IPLeiria SmartCampus

Projeto de Integração de Sistemas

André Cordeiro
Engenharia Informática
(of Affiliation)
Instituto Politécnico de Leiria
(of Affiliation)
Leiria, Portugal
2170972@my.ipleiria.pt

André Lopes
Engenharia Informática
(of Affiliation)
Instituto Politécnico de Leiria
(of Affiliation)
Leiria, Portugal
2170971@my.ipleiria.pt

António Barros
Engenharia Informática
(of Affiliation)
Instituto Politécnico de Leiria
(of Affiliation)
Leiria, Portugal
2170918@my.ipleiria.pt

Ricardo Bogalho
Engenharia Informática
(of Affiliation)
Instituto Politécnico de Leiria
(of Affiliation)
Leiria, Portugal
2170952@my.ipleiria.pt

***Abstract*—This document consists of the specification for the project IPLeiria SmartCampus**

Keywords—REST, API, MQQT, Message Broker, XML, XSD

I. INTRODUCTION

Este relatório consiste na especificação do projeto realizado no âmbito da unidade curricular de integração de sistemas, especificando a arquitetura global do projeto e cada componente individual, assim como as suas funcionalidades e utilização.

II. SYSTEM ARCHITECTURE

A arquitetura do sistema implementado consiste em várias aplicações, cada uma desempenhando apenas uma só funcionalidade como por exemplo, mostrar dados de uma forma legível ao utilizador. Foram utilizadas tecnologias tais como REST (Representational State Transfer) e MQTT (mosquitto) broker para a transferência de mensagens entre as diferentes aplicações.

As aplicações foram concebidas para funcionarem de forma isolada, sendo que, a arquitetura permite a adição de novas aplicações sem qualquer alteração das aplicações já existentes. Da mesma forma, caso seja necessário a alteração ou remoção de aplicações já existentes não afetará a utilização das outras.

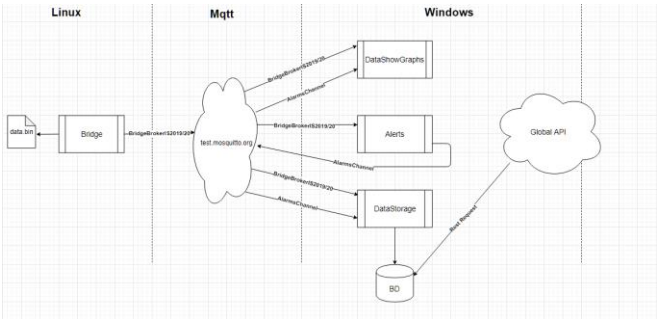

Fig. 1. System Architecture.

A. Mqtt

As aplicações desenvolvidas neste projeto interagem entre elas pelo uso de um *message broker* neste casso o MQQT. É usado o servidor publico "test.mosquitto.org" pois não nos é possível gerir o nosso próprio servidor devido a restrições monetárias. Sendo que devido a isto existem diversos problemas com o nosso sistema de mensagem sendo de notar a falta de autenticação por parte do broker para não permitir a partilha de mensagens para os canais utilizados de utilizadores maliciosos. Sendo que tentamos minimizar este problema validando sempre os dados recebidos por parte do *message broker*. Outro dos problemas é o facto de este servidor ser gerido por uma organização externa podendo então ser encerrado sem qualquer aviso por parte da organização, impossibilitando assim a partilha de mensagens entre as aplicações.

Os canais utilizados pelas aplicações são o canal "BridgeBrokerIS2019/20" para a partilha dos dados dos sensores, e o canal "AlarmsChannel" para a partilha dos dados com os alertas disputados pela aplicação "Alerts", estes canais não são possíveis alterar pelo utilizador para não aumentar a complexidade da utilização das aplicações.

B. Bridge

A aplicação Bridge foi desenvolvida para o sistema operativo Linux e trata da parte de interpretação e transmissão dos dados binários provenientes dos sensores da biblioteca.

A aplicação foi desenvolvida para ser o mais flexível possível podendo assim interpretar qualquer ficheiro binário proveniente dos sensores, tenha ele a temperatura, a humidade ou qualquer outro tipo de sensor como luminosidade, ruido entre outros que o utilizador desejar. Deste modo a aplicação permite adicionar novos campos para serem lidos do ficheiro binário assim como remover campos que já não sejam necessários para além dos parâmetros obrigatórios como a "bateria", "timestamp" e "id" do sensor. Também é permitido ao utilizador manter uma lista de parâmetros guardados no "backlog" que não serão lidos do ficheiro binário assim como alterar a ordem dos parâmetros

que serão lidos para terem a mesma ordem de como se encontram no ficheiro. A aplicação irá também guardar o "backlog" assim como os parâmetros quando a aplicação é encerrada, minimizando assim a necessidade de voltar a introduzir os parâmetros já existentes quando a aplicação é iniciada novamente. Encontra se também um botão de ajuda na aplicação em caso do utilizador não ser experiente com a aplicação e necessitar de ajuda. Depois dos parâmetros estarem de acordo com a necessidade do utilizador, este pode iniciar a leitura do ficheiro clicando no botão "start", a aplicação vai então pedir ao servidor "MQQT" a ser utilizado para a transmissão de mensagens. Por omissão o servidor utilizado é o "test.mosquitto.org" e o canal utilizado é o "BridgeBrokerIS2019/20" sendo que este não é possível ser alterado. É também permitido ao utilizador ativar a opção de "log" que irá gravar num ficheiro todos os dados que serão enviados. Por omissão a aplicação ira ler o ficheiro a partir da última leitura feita a esse mesmo ficheiro, sendo que o utilizador pode mudar este comportamento escolhendo uma das outras duas opções:

- "Only read new": que irá ler apenas novas entradas no ficheiro a partir do momento em que o utilizador der início á leitura do ficheiro.
- "Read complete File": que irá ler o ficheiro todo independentemente se essas entradas já tiverem sido enviadas.

É de notar que os registos do ficheiro binário serão enviados em formato XML (Extensible Markup Language) e em *clear text*, pois apesar de considerarmos que o envio de dados em *clear text* normalmente corresponde a uma falha de segurança, neste caso, devido á natureza dos dados não ser informação critica, não consideramos necessário o uso de encriptação.

C. *DataShowGraphs*

A aplicação foi desenvolvida com o intuito de mostrar informação em tempo real para o utilizador com auxílio de gráficos e tabelas, essa informação é obtida a partir dos canais "AlarmsChannel" e "BridgeBrokerIS2019/20" do servidor MQTT "test.mosquitto.org". Os dados recebidos têm o formato XML e são validados por o respetivo XSD de forma a validar os dados antes de serem inseridos, os valores válidos são mostrados ao utilizador através de uma lista.

A aplicação dispõe de um gráfico para ajudar o utilizador a visualizar as alterações dos dados ao longo do tempo, assim como uma tabela para visualizar com mais pormenor os dados recebidos.

Estão disponíveis dois gráficos para visualização um para os dados dos alertas e outro para os dados sem alertas. O utilizador pode alternar entre o tipo de gráfico que quer visualizar a partir da seleção entre dois botões.

Atualmente, a aplicação permite fazer vários filtros aos dados deixando o utilizador selecionar o id específico, os ids disponíveis são atualizados automaticamente quando são recebidos novos valores com ids diferentes, de um sensor, o tipo de dados desejados e por localização, piso e tipo. Estes filtros são preenchidos à medida que são recebidas mensagens com dados novos.

A filtragem por localização vai ter uma lista de pisos, como no momento não são recebidos dados em relação a localização e o piso em específico insere-se por defeito que a localização é a "Biblioteca" e o piso depende do id do sensor. Esta implementação foi feita para garantir que a aplicação está preparada para a eventual evolução do sistema.

O filtro dos pisos no gráfico linear apresenta pela ordem que recebemos os dados de todos os sensores associados a esse piso, o tempo não foi tido em consideração nesta fase, seria possível implementar futuramente fazendo a média dos sensores num intervalo de x a x tempo.

Uma das limitações corrente da aplicação é a implementação do gráfico que sempre que é recebido um valor válido, o gráfico é redesenhado, o que leva a um breve intervalo onde o gráfico passa a estar indisponível, caso estejam a ser recebidos valores novos constantemente, o gráfico vai estar constantemente a ser apagado e desenhado, o que leva à inibição da visualização dos gráficos por parte do utilizador. Foram então, implementados botões de forma a interromper e retomar o desenho automático dos gráficos.

Destacando que nenhum dado recebido é guardado localmente, enquanto a aplicação corre o utilizador pode aceder aos vários dados que esta recebeu, mas quando se termina, o acesso a estes é perdido.

D. *Alerts*

A aplicação de alertas foi desenvolvida para permitir ao utilizador gerir que valores dos sensores podem despoletar alarmes. Nesta aplicação o utilizador em vez de criar o alerta em si, define os intervalos ou valores entre os quais este quer que estejam, por exemplo, em vez de definir um alerta para caso o valor do sensor de temperatura passar os 40ºC, nesta aplicação o utilizador define que quer que a temperatura seja menor que 40ºC.

A aplicação quando inicia liga-se automaticamente ao canal "BridgeBrokerIS2019/20" e ao receber os dados deste canal, mostra-os ao utilizador, sendo estes dados o "id", "bateria", "timestamp", mais os valores de qualquer sensor recebido e apesar de ele mostrar os valores de todos os sensores que recebe nesta aplicação é apenas possível gerir valores para sensores de humidade e temperatura, outros sensores adicionais teriam de ser implementados.

Após ligar a aplicação o utilizador pode inserir os valores que quer para a temperatura ou a humidade, e adicionar um parâmetro, depois de adicionar este parâmetro este vai ser apresentado ao utilizador juntamente com todos os outros que o utilizador já tenha registado, estes parâmetros são persistidos para quando o utilizador fechar e voltar a abrir a aplicação estes continuarão registados para o utilizador não os ter de inserir novamente e caso o utilizador queira, este pode também remover os requisitos presentes a qualquer momento. O utilizador pode também ativar ou desativar um parâmetro clicando no botão *enable* ou no botão *disable* respetivamente. Depois de dar *enable* a todos os parâmetros necessários, quaisquer dados recebidos do canal "BridgeBrokerIS2019" vão ser analisados pela aplicação para ver se estão dentro de acordo com os parâmetros ativos, caso não estejam então uma mensagem de alerta mais os dados recebidos do canal "BridgeBrokerIS2019" vai ser enviada para o canal "AlarmsChannel" do servidor "test.mosquitto.org" em formato XML em *clear text*, se estiver de acordo com os requisitos então os dados recebidos não serão enviados.

E. *DataStorage*

A aplicação "DataStorage" tem como objetivo persistir os dados, por isso, foi criada uma base de dados modelada de forma a facilitar a inserção de dados e pesquisas.

Em termos visuais a aplicação só tem um botão para iniciar, quando clicado liga-se a base de dados remota e subscreve-se aos canais utilizados pelas outras aplicações de formar a receber os dados que serão persistidos.

Para persistir os dados sem alertas verifica-se se o id do sensor que enviou já existe na Base de Dados se não existir os dados não são persistidos por questões de segurança, também não aceitamos dados repetidos e também se valida a data recebida, se passou nestas validações os dados são persistidos.

No caso dos alertas, para alem das validações já descritas em cima, também se cria um registo quando se recebe um alarme cujos dados do sensor ainda não estão registados, pelo motivo que, se por alguma razão a aplicação parar de receber dados do canal dos sensores ou os dados ainda não terem sido recebidos, não se perde informação relativa aos dados que despoletaram o alarme.

F. BD

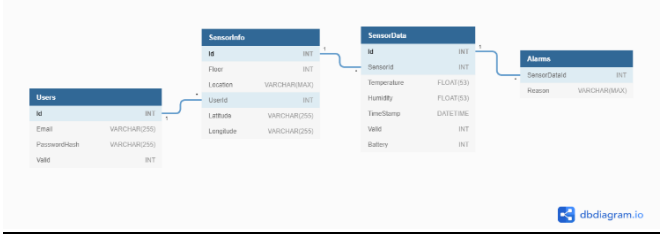

Fig. 2. Database Model.

Na criação da base de dados decidimos separar os dados dos sensores por 3 tabelas, tornando a mais apta a mudanças e permitir a consulta de informação mais eficiente.

A tabela SensorInfo é responsável por guardar as caracteristicas de um sensor, guarda tambem o id do utilizador que registou o mesmo.

A tabela SensorData contem a informação sobre os dados que o sensor enviou, onde guardamos a temperatura e outros valores que possamos receber, o id desta é incremental porque vai servir de foreign key para a tabela Alarms.

A tabela Alarms tem o id da tabela SensorData e uma messagem com a razão do alarme.

A tabela Users contem as informações dos utilizadores que se registam tendo estes que ser posteriormente aceites pelo administrador do sistema.

A base de dados está alojada no servidor cloud do app harbor o que vai permetir mais que uma aplicação aceder à base de dados sem terem que estar na mesma maquina.

G. Global API

O projeto proporciona uma API (Application Programming Interface) que é disponibilizada por um serviço para que os utilizadores possam usar as suas funcionalidades. Esta API proporciona á comunidade académica do IPL acesso aos dados obtidos através de sensores em várias localizações do campus.

Foi escolhido o padrão arquitetural REST para desenvolver a nossa API devido à sua simplicidade, e ao seu baixo consumo de recursos pois espera-se que esta seja consumida maioritariamente por dispositivos móveis.

A API apenas está disponível para utilizadores qualificados, isto é, utilizadores que tenham uma conta registada e ativa. Qualquer membro da instituição se pode registar com o seu email da instituição (*@my.ipleiria.pt) e uma password através da API mas apenas pode aceder aos seus recursos se a sua conta for aceite pelo administrador do sistema garantindo assim um ceto nível de segurança. Para implementar esta autenticação utilizamos “OAuth” que é um protocolo que permite implementar autorização de uma forma simples e segura.

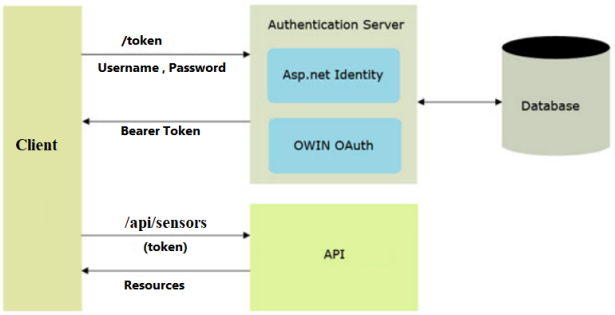

Fig. 3. OAuth architecture.

A documentação para a API está disponível ficheiro “APIDocumentation.docx”.

III. EVALUATION

A. Test bed

As aplicações foram testadas com o método “eat your own dog food” que consiste em testes realizados manualmente pelos desenvolvedores envolvidos.

B. Data analysis

A analise de dados é facultada pela aplicação “DataShowGraph” que consiste em gráficos e uma tabela, demonstrados nas imagens seguintes.

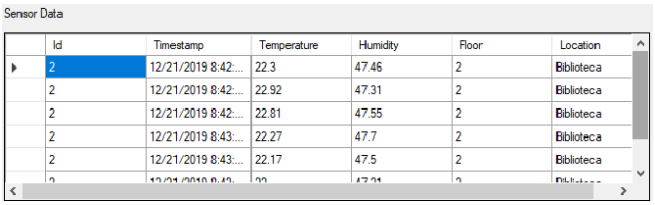

Fig. 4. Data Table.

Fig. 5. Linear Graph.

CONCLUSIONS AND FUTURE WORK

Em suma o projeto foi concebido com diversas funcionalidades em mente como o envio de dados de qualquer sensor, a analise desses dados em forma de gráficos e tabelas, gerir alarmes perante as necessidades do utilizador e persistir dados. Foi também implementado uma API com diversas funcionalidades para futuras aplicações.

Consideramos que os objetivos idealizados foram atingidos apesar de algumas nuances que podiam ter sido mais bem concebidas como o uso de um MQTT broker privado e mais seguro, entre outras.

No futuro podem ser realizadas aplicações para gestão da BD, aplicações para consumir a API concebida assim como outras aplicações que possam ser uteis para os nossos utilizadores.

IV. REFERENCES

[1] Erik Wilde and Cesare Pautasso, “REST: From Research to Practice”, Springer, 2011.

[2] Joe Fawcet, Liam Quin, and Danny Ayers, “Beginning XML”, fifth edition, John Wiley & Sons, Inc, 2012.

[3] Gregor Hohpe and Bobby Woolf, “Enterprise Integration Patterns - Designing, Building And Deploying Messaging Solutions”, Addison Wesley, 2003

APPENDIX

Trabalhos realizados: André Cordeiro – “Bridge”, André Lopes – “Global API” & “BD”, António Barros – “DataStorage” & “DataShowGraphs”, Ricardo Bogalho – “Alerts”, Relatório: Participação de todos os elementos.

Para inicializar a execução do sistema não é necessária qualquer configuração prévia, visto que todas as aplicações foram feitas o mais independente possível. As aplicações bastam ser inicializadas para começarem a executar as suas funcionalidades.