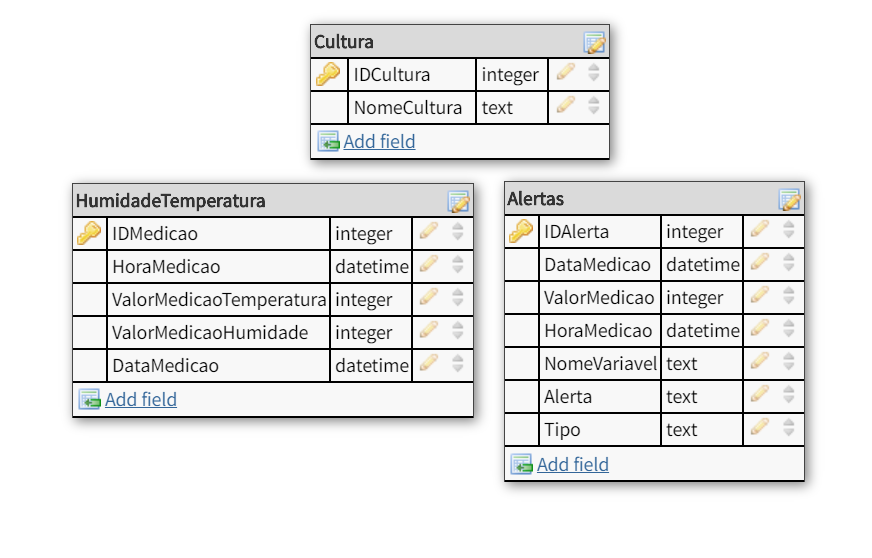
A diferença entre 10s e 20s não é muito relevante, contudo como no trigger que implementámos, comparamos a última mudança (de temperatura/humidade) de cada rajada (conjunto de 4 mensagens, se não houver erros nem atrasos) inserida na tabela, o Sybase fica menos sobrecarregado com a periodicidade de 20s. Isto porque o trigger estaria a comparar a última mudança dessas 2 linhas inseridas, e colocaria esta na tabela Alertas, que irá ser depois lida para ser mostrado o alerta no android. Assim, é melhor comparar a última mudança de 4 linhas inseridas, visto que assim serão inseridas menos mensagens na tabela Alertas, e consequentemente serão mostrados menos alertas no android, mantendo o investigador atualizado.

O nosso grupo especificou que a tabela HumidadeTemperatura tem a permissão de escrita, pelo userJDBC. Decidiu-se alterar este nome, para SGC (como a especificação que recebemos) visto que é o Sistema Gestão Dados que trata da exportação do mongo para o sybase e estávamos a atribuir um nome menos correto.

O utilizador, através dos SPs criados: readCulturas e readAlertas, pode consultar as culturas e alertas respetivos às culturas, que lhe pertencem. Não tem permissão total de leitura, visto que assim iria poder aceder aos alertas de outros investigadores.

# Android e Php

## Esquema da BD Lite Geral



## Layout Implementado no Android

<PrintScreen de um exemplo de interacção>