|  |  |  |
| --- | --- | --- |
| 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](#_Toc440214653)

[2. Módulo 1 – Sensor Data Controller and Alarming Module   3](#_Toc440214654)

[3. Módulo 2 - DBPersistence 4](#_Toc440214655)

[4. Módulo 3 – Service Conector 4](#_Toc440214656)

[5. Requisitos não implementados 5](#_Toc440214657)

[6. Contribuição de cada elemento do grupo 5](#_Toc440214658)

[7. Conclusão 5](#_Toc440214659)

[*Anexo A* – Listagem dos Ficheiro XML utilizados nos vários módulos 7](#_Toc440214660)

[*Anexo B* – Outras informações 7](#_Toc440214661)

##### 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 SensorDataController, Alarming 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

[Descrição das funcionalidades principais. Análise das diversas componentes do módulo d desenvolver. Explicar detalhadamente a abordagem seguida na implementação do módulo SensorDataController, implementação do HUB e do módulo de Alarme. Apresentação e justificação da estrutura adotada no ficheiro de configurações em XML. Possíveis subsecções sugeridas: Interface, Explicar Configurações; Funcionalidades; Extras, etc.]

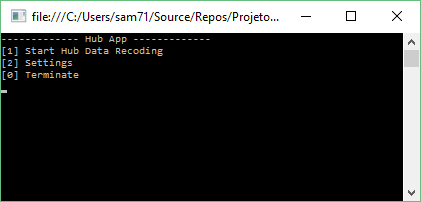
No nosso caso, o Sensor Data Controller e o Communication Hub são uma aplicação única do tipo *console* *application* com vários menus, é *publisher-subscriber*, ou seja publica os dados que o sensor envia para os clientes subscritos, e ao mesmo tempo, também se subscreve ao publisher de Alarmes do Alarm 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) para ser mais fácil na compreensão dos utilizadores na utilização da aplicação como mostra na figura.

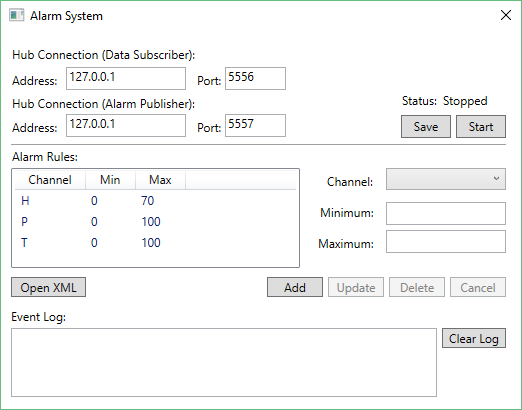


Figura 2-Aplicação Alarm System

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* e *publisher* de alarmes. As regras são guardadas num ficheiro XML para depois serem publicados à 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.]

O serviço DBPersistence permite ler os dados, em formato XML, armazenado no Communication HUB para serem analisados e enviar à base de dados.

##### Módulo 3 – Service Conector

[Explicação de todos as funcionalidades implementadas neste módulo ao nível da camada de serviço e da aplicação cliente, com justificação das mesmas.]

O serviço utilizado no Service Conector para aplicar em WebServices é SOAP, que é um protocolo de transferência de mensagens em formato XML para uso em aplicações. Uma das maiores vantagens disso é que várias linguagens e ferramentas conseguem ler e gerar mensagens facilmente, pois essas ferramentas corporativas tiram vantagem do padrão e possibilitam filtrar, enfileiramento, classificar e redirecionar as mensagens trocadas entre sistemas.

##### Requisitos não implementados

[Lista de todos as funcionalidades não implementadas, ou implementadas apenas parcialmente para cada módulo].

##### 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 *Communication HUB* 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 ainda 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 acabavam sempre por ser logica mal implementada, um pouco também pelo desconhecimento da biblioteca em questão.

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, com a 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.]

##### *Anexo B* – Outras informações

[Logins e passwords necessárias para testar o trabalho. Devem também indicar todos os ficheiros de configuração que é necessário alterar para executar o trabalho corretamente.

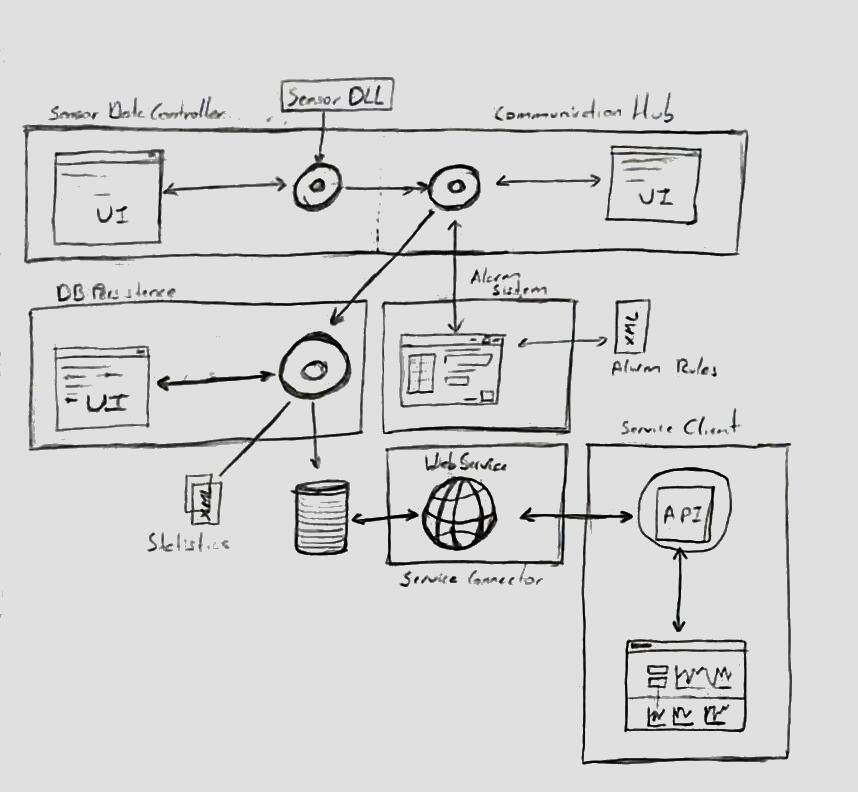


Figura 3 - Esboço Inicial da estrutura da aplicação