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 6](#_Toc471205788)

[1.1 Descrição Geral do Procedimento 6](#_Toc471205789)

[1.2 Apreciação Crítica sobre a descrição do processo 7](#_Toc471205790)

[1.3 Exemplo de dados armazenadas na colecção 8](#_Toc471205791)

[1.4 Código Mongo 9](#_Toc471205792)

[1.4.1 Implementado 9](#_Toc471205793)

[1.4.2 Divergências face ao especificado 10](#_Toc471205794)

[1.5 Código SQL 11](#_Toc471205795)

[1.5.1 Implementado 11](#_Toc471205796)

[1.5.2 Divergências face ao especificado 12](#_Toc471205797)

[2 Android e Php 13](#_Toc471205798)

[2.1 Esquema da BD Lite Geral 13](#_Toc471205799)

[2.2 Layout Implementado no Android 14](#_Toc471205800)

[2.3 Layout Implementado em php 15](#_Toc471205801)

[2.4 Questões técnicas de implenmentação 16](#_Toc471205802)

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

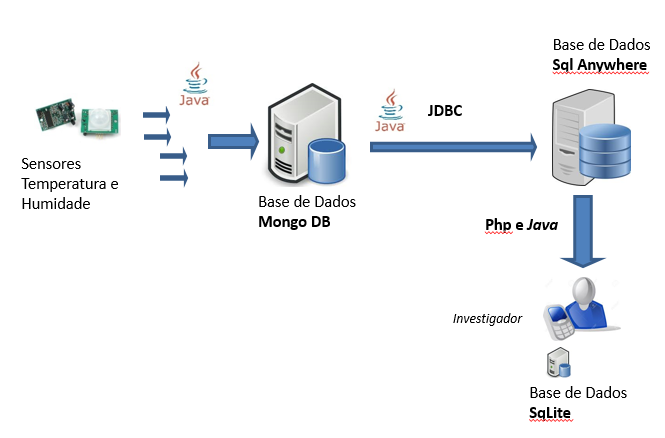


Sensor

*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;
* Nome da coleção do Mongo;
* Nome da DB Sybase;
* Nome da tabela no Sybase.
* Nome de utilizador e password de acesso ao Sybase (JDBC user);
* Nome de utilizador e password de acesso ao 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.

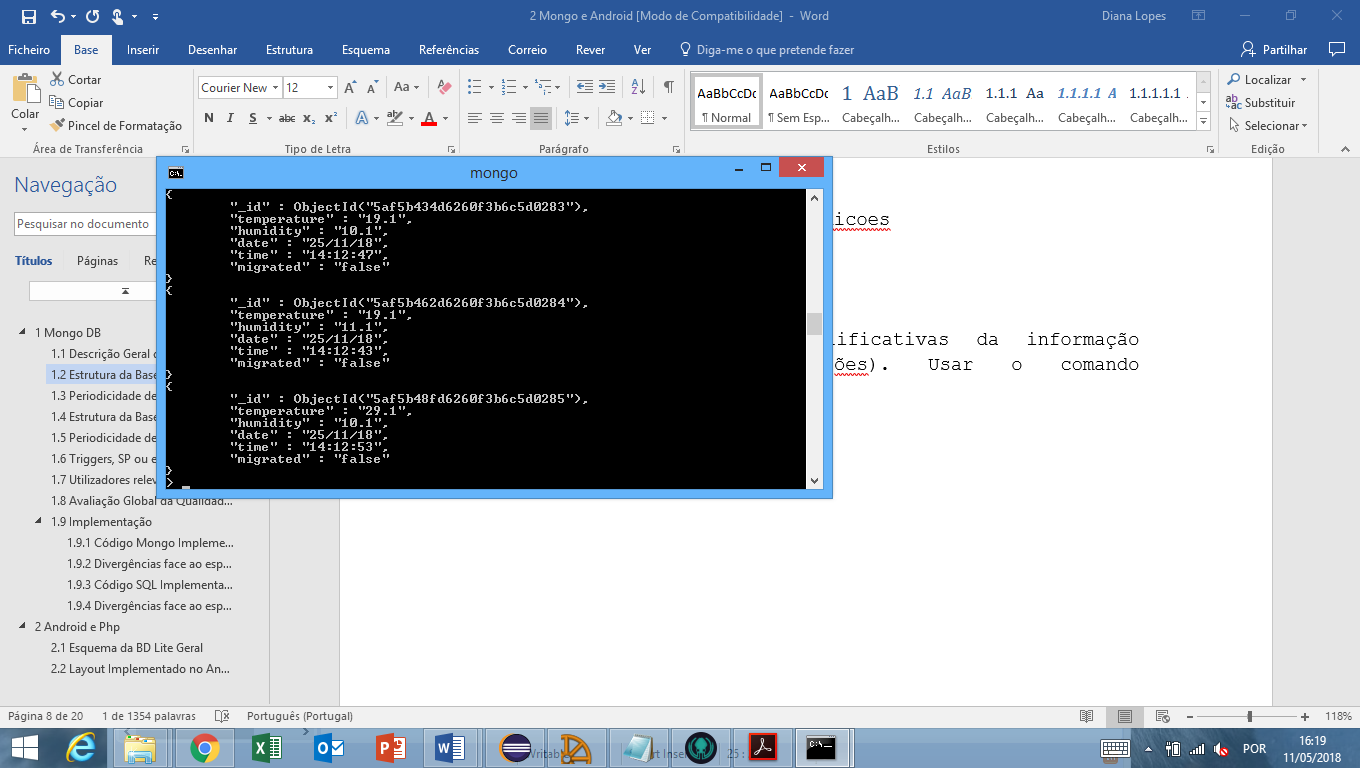


Este ficheiro é lido quando se inicia a aplicação.

## 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 (por exemplo, uma data null) que iriam dar erro no Sybase, fazendo o programa parar.

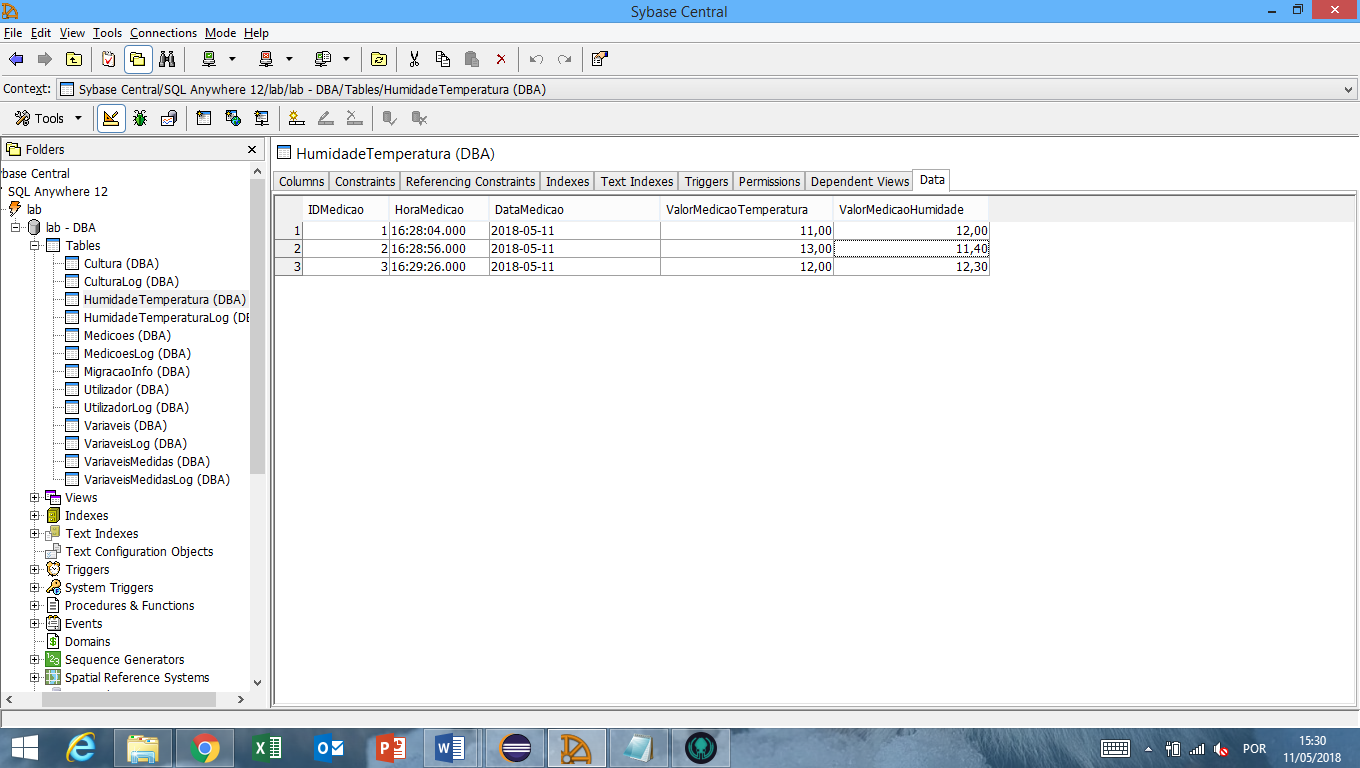
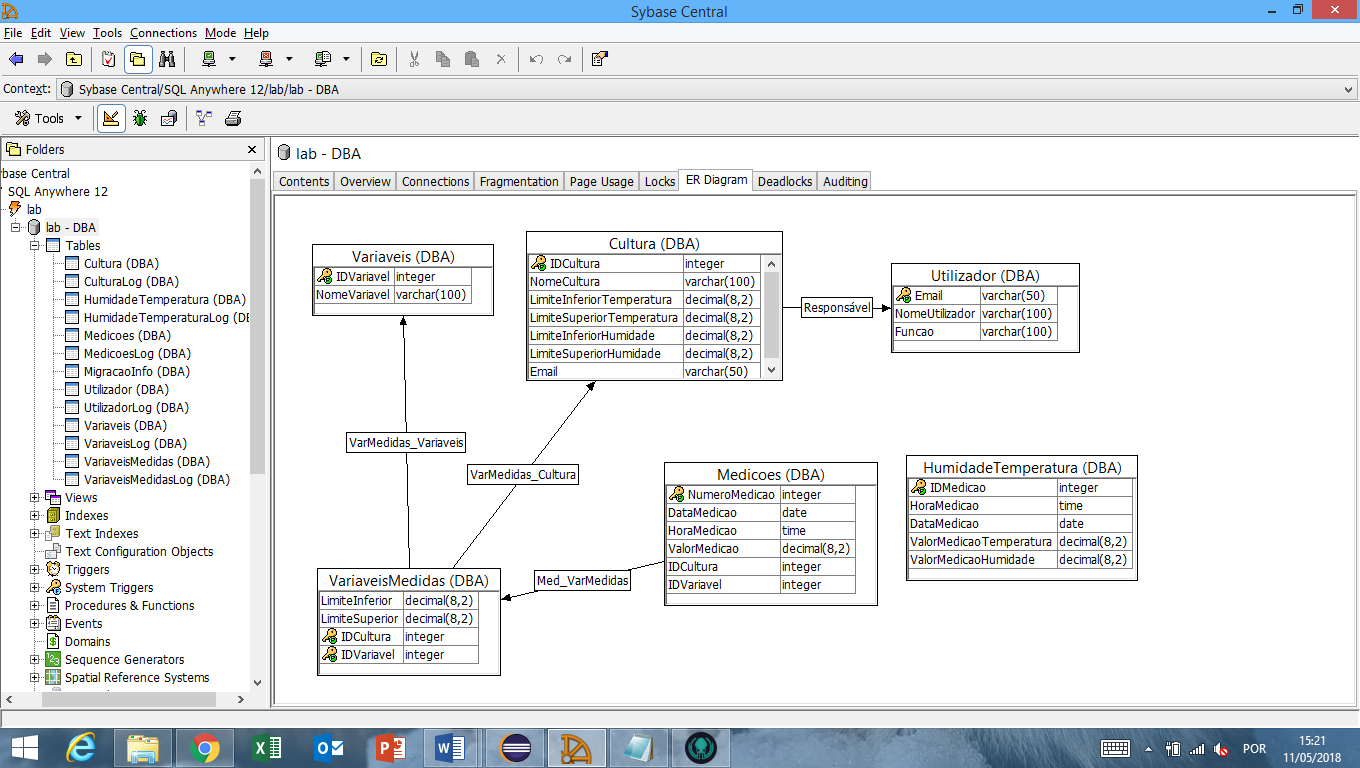
Validações a considerar:

* Analisar a estrutura do objeto JSON para ver se corresponde ao esperado;
* Verificar se cada keyvalue 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);
* 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

<Listar algumas linhas exemplificativas da informação guardada na (s) tabela(s).>



## Periodicidade de Leitura de Mongo e Escrita no Sybase

<Explicar de que forma e com que periodicidade o Java recebe informação do mongo e exporta para o 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.

## 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 |
| Variáveis | - | - |
| Cultura | - | - |
| VariáveisMe-didas | - | - |
| Medições | - | - |
| 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) : \_\_\_\_\_\_\_\_\_\_\_\_\_  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)

<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 >

### 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.>

### 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>

### 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.>

# Android e Php

## Esquema da BD Lite Geral

<Modelo relacional implementado no Android, tabelas e atributos>

## Layout Implementado no Android

<PrintScreen de um exemplo de interacção>