FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Introdução de máquinas de deformação de metal no mundo da Internet das Coisas

Tiago Manuel de Amaral Lopes

Mestrado em Engenharia Eletrotécnica e de Computadores

Orientador: Mário Jorge Rodrigues de Sousa

14 de outubro de 2023

Resumo

Este documento ilustra a dissertação desenvolvida no âmbito do Mestrado em Engenharia Eletrotécnica e de Computadores, da Faculdade de Engenharia da Universidade do Porto. O trabalho proposto consiste em desenvolver um sistema IoT para supervisionar máquinas quinadoras, monitorizando alarmes, eventos e os valores dos sensores, apresentando os mesmos num *dashboard* acessível através da internet. Neste mesmo, apresentar um parâmetro de manutenção preditiva de maneira a anteceder avarias e prevenir futuras falhas nos ativos.

O sistema desenvolvido neste trabalho é composto por três componentes, as máquinas que constituem o dispositivo inteligente, a aplicação IoT e uma interface gráfica.

Para simular uma máquina e ter poder de controlar todos os valores, foi criado um servidor OPC UA em Node-RED.

A aplicação IoT consiste nos programas, *softwares* e serviços *cloud* usados. Foi criada uma aplicação em Windows Forms para configurar parâmetros de entrada para o programa principal, que consiste num cliente OPC UA e um mensageiro para a *cloud*, que corre num *container*. Os serviços Azure utilizados neste trabalho foram Azure IoT Hub, Azure Stream Analytics e para base de dados Azure Data Lake Storage Gen2.

A seguir, os dados da *cloud* são encaminhados para o Power BI para popular o *dashboard* com a telemetria pretendia a monitorizar, bem como o tempo que falta para ser realizada a manutenção do equipamento. Agregando a este, também foi desenvolvido um segundo *dashboard* para controlar e visualizar o número de vendas e os seus clientes.

A manutenção preditiva e a monitorização proativa das máquinas traduz-se num aumento da produtividade e previne custos acrescidos em avarias, segurança, falta de qualidade do produto final, entre outros.

Palavras-chave: IoT, Cloud, Indústria 4.0, Internet das Coisas

Abstract

This document describes the dissertation developed within the scope of the Master in Electrical and Computer Engineering, Faculty of Engineering, University of Porto. The proposed work aims to develop an IoT system to supervise metal forming machines, monitoring alarms, events and sensor values, and presenting them in a dashboard accessible through the internet. In this same, present a predictive maintenance parameter in order to anticipate breakdowns and prevent future failures in assets.

The system developed in this work consists of three components, the machines that constitute the smart device, the IoT application and a graphical interface.

To simulate a machine and have the possibility to control all values, an OPC UA server was created in Node-RED.

The IoT application is composed of the programs, software and cloud services involved. A Windows Forms application was created to configure input parameters for the main program, which consists of an OPC UA client and a cloud messenger, which runs in a container. The Azure services used in this work were Azure IoT Hub, Azure Stream Analytics and for database Azure Data Lake Storage Gen2.

Afterwards, the data from the cloud is forwarded to Power BI to populate the dashboard with the telemetry intended to be monitored, as well as the time remaining for equipment maintenance. Furthermore, a second dashboard was also developed to control and visualize the number of sales and their customers.

Predictive maintenance and proactive monitoring of machines translates into increased productivity and prevents additional costs in breakdowns, safety, lack of quality of the final product, among others.

Keywords: IoT, Cloud, Industry 4.0, Internet of Things

Agradecimentos

Primeiramente gostaria de agradecer à Faculdade de Engenharia da Universidade do Porto por todo o conhecimento transmitido, das oportunidades geradas para desenvolver *soft skills*, de todos os bons e menos bons momentos que me proporcionaram destes últimos anos, fazendo de mim um trabalhador capaz de enfrentar qualquer desafio.

Ao meu orientador e professor Mário Jorge Rodrigues de Sousa por toda a sua mentoria no decorrer desta dissertação.

À ADIRA Metal Forming Solutions, em especial ao Jorge Borges Almeida, Fernando Bessa Neto, Luís Belinha Reis e Nuno Manuel Sousa que propuseram, acolheram e me guiaram durante todo o desenvolvimento deste trabalho.

Por fim, aos meus pais, avós, namorada e amigos que sem eles nada disto seria possível, devido ao constante, incansável e incondicional apoio, força, incentivo e conselhos durante estes anos, fazendo de mim a pessoa que sou hoje.

A todos um grande Obrigado,

Tiago Manuel de Amaral Lopes

'A person who never made a mistake never tried anything new."

Albert Einstein

Conteúdo

1 Introdução **1**
1.1 Enquadramento e objetivos . . . 1
1.2 Estrutura do documento . . . 1

2 Revisão Bibliográfica **3**
2.1 Indústria 4.0 . . . 3
2.2 IoT - Internet das Coisas . . . 5
2.3 Protocolos IoT . . . 6
2.3.1 HTTP - *Hyper Text Transfer Protocol* . . . 6
2.3.2 AMQP - *Advanced Message Queuing Protocol* . . . 7
2.3.3 MQTT - *Message Queuing Telemetry Transport* . . . 7
2.3.4 OPC UA - *Open Platform Communication Unified Architecture* . . . 8
2.4 *Containers* . . . 8
2.4.1 *Containers* vs. Máquinas virtuais(VMs) . . . 9
2.5 Computação na *Cloud* . . . 10
2.5.1 Tipos de *Cloud* . . . 10
2.5.2 Modelo de serviços . . . 11
2.5.3 Vantagens da computação na *cloud* . . . 11
2.6 Microsoft . . . 12
2.6.1 Azure IoT Hub . . . 12
2.6.2 Azure Stream Analytics . . . 13
2.6.3 Azure Data Lake Storage Gen2 . . . 13
2.6.4 Power BI . . . 14
2.7 Amazon . . . 15
2.7.1 AWS IoT Core . . . 15
2.7.2 AWS IoT Analytics . . . 16
2.7.3 Amazon S3 . . . 17
2.7.4 Amazon Quicksight . . . 17
2.8 Node-RED . . . 18

3 Abordagem **21**
3.1 Requisitos . . . 21
3.2 Plataforma/serviços IoT estudados . . . 22
3.2.1 Solução A - Microsoft Azure Cloud + PowerBI . . . 23
3.2.2 Solução B - Amazon AWS Cloud . . . 23
3.2.3 Solução C - Google Cloud . . . 24
3.2.4 Solução D - Thingsboard *Self Managed* + VM Azure . . . 24
3.2.5 Solução E - Thingsboard *Self Managed* + VM AWS . . . 24

3.2.6 Solução implementada 24
3.3 Arquitetura 25

4 Desenvolvimento **27**
4.1 Aplicação de configuração 27
4.2 Servidor OPC UA 31
4.3 Programa principal 32
4.4 Dados na *cloud* Azure 34
4.4.1 *Resource Group* 35
4.4.2 *IoT Hub* 35
4.4.3 *Stream Analytics* 36
4.4.4 *Storage Account* 38
4.5 *Dashboard* 39
4.5.1 *Dashboard* de telemetria 39
4.5.2 *Dashboard* de vendas 41

5 Conclusão **45**
5.1 Testes e resultados 45
5.2 Recomendações e Trabalho Futuro 46
5.3 Conclusão 47

A Estimativas de custo dos serviços *Cloud* **49**
A.1 Solução A - Microsoft Azure Cloud + PowerBI 49
A.2 Solução B - Amazon AWS Cloud 50
A.3 Solução D - Thingsboard *Self Managed* + VM Azure 50
A.4 Solução E - Thingsboard *Self Managed* + VM AWS 51

B Código servidor OPC UA **53**

C Ficheiro de telemetria armazenada **59**

D Ficheiro de dados de clientes **63**

Referências **65**

Lista de Figuras

2.1 Revoluções industriais [1] . . . 4
2.2 Rede IoT [2] . . . 5
2.3 Comunicação HTTP . . . 7
2.4 Comunicação HTTP . . . 7
2.5 Interações OPC UA em diferentes níveis [3] . . . 8
2.6 Arquitetura de uma Máquina Virtual vs. Arquitetura de um *Containers* [4] . . . 10
2.7 Diagrama de utilização do IoT Hub [5] . . . 12
2.8 Diagrama de utilização do Stream Analytics [5] . . . 13
2.9 Exemplo de entradas e uma possível saída no Power BI [6] . . . 14
2.10 Tipos de visualização de dados . . . 15
2.11 Diagrama de utilização do IoT Core [7] . . . 16
2.12 Diagrama de utilização do IoT Analytics [8] . . . 17
2.13 *Dashboard* em Amazon Quicksight [9] . . . 17
2.14 Fluxo em blocos Node-RED [10] . . . 18
2.15 Nós padrão do Node-RED [11] . . . 19

3.1 Maiores fornecedores de serviços *cloud* no Q1 2023 [12] . . . 22
3.2 Planos *self-managed* ThingsBoard [13] . . . 23
3.3 Arquitetura da solução . . . 25

4.1 Diagrama de atividade da aplicação de configuração . . . 28
4.2 Janela de selecionar o tipo de máquina . . . 28
4.3 Janela de configuração . . . 29
4.4 Exemplos de mensagens de saída da aplicação . . . 29
4.5 Script exemplo *CreateContainer.bat* . . . 30
4.6 Script exemplo *StartContainer.bat* . . . 30
4.7 Simulador do servidor OPC UA em Node-RED . . . 31
4.8 Tipo de nós utilizados . . . 31
4.9 Prosys cliente OPC UA . . . 32
4.10 Fluxo de dados na *Cloud* . . . 35
4.11 *Connection String* de um dispositivo [14] . . . 35
4.12 *Connection String* de um dispositivo [14] . . . 36
4.13 *Job* de enviar dados para o Power BI . . . 37
4.14 *Job* de enviar dados para a base de dados . . . 37
4.15 Diagrama de condições para armazenamento de alarmes na base de dados . . . 38
4.16 Ativar Data Lake Storage Gen2 . . . 38
4.17 UI do portal Azure dos ficheiros armazenados . . . 39
4.18 *Dashboard* de telemetria . . . 40

4.19 *Dashboard* de telemetria com filtro por ID e registo da tabela de alarmes 41
4.20 *Dashboard* de vendas 42
4.21 *Dashboard* de vendas com filtro de país 43

Lista de Tabelas

3.1 Tabela de requisitos. 21
3.2 Estimativa de cada Solução 24

5.1 Tabela de requisitos e resultado dos mesmos. 46

A.1 Estimativa da Solução A para 2 dispositivos 49
A.2 Estimativa da Solução A para 5 dspositivos 49
A.3 Estimativa da Solução A para 25 dispositivos 49
A.4 Estimativa da Solução B para 2 dispositivos 50
A.5 Estimativa da Solução B para 5 dispositivos 50
A.6 Estimativa da Solução B para 25 dispositivos 50
A.7 Estimativa da Solução D para até 10 dispositivos 50
A.8 Estimativa da Solução D para até 100 dispositivos 51
A.9 Estimativa da Solução D para até 500 dispositivos 51
A.10 Estimativa da Solução D dependento da subscrição 51
A.11 Estimativa da Solução E para até 10 dispositivos 51
A.12 Estimativa da Solução E para até 100 dispositivos 51
A.13 Estimativa da Solução E para até 500 dispositivos 52
A.14 Estimativa da Solução E dependento da subscrição 52

Abreviaturas e Símbolos

ADLS	Azure Data Lake Storage Gen2
ADT	Abstract Data Type
AWS	Amazon Web Service
C2D	*Cloud* para dispositivo
D2C	Dispositivo para *cloud*
DAX	Data Analysis Expressions
ERP	Enterprise Resour Planning
ERP	Sistemas de Gestão Empresariais
IaaS	Infraestrutura como serviço
IoT	Internet of Things
IT	Tecnologia de informação
kB	Kilobytes
M2M	Machine to Machine
MQTT	Message Queuing Telemetry Transport
OPC UA	Open Platform Communications Unified Architecture
OS	Sistema Operativo
PaaS	Plataforma como serviço
PLC	Controladores Lógicos Programáveis
SaaS	*Software* como serviço
SASL	Simple Authentication and Security Layer
SDKs	*Kits* de desenvolvimento de *software*
SQL	Structed Query Language
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TSL	Transport Layer Security
UI	User Interface
VM	Máquina Virtual
VMs	Máquinas Virtuais

Capítulo 1

Introdução

Na era da quarta revolução industrial, a Internet das Coisas (IoT) surge como uma tecnologia indispensável, revolucionando a maneira como as indústrias operam. Máquinas e sistemas formam uma rede inteligente proporcionando inúmeros benefícios à produção e processos.

A IoT desempenha uma função fundamental na melhoria de eficiência e produtividade na indústria. A interconexão de dispositivos, sensores e sistemas, permite coletar e compartilhar informação em tempo real permitindo a monitorização continua dos processos industriais. Outra área importante é a de implementação de planos de manutenção preditiva, que através da informação recolhida é possível monitorizar proativamente prevenindo ou prevendo futuras falhas [15].

1.1 Enquadramento e objetivos

A proposta desta dissertação foi realizada pela ADIRA Metal Forming Solutions, S.A., fundada em 1956 e atualmente sediada em Canelas, Vila Nova de Gaia. A ADIRA é um grande fabricante português e fornecedor global de soluções processamento de chapas de metal. Especializada no desenvolvimento e produção de máquinas de corte a laser, quinadoras, guilhotinas e células robotizadas. [16]

Este trabalho visa introduzir máquinas quinadoras no mundo da internet das coisas, implementando um sistema IoT capaz de supervisionar os seus sensores, alarmes e eventos via web como também obter um parâmetro de manutenção preditiva. Com isto pode-se evitar futuras possíveis avarias e precaver a falha dos ativos.

1.2 Estrutura do documento

O presente documento está estruturado, para além do atual capítulo 1 de introdução, por mais 4 capítulos, tendo um total de 5 capítulos.

No capítulo 2 é efetuada uma breve revisão bibliográfica que abrange pontos fundamentais que suportam este trabalho.

A seguir no capítulo 3 são referidos os requisitos deste trabalho bem como as soluções estudadas seguidamente da arquitetura implementada.

O capítulo 4 integra o trabalho desenvolvido, ou seja, servidor OPC UA, aplicação de configuração, programa principal, os serviços *cloud* utilizados, bem como os *dashboards* criados.

Por fim, no capítulo 5, estão descritos os testes e resultados realizados/obtidos, recomendações e trabalho futuro e por último a conclusão.

Capítulo 2

Revisão Bibliográfica

Este capítulo permite salientar tópicos relevantes para qualquer leitor adquirir o conhecimento básico e necessário no domínio desta dissertação. Os temas aludidos foram o que é a indústria 4.0, internet das coisas, alguns dos protocolos usados no IoT e os seus modelos, o que são *containers* e a comparação com as máquinas virtuais e o que é a computação na *cloud*.

Também é dado uma visão superficial sobre o que é a tecnologia Node-Red e quais são os serviços com maior destaque, para este trabalho, dos dois maiores fornecedores de *cloud*, Microsoft Azure e Amazon AWS.

2.1 Indústria 4.0

Com o passar dos anos, as pessoas sempre tentaram utilizar as tecnologias disponíveis para auxiliar a tornas as suas vidas mais acessíveis e as elevar a outro patamar [17] e com isto também a indústria passou por importantes e profundas mudanças [18]. Na figura 2.1 é possível observar as maiores revoluções industriais até à presente data, por ordem cronológica, bem como as tecnologias mais relevantes implementadas nessas transformações.

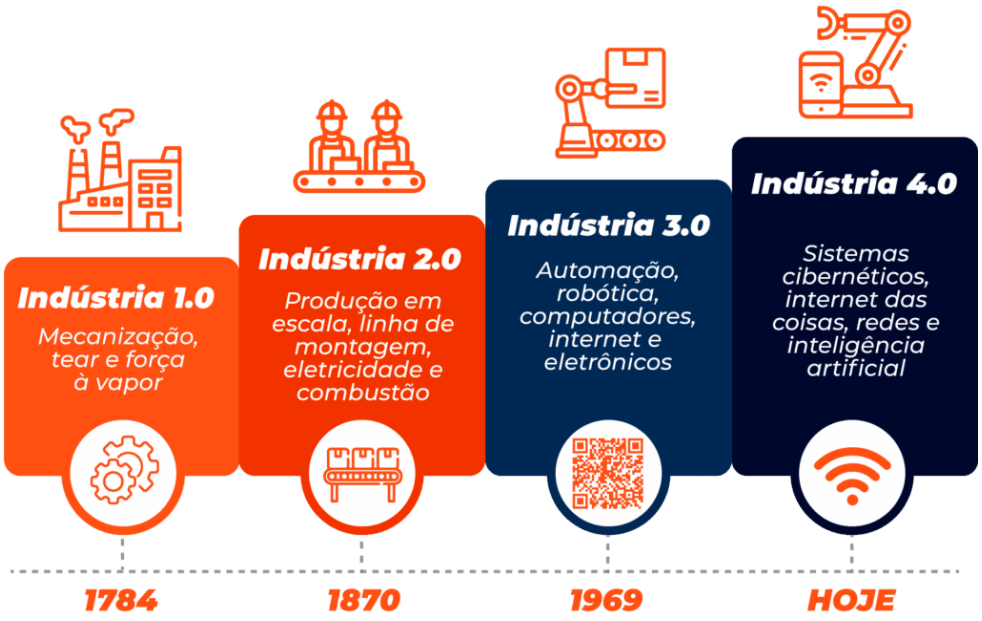


Figura 2.1: Revoluções industriais [1]

A primeira revolução industrial surge no final do século XVIII na Grã-Bretanha e esta ajudou a viabilizar a produção em massa, utilizando a energia obtida da água e do vapor, em vez da força exclusivamente humana e animal.

Um século depois foram introduzidas linhas de montagem, o uso do petróleo, gás e energia elétrica, assim como comunicações por telégrafo e telefone, dando assim início à segunda revolução industrial.

A terceira revolução começou em meados do século XX, acrescentando aos processos computadores, telecomunicações avançadas e análise de dados. A digitalização das fábricas começou com a integração de controladores lógicos programáveis (PLCs) nas máquinas de modo a ajudar a recolher dados e automatizar processos.

Estamos atualmente na quarta revolução industrial, também designada por Indústria 4.0. Caracterizada pela crescente automatização e pelo aparecimento de máquinas e fábricas inteligentes, os dados recolhidos ajudam a produzir produtos de forma mais eficiente e produtiva em toda a cadeia de valor. A flexibilidade é melhorada para os fabricantes poderem satisfazer melhor as exigências dos clientes. Ao recolher mais dados do chão de fábrica e ao combiná-los com outros dados operacionais da empresa, uma fábrica inteligente pode obter transparência de informação e tomar melhores decisões [19].

Os fabricantes estão a integrar novas tecnologias, tais como a internet das coisas (IoT), computação na *cloud* e análise de dados, além de inteligência artificial e *machine learning*. Para isto estas fábricas inteligentes são equipadas com sensores, softwares e robótica que coletam e analisam os dados. Obtém-se ainda mais valor quando a informação das operações de produção são combinadas com os dados operacionais dos sistemas de gestão empresariais (ERP), *supply chain*,

atendimento ao cliente e outros sistemas corporativos para serem criadas visibilidade e perceção a partir de dados anteriormente isolados [19].

O termo indústria 4.0 surgiu de um programa de estratégia de alta tecnologia do governo alemão em 2011, não se tendo popularizado até o Fórum Económico Mundial o adotar em 2016. Em comparação com as suas predecessoras, esta revolução está a evoluir num ritmo exponencial [20].

2.2 IoT - Internet das Coisas

A internet das coisas, como referido anteriormente, é umas das tecnologias pertencentes à quarta revolução industrial. IoT descreve a rede de objetos físicos - ”coisas” - incorporados com sensores, *software* e outras tecnologias visando ligar e trocar dados com outros dispositivos e sistemas através da Internet. Estes dispositivos vão desde objetos domésticos comuns a ferramentas industriais sofisticadas [21], figura 2.2.

Figura 2.2: Rede IoT [2]

Um sistema típico de IoT funciona por meio da recolha e troca de informação em tempo real, tendo como seus componentes os dispositivos inteligentes, aplicação IoT (conjunto de serviços e *softwares*) e uma interface gráfica para o utilizador [22].

O IoT permite melhorar a eficiência e produtividade em todas a áreas. Consegue alertar potenciais falhas, evitando paragens inesperadas aumentando os níveis de disponibilidade, fornecer dados relevantes e uma visão global dos processos para auxiliar na tomada de decisão, reduzir os riscos de acidentes e maior controlo da produção.

Para além de controlo da produção, a internet das coisas também tem um papel importante na gestão de equipas como, por exemplo, no rastreamento da localização de funcionários, controlo de horas extras, comunicação, alertas e notificações em caso de imprevistos e emergências e no feedback e avaliação de desempenho [15].

2.3 Protocolos IoT

Um protocolo é um conjunto de regras, normas e padrões, que permitem padronizar a forma de como é realizada a troca de informação entre dispositivos, criando uma linguagem única e compreensível entre os elementos da rede. Sem os protocolos tornar-se-ia uma tarefa bastante crítica integrar soluções de fabricantes e desenvolvedores diferentes. Alguns dos protocolos IoT, da camada de aplicação segundo o modelo OSI e modelo TCP/IP, utilizados para comunicar numa rede utilizando internet são HTTP, AMQP, MQTT e OPC UA. [23] Protocolos esses que serão apresentados de forma introdutória nas páginas 6-8.

Os modelos mais famosos para a comunicação em rede destes protocolos são cliente/servidor e publicar/subscrever.

Cliente/Servidor

No modelo cliente/servidor é composto por duas partes lógicas, o cliente que faz pedidos ao servidor para obter serviços ou informação para finalidade própria e o servidor que providencia esses serviços e informação. [24] Também conhecido como modelo pedido/resposta.

Publicar/Subscrever

O princípio do modelo de comunicação publicar/subscrever é que componentes que tenham interesse em consumir determinada informação que registem o seu interesse. Este processo de demonstrar interesse chama-se subscrição e quem o faz é o subscritor. Os componentes que publicam a informação são os publicadores. A entidade coordena as subscrições e que garante que os dados publicados chegam aos subscritores é o *broker*. [25]. Neste modelo as mensagens são publicadas para um endereço denominado tópico. Os clientes podem subscrever múltiplos tópicos e receber todas as mensagens que para este são enviadas. [26]

2.3.1 HTTP - *Hyper Text Transfer Protocol*

HTTP é o protocolo de mensagens predominante utilizado na *World Wide Web* desenvolvido por IETF e W3C, introduzido como padrão em 1997. É baseado em texto e não tem limite de tamanho de cabeçalho e *payload* [27]. A informação é transportada sobre a camada de transporte TCP e para segurança usa TLS/SSL. O tipo de comunicação do protocolo HTTP é do modelo cliente/servidor, em que usa a arquitetura de mensagem é do tipo pedido/resposta, figura 2.3. Este protocolo pode transferir largos montantes de dados, que mesmo podendo ser divididos em pequenos pacotes pode causar sobrecarga e problemas de largura de banda na rede. [26]

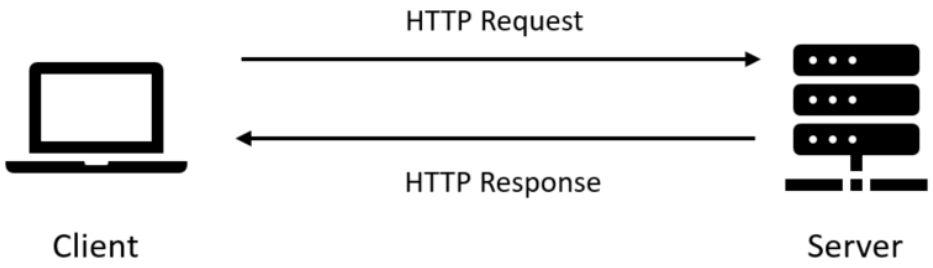


Figura 2.3: Comunicação HTTP

2.3.2 AMQP - *Advanced Message Queuing Protocol*

AMQP é um protocolo máquina para máquina (M2M) que suporta as duas arquiteturas publicar/subscrever e pedido/resposta, desenvolvido por JPMorgan Chase e introduzido em 2003. [28] Os pacotes AMQP contêm um cabeçalho fixo de 8 bytes e o tamanho do *payload* depende da tecnologia usada ou das capacidades do *broker*. É um protocolo binário e corre sobre a camada de transporte TCP e usa TLS/SSL e SASL como segurança. [27]

2.3.3 MQTT - *Message Queuing Telemetry Transport*

MQTT é um dos protocolos de comunicação M2M mais antigos, o qual foi introduzido em 1999. A sua arquitetura de mensagens é do tipo publicar/subscrever, figura 2.4, e concebido para comunicações M2M leves e para redes limitadas.

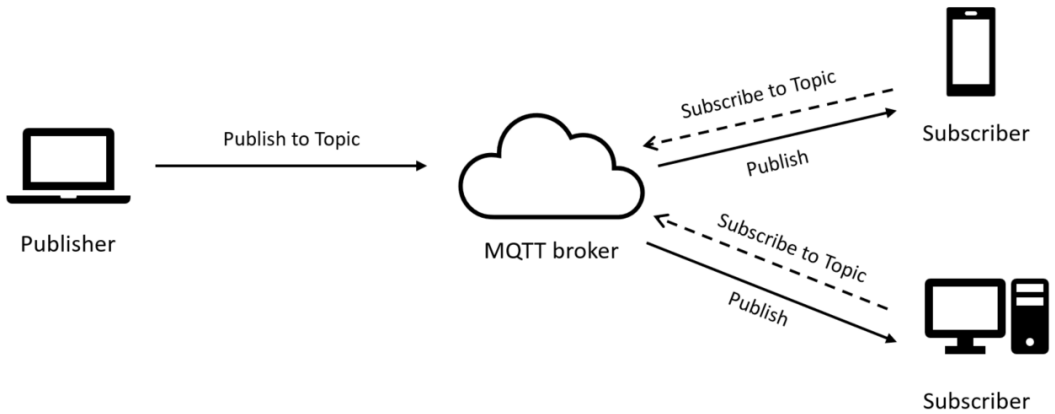


Figura 2.4: Comunicação HTTP

MQTT é um protocolo binário e corre sobre o protocolo de transporte TCP e para segurança TLS/SSL. O pacote da sua mensagem contem um cabeçalho fixo de 2 bytes e um *payload* máximo de 256Mb.

MQTT tem três níveis de qualidade de serviço para entregar as mensagens:

- Enviar exatamente uma vez e sem garantir a entrega;
- Enviar no mínimo uma vez com possibilidade de entrega;
- Exatamente um envio, mas com garantia que a mensagem foi recebida[27].

2.3.4 OPC UA - *Open Platform Communication Unified Architecture*

OPC UA é um protocolo de comunicação industrial, usualmente usado no ramo da automação, desenvolvido pela OPC Foundation e normalizado na IEC-62541 [29]. Desenvolvido para troca segura e confiável de dados e independente da plataforma e garante o fluxo contínuo de informação entre dispositivos de vários fornecedores e sistemas de supervisão [30], figura 2.5.

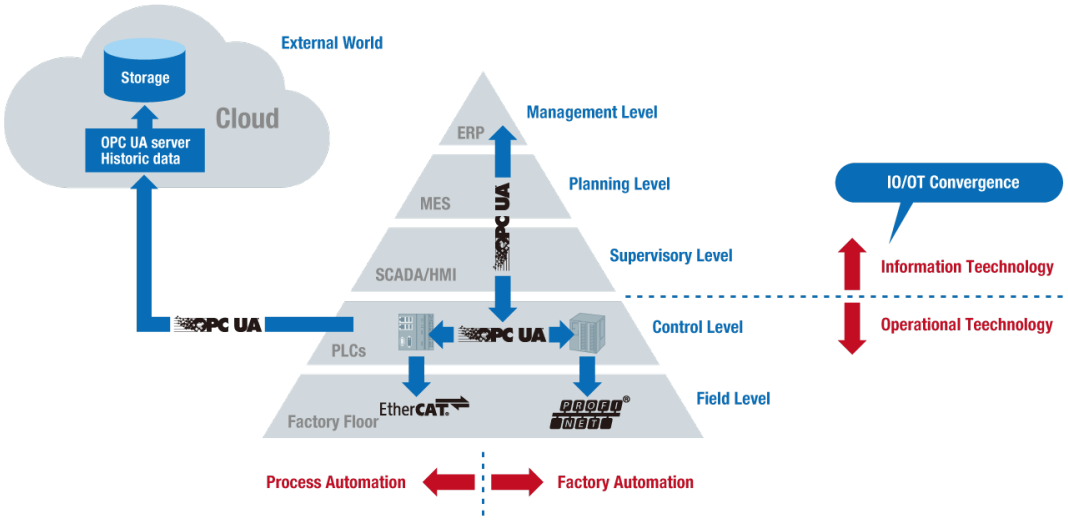


Figura 2.5: Interações OPC UA em diferentes níveis [3]

É um protocolo de arquitetura cliente/servidor. O servidor fornece acesso aos dados estruturados num modelo orientado a objetos com os quais os clientes podem interagir por meio de serviços padronizados. [31] Estes objetos chamam-se ”nós” sendo identificados por um *nodeid* e um nome.

Como referido anteriormente, as interações entre cliente OPC UA e servidor são padronizadas por meio de um conjunto de serviços/métodos fornecidos pelo servidor. Estes serviços permitem gerir o nó e o modelo de informação, ler e gravar dados nos nós e estabelecer canais de comunicação [29].

2.4 *Containers*

Containerization é um dos últimos desenvolvimentos na evolução da computação na *cloud*. Muitas organizações, grandes e pequenas, olham para os *containers* como um meio de melhorar

a gestão do ciclo de vida das aplicações através de capacidades como a integração contínua e a entrega contínua. Além disso, determinadas implementações de *containers* estão em concordância com os princípios de *open-source*. [32]

Os *containers* são uma solução para o problema de como fazer com que um *software* funcione de forma rápida e fiável quando transportado de um ambiente computacional para outro. Isto pode ser de um computador de um programador para um ambiente de testes, de um ambiente de simulação para a produção ou de uma máquina física num centro de dados para uma máquina virtual numa *cloud* privada ou pública. Um *container* consiste num ambiente de execução na sua totalidade: uma aplicação, todas as suas dependências, bibliotecas, ficheiros binários, e ficheiros de configuração necessários para a sua execução, agrupados num único pacote. Ao agrupar a plataforma da aplicação e as suas dependências, as diferenças nas distribuições de sistema operativo (OS) e na infraestrutura(*hardware*, rede e armazenamento) subjacente são abstraídas. [33]

2.4.1 *Containers* vs. Máquinas virtuais(VMs)

Uma forma de compreender melhor o conceito de *container* é compreender como este se difere do método mais tradicional de virtualização, a máquina virtual (VM).

Numa VM, quer seja local ou na *cloud*, uma camada de *software* designada *hypervisor*, virtualiza o *hardware*, permitindo que vários OS funcionem em simultâneo, partilhando os mesmo recursos físicos. Cada VM contém então um OS *guest*("secundário"), uma cópia virtual do hardware que o OS requer para correr, juntamente com a aplicação e as suas bibliotecas e dependências associadas.

Em contraste, os *containers* virtualizam o sistema operativo(geralmente Linux ou Windows) aproveitando as características e recursos do OS *host*("principal") ou do *kernel* do sistema fazendo com que não seja incluído um OS *guest* em cada instância, tornando os *containers*: [34]

- **Leves** — Podendo ter apenas dezenas de megabytes de tamanho, enquanto uma máquina virtual ultrapassa com facilidade vários gigabytes;
- **Rápidos** — São iniciados quase instantaneamente na ordem dos milissegundos, enquanto uma máquina virtual pode demorar minutos a arrancar com todo o OS e executar as aplicações;
- **Portáteis** — As aplicações em *containers* são facilmente implementadas em múltiplos sistemas operativos e diferentes plataformas de *hardware*;
- **Eficazes** — Rapidamente implementados, remendados e expandidos;
- **Melhoradores de desenvolvimento de aplicações** — Apoiam as metodologias ágeis e práticas *DevOps* para acelerar os ciclos de desenvolvimento, teste e produção.

Na figura 2.6 da página 10 é representada a arquitetura de uma VM e a arquitetura de um *container* conforme as características citadas anteriormente.

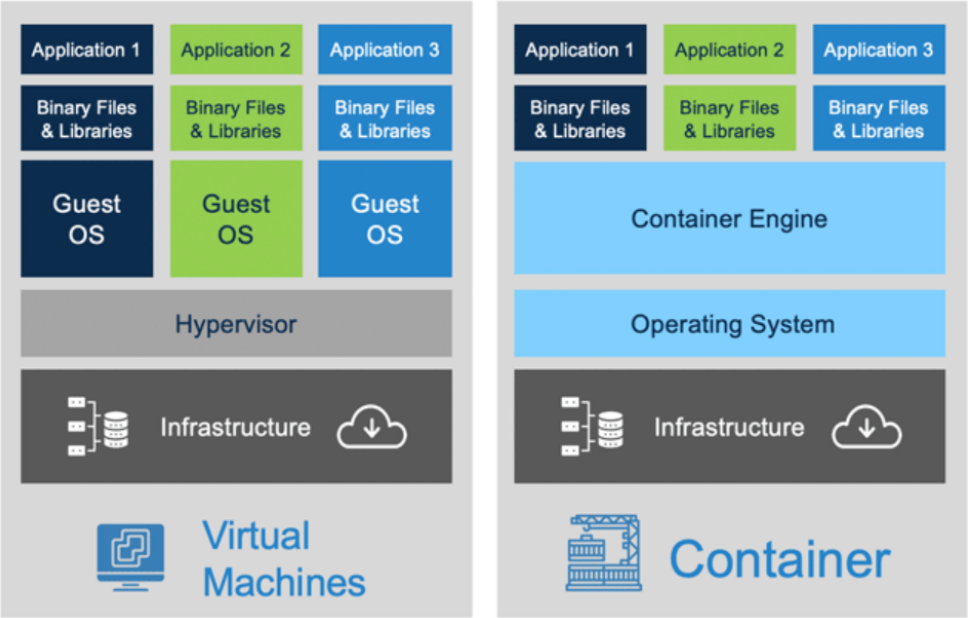


Figura 2.6: Arquitetura de uma Máquina Virtual vs. Arquitetura de um *Containers* [4]

2.5 Computação na *Cloud*

A computação na *Cloud* é a entrega de recursos de tecnologia de informação (IT) sob demanda por meio da Internet com um custo conforme o uso. Em vez de se comprar, ter e manter *datacenters* e servidores físicos, é possível contratar e aceder a serviços de tecnologia, capacidade computacional, armazenamento e base de dados, conforme a necessidade, usando um fornecedor *cloud* [35]. Desta forma, a computação em nuvem permite que o utilizador os acesse por meio de qualquer computador, tablet ou telemóvel.

Exemplos comuns de computação na *cloud* é aceder ou editar um ficheiro no Google Docs, ouvir música no Spotify ou assistir um filme na Netflix.

2.5.1 Tipos de *Cloud*

Existem três tipos de *cloud*, a pública, privada e híbrida.

2.5.1.1 *Cloud* pública

A *cloud* pública é fornecida por terceiros e disponibiliza para qualquer pessoa ou empresa, que deseje contratar, os seus recursos computacionais, serviços e armazenamento. Nesta, tudo está disponível na web e compartilhado entre vários utilizadores que o usam de maneira simultânea, mas separadamente. Este tipo de *cloud* é a mais barata.

2.5.1.2 *Cloud* privada

Neste modelo, a empresa ou pessoa mantém a infraestrutura da *cloud* no seu domínio interno, oferecendo acesso restrito aos utilizadores. É dada a possibilidade de personalizar as funções e suporte às necessidades do cliente, por ser projetada exclusivamente para o mesmo. De modo geral, estas *clouds* são usadas em organizações que necessitam seguir regulamentos e regras rigorosas sobre segurança e privacidade de informação, como o caso de instituições governamentais e financeiras.

2.5.1.3 *Cloud* híbrida

A *cloud* híbrida é a união dos dois anteriores, alguns recursos são usados na nuvem privada, outros publicamente ou então interligados entre si por meio de tecnologias [36].

2.5.2 Modelo de serviços

Para servir as necessidades de diversos clientes, a computação na *cloud* oferece três modelos de serviços, infraestrutura como serviço, plataforma como serviço e *software* como serviço.

2.5.2.1 IaaS - Infraestrutura como serviço

O IaaS contém os componentes básicos do IT na *cloud*. Normalmente, oferece acesso a recursos de rede, máquinas virtuais ou *hardware* dedicado e espaço de armazenamento de dados. IaaS oferece o nível mais elevado de flexibilidade e gestão sobre os recursos contratados.

2.5.2.2 PaaS - Plataforma como serviço

Com PaaS, não há necessidade de gerir a infraestrutura, normalmente *hardware* e sistemas operativos, não havendo preocupação com aquisição de recursos, planear capacidade e manutenção de *software*, mantendo assim o foco na implementação e gestão da aplicação.

2.5.2.3 SaaS - *Software* como serviço

O SaaS oferece um produto completo, executado e gerido pelo fornecedor dos serviços, ou seja, aplicações de utilizador final. Sendo apenas necessário, como utilizador, saber operar com o *software* [35].

2.5.3 Vantagens da computação na *cloud*

Algumas das vantagens de usar computação na nuvem são:

- **Redução de custos** - Não há necessidade de grandes investimentos para adquirir, manter e instalar infraestruturas e *softwares*, pagando apenas pelo o que se usa evitando investimentos superiores ao necessário;

- **Segurança** - Os fornecedores seguem rigidamente políticas e regras de segurança mantendo os dados protegidos;
- **Flexibilidade e escalabilidade** - É possível aumentar ou diminuir os recursos das soluções sob procura;
- **Mobilidade** - Soluções disponíveis a partir de qualquer dispositivo conectado à internet, a qualquer hora e lugar;
- **Recuperação de desastres** - Os dados podem ser recuperados em emergências aplicando redundância;
- **Atualizações automáticas** - Utilizar últimas versões e tecnologias [37].

2.6 Microsoft

A plataforma de *cloud* da Microsoft, a Azure, possui mais de 200 serviços e produtos de *cloud*. [38] Nos seguintes tópicos serão introduzidos, brevemente, os conceitos de cada um dos serviços utilizados no âmbito deste trabalho realizado.

2.6.1 Azure IoT Hub

O IoT Hub é um serviço gerido e alojado na *cloud*, que funciona como um centro de mensagens para a comunicação bidirecional fiável e altamente segura entre as soluções IoT e os dispositivos que gere, figura 2.7 da página 12.

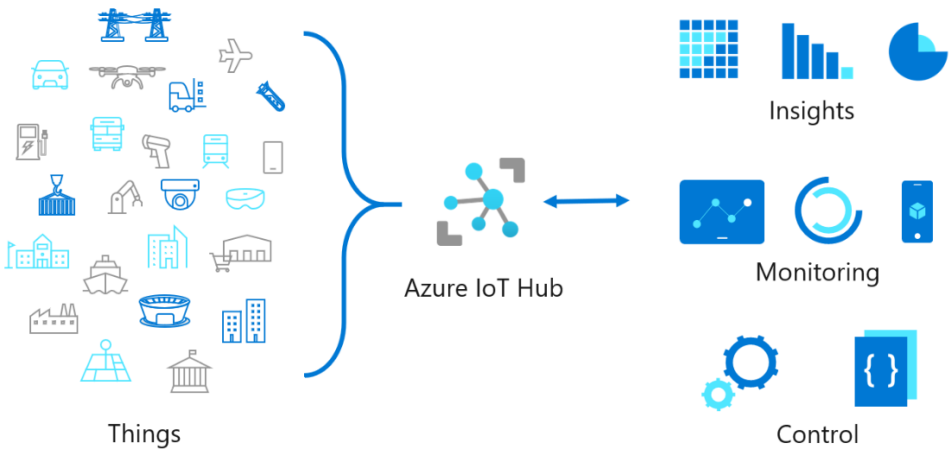


Figura 2.7: Diagrama de utilização do IoT Hub [5]

IoT Hub permite proteger as comunicações do dispositivo para a *cloud*(D2C) e da *cloud* para o dispositivo(C2D) recorrendo a várias categorias de autenticação como uso de chave simétrica,

X.509 *Self-Signed* e X.509 *CA Signed*. Permite suportar cargas de trabalho com milhões de dispositivos ligados em simultâneo e milhões de eventos por segundo, monitorizar falhas de comunicação, estado do dispositivo bem como reagir a uma alteração do mesmo e por fim, permite ligar praticamente qualquer dispositivo, sendo possível ligar diretamente dispositivos que utilizam protocolos como MQTT, HTTPS 1.1 ou AMQP, caso utilize outro protocolo a Azure disponibiliza *kits* de desenvolvimento de *software*(SDKs) *open source* para se construir soluções permitindo a comunicação com o IoT Hub. [39]

2.6.2 Azure Stream Analytics

O Stream Analytics é um motor de análise em tempo real e de processamento de eventos, concebido para analisar e processar grandes volumes de dados de fluxo rápido de múltiplas fontes simultaneamente.

Um trabalho de análise consiste numa entrada(s), consulta(*query*) e uma saída(s). A consulta dos dados das entradas, utilizando a Stream Analytics Query Language baseada na linguagem SQL, permite filtrar, classificar e agregar os dados ao longo de um intervalo de tempo. Para *querys* mais complexas é possível estender esta linguagem com funções JavaScript e C# definidas pelo utilizador. [40]

Na figura 2.8, da página 13, é ilustrado um exemplo de possíveis dados injetados e possíveis saídas.

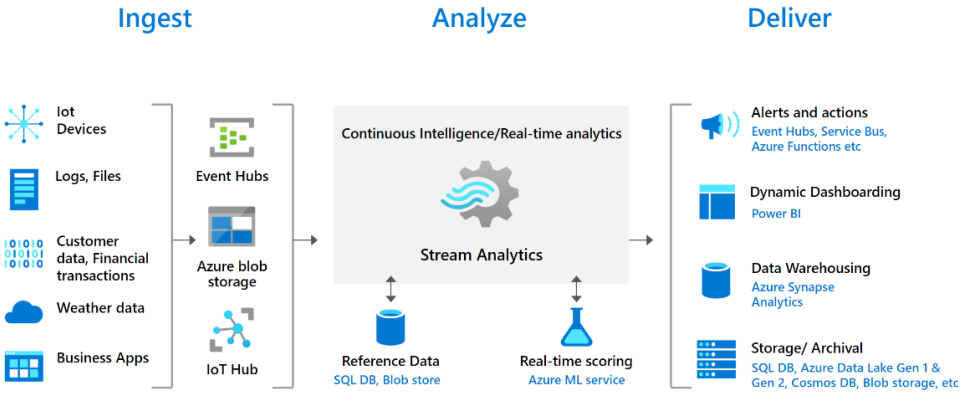


Figura 2.8: Diagrama de utilização do Stream Analytics [5]

2.6.3 Azure Data Lake Storage Gen2

O serviço Azure Data Lake Storage Gen2(ADLS) é um repositório na *cloud* para dados estruturados e não estruturados. Oferece características como alta disponibilidade, gestão do ciclo de vida, armazenamento hierárquico, segurança e durabilidade. [41]

2.6.4 Power BI

O Power BI é um conjunto de serviços de *software*, aplicações, e conexões trabalhando em conjunto para transformar as suas fontes de dados não relacionadas em perceções coerentes, visualmente envolventes, e interativas. Quer os dados sejam uma simples folha de Excel, ou provenientes de uma base de dados local, hibrida ou alojada na *cloud*, figura 2.9. O Power BI permite ligar facilmente as origens da informação, filtrar e modelar a informação sem afetar a fonte subjacente, visualizar (ou descobrir) o que é importante, e partilhá-lo com qualquer pessoa. [6]

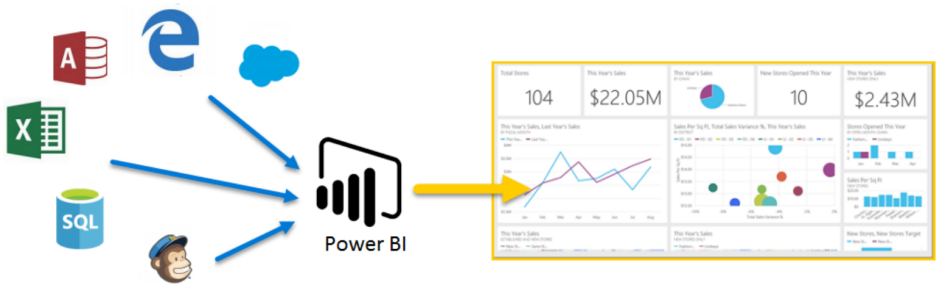


Figura 2.9: Exemplo de entradas e uma possível saída no Power BI [6]

Os principais pilares do Power BI são os conjuntos de dados(*datasets*), relatórios e *dashboards*, estando organizados em espaços de trabalho.

2.6.4.1 *Dataset*

Um *dataset* é uma coleção de dados que podem ser importados ou aos quais nos podemos conectar. O Power BI permite ligar e importar todo o género de dados e juntá-los todos num único local.

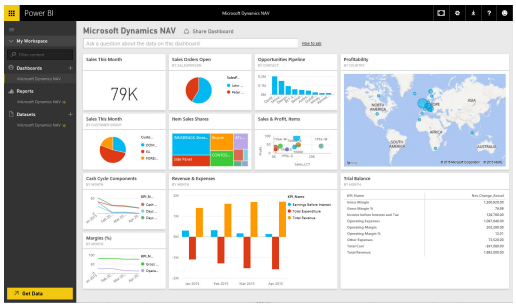
2.6.4.2 Relatórios

Um relatório é composto por uma ou mais páginas com elementos visuais, tais como gráficos de linhas, de barras, pontos, tarte, tabelas, mapas e entre muitos outros, figura 2.10 (a) da página 15. Podem ser importados como *dashboards* e com *dashboards* partilhados por outras pessoas.

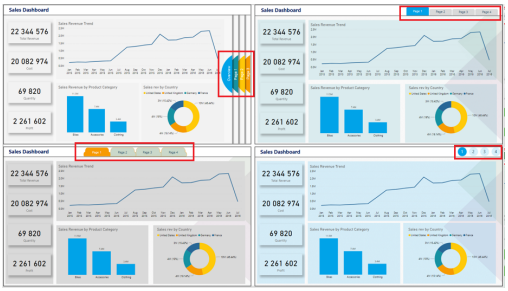
A "linguagem" utilizada para tratar dos dados é *Data Analysis Expressions* (DAX). DAX é uma coleção de funções, operadores e constantes que podem ser usados numa fórmula, ou expressão, para calcular e devolver um ou mais valores, permitindo criar novas informações a partir de dados já existentes. [42]

2.6.4.3 *Dashboard*

Trata-se de uma única tela, que contém zero ou mais mosaicos e *widgets*, figura 2.10 (b) da página 15, cada mosaico apresenta uma única visualização criada a partir de um conjunto de dados, contudo é possível afixar uma página de um relatório a um *dashboard* como um mosaico único.



(a) Relatório [43]



(b) *Dashboard* [44]

Figura 2.10: Tipos de visualização de dados

Os *dashboards* permitem observar rapidamente as informações necessárias para tomar decisões, monitorizar elementos importantes sobre o negócio e garantir que todas as pessoas envolventes estão em sintonia e conseguem ver e utilizar as mesmas informações. [6]

2.7 Amazon

Tal como a Microsoft tem a Azure, a Amazon também tem a sua plataforma de *cloud*, a Amazon Web Service (AWS), atualmente líder de mercado, figura 3.1. Os serviços AWS idênticos aos Azure utilizados neste trabalho são o AWS IoT Core, AWS IoT Analytics, Amazon S3 e Amazon Quicksight.

2.7.1 AWS IoT Core

O AWS IoT Core é usado para conectar os dispositivos à *cloud* e gerir os mesmos de maneira fácil e confiável, figura 2.11. Os protocolos usados são MQTT, HTTPS, MQTT sobre Websocket Secure e LoRaWAN.

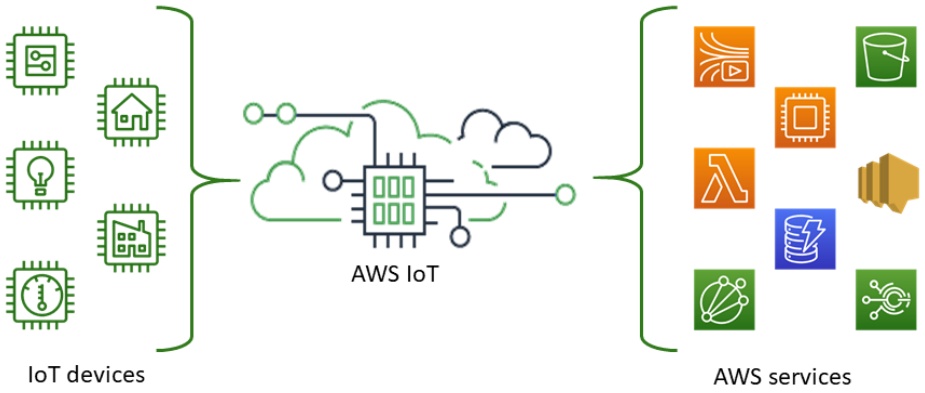


Figura 2.11: Diagrama de utilização do IoT Core [7]

Com o IoT Core é possível transmitir mensagens de e para dispositivos e aplicações IoT, com o Device Shadow podemos armazenar o estado mais recente de um dispositivo de forma a ler e configurar dispositivos com coneções intermitentes, Alexa integrada e gerir redes privada de LoRaWAN sem desenvolver ou operar um servidor de rede LoRaWAN [7].

2.7.2 AWS IoT Analytics

O IoT Analytics é um serviço totalmente gerido que facilita a execução e a operacionalização de análises sofisticadas com volumes massivos de dados. Forma fácil de executar análises de dados e obter visões que ajudam a tomar melhores decisões e mais precisas para aplicações IoT e de *machine learning*.

O IoT Analytics pode ser configurado para filtrar apenas os dados necessários, aplicar transformações matemáticas e enriquecer com metadados como, por exemplo, a localização do dispositivo. Estes dados podem ser analisados por consultas SQL ou com análises mais complexas e com interferência de *machine learning*. Para além disto também é possível análises personalizadas criadas no Notebook Jupyter ou em ferramentas como o Matlab e Octave [45].

Apoós este processamento os dados podes ser transportados para outros serviços para serem armazenados ou visualizados como, por exemplo, o Amazon Quicksight, figura 2.12 apresenta uma possível utilização.

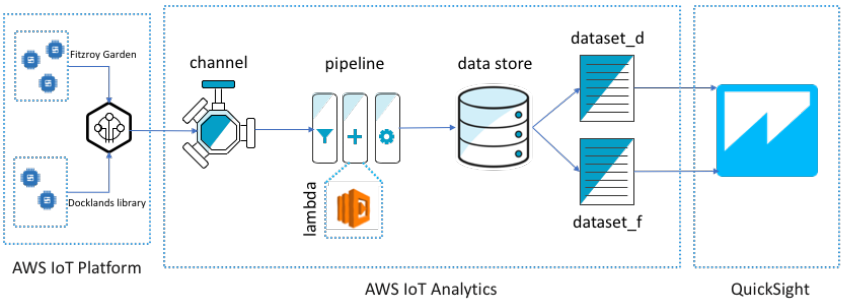


Figura 2.12: Diagrama de utilização do IoT Analytics [8]

2.7.3 Amazon S3

Amazon Simple Storage Servie (Amazon S3) é um serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados, segurança e desempenho. Os clientes podem armazenar e proteger qualquer quantidade de dados de praticamente qualquer caso de uso.

Apresenta pacotes de armazenamento económicos, organização de dados e é possível configurar o acesso à informação para atender os requisitos de cada negócio [46].

2.7.4 Amazon Quicksight

O serviço Amazon Quicksight permite mostrar com uma visão fácil os dados através da internet, permitindo os clientes entenderem os dados, usando de painéis interativos e procurar automaticamente padrões e discrepâncias com tecnologias de *machine learning* [9].

Figura 2.13: *Dashboard* em Amazon Quicksight [9]

O Amazon Quicksight conecta-se não só aos dados na *cloud* mas também de diversas fontes como, por exemplo, bases de dados fora da nuvem. Serviço totalmente gerido, fornece segurança

de nível empresarial, disponibilidade global, redundância integrada e ferramentas de gestão de utilizadores [47].

2.8 Node-RED

O Node-RED é uma plataforma *open source* de programação visual desenvolvida em Node.js pela IBM, acessível por *browser*, que permite criar fluxos por meio de interligação de blocos de código (*nodes*), figura 2.14. Foi concebida para atender à necessidade de conectar rapidamente *hardware* e dispositivos a serviços web ou *softwares* [48].

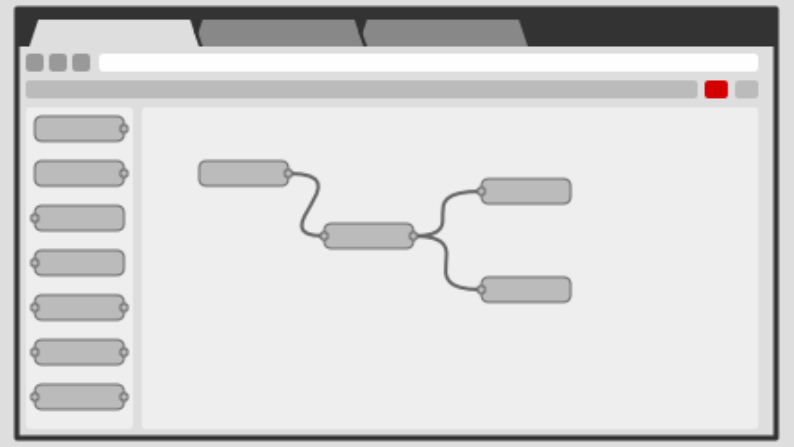


Figura 2.14: Fluxo em blocos Node-RED [10]

Para além dos nós padrão do Node-RED, figura 2.15, devido à sua popularidade e de ser *open source* é possível importar blocos ou fluxos desenvolvidos pela comunidade e até mesmo empresas, auxiliando na interação com os seus dispositivos ou *softwares*. No entanto, também é possível desenvolver as nossas próprias funções e lógicas usando a linguagem JavaScript no próprio *rich text editor* da ferramenta.

Figura 2.15: Nós padrão do Node-RED [11]

Capítulo 3

Abordagem

3.1 Requisitos

Como intuito de garantir que este trabalho atende exatamente ou ultrapasse ao que é esperado pela empresa, é essencial reunir toda a informação possível. Com esta informação foram levantados os requisitos presentes na tabela 3.1.

Requisito	**Descrição**	**Importância**	**Obrigatório**
RF01	Aceder aos valores dos sensores, eventos e alarmes da máquina. (servidor OPC UA)	Crítica	Sim
RF02	*Dashboard* com os valores dos sensores, eventos e alarmes da máquina.	Crítica	Sim
RF03	*Dashboard* com vendas e clientes.	Valorizado	Não
RF04	Comunicação unidirecional, da máquina para cloud.	Crítica	Sim
RF05	Guardar os alarmes e eventos da máquina numa base de dados.	Crítica	Sim
RNF01	Programa a correr numa unidade computacional externa. (Windows)	Crítica	Sim
RNF02	Programa a correr numa unidade computacional independentemente do OS.	Valorizado	Não
RNF03	Taxa de atualização do dashboard quase em tempo real, <= 10 segundos.	Crítica	Sim
RNF04	*Dashboards* acessíveis via web.	Crítica	Sim

Tabela 3.1: Tabela de requisitos.

Os requisitos funcionais (RF) representam as necessidades que o projeto precisa atender e os requisitos não funcionais (RNF) caraterizam a maneira de como o sistema deve responder a estas imposições.

Os requisitos *RF03* e *RNF04* são de carácter não obrigatório, onde o primeiro surgiu de um *mockup*, apresentado pela empresa, de um exemplo pedido anteriormente a uma empresa terceira

que desenvolve estes serviços. Numa reunião a empresa demonstrou que idealizava e tinha interesse e por conseguinte, por ânsia pessoal, foi deliberado elaborar também este *dashboard*. O *RNF04* foi uma melhoria de escalabilidade e versatilidade proposta por mim para este trabalho.

3.2 Plataforma/serviços IoT estudados

Tendo em conta os 3 maiores fornecedores de serviços *cloud*, figura 3.1, as possibilidades estudadas foram Microsoft Azure, Amazon AWS e Google Cloud [12]. Para além destas, também foi analisada uma das plataformas *open source* mais populares, Thingsboard [49].

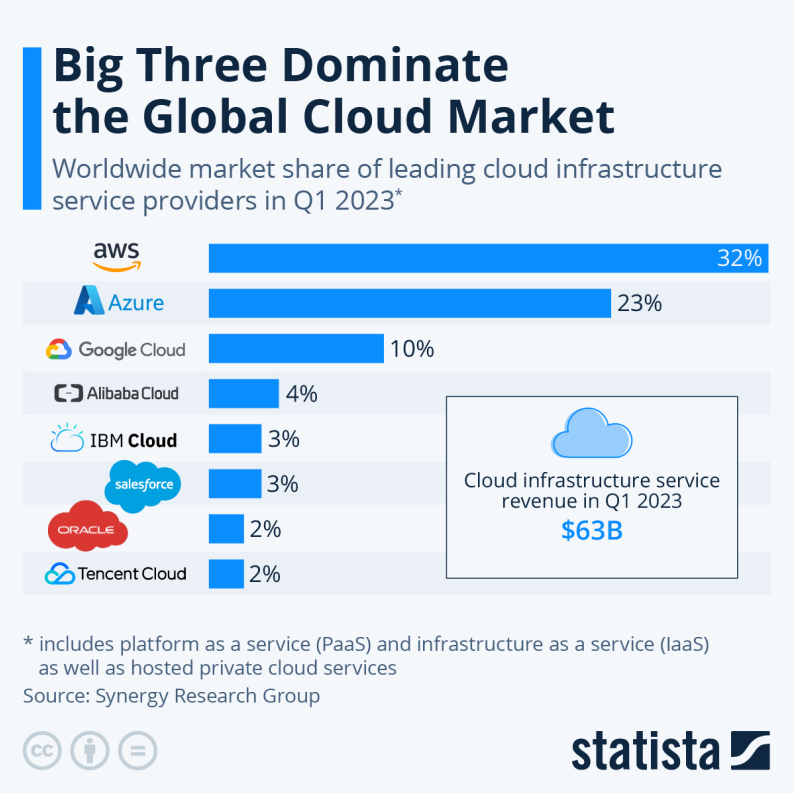


Figura 3.1: Maiores fornecedores de serviços *cloud* no Q1 2023 [12]

As estimativas de custo mensal, por cada cliente da empresa, das soluções de *cloud* foram realizadas para 2, 5 e 25 dispositivos. Para a plataforma *open source* foram realizadas estimativas com base nos escalões de subscrições fornecidas para 10, 100 e 500 dispositivos e depois calculado o preço por dispositivo.

As soluções com a plataforma ThingsBoard têm como base a contratação das suas subscrições, sendo as seguintes da figura 3.2.

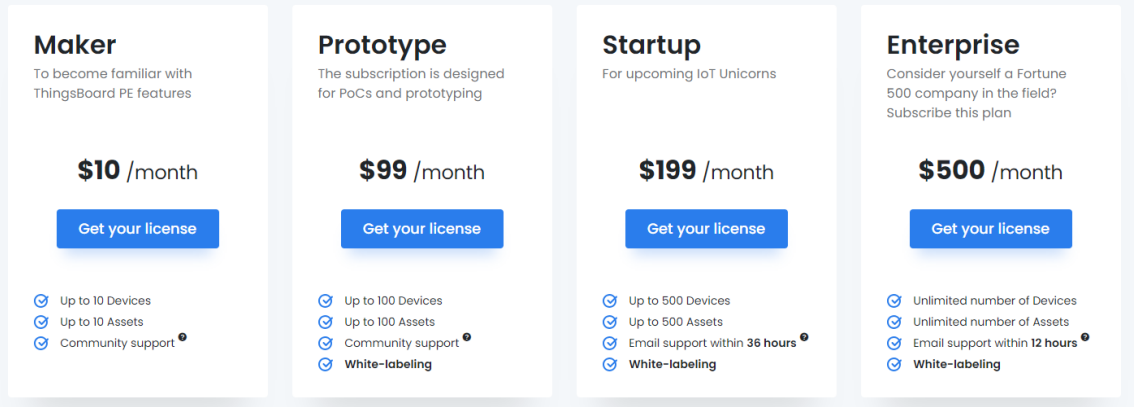


Figura 3.2: Planos *self-managed* ThingsBoard [13]

Os valores de custo descritos no anexo A são referentes à presenta data de realização deste trabalho.

3.2.1 Solução A - Microsoft Azure Cloud + PowerBI

A proposta ”A” é composta pelos seguintes serviços Azure e PowerBI também da Microsoft:

- IoT Hub: Conectar a aplicação IoT com os dados dos dispositivos;
- Stream Analytics: Executar regras e transportar os dados do IoT Hub para o PowerBI e base de dados;
- ADLS Database: Armazenar telemetria;
- PowerBI: Visualizar, analisar e transformar os dados;

As estimativas desta solução estão presentes nas tabelas A.1, A.2 e A.3 no anexo A.

3.2.2 Solução B - Amazon AWS Cloud

A proposta ”B” é composta pelos seguintes serviços *cloud* da AWS:

- IoT Core: Conectar a aplicação IoT com os dados dos dispositivos;
- IoT Analytics: Executar regras e transportar os dados do IoT Core para o Quicksight e para o armazenamento S3;
- Amazon S3: Armazenar telemetria;
- Amazon Quicksight: Visualizar, analisar e transformar os dados;

As estimativas desta solução estão presentes nas tabelas A.4, A.5 e A.6 no anexo A.

3.2.3 Solução C - Google Cloud

O serviço IoT Core da Google Cloud, utilizado para conectar e administrar dispositivos, será desativado em 16 de agosto de 2023, abandonando assim o mundo da internet das coisas. [50].

3.2.4 Solução D - Thingsboard *Self Managed* + VM Azure

Esta solução consiste em implementar o *software* da ThingsBoard e uma base de dados não relacional como, por exemplo, Apache Cassandra numa máquina virtual alojada na Azure.

As estimativas desta solução estão presentes nas tabelas A.7, A.8 e A.9 no anexo A. O Valor total apresentado nas tabelas não representa o valor mensal para por cliente uma vez que podemos contratar até 100 dispositivos e dividir por diversos clientes desde que a sua soma de dispositivos não ultrapasse o contratado. Com isto foi calculado o preço por dispositivo e na tabela A.10 está presente o valor para 2, 5 e 25 dispositivos.

3.2.5 Solução E - Thingsboard *Self Managed* + VM AWS

Para a Solução ”E” é análoga à solução ”D” com exceção de que a máquina virtual a usar estaria hospedada na AWS.

As estimativas desta solução estão presentes nas tabelas A.11, A.12, A.13 e A.14 no anexo A.

3.2.6 Solução implementada

A cloud usada foi a Azure devido à empresa ser um parceiro Microsoft e ter preferência pelos serviços dos mesmos.

Na tabela 3.2 estão compilados todos os valores estimados para cada solução, tendo e atenção que no caso das soluções ”D” e ”E” os valores são para o máximo de dispositivos de cada subscrição obtendo assim o valor mais barato possível. Na solução ”A” o sei valor está sobrestimado devido ao cálculo do preço do serviço Stream Analytics fornecido pela Microsoft não ser transparente.

			Solução D			**Solução E**		
Dispositivos	**Solução A**	**Solução B**	Maker	Prototype	Startup	Maker	Prototype	Startup
2	45.84 €	23.89 €	8.35 €	3.08 €	0.98 €	7.09 €	2.79 €	0.92 €
5	55.18 €	24.50 €	20.88 €	7.70 €	2.45 €	17.73 €	6.97 €	2.30 €
25	55.18 €	27.64 €	—	38.49 €	12.25€	—	34.83 €	11.52 €

Tabela 3.2: Estimativa de cada Solução

As soluções mais económicas por cada dispositivos são as que utilizam a plataforma *open source*, propostas ”D” e ”E”. Contudo, uma vez que há a necessidade de um autogerenciamento dos serviços do ThingsBoard e da base de dados, estas opções foram descartadas.

A seguir confrontando a solução ”A” com a ”B” podemos observar que a proposta da Amazon é mais acessível do que a da Microsoft. Todavia devido à sua real diferença de custo não ser muito discrepante e a empresa ser parceiro da Microsoft foi priorizada a solução ”A”.

3.3 Arquitetura

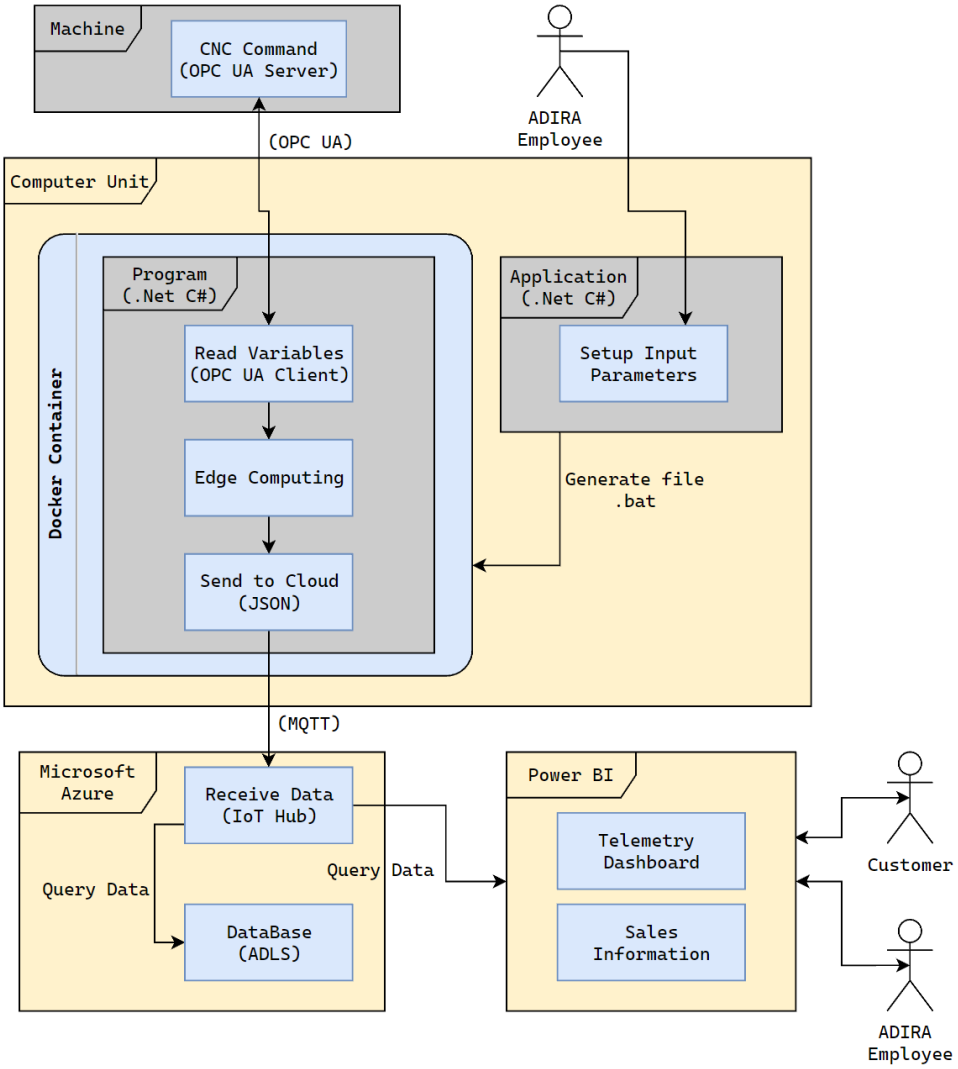


Figura 3.3: Arquitetura da solução

Na figura 3.3 está representado um diagrama da arquitetura da solução proposta para este projeto, tento em consideração todos os requisitos, secção 3.1, levantados com o cliente na fase preliminar deste projeto.

As máquinas estudadas/analisadas para desenvolver este trabalho têm na sua composição um comando CNC que integra um servidor OPC UA. Este comando CNC é quem comanda todas as ações da máquina e corre sobre um sistema operativo Windows.

O programa e aplicação desenvolvido correm numa unidade de computação externa ao computador que corre a CNC. Esta decisão de tornar num produto ”extra” foi tomada com base na lógica de negócio do cliente.

O programa principal foi desenvolvido e pensado para correr num *container*, oferecendo diversidade e escalabilidade, dando a liberdade do cliente decidir qual OS deseja utilizar e dispositivo de computação e acrescentar monitorização de novas máquinas em poucos minutos. Contudo, o cliente comunicou que a sua principal escolha seria Windows, mas caso tencione alterar para outra solução como, por exemplo, um Raspberry Pi, o programa funciona do mesmo jeito.

A aplicação de configuração foi realizada e testada em Windows. Esta pode estar instalada na mesma unidade de computação do programa principal ou então estar num computador portátil/remoto do funcionário do cliente que estejam a fazer a configuração da mesma. Uma vez que o importante são os dois ficheiros de saída da aplicação e sendo posteriormente transferidos para a unidade computacional. A criação desta aplicação advém de o cliente ter diferentes tipo de máquina, com sensores diferentes e períodos de manutenção dispare, e desta forma obtemos uma monitorização personalizada para cada tipo de máquina bem como a configuração da mesma mais fácil para qualquer utilizador.

A *cloud* utilizada neste trabalho, como já referido anteriormente, foi a Azure, *cloud* da Microsoft. O programa principal comunica com o serviço IoT Hub através do protocolo MQTT, um dos três possíveis indicados na documentação da própria Microsoft [51]. Na Azure os dados vão ser filtrados e caso ultrapassem valores estipulados ou os alarmes ativados serão guardados numa base de dados não relacional para se manter um histórico de alertas da máquina. Os dados também são enviados para o Power BI para serem tratados e popular os *dashboards*.

O *dashboard* de telemetria por meio de permissões poderá ser partilhado com os clientes da empresa. O *dashboard* de vendas será apenas de uso interno da empresa.

Capítulo 4

Desenvolvimento

Na fase do desenvolvimento foram realizadas 5 tarefas distintas:

- Aplicação de configuração;
- Servidor OPC UA;
- Programa principal;
- Dados na *Cloud*;
- *Dashboards*/Relatórios.

4.1 Aplicação de configuração

Para tornar o programa de leitura das variáveis geral e não ter parâmetros únicos que teriam de ser alterados, manualmente no código raiz, foi desenvolvida uma aplicação Windows em C# utilizando a *framework* de interface do utilizador Windows Forms (.NET 5.0), que conforme o tipo de máquina, são introduzidos alguns dados que servirão como parâmetros de entrada para o programa.

A Aplicação tem como diagrama de atividade a figura 4.1 e consiste em duas janelas, uma inicial em que se escolhe o tipo de máquina, figura 4.2, e a que a sucede com os parâmetros necessários a introduzir para esse mesmo modelo de máquina, figura 4.3. Esta tem 5 campos para preencher, nome do *container*, endereço do servidor OPC UA, *string* de conexão do dispositivo no IoT Hub, período de manutenção e o diretório onde serão criados os ficheiros. Caso um destes campos não for preenchido, os ficheiros não serão criados.

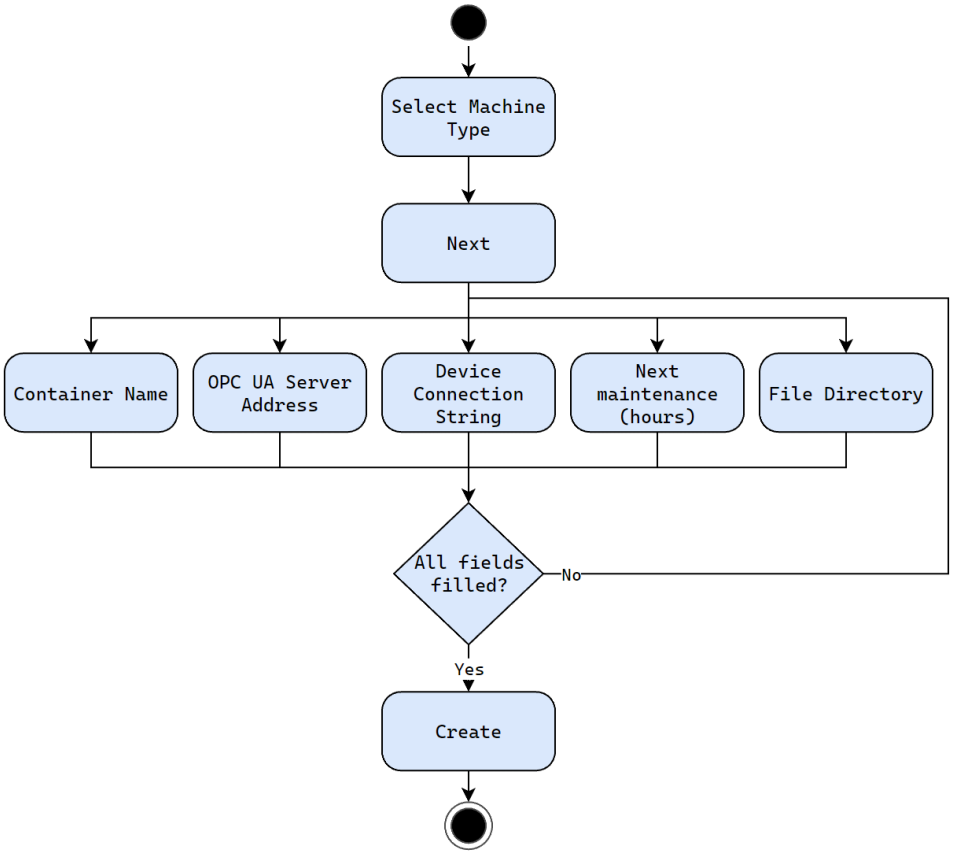


Figura 4.1: Diagrama de atividade da aplicação de configuração

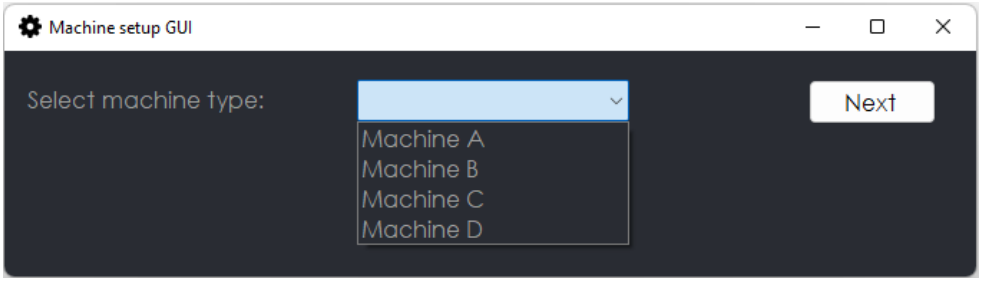


Figura 4.2: Janela de selecionar o tipo de máquina

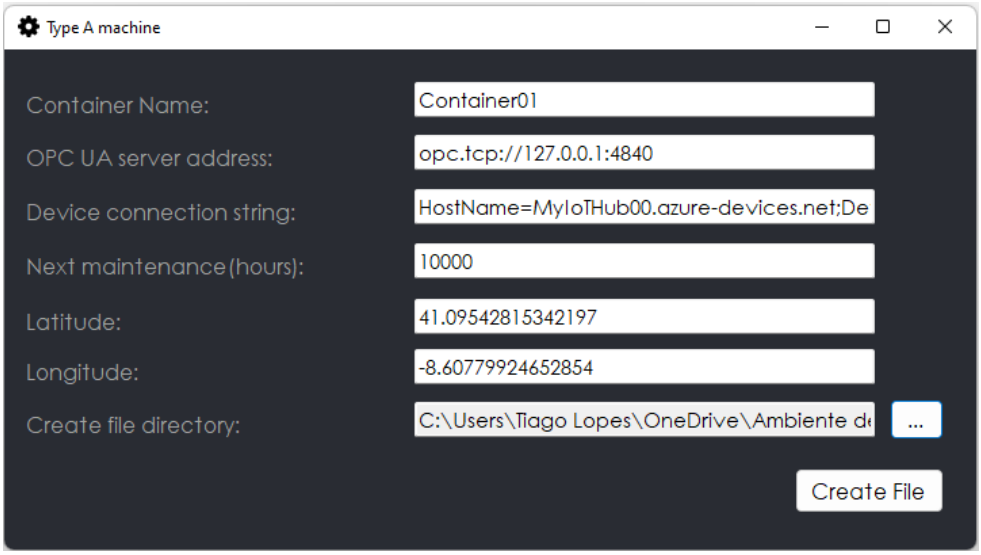
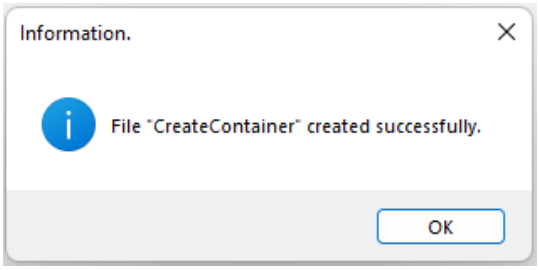
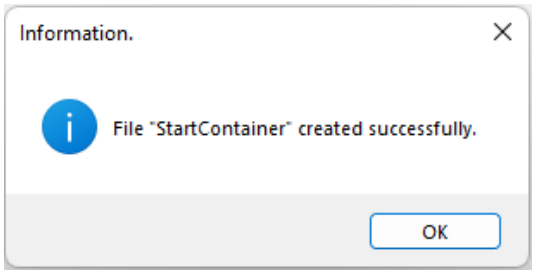


Figura 4.3: Janela de configuração

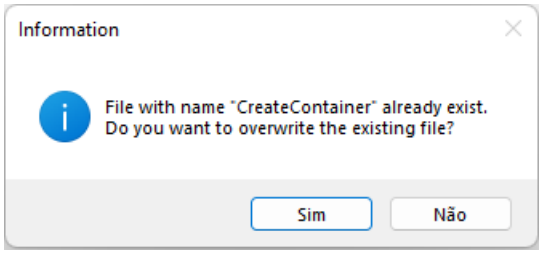
No caso da máquina A, que foi a idealizada, os campos obrigatórios são o nome a dar ao container, o endereço do servidor OPC UA, *connection string* do dispositovo no serviço IoT Hub, período entre manutenções, e o diretório onde serão criados dois ficheiros *batch*, o ficheiro *CreateContainer.bat* e o *StartContainer.bat*, figura 4.5 e figura 4.6 respetivamente. As mensagens de saída da aplicação quando os ficheiros são criados estão representadas na figura 4.4, dando informação ao utilizador se os *scripts* foram gerados com sucesso não.



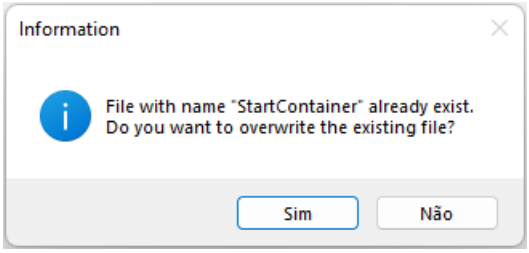
(a) Ficheiro *CreateContainer.bat* criado com sucesso



(b) Ficheiro *StartContainer.bat* criado com sucesso



(c) *CreateContainer.bat* caso já exista



(d) *StartContainer.bat* caso já exista

Figura 4.4: Exemplos de mensagens de saída da aplicação

```
@ECHO OFF
TITLE Execute docker container
ECHO Executing docker command and creating container Container01
docker create -t -v "C:\Users\Tiago Lopes\OneDrive\Ambiente de Trabalho\FicheirosAppIoT:/tmp"
--name Container01 up201909558/tese-repository:maqA-version1.4 opc.tcp://127.0.0.1:4840
"HostName=MyIoTHub00.azure-devices.net;DeviceId=MaqA01;SharedAccessKey=[illegible]" "41.09542815342197" "-8.60779924652854" 1000
PAUSE
```

Figura 4.5: Script exemplo *CreateContainer.bat*

```
@ECHO OFF
TITLE Start docker container
ECHO Starting container: Container01
docker start Container01
PAUSE
```

Figura 4.6: Script exemplo *StartContainer.bat*

O script *CreateContainer.bat* cria um *container* com a imagem do programa principal que por sua vez este será inicializado pelo ficheiro *StartContainer.bat* permitindo também, que o *container* arranque automaticamente, em situações diversas que causem o *shutdown* da máquina.

O comando que compõe o *CreateContainer.bat* tem a seguinte estrutura:

*docker create -t -v "**Diretório no Host:Diretório no Container**" --name "**Nome do Container**" "**Repositório:tagName**" "**Parâmetros**"*

- **Diretório no *Host***: "C:\Users\Tiago Lopes\OneDrive\Ambiente de Trabalho\TeseFiles";
- **Diretório no *Container***: "/tmp";
- **Nome do *Container***: "Container01";
- **Repositório**: Repositório da imagem docker "up201909558/tese-repository";
- **tagName**: Versão da imagem docker "maqA-version1.4";
- **Parâmetros**:
  - Endereço servidor OPC UA;
  - *Connection string* do dispositivo criado no portal Azure;
  - Latitude;
  - Longitude;
  - Período entre manutenções;

A *flag* de volume (-v) é utilizada para criar um mecanismo para persistir os ficheiros utilizados e gerados pelo *container*, ou seja, o sistema de ficheiros montado no *container* é guardado na máquina *host* prevenindo a perda de dados caso o *container* seja destruído.

4.2 Servidor OPC UA

Para simular o servidor OPC UA, que se encontra presente no painel de comando das máquinas, e os sensores das mesmas foi utilizado a *framework* Node-RED que proporcionou, de uma forma rápida e rigorosa, variar/adicionar as variáveis de leitura a supervisionar para tornar os testes do código do programa principal e dashboard minuciosos.

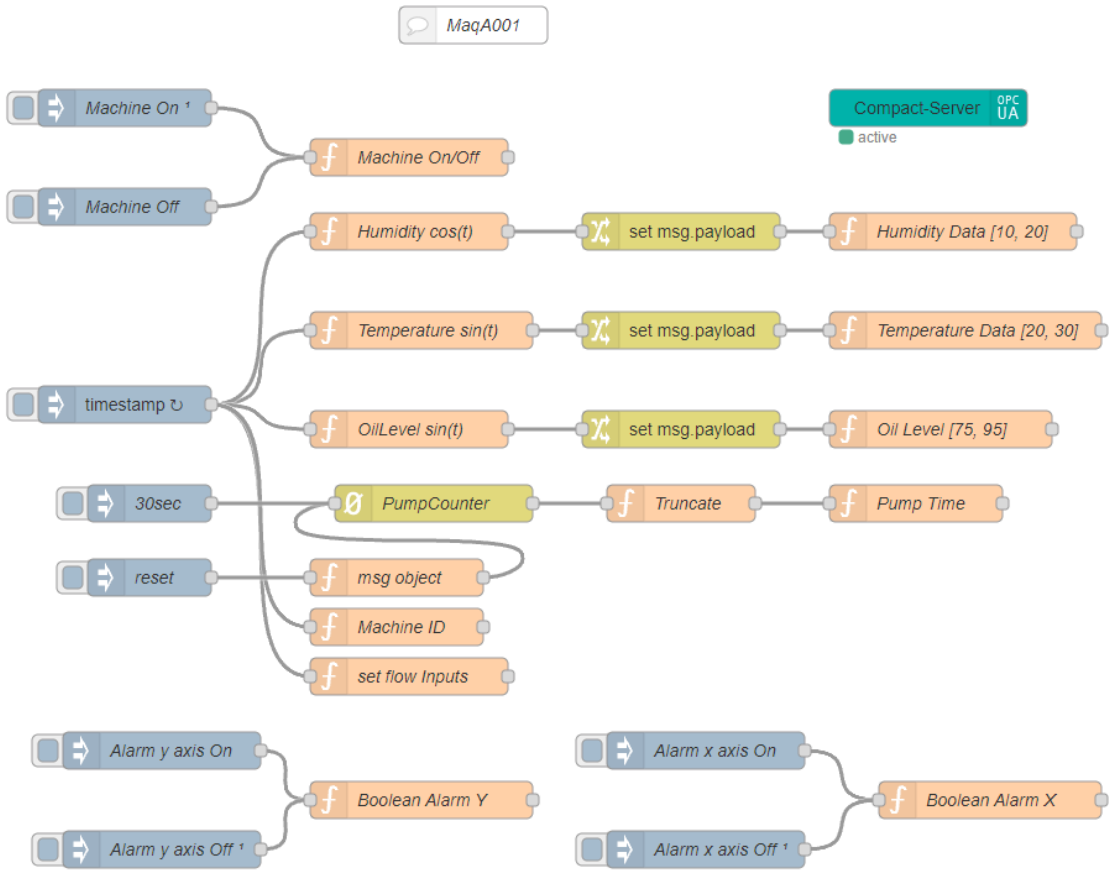


Figura 4.7: Simulador do servidor OPC UA em Node-RED

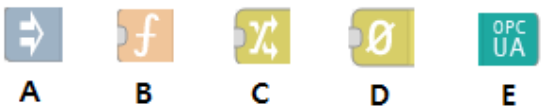


Figura 4.8: Tipo de nós utilizados

Na figura 4.7 está presente o diagrama de fluxo e na figura 4.8 os nós, etiquetados, utilizados na simulação do servidor OPC UA. O nó do tipo A é utilizado para injetar mensagens cujo *payload* é booleano para criar "impulsos" como, por exemplo, ligar e desligar a máquina. Do tipo B e D são funções, sendo que o nó B é utilizado para construir as nossas próprias funções em javaScript

e o nó D é uma função de contador. O nó C é utilizado para modificar propriedades da mensagem como, por exemplo, fazer um arredondamento a duas casas decimais do *payload* e por último o nó E é o servidor OPC UA onde são configuradas as suas definições para o *endpoint* bem como criar todas a variáveis que se planeou monitorizar na máquina, o código desde bloco está no anexo B.

De maneira a testar valores que fossem variáveis ao longo do tempo, simulando a realidade, e que ativassem os alarmes a partir de um determinado*threshold*, foram usadas as expressões matemáticas *cos* e *sin* para os valores dos sensores de temperatura, humidade e de nível de óleo que variam, respetivamente, entre 20 ºC e 30 ºC, 10% e 20% e entre 75% e 95%. O estado da máquina, alarme X e alarme Y são impulsos booleanos, podendo os seus estados serem alterados manualmente no UI do Node-RED. A variável da bomba de óleo é incrementada a cada 30 segundos, a título experimental equivalendo a 1 hora na realidade, sendo esta bastante importante para contabilizar o tempo que falta para a manutenção da máquina.

Para testar conexões e monitorizar as variáveis do servidor foi utilizado o Prosys que simula um cliente OPC UA. 4.9

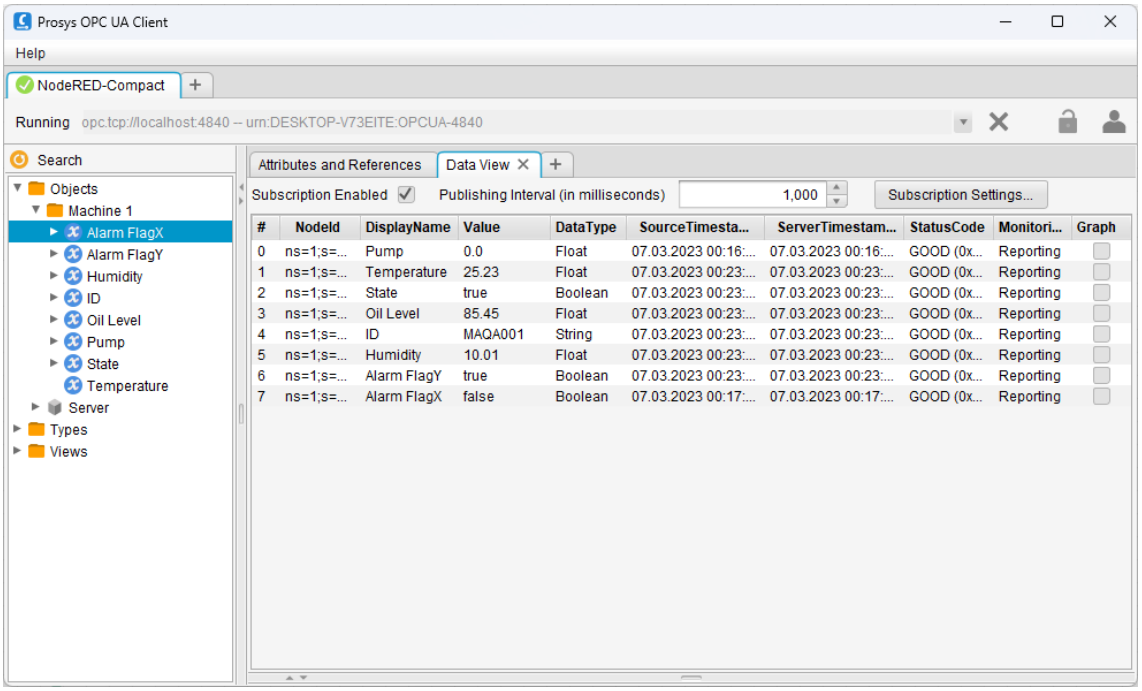


Figura 4.9: Prosys cliente OPC UA

4.3 Programa principal

O programa principal foi realizado na linguagem C# com recurso à *framework* .NET 5 tendo sido usadas as seguintes livrarias System.* [52], Microsoft.Azure.Devices.Client [53], Newtonsoft.Json [54] e Opc.UaFx.Client [55].

Para a função *Main* são passados 5 argumentos, endereço do servidor OPC UA, *connection string* do IoT Hub, latitude, longitude e período de manutenção, argumentos estes que são enviados na criação do container 4.5. A seguir todas as variáveis pertinentes são inicializadas.

Usando o SDK IoT da Microsoft é criado um cliente para se conectar ao recurso IoT Hub usando o protocolo MQTT. Como se pode conferir na função 4.1 é preciso passar a *connection string* do serviço e o tipo de protocolo que queremos usar nos seus argumentos.

```
deviceClient = DeviceClient.CreateFromConnectionString(connectionString,
    TransportType.Mqtt);
```

Listing 4.1: Método para se conectar ao IoT Hub na Azure

Para além do cliente para a conexão ao serviço na *cloud*, é necessário criar também um cliente OPC UA para podermos ler os valores das variáveis do servidor. Para isto foi usado o método *OpcClient* da livraria Opc.UaFx.Client como único argumento o endereço do servidor OPC UA 4.2.

```
client = new OpcClient($"{opcAddress}");
```

Listing 4.2: Método para criar um cliente OPC UA

Após a conexão ser estabelecida são despontadas duas *threads* adicionais, t1 e t2, para além da *threads* da *Main*.

A *threads* t1 é responsável por ler os valores dos alarmes X e Y e sempre que existir uma troca de estado de ON para OFF ou de OFF para ON num dos alarmes, uma mensagem é enviada para o IoT Hub com o estado dos mesmos. Esta leitura é realizada de meio em meio segundo.

Já a *threads* t2 tem como finalidade a leitura das variáveis do servidor OPC UA. São lidos os valores de 1 em 1 segundo e adicionados a um *array* de listas (dataArrayList). Este *array* tem 3 posições, em que cada posição corresponde a uma lista com os valores medidos de temperatura, humidade e nível de óleo respetivamente. Nesta *threads* também é lido o estado da máquina, se ligada ou desligada. Este estado é usado para calcular o tempo de funcionamento da bomba de óleo da máquina que servirá para determinar o tempo que falta até realizar a próxima manutenção. Para realizar a leitura dos valores do servidor foi usado método *ReadNodeIdValue* 4.3 disponibilizado pela livraria de OPC já referenciada.

```
dataArrayList[0].Add((float)ReadNodeIdValue(client, NodeId_Temperature).Value);
```

Listing 4.3: Método para ler variável do servidor OPC UA

Após isto, dentro da *Main* tem um *while* infinito que zela pelo tratamento de dados antes de serem enviados para a *cloud*. Este tratamento consiste em ler o *array* dataArrayList e calcular o valor máximo, médio, mínimo e atual de temperatura, humidade e nível de óleo. Estes valores serão guardados noutro *array* (telemetryValuesArray).

Posteriormente o dataArrayList é limpo para receber os novos valores na *thread* t2.

Seguidamente é calculado o número de horas até à próxima manutenção da máquina, este valor é determinado através do número de horas recomendadas inserido na aplicação de configuração, na fase de criar o container, subtraindo o número total de horas em que a máquina está ativa, que como visto anteriormente, vem do contador na *thread* t1. De modo a esta informação não

se perder em diversos casos espontâneos como, por exemplo, falha de energia elétrica, unidade computacional que corre o container se desligar, "etc", o tempo total é guardado num ficheiro de texto. Depois da manutenção efetuada, a equipa de suporte apenas tem de eliminar o ficheiro ou editar para o número zero.

Por último a mensagem com os dados dos sensores, alarmes, latitude, longitude, número de horas até manutenção e período de manutenção é enviada para a *cloud*. *SendDeviceToCloudTelemetry* 1 foi o método criado para esta finalidade. Para transformar todos os valores num formato estruturado, neste caso em JSON, foi criada a função *CreateTelemetryMessageString* 2 que com recurso à livraria Newtonsoft.Json converteu-se um objeto em JSON.

Função *SendDeviceToCloudTelemetry*

```
procedure SENDDEVICETOCLOUDTELEMETRY(data)
    messageString ← CreateTelemetryMessageString(data)        ▷ Criar string JSON
    message ← Message(messageString)        ▷ criar array de bytes usando ASCII encoding
    Enviar mensagem para a cloud
end procedure
```

Função *CreateTelemetryMessageString*

```
procedure CREATETELEMETRYMESSAGESTRING(data)
    localTime ← DateTime.Now                                   ▷ Data e hora local
    telemetryData ← telemetryObject        ▷ Objeto com toda a telemetria para a cloud
    Converter objeto em JSON
    Retornar JSON
end procedure
```

A título de resumo a telemetria é enviada a cada 15 segundos (período da *Main*) ou quando um alarme troca de estado. Este intervalo de tempo de envio foi pensado de maneira minimizar o número de mensagens enviados para a Cloud, uma vez que quantas mais mensagens forem recebidas no IoT Hub maior é o valor do recurso e as métricas a ler dos sensores não possuem variações recorrentes, não se justificando a monitorização em tempo real.

4.4 Dados na *cloud* Azure

Na figura 4.10 está presente um diagrama de fluxo de dados, de alto nível, que mostra a sequência e os serviços *cloud* utilizados, Azure Iot Hub, Azure *Stream Analytics*, Azure Storage account (*data lake storage*) e Power BI.

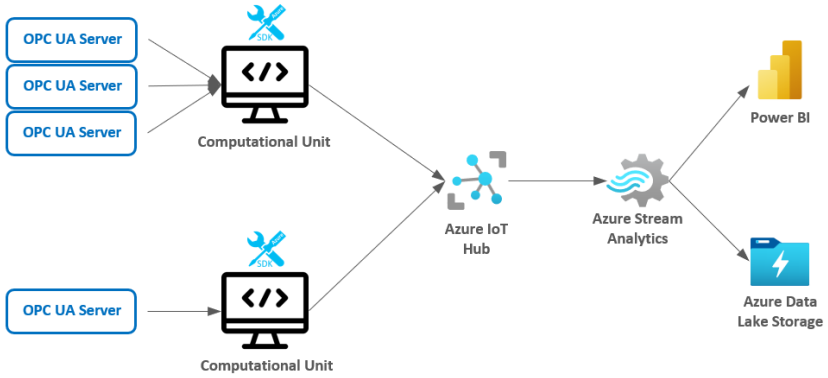


Figura 4.10: Fluxo de dados na *Cloud*

4.4.1 *Resource Group*

Para manter os serviços *cloud* organizados por projetos, sendo uma boa prática, foi criado um grupo de recursos que agrupa os serviços Azure utilizados na realização deste trabalho de dissertação, figura 4.11.

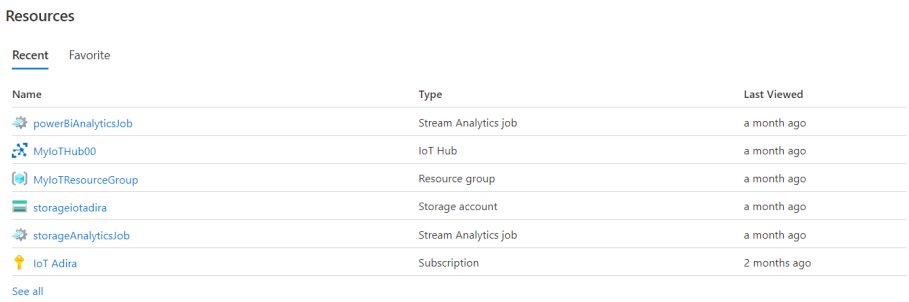


Figura 4.11: *Connection String* de um dispositivo [14]

4.4.2 *IoT Hub*

O IoT Hub é o serviço responsável por receber as mensagens com a telemetria enviada pelo programa principal. Após o Hub ser criado é altura de adicionar os dispositivos, estes são utilizados para virtualizar as máquinas, para interligar a máquina física e o seu dispositivo correspondente na *cloud* é utilizado a *connection string* 4.12, a mesma é usada para preencher o campo apropriado na aplicação de configuração já referida anteriormente na figura 4.3 da página 29.

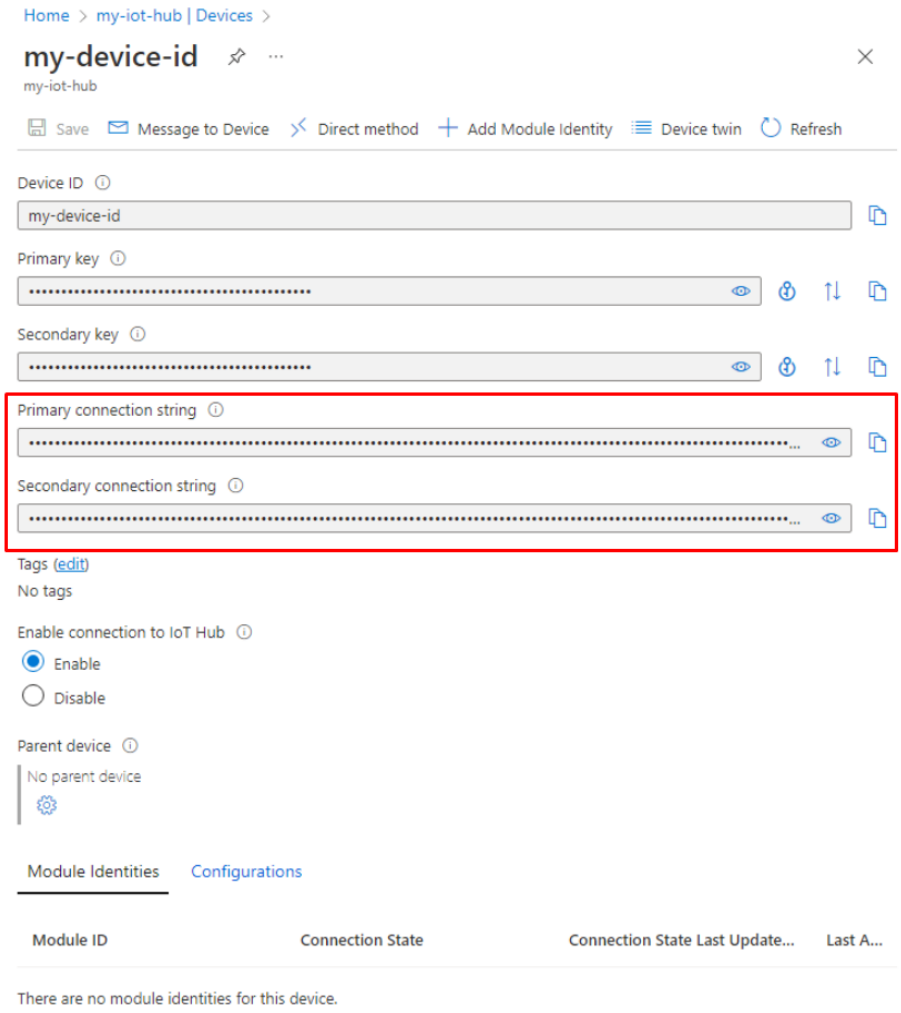


Figura 4.12: *Connection String* de um dispositivo [14]

O escalão escolhido para este serviço foi o gratuito, usufruindo de uma unidade IoT Hub com capacidade de 8000 mensagens por dia e com um tamanho de 0,5 KB [56].

4.4.3 *Stream Analytics*

Para processar, filtrar e encaminhar os dados recebidos pelo IoT Hub foi usado o serviço *Stream Analytics*. A execução de um processamento de dados neste serviço é denominado *job* tendo sido criados dois *jobs* distintos, um para enviar os dados para o Power BI e outro para os armazenar na base de dados não relacional.

O plano escolhido para este serviço foi o *standard* com um custo de 0,10€ por hora por unidade de transmissão [57].

Na figura 4.13 está representada a *query* criada para selecionar a informação que queremos monitorizar no *dashboard*. A *job* tem como entrada os dados no IoT Hub e saída o espaço de trabalho e *dataset* no Power BI. A informação filtrada para monitorizar é o ID que identifica a

máquina, a data atual, latitude, longitude, temperatura, percentagem de humidade, nível de óleo, estados do alarme X e alarme Y e o tempo total de funcionamento da bomba de óleo.

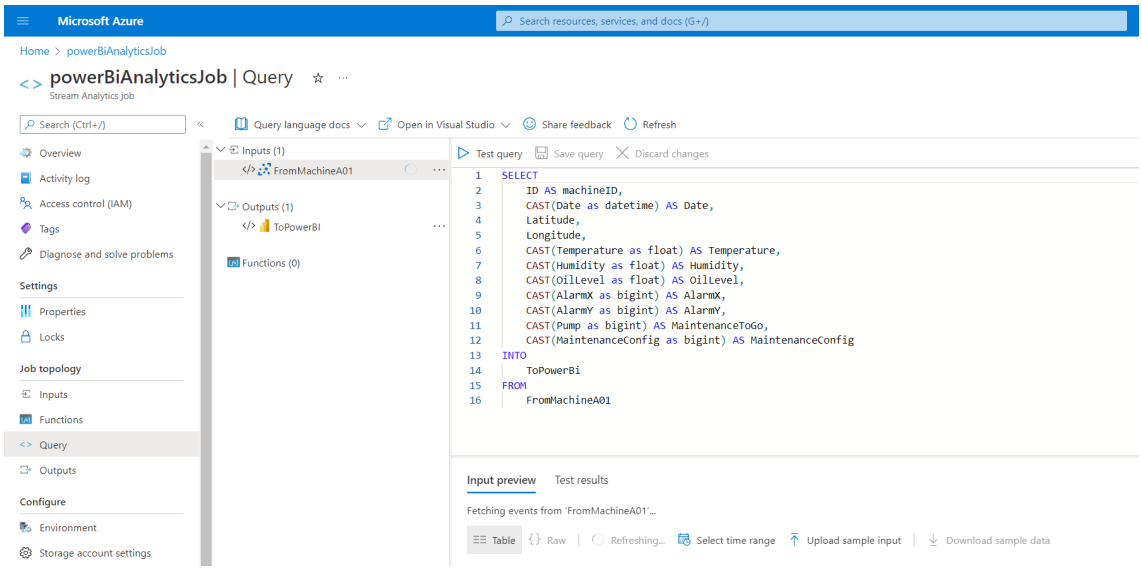


Figura 4.13: *Job* de enviar dados para o Power BI

A *query* do *job* que armazena os dados na base de dados é a da figura 4.14, os dados do IoT Hub são sua entrada e como saída o serviço de base de dados usado *storage account*. A informação guardada é o ID da máquina, a data, tempo total de funcionamento da bomba de óleo, latitude, longitude, temperatura, percentagem de humidade, nível de óleo, estados do alarme X e alarme Y.

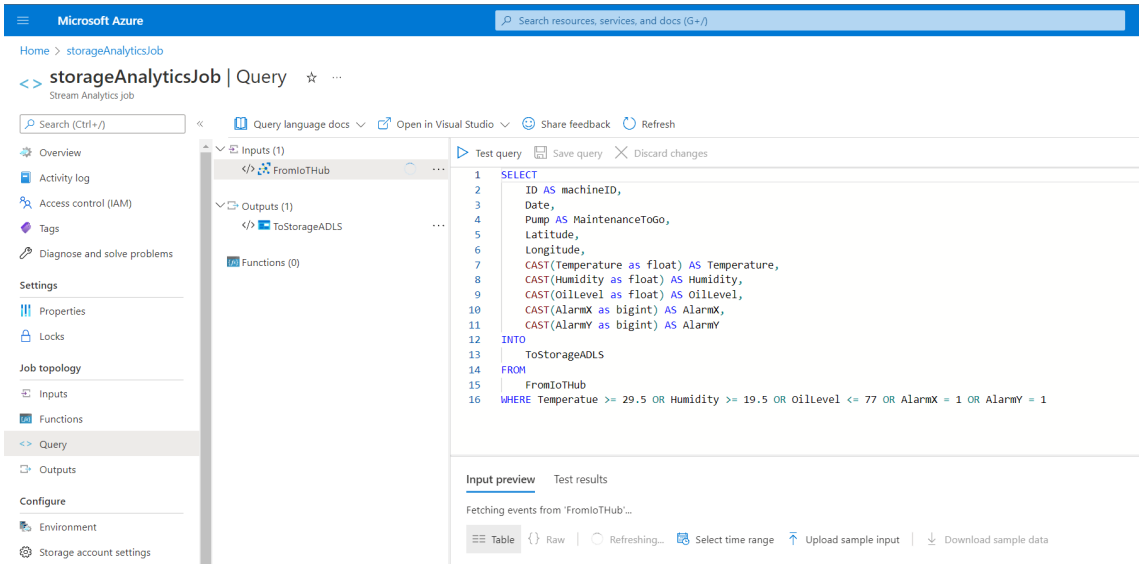


Figura 4.14: *Job* de enviar dados para a base de dados

Estes dados apenas são armazenados quando uma das cinco condições, esquema da figura 4.15,

são cumpridas, temperatura superior ou igual a 29,5 graus Celsius, humidade igual ou superior a 19,5%, nível de óleo abaixo de 77% e um dos dois alarmes estiver ativo, ou seja, igual a 1.

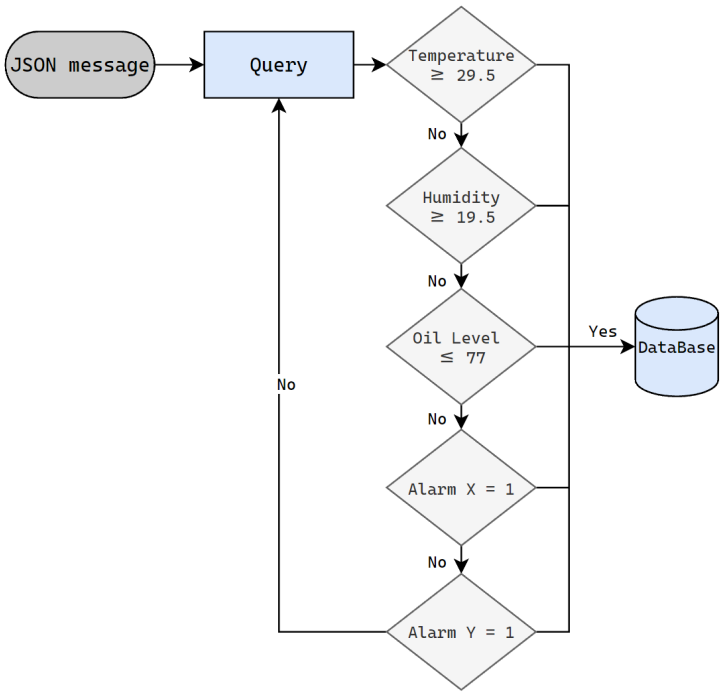


Figura 4.15: Diagrama de condições para armazenamento de alarmes na base de dados

4.4.4 *Storage Account*

Para obtermos o armazenamento de objetos (*blob storage* e desempenho para analisar grande volume de dados foi ativado a hierarquia de espaços no campo da figura 4.16 no momento de criação do recurso *storage account*. Esta funcionalidade permite estruturar todos os objetos/ficheiros em diretórios e subdiretórios, tornando o armazenamento mais organizado.

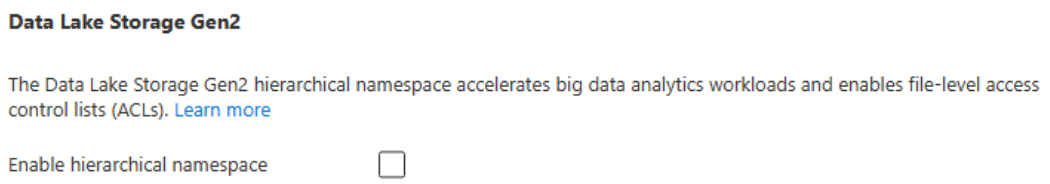


Figura 4.16: Ativar Data Lake Storage Gen2

Após o serviço de retenção de dados ser gerado foi necessário criar um *container*, que neste contexto, é como se fosse um diretório do nosso sistema de ficheiros convencional. Dentro dos *containers* são armazenados, em ficheiros, os dados enviados pelo *job*, previamente mencionado,

no *Stream Analytics*. Posteriormente a visão da base de dados não estruturados no portal Azure é a seguinte da figura 4.17.

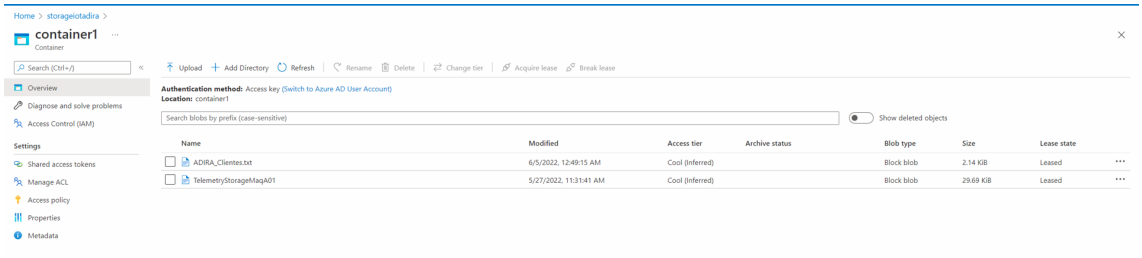


Figura 4.17: UI do portal Azure dos ficheiros armazenados

Na figura 4.17 é possível observar dois ficheiros.

O *TelemetryStorageMaqA01* é o ficheiro de retenção de alarmes e de telemetria fora dos valores padrão, no anexo C está presente um excerto dos dados gerados. Os dados são guardados em formato JSON.

O ficheiro *ADIRA_Clientes*, em anexo D, foi inserido na *cloud* manualmente com informação de clientes (de teste) para ser criado um *dashboard* adicional com dados de vendas. A estrutura dos dados deste ficheiro é JSON e tem informação do nome do cliente, o tipo e número de máquina comprada, preço, ID, data, localização, latitude e longitude.

4.5 *Dashboard*

Usando o serviço Power BI da Microsoft, foram criados dois relatórios na aplicação Power BI Desktop, um com a informação da telemetria de cada máquina e outro com informação de vendas da empresa. Estes relatórios permitem transformar-se em *dashboards* de monitorização devido à baixa taxa de atualização que podemos alcançar, sendo o mínimo de 1 segundo para o modo de dados *Live Connection*. *Live Connection* consiste em ir buscar informação a um banco de dados, neste caso o *dataset* criado pelo *job* de envio de telemetria do *Stream Analytics*.

Estes relatórios, como já referido no capítulo 2 secção 2.6.4.2, são construidos usando a linguagem DAX, idêntica à utilizada na ferramenta Excel, que permite manipular e criar informação com base nos dados presentes no *dataset*.

A seguir a informação é utilizada para popular gráficos, tabelas, visuais personalizados e compartilhados pela comunidade, entre outros.

4.5.1 *Dashboard* de telemetria

Na figura 4.18 está ilustrado o *dashboard* de telemetria. Neste, para cada máquina conseguimos constatar o número de horas até ser recomendado um intervenção de manutenção/revisão de maneira a prevenir avarias e possíveis falhas nas máquinas. Conjuntamente é possível ver os dados

atuais de temperatura, nível de óleo e humidade numericamente e em gráfico temporal. Também podemos observar o estado dos alarmes e o seu histórico.

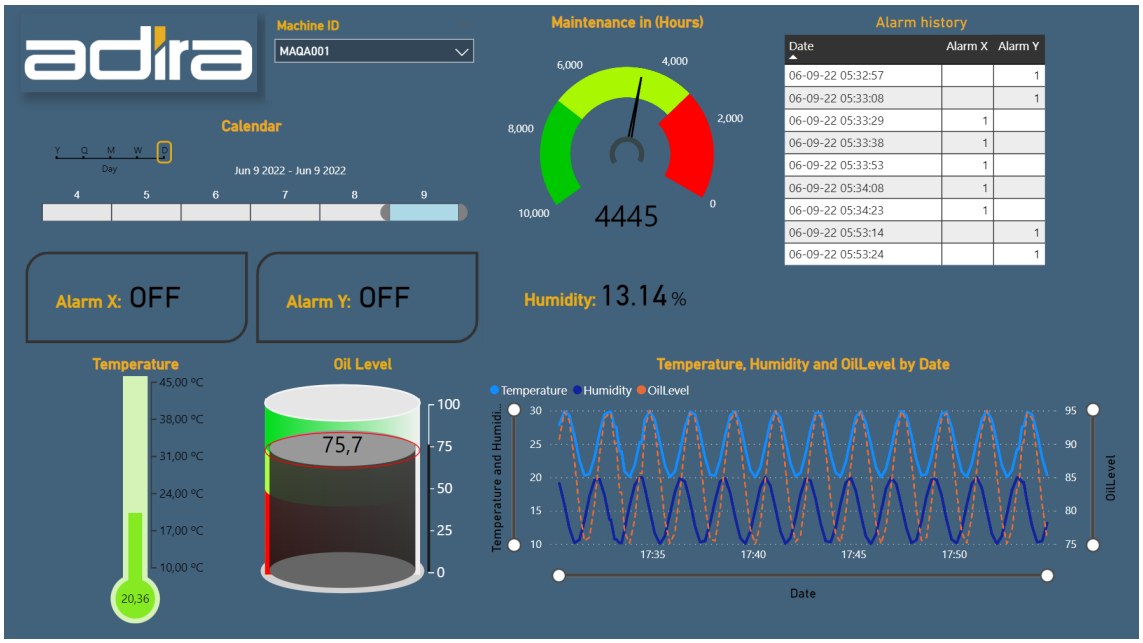


Figura 4.18: *Dashboard* de telemetria

No canto superior esquerdo do *dashboard* podemos filtrar pelo ID da máquina que queremos observar, bem como filtrar o grupo de dados para o dia, semana, mês, quadrimestre e ano. Interagindo no gráfico ou tabela de histórico de alarmes, todos os visuais irão mostrar a telemetria para aquele exato segundo, exemplo figura 4.19.

Figura 4.19: *Dashboard* de telemetria com filtro por ID e registo da tabela de alarmes

4.5.2 *Dashboard* de vendas

O *dashboard* de vendas de máquinas para clientes, figura 4.20, como referido na tabela de requisitos **??** do capítulo 3, não é de caráter obrigatório, foi uma ideia própria de valorizar este trabalho. A ideia foi retirada de um *mockup* apresentado pela própria empresa de um exemplo que houvera pedido anteriormente a outra empresa que desenvolve estes serviços.

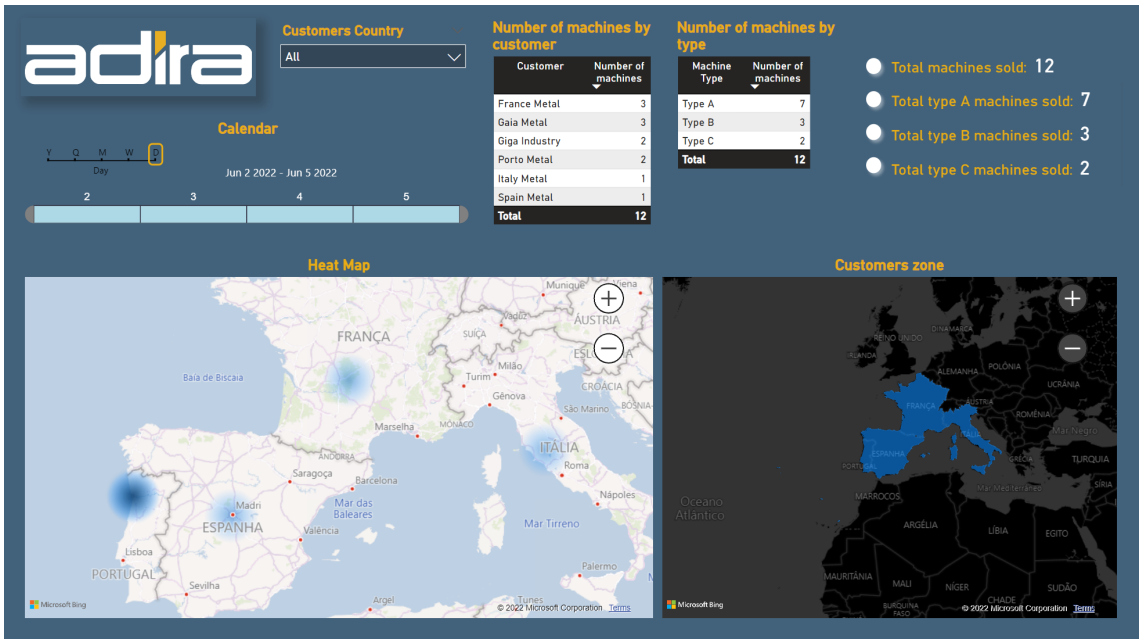


Figura 4.20: *Dashboard* de vendas

Neste *dashboard* é possível visualizar as zonas geográficas nas quais foram vendidas máquinas, em que quando mais escuro são os pontos no mapa mais máquinas estão presentes, bem como todos os países clientes. Também é visível estatísticas de total de vendas por cada tipo de máquina, tabela com os clientes e número de máquinas vendidas para os mesmos. Tal como o *dashboard* da telemetria, este é interativo, ou seja, ao selecionar a data apenas irá mostrar os dados daqueles dias, se selecionar um país ou clientes irá mostrar os dados daquele país ou clientes, exemplo figura 4.21.

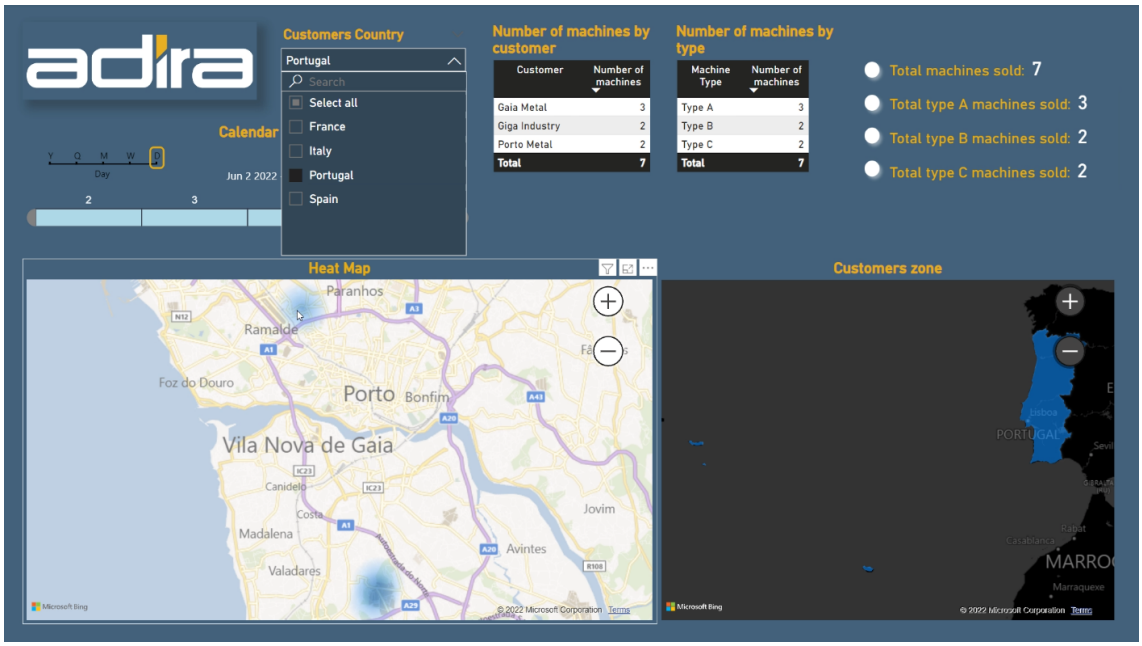


Figura 4.21: *Dashboard* de vendas com filtro de país

Como referido anteriormente na secção 4.4.4 da página 38, os dados de clientes e vendas para realizar este relatório dinâmico são valores experimentais, tendo sido inseridos manualmente na base de dados. Que tal como os dados de telemetria são enviados por um *job* do *Stream Analytics* para um *dataset*, no Power BI, que irá utilizado para popular o *dashboard*.

Capítulo 5

Conclusão

5.1 Testes e resultados

Os testes foram efetuados num computador com sistema operativo Windows 11 Pro com Docker versão 4.8.2, Node-RED v2.2.0 e Node.js v16.14.0. Para o teste de correr o *container* do programa principal em Linux foi usado um Raspberry Pi 4 modelo B de 2Gb RAM.

Os dados de telemetria utilizados, como referido na secção 4.2, foram 8 variáveis do servidor OPC UA que após o tratamento no programa principal cada mensagem, enviada para a *cloud*, tem um tamanho aproximadamente de 0,2 kB.

Para além de todos os testes realizados durante o desenvolvimento do trabalho, foi efetuado um último que consistiu em deixar 1h, sem interrupção, duas máquinas (simuladas) enviar dados para popular o *dashboard* de telemetria. Seguidamente o teste de interromper a ligação entre o servidor e o cliente OPC UA para testar a robustez do programa principal em ficar em ”espera” até haver uma nova conexão entre os mesmos.

O *dashboard* das vendas dos clientes foi ensaiado com valores aleatórios inseridos manualmente na base dados no serviço Azure Data Lake.

Com o intuito de avaliar a desempenho deste trabalho e se se encontra nos conformes e expectativas da empresa, foram testados todos os requisitos da tabela 3.1 da secção 3.1. Abaixo é possível ver os testes realizados para cada requisito:

- RF01 foi cumprido e o testes realizados foi num primeiro momento fazer *print* dos valores de telemetria recebidos pelo cliente OPC UA;
- RF02 foi cumprido e o teste realizado foi visualizar a telemetria enviada pelo servidor OPC UA.
- RF03 foi cumprido e o teste realizado foi visualizar os dados de vendas de cada cliente presentes na base de dados.
- RF04 foi cumprido e o teste realizado foi visualizar os dados recebidos no IoT Hub.
- RF05 foi cumprido e o teste realizado foi ver a telemetria armazenada na base de dados.

- RNF01 foi cumprido e o teste foi correr o *container* do programa num computador Windows.
- RNF02 foi cumprido e o teste foi correr o *container* do programa num Raspberry Pi 4 B com OS Linux.
- RNF03 não testado.
- RNF04 foi cumprido e o teste foi visualizar os relatórios no Power BI *online*.

Como é possível conferir, sucintamente, na tabela 5.1 foram testados e cumpridos 8 dos 9 requisitos. O requisito RNF03 não foi testado devido a não ter sido possível obter uma subscrição *premium* do Power BI, contudo pela documentação oficial da Microsoft para o tipo de armazenamento por *datasets* (*connected live*) a taxa de atualização mínima é de 1 segundo para capacidades dedicadas [58].

Requisito	**Descrição**	**Resultado**
RF01	Aceder aos valores dos sensores, eventos e alarmes da máquina. (servidor OPC UA)	Testado e cumprido
RF02	*Dashboard* com os valores dos sensores, eventos e alarmes da máquina.	Testado e cumprido
RF03	*Dashboard* com vendas e clientes.	Testado e cumprido
RF04	Comunicação unidirecional, da máquina para cloud.	Testado e cumprido
RF05	Guardar os alarmes e eventos da máquina numa base de dados.	Testado e cumprido
RNF01	Programa a correr numa unidade computacional externa. (Windows)	Testado e cumprido
RNF02	Programa a correr numa unidade computacional independentemente do OS.	Testado e cumprido
RNF03	Taxa de atualização do dashboard quase em tempo real, <= 10 segundos.	Não testado
RNF04	*Dashboard* acessíveis via web.	Testado e cumprido

Tabela 5.1: Tabela de requisitos e resultado dos mesmos.

5.2 Recomendações e Trabalho Futuro

Como trabalho futuro fica desenvolver, caso necessário, uma versão Linux da aplicação de configuração, implementar algumas medidas de segurança como, por exemplo, autenticação encriptada entre servidor e cliente OPC UA e testar a taxa de atualização do *dashboard* com uma subscrição premium do PowerBI.

É recomendado implementar, em caso de haver um maior número de máquinas ou até mesmo para uma só, Kubernetes, para orquestrar os *container* e assim termos uma maior robustez no caso de, por alguma razão, o *container* ”crashar”, este ser removido e ser levantado um novo, assegurando assim, de forma rápida e eficaz, o funcionamento de todo o processo. Também recomendo

adicionar um sistema de notificação para quando uma máquina estiver próxima do período de manutenção ou sair dos valores padrão a empresa ser notificada e proceder de acordo.

5.3 Conclusão

Em jeito de conclusão, a manutenção preditiva e a monitorização proativa das máquinas traduz-se num aumento da produtividade e previne custos maiores em avarias.

Nesta dissertação foram estudadas diversas soluções de plataformas IoT, desde os três maiores fornecedores de serviços *cloud* até plataformas *open source*. Este projeto, na sua totalidade, foi uma grande aprendizagem devido a terem sido utilizadas tecnologias que nunca havia utilizado, tais como Docker, Node-RED, Windows Forms, *cloud* e PowerBI.

O estudo e pesquisa das arquiteturas compostas pelos serviços da Azure e AWS, devido à elevada oferta pelos mesmo fornecedores, foram os maiores contribuintes da curva de aprendizagem.

Como já referido, foram testados oito dos nove requisitos e estes foram alcançados com sucesso. O requisito RNF03 não foi possível ensaiar devido à falta de licença premium do PowerBI. Contudo, pela documentação disponibilizada pela Microsoft podemos constatar que a taxa mínima de atualização para a arquitetura usada (*Live Connection*) é de 1 segundo, valor este bem inferior ao pretendido (10 segundos).

Com isto, acrescentando o parecer extremamente positivo dado pela empresa, podemos concluir que este trabalho foi realizado com sucesso.

Anexo A

Estimativas de custo dos serviços *Cloud*

A.1 Solução A - Microsoft Azure Cloud + PowerBI

Dispositivos: 2			
Serviços	**Pacote**		**Preço (€)**
IoT Hub	8k Mensagens/dia Grátis	0,5KB/msg	0
Stream Analytics	1 Streaming Unit	0,113€/h	27.12
ADLS Database	Primeiros 50TB	0.00935€/GB	≈ 0
Power BI	Premium	18.7€/utilizador	18.70
Total			**45.84**

Tabela A.1: Estimativa da Solução A para 2 dispositivos

Dispositivos: 5			
Serviços	**Pacote**		**Preço (€)**
IoT Hub	Escalão Basic B1 400k msg/dia	4KB/msg	9.34
Stream Analytics	1 Streaming Unit	0,113€/h	27.12
ADLS Database	Primeiros 50TB	0.00935€/GB	≈ 0
Power BI	Premium	18.7€/utilizador	18.70
Total			**55.18**

Tabela A.2: Estimativa da Solução A para 5 dspositivos

Dispositivos: 25			
Serviços	**Pacote**		**Preço (€)**
IoT Hub	Escalão Basic B1 400k msg/dia	4KB/msg	9.34
Stream Analytics	1 Streaming Unit	0,113€/h	27.12
ADLS Database	Primeiros 50TB	0.00935€/GB	≈ 0
Power BI	Premium	18.7€/utilizador	18.70
Total			**55.18**

Tabela A.3: Estimativa da Solução A para 25 dispositivos

A.2 Solução B - Amazon AWS Cloud

Dispositivos: 2		
Serviços	**Descrição**	**Preço (€)**
IoT Core	2016 Msg/dispositivo por mês	0.01
IoT Analytics	12.5MB/dispositivo por mês	0.16
Amazon S3	S3 Intelligent-Tiering 0.005€/GB	≈ 0
Amazon Quicksight	1 autor e 1 leitor	23.72
Total		**23.89**

Tabela A.4: Estimativa da Solução B para 2 dispositivos

Dispositivos: 5		
Serviços	**Descrição**	**Preço (€)**
IoT Core	2016 Msg/dispositivo por mês	0.03
IoT Analytics	12.5MB/dispositivo por mês	0.75
Amazon S3	S3 Intelligent-Tiering 0.005€/GB	≈ 0
Amazon Quicksight	1 autor e 1 leitor	23.72
Total		**24.5**

Tabela A.5: Estimativa da Solução B para 5 dispositivos

Dispositivos: 25		
Serviços	**Descrição**	**Preço (€)**
IoT Core	2016 Msg/dispositivo por mês	0.18
IoT Analytics	12.5MB/dispositivo por mês	3.74
Amazon S3	S3 Intelligent-Tiering 0.005€/GB	≈ 0
Amazon Quicksight	1 autor e 1 leitor	23.72
Total		**27.64**

Tabela A.6: Estimativa da Solução B para 25 dispositivos

A.3 Solução D - Thingsboard *Self Managed* + VM Azure

Serviços	**Descrição**	**Preço (€)**
Subscrição Maker	até 10 dispositivos	9.1
Instância EC2	D2pls v5 (2 vCPUs, 4GB RAM) + 32GB HDD	32.65
Total		**41.75**
Por dispositivo		**4.18**

Tabela A.7: Estimativa da Solução D para até 10 dispositivos

Serviços	**Descrição**	**Preço (€)**
Subscrição Prototype	até 100 dispositivos	90.09
Instância EC2	D4pls v5 (4 vCPUs, 8GB RAM) + 32GB HDD	63.86
Total		**153.95**
Por dispositivo		**1.54**

Tabela A.8: Estimativa da Solução D para até 100 dispositivos

Serviços	**Descrição**	**Preço (€)**
Subscrição Startup	até 500 dispositivos	181.09
Instância EC2	D4pls v5 (4 vCPUs, 8GB RAM) + 32GB HDD	63.86
Total		**244.95**
Por dispositivo		**0.49**

Tabela A.9: Estimativa da Solução D para até 500 dispositivos

	Subscrição		
Dispositivos	**Maker**	**Prototype**	**Startup**
2	8.35 €	3.08 €	0.98 €
5	20.88 €	7.70 €	2.45 €
25	—	38.49 €	12.25 €

Tabela A.10: Estimativa da Solução D dependento da subscrição

A.4 Solução E - Thingsboard *Self Managed* + VM AWS

Serviços	**Descrição**	**Preço (€)**
Subscrição Maker	até 10 dispositivos	9.1
Instância EC2	t2.medium (2 vCPUs, 4GB RAM) + 25GB HDD	26.36
Total		**35.46**
Por dispositivo		**3.55**

Tabela A.11: Estimativa da Solução E para até 10 dispositivos

Serviços	**Descrição**	**Preço (€)**
Subscrição Protoype	até 100 dispositivos	90.09
Instância EC2	a1.xlarge (4 vCPUs, 8GB RAM) + 35GB HDD	49.22
Total		**139.31**
Por dispositivo		**1.39**

Tabela A.12: Estimativa da Solução E para até 100 dispositivos

Serviços	**Descrição**	**Preço (€)**
Subscrição Startup	até 500 dispositivos	181.09
Instância EC2	a1.xlarge (4 vCPUs, 8GB RAM) + 35GB HDD	49.22
Total		**230.31**
Por dispositivo		**0.46**

Tabela A.13: Estimativa da Solução E para até 500 dispositivos

	Subscrição		
Dispositivos	**Maker**	**Prototype**	**Startup**
2	7.09 €	2.79 €	0.92 €
5	17.73 €	6.97 €	2.30 €
25	—	34.83 €	11.52 €

Tabela A.14: Estimativa da Solução E dependento da subscrição

Anexo B

Código servidor OPC UA

```
function constructAlarmAddressSpace(server, addressSpace, eventObjects, done) {


  const opcua = coreServer.choreCompact.opcua;
  const LocalizedText = opcua.LocalizedText;
  const namespace = addressSpace.getOwnNamespace();

  const Variant = opcua.Variant;
  const DataType = opcua.DataType;
  const DataValue = opcua.DataValue;

  var flexServerInternals = this;

  this.sandboxFlowContext.set("machine1State", false);
  this.sandboxFlowContext.set("machine1Oil", 0);
  this.sandboxFlowContext.set("machine1Temperature", 0);
  this.sandboxFlowContext.set("machine1Humidity", 0);
  this.sandboxFlowContext.set("machine1AlarmFlagY", false);
  this.sandboxFlowContext.set("machine1AlarmFlagX", false);
  this.sandboxFlowContext.set("machine1ID", "MAQA001");
  this.sandboxFlowContext.set("machine1Pump", 0);


  coreServer.debugLog("init dynamic address space");
  const rootFolder = addressSpace.findNode("RootFolder");

  node.warn("construct new address space for OPC UA");

  const machine1Folder = namespace.addFolder(rootFolder.objects, {"browseName": "
    Machine 1" });

  const machine1State = namespace.addVariable({
    "organizedBy": machine1Folder,
    "browseName": "State",
    "nodeId": "ns=1;s=Machine1_State",
```

```
35    "dataType": "Boolean",
36    "value": {
37      "get": function() {
38        return new Variant({
39          "dataType": DataType.Boolean,
40          "value": flexServerInternals.sandboxFlowContext.get("machine1State")
41        });
42      },
43      "set": function(variant) {
44        flexServerInternals.sandboxFlowContext.set(
45          "machine1State",
46          parseFloat(variant.value)
47        );
48        return opcua.StatusCodes.Good;
49      }
50    }
51    });
52
53  const machine1Level = namespace.addVariable({
54    "organizedBy": machine1Folder,
55    "browseName": "Oil Level",
56    "nodeId": "ns=1;s=Machine1_Oil",
57    "dataType": "Float",
58    "value": {
59      "get": function() {
60        return new Variant({
61          "dataType": DataType.Float,
62          "value": flexServerInternals.sandboxFlowContext.get("machine1Oil")
63        });
64      },
65      "set": function(variant) {
66        flexServerInternals.sandboxFlowContext.set(
67          "machine1Oil",
68          parseFloat(variant.value)
69        );
70        return opcua.StatusCodes.Good;
71      }
72    }
73  });
74
75    const machine1Temperature = namespace.addVariable({
76    "organizedBy": machine1Folder,
77    "browseName": "Temperature",
78    "nodeId": "ns=1;s=Machine1_Temperature",
79    "dataType": "Float",
80    "value": {
81      "get": function() {
82        return new Variant({
83          "dataType": DataType.Float,
```

```
          "value": flexServerInternals.sandboxFlowContext.get("machine1Temperature")
    )
        });
      },
      "set": function(variant) {
        flexServerInternals.sandboxFlowContext.set(
          "machine1Temperature",
          parseFloat(variant.value)
        );
        return opcua.StatusCodes.Good;
      }
    }
  });

    const machine1Humidity = namespace.addVariable({
    "organizedBy": machine1Folder,
    "browseName": "Humidity",
    "nodeId": "ns=1;s=Machine1_Humidity",
    "dataType": "Float",
    "value": {
      "get": function() {
        return new Variant({
          "dataType": DataType.Float,
          "value": flexServerInternals.sandboxFlowContext.get("machine1Humidity")
        });
      },
      "set": function(variant) {
        flexServerInternals.sandboxFlowContext.set(
          "machine1Humidity",
          parseFloat(variant.value)
        );
        return opcua.StatusCodes.Good;
      }
    }
  });

    const machine1AlarmFlagY = namespace.addVariable({
    "organizedBy": machine1Folder,
    "browseName": "Alarm FlagY",
    "nodeId": "ns=1;s=Machine1_AlarmFlagY",
    "dataType": "Boolean",
    "value": {
      "get": function() {
        return new Variant({
          "dataType": DataType.Boolean,
          "value": flexServerInternals.sandboxFlowContext.get("machine1AlarmFlagY")
        });
      },
      "set": function(variant) {
```

```
132         flexServerInternals.sandboxFlowContext.set(
133           "machine1AlarmFlagY",
134           parseFloat(variant.value)
135         );
136         return opcua.StatusCodes.Good;
137       }
138     }
139     });
140
141     const machine1AlarmFlagX = namespace.addVariable({
142     "organizedBy": machine1Folder,
143     "browseName": "Alarm FlagX",
144     "nodeId": "ns=1;s=Machine1_AlarmFlagX",
145     "dataType": "Boolean",
146     "value": {
147       "get": function() {
148         return new Variant({
149           "dataType": DataType.Boolean,
150           "value": flexServerInternals.sandboxFlowContext.get("machine1AlarmFlagX")
151         });
152       },
153       "set": function(variant) {
154         flexServerInternals.sandboxFlowContext.set(
155           "machine1AlarmFlagX",
156           parseFloat(variant.value)
157         );
158         return opcua.StatusCodes.Good;
159       }
160     }
161     });
162
163   const machine1ID = namespace.addVariable({
164     "organizedBy": machine1Folder,
165     "browseName": "ID",
166     "nodeId": "ns=1;s=Machine1_ID",
167     "dataType": "String",
168     "value": {
169       "get": function() {
170         return new Variant({
171           "dataType": DataType.String,
172           "value": flexServerInternals.sandboxFlowContext.get("machine1ID")
173         });
174       },
175       "set": function(variant) {
176         flexServerInternals.sandboxFlowContext.set(
177           "machine1ID",
178           parseFloat(variant.value)
179         );
180         return opcua.StatusCodes.Good;
```

```
      }
    }
  });

  const machine1Pump = namespace.addVariable({
    "organizedBy": machine1Folder,
    "browseName": "Pump",
    "nodeId": "ns=1;s=Machine1_Pump",
    "dataType": "Float",
    "value": {
      "get": function() {
        return new Variant({
          "dataType": DataType.Float,
          "value": flexServerInternals.sandboxFlowContext.get("machine1Pump")
        });
      },
      "set": function(variant) {
        flexServerInternals.sandboxFlowContext.set(
          "machine1Pump",
          parseFloat(variant.value)
        );
        return opcua.StatusCodes.Good;
      }
    }
  });

  coreServer.debugLog("create dynamic address space done");
  node.warn("construction of new address space for OPC UA done");

  done();
}
```

Listing B.1: Node-RED OPC UA Server

Anexo C

Ficheiro de telemetria armazenada


```
{"machineID":"MAQA001","Date":"05/27/2022 08:48:38","Latitude":41.09542815342197,"Longitude":-8.60779924652854,"Temperature":25.29,"Humidity":10.01,"OilLevel":85.59,"AlarmX":1,"AlarmY":0}
{"machineID":"MAQA001","Date":"05/27/2022 08:48:53","Latitude":41.09542815342197,"Longitude":-8.60779924652854,"Temperature":21.85,"Humidity":11.12,"OilLevel":78.69,"AlarmX":1,"AlarmY":0}
{"machineID":"MAQA001","Date":"05/27/2022 08:49:08","Latitude":41.09542815342197,"Longitude":-8.60779924652854,"Temperature":20.05,"Humidity":14.26,"OilLevel":75.11,"AlarmX":1,"AlarmY":0}
{"machineID":"MAQA001","Date":"05/27/2022 08:49:23","Latitude":41.09542815342197,"Longitude":-8.60779924652854,"Temperature":20.94,"Humidity":17.91,"OilLevel":76.87,"AlarmX":1,"AlarmY":0}
{"machineID":"MAQA001","Date":"05/27/2022 08:49:38","Latitude":41.09542815342197,"Longitude":-8.60779924652854,"Temperature":23.92,"Humidity":19.89,"OilLevel":82.94,"AlarmX":1,"AlarmY":0}
{"machineID":"MAQA001","Date":"05/27/2022 08:50:53","Latitude":41.09542815342197,"Longitude":-8.60779924652854,"Temperature":23.03,"Humidity":10.4,"OilLevel":81.06,"AlarmX":0,"AlarmY":1}
{"machineID":"MAQA001","Date":"05/27/2022 08:51:08","Latitude":41.09542815342197,"Longitude":-8.60779924652854,"Temperature":20.44,"Humidity":12.94,"OilLevel":75.89,"AlarmX":0,"AlarmY":1}
{"machineID":"MAQA001","Date":"05/27/2022 08:51:23","Latitude":41.09542815342197,"Longitude":-8.60779924652854,"Temperature":20.25,"Humidity":16.55,"OilLevel":75.49,"AlarmX":0,"AlarmY":1}
{"machineID":"MAQA001","Date":"05/27/2022 08:51:53","Latitude":41.09542815342197,"Longitude":-8.60779924652854,"Temperature":26.27,"Humidity":19.84,"OilLevel":87.53,"AlarmX":0,"AlarmY":0}
{"machineID":"MAQA001","Date":"05/27/2022 08:53:23","Latitude":41.09542815342197,"Longitude":-8.60779924652854,"Temperature":20.0,"Humidity":15.18,"OilLevel":75.01,"AlarmX":0,"AlarmY":0}
{"machineID":"MAQA001","Date":"05/27/2022 08:53:53","Latitude":41.09542815342197,"Longitude":-8.60779924652854,"Temperature":24.87,"Humidity":20.0,"OilLevel":84.75,"AlarmX":0,"AlarmY":0}
```

```
12 {"machineID":"MAQA001","Date":"05/27/2022 08:55:23","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.15,"Humidity":13.79,"OilLevel"
       :75.3,"AlarmX":0,"AlarmY":0}
13 {"machineID":"MAQA001","Date":"05/27/2022 08:55:38","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.68,"Humidity":17.52,"OilLevel"
       :76.36,"AlarmX":0,"AlarmY":0}
14 {"machineID":"MAQA001","Date":"05/27/2022 08:55:53","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":23.51,"Humidity":19.77,"OilLevel"
       :82.02,"AlarmX":0,"AlarmY":0}
15 {"machineID":"MAQA001","Date":"05/27/2022 08:56:08","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":27.11,"Humidity":19.53,"OilLevel"
       :89.23,"AlarmX":0,"AlarmY":0}
16 {"machineID":"MAQA001","Date":"05/27/2022 08:57:23","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.66,"Humidity":12.52,"OilLevel"
       :76.32,"AlarmX":0,"AlarmY":0}
17 {"machineID":"MAQA001","Date":"05/27/2022 08:57:38","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.12,"Humidity":16.1,"OilLevel"
       :75.24,"AlarmX":0,"AlarmY":0}
18 {"machineID":"MAQA001","Date":"05/27/2022 08:58:08","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":25.81,"Humidity":19.93,"OilLevel"
       :86.62,"AlarmX":0,"AlarmY":0}
19 {"machineID":"MAQA001","Date":"05/27/2022 08:59:38","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.01,"Humidity":14.77,"OilLevel"
       :75.01,"AlarmX":0,"AlarmY":0}
20 {"machineID":"MAQA001","Date":"05/27/2022 09:00:08","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":24.43,"Humidity":19.97,"OilLevel"
       :83.86,"AlarmX":0,"AlarmY":0}
21 {"machineID":"MAQA001","Date":"05/27/2022 09:01:38","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.27,"Humidity":13.38,"OilLevel"
       :75.54,"AlarmX":0,"AlarmY":0}
22 {"machineID":"MAQA001","Date":"05/27/2022 09:01:53","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.42,"Humidity":17.01,"OilLevel"
       :75.84,"AlarmX":0,"AlarmY":0}
23 {"machineID":"MAQA001","Date":"05/27/2022 09:02:08","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":23.07,"Humidity":19.61,"OilLevel"
       :81.13,"AlarmX":0,"AlarmY":0}
24 {"machineID":"MAQA001","Date":"05/27/2022 09:02:23","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":26.73,"Humidity":19.69,"OilLevel"
       :88.46,"AlarmX":0,"AlarmY":0}
25 {"machineID":"MAQA001","Date":"05/27/2022 09:03:38","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.91,"Humidity":12.13,"OilLevel"
       :76.81,"AlarmX":0,"AlarmY":0}
26 {"machineID":"MAQA001","Date":"05/27/2022 09:03:53","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.06,"Humidity":15.75,"OilLevel"
       :75.11,"AlarmX":0,"AlarmY":0}
27 {"machineID":"MAQA001","Date":"05/27/2022 09:04:23","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":25.31,"Humidity":19.99,"OilLevel"
       :85.62,"AlarmX":0,"AlarmY":0}
```

```
28 {"machineID":"MAQA001","Date":"05/27/2022 09:05:53","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.05,"Humidity":14.33,"OilLevel"
       :75.09,"AlarmX":0,"AlarmY":0}
29 {"machineID":"MAQA001","Date":"05/27/2022 09:06:08","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.89,"Humidity":17.85,"OilLevel"
       :76.78,"AlarmX":0,"AlarmY":0}
30 {"machineID":"MAQA001","Date":"05/27/2022 09:06:23","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":23.89,"Humidity":19.88,"OilLevel"
       :82.78,"AlarmX":0,"AlarmY":0}
31 {"machineID":"MAQA001","Date":"05/27/2022 09:07:53","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.44,"Humidity":12.95,"OilLevel"
       :75.88,"AlarmX":0,"AlarmY":0}
32 {"machineID":"MAQA001","Date":"05/27/2022 09:08:08","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.25,"Humidity":16.56,"OilLevel"
       :75.5,"AlarmX":0,"AlarmY":0}
33 {"machineID":"MAQA001","Date":"05/27/2022 09:08:38","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":26.22,"Humidity":19.85,"OilLevel"
       :87.44,"AlarmX":0,"AlarmY":0}
34 {"machineID":"MAQA001","Date":"05/27/2022 09:10:08","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.01,"Humidity":15.24,"OilLevel"
       :75.01,"AlarmX":0,"AlarmY":0}
35 {"machineID":"MAQA001","Date":"05/27/2022 09:10:38","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":24.87,"Humidity":20.0,"OilLevel"
       :84.75,"AlarmX":0,"AlarmY":0}
36 {"machineID":"MAQA001","Date":"05/27/2022 09:12:08","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.15,"Humidity":13.78,"OilLevel"
       :75.3,"AlarmX":0,"AlarmY":0}
37 {"machineID":"MAQA001","Date":"05/27/2022 09:12:23","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":20.65,"Humidity":17.46,"OilLevel"
       :76.3,"AlarmX":0,"AlarmY":0}
38 {"machineID":"MAQA001","Date":"05/27/2022 09:12:38","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":23.46,"Humidity":19.76,"OilLevel"
       :81.91,"AlarmX":0,"AlarmY":0}
39 {"machineID":"MAQA001","Date":"05/27/2022 09:12:53","Latitude":41.09542815342197,"
       Longitude":-8.60779924652854,"Temperature":27.16,"Humidity":19.51,"OilLevel"
       :89.32,"AlarmX":0,"AlarmY":0}
```


Listing C.1: Ficheiro TelemetryStorageMaqA01

Anexo D

Ficheiro de dados de clientes


```
{"Client":"Gaia Metal","MachineType":"Type A","NumberOfMachinesPurchased":"1","Price":"20000","machineID":"MAQA-v1","Date":"2022/06/4","Location":"Portugal","Latitude":41.09723417412844,"Longitude":-8.607355719376297}
{"Client":"Gaia Metal","MachineType":"Type A","NumberOfMachinesPurchased":"1","Price":"20000","machineID":"MAQA-v2","Date":"2022/06/2","Location":"Portugal","Latitude":41.09723417412844,"Longitude":-8.607355719376297}
{"Client":"Porto Metal","MachineType":"Type A","NumberOfMachinesPurchased":"1","Price":"40000","machineID":"MAQA-v3","Date":"2022/06/3","Location":"Portugal","Latitude":41.17518341702638,"Longitude":-8.636076977620599}
{"Client":"Porto Metal","MachineType":"Type B","NumberOfMachinesPurchased":"1","Price":"10000","machineID":"MAQB-v1","Date":"2022/06/4","Location":"Portugal","Latitude":41.17518341702638,"Longitude":-8.636076977620599}
{"Client":"Giga Industry","MachineType":"Type C","NumberOfMachinesPurchased":"1","Price":"50000","machineID":"MAQC-v1","Date":"2022/06/4","Location":"Portugal","Latitude":41.092679349564435,"Longitude":-8.609141250373497}
{"Client":"Giga Industry","MachineType":"Type C","NumberOfMachinesPurchased":"1","Price":"45000","machineID":"MAQC-v2","Date":"2022/06/5","Location":"Portugal","Latitude":41.092679349564435,"Longitude":-8.609141250373497}
{"Client":"Spain Metal","MachineType":"Type A","NumberOfMachinesPurchased":"1","Price":"35000","machineID":"MAQA-v4","Date":"2022/06/2","Location":"Spain","Latitude":40.39070206504304,"Longitude":-3.780580317662625}
{"Client":"Italy Metal","MachineType":"Type B","NumberOfMachinesPurchased":"1","Price":"20000","machineID":"MAQB-v2","Date":"2022/06/4","Location":"Italy","Latitude":42.81365110030954,"Longitude":11.683599543575266}
{"Client":"France Metal","MachineType":"Type A","NumberOfMachinesPurchased":"1","Price":"29000","machineID":"MAQA-v5","Date":"2022/06/5","Location":"France","Latitude":45.2794185199343,"Longitude":2.0175636697454795}
{"Client":"France Metal","MachineType":"Type A","NumberOfMachinesPurchased":"1","Price":"29000","machineID":"MAQA-v6","Date":"2022/06/5","Location":"France","Latitude":45.2794185199343,"Longitude":2.0175636697454795}
```


Listing D.1: Ficheiro ADIRA_Clientes.txt

Referências

[1] O que é a indústria 4.0 - 49 educação. https://49educacao.com.br/inovacao/industria-4-0/. (Accessed on 07/09/2023).

[2] ¿cuáles son los dispositivos iot? | registratuimei.cl. https://www.registratuimei.cl/blog/cuales-son-los-dispositivos-iot/. (Accessed on 07/10/2023).

[3] What is opc ua- quick guide & faq. https://page.advantech.com/en/global/industrial-automation/industrial-io/opc-ua#session2. (Accessed on 06/08/2023).

[4] What's the difference between vms & containers? | akf partners. https://akfpartners.com/growth-blog/vms-vs-containers. (Accessed on 06/01/2022).

[5] Introduction - learn | microsoft docs. https://docs.microsoft.com/en-us/learn/modules/introduction-to-iot-hub/1-introduction. (Accessed on 06/08/2022).

[6] What is power bi? - learn | microsoft docs. https://docs.microsoft.com/en-us/learn/modules/introduction-power-bi/2-what-power-bi. (Accessed on 06/14/2022).

[7] O que é o aws iot? - aws iot core. https://docs.aws.amazon.com/pt_br/iot/latest/developerguide/what-is-aws-iot.html. (Accessed on 07/28/2023).

[8] Using aws iot analytics to prepare data for quicksight time-series visualizations | the internet of things on aws – official blog. https://aws.amazon.com/pt/blogs/iot/using-aws-iot-analytics-to-prepare-data-for-quicksight-time-series-visualiz (Accessed on 07/28/2023).

[9] Amazon quicksight - business intelligence service - amazon web services. https://aws.amazon.com/pt/quicksight/. (Accessed on 07/28/2023).

[10] Node-red. https://nodered.org/. (Accessed on 05/30/2023).

[11] Node-red: Lecture 4 – a tour of the core nodes – node red programming guide. https://noderedguide.com/node-red-lecture-4-a-tour-of-the-core-nodes/. (Accessed on 05/30/2023).

[12] Chart: Big three dominate the global cloud market | statista. https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/. (Accessed on 06/18/2023).

[13] Pricing | thingsboard. https://thingsboard.io/pricing/?product=thingsboard-pe. (Accessed on 06/25/2023).

[14] Use the azure portal to create an iot hub | microsoft learn. https://learn.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal. (Accessed on 03/30/2023).

[15] A iot - um dos grandes aliados da indústria 4.0 - manusis4. https://manusis4.com/pt-br/iot-e-industria-4-0/. (Accessed on 07/26/2023).

[16] Adira – wikipédia, a enciclopédia livre. https://pt.wikipedia.org/wiki?curid=6925579. (Accessed on 07/26/2023).

[17] The 4 industrial revolutions - institute of entrepreneurship development. https://ied.eu/project-updates/the-4-industrial-revolutions/. (Accessed on 07/11/2023).

[18] Ruudi Sakurai e Jederson Donizete Zuchi. REVOLUçõES INDUSTRIAIS ATé a INDUSTRIA 4.0. *Revista Interface Tecnológica*, 15(2):480–491, Dezembro 2018. URL: https://doi.org/10.31510/infa.v15i2.386, doi:10.31510/infa.v15i2.386.

[19] What is industry 4.0 and how does it work? | ibm. https://www.ibm.com/topics/industry-4-0. (Accessed on 07/26/2023).

[20] O que é a indústria 4.0 | oracle brasil. https://www.oracle.com/br/scm/manufacturing/what-is-manufacturing/what-is-industry-4-0/. (Accessed on 07/09/2023).

[21] O que é iot (internet das coisas)? | oracle brasil. https://www.oracle.com/br/internet-of-things/what-is-iot/. (Accessed on 07/09/2023).

[22] O que é iot – explicação sobre a internet das coisas – aws. https://aws.amazon.com/pt/what-is/iot/. (Accessed on 07/09/2023).

[23] Protocolos de iot internet das coisas - mqtt - aplicando os protocolos de iot para digitalização em sistemas de automação industrial | linkedin. https://www.linkedin.com/pulse/protocolos-de-iot-internet-das-coisas-mqtt-os-para-em-venturelli/?trk=pulse-article_more-articles_related-content-card&originalSubdomain=pt. (Accessed on 07/04/2022).

[24] Scott M Lewandowski. Frameworks for component-based client/server computing. *ACM Computing Surveys (CSUR)*, 30(1):3–27, 1998.

[25] Urs Hunkeler, Hong Linh Truong, e Andy Stanford-Clark. Mqtt-s — a publish/subscribe protocol for wireless sensor networks. Em *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, páginas 791–798, 2008. doi:10.1109/COMSWA.2008.4554519.

[26] Bharati Wukkadada, Kirti Wankhede, Ramith Nambiar, e Amala Nair. Comparison with http and mqtt in internet of things (iot). Em *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, páginas 249–253, 2018. doi:10.1109/ICIRCA.2018.8597401.

[27] Cavide Balkı Gemirter, Çağatay Şenturca, e Şebnem Baydere. A comparative evaluation of amqp, mqtt and http protocols using real-time public smart city data. Em *2021 6th International Conference on Computer Science and Engineering (UBMK)*, páginas 542–547, 2021. doi:10.1109/UBMK52708.2021.9559032.

[28] Nitin Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. Em *2017 IEEE International Systems Engineering Symposium (ISSE)*, páginas 1–7, 2017. doi:10.1109/SysEng.2017.8088251.

[29] Hasan Derhamy, Jesper Rönnholm, Jerker Delsing, Jens Eliasson, e Jan van Deventer. Protocol interoperability of opc ua in service oriented architectures. Em *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, páginas 44–50, 2017. doi:10.1109/INDIN.2017.8104744.

[30] What is opc? - opc foundation. https://opcfoundation.org/about/what-is-opc/. (Accessed on 06/08/2023).

[31] Sten Grüner, Julius Pfrommer, e Florian Palm. Restful industrial communication with opc ua. *IEEE Transactions on Industrial Informatics*, 12(5):1832–1841, 2016. doi:10.1109/TII.2016.2530404.

[32] The benefits of containerization and what it means for you | ibm. https://www.ibm.com/cloud/blog/the-benefits-of-containerization-and-what-it-means-for-you. (Accessed on 05/27/2022).

[33] What are containers and why do you need them? https://www.cio.com/article/247005/what-are-containers-and-why-do-you-need-them.html. (Accessed on 05/26/2022).

[34] Containers vs. virtual machines (vms): What's the difference? | ibm. https://www.ibm.com/cloud/blog/containers-vs-vms. (Accessed on 06/01/2022).

[35] O que é cloud computing (computação em nuvem)? - amazon web services. https://aws.amazon.com/pt/what-is-cloud-computing/. (Accessed on 07/27/2023).

[36] Cloud computing: o que é, para que serve e como usá-la? https://rockcontent.com/br/blog/cloud-computing/. (Accessed on 07/27/2023).

[37] 13 vantagens da computação em nuvem. | blog sydle. https://www.sydle.com/br/blog/vantagens-computacao-em-nuvem-6155bf940e94dd1b0f13489e. (Accessed on 07/27/2023).

[38] O que é o azure — serviços cloud da microsoft | microsoft azure. https://azure.microsoft.com/pt-pt/overview/what-is-azure/. (Accessed on 06/18/2022).

[39] What is iot hub? - learn | microsoft docs. https://docs.microsoft.com/en-us/learn/modules/introduction-to-iot-hub/2-what-is-iot-hub. (Accessed on 06/08/2022).

[40] Introduction to azure stream analytics | microsoft docs. https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction. (Accessed on 06/09/2022).

[41] Azure data lake storage gen2 introduction | microsoft docs. https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-introduction. (Accessed on 06/09/2022).

[42] Aprender as noções básicas de dax no power bi desktop - power bi | microsoft learn. https://learn.microsoft.com/pt-pt/power-bi/transform-model/desktop-quickstart-learn-dax-basics. (Accessed on 04/03/2023).

[43] Report design ideas in power bi - databear powerbi training. https://databear.com/report-design-ideas-in-power-bi/. (Accessed on 06/16/2022).

[44] Integração dynamics nav 2016 e power bi • hydra it. https://www.hydra.pt/2016/04/08/integracao-dynamics-nav-2016-e-power-bi/. (Accessed on 06/16/2022).

[45] Visão geral do aws iot analytics – amazon web services. https://aws.amazon.com/pt/iot-analytics/. (Accessed on 07/28/2023).

[46] Armazenamento s3 - simple storage service - amazon web services. https://aws.amazon.com/pt/s3/. (Accessed on 07/28/2023).

[47] O que é o amazon quicksight? - amazon quicksight. https://docs.aws.amazon.com/pt_br/quicksight/latest/user/welcome.html. (Accessed on 07/28/2023).

[48] Node-red: Lecture 1 – a brief introduction to node-red – node red programming guide. https://noderedguide.com/nr-lecture-1/. (Accessed on 05/30/2023).

[49] Top open-source iot platforms in 2023. https://bytebeam.io/blog/top-open-source-iot-platforms/#top-open-source-iot-platforms-in-2023. (Accessed on 06/18/2023).

[50] Documentação do google cloud iot core | documentação do cloud iot core. https://cloud.google.com/iot/docs?hl=pt-br. (Accessed on 06/18/2023).

[51] Protocolos e portas de comunicação do azure iot hub | microsoft learn. https://learn.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-protocols. (Accessed on 05/25/2023).

[52] System namespace | microsoft learn. https://learn.microsoft.com/en-us/dotnet/api/system?view=net-5.0. (Accessed on 05/14/2023).

[53] Microsoft.azure.devices.client namespace - azure for .net developers | microsoft learn. https://learn.microsoft.com/en-us/dotnet/api/microsoft.azure.devices.client?view=azure-dotnet. (Accessed on 05/14/2023).

[54] Json.net - newtonsoft. https://www.newtonsoft.com/json. (Accessed on 05/14/2023).

[55] Opc.uafx.client namespace - traeger docs. https://docs.traeger.de/en/software/sdk/opc-ua/net/api/opc.uafx.client. (Accessed on 05/14/2023).

[56] Pricing—iot hub | microsoft azure. https://azure.microsoft.com/en-us/pricing/details/iot-hub/. (Accessed on 05/16/2023).

[57] Pricing - stream analytics | microsoft azure. https://azure.microsoft.com/en-us/pricing/details/stream-analytics/#pricing. (Accessed on 05/16/2023).

[58] Automatic page refresh in power bi desktop - power bi | microsoft learn. https://learn.microsoft.com/en-us/power-bi/create-reports/desktop-automatic-page-refresh. (Accessed on 07/15/2023).