Sistemas de Informação Distribuídos

Licenciaturas em Engenharia Informática e Informática e Gestão de Empresas

2018-2019, Segundo Semestre

Monitorização de Culturas em Laboratório

Mongo DB e Android

Identificação do grupo autor

da especificação (Etapa A):

Identificação do grupo autor da implementação (Etapas B):



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 atualizado;
* Na folha de rosto (anterior) têm de constar toda a informação solicitada, nomeadamente todas as fotografias de todos os elementos dos dois grupos. É obrigatório que caiba tudo numa única página;
* A formatação das “zonas” (umas sombreadas outras não sombreadas) não pode ser alterada;
* O grupo que primeiro edita o documento (Etapa A) apenas escreve até à secção 1.7, e o outro grupo apenas em todas as outras secções.

Índice

[1 Mongo DB 7](#_Toc5657703)

[1.1 Descrição Geral do Procedimento 7](#_Toc5657704)

[1.2 Estrutura da Base de Dados Mongo 12](#_Toc5657705)

[1.3 Periodicidade de Leitura de Sensores e Escrita no Mongo 13](#_Toc5657706)

[1.4 Estrutura da Base de Dados Mysql 14](#_Toc5657707)

[1.5 Periodicidade de Leitura de Mongo e Escrita no MySql 15](#_Toc5657708)

[1.6 Triggers, SP ou eventos no MySql (caso relevante) 16](#_Toc5657709)

[1.7 Utilizadores relevantes no Mysql e respectivos privilégios 17](#_Toc5657710)

[Avaliação Global da Qualidade das Especificações 18](#_Toc5657711)

[1.8 Implementação 19](#_Toc5657712)

[1.8.1 Divergências face ao recebido/especificado 19](#_Toc5657713)

[1.8.2 Código Mongo Implementado (dentro do java) 20](#_Toc5657714)

[1.8.3 Código SQL Implementado 21](#_Toc5657715)

[1.8.4 Tempo Médio 22](#_Toc5657716)

[1.8.5 Alertas 22](#_Toc5657717)

[2 Android e Php 23](#_Toc5657718)

[2.1 Esquema da BD Lite Geral 23](#_Toc5657719)

[2.2 Layout Implementado no Android 24](#_Toc5657720)

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 afeta todas as culturas existentes na estufa).

Periodicamente os investigadores dirigem-se à estufa para efetuarem 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, atualizar e consultar os dados de medições das suas culturas. Esta *proteção de dados* é um aspeto 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 “Registo Temperatura”, “Registo Luz”, “Consulta Informação Android”, “Transporte de Dados de Sensor Para Mongo”, “Exportação de Mongo para MySql” e “Exportação de Dados entre Mysql”.

Diagrama de Classes de Suporte à Base de Dados



Sensor

*Exemplo Mensagens*

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

Esquema de Importação e Migração



# Mongo DB

## Descrição Geral do Procedimento

<Nesta secção deverá ser dada uma descrição genérica sobre a forma como decorre o processo de migração:

1. Receber a informação dos sensores e guardá-lo numa base de dados MongoDB;
2. Exportar de **forma incremental** a informação do MongoDB para a bd Sql Anywhere.

A informação apresentada deverá ser suficiente para que o grupo que a receba consiga implementar as várias etapas. Deve ser clara e estar bem estruturada em secções. Cabe ao grupo decidir qual a melhor forma de estruturar a exposição.

Apesar de não ser para escrever código, se o grupo considerar que o grupo que vai implementar pode desconhecer algum aspecto (biblioteca, algoritmo, etc.) pode exemplificar/ilustrar a forma de implementação. Considerar que o grupo que vai implementar tem conhecimentos razoáveis de Java (POO e PCD) e relacional, tem acesso à documentação colocado no E-Learning sobre MongoDB, e a mais nada.

Alguns exemplos de informação que poderão constar nos requisitos: utilização ou não de threads em Java, número de Mains Java, parâmetros a utilizar no MQTT, quais as leituras que são descartadas.>

1. A interação entre os sensores e a base de dados Mongo deve ter por base uma conexão que use o protocolo MQTT. Desta forma, deve haver uma classe responsável pelo estabelecimento da conexão, que é feito pelo método ‘connect’ da API do MQTT.

Para se poder estabelecer a ligação é necessário definir um cliente, que deve ser uma instância da classe ‘MqttClient’.

Para a criação desta instância, são necessários os parâmetros: identificador do cliente, identificador da máquina hospedeira e respetivo porto (broker) e uma instancia da classe ‘MemoryPersistance’ para indicar se a ligação deve ou não ser memorizada.

Para a conexão, deve ser utilizado o método connect da classe ‘MqttClient’, que poderá receber como argumento uma instância da classe ‘MqttConnectOptions’, onde podem ser definidas mais opções.

Depois da conexão estar estabelecida deve ser utilizado o método ‘messageArrived’ da interface ‘MqttCallback’ que permite receber as mensagens vindas dos sensores (<https://www.eclipse.org/paho/files/javadoc/org/eclipse/paho/client/mqttv3/MqttCallback.html#messageArrived-java.lang.String-org.eclipse.paho.client.mqttv3.MqttMessage->). Este método recebe dois argumentos, uma string relativa ao tópico da mensagem e um objeto do tipo ‘MqttMessage’.

Depois de recebidas as mensagens, as mesmas devem ser tratadas para se extrair os dados necessários e inseri-los na base de dados Mongo. Para isto, é necessário previamente estabelecer a ligação à base de dados NoSQL, sendo por isso necessário criar uma instância da classe ‘MongoClient’. Depois de estabelecida a ligação é necessário colocar os dados sob a forma de documentos e inseri-los na base de dados.

1. Para migrar os dados de forma incremental da base de dados não relacional para a base de dados relacional, deve ser usado um campo ‘exportado’, que é um inteiro e se encontra presente em cada documento da coleção. Este inteiro é criado a 0 por default e sempre que ocorre a exportação é alterado para 1. Desta forma, são exportados apenas dos dados da coleção que tenham esse mesmo campo a 0.

Para realização do processo de escrita de dados na base de dados Mongo deverá ser utilizado apenas um Main.

O main terá necessidade de instanciar classes dos seguintes tipos:

1. Classe que faz a conexão do programa java com a base de dados Mongo.
2. Classe que faz a conexão do java com a base de dados relacional.
3. Classe que faz a conexão com os sensores e que está sempre à espera de receber novas medições para as inserir no MongoDB, pelo que deve ter uma thread.
4. Classe que deve conter uma thread, cuja função é enviar os dados de forma incremental da base de dados Mongo para a base de dos relacional.

Para fazer uma verificação se as medições recebidas se encontram dentro de valores aceitáveis deve ser sempre feita a cada nova medição recebida, uma comparação com a data da anterior medição e com a data do sistema. Assim, deve ser guardada numa estrutura de dados adequada (que sincronize os acessos de escrita e leitura) o valor de todas a medições, sendo que a primeira inserção só vai ser considerada quando vier uma medição cuja data faça sentido com a data do sistema.

Daí em diante, de todas as medições que tenham uma data presumivelmente errada, nenhuma deve ser descartada e deve ser comparado o seu valor com o da medição anterior e com o do sistema afim de perceber qual a data adequada. Os erros podem ser de cada um dos seguintes tipos e devem ser resolvidos da forma que se encontra em seguida:

1. A diferença em módulo da data recebida para a data do sistema é inferior a 10 segundos, bem como a diferença em módulo da data recebida para a data da última medição.

Neste caso, concluímos que a data recebida está correta e usamo-la para inserir na base de dados relacional.

1. A diferença em módulo da data recebida para a data do sistema é inferior a 10 segundos, contudo a diferença em módulo da data recebida para a data da última medição é superior a 10 segundos.

Neste caso, concluímos que o sensor pode ter estado desligado durante algum tempo pelo que a hora recebida está correta e deve ser usada para inserir os dados na base de dados relacional.

1. A diferença em módulo da data recebida na medição para a data do sistema é superior a 10 segundos, contudo a diferença em módulo entre a data recebida e a data da medição anterior é inferior a 10 segundos.

Neste caso concluímos que a data do sistema está errada e usamos a data recebida para inserir na base de dados relacional.

1. A diferença em módulo da data recebida com a data do sistema é superior a 10 segundos bem como a diferença da data recebida para a data da medição anterior.

Neste caso concluímos que tanto a data da medição como a data do sistema podem estar erradas pelo que podemos estar perante um dos seguintes casos:

1. A data do sistema é muito inferior à data da última medição somada de 3 segundos (intervalo entre medições) , pelo que concluímos que a data do sistema está errada e o sensor foi desligado durante algum tempo. Assim deve ser usada a data recebida do sensor.
2. A data do sistema é diferente da data da última medição somada de 3 segundos (intervalo entre medições) , pelo que concluímos que a data da nova medição está errada. Assim deve ser considerada a data da última medição somada de e segundos.
3. A data do sistema é muito superior à data da última medição somada de 3 segundos (intervalo entre medições) , pelo que concluímos que o sensor esteve desligado durante algum tempo e surgiu um erro na medição da data quando voltou a funcionar. Neste caso deve ser usada a data do sistema.

Pode também dar-se o caso do sensor enviar medições erradas de temperatura ou luminosidade que podem desencadear falsos alertas. Assim, deve-se tentar perceber se estamos perante um falso alerta através da seguinte estratégia.

Como referido anteriormente, as últimas medições devem ser guardadas numa estrutura de dados que sincronize os acessos de escrita e de leitura.

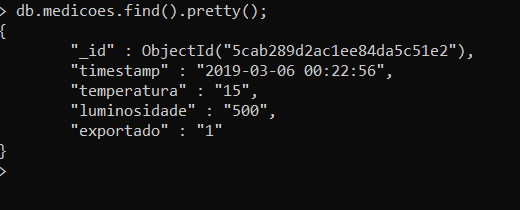
Sempre que uma nova medição chega deve ser inserida na base de dados não relacional e em seguida devem ser comparados os valores das últimas 5 medições com o valor recebido afim de perceber se o valor faz sentido no seguimento dos valores anteriores ou se possivelmente será um erro. Se for um valor muito díspar o mesmo deve ser enviado para a base de dados relacional, contudo não deve ser inserido na tabela de alertas. Se forem recebidos dois ou mais valores seguidos muito diferentes dos anteriores mas que façam sentido entre si então possivelmente estamos perante uma situação de alerta pelo que esses mesmos valores devem ser inseridos na base de dados relacional nas medições respetivas e na tabela de alertas que será explicada nas secções seguintes.

## Estrutura da Base de Dados Mongo

<Nome da base de Dados e das colecções

Listar algumas linhas exemplificativas da informação guardada na (s) colecção(ões). Usar o comando find().pretty() sem critérios>

A base de dados deverá ter o nome ‘sensores’ e deverá ter apenas uma coleção denominada ‘medicoes’ com a seguinte estrutura.



## Periodicidade de Leitura de Sensores e Escrita no Mongo

<Explicar de que forma e com que periodicidade o Java recebe informação dos sensores e exporta para Mongo>

Os sensores enviam novas medições a cada 3 segundos.

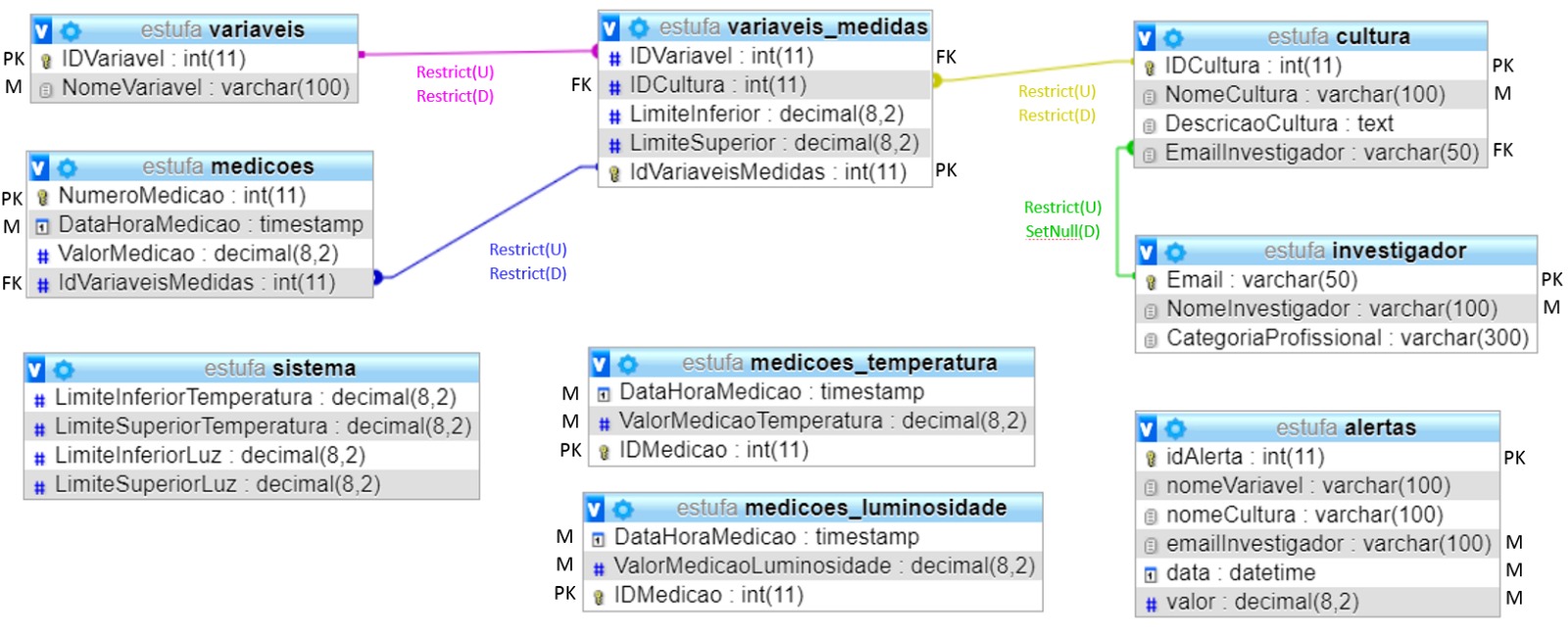
Os mesmos são recebidos através do método ‘messageArrived’ da interface ‘MqttCallback’ que deverá constar numa classe com uma thread dedicada a receber novas medições. Assim que uma nova medição chega ao java deve ser feito o parse da estrutura JSON recebida, e a informação depois de tratada deve ser registada numa estrutura de dados bloqueante que faça a gestão de escritas e leituras em simultâneo. Assim que um anova medição chega, a mesma deve ser registada na base de dado não relacional

## Estrutura da Base de Dados Mysql

<Apenas as tabelas relevantes para esta fase. Utilizar formato de relacional do relatório anterior

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

Sugestão: ter uma tabela para armazenar os alertas que vão ser consultados a partir do android.>



A tabela alertas deve conter a hora e data em que a medição que despoletou o alerta foi feita, o que causou o alerta, ou seja, a variável cujos valores limite foram ultrapassados, o nome da cultura que está associado a essa variável, o email do investigador responsável e o valor que despoletou o alerta.

Um exemplo do que poderia aparecer na tabela alertas é o que se encontra na imagem seguinte.



## Periodicidade de Leitura de Mongo e Escrita no MySql

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

O java deve conectar-se à base de dados mongo a cada X…..

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

<Especificar que triggers ou SP pretendem que sejam implementados (por exemplo, para alertas).>

Deve ser criado um trigger after insert na tabela de medições. O trigger é responsável por verificar sempre que há uma inserção na respetiva tabela, se foram ultrapassados os valores limite da cultura em causa, e caso isso aconteça deve ser registado o respetivo alerta na tabela de alertas.

## Utilizadores relevantes no Mysql e respetivos privilégios

<Utilizar formato de tabela do relatório anterior.>

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabela** | **Tipo de Utilizador** | | |
| Investigador | Administrador Aplicação | Java | | phpUser |
| sistema | L | E,L | - | | - |
| variaveis | E,L | E,L | - | | - |
| investigador | E,L | E,L | - | | - |
| medicoes | E,L | E,L | E | | - |
| variaveis\_medidas | E,L | E,L | - | | - |
| medicao\_luminosidade | L | E,L | E | | - |
| medicao\_temperatura | L | E,L | E | | - |
| alertas | - | - | E | | - |
| cultura | E,L | E,L | - | | - |
| **Stored Proc.** |  |  |  | |  |

O utilizador Java deve ter privilégios para inserir medições nas tabelas medições luminosidade e medições temperatura, bem como inserir alertas na respetiva tabela.

O utilizador phpUser é responsável pelas ligações à base de dados para efetuar migrações e consultas dos alertas para serem enviados em android.

|  |
| --- |
| Avaliação Global da Qualidade das Especificações recebidas  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

### Divergências face ao recebido/especificado

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

Tem de ficar claro:

(i)que ideias aproveitaram da própria especificação;

(ii)que ideias aproveitaram da especificação do outro grupo;

(iii)que ideias novas foram introduzidas.

>

### 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.>

### 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 MySQL, se existirem.>

### Tempo Médio

<Indicar o tempo médio que demora um valor do sensor a chegar a base de dados Mysql.>

### Alertas

<Exemplificar alguns alertas gerados automaticamente.>

# 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>