Sistemas de Informação Distribuídos

Licenciaturas em Engenharia Informática e Informática e Gestão de Empresas

2019-2020, Segundo Semestre

Deteção de Intrusão e Incêndio em Museus

Mongo DB e Android

Identificação do grupo autor da especificação (Etapa A): **20**

|  |  |  |
| --- | --- | --- |
| Número | Nome | Foto |
| 82608 | Francisco Barros |  |
| 20687 | Pedro Santiago | C:\Users\pmsan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D03016C7.tmp |
| 82286 | Bin Guan | C:\Users\pmsan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\948FF36D.tmp |
| 82652 | Alexandre Ferreira | C:\Users\pmsan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\952698FF.tmp |
| 79142 | Tomás Ferreira | C:\Users\pmsan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D9744503.tmp |
| 82453 | José Pedro Cruz Fernandes | C:\Users\pmsan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B4500909.tmp |

Identificação do grupo autor da implementação (Etapas B):

|  |  |  |
| --- | --- | --- |
| Número | Nome | Foto |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Índice

[1 Estrutura da Base de Dados Mongo 6](#_Toc38109446)

[2 Transporte Broker para Mongo (java) 6](#_Toc38109447)

[2.1 Periodicidade de Leitura de Sensores e Escrita no Mongo 6](#_Toc38109448)

[2.2 Utilização de Filtros que impeçam informação de ir para o Mongo 6](#_Toc38109449)

[3 Estrutura da Base de Dados Mysql 7](#_Toc38109450)

[3.1 Diagrama relacional 7](#_Toc38109451)

[3.2 Privilégios de Utilizadores 7](#_Toc38109452)

[4 Transporte Mongo para Mysql (java) 7](#_Toc38109453)

[4.1 Periodicidade de Leitura de Mongo e Escrita no Mysql 7](#_Toc38109454)

[4.2 Utilização de Filtros que impeçam informação de ir para o Mysql 7](#_Toc38109455)

[4.3 Mecanismo para assegurar que a migração é incremental 7](#_Toc38109456)

[5 Geração de Alertas 7](#_Toc38109457)

[5.1 Local/Momento de Deteção 7](#_Toc38109458)

[5.2 Regras para deteção de alertas 8](#_Toc38109459)

[Avaliação Global da Qualidade das Especificações 8](#_Toc38109460)

[6 Implementação 9](#_Toc38109461)

[6.1 Divergências face ao recebido/especificado 9](#_Toc38109462)

[6.1.1 Ideias que aproveitaram do outro grupo 9](#_Toc38109463)

[6.1.2 Ideias que abandonaram do próprio grupo 9](#_Toc38109464)

[6.1.3 Novas ideias introduzidas 9](#_Toc38109465)

[6.1.4 Código Mongo Implementado (dentro do java) 9](#_Toc38109466)

[7 Demonstração 9](#_Toc38109467)

[7.1 Situação 1 10](#_Toc38109468)

[7.1.1 Dados enviados pelo sensor 10](#_Toc38109469)

[7.1.2 Alerta registado na tabela de alertas 10](#_Toc38109470)

[7.1.3 Informação recebida no telemóvel 10](#_Toc38109471)

[7.1.4 Informação Adicional (se relevante) 10](#_Toc38109472)

[7.2 Situação 2 10](#_Toc38109473)

[7.2.1 Dados enviados pelo sensor 10](#_Toc38109474)

[7.2.2 Alerta registado na tabela de alertas 10](#_Toc38109475)

[7.2.3 Informação recebida no telemóvel 10](#_Toc38109476)

[7.2.4 Informação Adicional (se relevante) 10](#_Toc38109477)

[7.3 Situação 3 10](#_Toc38109478)

[7.3.1 Dados enviados pelo sensor 10](#_Toc38109479)

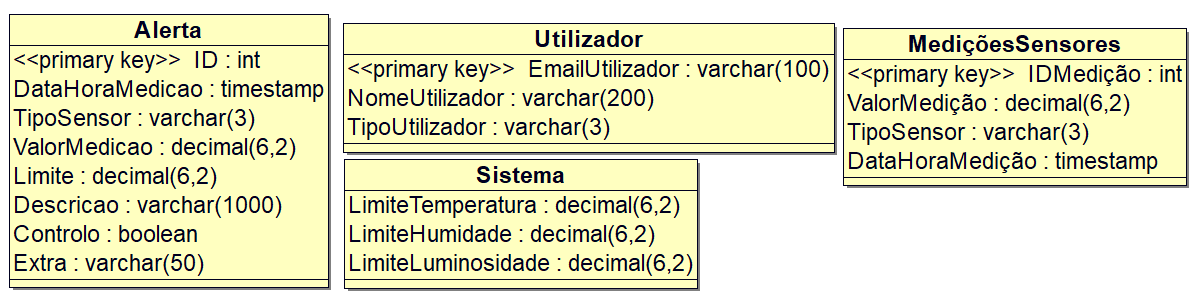
[7.3.2 Alerta registado na tabela de alertas 10](#_Toc38109480)

[7.3.3 Informação recebida no telemóvel 10](#_Toc38109481)

[7.3.4 Informação Adicional (se relevante) 10](#_Toc38109482)

Deteção de Intrusão e Incêndio em Museus

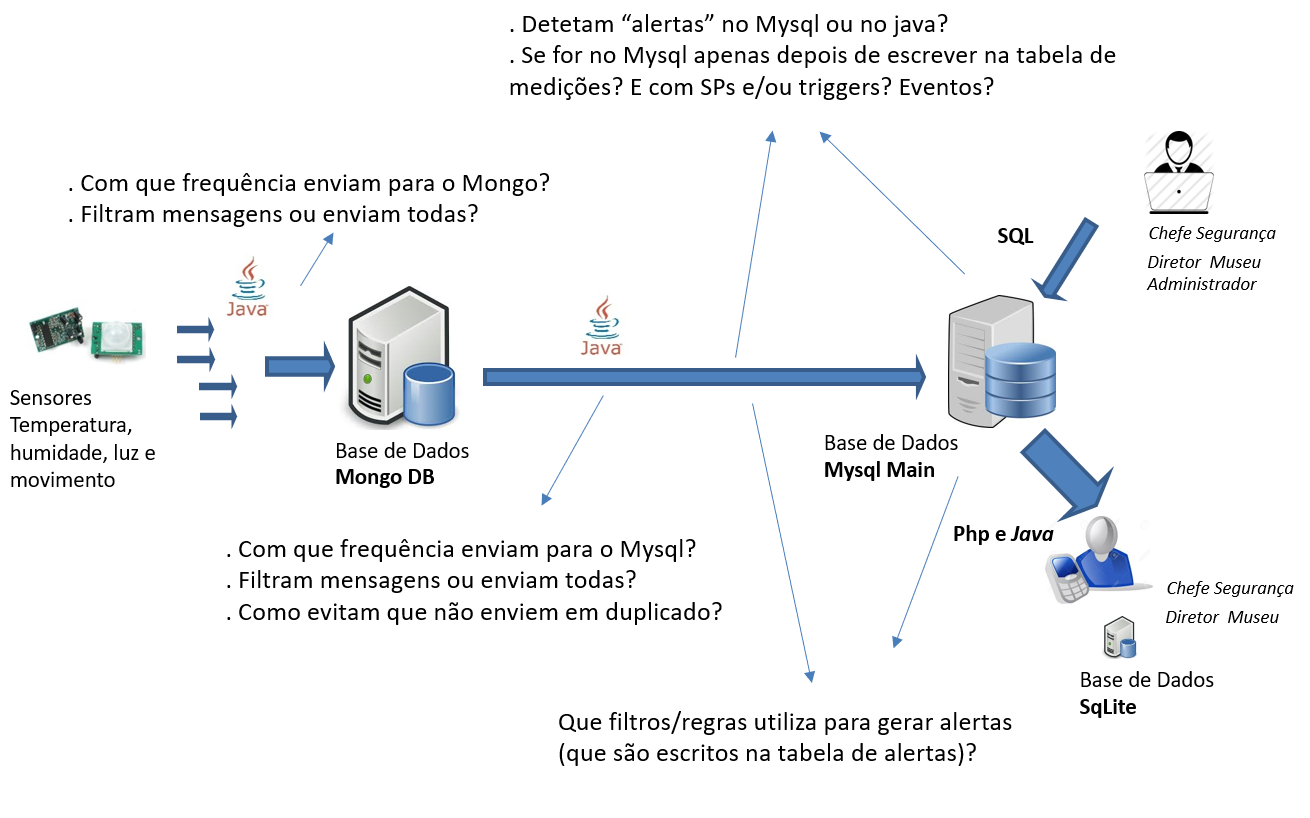
**Base de Dados Mysql**



**Sensor *Exemplo Mensagens***

{"tmp":"19.30","hum":"95.00","dat":"19/4/2020","tim":"9:50:51","cell":"228""mov":"0",”sens":"eth"} (\*)

**Esquema de Importação e Migração**



(\*) Enquanto o sensor não puder ser reparado, a mensagem é:

{"tmp":"19.30","hum":"95.00","dat":"19/4/2020","tim":"9:50:51","cell":"228""mov":"0""mov":"1""sens":"eth"} Para efeitos de trabalho removam **"mov":"0"** deve ser substituído por uma vírgula.

# Base de Dados Mongo

Nome bd: MongoSensores

Nome Coleção1(\*): DadosSensor

Nome Coleção2(\*): \_\_\_\_\_\_\_\_\_\_\_\_\_

Nome Coleção3(\*): \_\_\_\_\_\_\_\_\_\_\_\_\_

Exemplo documento coleção1:

Colecao DadosSensor

{

idMedicao:1

temperatura:23.60

humidade:95.00

luz:388

movimento:0

timestamp:22/4/2020 14:37:24

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Exemplo documento coleção3:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Exemplo documento coleção3:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Número de réplicas: 3

(\*) O número de coleções é definido pelo grupo, pode optar só por uma ou por um qualquer número.

# Transporte Broker para Mongo (java)

## Periodicidade de Leitura de Sensores e Escrita no Mongo

<Explicar com que periodicidade o Java recebe informação dos sensores e envia para Mongo>

O programa, em Java, vai recebendo os dados do broker e inserir na BD Mongo o mais rápido possível (depois de eventuais filtragens). Ou seja, as medições devem ser enviadas à medida que chegam do broker.

## Utilização de Filtros que impeçam informação de ir para o Mongo

<Indicar se pretendem que o java implemente filtros que evitem que algum tipo de dados/situação vá para a base de dados MongoDB>

O programa, em Java, deve verificar a formatação dos dados recebidos, bem como possíveis dados em duplicado.

Tendo em conta a formatação de dados, o programa deve verificar caso falte um dado na medição, se for o caso, deve assumir um valor sem informação (Null). Assim, evita-se que ao falhar um sensor (ex: movimento), não há o comprometimento de outros dados importantes (temperatura,humidade, luz).

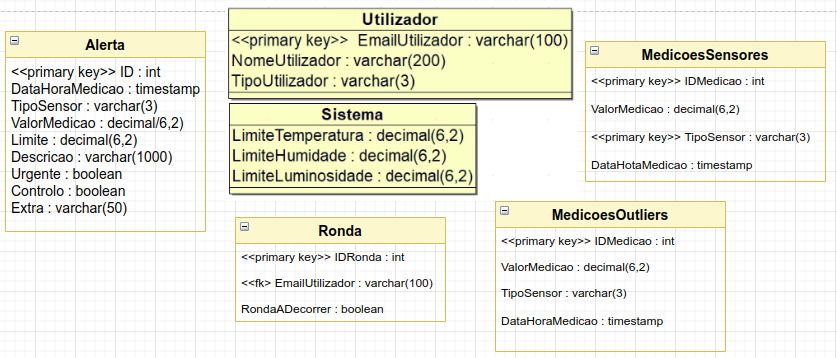
A filtragem de duplicados (medições com o mesmo timestamp) será feita no transporte do broker para a BD Mongo (java), assim, evita-se que existam dados em duplicado por algum possível erro do sensor.

Para fazer a verificação de timestamps, poderá-se utilizar o comando db.DadosSensor.find(<timestamp>).Se for encontrado um dado com o mesmo timestamp do recebido do broker, podemos assumir que há uma duplicação e o duplicado não deve escrito na BD Mongo.

# Estrutura da Base de Dados Mysql

## Diagrama relacional

<Diagrama relacional da base de dados Mysql. Podem efectuar alterações à base de dados apresentada no início do documento, no entanto, apenas os 8 campos da tabela Alerta indicados são reconhecidos pela aplicação Android Disponibilizada (os campos Controlo e Extra foram colocados apenas para antecipar eventuais necessidades dos grupos, não são processados pela aplicação Android.>



Em relação à tabela **MedicoesSensores**, alterámos a primary key para ser o par (IDMedicao, TipoSensor). Fizemos esta alteração para conseguirmos comparar o ID que recebemos do Mongo com os ID's que vão ser inseridos no Mysql. Ou seja, os ID's do Mongo corresponderão aos ID's do Mysql.

Por outras palavras, teremos que para cada informação lida do Mongo serão inseridos 4 registos na tabela MedicoesSensores do Mysql - todas com o mesmo ID mas com TipoSensor diferente.

Exemplo:

Medição no mongo:

{

idMedicao:1

temperatura:23.60

humidade:95.00

luz:388

movimento:0

timestamp:22/4/2020 14:37:24

}

Medição no Mysql (MedicaoSensores):

| IDMedicao  <<pk>> | TipoSensor  <<pk>> | ValorMedicao | DataHoraMedicao |
| --- | --- | --- | --- |
| 1 | Luz | 388 | 22/4/2020 14:37:24 |
| 1 | Mov | 0 | 22/4/2020 14:37:24 |
| 1 | Hum | 95.00 | 22/4/2020 14:37:24 |
| 1 | Tmp | 23.60 | 22/4/2020 14:37:24 |

Adicionámos o campo Urgente à tabela **Alerta** para sabermos se o evento que gera o alerta é apenas um aviso, ou se é um evento preocupante que deve requerer a especial atenção do Chefe do Museu.

Urgente = false : não muito preocupante (não ultrapassa Limites, mas aproxima-se de tal)

Urgente = true : evento preocupante (ultrapassa os Limites)

Adicionámos a tabela **MedicoesOutliers** para receber os valores discrepantes. Ou seja, que se desviam muito do resto dos valores, e que devem ser eventuais erros.

Também decidimos pôr uma tabela **Ronda**, para não gerar alertas devido às medições de movimento e luz durante as rondas que estejam a decorrer.

RondaADecorrer = true : não deve considerar luz e movimento no momento de gerar alertas.

RondaADecorrer = false : deve considerar todas as medições no momento de gerar alertas.

## Privilégios de Utilizadores

<Quadro com tipos de utilizadores e indicação privilégios sobre tabelas e SPs. Apenas utilizadores relevantes para esta fase >

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Tipo de Utilizador** | | | |
| **Administrador** | **Director de Museu** | **Chefe de Segurança** |  |
| **MediçõesSensores** | **L** | **L** | **L** |  |
| **Utilizador** | **REL** | **REL** | **L** |  |
| **Alerta** | **L** | **L** | **L** |  |
| **Sistema** | **L** | **L** | **REL** |  |
| **Ronda** | **L** | **L** | **REL** |  |
| **MedicoesOutliers** | **L** | **L** | **L** |  |
| **Stored Proc.** |  |  |  |  |
| **AtivarRonda** | **-** | **-** | **X** |  |
| **ControlarAlerta** | **X** | **X** | **X** |  |

Consideramos que o segurança não é importante para esta fase do trabalho.

Em relação à tabela **Utilizador**, tanto o diretor do museu, como o administrador, poderão fazer alterações nesta tabela. Já o chefe de segurança, apenas poderá consultar a mesma.

No que diz respeito à tabela **Sistema**, consideramos que apenas o chefe de segurança pode alterar estes valores pois será ele que está presente no museu e, como tal, deverá ser o utilizador ideal para ajustar valores de limites de medições (ex: poderá detetar uma maior humidade natural, e, para evitar falsos alarmes, aumenta o valor do limite).

Limitámos a tabela **Alerta** para apenas se poderem fazer leituras sobre a mesma (para qualquer tipo de utilizador). Assim, limitamos ao máximo o potencial erro humano (inserções ou remoções acidentais) - mais do que esta permissão, poderia pôr em causa o sistema de alerta. A mesma lógica se aplica às tabelas **MedicoesSensores e MedicoesOutliers**, que apenas deverão poder ser consultadas.

Tendo em conta o SP **AtivarRonda**, terá o propósito de mudar o estado de RondaADecorrer para true (se for começar a ronda) ou false (se estiver a acabar a ronda).

Também criámos um SP **ControlarAlerta** por forma a poder mudar o estado de Controlo para true (se a situação estiver a ser resolvida). Por defeito, Controlo vai estar a false até alguém ativar o SP.

# Transporte Mongo para Mysql (java)

## Periodicidade de Leitura de Mongo e Escrita no Mysql

<Explicar com que periodicidade o Java recebe transporta informação para o Mysql>

Na nossa opinião, devemos escrever os dados no Mysql logo a seguir a recebê-los do Mongo.

A nossa justificação é que, para além de penalizar o menos possível o tempo de reação do sistema de alarme, também o sistema de migração fica mais impermeável a falhas, porque vai escrevendo a informação no Mysql à medida que a recebe do Mongo. Ex: Caso haja uma falha de energia, temos a certeza que as medições vão sendo escritas no Mysql até ao momento da falha.

## Utilização de Filtros que impeçam informação de ir para o Mysql

<Indicar se pretendem que o java implemente filtros que evitem que algum tipo de dados/situação vá para a base de dados Mysql (tabela Medições)>

Achamos que não é necessário impedir informação de ir para o Mysql, apenas filtrar em que tabela é escrita a informação (duplicação e má formatação de mensagens já são filtrados no transporte Sensor(broker) -> Mongo).

No caso dos outliers e dos alertas, temos que garantir que são escritos na tabela MedicoesOutliers e Alerta, respectivamente. Para filtrar essa informação, consideramos que guardar as últimas 9 medições para poder comparar com a última medição recebida é uma amostra suficiente para uma boa análise.

Caso haja uma medição alarmante, depois de ser verificada que não é outlier, deve ser escrita em MedicoesSensores, bem como Alerta.

Se houver um outlier impede-se essa medição de ser escrita na tabela MedicoesSensores, mas guarda-se esse valor na tabela MedicoesOutliers para futura análise (possivel avaria do sensor). Consideram-se outliers as medições que tenham um desvio grande em relação às anteriores medições, ou, que tenham um valor mal formatado (Null).

Exemplo de cálculo para deteção de outlier:

Depois de alguma procura, descobrimos um método estatistíco que é eficaz na deteção de um valor outlier num conjunto de dados. (Pode ser consultado aqui: <http://www.mathwords.com/o/outlier.htm>)

Considerando que:

[27] -> medição Temperatura recebida do Mongo

[31,35,40,41,42,43,44,45,48] -> últimas 9 medições (ordem crescente)

Primeiro, temos que calcular:

minimo = 31

primeiro\_quartil = 37 i.e. (31+35+40+41)/4

mediana = 42

terceiro\_quartil = 45 i.e. (43+44+45+48)/4

maximo = 48

IQR = 45 - 37 = 8 i.e (1ºquartil - 3ºquartil)

IQR \* 1.5 = 8 \* 1.5 = 12

Agora, podemos definir o intervalo de valores "aceitáveis", ou seja, valores dentro deste intervalo nunca são considerados outliers.

intervalo\_inferior = 37 - 12 = 25 i.e 1ºQuartil - IQR\*1.5

intervalo\_superior = 45 + 12 = 57 i.e 3ºQuartil + IQR\*1.5

Como o valor da medição recebida [27] está incluído no intervalo [25,57], podemos concluir que não é um outlier.

Este método pode ser usado para Temperatura, Luz e Humidade. Para Movimento não faz sentido fazê-lo pois só assume 2 valores (0 ou 1) e assim, deve-se assumir que medições relativas a Movimento são sempre medições correctas, excepto se for mal formatado (Null).

## Mecanismo para assegurar que a migração é incremental

<Indicar de que forma garantem que não vai informação duplicada para a a base de dados Mysql (tabela Medições)>

Comparamos se para cada ID do Mongo, existem 4 registos correspondentes (com o mesmo ID) no Mysql. Se não existirem os 4 na tabela MedicoesSensores, então, verifica-se também na tabela MedicoesOutliers. Se não houverem 4 no total, podemos concluir que houve problemas com a migração.

# Geração de Alertas

## Local/Momento de Deteção

Assinalar com X as situações pretendidas (podem ser várias)

No transporte Java \_X\_

No Mysql antes de inserir na tabela de medições \_

De que forma: SP \_ Trigger: \_\_

No Mysql depois de inserir na tabela de medições \_

De que forma: SP \_ Trigger: \_\_

Recorrer a eventos: \_\_\_

Vantagens de cada momento de deteção:

**Transporte Java:**

* **[+]** Mais rápido de todos a poder actuar - será o primeiro a receber as medições da BD Mongo (pode inserir diretamente na tabela Alerta, para além de MedicoesSensores)
* **[-]** Teria que ler primeiro a tabela LimiteMedicao para tirar conclusões sobre o alerta.

**Mysql antes de inserir:**

* **[+]** Mais rápido a detetar do que após a inserção, mais fácil do que no Transporte Java (+ fácil verificar o limite na tabela LimiteMedicao, sendo que já é feito dentro da BD Mysql).
* **[-]** Não é tão rápido a detetar o alerta como no transporte Java - terá que esperar que o java envie a informação para a BD Mysql.

**Mysql depois de inserir:**

* **[+]** Mais fácil do que no transporte Java a verificar os limites de medição.
* **[-]** Mais lento a detetar do que antes da inserção - as medições têm que ser inseridas primeiro na MedicoesSensores, e só depois poderá ser analisado o possível alerta.

## Regras para deteção de alertas

<Explicar de forma muito clara que regras devem ser implementadas, que tipos de alerta elas despoletam, de que forma os alertas são guardados na tabela alertas. **Obrigatório dar exemplos**, quer de dados que dão origem aos alertas, quer dos registos que são escritos na tabela alertas. De preferência recorrer a diagramas/ilustrações. Devem indicar explicitamente o nome dos sps que pretendem e/ou o nome das funções java.>

Tipos de alerta:

* **Aviso\_Tmp** : Urgente=false, Descriçao="Temperatura a aproximar-se do limite"
* **Aviso\_Hum** : Urgente=false, Descriçao="Humidade a aproximar-se do limite"
* **Aviso\_Luz** : Urgente=false, Descriçao="Luz a aproximar-se do limite"
* **Alerta\_Tmp** : Urgente=true, Descriçao="Temperatura passou do limite"
* **Alerta\_Hum** : Urgente=true, Descriçao="Humidade passou do limite"
* **Alerta\_Luz** : Urgente=true, Descriçao="Luz passou do limite"
* **Alerta\_Mov** : Urgente=true, Descriçao="Foi detetado movimento irregular"

Exemplo de registo de um **alerta** (Tabela Alerta):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | DataHoraMed | TipoSensor | ValorMed | Limite | Descrição | Urgente | Controlo | Extra |
| 1 | 22/4/2020 14:37:24 | Tmp | 51.00 | 50.00 | Temperatura passou do limite | **true** | false |  |

Se o valor da medição atingir, ou ultrapassar, o valor do Limite, irá despoletar um alerta. Aplicável para Humidade, Luz e Temperatura.

No caso de Movimento, se a medição assumir o valor 1, será prontamente registado como um alerta - não havendo qualquer aviso antes.

Exemplo de registo de um **aviso** (Tabela Alerta):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | DataHoraMed | TipoSensor | ValorMed | Limite | Descrição | Urgente | Controlo | Extra |
| 1 | 22/4/2020 15:37:24 | Tmp | 40.00 | 50.00 | Temperaturaproxima-se do limite | **false** | false | 1,2,3,4,5,6,7,8,9, 10 |

Se os valores de temperatura das últimas 5 medições forem todos maiores ou iguais que 80% do valor limite de temperatura, então é emitido um aviso. Se for emitido um aviso, só poderá ser emitido outro após 5 mensagens.

Exemplo:

LT = 50 (Limite Temperatura)

80%LT = 40 (80% do limite de temperatura)

Últimos 10 valor medicao Temperatura = **[**30,31,35,40,41,42,42,43,43,43]

Segundo o critério acima descrito, neste caso, para estas dez mensagens, irá ser emitido um aviso após a oitava, uma vez que as cinco anteriores são iguais ou maiores que 40. Este critério verifica-se também após a nona e décima mensagem, mas não deve haver logo novo aviso, uma vez que um já foi enviado poucos segundos antes.

|  |
| --- |
| Avaliação Global da Qualidade das Especificações recebidas  Avaliação (A,B,C,D,E) : \_\_\_\_C+\_\_\_\_\_  Utilize a seguinte escala:  A: - 1 – 5 valores B: 6 – 9 valores C: 10 – 13 Valores D: 14 – 17 valores E: 18 – 20 valores  **Análise crítica (clareza, completude, rigor):**  Uma coleção MongoDB não é suficiente uma vez que se pode utilizar coleções para cada tipo de informação para a gestão ser mais fácil. Os objetos da coleção DadosSensor não funcionam uma vez que um objeto contém os dados de temperatura, humidade, luz e movimento e assim está se a presumir que todas essas informações estão a ser enviadas simultaneamente ao mesmo segundo. Se acontecer algum erro com os sensores, o valor null que é enviado depois terá de ser tratado e é uma complicação que seria resolvida caso se utilizasse várias coleções.  O SP AtivarRonda não devia existir. Era melhor ter um Trigger “CheckRonda” AFTER\_INSERT na tabela MedicoesSensores pois, após os registos serem adicionados, é mais fácil ter um trigger gerir alertas quando uma ronda está ativa ou não. No SP, basta o chefe de segurança esquecer de ativar a ronda e o sistema manda alertas sem necessidade. O SP ControlarAlerta é inútil, porque só mostra informação adicional.  Nos privilégios dos utilizadores não faz muito sentido o chefe de segurança ser o único a poder alterar as tabelas de Sistema uma vez que o administrador ou o Diretor de museu possam querer alterar o valor do limite ou até mesmo por fins de administração. Não são especificados os privilégios das aplicações java nas tabelas.  A deteção de outlier é uma ideia engraçada mas muito complexo.  Técnica de filtragem pouco eficaz (exige comparação de muitos registos). Filtragem e migração incremental não impede que haja registos inseridos de datas futuras (ao presente) e datas passadas (relativas a ultima data inserida no sql) |

# 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.>

### Ideias que aproveitaram do outro grupo

<Descrição clara, utilizar bullets.>

### Ideias que abandonaram do próprio grupo

<Descrição clara, utilizar bullets.>

- Leitura periódica do broker, passamos a receber todas as mensagens em tempo real.

-

### Novas ideias introduzidas

<Descrição clara, utilizar bullets.>

- Por sugestão da docente introduzimos um valor mínimo para a temperatura e humidade.

-

### Código Mongo Implementado (dentro do java)

<Listar todo o código Mongo utilizado dentro do java, quer para importar, quer para exportar. O código tem de ser comentado para que se torne legível para quem sabe uns rudimentos de MongoDB. Não é para colocar código java (excepto se for necessário para perceber o código mongo.>

# Demonstração

<Ilustrar o funcionamento da aplicação preferencialmente com capturas de imagens. Os exemplos têm de ilustrar todo o tipo de alertas, com a indicação dos respectivos tempos e da forma como visualmente a informação chega ao telemóvel.>

## Situação 1

Tipo de Alerta: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Dados enviados pelo sensor

### Alerta registado na tabela de alertas

<indicar também o tempo entre o envio do primeiro registo do sensor e o registo na tabela de alertas>

### Informação recebida no telemóvel

### Informação Adicional (se relevante)

## Situação 2

Tipo de Alerta: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Dados enviados pelo sensor

### Alerta registado na tabela de alertas

<indicar também o tempo entre o envio do primeiro registo do sensor e o registo na tabela de alertas>

### Informação recebida no telemóvel

### Informação Adicional (se relevante)

## Situação 3

Tipo de Alerta: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Dados enviados pelo sensor

### Alerta registado na tabela de alertas

<indicar também o tempo entre o envio do primeiro registo do sensor e o registo na tabela de alertas>

### Informação recebida no telemóvel

### Informação Adicional (se relevante)

…