|  |  |  |
| --- | --- | --- |
| Description: logo | **DEI** | **Departamento**  **Engenharia**  **Informática** |
|  | |
| **Integração de Sistemas**  Engenharia Informática  (3º ano, 1º sem - 2015/2016) | | |

***Data Integration Hub***

*Relatório do projeto*

|  |
| --- |
| ***Nome:*** Rúben Santos  ***Número:*** 2130725  ***Turno Prático:*** PL3  ***Docente do Turno Prático:*** Marisa Maximiano |

|  |
| --- |
| ***Nome:*** Joel Francisco  ***Número:*** 2121000  ***Turno Prático:*** PL3  ***Docente do Turno Prático:*** Marisa Maximiano |

|  |
| --- |
| ***Nome:*** Mickael Gomes  ***Número:*** 2130693  ***Turno Prático:*** PL3  ***Docente do Turno Prático:*** Marisa Maximiano |

Índice

[1. Introdução 3](#_Toc440237021)

[2. Módulo 1 – Sensor Data Controller and Alarming Module   3](#_Toc440237022)

[3. Módulo 2 - DBPersistence 4](#_Toc440237023)

[4. Módulo 3 – Service Conector 5](#_Toc440237024)

[5. Requisitos não implementados 7](#_Toc440237025)

[6. Contribuição de cada elemento do grupo 7](#_Toc440237026)

[7. Conclusão 8](#_Toc440237027)

[*Anexo A* – Listagem dos Ficheiro XML utilizados nos vários módulos 9](#_Toc440237028)

[*Anexo B* – Outras informações 11](#_Toc440237029)

##### Introdução

No âmbito da Unidade Curricular de Integração de Sistemas foi-nos solicitado um projeto com o objetivo de pôr em prática os conhecimentos adquiridos ao longo do semestre.

O objetivo do projeto consiste em desenvolver uma aplicação que permitisse ler os dados de sensores (temperatura, humidade e pressão barométrica) que distribua esses mesmos dados por várias aplicações, independentemente do seu tipo. Para essa comunicação é necessário existir um SensorDataController (dados de sensores em ficheiro dll) e Communication HUB (Comunicação entre aplicações) onde este vai receber dados do Sensor Data Controller, Alarm System (gestor de regras do alarme) que permite gerir as regras dos dados recebidos pelo Communication HUB, DBPersistence que recebe os dados do Communication HUB e envia esses dados para a base de dados. Logo de seguida temos um Serviço do tipo SOAP, *Web* API que fornece as informações exigidas pelo GUI cliente (Service Client) onde o utilizador pode monitorar as informações dos sensores de uma forma mais amigável.

##### Módulo 1 – Sensor Data Controller and Alarming Module

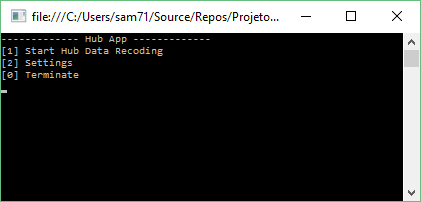
No nosso caso, o Sensor Data Controller e o Communication Hub são uma aplicação única do tipo *console* *application* com vários menus, é *publisher-subscriber*, ou seja publica os dados que o sensor envia para os clientes subscritos, e ao mesmo tempo, também se subscreve ao publisher de Alarmes do Alarm System apenas para os receber. O Hub para o nosso ponto de vista é algo transparente, ou seja a gestão do envio e receção de dados é totalmente gerida pelos componentes da biblioteca ZeroMQ, para além disso, para os subscritores é enviado um objeto serializado em XML que posteriormente, é recriado por quem o consume pelo simples facto de ser rápido e muito fácil de trabalhar, pois temos garantia de que os valores estão nos seus tipos corretos sem necessidade de *parses* adicionais nem *try catch*. No Hub damos a possibilidade alterar o intervalo de tempo em que o sensor envia dados, pois não fazia sentido ter um valor por fixo, dando um maior controlo ao utilizador. As opções de ligação do *publisher*-*subscriber* tem o objetivo de tornar aplicação o mais dinâmica possível.

Figura 1 - Communication Hub

A aplicação de alarmes gere as regras de alarme dos sensores (Temperatura, Humidade e Pressão), numa aplicação em Windows Presentation Foundation (WPF) com interface amigável aos utilizadores na utilização da aplicação como mostra na figura.

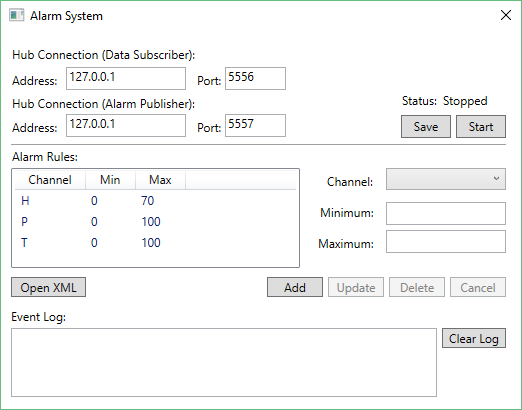


Figura 2-Aplicação Alarm System

Os sensores representam diferentes tipos de canais (T, H ou P) com os seus respetivos valores. A aplicação permite ao utilizador definir regras para cada canal, o valor máximo e mínimo serve para alertar o Communication Hub que o valor do respetivo canal ultrapassa os valores definidos pelo utilizador, mas este por sua vez, vai ainda permitir definir o endereço IP e a porta para o *subscriber* dos dados e *publisher* de alarmes. As regras são guardadas num ficheiro XML para depois serem publicados no Communication Hub informando que os valores ultrapassam os valores definidos nas regras, existe também um ficheiro de validação XSD do XML que está embutido no EXE da aplicação. Ainda vai permitir ao utilizador visualizar o ficheiro XML através do botão “Open XML”. A aplicação, para obter os dados do Communication Hub, apenas é iniciado quando o utilizador clicar o botão *Start*. Em relação à estrutura do XML temos o elemento principal Rules, com o Rule dentro (obrigatórios), dentro tem 3 elementos igualmente obrigatórios, *channel (string), min (unsignedByte) e max (unsignedByte)* .

##### Módulo 2 - DBPersistence

[Explicação da abordagem adotada na implementação de todos as funcionalidades implementadas, esquema da base de dados e justificação do mesmo, apresentação e justificação da estrutura do ficheiro XML proposto para este módulo, etc.]

Mais uma vez, e como no resto de todas as aplicações do projeto, o objetivo foi implementar as funcionalidades pedidas e manter a aplicação o mais customizável possível dando possibilidade de alterar a ligação à base de dados, assim como o endereço do Hub. Como base de dados usamos um *plugin* de MySQL no serviço AppHarbor, usamos esta BD pelo simples motivo de estamos mais à vontade com esta, mas sendo MySQL o único tipo suportado, remotamente ou não. Sendo esta aplicação um *subscriber* do Hub, poderá estar, no máximo, a receber dados do sensor de 1 em 1 segundo, sendo este registado na base de dados imediatamente. Esta inserção faz recurso da Entity Framework, que nos facilita a inserção e as consultas na base de dados usando objetos.//

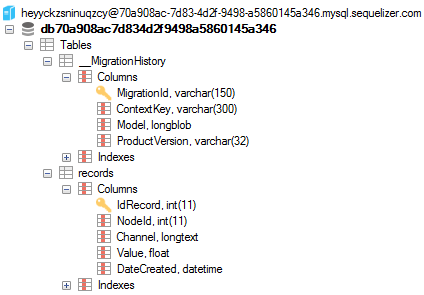


Figura 3 - Constituição da Base de Dados

Como foi pedido neste módulo, todos os dias são gerados/atualizados vários ficheiros de estatisticas. Junto do executabel é possibel encontrar uma pasta com nome “Statistics”, pois a estrutura dentro desta é a seguinte:

* uma pasta para cada ano;
  + Ficheiro xml com o nome **all\_year\_stats** com as médias max, min e avg por cada dia do ano e por cada tipo de channel;

\*year = ano

* por sua vez dentro do ano, um pasta com o nome do mês;
  + Ficheiro xml com o nome **all\_month\_stats** com as médias max, min e avg por cada mês;

\*month = nome dos mês por extenso

* ainda dentro do mês um ficheiro **day\_dayNum\_stats** por dia, com as media por channel

\*dayNum = dia da semana

Esta atualização é feita todos os dias à meia-noite + 1 minuto.

##### Módulo 3 – Service Conector

Neste módulo implementamos um *WEB* *Service* SOAP, para fazer pedidos à base de dados, por sua vez devolver à uma WEB API a comunicação com a GUI do cliente.

Quanto ao *WEB* *Service,* ele tem duas funcionalidades:

1 - GetRecordsByChannelAndDatespan - Permite obter uma listagem dos records mais recentes limitando os resultados por 3 possíveis períodos de tempo predefinidos (Ultimas 24h, Últimos 7 dias, Últimos 30 dias). Esta funcionalidade também permite restringir a pesquisa a um, dois ou aos 3 *channels* e ainda aos resultados. A vantagem desta funcionalidade é que se pode obter uma listagem completa em vez de uma aproximação dos resultados, útil quando um utilizador pretender analisar os dados com mais detalhe.

A desvantagem desta funcionalidade é o grande numero de registos, apesar da restrição dos possíveis períodos de tempo, foi necessário aumentar as constantes que limitam o tamanho da mensagem enviada pelo servidor, visto que o tamanho da mensagem num período superior a 24h, ultrapassava os limites por predefinição. Outra vantagem é a possibilidade de se poder limitar os resultados por um ou mais channels.

2 - GetRecordsByDataRange - Funcionalidade com o objetivo de obter uma listagem estatística dos records que se delimitam por um período de duas datas e por um tipo de função de grupo (Min, Average, Max).

A utilidade desta funcionalidade encontra-se na possibilidade de fazer pedidos estatísticos ao servidor com um intervalo entre a duas datas elevado, sem que haja perda de performance e sem a necessidade do utilizador receber milhares de registos, a funcionalidade analisa o intervalo entre as datas e agrupa os registos da BD em grupos cujo tamanho depende do tamanho do intervalo. Por exemplo, se o intervalo de datas do pedido for superior a 7 dias, os registos são agrupados de 8 em 8 horas, no entanto se o intervalo for superior a 30 dias, os registos são agrupados dia. Depois do agrupamento dos registos, a funcionalidade dá a liberdade ao utilizador de escolher qual o tipo de função de grupo a usar, Min, Average ou Max.

O WebClient é um projeto ASP.Net MVC que usa a framework .Net framework 4.5. Quanto a esta aplicação, os pedidos html, css, e js são carregados pela secção MVC, enquanto que os dados são carregados pela WebAPI, oferecendo ao cliente um melhor conforto no uso da aplicação para o carregamento de dados, foi usada a framework angularJS (sem as funcionalidades de *routing*).

O Website permite fazer download de um Excel com estatísticas (de um channel selecionado ou de todos os channels) limitado pelo intervalo máximo de 1 mês de maneira a evitar o carregamento a dezenas de milhares de registos para o Excel.

O Excel apresenta dois *workbooks*, o primeiro com uma pequena listagem dos registos que se encontram na estatística, e um outro *workbook* com a contagem de registos por channel, min, average e max de cada channel, assim como gráficos correspondentes a cada estatística.

##### Requisitos não implementados

Implementamos todas as funcionalidades pedidas, inclusive alguns extras.

##### Contribuição de cada elemento do grupo

**Módulo 1** (*Hub* e *Data Sensor Controller*) **–** O colega de equipa Joel, juntamente com o Rúben, implementaram os controladores *CommunicationHubController* e *SensorDataController*. A implementação do *Hub* como *Publisher-Subscriber* e os Menus da *console* *application* foram da exclusiva responsabilidade do Rúben.

**Módulo 1.1** (Sistema de Alarmes) **–** Michael - Implementação de formulários e os seus respetivos objetos (botões, *listbox*, *combobox*, etc). Elaboração de elementos e atributos para guardar regras de alarme no ficheiro XML para ser lido através da aplicação e ainda o XSD para a validação do mesmo. Implementação de várias funções para Criar, editar, guardar e apagar regras de cada tipo de canais, receber dados que vêm do *Communication Hub*. Ligação e receção de dados que vêm do CommunicationHub e, ao mesmo tempo, verificar se os valores dos canais estão dentro das regras criadas, publicando os alarmes gerados no Hub. O Rúben neste módulo implementou o *subscriber* dos dados vindos do Hub e o *Publisher* de Alarmes, todo o algoritmo a isso correspondente, assim como os endereços de comunicação de cada um destes.

**Módulo 2** (*DBPercistence*) **–** Este módulo é da responsabilidade do Rúben, implementando ligações dinâmicas, da ligação ao *publisher* Hub e a ligação à base de dados. Os dados vindos do *publisher* são imediatamente registados numa base de dados MySql na *Cloud*. Ainda foi implementado a criação dos ficheiros XML de estatísticas.

**Módulo 3** (Soap Service, WebApi e Service Client) **-** Este módulo é da responsabilidade do Joel, que implementou um serviço Soap com várias funcionalidades para questionar a base dados e devolver a uma *WebApi* de facilitação da comunicação do *Website* (*Service Client*) com o serviço Soap. A página *Web* implementa gráficos e exportação de dados para Excel.

##### Conclusão

Este projeto foi para nós extremamente interessante, apesar da integração de sistemas, de diferentes tipos ser um pouco complicada, mas ao mesmo tempo desafiante, dando para perceber o quanto difícil ade ser integrar sistemas antigos, comparado com o que fizemos neste projeto. Durante do desenvolvimento tivemos vários problemas, com a biblioteca ZeroMQ, mas que acabaram por ser logica mal implementada, um pouco também pelo desconhecimento da biblioteca em questão, na ligação entre a *Web* API e o GUI (*Service cliente*),

Gostava que no futuro ao implementarem sensores como a dll utilizada neste projeto, que fosse realmente implementada do inicio ao fim e com mais funcionalidades, pois causou alguns problemas na nossa implementação do Hub, nomeadamente na funcionalidade *Start Stop*.

Em conclusão, todos os elementos do grupo concordam que foi uma experiencia positiva e que certamente nos vai ser útil num futuro muito próximo. Fica um agradecimento à professora Marisa Maximiano pela ajuda prestada.

##### *Anexo A* – Listagem dos Ficheiro XML utilizados nos vários módulos

[Colocar o XSD e respetivo XML para o(s) diverso(s) ficheiro(s) de XML que foram criados no projeto.]

Modulo 1 – XSD validar regras:

<?xml version="1.0" encoding="utf-8"?>

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="Rules">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Rule">

<xs:complexType>

<xs:sequence>

<xs:element name="channel" type="xs:string" />

<xs:element name="min" type="xs:unsignedByte" />

<xs:element name="max" type="xs:unsignedByte" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Modulo 1 – XML Regras:

<?xml version="1.0" encoding="utf-8"?>

<Rules>

<Rule>

<channel>H</channel>

<min>0</min>

<max>70</max>

</Rule>

<Rule>

<channel>P</channel>

<min>0</min>

<max>100</max>

</Rule>

<Rule>

<channel>T</channel>

<min>0</min>

<max>100</max>

</Rule>

</Rules>

Modulo 2 – XSD Statistics

<?xml version="1.0" encoding="utf-8"?>

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="statistics">

<xs:complexType>

<xs:sequence minOccurs="0">

<xs:element maxOccurs="unbounded" name="statistic">

<xs:complexType>

<xs:attribute name="date" type="xs:string" use="required" />

<xs:attribute name="channel" type="xs:string" use="required" />

<xs:attribute name="avg" type="xs:decimal" use="required" />

<xs:attribute name="min" type="xs:decimal" use="required" />

<xs:attribute name="max" type="xs:decimal" use="required" />

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

##### *Anexo B* – Outras informações

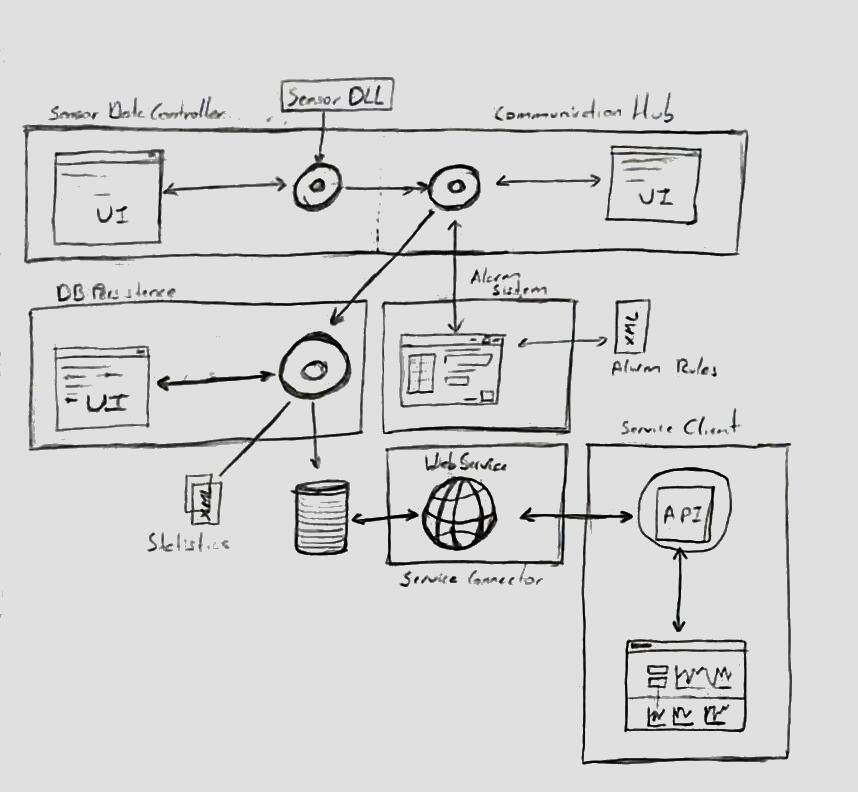


Figura 4 - Esboço Inicial da estrutura da aplicação