

Sistemas de Informação Distribuídos (20/21)

Licenciaturas em Engenharia Informática e Informática e Gestão de Empresas

Monitorização de Culturas em Laboratório

Identificação do grupo : __ 20 __

Número	Nome	Foto
88110	Beatriz de Sousa Lima	
83687	Inês Leónidas Xavier Esteves Ferreira da Rocha	
88706	Maria Inês Fragoso Rebelo Penha Monteiro	
87993	Mauro Filipe Sousa Afonso	
87475	Sofia de Almeida Correia	
87998	Tiago Veríssimo Bombas	
Especificação: <input checked="" type="checkbox"/> Direta <input type="checkbox"/> MQTT		

Identificação do grupo que completou documento: _____

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 instruções (delimitadas por <>) têm de ser removidas no documento final;
- A paginação tem de ser sequencial e não ter falhas;
- 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;
- A informação apresentada deverá ser suficiente para que o grupo que a receba consiga implementar o que pretendem. 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 aspeto (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), tem acesso à documentação colocado no E-Learning sobre MongoDB, e a mais nada.
- Às vezes os diagramas são uma boa forma de completar a especificação.

Índice

3	Especificação entregue ao outro grupo	4
3.1	Transporte de MongoDB para Mysql	4
3.1.1	Coleções em cada uma das réplicas do Mongo	4
3.1.2	Descrição Geral do Procedimento	5
3.1.3	Migração por MQTT (apenas preencher se for a forma de migração atribuída ao grupo)	6
3.1.4	Tratamento de medições pré-Mysql	6
3.1.5	Tratamento de medições pós-Mysql	9
3.1.5.1	Especificação de Triggers (caso existam)	9
3.1.5.2	Especificação de Procedimentos e Funções (caso existam)	9
3.1.6	Utilizadores Base de Dados Mysql	9
3.1.6.1	Especificação de Procedimentos	10
3.1.7	Eventos de suporte à aplicação	12
3.1.8	Layout dos formulários HTML	12
3.1.9	Associar SP a formulários HTML	18
3.2	Avaliação Global de especificações entregues outro grupo	19
4	Alteração de Especificação recebida de migração do outro grupo	20
5	Comparação de Implementações de Migrações Mongo/Mysql	244
6	Implementação de Alertas, Utilizadores, Php	277
6.1	Alterações das especificações próprias implementadas face a sugestões de outro grupo	277
6.2	Alterações das especificações próprias implementadas de iniciativa própria	277
6.3	Coleções em cada uma das réplicas do Mongo	277
6.4	Base de Dados Mysql	288
6.5	Tratamento de medições pré-Mysql	299
6.6	Tratamento de medições pós-Mysql	32
6.6.1	Especificação de Triggers implementados	32
6.6.2	Código de Triggers implementados	32
6.6.3	Especificação de todos Procedimentos e Funções implementados	33
6.6.4	Stored Procedures e Funções implementadas	355
6.7	Utilizadores Base de Dados Mysql	388
6.8	Eventos de suporte à aplicação	39
6.9	PrintScreen dos formulários HTML	39
6.10	PrintScreen dos formulários Android	43

Monitorização de Culturas em Laboratório

3 Especificação entregue ao outro grupo

3.1 Transporte de MongoDB para Mysql

3.1.1 Coleções em cada uma das réplicas do Mongo

<pre>db.sensorh1.find().pretty() { "_id" : ObjectId("6036c771967bf6108c5b7ca9"), "Zona" : "Z1", "Sensor" : "H1", "Data" : "2021-02-24 at 21:38:56 GMT", "Medicao" : "24.61639494871795"} { "_id" : ObjectId("6036c771967bf6108c5b7caa"), "Zona" : "Z1", "Sensor" : "H1", "Data" : "2021-02-24 at 21:38:57 GMT", "Medicao" : "15.595232230769232"}</pre>	<pre>db.sensorh2.find().pretty() { "_id" : ObjectId("6036c75c967bf61560213019"), "Zona" : "Z2", "Sensor" : "H2", "Data" : "2021-02-24 at 21:38:36 GMT", "Medicao" : "5.0"} { "_id" : ObjectId("6036c75c967bf6156021301a"), "Zona" : "Z2", "Sensor" : "H2", "Data" : "2021-02-24 at 21:38:36 GMT", "Medicao" : "5.0"}</pre>
<pre>db.sensorl1.find().pretty() { "_id" : ObjectId("6036c77f967bf612b4486142"), "Zona" : "Z1", "Sensor" : "L1", "Data" : "2021-02-24 at 21:39:11 GMT", "Medicao" : "318.27"} { "_id" : ObjectId("6036c780967bf612b4486143"), "Zona" : "Z1", "Sensor" : "L1", "Data" : "2021-02-24 at 21:39:11 GMT", "Medicao" : "327.81809999999996"}</pre>	<pre>db.sensorl2.find().pretty() { "_id" : ObjectId("6036c78c967bf61b7c33dd27"), "Zona" : "Z2", "Sensor" : "L2", "Data" : "2021-02-24 at 21:39:24 GMT", "Medicao" : "1.0256410256410255"} { "_id" : ObjectId("6036c78d967bf61b7c33dd28"), "Zona" : "Z2", "Sensor" : "L2", "Data" : "2021-02-24 at 21:39:24 GMT", "Medicao" : "1.282051282051282"}</pre>
<pre>db.sensort1.find().pretty() { "_id" : ObjectId("603be729967bf6135c29555c"), "Zona" : "Z1", "Sensor" : "T1", "Data" : "2021-02-28T18:55:37Z", "Medicao" : "7.160838461538458"} { "_id" : ObjectId("603be729967bf6135c29555d"), "Zona" : "Z1", "Sensor" : "T1", "Data" : "2021-02-28T18:55:37Z", "Medicao" : "6.296503076923074"}</pre>	<pre>db.sensort2.find().pretty() { "_id" : ObjectId("6036c7a5967bf62d2c982203"), "Zona" : "Z2", "Sensor" : "T2", "Data" : "2021-02-24 at 21:39:48 GMT", "Medicao" : "20.6"} { "_id" : ObjectId("6036c7a5967bf62d2c982204"), "Zona" : "Z2", "Sensor" : "T2", "Data" : "2021-02-24 at 21:39:49 GMT", "Medicao" : "21.218"}</pre>

3.1.2 Descrição Geral do Procedimento

No processo de migração, o nosso grupo utilizou duas bibliotecas, `mongodb-driver-3.4.3.jar`, utilizada para a ligação da nossa base de dados local à base de dados mongo na cloud, e `mysql-connector-java-8.0.15.jar` que possibilita a inserção de dados no Mysql, e criou as seguintes classes: `Medicao`, `MongoConnect` e `MysqlConnect`.

Numa primeira fase, através da classe `MongoConnect` é estabelecida uma ligação entre a MongoDB local e a MongoDB da Cloud.

Após o estabelecimento desta ligação, é necessário recolher os dados existentes em cada uma das MongoDB (Local e Cloud). E, para tal, acedemos a cada uma das coleções das bases de dados através de dois objetos do tipo `MongoCollection<TDocument>` (um para cada `MongoDatabase`).

Os dados são enviados em tempo real para a nossa base de dados (BD) local.

De modo a garantirmos que não são enviados dados repetidos, sempre que chegam novos dados vindos da Cloud, verificamos através de uma função `db.collection.find()` se estes já existem na nossa BD local. Em caso afirmativo, descartamos os dados, caso contrário iremos inseri-los. Se por algum motivo a transferência dos dados for abaixo, e assumindo que a DB da Cloud se mantém inalterável, isto é, os dados que já descarregamos até ao momento não sofreram alterações, é possível recolher os dados a partir do ponto onde ficamos, através do método `skip(int count)`. Neste método, o `count` corresponde ao número de documentos já descarregados para a nossa MongoDB local.

Após termos inserido devidamente os dados no MongoDB local, iremos então filtrá-los, de modo a isolar as componentes cruciais para posteriormente serem colocadas as medições no MySQL. Este tratamento de dados pré-sql é feito através do método `getDataToInsert` da classe `Medicao`. Após esta função ser invocada, existe, na mesma classe, a função `insertNewMedicao`, que devolve uma String (“Query”) que irá permitir a inserção da nova medição no MySQL.

Para finalizar o processo de migração dos dados, na classe `MysqlConnect`, o método `analyzeData` percorre uma `MongoCollection<TDocument>` que acede aos dados de cada coleção da nossa MongoDB local e, para cada um destes, é criada uma nova medição. Medição esta que com recurso a uma `LinkedList` será inserida numa lista de medições que será submetida às deteções dos quatro tipos de alertas identificados pelo grupo.

Concluído todo o processo em cima descrito, os dados serão finalmente inseridos no MySQL, através de transporte direto.

Optámos por ter um único main, que inicia e conclui todo o processo. Este inicia duas threads: 1) `MongoConnect`, responsável pela migração dos dados desde a MongoDB da nuvem até à MongoDB local e 2) `MysqlConnect`, responsável pelo tratamento de dados existentes na MongoDB local, para que estes possam ser corretamente inseridos no MySQL. Esta última thread encarrega-se também de assegurar que todas as medições são devidamente avaliadas, de modo a garantir que as mesmas não contêm valores anómalos para as culturas.

Caso seja necessário um *reset* do processo de migração de dados, existe o método `resetMongo()` que reinicia a thread `MongoConnect`.

3.1.3 Migração por MQTT (apenas preencher se for a forma de migração atribuída ao grupo)

3.1.4 Tratamento de medições pré-Mysql

Na classe *Medicao* dispomos de dois métodos. O primeiro, *getDataToInsert()*, recebe como argumento uma String *s*. Esta String será o resultado de cada um dos objetos da respetiva coleção existente na BD local.

Exemplo: String *s* = "Document{{_id=604f7f79967bf60db42f88c0, Zona=Z1, Sensor=T1, Data=2021-03-15T15:38:33Z, Medicao=13.666666666666664}}".

Recorrendo às funções *split()* e *replace()*, foi possível isolar num vetor (String [] *data*) os dados que pretendemos, obtendo o seguinte resultado:

```
data[0]=604f7f79967bf60db42f88c0
data[1]=Z1
data[2]=T1
data[3]=2021-03-15 15:38:33
data[4]=13.666666666666664
```

Após invocarmos estas duas últimas funções, colocamos os atributos (*idMedicao*, *idZona*, *idSensor*, *data* e *medicao*) com os respetivos valores.

Depois desta atribuição, a função *insertNewMedicao()* devolve, no formato String, a Query que servirá para inserirmos no MySQL os dados relativos a cada medição. No nosso exemplo, esta função retornará o seguinte: "INSERT INTO `medicoes` (`IDMedicao`, `IDSensor`, `Data`, `Medicao`) VALUES ('604f7f79967bf60db42f88c0', 'T1', '2021-03-15 15:38:33', '13.666666666666664')". Tal como explicado no modelo relacional, um sensor já tem uma zona associada, logo o atributo *IDZona* é omitido.

A thread *MysqlConnect* está encarregue de realizar todo este tratamento e filtragem de dados, para que seja possível inseri-los corretamente no MySQL.

Alertas

Através de uma "MongoCollection<TDocument>" acedemos aos dados de cada coleção da nossa MongoDB local, para cada um destes é criado um objeto medição que através do método *getDataToInsert(String s)* da classe "Medicao" atribui valores aos seus atributos: "idMedicao", "idZona", "idSensor", "data" e "medicao". Através da função *getMedicao()*, obtemos o valor associado ao objeto "Medicao" que é automaticamente inserido numa *LinkedList*.

Após cada inserção na *LinkedList* é criada uma thread "UpdateZona", que recebe como parâmetros o *IDZona* e um objeto do tipo "Connection". Esta thread é responsável por atualizar os valores da temperatura, humidade e luz para cada zona, através do método *updateMedicao()*, no qual é criada uma query com as medições atualizadas e, posteriormente, enviada para o MySQL.

Esta *LinkedList* é submetida a uma análise de deteção dos quatro tipos de alertas. Para facilitar a análise, foram criadas várias threads e uma classe "Alerta", com os atributos "IDAlerta", "IDSensor", "Data", "Medicao", "NivelAlerta", "Cultura", "Mensagem", "IDUtilizador", "IDCultura" e "HoraEscrita". Em cada thread, com recurso ao método *executeQuery(String s)*, são extraídos os valores que posteriormente serão colocados nos atributos, à exceção da mensagem e do nível de alerta, que serão apenas definidos após a verificação das condições impostas pelas threads.

Esta análise começa pela *thread* “LerMedicao”, que extrai cada medição inserida e verifica se esta se encontra dentro dos limites definidos para a zona correspondente.

Caso não esteja dentro dos limites, é invocada a *thread* “AlertaUm”, na qual é verificada, após 60 medições, se estas se encontram acima/abaixo dos limites da zona. Caso isso se verifique, então é criado um objeto da classe Alerta (o Alerta 1), com o atributo nível de alerta e a mensagem. Através da função `insertNewAlerta()`, é criada uma *query* que será dada como parâmetro no método `executeUpdate(String query)`, responsável por enviar o alerta para o MySQL.

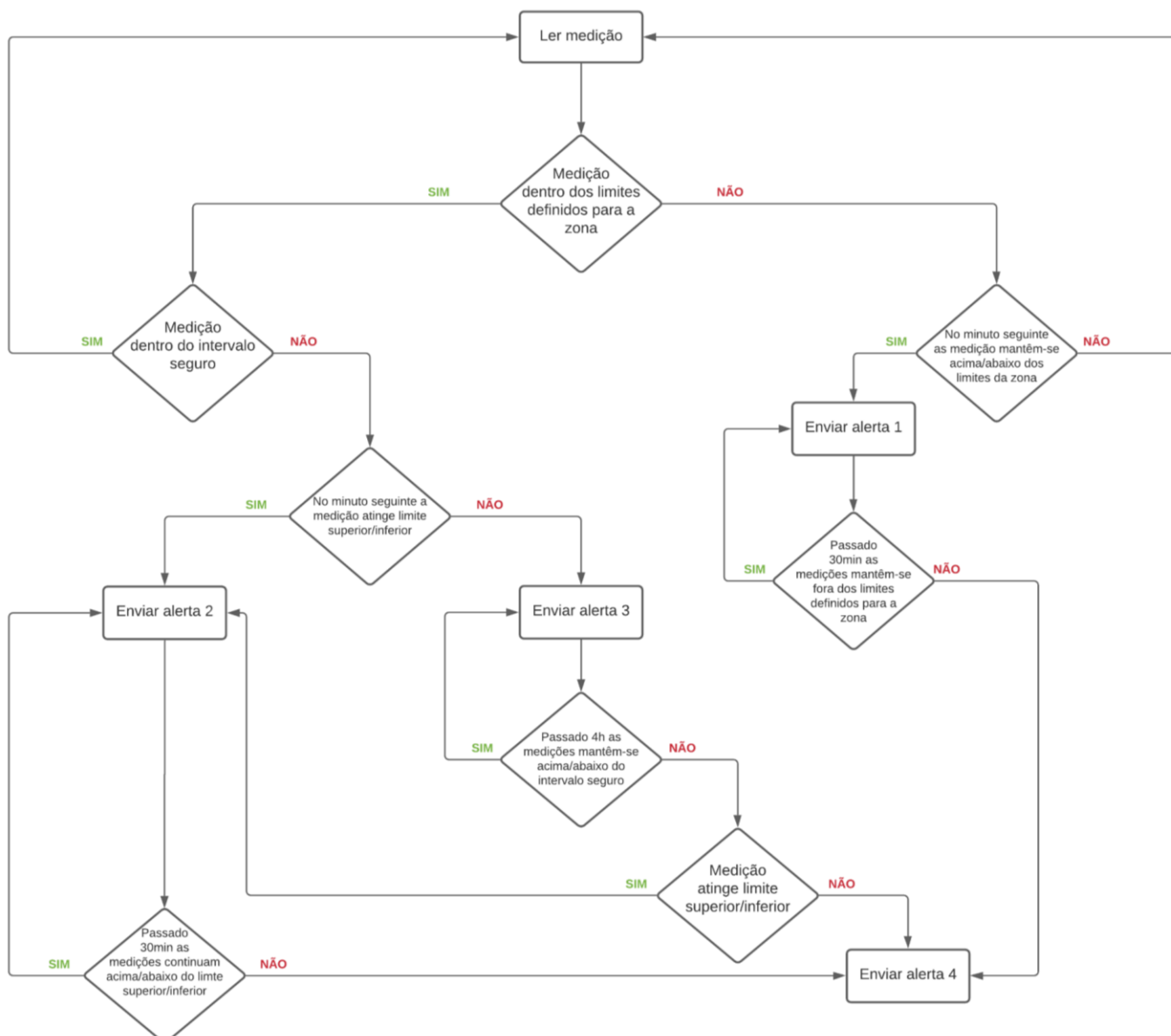
Após ser enviado o alerta acima, é criada uma *thread* “TimerAlertaUm” que, passados 30 minutos, irá verificar se as medições que estão a ser recebidas na *LinkedList* se mantêm ou não fora dos limites definidos para a zona. Caso isso se verifique, então é criado um objeto Alerta (o Alerta 1), que será enviado novamente para o MySQL. Após este envio, é criada, de novo, uma *thread* “TimerAlertaUm”. Caso contrário, é gerado um objeto Alerta (o Alerta 4), com o atribuído nível de alerta e a mensagem, que será encaminhado para o MySQL.

Caso a medição se encontre dentro dos limites definidos para a zona, é invocada a *thread* “AlertaTres” que, através do método `executeQuery(String query)`, extrai as temperaturas máxima e mínima para cada cultura e, com base nesses dados, calcula o seu intervalo seguro, comparando-o com a medição recebida na *LinkedList*. Se a medição não estiver dentro do intervalo seguro da cultura, é verificado, após 60 medições, se estas atingem ou não limite superior/inferior da cultura. Em caso afirmativo, é criada uma *thread* “AlertaDois”, que irá criar um objeto da classe Alerta (o Alerta 2) com o atribuído nível de alerta e a mensagem, e envia este alerta para o MySQL.

Após ser enviado o alerta, é criada uma *thread* “TimerAlertaDois” que, passados 30 minutos, irá verificar se as medições que estão a ser recebidas na *LinkedList* se mantêm ou não fora dos limites definidos para a cultura. Caso isso não se verifique, então é gerado um objeto Alerta (o Alerta 4) com o atribuído nível de alerta e a mensagem, que será encaminhado para o MySQL. Por outro lado, caso se confirme que a medição se encontra fora dos limites, é criado um objeto Alerta (Alerta 2) que será enviado novamente para o MySQL. Após o envio, é gerada uma nova *thread* “TimerAlertaDois”.

Se após 60 medições, se confirma que estas não atingem o limite superior/inferior da cultura, então é criado um objeto da classe Alerta (o Alerta 3) com o atribuído nível de alerta e a mensagem, encaminhando de seguida o alerta para o MySQL. Após o envio do alerta 3, é criada uma *thread* “TimerAlertaTres” que, passadas 4 horas, verifica, através de uma condição *if*, se a medição contínua acima/abaixo do intervalo seguro. Se a condição se verificar, será criada novamente uma *thread* “TimerAlertaTres”, caso contrário é gerado um objeto Alerta (o Alerta 4) com o atribuído nível de alerta e a mensagem.

Caso exista uma falha na migração de dados dos sensores, é criada uma *String query* com a descrição “Falha na migração de dados! Reinicie a aplicação” e, através do método `executeUpdate(String query)`, é encaminhada para o MySQL.



Legenda

- Intervalo seguro: [limite inferior + 10% do número de valores entre o limite mínimo e máximo, limite superior - 10% do número de valores entre o limite mínimo e máximo]. Exemplo: A cultura A aguenta temperaturas entre os 3°C e os 30°C. Assim o seu intervalo seguro será $[3+(10\%*(30-3)), 30-(10\%*(30-3))]$ = [5.7, 27.3]
- Alerta 1 (muito urgente)**
 “O sensor t da zona 1 apresenta valores superiores/inferiores aos definidos para a zona.”
 É enviado se, após 60 medições, a medição continuar acima/abaixo dos valores definidos para a zona.
 É reenviado se, após 30 minutos, a medição continuar fora dos limites definidos para a zona.

- **Alerta 2 (muito urgente)**
 “A temperatura da cultura x ultrapassou a temperatura limite.”
 É enviado se a medição atingir um valor acima/abaixo do limite superior/inferior da cultura. É reenviado de 30 em 30 minutos se a medição continuar acima/abaixo do limite superior/inferior da cultura.
- **Alerta 3 (urgente)**
 “A temperatura da cultura x está a aproximar-se da temperatura limite.”
 É enviado se, após 60 medições, a medição continuar acima/abaixo do intervalo seguro. É reenviado de 4 em 4 horas se a medição continuar acima/abaixo do intervalo seguro.
- **Alerta 4 (pouco urgente)**
 “A temperatura da cultura x encontra-se dentro dos limites definidos.”
 É enviado após 30 minutos dos alertas 1 e 2, e 4 horas do alerta 3 se a medição voltar a ficar dentro dos limites definidos.

3.1.5 Tratamento de medições pós-Mysql

3.1.5.1 Especificação de Triggers (caso existam)

3.1.5.2 Especificação de Procedimentos e Funções (caso existam)

3.1.6 Utilizadores Base de Dados Mysql

Tabela	Tipo de Utilizador	
	Investigador	Administrador
alertas	L	L
medicoes	L	L
utilizador	-	E
sensor	-	E
zona	-	L
cultura	E	E
Stored Procedures		
ListarInvestigadores	-	X
ListarCulturasAdmin	-	X
VisualizarCulturaAdmin	-	X
ListarAlertasCultura	X	X
ListarNotificacoesAdmin	-	X
ListarCulturasZona	X	-
VisualizarCulturaInv	X	-
ListarNotificacoesInv	X	-
CriarUtilizador	-	X
RemoverUtilizador	-	X
CriarCultura	-	X
RemoverCultura	-	X
EditarCultura	-	X
EditarParametrosCultura	X	-

3.1.6.1 Especificação de Procedimentos

Nome Procedimento / Função	Parâmetros Entrada	Parâmetros Saída	Valores Retorno	Muito breve descrição
CriarUtilizador	IDUtilizador Nome EmailUtilizador TipoUtilizador Password	Utilizador	-	Permite ao administrador criar um utilizador, atribuindo-lhe um ID, nome, email, tipo e password
RemoverUtilizador	IDUtilizador	-	-	Permite ao administrador remover um utilizador e todas as culturas associadas, se for investigador
ListarInvestigadores	-	Tabela com os nomes dos investigadores	-	Permite ao administrador visualizar todos os investigadores existentes.
CriarCultura	IDCultura NomeCultura IDUtilizador Estado IDZona TemperaturaMin TemperaturaMax HumidadeMin HumidadeMax LuzMin LuzMax	Cultura	-	Permite ao administrador criar uma cultura, atribuindo-lhe os parâmetros que a definem, podendo ou não lhe ser atribuída um investigador responsável.
RemoverCultura	IDCultura	-	-	Permite ao administrador remover uma cultura.
EditarCultura	IDCultura IDUtilizador	-	-	Permite ao administrador editar uma cultura, podendo alterar ou atribuir o investigador responsável pela mesma.
ListarCulturasAdmin	-	Tabelas com todas as culturas	-	Permite ao administrador visualizar todas as culturas do laboratório.


VisualizarCulturaAdmin	IDCultura	IDCultura NomeCultura IDUtilizador Estado IDZona TemperaturaMin TemperaturaMax HumidadeMin HumidadeMax LuzMin LuzMax	-	Permite ao administrador visualizar todos os parâmetros associados à cultura selecionada e respetivas medições atuais.
ListarAlertasCultura	IDCultura	Tabela com todos os alertas da cultura selecionada, com respetiva hora, nível de alerta e mensagem	-	Permite ao administrador/ investigador visualizar todos os alertas da cultura selecionada.
ListarNotificacoesAdmin	-	Tabela com todos os alertas com respetiva hora, nível de alerta, nome da cultura e mensagem	-	Permite ao administrador visualizar todas as notificações de todas as culturas.
ListarCulturasZona	IDZona	Tabela com todas as culturas da zona selecionada e tabela com as medições atuais	-	Permite ao investigador visualizar todas as culturas da zona selecionada e respetivas medições atuais, também representadas graficamente.
VisualizarCulturaInv	IDCultura	IDCultura NomeCultura Estado IDZona TemperaturaMin TemperaturaMax HumidadeMin HumidadeMax LuzMin LuzMax e tabela com as medições atuais da zona em que a cultura se encontra	-	Permite ao investigador visualizar todos os parâmetros associados à cultura selecionada e respetivas medições atuais.

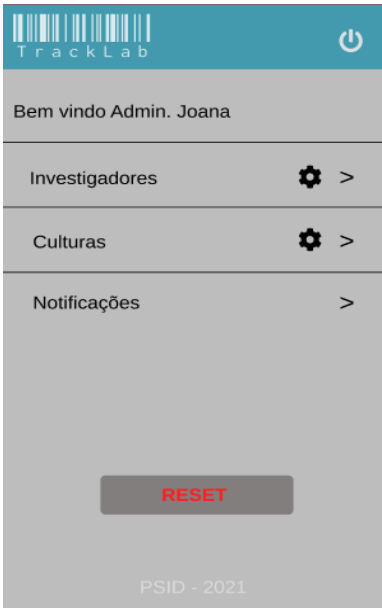

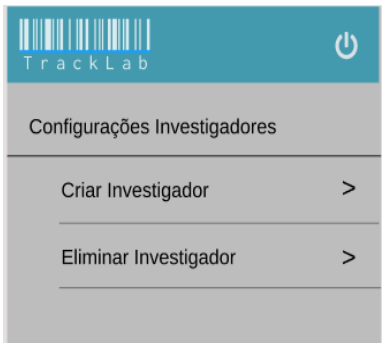



EditarParametrosCultura	IDCultura NomeCultura TemperaturaMin TemperaturaMax HumidadeMin HumidadeMax LuzMin LuzMax	-	-	Permite ao investigador editar os parâmetros da cultura.
ListarNotificacoesInv	IDUtilizador	Tabela com todos os alertas do investigador com respetiva hora, nível de alerta, nome da cultura e mensagem	-	Permite ao investigador visualizar todos os alertas de todas as suas culturas.

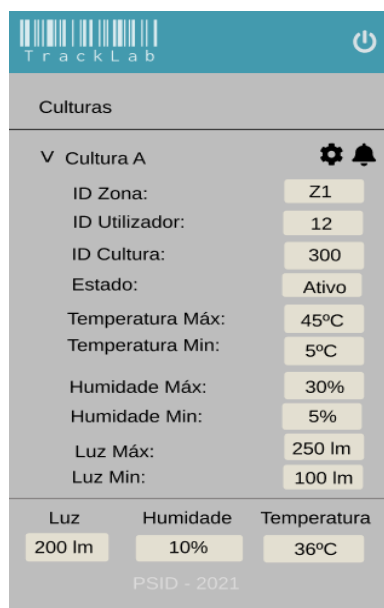
3.1.7 Eventos de suporte à aplicação

Nome Evento	Local Execução (MySQL/Windows)	Muito breve descrição
Alerta 1	Windows	O Utilizador recebe uma notificação (aviso), consoante o alerta gerado.
Alerta 2	Windows	
Alerta 3	Windows	
Alerta 4	Windows	
RESET	Windows	Em caso de falha das migrações das leituras dos sensores do MongoDB para o Mysql, o administrador da aplicação pode reinicializar o processo, após receber o aviso.

3.1.8 Layout dos formulários HTML


TRACKLAB	
Login	
	<p>O utilizador faz o login com as credenciais que lhe foram atribuídas.</p> <p>Se for o administrador da aplicação, será direcionado para aplicação como tal (1).</p> <p>Caso seja investigador, será direcionado para a aplicação como tal (2).</p> <p>No caso de o investigador não se lembrar da sua password, terá que enviar um e-mail ao administrador da app, a requisitar a reposição da mesma.</p>


Administrador (1)	
	<p><u>Investigadores</u> Ao clicar neste botão, o administrador terá acesso à lista de investigadores existentes. Pode, ainda, configurar os investigadores.</p> <p><u>Culturas</u> Ao clicar neste botão, o administrador terá acesso à lista de culturas existentes. Pode, ainda, configurar as culturas.</p> <p><u>Notificações</u> Ao clicar neste botão, o administrador terá acesso ao seu histórico de avisos/notificações.</p> <p><u>RESET</u> Em caso de falha das migrações das leituras dos sensores MongoDB para MySQL, o administrador recebe um aviso de falha da migração de dados, podendo reinicializar o processo através do botão RESET, que invoca o método resetMongo().</p> <p>Ainda é possível, através do botão de log-out , no canto superior direito da imagem, fazer o log-out da aplicação.</p>
	<p>Na secção Investigadores, ao clicar no botão , o administrador pode aceder às configurações:</p> <ul style="list-style-type: none"> • Criar um investigador novo; • Eliminar um investigador existente;
	<p>Na secção Culturas, ao clicar no botão , o administrador pode aceder às configurações:</p> <ul style="list-style-type: none"> • Criar uma nova cultura; • Apagar uma cultura existente; • Editar uma cultura, podendo exclusivamente alterar o investigador responsável pela mesma.



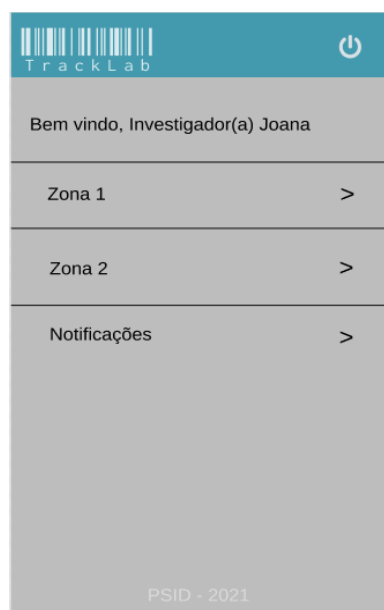
Aquando da visualização da lista de culturas existentes, o administrador pode selecionar uma dada cultura (ex.: cultura A).

Quando é selecionada a cultura, o administrador pode visualizar os seus dados: ID da zona a que pertence, ID do utilizador responsável (caso exista um), ID dessa mesma cultura, estado (ativo/inativo) e os valores máximos e mínimos de temperatura, humidade e luz (parâmetros introduzidos pelo investigador). Também é possível visualizar os valores atuais de luz, humidade e temperatura, no canto inferior do ecrã.

Ainda, dentro de uma determinada cultura, é possível eliminá-la e/ou atribuí-la a um Investigador, através do botão de configurações .


O administrador consegue também aceder ao histórico de notificações (avisos) da cultura selecionada, através do botão de notificações .

Investigador (2)



Após ter efetuado o login com sucesso, o investigador consegue visualizar uma mensagem de boas-vindas com o seu nome.

É visível um menu no qual o investigador poderá selecionar uma das seguintes opções: a zona a que deseja aceder (Zona 1 ou Zona 2) e as "Notificações", onde irá aparecer uma lista com o histórico de notificações (avisos) recebidas.

Ainda é possível, através do botão de log-out , no canto superior direito da imagem, fazer o log-out da aplicação.

	<p>Ao selecionar a Zona 1, irão aparecer as medições, no momento (atual), das variáveis luz, humidade e temperatura. O investigador pode selecionar a variável (luz, humidade ou temperatura) sobre a qual pretende visualizar o gráfico com as medições.</p> <p>Por exemplo, ao clicar em cima do nome “<u>Temperatura</u>” (), consegue visualizar o gráfico com as respetivas medições.</p> <p>É também possível visualizar uma lista com as culturas pertencentes à Zona 1 e selecionar, caso queira ver em detalhe, cada uma delas.</p> <p>O <i>refresh</i> da página é automático.</p>
	<p>No botão “Filtros”  , aparecerá uma janela pop-up, na qual o investigador poderá selecionar qual o intervalo de tempo através do qual deseja visualizar o gráfico da temperatura, luz, humidade.</p>
	<p>Ao selecionar uma determinada cultura, irá aparecer o nome da cultura, o ID e o respetivo estado.</p> <p>Aparecem também os parâmetros definidos para esta cultura, que podem ser editados pelo investigador ao carregar no botão  .</p> <p>O investigador pode também visualizar as notificações relativas a esta cultura em particular, carregando no botão “Notificações”  .</p> <p>No canto inferior do ecrã, estão visíveis os valores atuais (no momento) das variáveis luz, humidade e temperatura da zona na qual a cultura se encontra inserida.</p>

Para que o investigador possa visualizar as medições de cada cultura ao longo do tempo, será apresentado um gráfico por medição, temperatura, humidade ou luz, onde o eixo horizontal representa o tempo e o eixo vertical a medição.

Inicialmente o eixo do tempo estará dividido em horas, sendo possível visualizar as medições registadas nas passadas 24 horas. Assim sendo, o eixo horizontal será variável, de acordo com a hora a que o investigador entra na aplicação para ver o gráfico.

O investigador terá também, a possibilidade de alterar a escala do eixo horizontal através dos filtros:

- de 20 em 20 minutos, nas passadas 8h (24 medições);
- de 10 em 10 minutos, nas passadas 4h (24 medições);
- de 1 em 1 segundo, nos passados 24s (24 medições);

O eixo da medição, terá como valor mínimo/máximo, o valor mínimo/máximo suportado pelo sensor de acordo com a zona onde está instalado. Tendo em conta as medições mínimas/máximas suportadas por cada cultura, será também indicado no gráfico, através duma área sombreada, estes valores, para facilitar a compreensão dos dados aos investigadores.

Exemplo:

Se o investigador vê o gráfico às 14h então:

- no eixo horizontal, estarão visíveis medições desde as 15h do dia anterior até à hora atual
- no eixo vertical, o limite mínimo/máximo, será o valor mínimo/máximo suportado pelo sensor

O investigador quer ver os gráficos, das passadas 24h, 8h, 4h e 24s, relativos à cultura A, que sobrevive em temperaturas entre os 3°C e os 30°C, e se encontra na zona 1, que tem como temperaturas limite os 2°C e os 50°C. Tendo em conta que, no intervalo de tempo selecionado, a temperatura mínima registada na cultura foi de 5°C e a temperatura máxima registada foi de 33°C, os gráficos teriam o seguinte aspeto:

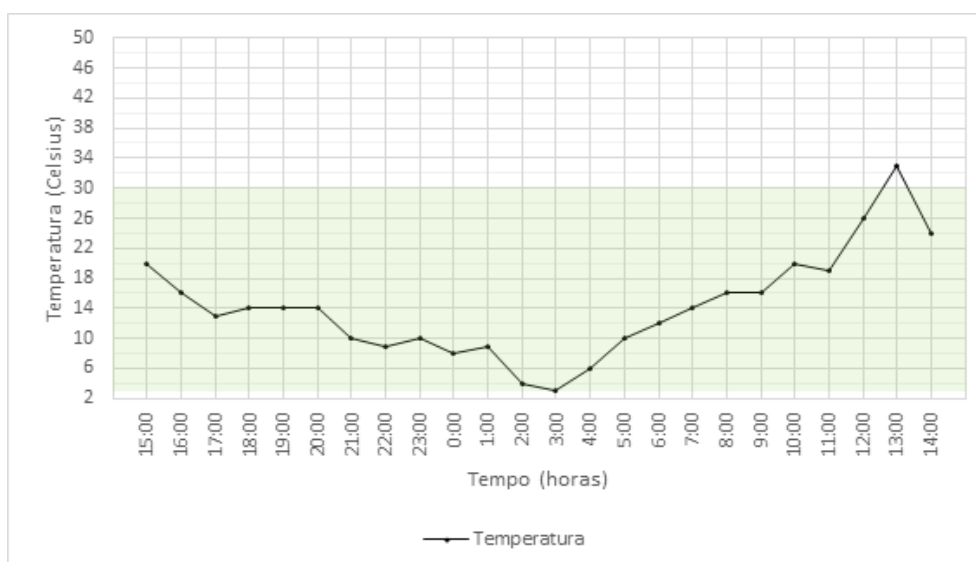


Gráfico 1: 24 em 24 horas

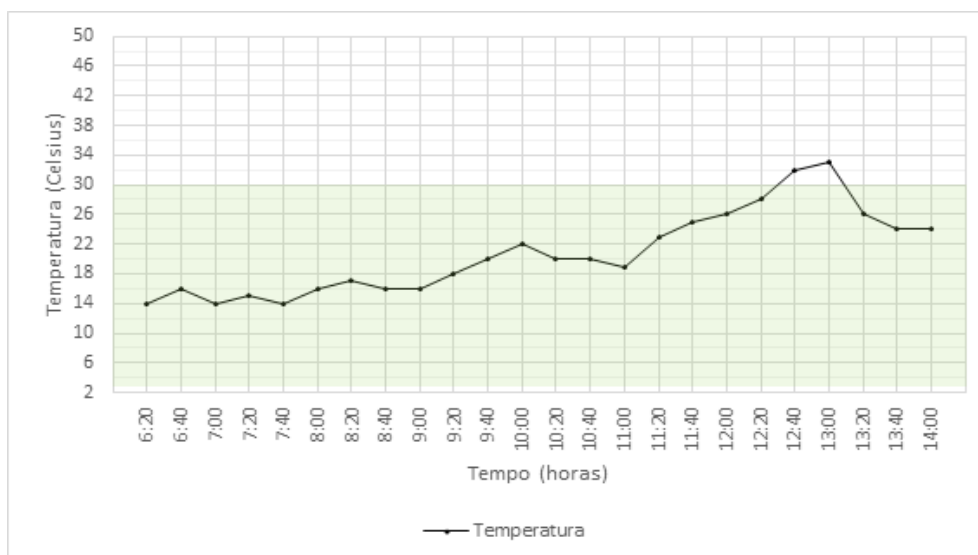


Gráfico 2: 20 em 20 minutos

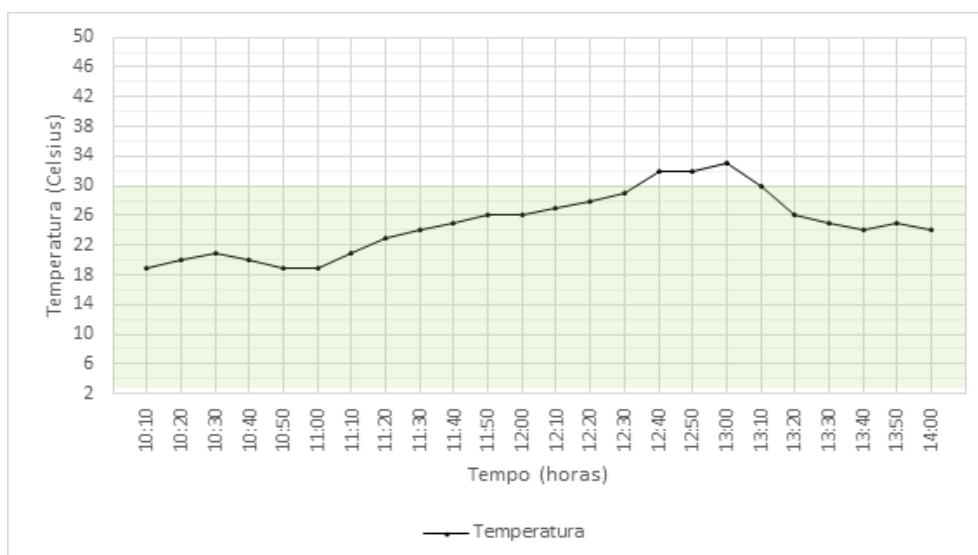


Gráfico 3: 10 em 10 minutos

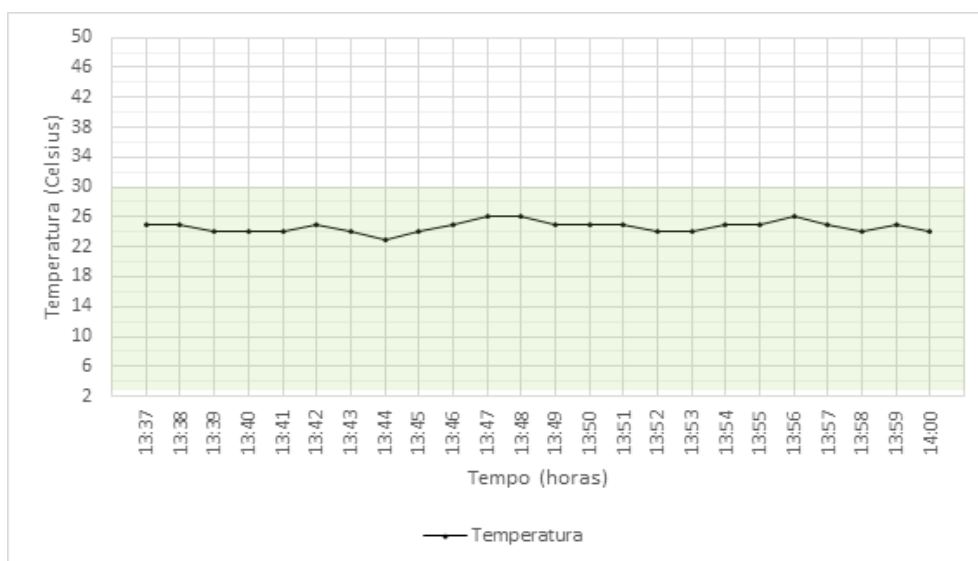


Gráfico 4: 1 em 1 segundo

3.1.9 Associar SP a formulários HTML

Procedimento	Botão
Administrador	
ListarInvestigadores	> Investigadores 
ListarCulturasAdmin	v Culturas 
CriarUtilizador	Criar Investigador >
RemoverUtilizador	Eliminar Investigador >
CriarCultura	Criar Cultura >
RemoverCultura	Apagar Cultura >
EditarCultura	Editar Cultura >
VisualizarCulturaAdmin	Cultura A >
ListarAlertasCultura	
ListarNotificacoesAdmin	Notificações >
Investigador	
ListarCulturasZona	Zona 1 >
ListarNotificacoesInv	Notificações >
VisualizarCulturaInv	Cultura A >
EditarParametrosCultura	
ListarAlertasCultura	

3.2 Avaliação Global de especificações entregues outro grupo

Avaliação Global da Qualidade das especificações que entregaram ao outro grupo

Quanto às especificações, o conteúdo é claro, está bastante extenso, mas explícito.

As tabelas de sql estão bem detalhadas com toda a informação e a classe alerta está muito pormenorizada com os diferentes tipos de alertas.

Existem bastantes procedimentos e o html está limpo e compreensível.

Avaliação (E,D,C,B,A) : A

Utilize a seguinte escala:

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

Três principais deficiências de especificação que tiveram impacto mais negativo na qualidade da implementação

Tratamento de erros na medição, como se deve tratar desse erro.

Tratamento de dados duplicados quando se recebe várias medições.

A não utilização de uma tabela descrita no enunciado parametrocultura com as respetivas chaves primárias e estrangeiras.

Resumo de Avaliações de Qualidade Anteriores (para cada linha assinalar com x em célula correspondente)

	Fraco	Razoável	Bom	Muito Bom
BD Mongo				x
BD Mysql			x	
Proc. Pré Mysql				x
Proc. Pós Mysql				
Stored P e Funções				x
Triggers				
Utilizadores			x	
Proc. Utilizadores				x
Eventos			x	
HTML				x

4 Alteração de Especificação recebida de migração do outro grupo

Na especificação recebida, o grupo 22 apresentava a seguinte proposta do modelo relacional (Figura 1).

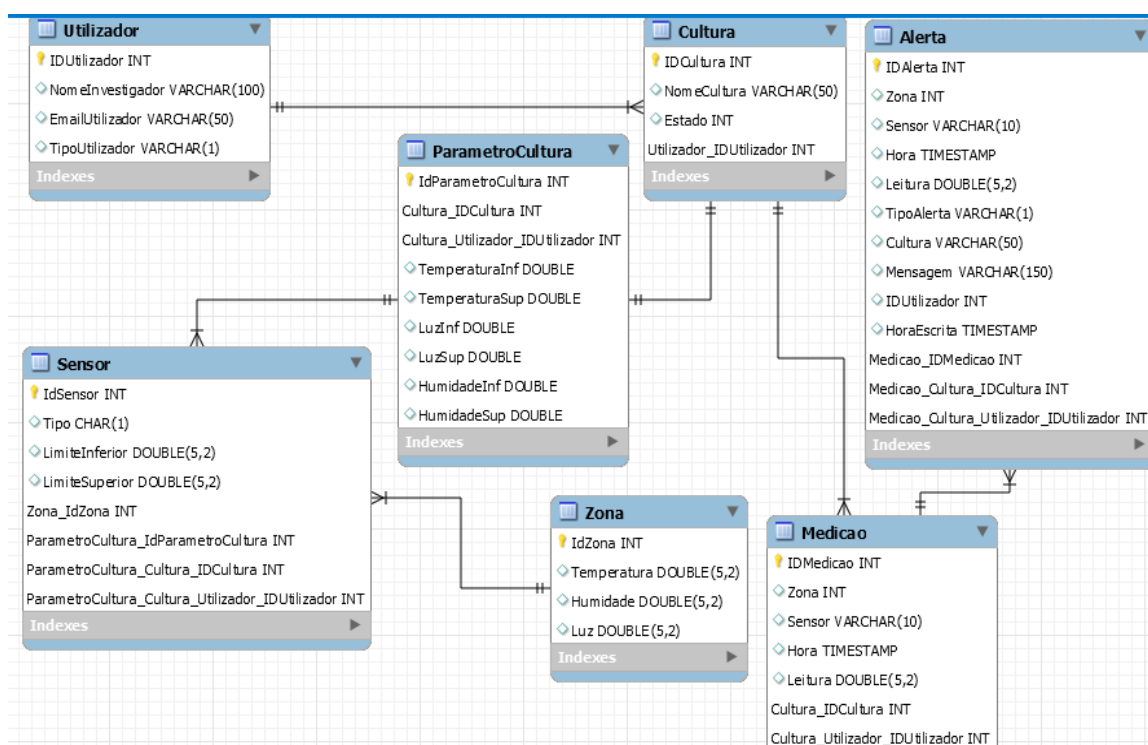


Figura 1 - Base de dados Grupo22

O nosso grupo optou por fazer as seguintes modificações, como pode ser visto na Figura 2.

- **Tabela Parâmetros Cultura:** A chave estrangeira é apenas o IDCultura.
- **Tabela Sensor:** A chave estrangeira é o IDZona.
- **Tabela Medição:** As medições vêm de um sensor, logo o IDSensor faz sentido ser chave estrangeira.
- **Tabela Utilizador:** Esta tabela mantém-se igual.
- **Tabela Cultura:** Acrescentamos o atributo IDZona como chave estrangeira, pois uma cultura pertence a uma certa zona.
- **Tabela Zona:** Mantém-se igual à especificação proposta.
- **Tabela Alerta:** Na especificação recebida existiam vários atributos repetidos, logo optamos por ter como chave estrangeira o IDSensor, IDUtilizador, IDCultura, este último pode estar a null, pois há alertas que não são referentes a uma cultura só. Os outros atributos são a Data, Medição, NivelAlerta, NomeCultura, Mensagem e a HoraEscrita.

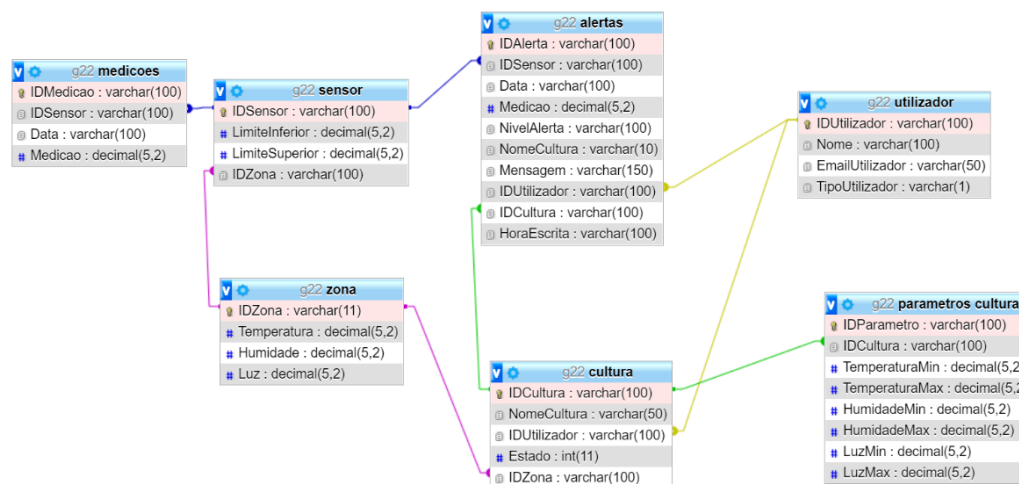


Figura 2 - Base de dados Grupo 20

Devido à falta de especificação de como a migração de dados por MQTT foi realizada por parte do outro grupo não existe um termo de comparação entre o que divergiu da especificação recebida e o que realmente foi implementado. E por essa razão, a implementação foi feita ao nosso critério.

No processo de migração, foi utilizado duas bibliotecas, `mongodb-driver-3.4.3.jar`, utilizada para a ligação à base de dados mongo na cloud, e `mysql-connector-java-8.0.15.jar` que possibilita a inserção de dados no Mysql.

Numa primeira fase, através da thread `MongoToCloud` é estabelecida uma ligação com a MongoDB da Cloud.

Após o estabelecimento desta ligação, é necessário recolher os dados existentes na MongoDB da Cloud. E, para tal, acedemos a cada uma das coleções da base de dados através de um objeto do tipo `MongoCollection<TDocument>`. À medida que são recebidos os dados, estes são enviados para o broker MQTT através do tópico:

- Nome Tópico: `medicoes_g20` Cliente: `sid2021_g20` QOS:0

Respeitando a especificação feita pelo outro grupo, foi considerada uma periodicidade de 5 segundos.

De modo a garantirmos que não são enviados dados repetidos, foi utilizado um QoS 0 “at most once”.

À medida que são enviados os dados para o broker MQTT, existe uma thread designada de `CloudToLocal`, que se encarrega de receber os dados do MQTT. Por cada documento recebido a thread `CloudToLocal` cria uma thread `LocalToSql` que se encarrega de filtrar os dados, de modo a isolar as componentes cruciais para posteriormente serem colocadas as medições no MySQL. Este tratamento de dados pré-sql é feito através do método `getDataToInsert` da classe `Medicao`. Após esta função ser invocada, existe, na mesma classe, a função `insertNewMedicao`, que devolve uma String (“Query”) que irá permitir a inserção da nova medição no MySQL.

Se por algum motivo a transferência dos dados for abaixo, e assumindo que a DB da Cloud se mantém inalterável, isto é, os dados que já descarregamos até ao momento não sofreram alterações, é possível

recolher os dados a partir do ponto onde ficamos, através do método `skip(int count)`. Neste método, o `count` corresponde ao número de medições já enviadas para o MySQL.

Para finalizar o processo de migração dos dados, na thread `LocalToSql`, através do método `adicionaMedicao`, é criada uma nova medição que é encaminhada para o MySQL através do método `enviaQuery(String query)` e posteriormente é adicionada numa `LinkedList`.

Esta `LinkedList` é submetida a uma análise de deteção do alerta do tipo:

- Alerta 1 (muito urgente) - “O sensor t da zona 1 apresenta valores superiores/inferiores aos definidos para a zona.”
 - É enviado se, após 60 medições, a medição continuar acima/abaixo dos valores definidos para a zona.
 - É reenviado se, após 30 minutos, a medição continuar fora dos limites definidos para a zona.
- Alerta 2 (pouco urgente) - “A temperatura da cultura x encontra-se dentro dos limites definidos.”
 - É enviado após 30 minutos do alerta 1 se a medição voltar a ficar dentro dos limites definidos.

Para facilitar a análise, foram criadas várias threads e uma classe “Alerta”, com os atributos “IDAlerta”, “IDSensor”, “Data”, “Medicao”, “NivelAlerta”, “Cultura”, “Mensagem”, “IDUtilizador”, “IDCultura” e “HoraEscrita”. Em cada thread, com recurso ao método `executeQuery(String s)`, são extraídos os valores que posteriormente serão colocados nos atributos, à exceção da mensagem e do nível de alerta, que serão apenas definidos após a verificação das condições imposta pelos alertas.

Esta análise começa pela classe “LerMedicao”, que extrai cada medição inserida e verifica se esta se encontra dentro dos limites definidos para a zona correspondente.

Caso não esteja dentro dos limites, é invocada a classe “AlertaUm”, na qual é verificada, após 60 medições, se estas se encontram acima/abaixo dos limites da zona. Caso isso se verifique, então é criado um objeto da classe Alerta (o Alerta 1), com o atributo nível de alerta e a mensagem. Através da função `insertNewAlerta()`, é criada uma query que será dada como parâmetro no método `executeUpdate(String query)`, responsável por enviar o alerta para o MySQL.

Após ser enviado o alerta acima, é criada uma thread “TimerAlertaUm” que, passados 30 minutos, irá verificar se as medições que estão a ser recebidas na `LinkedList` se mantêm ou não fora dos limites definidos para a zona. Caso isso se verifique, então é criado um objeto Alerta (o Alerta 1), que será enviado novamente para o MySQL. Após este envio, é criada, de novo, uma thread “TimerAlertaUm”. Caso contrário, é gerado um objeto Alerta (o Alerta 2), com o atribuído nível de alerta e a mensagem, que será encaminhado para o MySQL.

Optámos por ter um único main presente na Classe “App”, que inicia e conclui todo o processo. Este inicia duas threads:

- 1) MongoToCloud, responsável pela migração dos dados desde a MongoDB da nuvem até ao Broker MQTT
- 2) CloudToLocal, que se encarrega de receber os dados do MQTT e de os enviar para a 3.ª thread LocalToSql.

Para além das threads iniciadas pelo main existem ainda:

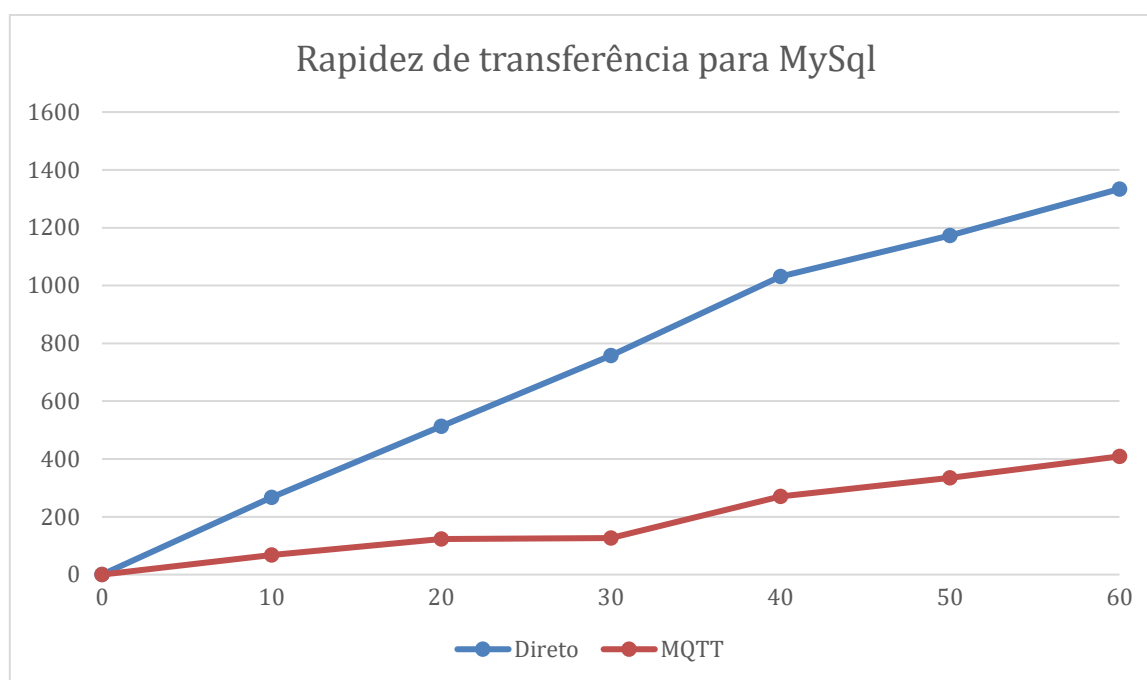
- 3) LocalToSql, que se encarrega de filtrar os dados , de modo a isolar as componentes cruciais para posteriormente serem colocadas as medições no MySQL, esta também se assegura que todas as medições são devidamente avaliadas, de modo a garantir que as mesmas não contêm valores anómalos para as culturas.
- 4) TimerAlertaUm, que verifica se as medições que estão a ser recebidas na LinkedList se mantêm ou não fora dos limites definidos para a zona.

5 Comparação de Implementações de Migrações Mongo/MySQL

1.ª Experiência

Descrição: Esta experiência teve o intuito de testar a velocidade com que as medições eram encaminhadas para o MySQL.

- Tempo da experiência: 1 minuto
 - Migração **Direta** → Linha Azul
 - Número de pacotes perdidos: 0
 - Total de medições encaminhadas para MySQL: 1334
 - Migração por **MQTT** → Linha Vermelha
 - Número de pacotes perdidos: 0
 - Total de medições encaminhadas para MySQL: 409



2.ª Experiência

Descrição: Reiniciar a aplicação passado 1 minuto de ser executada e contar o tempo que demora a voltar a enviar medições para o MySQL.

- Migração **Direta**
 - Tempo que levou a voltar a enviar medições para MySQL: ≈ 3 segundos
- Migração por **MQTT**
 - Tempo que levou a voltar a enviar medições para MySQL: ≈ 4 segundos

3.ªExperiência

Descrição: Iniciar app com o Serviço MySql em baixo, ligá-lo passado 30 segundos e analisar quantidade de pacotes perdidos.

- Migração **Direta**
 - N.º de pacotes perdidos = 0.
- Migração por **MQTT**
 - N.º de pacotes perdidos = 0.

4.ªExperiência

Descrição: App não consegue aceder a servidor Mongo Cloud.

- Migração **Direta**
 - Continua a enviar medições para o MySql até à última medição que descarregou para a Mongo Local.
- Migração por **MQTT**
 - Não envia nenhuma medição para o MySql.

Após as experiências efetuadas, concluímos que em termos de migração de dados, a direta, para o mesmo período de tempo, é mais eficiente visto que consegue transferir mais medições para o MySql (Gráfico 1).

Em termos de robustez, testamos reiniciar ambas as migrações e voltar a executá-las e ver quanto tempo demoravam a enviar medições para o MySql e concluímos que ambas apresentavam tempos semelhantes.

Para um cenário em que as duas implementações não conseguem aceder ao servidor, a migração por MQTT não consegue enviar nenhuma medição para o MySql enquanto a Direta consegue enviar até à última medição que descarregou antes de perder o acesso ao servidor.

Testamos também um cenário onde iniciávamos a aplicação com o Serviço MySql do xampp em baixo e iniciávamos o mesmo passado 30 segundos e percebemos que ambas as migrações não perderam nenhum tipo de medições durante a transferência das mesmas para o MySql.

Relativamente à flexibilidade, através de um ficheiro sensor.ini é possível configurar quais as coleções que pretendemos obter os dados (figura 3), caso se pretenda obter dados de mais coleções basta configurar o ficheiro e adicionar uma linha no fim (figura 4). A Thread readSensor está encarregue de obter as configurações do ficheiro .ini e, por cada coleção, de criar e executar uma Thread MongoConnect que irá obter os dados do servidor da cloud.

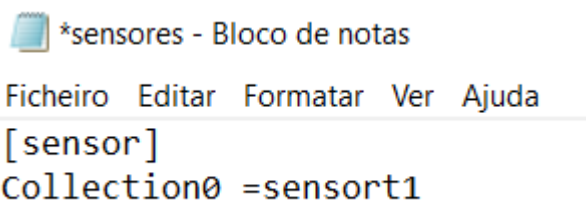


Figura 3

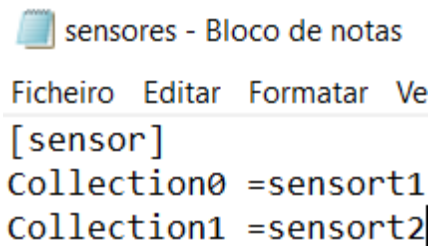


Figura 4

Relativamente à privacidade esta é assegurada posteriormente no MySQL através das roles administrador/investigador que garantem as permissões que definimos no ponto 3.1.6 e na tabela 3.1.6.1 e também através dos scripts definidos para a aplicação que permitem que o utilizador que fez login tenha acesso apenas aos alertas das culturas que possui.

6 Implementação de Alertas, Utilizadores, Php

6.1 Alterações das especificações próprias implementadas face a sugestões de outro grupo

6.2 Alterações das especificações próprias implementadas de iniciativa própria

Na nossa implementação decidimos alterar a periodicidade com que enviamos os dados para mysql para 1 segundo de forma a não sobrecarregar o utilizador com demasiadas medições em menos de 1 segundo e assim garantimos que são recebidas 1 medição a cada segundo.

Adicionamos também uma Thread “readSensor” que está encarregue de obter as configurações de um ficheiro .ini onde estão indicadas as coleções das quais se pretendem obter os dados e, por cada coleção, criar e executar uma Thread MongoConnect que irá obter os dados do servidor da cloud correspondentes a essa coleção. É importante referir que caso se pretenda obter mais coleções basta configurar nesse ficheiro .ini o nome da coleção.

No modelo relacional as chaves primárias passaram a ser do tipo int para facilitar a implementação dos SPs. O valor das medições passou a ser do tipo double para facilitar o uso na implementação da aplicação.

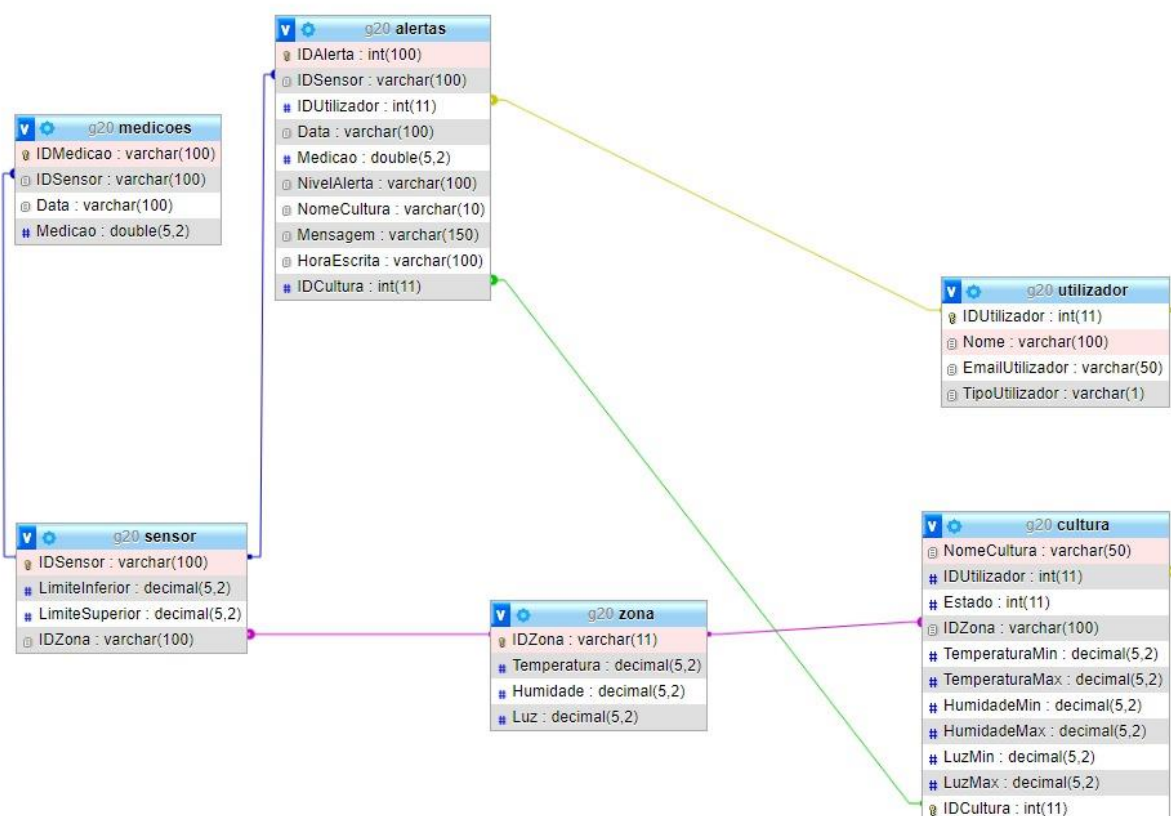
Alteramos o stored procedure criarUtilizador pois o SP antigo criava apenas uma linha nova na tabela Utilizador, este “novo”, para além de criar uma nova entrada na tabela, cria uma conta de acesso à BD conforme o role que tenha associado, investigador ou administrador, assim facilitando assim a atribuição de permissões a tabelas e atributos.

6.3 Coleções em cada uma das réplicas do Mongo

<pre>db.sensorh1.find().pretty() { "_id" : ObjectId("6036c771967bf6108c5b7ca9"), "Zona" : "Z1", "Sensor" : "H1", "Data" : "2021-02-24 at 21:38:56 GMT", "Medicao" : "24.61639494871795"} { "_id" : ObjectId("6036c771967bf6108c5b7caa"), "Zona" : "Z1", "Sensor" : "H1", "Data" : "2021-02-24 at 21:38:57 GMT", "Medicao" : "15.595232230769232"}</pre>	<pre>db.sensorh2.find().pretty() { "_id" : ObjectId("6036c75c967bf61560213019"), "Zona" : "Z2", "Sensor" : "H2", "Data" : "2021-02-24 at 21:38:36 GMT", "Medicao" : "5.0"} { "_id" : ObjectId("6036c75c967bf6156021301a"), "Zona" : "Z2", "Sensor" : "H2", "Data" : "2021-02-24 at 21:38:36 GMT", "Medicao" : "5.0"}</pre>
<pre>db.sensorl1.find().pretty() { "_id" : ObjectId("6036c77f967bf612b4486142"), "Zona" : "Z1", "Sensor" : "L1", "Data" : "2021-02-24 at 21:39:11 GMT", "Medicao" : "318.27"} { "_id" : ObjectId("6036c780967bf612b4486143"), "Zona" : "Z1",</pre>	<pre>db.sensorl2.find().pretty() { "_id" : ObjectId("6036c78c967bf61b7c33dd27"), "Zona" : "Z2", "Sensor" : "L2", "Data" : "2021-02-24 at 21:39:24 GMT", "Medicao" : "1.0256410256410255"} { "_id" : ObjectId("6036c78d967bf61b7c33dd28"), "Zona" : "Z2",</pre>

"Sensor" : "L1", "Data" : "2021-02-24 at 21:39:11 GMT", "Medicao" : "327.81809999999996"}	"Sensor" : "L2", "Data" : "2021-02-24 at 21:39:24 GMT", "Medicao" : "1.282051282051282"}
db.sensor1.find().pretty() { "_id" : ObjectId("603be729967bf6135c29555c"), "Zona" : "Z1", "Sensor" : "T1", "Data" : "2021-02-28T18:55:37Z", "Medicao" : "7.160838461538458"} { "_id" : ObjectId("603be729967bf6135c29555d"), "Zona" : "Z1", "Sensor" : "T1", "Data" : "2021-02-28T18:55:37Z", "Medicao" : "6.296503076923074"}	db.sensor2.find().pretty() { "_id" : ObjectId("6036c7a5967bf62d2c982203"), "Zona" : "Z2", "Sensor" : "T2", "Data" : "2021-02-24 at 21:39:48 GMT", "Medicao" : "20.6"} { "_id" : ObjectId("6036c7a5967bf62d2c982204"), "Zona" : "Z2", "Sensor" : "T2", "Data" : "2021-02-24 at 21:39:49 GMT", "Medicao" : "21.218"}

6.4 Base de Dados Mysql



6.5 Tratamento de medições pré-Mysql

Na classe *Medicao* dispomos de dois métodos. O primeiro, *getDataToInsert()*, recebe como argumento uma String *s*. Esta String será o resultado de cada um dos objetos da respetiva coleção existente na BD local.

Exemplo: String *s* = "Document{{_id=604f7f79967bf60db42f88c0, Zona=Z1, Sensor=T1, Data=2021-03-15T15:38:33Z, Medicao=13.666666666666664}}".

Recorrendo às funções *split()* e *replace()*, foi possível isolar num vetor (String [] *data*) os dados que pretendemos, obtendo o seguinte resultado:

```
data[0]=604f7f79967bf60db42f88c0
data[1]=Z1
data[2]=T1
data[3]=2021-03-15 15:38:33
data[4]=13.666666666666664
```

Após invocarmos estas duas últimas funções, colocamos os atributos (*idMedicao*, *idZona*, *idSensor*, *data* e *medicao*) com os respetivos valores.

Depois desta atribuição, a função *insertNewMedicao()* devolve, no formato String, a Query que servirá para inserirmos no MySQL os dados relativos a cada medição. No nosso exemplo, esta função retornará o seguinte: "INSERT INTO `medicoes` (`IDMedicao`, `IDSensor`, `Data`, `Medicao`) VALUES ('604f7f79967bf60db42f88c0', 'T1', '2021-03-15 15:38:33', '13.666666666666664')". Tal como explicado no modelo relacional, um sensor já tem uma zona associada, logo o atributo *IDZona* é omitido.

A thread *MysqlConnect* está encarregue de realizar todo este tratamento e filtragem de dados, para que seja possível inseri-los corretamente no MySQL.

Alertas

Através de uma "MongoCollection<TDocument>" acedemos aos dados de cada coleção da nossa MongoDB local, para cada um destes é criado um objeto medição que através do método *getDataToInsert(String s)* da classe "Medicao" atribui valores aos seus atributos: "idMedicao", "idZona", "idSensor", "data" e "medicao". Após criada, a medição é automaticamente inserida numa *LinkedList*.

Após cada inserção na *LinkedList* é criada uma *thread* "UpdateZona", que recebe como parâmetros o *IDZona* e um objeto do tipo "Connection". Esta *thread* é responsável por atualizar os valores da temperatura, humidade e luz para cada zona, através do método *updateMedicao()*, no qual é criada uma query com as medições atualizadas e, posteriormente, enviada para o MySQL.

Esta *LinkedList* é submetida a uma análise de deteção dos quatro tipos de alertas. Para facilitar a análise, foram criadas 3 classes (*AlertaUm*, *AlertaDois* e *AlertaTres*), 3 threads (*TimerAlertaUm*, *TimerAlertaDois* e *TimerAlertaTres*) e uma classe "Alerta", com os atributos "IDAlerta", "IDSensor", "Data", "Medicao", "NivelAlerta", "Cultura", "Mensagem", "IDUtilizador", "IDCultura" e "HoraEscrita". Em cada uma das classes *AlertaUm*, *AlertaDois* e *AlertaTres*, com recurso ao método *executeQuery(String s)*, são extraídos os valores que posteriormente serão colocados nos atributos do objeto da classe *Alerta*, à exceção da mensagem e do nível de alerta, que serão apenas definidos após a verificação das condições impostas.

Esta análise começa pela classe “LerMedicao”, que extrai cada medição inserida e verifica se esta se encontra dentro dos limites definidos para a zona correspondente.

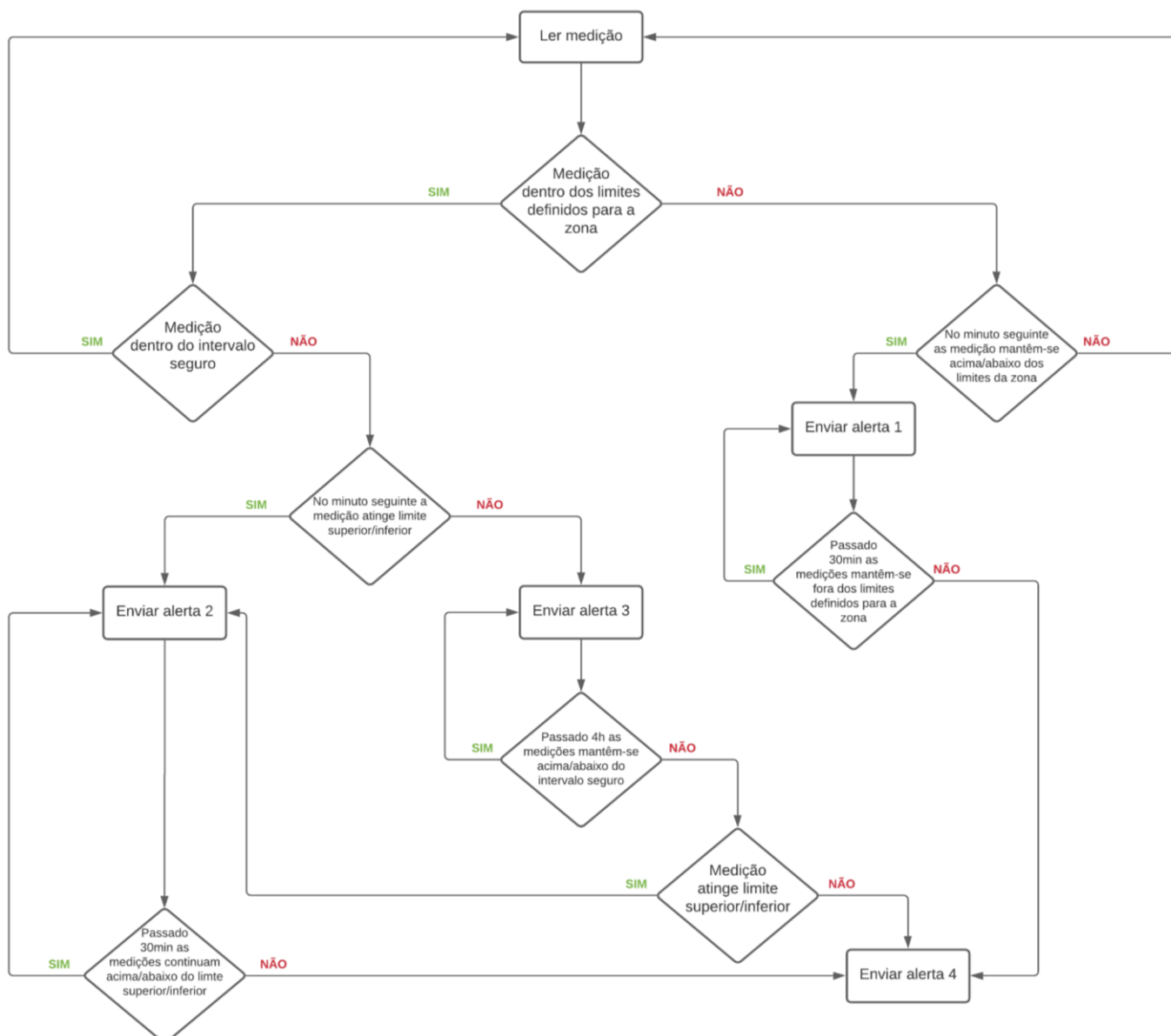
Caso não esteja dentro dos limites, é invocada a classe “AlertaUm”, na qual é verificada, após 60 medições, se estas se encontram acima/abaixo dos limites da zona. Caso isso se verifique, então é criado um objeto da classe Alerta (o Alerta 1), com o atributo nível de alerta e a mensagem. Através da função insertNewAlerta(), é criada uma *query* que será dada como parâmetro no método executeUpdate(String query), responsável por enviar o alerta para o MySQL.

Após ser enviado o alerta acima, é criada uma *thread* “TimerAlertaUm” que, passados 30 minutos, irá verificar se as medições que estão a ser recebidas na *LinkedList* se mantêm ou não fora dos limites definidos para a zona. Caso isso se verifique, então é criado um objeto Alerta (o Alerta 1), que será enviado novamente para o MySQL. Após este envio, é criada, de novo, uma *thread* “TimerAlertaUm”. Caso contrário, é gerado um objeto Alerta (o Alerta 4), com o atribuído nível de alerta e a mensagem, que será encaminhado para o MySQL.

Caso a medição se encontre dentro dos limites definidos para a zona, é invocada a classe “AlertaTres” que, através do método executeQuery(String query), extrai as temperaturas máxima e mínima para cada cultura e, com base nesses dados, calcula o seu intervalo seguro, comparando-o com a medição recebida na *LinkedList*. Se a medição não estiver dentro do intervalo seguro da cultura, é verificado, após 60 medições, se estas atingem ou não limite superior/inferior da cultura. Em caso afirmativo, é criada uma classe “AlertaDois”, que irá criar um objeto da classe Alerta (o Alerta 2) com o atribuído nível de alerta e a mensagem, e envia este alerta para o MySQL.

Após ser enviado o alerta, é criada uma *thread* “TimerAlertaDois” que, passados 30 minutos, irá verificar se as medições que estão a ser recebidas na *LinkedList* se mantêm ou não fora dos limites definidos para a cultura. Caso isso não se verifique, então é gerado um objeto Alerta (o Alerta 4) com o atribuído nível de alerta e a mensagem, que será encaminhado para o MySQL. Por outro lado, caso se confirme que a medição se encontra fora dos limites, é criado um objeto Alerta (Alerta 2) que será enviado novamente para o MySQL. Após o envio, é gerada uma nova *thread* “TimerAlertaDois”.

Se após 60 medições, se confirma que estas não atingem o limite superior/inferior da cultura, então é criado um objeto da classe Alerta (o Alerta 3) com o atribuído nível de alerta e a mensagem, encaminhando de seguida o alerta para o MySQL. Após o envio do alerta 3, é criada uma *thread* “TimerAlertaTres” que, passadas 4 horas, verifica, através de uma condição *if*, se a medição contínua acima/abaixo do intervalo seguro. Se a condição se verificar, será criada novamente uma *thread* “TimerAlertaTres”, caso contrário é gerado um objeto Alerta (o Alerta 4) com o atribuído nível de alerta e a mensagem.



Legenda

- Intervalo seguro: $[\text{limite inferior} + 10\% \text{ do número de valores entre o limite mínimo e máximo, limite superior} - 10\% \text{ do número de valores entre o limite mínimo e máximo}]$. Exemplo: A cultura A aguenta temperaturas entre os 3°C e os 30°C. Assim o seu intervalo seguro será $[3+(10\%*(30-3)), 30-(10\%*(30-3))] = [5.7, 27.3]$
- Alerta 1 (muito urgente)**
 “O sensor t da zona 1 apresenta valores superiores/inferiores aos definidos para a zona.”
 É enviado se, após 60 medições, a medição continuar acima/abaixo dos valores definidos para a zona.
 É reenviado se, após 30 minutos, a medição continuar fora dos limites definidos para a zona.

- **Alerta 2 (muito urgente)**
 “A temperatura da cultura x ultrapassou a temperatura limite.”
 É enviado se a medição atingir um valor acima/abaixo do limite superior/inferior da cultura. É reenviado de 30 em 30 minutos se a medição continuar acima/abaixo do limite superior/inferior da cultura.
- **Alerta 3 (urgente)**
 “A temperatura da cultura x está a aproximar-se da temperatura limite.”
 É enviado se, após 60 medições, a medição continuar acima/abaixo do intervalo seguro. É reenviado de 4 em 4 horas se a medição continuar acima/abaixo do intervalo seguro.
- **Alerta 4 (pouco urgente)**
 “A temperatura da cultura x encontra-se dentro dos limites definidos.”
 É enviado após 30 minutos dos alertas 1 e 2, e 4 horas do alerta 3 se a medição voltar a ficar dentro dos limites definidos.

6.6 Tratamento de medições pós-Mysql

6.6.1 Especificação de Triggers implementados

Nome Trigger	Tabela	Tipo de Operação (I, U, D)	Evento (After, Before)	=, Novo, Alterado

6.6.2 Código de Triggers implementados

```

1. Nome Trigger: _____
// Breve Descrição
Código

2. Nome Trigger: _____
// Breve Descrição
Código

3. Nome Trigger: _____
// Breve Descrição
Código

```


6.6.3 Especificação de todos Procedimentos e Funções implementados

Nome Procedimento / Função	Parâmetros Entrada	Parâmetros Saída	Valores Retorno	=, Novo, Alterado
CriarUtilizador	Nome EmailUtilizador TipoUtilizador Password	Utilizador	-	Alterado
RemoverUtilizador	IDUtilizador	-	-	=
ListarInvestigadores	-	Tabela com os nomes dos investigadores	-	=
CriarCultura	IDCultura NomeCultura IDUtilizador Estado IDZona TemperaturaMin TemperaturaMax HumidadeMin HumidadeMax LuzMin LuzMax	Cultura	-	=
RemoverCultura	IDCultura	-	-	=
EditarCultura	IDCultura IDUtilizador	-	-	=
ListarCulturasAdmin	-	Tabelas com todas as culturas	-	=
VisualizarCulturaAdmin	IDCultura	IDCultura NomeCultura IDUtilizador Estado IDZona TemperaturaMin TemperaturaMax HumidadeMin HumidadeMax LuzMin LuzMax	-	=
ListarAlertasCultura	IDCultura	Tabela com todos os alertas da cultura selecionada, com respetiva hora, nível de alerta e mensagem	-	=
ListarNotificacoesAdmin	-	Tabela com todos os alertas com respetiva hora, nível de alerta, nome da cultura e mensagem	-	=

ListarCulturasZona	IDZona	Tabela com todas as culturas da zona selecionada e tabela com as medições atuais	-	=
VisualizarCulturaInv	IDCultura	IDCultura NomeCultura Estado IDZona TemperaturaMin TemperaturaMax HumidadeMin HumidadeMax LuzMin LuzMax e tabela com as medições atuais da zona em que a cultura se encontra	-	=
EditarParametrosCultura	IDCultura NomeCultura TemperaturaMin TemperaturaMax HumidadeMin HumidadeMax LuzMin LuzMax	-	-	=
ListarNotificacoesInv	IDUtilizador	Tabela com todos os alertas do investigador com respetiva hora, nível de alerta, nome da cultura e mensagem	-	=

6.6.4 Stored Procedures e Funções implementadas

1. Nome SP: CriarCultura

Permite ao administrador criar uma cultura, atribuindo-lhe os parâmetros que a definem, podendo ou não lhe ser atribuída um investigador responsável.

Código

```
BEGIN
DECLARE new_cultura integer;
SELECT COUNT(*) into new_cultura FROM cultura cul where
id=cul.IDCultura;
if(new_cultura=0) THEN
    INSERT INTO cultura
        (IDCultura, NomeCultura, IDUtilizador, Estado, IDZona, Temperatura
        Min, TemperaturaMax, HumidadeMin, HumidadeMax, LuzMin, LuzMax)
        VALUES
        (id, NomeCultura, IDUtilizador, Estado, IDZona, TemperaturaMin, Tem
        peraturaMax, HumidadeMin, HumidadeMax, LuzMin, LuzMax);
end if;
END
```

2. Nome SP: CriarUtilizador

Permite ao administrador criar um utilizador, atribuindo-lhe um nome, email, tipo e password, e as permissões necessárias conforme o tipo de role (I=Investigador ou A=Administrador) que desempenha.

Código

```
BEGIN
INSERT INTO utilizador
(IDUtilizador, Nome, EmailUtilizador, TipoUtilizador, Pass) VALUES
(null, nome, email, tipo, pass);
SET @query = CONCAT('CREATE USER "', email, '"@"', 'localhost', '"
IDENTIFIED BY "', pass, '";');
PREPARE stmt FROM @query;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

IF tipo='I' THEN
    SET @query = CONCAT('GRANT "', 'investigator', '" TO "', email,
'"@"', 'localhost', '";');
    PREPARE stmt FROM @query;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    SET @query = CONCAT('SET DEFAULT ROLE "', 'investigator', '"
FOR "', email, '"@"', 'localhost', '";');
    PREPARE stmt FROM @query;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
END IF;
```

```

IF tipo='A' THEN
    SET @query = CONCAT('GRANT "', 'administrador', '" TO "',
        email, '"@"', 'localhost', '"');
    PREPARE stmt FROM @query;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    SET @query = CONCAT('SET DEFAULT ROLE "', 'administrador', '"
        FOR "', email, '"@"', 'localhost', '"');
    PREPARE stmt FROM @query;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
END IF;
END

```

3. Nome SP: EditarCultura

Permite ao administrador editar uma cultura, podendo alterar ou atribuir o investigador responsável pela mesma.

Código

```

BEGIN
    UPDATE cultura SET IDUtilizador=id WHERE IDCultura=id_cultura;
END

```

4. Nome SP: EditarParametrosCultura

Permite ao investigador editar os parâmetros da cultura.

Código

```

BEGIN
    UPDATE cultura SET TemperaturaMin=TempMin WHERE
        IDCultura=id_cultura;
    UPDATE cultura SET TemperaturaMax=TempMax WHERE
        IDCultura=id_cultura;
    UPDATE cultura SET HumidadeMin=HumMin WHERE IDCultura=id_cultura;
    UPDATE cultura SET HumidadeMax=HumMax WHERE IDCultura=id_cultura;
    UPDATE cultura SET LuzMin=LMin WHERE IDCultura=id_cultura;
    UPDATE cultura SET LuzMax=LMax WHERE IDCultura=id_cultura;
END

```

5. Nome SP: ListarAlertasCultura

Permite ao administrador/investigador visualizar todos os alertas da cultura selecionada.

Código

```

BEGIN
    SELECT NomeCultura,HoraEscrita,NivelAlerta,Mensagem FROM alertas
        WHERE id=alertas.IDCultura;
END

```

6. Nome SP: ListarCulturasAdmin

Permite ao administrador visualizar todas as culturas do laboratório.

Código

```

BEGIN
    SELECT NomeCultura FROM cultura;
END

```

7. Nome SP: ListarCulturasZona

Permite ao investigador visualizar todas as culturas da zona selecionada e respetivas medições atuais, também representadas graficamente.

Código

```
BEGIN
SELECT NomeCultura FROM cultura WHERE id=IDZona;
SELECT Temperatura,Humidade,Luz FROM zona WHERE id=IDZona;
END
```

8. Nome SP: ListarInvestigadores

Permite ao administrador visualizar todos os investigadores existentes.

Código

```
BEGIN
SELECT Nome FROM utilizador WHERE TipoUtilizador='I';
END
```

9. Nome SP: ListarNotificacoesAdmin

Permite ao administrador visualizar todas as notificações de todas as culturas.

Código

```
BEGIN
SELECT NomeCultura,HoraEscrita,NivelAlerta,Mensagem FROM alertas;
END
```

10. Nome SP: ListarNotificacoesInv

Permite ao investigador visualizar todos os alertas de todas as suas culturas.

Código

```
BEGIN
SELECT NomeCultura,HoraEscrita,NivelAlerta,Mensagem FROM alertas
WHERE id=IDUtilizador;
END
```

11. Nome SP: RemoverCultura

Permite ao administrador remover uma cultura.

Código

```
BEGIN
DELETE FROM cultura WHERE IDCultura=id;
END
```

12. Nome SP: RemoverUtilizador

Permite ao administrador remover um utilizador e todas as culturas associadas, se for investigador

Código

```
BEGIN
DELETE FROM utilizador WHERE IDUtilizador=id;
END
```

13. Nome SP: VisualizarCulturaInv

Permite ao investigador visualizar todos os parâmetros associados à cultura selecionada e respetivas medições atuais.

Código

```
BEGIN
SELECT
IDCultura, NomeCultura, Estado, IDZona, TemperaturaMin, TemperaturaMax,
HumidadeMin, HumidadeMax, LuzMin, LuzMax FROM cultura WHERE
id_cultura=IDCultura;
SELECT Temperatura, Humidade, Luz FROM zona, cultura WHERE
id_cultura=IDCultura AND cultura.IDZona=zona.IDZona;
END
```

14. Nome SP: VisualizarCulturasAdmin

Permite ao administrador visualizar todos os parâmetros associados à cultura selecionada e respetivas medições atuais.

Código

```
BEGIN
SELECT * FROM cultura WHERE id=IDCultura;
END
```

6.7 Utilizadores Base de Dados Mysql


Tabela	Tipo de Utilizador	
	Investigador	Administrador
alertas	L	L
medicoes	L	L
utilizador	-	E
sensor	-	E
zona	-	L
cultura	E (apenas os parâmetros)*	E (apenas IDUtilizador)*
Stored Procedures		
ListarInvestigadores	-	X
ListarCulturasAdmin	-	X
VisualizarCulturaAdmin	-	X
ListarAlertasCultura	X	X
ListarNotificacoesAdmin	-	X
ListarCulturasZona	X	-
VisualizarCulturaInv	X	-
ListarNotificacoesInv	X	-
CriarUser	-	X
RemoverUtilizador	-	X
CriarCultura	-	X
RemoverCultura	-	X
EditarCultura	-	X
EditarParametrosCultura	X	-



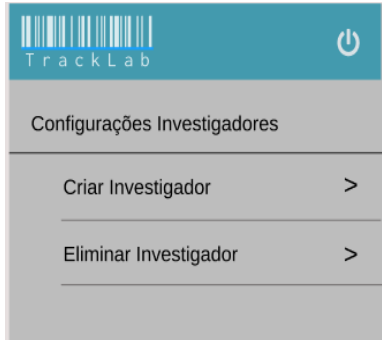



*através dos stored procedures

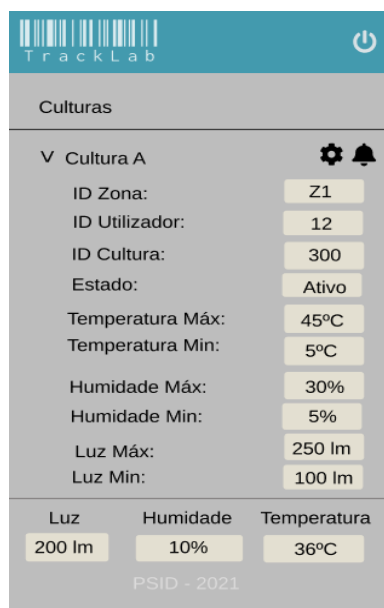
6.8 Eventos de suporte à aplicação

Nome Evento	Local Execução (MySQL/Windows)	Muito breve descrição
Alerta 1	Windows	O Utilizador recebe uma notificação (aviso), consoante o alerta gerado.
Alerta 2	Windows	
Alerta 3	Windows	
Alerta 4	Windows	

6.9 PrintScreen dos formulários HTML


TRACKLAB	
Login	
	<p>O utilizador faz o login com as credenciais que lhe foram atribuídas.</p> <p>Se for o administrador da aplicação, será direcionado para aplicação como tal (1).</p> <p>Caso seja investigador, será direcionado para a aplicação como tal (2).</p> <p>No caso de o investigador não se lembrar da sua password, terá que enviar um e-mail ao administrador da <i>app</i>, a requisitar a reposição da mesma.</p>


Administrador (1)	
	<p><u>Investigadores</u> Ao clicar neste botão, o administrador terá acesso à lista de investigadores existentes. Pode, ainda, configurar os investigadores.</p> <p><u>Culturas</u> Ao clicar neste botão, o administrador terá acesso à lista de culturas existentes. Pode, ainda, configurar as culturas.</p> <p><u>Notificações</u> Ao clicar neste botão, o administrador terá acesso ao seu histórico de avisos/notificações.</p> <p><u>RESET</u> Em caso de falha das migrações das leituras dos sensores MongoDB para MySQL, o administrador recebe um aviso de falha da migração de dados, podendo reinicializar o processo através do botão RESET, que invoca o método resetMongo().</p> <p>Ainda é possível, através do botão de log-out , no canto superior direito da imagem, fazer o log-out da aplicação.</p>
	<p>Na secção Investigadores, ao clicar no botão , o administrador pode aceder às configurações:</p> <ul style="list-style-type: none"> • Criar um investigador novo; • Eliminar um investigador existente;
	<p>Na secção Culturas, ao clicar no botão , o administrador pode aceder às configurações:</p> <ul style="list-style-type: none"> • Criar uma nova cultura; • Apagar uma cultura existente; • Editar uma cultura, podendo exclusivamente alterar o investigador responsável pela mesma.



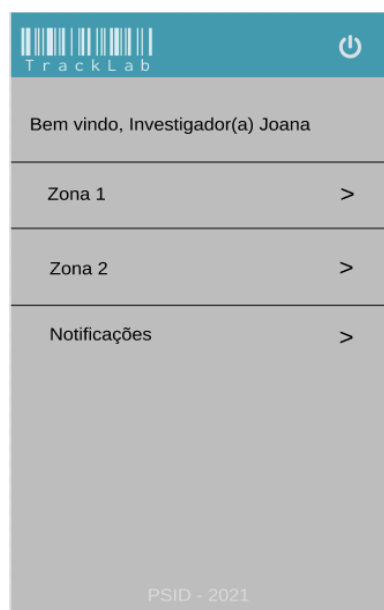
Aquando da visualização da lista de culturas existentes, o administrador pode selecionar uma dada cultura (ex.: cultura A).

Quando é selecionada a cultura, o administrador pode visualizar os seus dados: ID da zona a que pertence, ID do utilizador responsável (caso exista um), ID dessa mesma cultura, estado (ativo/inativo) e os valores máximos e mínimos de temperatura, humidade e luz (parâmetros introduzidos pelo investigador). Também é possível visualizar os valores atuais de luz, humidade e temperatura, no canto inferior do ecrã.

Ainda, dentro de uma determinada cultura, é possível eliminá-la e/ou atribuí-la a um Investigador, através do botão de configurações .


O administrador consegue também aceder ao histórico de notificações (avisos) da cultura selecionada, através do botão de notificações .

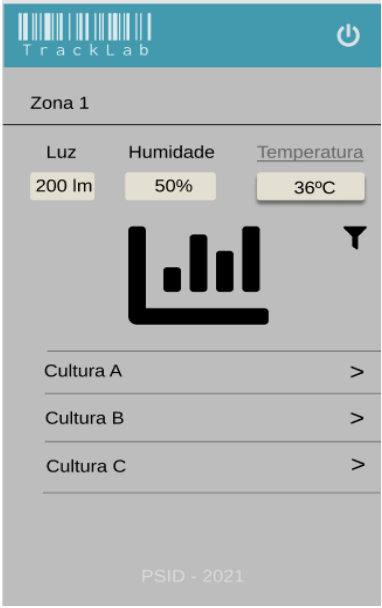
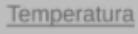





Investigador (2)



Após ter efetuado o login com sucesso, o investigador consegue visualizar uma mensagem de boas-vindas com o seu nome.

É visível um menu no qual o investigador poderá selecionar uma das seguintes opções: a zona a que deseja aceder (Zona 1 ou Zona 2) e as "Notificações", onde irá aparecer uma lista com o histórico de notificações (avisos) recebidas.

Ainda é possível, através do botão de log-out , no canto superior direito da imagem, fazer o log-out da aplicação.

	<p>Ao selecionar a Zona 1, irão aparecer as medições, no momento (atual), das variáveis luz, humidade e temperatura. O investigador pode selecionar a variável (luz, humidade ou temperatura) sobre a qual pretende visualizar o gráfico com as medições.</p> <p>Por exemplo, ao clicar em cima do nome “<u>Temperatura</u>” () , consegue visualizar o gráfico com as respetivas medições.</p> <p>É também possível visualizar uma lista com as culturas pertencentes à Zona 1 e selecionar, caso queira ver em detalhe, cada uma delas.</p> <p>O <i>refresh</i> da página é automático.</p>
	<p>No botão “Filtros” () , aparecerá uma janela pop-up, na qual o investigador poderá selecionar qual o intervalo de tempo através do qual deseja visualizar o gráfico da temperatura, luz, humidade.</p>
	<p>Ao selecionar uma determinada cultura, irá aparecer o nome da cultura, o ID e o respetivo estado.</p> <p>Aparecem também os parâmetros definidos para esta cultura, que podem ser editados pelo investigador ao carregar no botão “Editar” () .</p> <p>O investigador pode também visualizar as notificações relativas a esta cultura em particular, carregando no botão “Notificações” () .</p> <p>No canto inferior do ecrã, estão visíveis os valores atuais (no momento) das variáveis luz, humidade e temperatura da zona na qual a cultura se encontra inserida.</p>

6.10 PrintScreen do formulários Android

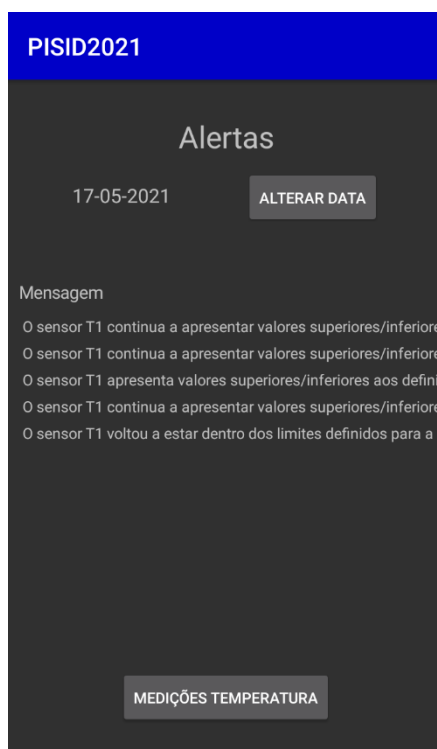
Acesso aos alertas do dia 17-05-2021:



Página Inicial da aplicação

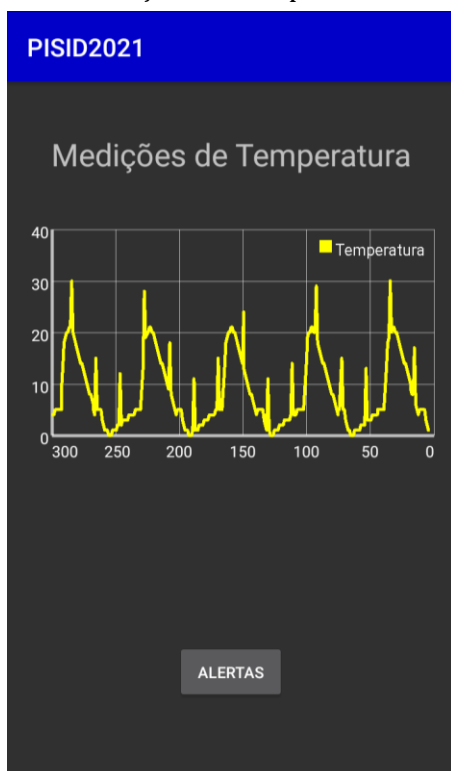


Acesso ao calendário através do botão "Alterar Data"



Visão geral dos Alertas do dia 17-05-2021

Acesso às medições da temperatura:



Acesso ao gráfico das medições da temperatura através do botão "Medições Temperatura"