INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Real Time Integrated Risk Assessment - Utilização de tecnologias IoT e análise preditiva para prevenir acidentes de trabalho.

DIOGO FILIPE ANTUNES VIGO
Julho de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Real Time Integrated Risk Assessment

Utilização de tecnologias *IoT* e análise preditiva para prevenir acidentes de trabalho

Diogo Filipe Antunes Vigo

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Paulo Manuel Baltarejo De Sousa

Porto, julho de 2019

À minha família, namorada e amigos, por todo o apoio durante a execução deste projeto.

Resumo

A presente dissertação pretende descrever todo o processo de desenvolvimento do projeto realizado no âmbito da unidade curricular de Tese/Dissertação/Estágio do Mestrado em Engenharia Informática, na área de especialização de Engenharia de *software.*

Este projeto, proposto pela empresa Abaco consultores, tem como objetivo principal desenvolver uma solução que consiga prevenir e evitar acidentes de trabalho em ambientes fabris. Através do uso de dispositivos com capacidade de ligação à rede, deve ser possível fazer o monitoramento e gestão de áreas e trabalhadores, controlando, sempre, questões ambientais que possam colocar em risco tanto material como vidas humanas.

Na solução inicial, este projeto estaria interligado com um serviço oferecido pela Abaco consultores denominado de *Incloud for SafeMed*, relacionando dados clínicos de trabalhadores com dados ambientais e físicos de um local. Para isso, o trabalhador teria de possuir um dispositivo que conseguisse ler alguns dados como frequência cardíaca ou temperatura corporal. Esta função acabou mais tarde por ser posta de parte, mudando o foco apenas para a criação de sistema de monitorização ambiental que em nada dependesse do *Incloud for SafeMed*. Assim, e depois de estudadas algumas tecnologias, foi desenvolvido um sistema capaz de apresentar aos clientes dados emitidos por dispositivos, através de uma aplicação *web* acessível em qualquer lugar. Este desenvolvimento teve como objetivo final responder à questão: É possível usar dispositivos inteligentes para evitar acidentes?

Depois de realizados os testes necessários conseguiu-se concluir que o sistema criado pode evitar certos acidentes. Porém, para a inclusão do mesmo no mercado de trabalho seria necessário efetuar mais testes em ambiente real. A inclusão destas tecnologias em ambientes fabris pode minimizar ou mesmo evitar estragos de maior dimensão aquando de um acidente.

No sentido de melhorar a solução final é ainda sugerido o uso de *machine learning* com os dados reais de leituras. Assim, além de reduzir acidentes, será possível, também, prever os mesmos e atuar de forma preventiva.

Palavras-chave: *IoT*, Saúde e segurança no trabalho, Big *data, Cloud.*

Abstract

The present dissertation intends to describe all the development of the carried out in the scope of the Thesis/Dissertation/Internship course of the Master's in Informatics Engineering, in the specialization of Software Engineering.

This project, proposed by Abaco Consultores, has the main objective of developing a solution that can both avoid and prevent work accidents in a factory environment. Using devices that have the possibility of connecting to the network, it should be able to monitor both workers and areas around them, controlling environmental factors that can risk humans and material safety.

In the early solution, this project would relate to the service offered by Abaco Consultores called Incloud for Safemed, relating workers clinical data with environmental and personal data. To do that, the worker had to possess a device that could read a few data like heart rate or body temperature. This function ended up being postponed, focusing only on creating an independent environmental monitoring system. To do that, and after studying a few technologies, it was developed a system with the capability of showing sensor data to the clients using a web app that can accessed in any place. The development of this system had the objective of answering to the question: Is it possible to avoid work accidents using smart devices?

Performing a few needed tests revealed that the system as it is would be able to prevent certain accidents. However, in order to prepare it to a real system, there is the need of doing a few more tests in this type of environment. The inclusion of these technologies in a factory environment can minimize or even avoid accidents damage on people and material.

In order to improve the final solution, it is mentioned to use machine learning with the data coming from the sensors. Using that it is possible not only to avoid accidents but also predict them and act preventively.

Keywords: IoT, Safety and Health at Work, Big Data, Cloud

Agradecimentos

Começar por agradecer à minha família e namorada por todo o apoio não só ao longo deste último ano, mas também por tudo que fizeram e continuam a fazer para poder chegar onde cheguei.

Ao Professor Dr. Paulo Baltarejo de Sousa, por todo o trabalho que teve comigo, pela tutoria e por todo o conhecimento que me transmitiu.

Aos meus colegas de licenciatura, mestrado e estágio que, sem eles, seria bem mais complicado chegar a este ponto.

Por fim agradecer ao *muy nobre* Instituto Superior de Engenharia do Porto, por me ter feito crescer e por me ter proporcionado tantas alegrias e noites mal dormidas.

Obrigado.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Amostras de código

Acrónimos e Símbolos

Lista de Acrónimos

ACT	Autoridade para as Condições do Trabalho
AT	Autoridade Tributária
ATEX	Atmosfera Explosiva
AWS	*Amazon Web Services*
BD	Base de dados
BLE	*Bluetooth Low Energy*
CoAP	*Constrained Application Protocol*
CPU	Unidade Central de Processamento
FC	Frequência Cardíaca
FFE	*Fuzzy Front End*
GCP	*Google Cloud Plataform*
HTTP	*Hyper text transfer protocol*
IA	Inteligência artificial
IIoT	*Industrial Internet of Things*
IoT	*Internet of Things* (Internet das coisas)
JWT	*JSON Web Token*
KB	*Kilobyte*
M2M	*Machine to Machine*
MQTT	*Message Queue telemetry transport*
NCD	*New Concept Development*
NPD	*New Product Development*
PaaS	*Platform as a Service*
QoS	*Quality of service*
RFID	*Radio Frequency Identifier*
RGPD	Regulamento Geral sobre a Proteção de Dados
RTO	*Recovery Time Objetive*
SaaS	*Software as a Service*
SDK	*Software Development Kits*
SPoF	*Single Point of Failure*

SST	Segurança e Saúde no Trabalho
TLS	*Transport Layer Security*
TMDEI	Tese / Dissertação / Estágio - TMDEI
UDP	*User Datagram Protocol*
VC	Valor para Cliente

1 Introdução

Neste capítulo é feita uma primeira abordagem ao contexto em que se insere esta dissertação. É também apresentado o problema abordado assim como os objetivos traçados para a execução do projeto. De seguida são apresentados os resultados esperados e é feita uma pequena abordagem à análise de valor. Por fim é analisada a abordagem preconizada e apresentada a estrutura do documento.

1.1 Contexto

A evolução das tecnologias tem vindo a aproximar, cada vez mais, os dispositivos eletrónicos entre si. A interação entre estes, pode agora, ser completamente autónoma, sem qualquer tipo de intervenção humana. Este crescimento deve-se, também, ao crescimento da *Internet of Things* (IoT)*,* dispositivos que, além das suas funções especificas, têm também a capacidade de conectividade a redes informáticas. Alguns exemplos são relógios, frigoríficos, sensores, etc. A ligação entre estes resulta numa produção de informação massiva, dificilmente prevista e mensurável. Em 2017 [1], estimou-se que haveriam cerca de 8.4 biliões de dispositivos ligados entre si à escala global e espera-se que em 2020 este número suba para os 30 biliões. Por esse ano é expectável, também, que a quantidade de informação esteja perto dos 44 *Zettabytes*[1] [2] ou 44 triliões de *Gigabytes*.

Apesar da criação massiva de informação, atualmente, apenas 1% da informação foi analisada e usada [3]. A este processo de criação, conservação e análise de informação, é dado o nome de *Big data,* tema que é debatido nos Capítulos que se seguem.

A aplicação destas tecnologias está associada a muitas áreas nomeadamente: medicina, serviços financeiros, venda em retalho, indústria automóvel e qualquer indústria com ambiente fabril [4]. Na atualidade já se verificam casos em que do uso de tecnologias IoT advém grandes benefícios para as empresas. Estes benefícios estão associados a vários campos como:

[1] 1 *ZettaByte* = 10^{21} *bytes* [68]

- **Fábricas digitais** – Através do uso de dispositivos ligados à rede, é possível gerir uma fábrica de forma remota;
- **Gestão de maquinaria** - O uso de sensores permite identificar em tempo real falhas de equipamento ou outras;
- **Gestão de inventário** - É possível ter acesso a dados específicos de inventário quando associados a um dispositivo. Exemplo: Identificador por radiofrequência (*Radio Frequency Identification* - RFID);
- **Controlo de qualidade** – Sensores recolhem dados de vários pontos da linha de produção e verificam quaisquer anomalias;
- **Otimização de linha de produção** – A análise de dados pode resultar numa melhoria num determinado processo de uma linha de produção, aumentando a eficiência e eficácia do processo;
- **Segurança** – A associação de IoT com *Big Data* pode melhorar de uma forma geral a segurança e saúde numa fábrica.

É sobre este último ponto, **Segurança,** que o trabalho, neste documento apresentado incide. O uso, em simbiose, das tecnologias pode, de forma preventiva, analisar comportamentos, ambientes e estados ao ponto de reduzir de forma acentuada a probabilidade de acidentes e assim aumentar o nível de segurança num ambiente fabril. Esta solução representa, não só, uma mais valia para o trabalhador, mas também para a empresa que a adota. Diminuindo o risco de acidente, diminui também a probabilidade de paragem imprevista de linha de produção e gastos monetários por pagamentos de ressarcimentos a trabalhadores ou danos materiais. Por outro lado, aumenta-se o nível de produção, a confiança dos trabalhadores, a imagem da empresa face aos concorrentes e, também, o lucro financeiro, objetivo final de qualquer empresa.

A este processo de uso de IoT em ambiente industrial dá se o nome de *Industrial* IoT (IIoT) [5]. O termo IIoT pode ser descrito como a associação entre máquinas, computadores e pessoas com operações industriais através da análise de dados com o intuito de transformar os requisitos de negócio [5].

Pode-se, então, concluir que o IIoT pode trazer benefícios para uma indústria como aumento da eficiência e do tempo de produção/serviço, entre muitos outros. Porém, alguns executivos consideram, ainda, que existem alguns entraves para a adoção desta tecnologia [5], nomeadamente:

- Integração de informação de diversas fontes e extração de conclusões valiosas para o negócio é complexo e demorado;
- Atualmente ainda é complicado encontrar profissionais com as capacidades necessárias para implementação dos serviços visto ser uma tecnologia relativamente recente;
- Entre outras.

Apesar da sua difícil implementação, devido ao custo associado e ao próprio receio à mudança, num futuro próximo, as indústrias começarão a adotar a tecnologia reconhecendo a sua capacidade e a possibilidade de lucro que dela advém. O projeto, neste documento descrito, tem o intuito de demonstrar as capacidades dos sistemas IoT na área da segurança de

trabalhadores num ambiente fabril, tendo como objetivo final a inserção da solução construída no mercado.

Este projeto enquadra-se na unidade curricular de Tese Mestrado (TMDEI) do Mestrado em Engenharia Informática, no ramo de Engenharia de *Software* do Instituto Superior de Engenharia do Porto e teve como base do seu desenvolvimento a empresa Abaco Consultores[2].

1.2 Problema

Os acidentes no trabalho são a principal razão do desenvolvimento do projeto. Entre as variantes deste problema, é realçado o foco nos acidentes ocorridos em ambiente fabril e que têm como consequência perdas de capacidade ou fatalidades.

Considera-se um acidente de trabalho não fatal qualquer incidente que decorra durante o horário de trabalho, que leve a danos físicos ou psicológicos e que envolvam uma paragem do trabalhador, no mínimo, de quatro dias. Como acidente fatal, considera-se qualquer acidente que leve à morte de um trabalhador no período de até um ano após o acidente [6].

Na Figura 1 e Figura 2 são apresentados alguns dados estatísticos sobre acidentes de trabalho fatais e não fatais durante o ano de 2015 em vários países europeus, assim como uma comparação com outros continentes.

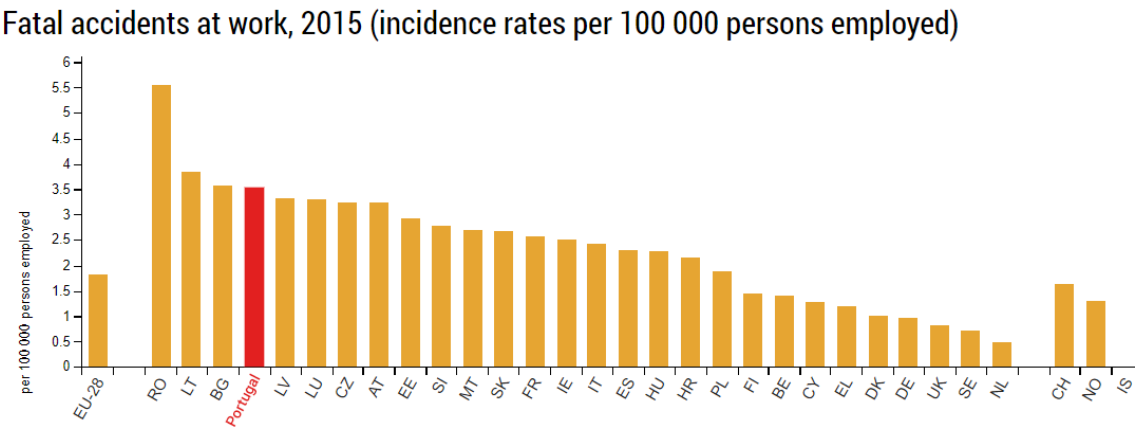


Figura 1 - Acidentes de trabalho mortais.

Adaptado de:

: https://ec.europa.eu/eurostat/statistics-explained/index.php/Accidents_at_work_statistics

[2] https://abaco.consulting/

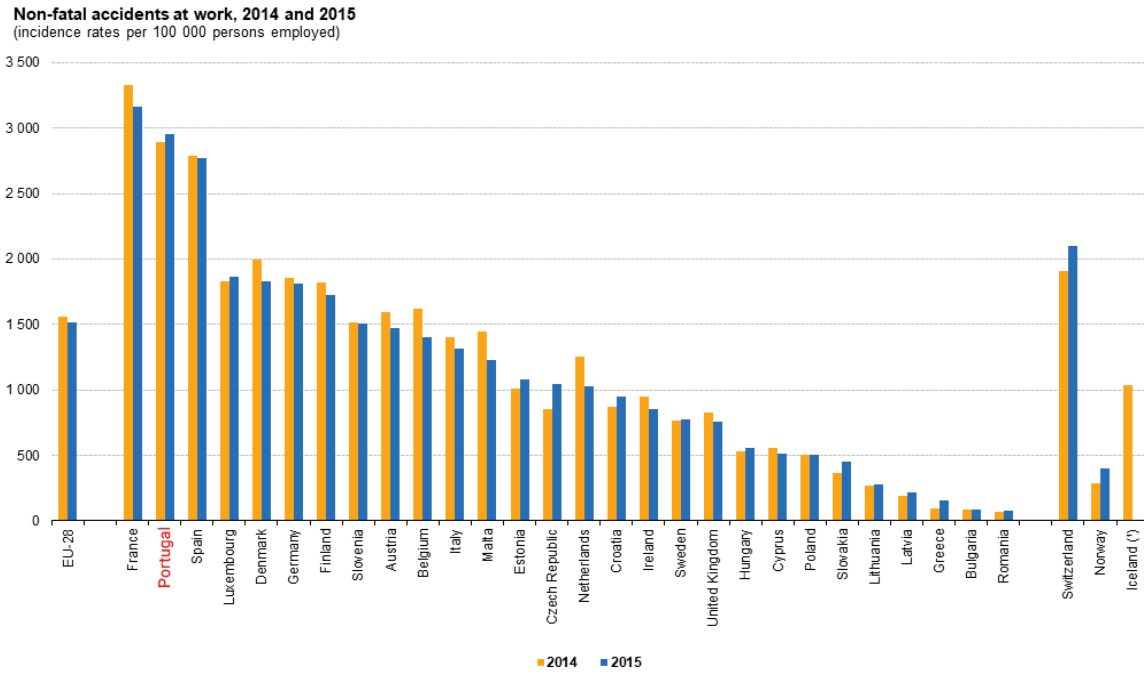


Figura 2 - Acidentes de trabalho não mortais
Adaptado de: https://ec.europa.eu/eurostat/statistics-explained/index.php/Accidents_at_work_statistics#Number_of_accidents

De acordo com os dados estatísticos apresentados pela Eurostat (Figura 1 e Figura 2) relativos ao ano de 2015, Portugal apresenta-se como o 4º país com mais acidentes de trabalho fatais, indicando uma média de 3.54 por cada 100.000 habitantes e o 2º com mais acidentes de trabalho não fatais, registando um aumento de número de casos entre 2014 e 2015.

O Gabinete de Estratégia e Planeamento Português emitiu recentemente um relatório [7] estatístico sobre acidentes de trabalho no ano 2016. Neste, é possível observar que dentro das 207.567 ocorrências existentes, 138 foram mortais. Estes incidentes acontecem maioritariamente, na indústria de construção civil, seguida pelas indústrias transformadoras e agricultura. Destes acidentes não mortais, 46,8% provocaram feridas e lesões superficiais, 36,4% provocaram lesões nas extremidades superiores (mãos, braços e cabeça) o que representa, em média, 37,4 dias de trabalho perdidos por acidente.

A estes acidentes estão associadas várias causas tanto fisiológicas, psicológicas e profissionais. Em Portugal a maioria dos acidentes deve-se a intoxicações químicas, quedas e soterramentos associados à falta de cumprimento das regras de segurança e do uso de material adequado à atividade [8]. Alguns destes incidentes são agravados também pela demora e falta de informação das equipas de emergência.

Tendo estes dados em causa é notório a falta de alguma tecnologia nestas áreas. Esta tecnologia, associada aos trabalhadores, poderá reduzir significativamente o número de incidentes, recaindo sobre uma precaução proativa. Através do registo de diversos fatores ambientais é possível prever algumas situações em ambiente fabril, assim como:

- Cansaço excessivo do trabalhador, medido através da sua pulsação média;
- Exposição prolongada a um determinado fator como calor, químicos, Ruídos…;
- Deteção de paragem prolongada que podem incorrer de uma possível queda;
- Entre outras.

Além da deteção proativa destas situações, uma tecnologia bem aplicada pode reduzir tempos de espera em caso de acidente. As equipas de emergência seriam ativadas de forma automática tendo, estas, acesso a dados relativos ao acidente em concreto. Com esta informação, será possível detetar a posição do acidentado e ter informação sobre qual o tipo de equipamento adequado para o salvamento, evitando sobrecarga ou transporte de material não indicado para a situação.

1.3 Objetivos

O objetivo deste trabalho será criar um produto que poderá ser integrado no serviço oferecido pelo *inCloud for SafeMed*. Este, é um serviço oferecido pela Abaco Consultores para a gestão de saúde e segurança no trabalho. Este serviço é apresentado de forma mais detalhada na Secção 2.1.1.

O produto a desenvolver, através do uso de diversos sensores conseguirá monitorizar áreas de produção onde aplicados, de forma a controlar fatores ambientais e evitar determinados acidentes.

De forma a monitorizar o ambiente será necessário recorrer a diversos sensores como:

- Temperatura;
- Humidade;
- Qualidade de ar;
- Luminosidade;
- Ruido;
- *Smartwatch*;
- Entre outros (Dependendo dos produtos produzidos e ambiente da fábrica).

Tendo acesso a todos estes dados ambientais e aos dados de saúde dos trabalhadores, o sistema deverá detetar situações anómalas e no caso de ocorrerem, ter acesso às respostas para as seguintes perguntas:

- O que aconteceu?
- Quando aconteceu?
- Quantas pessoas estão associadas?

Através destas respostas, poderá ser fornecido um serviço de emergência mais eficaz e eficiente aos trabalhadores em causa. Além disso, será, também, possível minimizar a propagação da situação (ex: Incêndio ou fugas de gás) e informar de forma quase instantânea todos os outros trabalhadores, criando rotas de fuga que evitem zonas com perigo.

O objetivo final, contudo, será evitar acidentes de trabalho e reduzir o risco de perda de vidas de forma preventiva.

De forma a cumprir este objetivo, será necessário encontrar respostas para as seguintes questões:

Q1. Que tecnologias devem ser estudadas que tragam benefícios à solução?

Q2. Como são obtidos dados de um determinado local? E quais os dados?

Q3. Quais os sensores disponíveis no mercado?

Q4. Como devem ser tratados os dados enviados pelos sensores?

Q5. Como devem ser notificados os gestores e outros trabalhadores de um acidente?

Q6. Qual a arquitetura a adotar?

Q7. Que custos poderão estar associados com a arquitetura escolhida?

Portanto, o que se pretende com o trabalho desenvolvido no contexto desta dissertação é responder à seguinte questão:

Q8. É possível utilizar dispositivos IoT para minimizar acidentes de trabalho?

Ao longo deste documento irão ser apresentadas as respostas a todas estas questões.

1.4 Abordagem inicial

Antes do início do projeto existia já, por parte dos responsáveis da Abaco, uma abordagem a uma possível solução a adotar. Esta, depois de apresentada, iniciou um processo de estudo superficial das tecnologias associadas de forma a validar essas mesmas opções. Após este estudo procedeu-se ao desenvolvimento de um diagrama de fluxo que representa o que seria, resumidamente, o funcionamento final do projeto. Este diagrama é apresentado na Figura 3. Esta abordagem foi apresentada com o intuito de demonstrar o fluxo de dados e informação que seria esperado numa solução final. A abordagem é apenas informativa e não define de forma definitiva nenhum ponto de implementação.

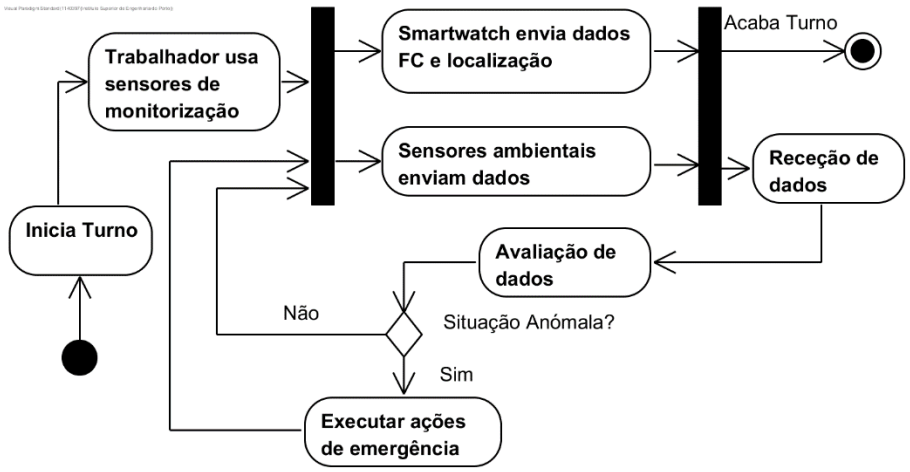


Figura 3 - Fluxo de Processos

Com esta disposição espera-se que, através da avaliação de dados emitidos pelos sensores ambientais, os trabalhadores sejam avisados aquando de um incidente na fábrica onde se encontram. Esta análise é feita estudando não só os fatores ambientais de uma determinada área, mas também relacionando esses fatores com os dados físicos de cada um dos trabalhadores presente nesse espaço.

Além disto, em qualquer momento ou lugar, um gestor poderá ter acesso aos dados relativos à fábrica e ao trabalhador. Em caso de acidente, o sistema emite uma notificação geral aos trabalhadores para evacuação, comunicando qual a rota de fuga certa a realizar de forma a evitar rotas bloqueadas ou congestionadas. Caso o sistema não detete uma situação, é também dada a opção ao trabalhador de emitir um estado de emergência, o que despoletará um processo semelhante ao anterior.

1.5 Análise de Valor

A Abaco Consultores, através do seu serviço *inCloud for SafeMed* apresenta uma solução para a área de segurança e saúde no trabalho para todas as empresas. Após uma análise de mercado, foi notória uma falha associada a este campo, a falta de sistema de monitoramento, em tempo real, de ambiente fabril. Posto isto, houve a necessidade de idealizar um sistema que, integrando com o serviço já oferecido, criasse uma ferramenta poderosíssima para monitoramento de ambientes e trabalhadores.

A Figura 4 representa o valor que o sistema representa. Do lado esquerdo, especifica-se o valor para o cliente e do lado direito, o valor da solução para a empresa. Na Secção 2.3 é feita uma apresentação mais detalhada à análise de valor.

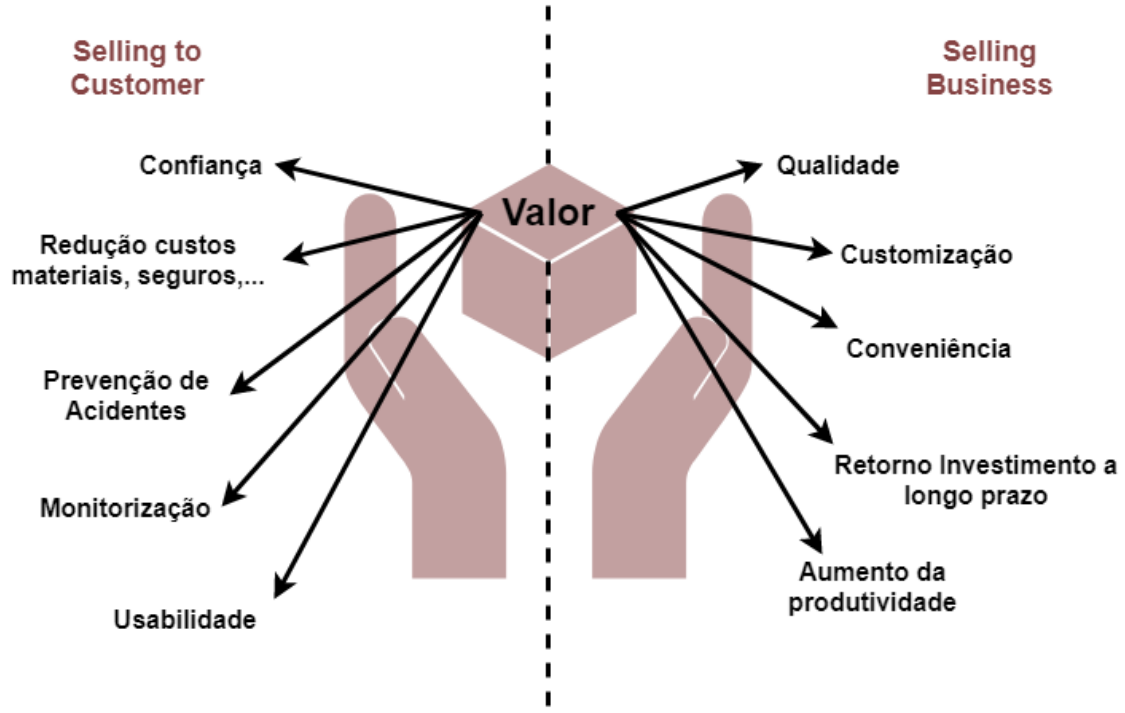


Figura 4 - Valor para o negócio

1.6 Metodologia de trabalho

Durante a execução do projeto pretende-se seguir uma metodologia. Representada a um alto nível, é de seguida exemplificada essa metodologia:

1. Identificar, contextualizar e analisar o problema;
2. Proposta de valor da solução;
3. Identificar e analisar uma ou várias soluções;
4. Identificar possíveis tecnologias associáveis à solução;
5. Escolha de solução mais acertada;
6. Definição dos requisitos do sistema;
7. Estado de arte de soluções existentes;
8. Proposta de arquitetura(s);
9. Escolha de arquitetura;
10. Desenvolvimento da solução;
11. Testes à arquitetura;
12. Testes à solução em ambiente real;
13. Conclusão e trabalho futuro.

1.7 Estrutura do documento

O presente documento é composto por oito Capítulos: Introdução, Contexto e Estado da arte, Tecnologia Relevante, Desenvolvimento da solução, Experimentação e Avaliação, Conclusões e trabalho futuro, Referências e Anexos.

A primeira parte, **Introdução**, está dividida em sete Subcapítulos. Nestes, pretende-se enquadrar o leitor para o problema e o contexto em que a solução incide. São definidos os objetivos assim como resultados esperados e é feita uma abordagem detalhada dos passos a dar para chegar a estes. É também referida a organização da estrutura do documento, com o intuito de guiar o leitor da melhor maneira.

O Capítulo 2, **Contexto e Estado da arte,** está dividido em três partes distintas. A primeira, Contexto, apresenta alguns detalhes relevantes para a solução final do sistema. Composta por quatro Subcapítulos, começa-se por apresentar o sistema *inCloud for SafeMed* da Abaco Consultores. De seguida são apresentados alguns conceitos de negócio, apresentados os intervenientes de sistema e referidas algumas restrições possíveis. Na segunda parte, Estado da arte, é apresentado o estudo e trabalho de pesquisa realizado acerca de outras abordagens ao problema e é feita a comparação ao sistema, neste documento proposto. Feita a comparação é então analisado o valor que a solução projetada assume através da Análise de valor. Esta, tem como objetivo, demonstrar o valor que o sistema final poderá criar a um possível cliente. Através do uso de modelos como FFE, NCD e CANVAS é possível verificar os pontos fortes e fracos que são perspetivados para o sistema que se pretende desenvolver.

No terceiro Capítulo, **Tecnologia Relevante** é demonstrado o estudo realizado sobre as tecnologias associadas ao projeto em si. Sendo assim, este capítulo inicia-se com a demonstração do estudo realizado sobre o IoT e *big data*. De seguida é abordado o tema de protocolos de comunicação entre dispositivos assim como plataformas computacionais de baixo consumo. O Capítulo termina com o estudo sobre as plataformas de gestão de dados de IoT tanto *open source* como alojadas na *cloud*.

O Capítulo 4, **Desenvolvimento da Solução**, define toda a arquitetura do sistema a implementar. Inicia-se com um Subcapítulo onde são definidas quais as melhores opções para escolha das tecnologias anteriormente apresentadas. Neste ponto define-se quais os caminhos a tomar e quais as possibilidades a ter em conta aquando do desenvolvimento da solução. De seguida apresentam-se as soluções em si. Através da apresentação de diferentes diagramas é exposta a arquitetura do sistema a implementar. Escolhida a solução arquitetural, são feitas e apresentadas algumas decisões sobre os componentes a desenvolver. No final do Capítulo é referenciado o trabalho realizado para a criação do sistema.

No quinto Capítulo, **Experimentação e Avaliação**, é demonstrado o plano de testes e avaliação realizado para avaliação do sistema. Neste, são definidas métricas que avaliam o sucesso ou não da solução final assim como apresentados os testes realizados à solução desenvolvida.

O Capítulo 6, **Conclusões e trabalho futuro,** apresenta uma abordagem geral critica ao trabalho desenvolvido. São identificadas as dificuldades e limitações encontradas, expostos os objetivos alcançados e, por fim, é referenciado algum trabalho que pode vir a ser feito para melhorar a solução final.

Por fim, os dois últimos Capítulos, **Referências** e **Anexos**, apresentam, respetivamente, quais as fontes de informação para execução deste documento e algumas figuras e outros recursos relevantes na execução da solução.

2 Contexto e Estado da arte

Este Capítulo encontra-se dividido em três partes distintas: Contexto, Estado de arte e por fim Análise de Valor. Tem como objetivo primário responder de forma parcial à questão **Q1** apresentada anteriormente e em certa parte reforçar o valor que a solução preconizada poderá trazer aos clientes.

2.1 Contexto

Esta secção tem como objetivo apresentar alguns detalhes relativos ao contexto do problema a ser tratado. Inicialmente é apresentada o sistema da Abaco Consultores, denominado de *InCloud for Safemed*. De seguida, são descritos os processos e intervenientes do sistema assim como algumas restrições que possam existir

2.1.1 InCloud for SafeMed

O *InCloud for SafeMed* é uma solução *cloud* para a área de Segurança e Saúde no Trabalho (SST) que permite a gestão de todos os requisitos legislados pela Autoridade para as Condições do Trabalho (ACT) [9]. Dentro da gestão, incluem-se:

- **Gestão operacional** – Gestão das operações da empresa (ex: gestão de turnos, gestão de recursos, etc.);
- **Gestão administrativa** – Coordenação de recursos (ex: Convocatórias, marcação de formações, etc.);
- **Gestão clínica** – Gestão de processos de SST (ex: Marcação de consultas, ficha clínica, etc.).

Este fator permite que esta solução seja adaptável tanto às empresas prestadoras deste serviço como às que efetuam gestão interna dos processos de SST.

O *inCloud for SafeMed*, através das suas funcionalidades, permite a qualquer cliente reduzir custos operacionais e, estando alojada na *cloud*, impossibilita a perda de dados críticos por falhas de sistema. A solução está associada a dados clínicos sensíveis e, consequentemente, são cumpridas as regras ditadas pelo Regulamento Geral sobre a Proteção de Dados (RGPD).

É sobre este último ponto que incide o projeto. Tendo acesso a dados clínicos de trabalhadores seria possível fazer uma gestão em tempo real, associando estes com os dados recolhidos pelos sensores e *wearables* do trabalhador. Esta associação poderia prevenir qualquer tipo de situação de risco para uma determinada pessoa. Um exemplo deste processo seria a identificação de uma funcionária grávida que entra numa sala onde a qualidade de ar não é suficiente para o seu estado físico e, se exposta durante determinado tempo, pode afetar o seu estado de saúde. Neste caso, o sistema iria intervir, e a funcionária seria aconselhada a sair do ambiente onde está.

2.1.2 Detalhes do Contexto

Desde a criação da indústria fabril que se tem estudado e evoluído os processos usados nas empresas. Estas mudanças começaram aquando da primeira revolução industrial que emergiu do uso do vapor. Esta constante evolução acabou por resultar em mais duas revoluções, datadas de 1870 e 1969 [10]. A quarta revolução industrial (denominada também de indústria 4.0 [11]) trouxe ao mundo empresarial ferramentas poderosas capazes de aumentar a produtividade e fiabilidade de qualquer negócio [12]. Entre estas evidenciam-se os processos automatizados com recurso a Inteligência Artificial (IA), mecanismos de *Big Data*, robótica, entre outros. Porém, o aumento destes patamares exige um maior controlo no que toca à saúde e bem-estar dos trabalhadores. Para isso, e ainda não havendo uma solução funcional para controlo, é necessário elaborar um sistema de monitorização capaz de prever situações de risco e evitar perdas humanas e materiais. Além do controlo de saúde, este sistema, permitirá identificar trabalhadores que apresentem menor eficiência no trabalho e assim corrigir possíveis falhas.

Esta solução surgiu com o intuito de ser incluído no serviço oferecido pela empresa Abaco Consultores, *inCloud for SafeMed*.

2.1.3 Conceitos de negócio

A Figura 5 representa, de forma não técnica e de alto nível, o sistema pensado pelos responsáveis do projeto para responder aos possíveis problemas. É representado o fluxo de informação, desde os sensores até ao consumidor final, neste caso, o *Incloud for SafeMed*.

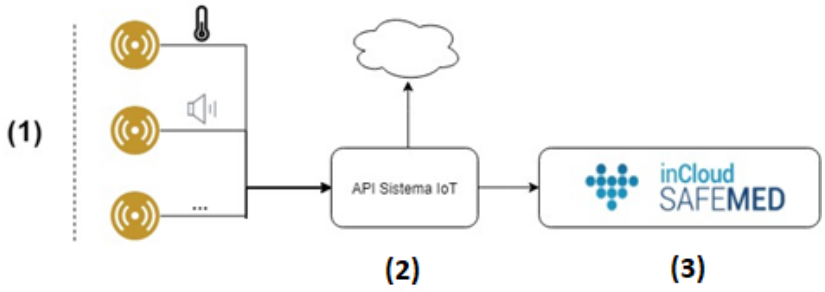


Figura 5 - Modelo de negócio de alto nível

Através da análise da Figura 5 é possível identificar os componentes constituintes do sistema:

1- **Sensores -** Responsáveis por estudar fatores ambientais de determinados locais. Consoante o tipo de negócio da empresa onde é inserido o sistema, podem ser necessários diferentes tipos de sensores;
2- **API Sistema IoT -** Sistema responsável por tratamento de informação advinda dos sensores. Quando detetadas situações de perigo deve, prontamente, iniciar um processo de contra medidas, conforme a situação;
3- ***InCloud for SafeMed* -** Entidade final que terá acesso a todos os dados armazenados e tratados na entidade anterior.

2.1.4 Restrições Externas

Tratando-se de dados sensíveis a uma empresa e aos seus trabalhadores, deve ser feito um trabalho reforçado para encriptação de dados e para cumprimento da legislação do RGPD.

A segurança é também um fator crucial. Visto que o projeto inclui dispositivos IoT, é necessário ter maiores cuidados de forma a evitar quaisquer tipos de ataques aos mesmos. É importante referir que alguns dos maiores ataques informáticos foram feitos tendo como acesso dispositivos IoT.

2.2 Estado da arte

Nesta secção são apresentadas outras soluções já existentes no mercado com o mesmo objetivo da, neste documento, apresentada. A análise aos concorrentes terminará com uma comparação entre essas e a solução preconizada, sendo também efetuada a análise de valor desta.

2.2.1 Soluções e abordagens existentes

Nesta secção são apresentadas algumas soluções ao problema já existentes no mercado. O objetivo será analisar as tecnologias e plataformas usadas por estas. No final da secção é também feita uma comparação com o sistema previsto para a Abaco Consultores.

2.2.1.1 Iot Connect – Smart Connected Worker

A empresa *IoT Connect* [13] oferece uma solução aproximado à projetada pela Abaco consultores. Através do serviço *Smart Connected Worker*, esta solução oferece serviços como:

- Análise de métodos de segurança;
- Criação de áreas na planta da fábrica;
- Monitorização local de trabalhadores;
- Monitorização vital de trabalhadores;
- Orientação em locais fechados;
- Análise fatores ambientais.

Como áreas de intervenção, são incluídas indústrias de construção, transporte e logística, petrolíferas, mineração, agricultura, entre outras.

Esta solução é oferecida como uma *Platform as a Service* (PaaS). Usando serviços oferecidos pela Microsoft, nomeadamente *Azure IoT*, esta PaaS permite a gestão total de comunicações entre dispositivos e clientes, incluindo o armazenamento das mesmas. A Figura 6 apresenta as quatro áreas que compõe esta solução:

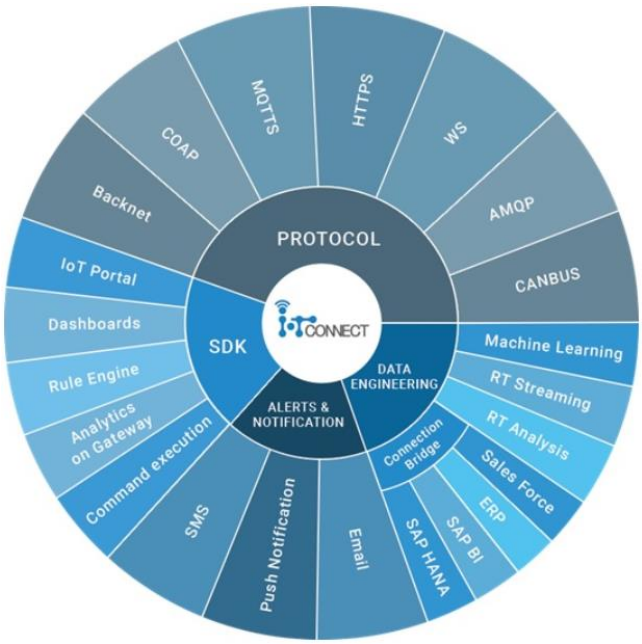


Figura 6 - Componentes IoTConnect.io

Fonte: https://help.iotconnect.io/2018/08/21/hello-world/?section=high-level-architecture-of-iotconnect-4

- ***Protocol*** – Representa os protocolos de comunicação existentes. Inclui protocolos de comunicao entre dispositivos como MQTTs e CoaP;
- ***Data Engineering*** – Apresenta os mecanismos de processamento de dados assim como ligações a serviços especialistas deste serviço como *SAP Hana*;
- ***Alerts & Notification*** – Apresenta as possíveis ações de aviso a clientes aquando da deteção de anomalias;

- ***SDK*** – Representa a forma como os clientes podem interagir com o sistema nomeadamente: consulta de dados através de *Dashboard*, definição de regras, entre outros.

Segundo a empresa, a solução é composta por cinco módulos [14]:

- ***IoT Device Managment*** – Depois da instalação dos dispositivos nos locais, é oferecido um sistema de gestão dos mesmos. Estes dispositivos vão desde microcontroladores a sensores comerciais mais complexos independentemente do tipo de protocolos de comunicação;
- ***Multitenancy*** – É oferecida uma ferramenta para gestão de grupos de utilizadores que partilham os mesmos serviços;
- ***Smart Rules*** – Criação de regras com notificações personalizáveis;
- ***Firmware & OTA*** – São aceites *updates* aos sensores sem necessitar de ir ao local dos mesmos. Este fator pode depender do tipo de dispositivos;
- ***Events*** – Sistema de notificações através da ocorrência de eventos.

2.2.1.2 Avnet – Smart Connected Worker

À semelhança do ponto anterior, a empresa Avnet [15], oferece um serviço com a mesma denominação e com um objetivo semelhante. Através do uso de sensores, coletes de alta visibilidade e *wearables*, esta solução possui como funcionalidades:

- Dados de suporte para decisões;
- Monitorização de diversas áreas;
- Partilha de informação entre trabalhadores, sistemas e sensores.

O serviço é oferecido para ambientes fabris, refinarias, operações mineiras e fábricas de produção de energia. Como plataforma de suporte para este serviço, a Avnet, à semelhança do seu concorrente anteriormente apresentado, utiliza *Azure* da Microsoft. Segundo a empresa, os tópicos de segurança, disponibilidade e eficiência são também pontos que foram tidos em consideração durante a criação do sistema.

2.2.1.3 Comparação entre soluções

A Tabela 1 apresenta uma comparação entre as duas soluções anteriores com a solução que a Abaco Consultores pretende criar (Solução Projetada).

Tabela 1 - Comparação soluções existentes

	Abaco Consultores Solução projetada	**IoTConnect**	**Avnet**
Segurança de trabalhador	✓	✓	✓
Análise local	✓	✓	✓
Áreas geográficas	✓	✓	✓

	Abaco Consultores Solução projetada	**IoTConnect**	**Avnet**
Monitorização Local trabalhadores	✓	✓	✗
Monitorização ambiental	✓	✓	✓
Monitorização vital	✓	✓	✓
Navegação interna	✓	✓	✗
Análise preditiva em tempo real	✓	✗	✗
Relação com dados de saúde	✓*	✗	✗

* - Restrito a empresas que sejam clientes *inCloud for SafeMed*

Analisando a Tabela 1, são visíveis as vantagens do serviço idealizado pela Abaco Consultores. Caso a empresa cliente não seja também cliente do *inCloud for SafeMed*, o serviço oferecido é semelhante ao *IoT Connect*. Não havendo registos clínicos dos trabalhadores, apenas pode ser feita a análise através dos dados de sensores em relação com os dados vitais enviados pelos *wearables*. Feita a análise das soluções já existentes no mercado é necessário, agora, averiguar o valor que a solução a desenvolver apresenta. Esta análise é feita no Subcapítulo que se segue.

2.3 Análise de Valor

O desenvolvimento e lançamento de novos produtos acarreta um risco bastante elevado no que toca a orçamento e aceitação por parte dos possíveis clientes. De forma a reduzir este risco é necessário investir numa melhoria continua das técnicas utilizadas no Processo de Desenvolvimento de Produtos. Este processo está separado em três fases distintas: Pré-desenvolvimento, desenvolvimento e pós desenvolvimento.

Neste Subcapítulo além de apresentados modelos que incidem sobre este processo, é também detalhado o valor gerado pela solução esperada. Por fim é apresentado o modelo de negócio, com recurso ao modelo CANVAS.

2.3.1 Modelo Fuzzy Front End (FFE)

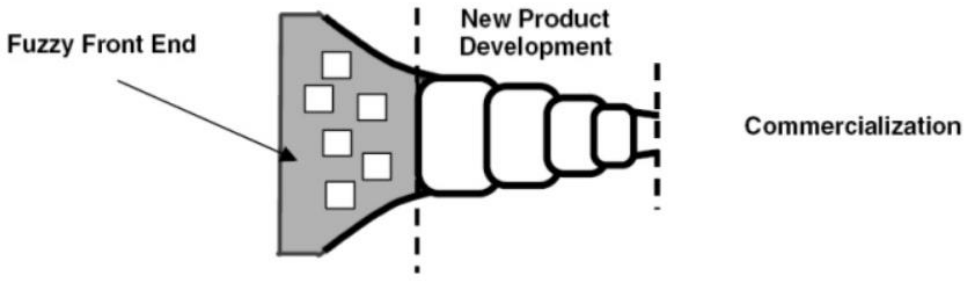


Figura 7 - Processo de inovação

Fonte: http://www.stevens-tech.edu/cce/NEW/PDFs/FuzzyFrontEnd_Old.pdf

Este processo, representado na Figura 7, está dividido em três fases distintas:

- ***FFE* -** Fase experimental da inovação. Pode ser caótica uma vez que não apresenta grande organização. Baseada nos momentos "Eureka". Tem como objetivo identificar possíveis oportunidades de negócio;
- ***New Product development (NPD)* -** Método disciplinado e orientado a objetivos. Tem como objetivo aperfeiçoar o trabalho feito no ponto anterior;
- **Comercialização -** Promoção e venda do produto criado.

De forma a poder fazer a gestão eficaz do FFE [16], foi criado o modelo *New Concept Development* (NCD) [17]. Este modelo proporciona a linguagem e terminologia necessária para otimizar o processo de inovação FFE já apresentado.

2.3.2 Modelo New Concept Development

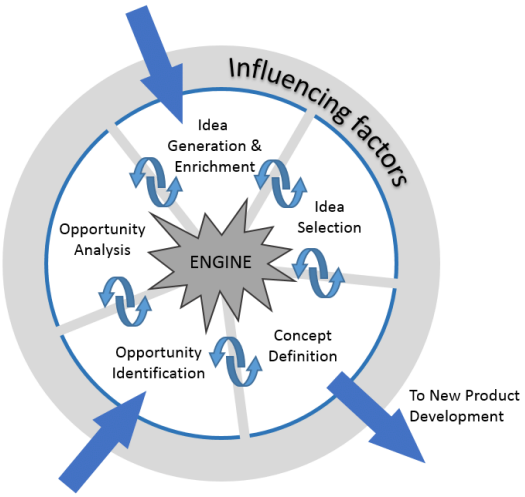


Figura 8 - Modelo NCD

Fonte: https://www.researchgate.net/figure/The-New-Concept-Development-NCD-model-Koen-et-al-2001_fig12_324208591

O modelo NCD, Figura 8, está dividido em três partes:

- ***Engine* (Motor) -** Onde se explica a liderança, cultura e estratégias de negócio que gerem o processo inovador. É o elo central do sistema e concentra em si os fatores principais de decisão;
- **Cinco áreas de atividade:**
  - ***Opportunity Identification*** – Área de identificação de novas oportunidades. A identificação inclui a necessidade de recursos, tecnologias e modelo de negócio. Como oportunidade define-se qualquer necessidade de negócio ou necessidade técnica de uma empresa que responda a um problema ou ameaça e que traga vantagens competitivas [18];

  - ***Opportunity Analysis***– Fase de análise das oportunidades identificadas no ponto anterior. A seleção da oportunidade a ser desenvolvida faz-se tendo em conta os recursos da empresa;
  - ***Idea Generation & Enrichment*** – Transformação das oportunidades em ideias de produtos. É um processo evolutivo em que existe combinação de pensamentos até se chegar a uma solução que atenda às necessidades dos clientes e à capacidade da empresa. A ideia é definida como a forma mais embrionária de um novo produto, serviço ou solução [18];
  - ***Idea Selection*** – Depois de elaborar a ideias é necessário selecionar uma ou mais ideias para desenvolvimento. Esta seleção tem como critérios: o risco de implementação tecnológica, riscos de investimento, análises de competição, capacidade organizacional e o retorno possível;
  - ***Concept Definition*** – Um conceito apresenta uma forma (Escrita ou visual) com características e benefícios únicos [18]. Fase de desenvolvimento de um plano de negócios com base em estimativas (potencial de mercado, necessidades dos clientes, entre outras). Este plano de negócios possui um nível de formalidade consoante a natureza da oportunidade, nível de recursos, etc.;
- **Fatores ambientais -** Representados na figura por ***Influencing Factors.*** São os fatores que influenciam o motor e as cinco áreas de atividade. Nestes incluem-se:
  - Capacidades de organização;
  - Ameaças de competidores;
  - Modas de compra de clientes;
  - Mudança de Regulamentos;
  - Conhecimento e capacidade das tecnologias.

Ao contrário dos processos lineares [19], como descritos anteriormente, este processo é circular o que implica a ideia de fluência, circulação e iteração entre as diferentes áreas de atividade. As setas no modelo, Figura 8, representam os pontos de partida possíveis (Identificação de oportunidade ou criação de ideias e enriquecimento) para projetos. Estes projetos têm como ponto de saída o início do desenvolvimento de um novo produto.

Relativamente ao projeto neste documento apresentado:

- **Identificação da oportunidade -** Apesar da fiscalização cada vez maior de fábricas, acidentes de trabalho continuam a estar presentes no dia-a-dia das empresas. Estes, além de prejudicar as empresas em horas de trabalho e monetariamente, são também uma ameaça à reputação de uma instituição.
  Observando o valor apresentado pela solução *inCloud for SafeMed*, os gestores da Abaco Consultores aperceberam-se da oportunidade de o aumentar, apresentando um projeto de IoT direcionado as empresas compostas por ambiente fabril. O sistema, integrado com o já existente, permite às empresas manter um nível de produtividade e confiança bastante elevado. A identificação desta oportunidade adveio de uma falha de mercado neste campo. No momento de escrita deste documento, o número de serviços que oferecem as mesmas capacidades é bastante limitado e difícil de encontrar;
- **Análise da oportunidade -** Considerando os dados já apresentados no Capítulo 1.2 sobre acidentes de trabalho, é percetível a necessidade de um sistema que possa

diminuir ou até impedir algum tipo de incidente. Analisando o mercado, como já referido, denota-se uma falta de oferta notória deste tipo de sistema sendo que os já existem não são fáceis de encontrar e perceber;

- **Elaboração de ideias e enriquecimento -** Identificada e avaliada a oportunidade foi iniciado o processo de elaboração de ideias e do seu enriquecimento. Numa primeira fase foi feito um estudo sobre que tipo de tecnologia se poderia utilizar para o efeito. Nesta, foi necessário identificar as capacidades que o sistema deveria possuir para que os clientes reconheçam valor. Posto isto foram apresentadas as seguintes características:
    1. Sistema deve saber onde os trabalhadores estão presentes a qualquer momento, assim como o histórico de movimentações;
    2. Deve ser feita a gestão de diversas áreas da fábrica;
    3. Sistemas de emergência;
    4. Definição de limites;
    5. Capacidade de observação dos dados.

  Para a primeira, e tendo a noção que os trabalhadores não devem ter a sua mobilidade condicionada por aparelhos grandes ou pesados, foram abordados os *smartwatches.* Estes possuem capacidade de comunicarem com o sistema e transmitirem informações sobre o seu utilizador. Para monitoramento do movimento, os *smartwatches* seriam usados em conformidade com sensores de *Bluetooth* de baixa frequência. Neste ponto, foi também considerado usar câmaras de vídeo que, através de IA, identificariam os trabalhadores. Para a segunda característica, seriam usados diversos sensores, dependendo do ambiente de fábrica. Estes, possuem a capacidade de se ligar à rede e, assim, transmitir dados. Para os pontos três, quatro e cinco, seria adotado um serviço disponível ou desenvolvido um sistema próprio que teria estas capacidades;
- **Seleção de ideias -** Analisando as ideias anteriormente descritas, foi feita a seleção que seria mais aconselhável. Posto isto, e tendo em consideração que o uso de IA poderia acarretar diversas falhas, foi selecionado o uso de *smartwatches* em conformidade com os sensores de localização. Os dados dos sensores são enviados para o sistema que os analisa e guarda numa base de dados;
- **Definição de conceito -** Com este sistema pretende-se aumentar o nível de segurança e confiança no ambiente de trabalho. O seu uso poderá vir a evitar possíveis casos de emergência e custos adicionais de pagamentos de coimas, indemnizações, material, entre outros. Além desta possibilidade, o administrador do sistema tem também acesso a dados de ambiente fabril que podem ser importantes para o negócio.

2.3.3 Valor, Valor para o cliente e Valor percecionado

A procura de vantagem das organizações face aos concorrentes é, cada vez mais, um objetivo crucial. A criação de valor é uma das formas de distinção de terceiros. Este é um componente chave para a sustentabilidade e crescimento de uma organização, porém, é também um conceito de difícil alcance.

2.3.3.1 Valor

O valor pode ser caracterizado como a razão entre os benefícios e os custos/sacrifícios [20]. Dentro dos benefícios, incluem-se benefícios práticos e intangíveis. Já nos custos, associam-se custos monetários, de tempo, de energia e intangíveis. Valor resume-se, então:

$$Valor = \frac{Beneficios}{Custos} = \frac{práticos + intangiveis}{monetários + de\ tempo + de\ energia + intangiveis}$$

Tendo isto em conta, é de seguida apresentada a tabela de benefícios e sacrifícios do projeto aqui apresentado:

Tabela 2 - Benefícios/Sacrifícios Valor

	Serviço	**Relacionamento**
Benefícios	Monitoramento de movimentos de trabalhadores; Previsão de incidentes; Redução do número de acidentes de trabalho; Controlo ambiental da fábrica; Usabilidade; Fiabilidade;	Confiança; Segurança;
Sacrifícios	Investimento inicial no sistema e *hardware;* Manutenção de sensores;	

2.3.3.2 Valor para o Cliente

Segundo *Woodall* [21], o valor para o cliente (VC) é qualquer tipo de perceção pessoal de vantagem comercial em comparação com a concorrência, que surge da oferta do sistema apresentado. Esta vantagem tanto pode ser apenas uma redução de esforço de produção, como, existência de benefícios, o resultado de uma operação Custo/Beneficio com resultado positivo, entre outros.

A solução aqui apresentada pode oferecer ao utilizador uma ou várias destas sensações de vantagem. A longo prazo pode reduzir custos de material e pessoal, evitando situações de risco. Assumindo que o custo inicial é relativamente alto, o cliente deve projetar a diferença Custo/Beneficio a longo prazo. Além da vantagem económica que pode vir a ser, o cliente terá também a perceção de segurança, tanto material como humana. Esta sensação estende-se também aos seus trabalhadores, o que, consequentemente, irá aumentar a produtividade e satisfação destes.

2.3.3.3 Valor Percecionado

O conceito de valor percecionado, segundo *Lindgreen* e *Wynstra* [22], define-se como o valor que cada cliente assume que um determinado produto tem para si e para o seu negócio. Estes

autores, defendem também que o valor percecionado pelo cliente é diferente do valor percecionado pelo produtor. Enquanto que este é mais sensível à qualidade do produto, o cliente assume uma posição mais cuidada no que toca ao preço do serviço oferecido.

Os possíveis clientes deste produto poderão ter diversas opiniões do mesmo. Dependendo do tipo de ambiente fabril que possuam, poderá ou não haver uma maior necessidade de implementação da solução. Enquanto que em ambientes de Atmosferas Explosivas (ATEX) é fulcral ter um sistema similar, num sistema comercial menos perigoso, o sistema poderá ser algo apenas de controlo, não fundamental. Porém, em ambos os casos, o sistema poderá vir a ser útil na redução de perda material e humana, resultando numa redução de custos.

2.3.4 Perspetiva longitudinal de valor

A perspetiva longitudinal de valor, criada por [21], representada na Figura 9, é composta por quatro campos temporais: Pré-compra, Implementação, Pós-compra (durante a utilização) e Pós-experimentação.

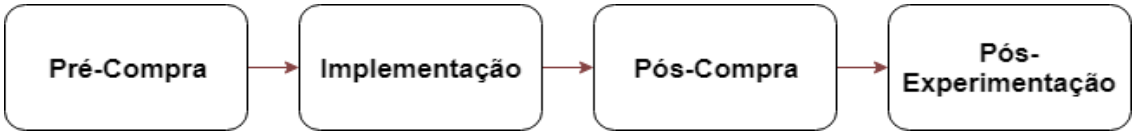


Figura 9 - Perspetiva Longitudinal

Estas fases, estando associados ao valor para o cliente, apresentam tanto benefícios como sacrifícios:

- **Pré-Compra –** O Cliente reconhece o valor que o sistema pode acrescentar na sua fábrica. Tem em consideração o investimento inicial a que será sujeito aquando da aquisição do mesmo. Além da análise financeira, o cliente, deve também analisar a necessidade do sistema e quais os sensores mais adequados para o seu ambiente;
- **Implementação –** Esta é a fase critica para o cliente. Conforme o número de divisões e trabalhadores que serão monitorizados, poderá haver um custo superior ao desejado. Nesta fase, é também necessário introduzir o sistema aos trabalhadores. Esta formação, inclui operários e possíveis gestores;
- **Pós-Compra –** Esta fase inclui a utilização do sistema. É nesta que o cliente verifica o valor do sistema adquirido. No caso da solução aqui apresentada, é esperado que apenas seja apenas usado para monitorização de ambiente e trabalhadores, estando, obviamente, pronto para reagir a qualquer situação de emergência que possa ocorrer;
- **Pós-Experimentação –** Nesta etapa, o cliente apresenta confiança no sistema. A capacidade de prevenção de acidentes é também já reconhecida por este. Além do constante monitoramento automático do sistema, o cliente pode também ter acesso a dados relativos aos trabalhadores e assim identificar falhas no ambiente fabril.

2.3.5 Proposta de valor

Considerando a seguinte definição de proposta de valor:

"A product's value proposition is a statement of the functional, emotional and self-expressive benefits delivered by the brand that provide value to the target customer." [23]

É possível verificar que a proposta de valor pode ser definida com resposta a algumas questões:

1. Que produto/serviço estamos a oferecer aos clientes?
2. Que tipo de benefício é que os clientes vão ter ao usar o produto/serviço?
3. Quais os clientes alvo para o nosso produto/serviço?
4. Qual a competição existente? De que forma o nosso produto/serviço é diferente?

Como possível resposta a estas perguntas pode-se ter:

1. Sistema de gestão de ambiente fabril onde se inclui monitorização dos trabalhadores através de *smartwatch*. Associado ao *inCloud for SafeMed* é feito a análise dos dados clínicos de um operador e relacionado com os dados ambientais do momento, verificando se existe algum tipo de incompatibilidade. O sistema é também capaz de reagir a qualquer situação de emergência, incluindo acidentes de trabalho, fugas de gás, incêndios, entre muitos outras possibilidades;
2. Os clientes podem monitorizar e controlar todo o ambiente da sua fábrica. Além do aumento do grau de confiança por parte dos gestores, também os operados sentirão este aumento o que, consequentemente, se refletirá na produtividade da fábrica. O monitoramento dos operários pode também permitir a identificação de falhas nas linhas de produção e, prontamente, corrigir as mesmas. No caso de acidente, o sistema diminui tempos de resposta e pode também reduzir custos a nível material e humano;
3. O sistema é adaptável a qualquer tipo de empresa que possua uma fábrica como o centro de produção. A customização é feita através de diversos fatores. Conforme a área de negócio de cada cliente, serão escolhidos diferentes sensores para registar diferentes tipos de valores. Como principal cliente alvo têm-se clientes que possuam ambiente ATEX;
4. Apesar de já existir concorrentes que efetuem um trabalho semelhante, este sistema é capaz de efetuar funções que nenhum outro consegue. A sua capacidade de previsão de incidentes traz um valor acrescentado. Além desta vantagem, quando usado em simbiose com o *inCloud for SafeMed* apresenta-se como um sistema proativo na manutenção da saúde de todos os trabalhadores.

2.3.6 Modelo CANVAS

O modelo CANVAS é uma ferramenta de gestão estratégica que permite desenvolver modelos de negócio, quer estes sejam novos ou existentes [24]. É constituído por nove blocos individuais que podem ser divididos em duas áreas fundamentais: Lado subjetivo e emocional e Lado lógico e estrutural. Apesar da distinção, todos os blocos comunicam entre si, complementando-se.

Estes blocos são:

- **Segmentos de Clientes** (*Customer Segments*) – Especifica quem vai ser o cliente alvo do negócio a criar. Define-se quais os clientes mais importantes;
- **Proposta de Valor** (*Value Proposition*) – Explica qual a proposta de valor que o negócio apresenta aos possíveis clientes. Define que tipos de problemas vão ser respondidos e quais as necessidades dos clientes que serão satisfeitas;
- **Canais de distribuição** (*Channels*) – Define quais os canais que serão usados para chegar aos clientes;
- **Relações com clientes** (*Customer Relationships*) – Neste bloco é esclarecido qual o tipo de relação que o cliente espera ter da empresa fornecedora do negócio;
- **Fontes de receita** (*Revenue Stream*) – Especifica de que forma e pelo que que os clientes irão pagar;
- **Recursos Chave** (*Key Resources*) – Define quais os recursos chave necessários para a proposta de valor apresentada;
- **Atividades Chave** (*Key Activities*) – Distingue quais as atividades chave necessárias para manter os clientes interessados no negócio e para fornecer os serviços definidos na proposta de valor;
- **Parceiros Chave** (*Key Partners*) – Enumera quais os parceiros mais importantes para fornecer o serviço/produto;
- **Estrutura de custos** (*Cost Structure*) – Especifica quais os custos mais importantes para o modelo de negócio.

A Tabela 3 apresenta a adaptação do modelo CANVAS ao projeto aqui apresentado. Relativamente aos parceiros chave, é importante denotar que durante a execução deste projeto ainda se estavam a contactar possíveis parceiros, não sendo ainda possível especificar os mesmos. Este fator é também importante para a resposta à pergunta **Q3**, relativa ao uso e produção de sensores. Nesta fase do projeto ainda não se obteve dados suficientes para ser feita a decisão do uso de sensores externos.

Tabela 3 - Modelo CANVAS

<table>
<tr>
<td rowspan="2">Key Partners

✓ Fornecedores de sensores (Ainda por definir)</td>
<td>Key Activities

✓ Manutenção de software;
✓ Melhoramento das capacidades de previsão;</td>
<td rowspan="2">Value Proposition

✓ Monitorização de ambientes e trabalhadores;
✓ Previsão de incidentes;
✓ Dados ambientais em tempo real acessíveis em qualquer plataforma;
✓ Resposta a emergências automática;
✓ Redução de custos materiais;
✓ Redução de danos físicos em trabalhadores;
✓ Customizável para as necessidades dos clientes.</td>
<td>Customer Relationships

✓ Constante melhoria de sensores e algoritmos de previsão;
✓ Substituição de hardware defeituoso;
✓ Suporte pós-venda;</td>
<td rowspan="2">Customer Segments

✓ Fábricas com e sem ambientes ATEX;
✓ Centros logísticos;
✓ Indústrias que lidem com material perigosos não explosivos;
✓ Seguradoras;
✓ Indústrias que possam ser adaptáveis a este serviço;</td>
</tr>
<tr>
<td>Key Resources

✓ Sensores;
✓ Smartwatches;
✓ Servidores Cloud;
✓ Internet;
✓ BackOffice;</td>
<td>Channels

✓ Clientes InCloud for SafeMed;
✓ Plataforma online com exemplos de aplicações;
✓ Demonstrações em clientes;
✓ Feiras de empreendedorismo e segurança no trabalho;</td>
</tr>
<tr>
<td colspan="3">Cost Structure

✓ Equipa de desenvolvimento e manutenção do software;
✓ Alocação Cloud;
✓ Deslocamentos aos clientes;
✓ Marketing;
✓ Compra de Sensores;</td>
<td colspan="3">Revenue Streams

✓ Venda do sistema por trabalhador monitorizado;
✓ Licenciamento da solução;
✓ Serviços de desenvolvimento à medida;</td>
</tr>
</table>

2.4 Sumário

Este Capítulo descreve a contextualização necessária para uma melhor compreensão da informação, nos seguintes capítulos apresentada. O sistema aqui apresentado, permite ao leitor perceber a abordagem que vai ser feita ao problema e compreender, a alto nível, qual a solução que é esperada. São também apresentadas soluções distintas da projetada com o intuito de fazer uma comparação e perceber se a solução que se pretende obter acrescenta valor quando comparada com outras. Este ponto termina com a análise ao valor que a solução preconizada poderá trazer aos clientes, identificando pontos fortes, fraquezas, mais valias, entre outras características recorrendo a modelos como CANVAS e NCD.

O Capítulo responde de forma parcial à questão **Q1**. Através da análise à solução assim como a comparação com algumas soluções já presentes no mercado, é possível identificar algumas tecnologias que devem ser objeto de um estudo mais intensivo. Entre estas podem-se evidenciar as plataformas computacionais, sensores existentes no mercado, serviços disponíveis na núvem, entre outras. Nos capítulos que se seguem será possível responder a esta questão de forma integral.

3 Tecnologia relevante

Neste capítulo são apresentadas as tecnologias mais relevantes para a execução da solução final. São abordadas as tecnologias associadas a *Big data* e IoT estudadas e apresentada a forma como podem ser usadas. Este capítulo tem como objetivo completar a resposta à questão **Q1** e responder na integra às questões **Q2**, **Q3**, **Q4**.

3.1 Internet of Things

O termo IoT foi primeiro usado por Kevin Ashton em 1999 durante uma apresentação sobre a interação entre o existente RFID e a emergente Internet. Nesse ano, Neil Gershenfeld lançou o livro *When things Start to Think,* que, não usando o termo IoT, conseguiu demonstrar o caminho que iria ser feito no futuro [25].

Desde então, o IoT evoluiu de forma exponencial, catapultada também pelo aparecimento de novas tecnologias *wireless*, microserviços e pela natural melhoria do serviço de Internet. Esta convergência ajudou a diminuir a distância entre as tecnologias operacionais (*operational Technologies* - OT) e as tecnologias de informação (*Information technologies* - IT), permitindo, consequentemente, a recolha e análise de informação.

Neste momento espera-se que no ano de 2020 haja cerca de 30 biliões de dispositivos ligados entre si. Este crescimento representa uma oportunidade de negócio que se espera ultrapassar os três triliões de euros em 2025 [26].

IoT pode ser aplicado a diversas áreas entre as quais [27]:

- **Indústria aeronáutica** – Sensores a bordo das aeronaves emitem dados importantes sobre as suas condições, incluindo aspetos ligados aos passageiros e à sua saúde;
- **Indústria Automóvel** – A ligação entre *smartphones* e automóveis já esta estabelecida atualmente e informações sobre o veículo podem ser consultadas através deste. Entre as várias fontes pode-se incluir: pressão de pneus, desempenho do motor, condições ambientais, etc. Esta tecnologia não só favorece o condutor como o fabricante e, em

alguns casos, a própria seguradora que tem acesso a diferentes tipos de dados que podem vir a ser uteis;

- **Indústria de defesa e segurança** – o desenvolvimento de análise de dados, redes, sensores e tecnologias de processamento de imagem apresentam uma oportunidade de reforçar e garantir a capacidade de defesa, sendo que esta tecnologia está, já, aplicada em diversos países;
- **Indústria da Energia** – Projetos *smart cities* estão bastante presentes no dia a dia e tem como objetivo melhor a eficiência e usabilidade das redes melhorando processos do dia-a-dia;
- **Indústria da saúde** – A interação entre dispositivos médicos e aplicações permite, já, monitorizar estados de saúde em tempo real, evitando assim perdas de vida. Através de *wearables* os pacientes têm agora acesso aos seus dados biométricos podendo ser detetada em tempo real qualquer anomalia.

Atualmente ainda não existe uma definição convencional e globalmente aceite sobre o que é o IoT. De forma muito resumida, IoT é um sistema capaz de ligar o mundo físico à Internet. Esta ligação consegue produzir dados que, sendo analisados, podem melhorar diversos aspetos nomeadamente produtividade e eficiência. As interações neste sistema não se restringem à comunicação máquina-máquina (M2M). As pessoas têm também um papel importante e adicionam ao sistema a comunicação maquina-pessoa e vice-versa [28].

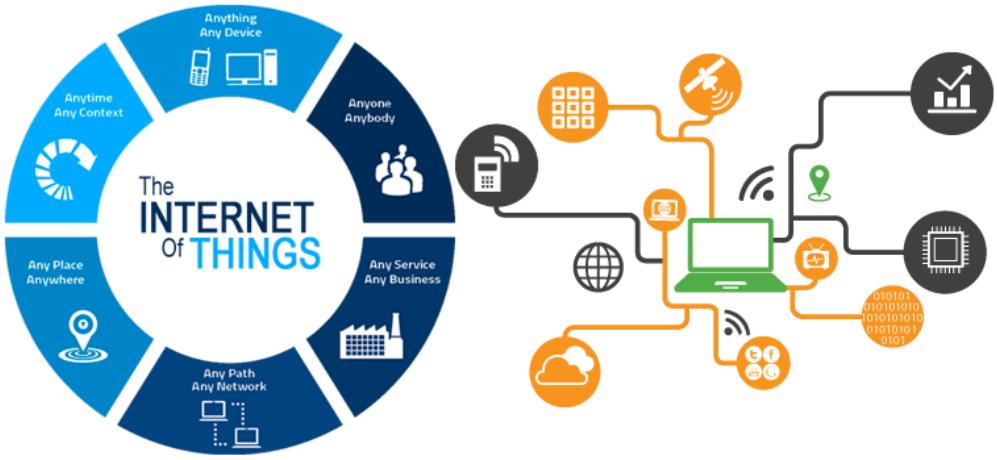


Figura 10 - Aplicabilidade de IoT

Fonte: https://courses.etcoe.in/Internet-of-things-training-coimbatore.php

O objetivo do IoT é claro, permitir que objetos se liguem em qualquer momento, em qualquer lado, autonomamente ou não usando uma ligação qualquer à Internet. Porém, devido à sua vulnerabilidade a ataques, é necessário ter alguma atenção no que toca à segurança dos dispositivos. A ligação entre milhões de dispositivos envolve uma quantidade de informação significativa. Esta informação não pode ser acessível por utilizadores não autorizados e por isso deve haver alguma preocupação com a segurança e privacidade. Sendo uma tecnologia relativamente nova, a segurança dos produtos IoT ainda não está no nível que seria desejado. Um dos problemas no uso destas tecnologias é o uso de passwords de rede inseridas

manualmente no dispositivo [29]. Este fator representa uma falha que pode ser explorada por pessoas que assim o desejam e que, consequentemente, podem dar acessos a sistemas outrora impossíveis de obter.

Para garantir a segurança, tanto os fabricantes como os utilizadores devem adotar boas práticas de desenvolvimento. Os primeiros devem assegurar aos clientes que os seus produtos foram testados exaustivamente e que apresentam mecanismos de defesa próprios. Caso existam vulnerabilidades, estas, devem também ser apresentadas de modo a que o utilizador tenha essa perceção e possa tentar ultrapassá-la. Os utilizadores devem também tomar algumas precauções como usar *software* de segurança, mudar passwords regularmente, encriptar dados e analisar constantemente o comportamento dos dispositivos, procurando encontrar irregularidades.

3.2 Big Data

O conceito de *Big Data* é um conceito relativamente novo. Em 2005 houve a perceção da grande quantidade de informação que estava a ser gerada pelos utilizadores da Internet [30]. Com este aumento, foi iniciado um processo de criação de sistemas que conseguissem armazenar e tratar destes dados. Desde então, o volume de dados cresceu exponencialmente e foram criadas tecnologias que permitem analisar informação ao ponto de encontrar padrões e tendências. Este facto pode gerar receitas significativas para as empresas que o fazem. Atualmente, estima-se que menos de 1% da informação gerada foi analisada, porem, prevê-se que por 2020 73% das organizações irão investir em *Big data*, reconhecendo o valor que pode trazer. Um estudo revela que esta tecnologia pode poupar à indústria da saúde cerca de 251 Biliões de euros [26].

Como alguns dados de interesse sobre *Big data* tem-se [3]:

- Em 2020, espera-se que cada pessoa gere cerca de 1.7 MB de dados por segundo;
- Há 3.5 biliões de pesquisas no Google por dia e cerca de 400 horas de vídeos são adicionados ao *Youtube* por minuto;
- Em média, são enviados 31 Milhões de mensagens por minuto no Facebook.

Com o avanço da tecnologia IoT, mais dispositivos vão estar em rede e, consequentemente, mais dados irão ser gerados.

A importância destes dados não se fica pela quantidade de dados, mas sim sobre aquilo que é feito aquando da análise destes. Esta análise permite, a curto/medio prazo:

- Reduzir custos;
- Economizar tempo;
- Desenvolver novos produtos que satisfaçam as necessidades dos clientes;
- Tomar decisões;
- Determinar causas de falhas ou atrasos;
- Detetar comportamentos fraudulentos;
- Etc.

Através destes dados, pode-se concluir que existe uma necessidade inerente às indústrias de realizar a analise de informação, tendo como objetivo final o crescimento das mesmas e consequentemente, maior sucesso.

Big data é caracterizado pelos 3 V's:

- **Volume** – As organizações recolhem dados de diversas fontes e armazenam-nos;
- **Velocidade** – Os dados são transmitidos com velocidades sem precedentes e devem ser analisados em tempo real;
- **Variedade** – Os dados possuem diversos formatos sendo que estes podem ser, ou não, estruturados.

Esta variedade nos dados requer uma tecnologia diferente do habitual para o seu armazenamento. As Bases de Dados (BDs) necessitam de ter uma liberdade maior, entrando em plano, bases de dados *schemaless*. Estas, não necessitam de conformidade com nenhum esquema (como tabelas, tipos de dados, etc.) e não apresentam limite no tipo de dados, podendo guardar informação independentemente da sua estrutura. Estas BDs são denominadas de NoSQL.

As BDs NoSQL apresentam uma grande capacidade de armazenamento assim como alta disponibilidade. A sua implementação revela-se bastante eficaz e muito económica o que levou várias empresas, Google incluída, a utilizar este tipo de tecnologias. Além destas, a grande mais valia das bases de dados *schemaless* é a capacidade de migração de dados que se revela bastante simples e rápida.

Existem vários tipos de bases de dados NoSQL:

- **Orientada a documentos -** Relacionam uma chave com uma estrutura de dados complexa, denominada de documento. Os documentos podem conter diversos pares de Chave/Valor ou até documentos embutidos;
- **Orientada a grafos -** São usadas para guardar informação sobre redes de dados, como conexões de uma rede social;
- **Chave/Valor -** São as bases de dados mais simples. Cada item da base de dados tem associado uma chave que o identifica;
- **Orientada a colunas** - Usa tabelas, linhas e colunas, porém os nomes e formatos das colunas podem variar de linha para linha. Bases de dados otimizadas para realizar *queries* sobre uma quantidade grande de dados.

Relativamente à distribuição de informação, geralmente, as bases de dados NoSQL permitem, nativa e automaticamente, espalhar a informação por diversos servidores sem que a aplicação dona destes dados se aperceba. Esta versatilidade possibilita fazer uma gestão dos servidores de forma a que nenhum seja sobrecarregado. A replicação de dados é também um fator importante e que está presente neste tipo de bases de dados. Algumas BD NoSQL permitem distribuir a informação por diversas áreas geográficas e possuem, de forma nativa, mecanismos de recuperação automáticos, não sendo necessário mão humana.

De seguida são apresentadas algumas bases de dados especificas para uso com *Big data.*

3.2.1 MongoDB

MongoDB é uma base de dados orientada a documentos que fornece alta *performance*, viabilidade e escalabilidade [31]. Os registos nesta base de dados são feitos através de documentos, que são estruturas de dados compostos por pares Chave/Valor, similares aos objetos *JSON.* Estes valores podem incluir outros documentos ou listas de documentos. A acoplação de documentos permite reduzir o número de I/O ao sistema e assim, evitar sobrecarga de pedidos. A vantagem de usar documentos deste tipo é que muitas linguagens de programação possuem forma de analisar esta informação através dos tipos nativos de dados e, evita-se assim, a criação de algoritmos de descodificação complexos.

MongoDB apresenta-se como uma base de dados de alta *performance* na persistência de dados. Através da sua escalabilidade, é possível distribuir a informação por vários nós em diversos centros de dados. Estes nós apresentam uma capacidade de armazenamento superior a 1 bilião de documentos sendo que suportam cerca de cem mil pedidos de leitura ou escrita por segundo, mantendo a latência mínima.

MongoDB e o uso no IoT

MongoDB permite armazenar, analisar e atuar sobre todos os detalhes de informação que o ambiente de IoT pode gerar [32]. Ao permitir alterar a estrutura de informação, é poupado tempo e dinheiro pois não existe a necessidade de fazer o *redesign* de toda a base de dados. Oferece análise de dados em tempo real sem a necessidade de investimento em armazéns de informação. A escalabilidade horizontal é também um ponto positivo. A MongoDB distribui a informação por diversos servidores (*Sharding*) e pode suportar milhares de nós nesta rede.

A segurança é também um fator importante, sendo que neste caso é assegurada através de autenticação, autorização e encriptação que protegem os dados dos sensores e o resultado das análises feitas a este.

3.2.2 Apache Cassandra

Apache Cassandra é uma base de dados distribuída, não relacional baseada em colunas [33]. Através da distribuição descentralizada de informação sobre diversos servidores (*sharding*), o Cassandra não apresenta nenhum ponto de falha visto que não existe um ponto central neste sistema. Nesta arquitetura, todos os nós têm um papel idêntico e comunicam entre si de forma igual. Esta comunicação é também responsável pela replicação da informação nos nós no mesmo servidor evitando assim falhas. Pode também haver a replicação para servidores distribuídos geograficamente. Esta arquitetura é responsável pela sua habilidade de escalonamento, *performance* e continuo serviço sem falhas. De forma a gerir os dados presentes na BD, a Cassandra oferece uma ferramenta de gestão denominada de Cassandra Query Language (CQL). Similiar ao SQL, típico de BDs SQL, o CQL delimita uma serie de funções capazes de gerir as tableas, linhas e colunas necessárias.

Devido à sua durabilidade, a Cassandra é mais aconselhável para situações onde não se pode perder nenhuma informação mesmo que o servidor tenha falhas. A sua arquitetura composta por nós é demonstrada na Figura 11

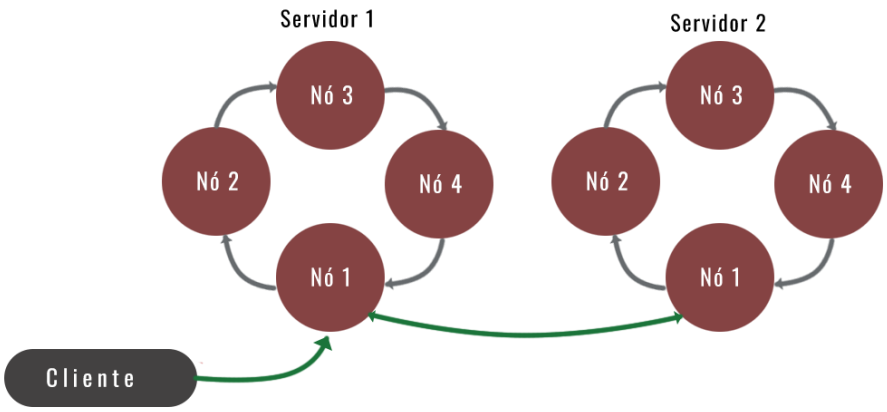


Figura 11 - Arquitetura Cassandra

Apache Cassandra e o uso no IoT

A elevada disponibilidade, escalabilidade e *performance* da Cassandra torna esta BD uma das mais usadas para o armazenamento e análise de dados proveniente de IoT [34]. Alguns dos usos mais frequentes com Apache Cassandra são: deteção de fraude, Chats de Mensagens, Personalização e *playlists* individuais.

3.2.3 Apache HBase

O Apache HBase é uma base de dados não relacional que tem como base da sua execução o sistema *Hadoop* [35]. Esta base de dados possui capacidade de escalabilidade linear ou modular o que a torna bastante eficaz a lidar com uma quantidade de registos considerável, sendo que a estrutura destes pode diferenciar.

O sistema é composto por um conjunto de tabelas, que, por sua vez, possuem linhas e colunas, sendo que cada tabela é caracterizada por uma chave primária. Esta chave é usada aquando de qualquer tipo de acesso ao HBase.

Ao contrário das bases de dados relacionais e com s*chemas*, o HBase permite juntar atributos naquilo que se chamam família de colunas, juntando assim diversos valores. A este conjunto podem ser adicionadas mais em qualquer altura, tornando o *schema* flexível e adaptável às diversas situações.

A sua arquitetura é composta por um nó mestre que gere o *cluster* de nós e os servidores regionais que são responsáveis por guardar e analisar a informação. Esta arquitetura é apresentada na Figura 12.

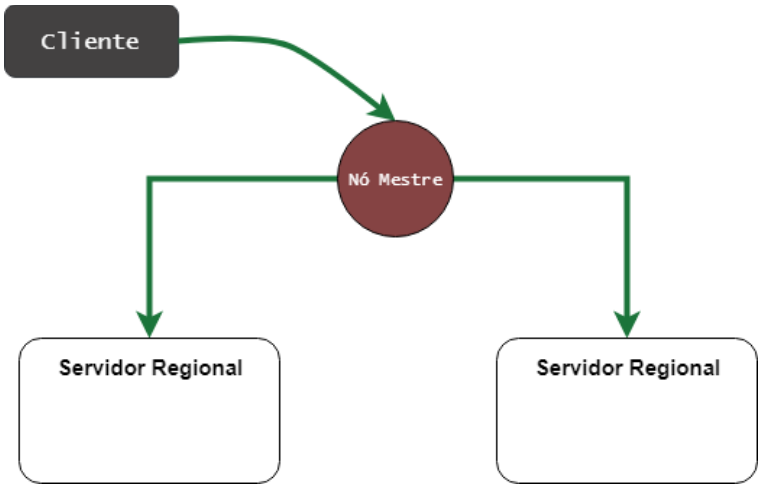


Figura 12 - Arquitetura HBase

3.2.4 Amazon DynamoDB

A DynamoBD é uma base de dados não relacional sem servidor capaz de guardar informação em grande escala tendo como base o modelo chave/valor [36]. O serviço oferecido é totalmente gerenciado e inclui características que ajudam os programadores a criar uma base de dados global e responsiva.

No serviço oferecido, inclui-se a replicação de dados, escalonamento automático quando necessário, encriptação, entre outros. Este serviço é o único não *open source*, logo acarreta um custo associado ao seu uso.

Sendo que esta BD está presente na nuvem existem várias mais valias que a tornam como uma possibilidade bastante coerente. Entre estas pode-se evidenciar o facto de a gestão de recursos ser feita de forma totalmente autónoma e automática, i.e. a BD nunca ficará sem capacidade de armazenamento de dados. Assim, não será necessário ter a preocupação da gestão e compra de *hardware*.

Além desta mais valia, BD DynamoDB apresenta também funções próprias para consumo e introdução de dados de forma direta. Assim, através de funções como *GetItem* ou *PutItem* é possível ter acesso a qualquer dado presente nas tabelas. Este acesso aos dados é feito através da arquitetura *Representational State Transfer* (REST) que permite a gestão e comunicação à BD.

3.2.5 Comparação

Da informação recolhida através do estudo, é possível fazer uma comparação entre as BDs apresentadas. A Tabela 4, em conformidade com os dados recolhidos de [37], apresenta essa comparação.

Tabela 4 - Comparação BDs NoSQL

	Mongodb	**Cassandra**	**HBase**	**DynamoDB**
Modelo base de dados	Documentos BSON	Colunas	Colunas	Documentos chave/valor
Modos de acesso	Protocolo proprietário usando JSON	CQL	Java API REST	REST
Responsável	MongoDB	Apache	Apache	Amazon
Licença	*Open Source*	*Open source*	*Open Source*	Comercial
Restrito Cloud?	Não	Não	Não	Sim
Métodos particionamento	*Sharding*	*Sharding* sem pontos de falha	*Sharding*	*Sharding*
Métodos de Replicação	Nó mestre responsável	Sim (Opcional)	Sim (Opcional)	Sim

Analisando a tabela é possível verificar a similaridade entre as quatro BDs. A diferença mais significativa e que pode afetar a escolha entre estas possibilidades é a licença de uso. As três primeiras colunas apresentam BDs *open source* que permitem o seu uso de forma grátis e flexível, porém, o seu uso acarreta o custo do material físico necessário para a sua implementação. A última coluna, DynamoDB é a única BD com uso restrito à nuvem. Este serviço apesar do custo que pode ter, não necessitará de custos adicionais de *hardware* ou manutenção.

3.3 Protocolos de comunicação IoT

A comunicação M2M pode ser feita através de diferentes protocolos. De seguida são apresentadas as possibilidades com maior relevância para esse efeito.

3.3.1 HTTP

O protocolo *Hyper Text Transfer Protocol* (HTTP) é usado para comunicação entre dispositivos ligados à rede [38]. É uma ligação sem estados e é necessário estabelecer ligação à rede sempre que se quer efetuar qualquer ação sobre a informação (ler - *Get*, guardar - *Post*, editar - *Put* ou eliminar - *Delete*). Cada um destes pedidos recebe uma resposta como demonstrado na Figura 13.

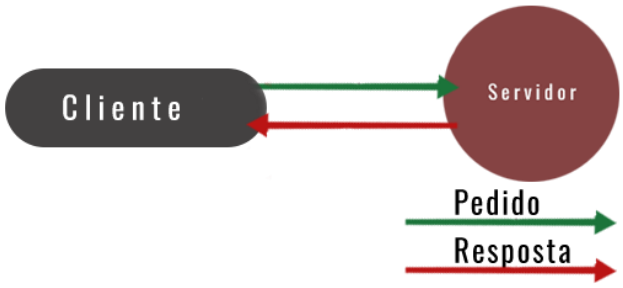


Figura 13 - Protocolo HTTP

Pode ser usado em IoT apesar de não ser um protocolo leve. Possui uma quantidade de *headers* significativa por pedido e não apresenta uma velocidade muito elevada. Sempre que é necessário obter alguma informação é feito um pedido *GET* ao serviço e este não garante que os dados recebidos são diferentes dos recebidos noutro pedido anterior. Isto gera, a longo prazo, uma sequência de pedidos que podem ser demasiado pesados para o servidor e, consequentemente, tornar o processo mais lento. A nível de segurança, o HTTP oferece uma implementação denominada de *Secure* (HTTPS) que adiciona uma camada de segurança ao protocolo. Esta, permite restringir a informação enviada entre cliente e servidor, evitando que terceiros consigam consumir.

3.3.2 MQTT

O protocolo *Message Queue Telemetry Transport* (MQTT) foi inventado no final dos anos 90 e apresenta-se como um sistema de transporte de mensagens através do mecanismo Publicação/Subscrição [39]. Esta característica, torna-o ideal para o uso em IoT visto que a comunicação existente nesta tecnologia se centra no M2M. Esta comunicação é feita através de TCP/IP o que garante comunicação assíncrona entre clientes e servidor. De forma a garantir a segurança dos dados, são usados os protocolos de segurança *Transport Layer Security* (TLS) assim como o seu antecessor *Secure Sockets Layer* (SSL). Através do uso destes, é possível garantir a privacidade e integridade dos dados entre emissor e recetor de mensagens.

A Figura 14 apresenta, de forma simples, como é a interação entre as partes.

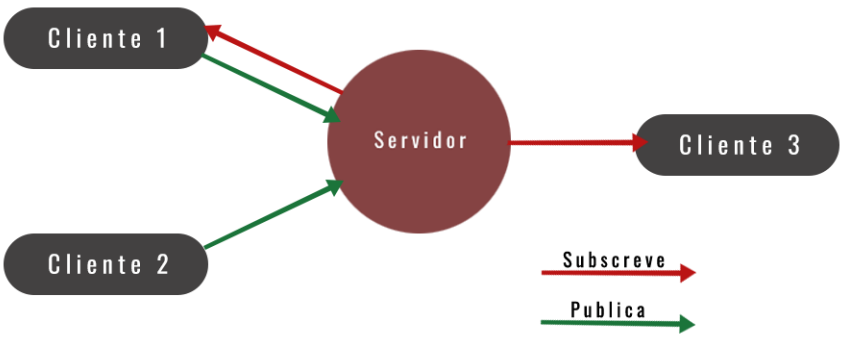


Figura 14 - Estrutura protocolo MQTT

Analisando a Figura 14, tem-se os seguintes componentes:

- **Cliente -** É um programa ou dispositivo que usa MQTT. Este, estabelece ligação à Internet e tem acesso ao servidor. Entre as principais funcionalidades podem ser destacadas a publicação de mensagens para tópicos e também a subscrição ou cancelamento da mesma para tópicos do interesse do cliente;
- **Servidor -** Denominado também de *Broker,* é um programa ou dispositivo que serve de intermediário entre clientes. Este tem como funções principais a ligação aos clientes e a consequente receção e envio de mensagens entre clientes. O servidor é também responsável pela gestão de tópicos, subscrições e cancelamentos.

As mensagens que são enviadas têm associadas a si um Tópico e um *Quality of service* (QoS) que permite garantir a entrega de uma determinada mensagem a um cliente. Além disso, o uso de QoS, permite garantir que, em casos onde a ligação à rede não é fiável, a mensagem é transmitida com sucesso. Este fator é essencial quando associado a uma fábrica onde a ligação poderá não ser estável. Este campo apresenta três possibilidades [40]:

- **QoS = 0** – No máximo uma vez: Este nível não garante a entrega da mensagem. É geralmente denominado de *Fire and Forget* (Dispara e esquece). O cliente envia a mensagem para o servidor, não esperando por uma resposta deste. A Figura 15 demonstra a relação existente neste caso.

Figura 15 - Publicação com QoS = 0

  Usado quando se possui uma conexão entre cliente e servidor muito estável (Ligação por cabo) ou quando não há interesse em garantir que todas as mensagens são transmitidas;
- **QoS = 1** – Pelo menos uma vez: Este nível garante que o cliente recebe a mensagem pelo menos uma vez. O cliente que publica a mensagem para o servidor espera por uma reposta por parte deste, denominada de *PUBACK*. Neste caso é possível que a mensagem seja enviada várias vezes até ser garantida a sua receção;

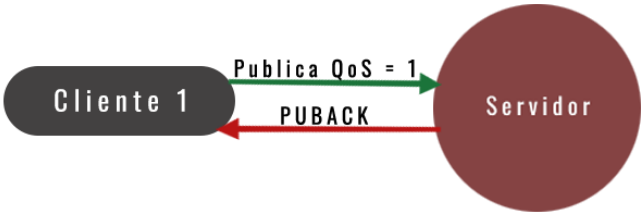


Figura 16 - Publicação com QoS = 1

Usado quando há necessidade de garantir que todas as mensagens são transmitidas desde que o sistema esteja preparado para lidar com dados duplicados. As mensagens são transmitidas significativamente mais rápidas que com QoS = 2;

- **QoS = *2*** – Exatamente uma vez: Este nível garante que o cliente recetor apenas recebe uma vez a mensagem. Apesar de ser o nível mais seguro, é também o mais lento. Exige no mínimo dois pares de pergunta/resposta entre cliente e servidor antes da mensagem poder ser apagada. O primeiro par, à semelhança do QoS = 1, garante que a mensagem é transmitida ao servidor esperando por uma resposta, denominada de *PUBREC*, que transporta o *id* com que a mensagem ficou no servidor. Caso não receba esta resposta envia novamente a publicação. Após a receção do PUBREC, o cliente guarda a informação recebida e apaga, localmente, a mensagem publicada, ficando apenas com o seu identificador. Depois deste processo, envia um *PUBREL* novamente para o servidor, que permite que a mensagem seja enviada para os subscritores, esperando um *PUBCOMP* que termina o processo e descarta toda a informação guardada. Caso haja uma falha neste processo, é da responsabilidade do cliente publicador reenviar a publicação inicial. A Figura 17 representa este fluxo de dados:

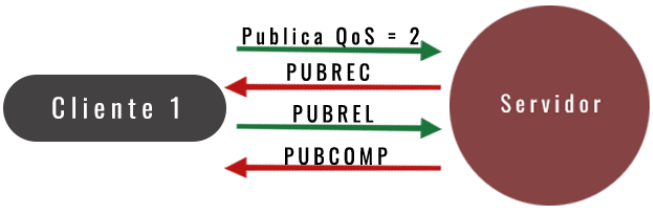


Figura 17 - Publicação com QoS = 2

Usado quando é critico para o sistema receber uma mensagem exatamente uma vez.

3.3.3 MQTT-SN

O MQTT-SN (*Sensor Network)* é uma variante do MQTT adaptada para comunicação sem fios com restrições numa rede de sensores [41]. Esta variante, usa o protocolo *User Datagram Protocol* (UDP) e apresenta-se como uma alternativa mais leve para transporte de mensagens. Foi desenvolvida para ser implementada em dispositivos que necessitem de ser alimentados por bateria própria e que tenham a possibilidade de ficar em modo *Sleep* de forma a não consumir a mesma.

As maiores diferenças para o MQTT *são:*

- A mensagem *CONNECT* é dividida em 3 partes sendo que a única obrigatória é a própria mensagem. As outras duas são opcionais e tem a responsabilidade de transmitir o tópico e último testamento do dispositivo;
- O tópico enviando num *PUBLISH* foi substituído por uma mensagem do tipo *long* com 2 *bytes*, denominada por *topic ID;*
- Foi introduzido o procedimento *Discovery* que permite a clientes descobrir os endereços de rede dos servidores operacionais;

- As mensagens são mantidas nos servidores através do procedimento *keep-alive* até que o cliente recetor acorde do seu estado de sono, permitindo assim poupança de bateria;
- A ligação do cliente ao servidor é feita através de *Gateways* ou de *Forwarders;*
- Apresenta quatro níveis de QoS. Com o novo nível, -1, a mensagem é enviada para o *gateway* e esquecida, não sabendo o destino final desta.

3.3.4 CoAP

O protocolo *Constrained Application Protocol* (CoAP) é um protocolo especializado em transferência *web* para uso em redes e nós restritos [42]. Foi desenhado para aplicação M2M sendo um protocolo aconselhável para automação e IoT. É baseado no modelo *REST* e dispõe os seus recursos através de URL's. O cliente pode aceder a estes, e tem acesso a ações de *GET, PUT, POST* e *DELETE.* O protocolo foi desenhado para funcionar em redes com baixa capacidade de transmissão de dados ou redes bastante constrangidas.

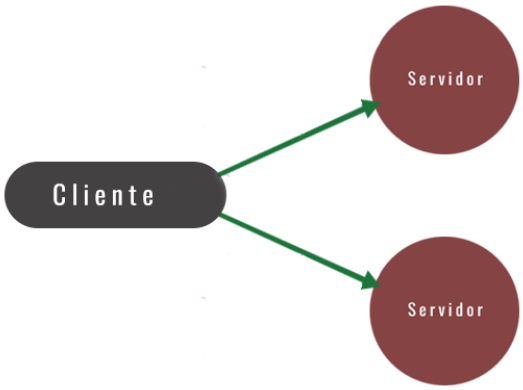


Figura 18 - Protocolo CoAP

O CoAP usa UDP para efetuar as suas comunicações. Isto permite realizar pedidos *multicast* ou *broadcast*, abrangendo todos os nós da rede, ou um grupo específico, respetivamente. Além disso, como se vê na Figura 18, é possível, para um nó, comunicar com diversos servidores, tomando assim, um lugar mais central na rede.

Assim como o MQTT, o CoAP apresenta também uma forma de garantir a entrega da mensagem [43]. Estas podem ser marcadas como *confirmable* ou *nonconfirmable*. A primeira, aquando do envio por parte de um cliente, tem como resposta um pacote *ack* que confirma o recebimento da mesma. Para o 2º caso, as mensagens são *Fire and Forget*, equivalente ao QoS 0. O CoAP oferece também a capacidade de um cliente se tornar um *Observer* para receber *updates* frequentes de um determinado tópico de interesse e assim evitar pedidos constantes. Para garantir a segurança, é usado o protocolo *Datagram Transport Layer Security* (DTLs) capaz de garantir autenticação, troca de chaves e proteção de dados em ambientes IoT. [44]

3.3.5 Comparação

A Tabela 5 resume o estudo feito e corelaciona com dados externos analisados [45].

Tabela 5 - Comparação protocolos de Comunicação

	HTTP	**MQTT**	**CoAP**	**MQTT-SN**
Modo de funcionamento	Pedidos ao Servidor	*Publicação/Subscrição* Tópicos	REST (Pedido/Resposta) URI's	*Publicação/Subscrição* Tópicos
Protocolo de transporte	TCP	TCP	UDP	UDP
Modo de comunicação	1 para 1	Muitos para muitos	1 para 1	Muitos para muitos
Consumo de energia	Elevado	*>CoAP*	*<MQTT*	*~CoAP*
Encriptação	HTTPS	TLS/SSL	DTLS	Encriptação rádio quando possível
Mensagens	Síncronas	Assíncronas	Assíncronas/Síncronas	Assíncronas
Qualidade de Serviço	-	3 Níveis (0,1 e 2)	2 Níveis (COM ou NON)	4 níveis (0,1,2,-1)
Tolerância a falha	SPoF[3] Servidor	SPoF Broker	Descentralizado	SPoF Broker

Analisando a tabela é possível verificar que as diferenças são substanciais entre os diversos protocolos. A escolha deve ser feita tendo em conta o ambiente onde vai ser instalado o sistema e a importância das mensagens a enviar. Entre os protocolos apresentados, o HTTP, apresenta-se como o menos indicado para sistema IoT e deve ser usado apenas em último recurso.

3.4 Plataformas computacionais de baixo consumo

Para criação de dados é necessário obter algum tipo de dispositivo que seja portátil, eficaz, independente e confiável. Para isso foi feita uma pesquisa por diversas fontes sobre dispositivos ou plataformas computacionais de baixo consumo que além de satisfazer todos as necessidades anteriores também tivessem um custo relativamente baixo. Além deste estudo, nesta secção, são também apresentados alguns dos sensores que poderiam vir a ser usados em alguns casos de uso.

[3] *Single Point of Failure* – Ponto de falha único

3.4.1 Arduino

Arduino é, de uma forma muito resumida, uma plataforma eletrónica capaz de correlacionar *hardware* com *software* [46]. As placas de Arduino conseguem ler *inputs* de qualquer tipo (Luz num sensor, ação num botão, …) e convertê-los em *outputs*, realizando uma ação de resposta especifica. Para isto é necessário programar a placa através da linguagem de programação Arduino baseado em C/C++.

A flexibilidade da placa Arduino fez com que seja usada em milhares de trabalhos abrangendo projetos do dia-a-dia até projetos cientificamente complexos. O facto de tanto a placa como o *software* serem *open source*, fizeram com que o Arduino fosse e seja um sucesso na criação de instrumentos científicos.

Algumas das vantagens do Arduino em relação aos outros sistemas é:

- **Custo** – As placas Arduino são relativamente baratas face à concorrência. Neste momento é possível adquirir uma placa UNO no Porto por cerca de 20€;
- **Multiplataforma** – O *software* de desenvolvimento de Arduino corre em Windows, Mac OS e Linux;
- **Ambiente de programação simples** – O ambiente de desenvolvimento é simples o suficiente para iniciantes sendo, ao mesmo tempo, flexível *q.b.* para que utilizadores mais experientes possam tirar vantagens deste;
- ***Open source* e *software* extensível** – A linguagem de programação é extensível através de bibliotecas C++;
- ***Open source* e *Hardware* extensível** – Sendo o *hardware open source*, os utilizadores podem criar os seus próprios módulos para melhorar e completar o que a placa já oferece.

3.4.2 Microcontroladores

Um microcontrolador é uma placa de pequena dimensão com a sua próprio Unidade Central de Processamento (CPU). Seguindo o registo da placa Arduino, um microcontrolador é um computador de pequenas dimensões que permite a sua programação para realizar uma função especifica. Entre as suas especificações, evidenciam-se algumas como:

- I/O digital e analógico;
- Temporizadores e contadores;
- Memória RAM (do inglês *Random Acess Memory*, é responsável por armazenar dados temporariamente, permitindo a sua leitura ou escrita de forma mais rápida);
- Capacidade de ligação a periféricos.

Além destas semelhanças com o Arduino, muitos dos microcontroladores possuem também a capacidade de se ligarem à rede por WiFi. Por esta razão, e devido ao seu reduzido preço, os microcontroladores apresentam-se como uma boa alternativa à placa Arduino.

Existem bastantes variantes de microcontroladores de diversos fabricantes. Devido à facilidade de aquisição e pelo seu preço, evidenciam-se dois tipos:

- **ESP8266** – Microcontrolador capaz de se ligar a redes 802.11 b/g/n e suporta comunicação TCP e UDP. CPU de 32 bits com 64 KB de memoria RAM. Padrão de segurança 802.11;
- **ESP32** – Microcontrolador mais capaz que o anterior. CPU 32 bits e 520 KB de memória RAM. Além da ligação à rede semelhante ao ESP8266, este, possui também *Bluetooth* v4.2. Possui também uma maior variedade de possibilidade de interação com periféricos [47].

A escolha deste tipo de microcontroladores deveu-se ao facto de ser possível, para ambos, a sua programação através do IDE do Arduino, facilitando a sua programação.

3.4.3 Sensores

O Arduino ou microcontroladores, por si mesmo, não apresentam grande valor para uma solução. Para que se tornem uteis em situações reais é necessário que estes consumam dados de diversos tipos, nomeadamente através de sensores ambientais. Estes sensores são caracterizados como pequenas peças de *hardware*, que, acoplados às placas apresentadas anteriormente, permitem a leitura de diversos dados. A Tabela 6 representa alguns desses aditivos e as suas características assim como o preço de venda ao público, à data de 13 de dezembro 2018.

Tabela 6 – Lista de Sensores compatíveis

Identificador	Descrição	~Preço
MPL3115A2	Pressão barométrica/ Temperatura/Altitude	9€
	Link: https://www.adafruit.com/product/1893	
SGP30	Qualidade de ar – CO2, VOC	17€
	Link: https://www.adafruit.com/product/3709	
MiCS5524	Qualidade de ar – CO, Amónia, Etanol, H2 e Metano/Propano/ Butano	13€
	Link: https://www.adafruit.com/product/3199	
DHT22/DHT11	Temperatura e humidade (22 mais preciso que o 11)	8€/4€
	Link: https://www.adafruit.com/product/385	
SI1145	Índice UV, Infravermelhos e luz invisível	8€
	Link: https://www.adafruit.com/product/1777	
eTape	Nível de líquido	35€
	Link: https://www.adafruit.com/product/464	

Identificador	**Descrição**	**~Preço**
ADXL335	Acelerómetro 3 eixos	13€
Link: https://www.adafruit.com/product/163		
Identificador	Descrição	~Preço
APDS9960	Proximidade, Luz, RGB e gestos	6€
Link: https://www.adafruit.com/product/3595		
SPH0645LM4H	Microfone	6€
Link: https://www.adafruit.com/product/3421		

Estes são apenas alguns dos elementos que podem ser utilizados. Dependendo do tipo de indústria, podem ser escolhidos diferentes sensores para controlo ambiental.

Resumidamente, através do uso deste *hardware* em compatibilidade com os microcontroladores ou Arduino, é possível efetuar leituras a fatores ambientais mantendo o investimento relativamente baixo. Os sensores, apresar do preço reduzido são de qualidade suficiente para efeitos de teste e permitem efetuar simulações credíveis.

3.4.4 Raspberry Pi

O Raspberry Pi é, de forma resumida, um computador de tamanho reduzido, originalmente criada para fins educativos. Ligando-se a periféricos normais (teclado, rato e monitor) pode efetuar qualquer tipo de ação que outros computadores fazem trazendo a vantagem da sua portabilidade [48]. Este computador corre sobre sistema operativo Linux e devido ao seu reduzido tamanho e preço, revela-se como um bom componente a ser usada para uma prova de conceito.

Na execução deste projeto, o Raspberry Pi pode ser usado em conformidade com o Arduino podendo servir de sensor ou de intermediário entre sensores e outros serviços.

Relativamente ao preço, atualmente, é possível adquirir o computador por cerca de 40€ sendo que este já apresenta uma complexidade e capacidade maior face ao próprio Arduino.

3.5 Plataformas *open source* de Gestão IoT

A produção de dados por parte dos sensores não teria valor se estes se perdessem de seguida. Para que isso não aconteça é necessário abordar diversas formas que, de alguma forma, recebam estes dados, processem-nos e reajam quando necessário. Para isso foram abordadas algumas plataformas de gestão de IoT *open source* que serão de seguida apresentadas.

3.5.1 ThingsBoard

ThingsBoard é, de forma resumida, uma plataforma de gestão de dispositivos IoT *open source* que permite escalabilidade, manutenção e desenvolvimento de projetos baseados em IoT. Sendo *open source* permite ser implementado na nuvem estando já implementado em cerca de 2000 servidores. Quando usado em *cluster*, todos os nós são idênticos e suportam a mesma carga, não havendo um nó mestre [49]. A conectividade entre dispositivos pode ser feita através dos protocolos MQTT, HTTP, CoAP ou OPC-UA. Esta última arquitetura, foi desenvolvida pela fundação OPC e tem como objetivo a comunicação M2M orientada a serviços.

Esta ferramenta foi desenhada para ser:

- **Escalável** – Plataforma escalável horizontalmente;
- **Tolerante a falhas** – Não apresenta SPoF. Cada nó é idêntico o que evita perda de dados;
- **Robusto e eficiente** – Servidores com apenas um nó podem suportar dezenas ou até centenas de milhares de dispositivos. Em modo *cluster* suporta milhões;
- **Customizável** – Adicionar novas funcionalidades é simples através dos nós de regras de negócio;
- **Duradouro** – Não existe perda de informação.

A *ThingsBoard* oferece três soluções de gestão:

- *ThingsBoard Community Edition;*
- *ThingsBoard Professional Edition;*
- *Thingsboard IoT Gateway.*

A edição profissional deste serviço acarreta um preço para implementação. Esta, pode ser feita em servidores *Amazon Web Services* (AWS) ou no Azure.

Relativamente ao terceiro, *IoT Gateway*, como o próprio nome indica, é usado como intermediário entre sensores IoT e outros sistemas. Este serviço oferece ferramentas como [50]:

- Extensão MQTT para controlar, configurar e recolher dados dos dispositivos;
- Persistência de dados para garantir que existe entrega da informação em caso de falha de sistema;
- Em caso de perda de conexão, esta é reposta automaticamente;
- Mapeamento de informação recebida pelos dispositivos.

3.5.2 MainFlux

À semelhança da solução anterior, esta apresenta-se também como uma plataforma *open source* para gestão de soluções para IoT [51]. Define-se como uma plataforma de elevada segurança e escalável que serve de infraestrutura de *software* para o desenvolvimento de soluções de IoT. As suas principais funcionalidades são:

- **Conectividade** – Pode ligar-se a um grande número de dispositivos e *gateways* através de diferentes protocolos (HTTP REST/MQTT/CoAP/Websocket (Ligação TCP entre cliente e servidor que permite interação bi-lateral));
- **Sistema de Servidor** – Pode servir de servidor para gestão de dispositivos e dados;
- **Gestão de eventos e análise de dados** – Processamento de eventos, analise de dados e criação de relatórios;
- **Plataforma *Cloud***;
- **Segurança** – Conexões seguras através de TLS e DTLS. Servidor com autorização e autenticação.
- ***Gateway* –** À semelhança da solução anterior, esta apresenta também a possibilidade de possuir um dispositivo *gateway* responsável pela receção de dados dos sensores e envio das mesmas para outros sistemas.

3.5.3 Kaa

Kaa é uma plataforma de IoT *open source* baseada em microserviços capaz de se adaptar a qualquer necessidade e aplicação. É escalável desde o uso doméstico ao uso comercial e industrial permitindo o *deployment* em soluções multi *Cloud* [52]. Esta permite funcionalidade como:

- Gestão de dispositivos via *cloud*;
- Recolha e análise de dados;
- Processamento de dados e criação de pipelines de análise;
- Segurança de dados;
- Conectividade usando protocolos (MQTT e CoAP);
- Construção de interfaces através de *widgets*;
- Capacidade de fazer *updates* a dispositivos.

Relativamente às características tecnológicas e arquiteturais, esta plataforma, apresenta uma abundante liberdade de escolha. Através do uso de *dockers* permite o uso de diversas linguagens de desenvolvimento para os microserviços, nomeadamente *Scala, R*, *Java*, entre outras. Esta plataforma permite também o *deployment* do serviço em plataformas *cloud* oferecidas por empresas como *Google, Amazon, Alibaba*, etc. Para comunicação entre dispositivos e sistema podem ser usados os protocolos MQTT ou CoAP. Relativamente à segurança dos dados, o Kaa utiliza o protocolo TLS ou DTLS que garante a autenticação dos dispositivos através da verificação de certificados e chaves privadas.

Esta plataforma apresenta-se, assim, como uma solução escalável, de elevada disponibilidade e capaz de recuperar automaticamente após falha.

3.5.4 Comparação

Relativamente as três ferramentas *open source* estudadas, é complicado concluir qual a mais poderosa visto que oferecem capacidades muito semelhantes. Para efeitos de comparação é apresentada, a Tabela 7 que expõe as três possibilidades.

Tabela 7 - Comparação Plataformas IoT Open source

	ThingsBoard	**Mainflux**	**Kaa**
Gestão Dispositivos	✓	✓	✓
Compatibilidade *Cloud*	✓	✗	✓
Segurança	SSL	TLS/DTLS	TLS/DTLS
Conectividade	MQTT, HTTP, CoAP, OPC-UA	HTTP, MQTT, WebSocket, CoAP	MQTT, CoAP
Análise de dados	✓	✓	✓
Interfaces para amostra de dados	✓	✓	✓
Bases de dados	Cassandra	Cassandra, MongoDB, InfluxDB	Cassandra, MongoDB, Hadoop, Oracle NoSQL

3.6 Plataformas *Cloud* de Gestão IoT

Além das plataformas *open source* já apresentadas existem também a possibilidade de o serviço ser integrado na *cloud*. De seguida são apresentadas algumas soluções de empresas reconhecidas como Amazon, Google e Microsoft.

3.6.1 AWS IoT

O AWS IoT é um serviço seguro e escalável oferecido pela empresa *Amazon* presente na *cloud* que permite a gestão IoT a nível industrial, doméstico, etc. Este, é composto por duas secções distintas [53] representadas na Figura 19.

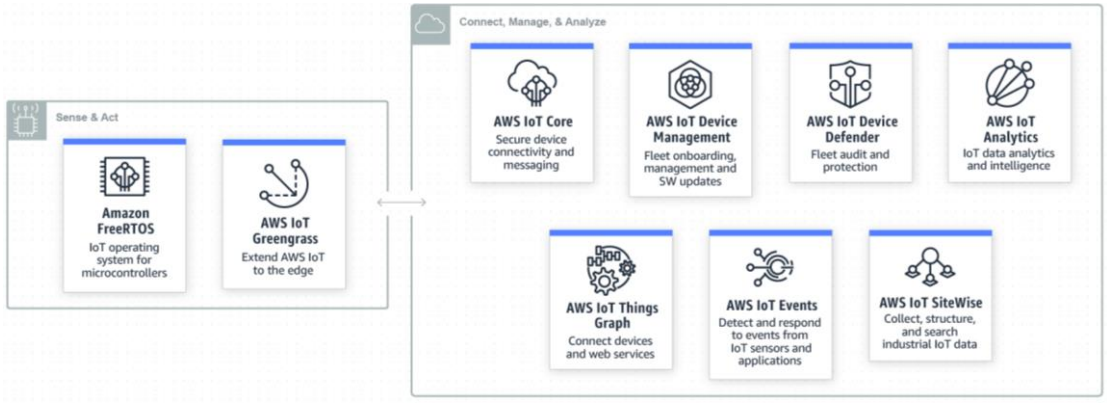


Figura 19 - Arquitetura Amazon AWS IoT
Fonte: https://aws.amazon.com/iot/

- ***Edge*** (Secção esquerda da figura)**–** Responsável pela ligação dos dispositivos à rede de forma segura. Tem também a função de recolher e analisar a informação em tempo real localmente, evitando a necessidade do envio e análise desta na *cloud*. Isto permite que, em caso de falha de rede, os dados possam ser analisados e tomadas ações sem necessidade de ligação;
- ***Cloud* –** Através de diferentes ferramentas, o AWS oferece na sua *cloud*, serviços de segurança de dispositivos e dados, gestão de dispositivos e capacidade de *update,* análise de dados, ligação a serviços web, criação de eventos através da análise de dados e o próprio armazenamento dos dados, analisados ou não. Além destas oferece também a possibilidade do uso de serviços de *machine learning* que trarão valor acrescentado a uma possível solução;

A Tabela 8 demonstra quais os serviços e funcionalidades distintas oferecidos pelo AWS IoT.

Tabela 8 - Serviços AWS IoT

	Descrição
AWS IoT	Plataforma *cloud* de gestão de dispositivos. Inclui serviços como: • AWS IoT Core; • AWS IoT Device Management; • AWS IoT Device Defender; • …
Amazon FreeRTOS	Sistema de controlo de microcontroladores. Instalado nos próprios sensores, é usado para programar e controlar pequenos dispositivos de pouco consumo energético. Faz a ligação entre dispositivos e o AWS IoT Core ou o AWS GreenGrass.
AWS GreenGrass	*Software* que permite, de forma segura, efetuar alguma lógica de negócio localmente. Caso exista falha de ligação à rede, este serviço mantém os dados até que esta seja reposta, impossibilitando perda de dados. Além disso pode responder a eventos através da análise destes dados.

Ao uso deste serviço está associado um custo à empresa. De seguida são apresentados os custos para o servidor situado em Frankfurt. No momento da criação deste exemplo o valor do dólar americano apresenta um valor em Euro de 0,88€ (13/12/2018):

- **Conectividade -** A conectividade é medida em incrementos de um minuto e é cobrada por milhão de minutos de conexão. Para o servidor em questão, está associado um valor de aproximadamente 0,084€. Sendo assim, para um dispositivo que está ligado 24/7, por ano o custo seria de:

$$1\ Conexão * \frac{0.084}{1000000} minutos * 525600 \frac{minutos}{ano} = \text{0,044€}$$

- **Mensagens -** As mensagens transmitidas entre os dispositivos e o AWS IoT são cobradas em conjuntos de 1 milhão. Para o servidor europeu, um sistema que possua um volume mensal de 1 bilião de mensagens apresenta um custo de 1,06€. O valor por milhão de mensagens diminui conforme o número total de mensagens por mês. Cada mensagem possui um tamanho máximo de 128 *Kilobytes* (KB);
- ***Device Shadow* e registo de dispositivos -** O *device shadow* armazena o estado desejado ou real de um determinado dispositivo. O uso deste serviço e do registo de dispositivos é medido pelo número de operações que acedem ou modificam dados do registo ou do *shadow*. Por um milhão deste tipo de operações, é cobrado um valor de 1,32€. Estas operações possuem um valor máximo de um KB. Caso ultrapassem, são medidas como duas ou mais operações, conforme o tamanho;
- **Mecanismo de Regras -** O uso do mecanismo de regras é cobrado em dois campos distintos: por milhão de regras acionadas e por milhão de ação executadas. Para o servidor europeu, ambas possuem o valor de 0,16€.

3.6.2 Google Cloud Plataform IoT

A Google presenta também uma solução *cloud* para gestão IoT. Similar ao serviço oferecido pela Amazon, o *Google Cloud Plataform* (GCP), apresenta uma variedade de serviços capazes de fazer uma gestão eficaz de dispositivos e dados. Entre estas, estão serviços como *Machine Learning* que usam *TensorFlow* e que são introduzidos localmente, maximizando, assim, o seu potencial [54]. A Figura 20 apresenta todos os componentes da solução GCP.

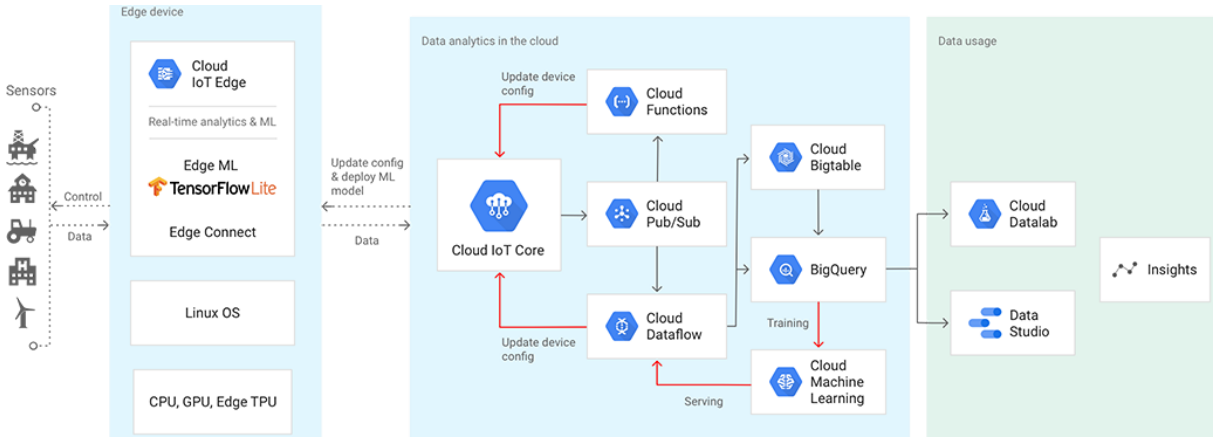


Figura 20 - Arquitetura GCP

Fonte: https://cloud.google.com/solutions/iot/

Entre as funcionalidades oferecidas pelo GCP, evidencia-se:

- **Cloud Iot Core -** É a base do produto IoT. É um serviço de gestão que permite, de forma segura, ligar-se e gerir dispositivos remotamente e analisar dados por estes provenientes. Define-se como uma solução completa capaz de guardar, processar, analisar e visualizar dados que são geridos de forma automática sem necessidade de intervenção humana [55];
- **Cloud Pub/Sub -** Serviço que serve de intermediário entre dispositivos e a *Cloud*. Recebe dados provenientes de dispositivos e tem como função reenviá-los para o serviço de processamento **Cloud Dataflow** ou para armazenamento. Para integração dos dispositivos, este serviço oferece bibliotecas para diferentes tipos de linguagens como: Android, JavaScript, iOS, PHP, Python e Ruby [56];
- **Cloud IoT Edge -** Como o nome indica, este serviço permite estender os serviços oferecidos pelo GCP aos dispositivos de forma local. É composto por dois componentes: **Edge IoT Core** e **Edge Machine Learning**. Este serviço apenas pode correr em ambientes Linux ou *Android things.*

Relativamente ao custo, o GCP IoT é cobrado pelo número de operações, pelo custo de armazenamento e pela ligação.

3.6.3 Azure IoT

A solução da Microsoft é semelhante às já apresentadas. Este serviço, denominado de Azure IoT, oferece um número de serviços *cloud* considerável, incluídos serviços de *mobile*, armazenamento, máquinas virtuais, entre outros. Entre os seus componentes, tem como serviços chave:

- **Azure IoT Hub -** Considerado como uma PaaS, este serviço é a base para construção de aplicações IoT. Permite ligar, monitorizar e gerir biliões de dispositivos de forma segura. A comunicação é bidirecional o que permite a atualização de *software* de dispositivos

de forma remota. Este serviço fornece diversos *Software Development Kits (SDK's)* para diferentes linguagens nomeadamente .NET, JavaScipt, Java, C e Python [57];

- **Azure IoT Edge -** Assim como no GCP e AWS, também a Microsoft possui um serviço que estende a capacidade da *cloud* aos dispositivos localmente. Este serviço é caracterizado pela baixa latência e tempos de repostas quase imediatos que por si só são uma mais valia no projeto aqui descrito [58];
- **Azure IoT Central -** Segundo a Microsoft, **Azure IoT Central** é um *Software as a Service* (SaaS) que permite a gestão de soluções IoT de forma global sem a necessidade de conhecimentos *cloud*. A sua utilização é feita através de uma interface simples que permite ligar, gerir e controlar o acesso de milhões de dispositivos [59];
- **Azure IoT Solution Accelerators -** Este serviço é um fator relevante para a escolha de um sistema. Entre outras possibilidades, esta ferramenta permite efetuar a simulação de dispositivos, o que permite testar as soluções criadas com milhares de dispositivos sem que haja a necessidade de os ter fisicamente [60].

Relativamente ao custo, quando comparado com os outros provedores de serviço, o Azure consegue fornecer um serviço a um preço mais reduzido, sendo também, este, um ponto positivo.

3.6.4 Comparação

De seguida é apresentada uma comparação entre as três possibilidades aqui apresentadas. A escolha entre estas é complicada visto que apresentam ferramentas e componentes semelhantes sendo que o fator financeiro pode ser decisivo na escolha.

Tabela 9 - Comparação Plataformas Cloud IoT

	AWS IoT	**Azure IoT**	**GCP IoT**
Responsável	Amazon	Microsoft	Google
Protocolos	HTTP, MQTT	HTTPS, AMQP, MQTT	MQTT, HTTP
SDK[4]	C, Node.JS, Java, Python, iOS	.NET, Java, C, Node.JS, Python	Java, JS, iOS, PHP, Python, Ruby
	AWS IoT	**Azure IoT**	**GCP IoT**
Funcionamento na *Edge*	✓	✓	✓
Segurança	TLS	TLS	TLS
Replicação virtual de dispositivos	✓	✓	Não Completo. Apenas guarda o estado dos dispositivos
Comunicações C - *Cloud* D - Dispositivos	C -> D D -> C	C -> D D -> C	C -> D D -> C
Custos (Ordem mais caro para mais barato)	2	3	1

[4] Permite o desenvolvimento de aplicações nos seguintes ambientes/linguagens

Tendo em conta estes dados, e analisando as capacidades de cada plataforma, é possível notar que, no momento de escrita do documento, a plataforma da AWS é a mais capaz. Esta apresenta-se pioneira em grande parte dos seus componentes e continua a trabalhar no funcionamento dos mesmos. Em termos de custo, apesar de não ser a mais barata, apresenta valores justos para o seu uso, oferecendo também um *free tier* em todos os seus componentes que permite o teste dos mesmos.

3.7 Sumário

O Capítulo começou por complementar a resposta à questão **Q1**. O estudo das tecnologias IoT e *Big Data* deram uma melhor perceção do que seria necessário para obter uma solução final compatível com os objetivos traçados. Nas duas secções seguintes respondeu-se às questões **Q2** e **Q3**. À primeira foi dada uma resposta total sobre o tipo de dispositivo seria necessário utilizar de forma a poder controlar o ambiente de um determinado local. O estudo iniciou-se pela placa Arduino que acabou por ser substituída pelos microcontroladores devido à sua capacidade de ligação à rede e preço reduzido. A questão **Q3** foi respondida de forma parcial. Apesar de apresentados alguns sensores mais comuns, esta questão, para ser respondida de forma completa ter-se-ia de estudar todos os sensores existentes no mercado, acabando por ser uma lista algo extensa. Finalmente, nas duas últimas secções respondeu-se à questão **Q4** referente à forma como se devem processar os dados depois de emitidos. Para esta resposta foram estudados dois tipos diferentes de plataformas: *cloud* e *open source* que, no total, ofereciam seis possibilidades de solução a esta questão. A resposta a esta questão é também valorizada através do estudo realizado ao Raspberry Pi realizado no Subcapítulo anterior. Este, devido às suas capacidades, poderá servir também como uma plataforma de gestão de dados.

4 Desenvolvimento da solução

Neste Capítulo apresenta-se toda a arquitetura e preparação de uma possível solução para o problema e que representa uma possível resposta à questão **Q6**. Começa-se por efetuar uma avaliação às tecnologias apresentadas anteriormente complementando as respostas às questões já referidas. De seguida é estruturado todo o sistema nos seus diferentes componentes e serviços. Por fim é feita a referência a todo o trabalho realizado para a implementação da arquitetura estruturada.

4.1 Avaliação das possíveis soluções

A composição da solução final acarreta diversas escolhas de tecnologias. A Figura 21, representa a sequência da informação desde a produção dos dados até à consulta dos mesmos pelo cliente. Através da informação já obtida com o estudo realizado, é possível neste capítulo abordar possíveis respostas às questões introduzidas na secção 1.3.

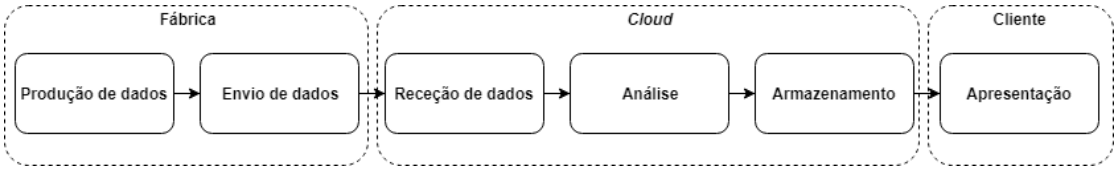


Figura 21 – Fluxo de Informação

Começando pela **Fábrica**, a produção de dados será realizada, numa fase inicial, através do microcontrolador ESP32, sendo que esta é a resposta à pergunta **Q2** nesta fase de desenvolvimento do projeto. Esta placa, pelo seu preço reduzido e capacidade de suporte para diferentes sensores torna-se uma boa escolha para a prova de conceito necessária. O uso do ESP32 permite uma escolha diversa dos sensores. Esta oferta variada permite responder à pergunta **Q3** relativa ao tipo de sensores. O serviço de *gateway*, responsável pelo envio dos dados, também será inserido no ambiente fabril, e implantado num Raspberry Pi que é responsável por fazer todas as comunicações ao *backend* via Internet. As comunicações entre

dispositivos e *gateway* são feitas através do protocolo MQTT. Esta ligação pode também ser realizada por outros protocolos, dependendo da capacidade dos sensores. Em ambiente de produção, poderão ser usados protocolos *offline* como *Bluetooth* ou OPC-UA em união com o protocolo MQTT, garantindo que existe sempre a entrega dos dados. O uso de um *gateway* permite a analise de dados em tempo real mesmo que não exista ligação à rede. Esta capacidade permite responder à pergunta **Q4** que incide sobre o tratamento de dados. No *gateway* é feita primeira abordagem a esta análise sendo que de seguida, na *cloud*, irá ser feito uma análise mais detalhada recorrendo a outros recursos com maior capacidade.

No *Backend*, inserido na ***Cloud*** é onde se reside a maior duvida na escolha da tecnologia. Esta surge no facto de haver duas possibilidades de implementação: Serviços *open source* ou serviços *cloud* (Também denominados de *serverless*). A dificuldade entre os dois deve-se a diversos fatores, como:

- **Preço pelo serviço** – Numa primeira fase, com um número de dispositivos a comunicar, o serviço *serverless* apresenta um valor menor. Com o aumento do número de dispositivos, estes valores podem subir de forma bastante acentuada, sendo que é necessário averiguar se é uma boa escolha;

- **Escalabilidade** – Ambas as possibilidades oferecem a escalabilidade horizontal necessária para o projeto. Nas plataformas *cloud* este processo é todo automático, sendo esta apenas a vantagem sobre as plataformas *open source*;

- **Segurança** – A segurança é assegurada através de diferentes protocolos já mencionados;

- **Suporte** – O suporte em caso de problemas é superior para os serviços *cloud*. As empresas que fornecem estes tipos de serviços apresentam uma elevada disponibilidade e continuidade de trabalho. Este último ponto é também importante. O surgimento de novas ferramentas será de fácil integração com o sistema quando usados serviços *serverless*.

Na última parte do sistema é necessário mostrar a informação recolhida e analisada ao **Cliente**. Para isso será criada uma plataforma *web* para amostra desta informação, usando *widgets* específicos para simplificar a interface com o utilizador. De forma a dar uma maior acessibilidade a estes dados será também desenvolvida uma aplicação móvel (Híbrida) que permitirá a portabilidade da solução. Será através desta que o gestor/responsável pela fábrica será alertado de possíveis anomalias ou acidentes no ambiente de trabalho. Caso necessite, pode também criar avisos de *email* ou mensagens aquando de um acontecimento. Estas funcionalidades permitem dar resposta à pergunta **Q5**.

4.2 *Design* da solução

De forma a atingir os objetivos propostos para a solução deste problema, pretende-se o desenvolver uma solução capaz de responder a todos os problemas associados às tecnologias

já apresentadas. Tendo isto em conta, existe uma série de funcionalidades que necessitam de estar presentes no sistema final, nomeadamente:

- Disposição de dados em tempo real;
- Identificação de estados de sensores;
- Estipulação de limites de valores;
- Em caso de deteção de valores anómalos, avisar responsáveis/Executar ações;
- Elevada disponibilidade;
- Persistência de dados.

Além destas funcionalidades, existem outras que, não sendo acessíveis a clientes, devem ser tidas em consideração durante o desenvolvimento da solução, como fiabilidade, escalabilidade, segurança, entre outras.

Para que este objetivo possa ser cumprido, são necessários três componentes chave, apresentados na Figura 22.

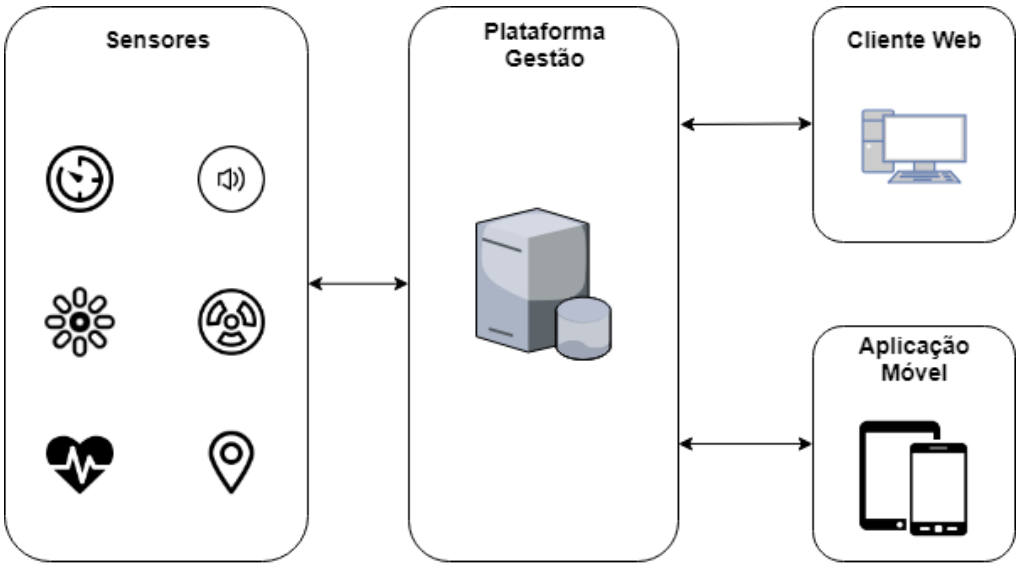


Figura 22 – Estrutura do sistema

Começando pelo componente **Sensores,** estes, conectados aos microcontroladores, serão colocados em pontos estratégicos da fábrica consoante a necessidade de monitoramento, não havendo a obrigação de inserir sensores em todas as áreas de ação da planta da fábrica. Isto permite a redução de custos para o cliente e a especificação dos pontos críticos das suas instalações. A escolha de sensores depende do ambiente de trabalho do cliente e pode abranger diversos tipos de dispositivos. Estes dispositivos poderão, numa fase posterior, ser obtidos através de fornecedores oficiais.

A **plataforma de Gestão** é responsável pela receção de dados provenientes dos sensores e tem como função analisar e guardar estes mesmos dados. Durante a sua análise podem ser ativadas diversas ações, conforme a situação. Esta plataforma é comum a todos os clientes, havendo, contudo, separação de dados e responsabilidades. Estará alojada na *cloud* o que permitirá uma

elevada disponibilidade e confiabilidade. Para a persistência de dados, será usada uma base de dados *NoSql* alojada também na *cloud*.

Relativamente ao lado **Cliente** da solução, será criada uma plataforma para monitoramento dos dados. Esta solução permite que sejam consultados os dados em qualquer dispositivo, desde que possua *browser*. A plataforma possibilita também efetuar algumas ações manualmente como acionar alarmes, avisar trabalhadores, entre outras opções, consoante as necessidades do cliente. De forma a estender esta possibilidade, o desenvolvimento da plataforma deve ter em atenção o uso de dispositivos móveis permitindo aceder às mesmas funcionalidades independentemente do tamanho e mantendo a interface com o utilizador dinâmica e acessível.

4.3 Arquitetura

Na escolha da arquitetura foi decidido, em conformidade com os responsáveis pelo projeto por parte da empresa Abaco Consultoreso, o uso da plataforma AWS IoT. Como já referido, esta é, atualmente, a plataforma com melhor relação preço/qualidade e apresenta ferramentas que podem aumentar o valor da solução final. Além disto, o uso de tecnologias AWS é também um fator importante para os possíveis clientes que conhecem o valor e capacidade da mesma. A Figura 23 representa os serviços a usar na solução final.

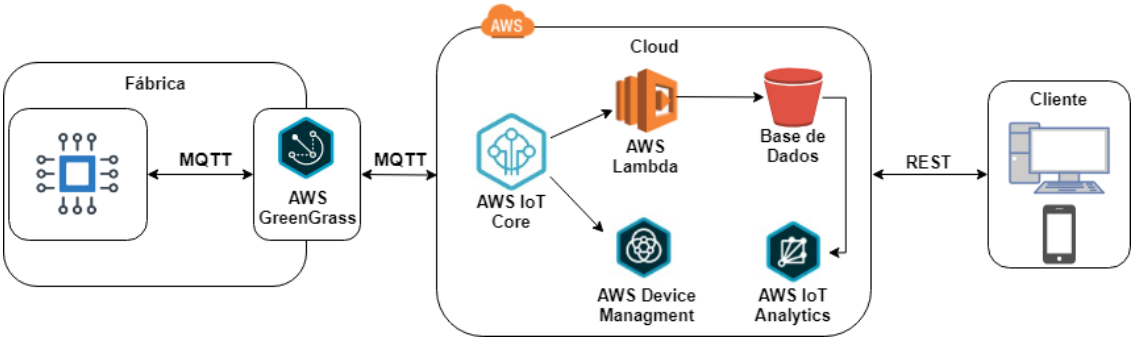


Figura 23 – Uso de Serviços AWS

Dependendo do tipo de sensores, a conexão ao *gateway* pode ser feita por diferentes protocolos como *Bluetooth*, *ZigBee,* MQTT, entre outros. Nesta fase do projeto, devido ao uso do microcontrolador ESP32, é usado MQTT para fazer a ligação:

- **AWS GreenGrass –** Esta ferramenta permite a análise de dados em tempo real na planta da fábrica onde está inserida. Este serviço não é necessariamente obrigatório para um sistema IoT, sendo que os dispositivos podem comunicar diretamente com a *cloud*. O uso, nesta situação, deve-se à necessidade da análise de dados mesmo quando não existe ligação à rede. Através do AWS GreenGrass garante-se que os dados são examinados e as ações executadas. Além desta vantagem, o uso de um *gateway* permite o controlo de envio de mensagens para a *cloud* que pode reduzir os custos associados ao sistema.

- **AWS IoT Core –** Este componente recebe os dados provenientes do AWS GreenGrass e está responsável por criação de regras e gestão de dispositivos. Através do *Device shadow* que este componente oferece, é possível ter uma replicação de um determinado dispositivo de forma virtual. Esta ferramenta permite gerir qualquer dispositivo e averiguar falhas ou problemas associados a estes. A definição de regras é também um fator importante. São estas que determinam quais as ações a executar no caso da deteção de dados que não cumpram as normas. É também através deste componente que são geridos os dispositivos. Aquando da criação virtual de um dispositivo é gerada uma chave associada a esse que permite a sua identificação e gestão;
- **AWS Device Managment –** O uso desta ferramenta irá permitir atualizações em massa aos dispositivos. Selecionando uma família de dispositivos, é possível alterar algum tipo de componente sem a necessidade de ir ao local;
- **AWS IoT Analytics –** Este componente permite a análise de dados provenientes dos sensores. Através da limpeza, transformação, enriquecimento e processamento de dados é possível identificar diversas situações como deteção de valores anómalos dos sensores ou identificação de tendências;
- **AWS Lambda –** Através do uso deste serviço é possível correr funções, denominadas de *lambda*, sem que sejam necessários servidores (*serverless*). Entre as principais funcionalidades deste componente tem-se: processamento de dados, reenvio de mensagens, entre outras. Além destas funcionalidades, uma das mais valias das funções *lambda* é o custo associado. Apenas o tempo de processamento de execução da função é pago, quando em modo *idle* não existe qualquer custo;

Através de pedidos REST o cliente poderá ter acesso a todos os dados relativos à sua fábrica. Esta exposição de dados será feita através de uma aplicação *web*, cujo acesso é garantido em qualquer lugar, independentemente do dispositivo.

De forma a melhor identificar a arquitetura, são de seguida apresentados os diagramas de componentes e de *deployment*, assim como uma breve explicação de ambos.

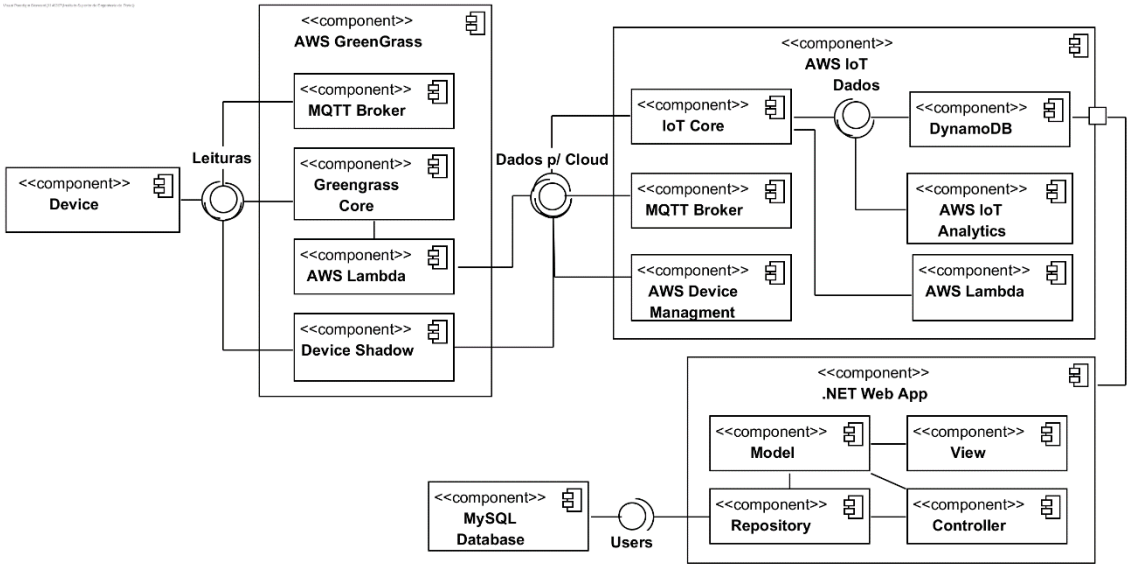


Figura 24 – Diagrama de componentes

A Figura 24, representa os diferentes componentes e a forma como a informação circula entre estes. A fonte, **Device**, envia dados para o **MQTT *Broker*** do **AWS GreenGrass**. Estes dados, são depois consumidos por diferentes componentes do *gateway*, nomeadamente pelo **GreenGrass Core** e **Device Shadow**. Depois de analisados os dados no *gateway*, podem ser enviados pela *cloud* para consulta do cliente. Encarregue desse envio estão as funções **lambda** que reenviam os dados para o **MQTT *Broker*** do **AWS IoT**. Estes dados são depois consumidos pelo **IoT core***,* semelhante ao AWS GreenGrass core. O componente **DynamoDB** é usado como base de dados para estes valores. Estes dados são também analisados através do **AWS IoT Analytics**. Esta informação pode, mais tarde, ser consumida pelo cliente através da .**NET *Web App***.

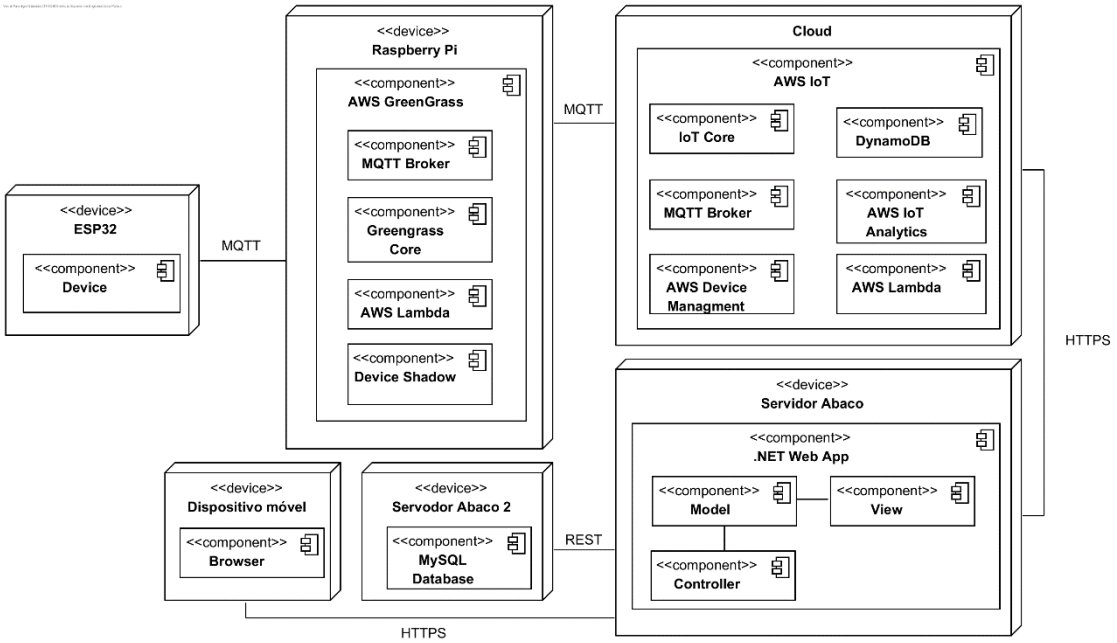


Figura 25 - Diagrama de *Deployment*

Observando a Figura 25 é possível notar como os componentes comunicam entre si e onde são implantados na fase de prova de conceito. O microcontrolador **ESP32** servirá de emissor de dados. Para receção destes e consequente emissão para a *cloud*, será usado um **Raspberry Pi**, implantado com **AWS GreenGrass**. Já os serviços de análise e armazenamento de dados estarão alojados na ***cloud***. Tanto a **.NET web app** como a sua base de dados estarão implantadas em servidores distintos pertencentes à Abaco Consultores. Para o cliente aceder a esta informação apenas precisa de possuir um dispositivo móvel que possua *browser* e ligação à rede.

A Figura 26 representa o fluxo de dados desde a sua leitura pelos sensores até à sua exposição ao cliente. Os componentes AWS são constituídos por operações não representadas no diagrama, operações essas que são todas realizadas automaticamente, sem qualquer intervenção humana.

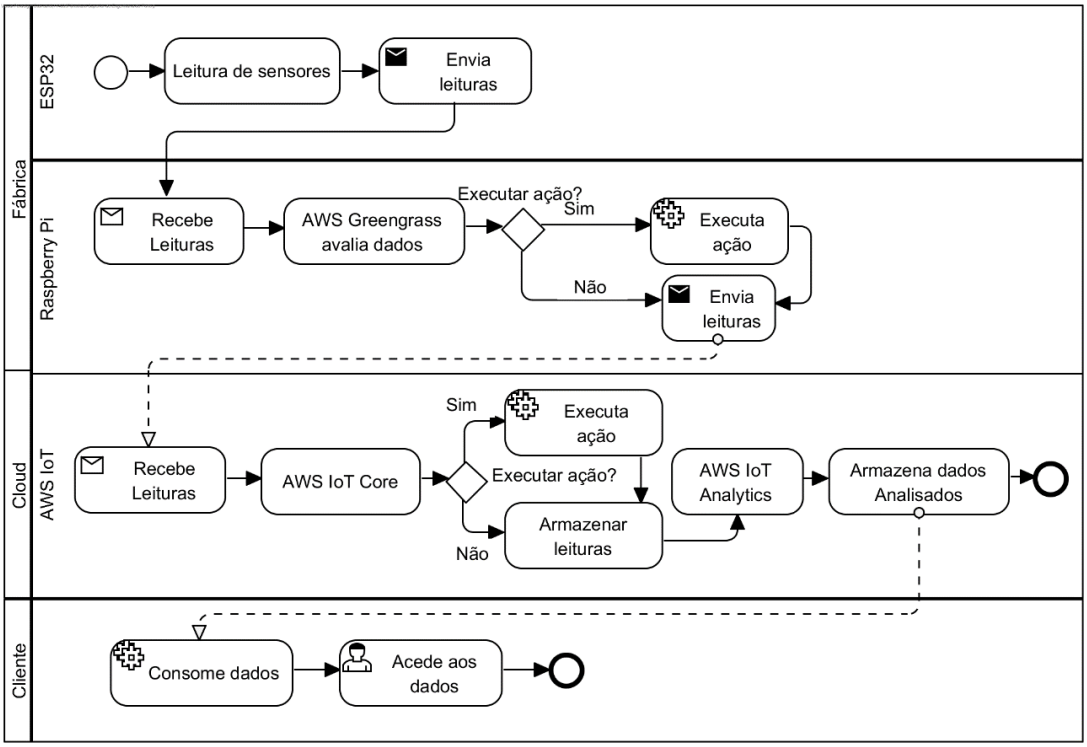


Figura 26 - Diagrama de atividades AWS

O **ESP32**, através dos seus componentes, realiza as leituras necessárias. A este, está associado um determinado identificador que será incluído no tópico da mensagem para o *gateway* (**Raspberry Pi**). Depois de feita a ligação ao tópico, o sensor publica a mensagem e fecha a ligação. A mensagem inclui o resultado das leituras assim como uma marca temporal (*timestamp*) do momento da leitura.

A Figura 27 apresenta o envio das leituras dos sensores assim como o consumo destas pelo *gateway*. Depois de definido o tópico, é publicado para o **MQTT Broker** a mensagem com QoS igual a um (Evento 3.2.1 da Figura 27), que garante a receção da mensagem. Depois de publicada, é redirecionada para os subscritores desse mesmo tópico, neste caso o **AWS GreenGrass** (Evento 3.2.3).

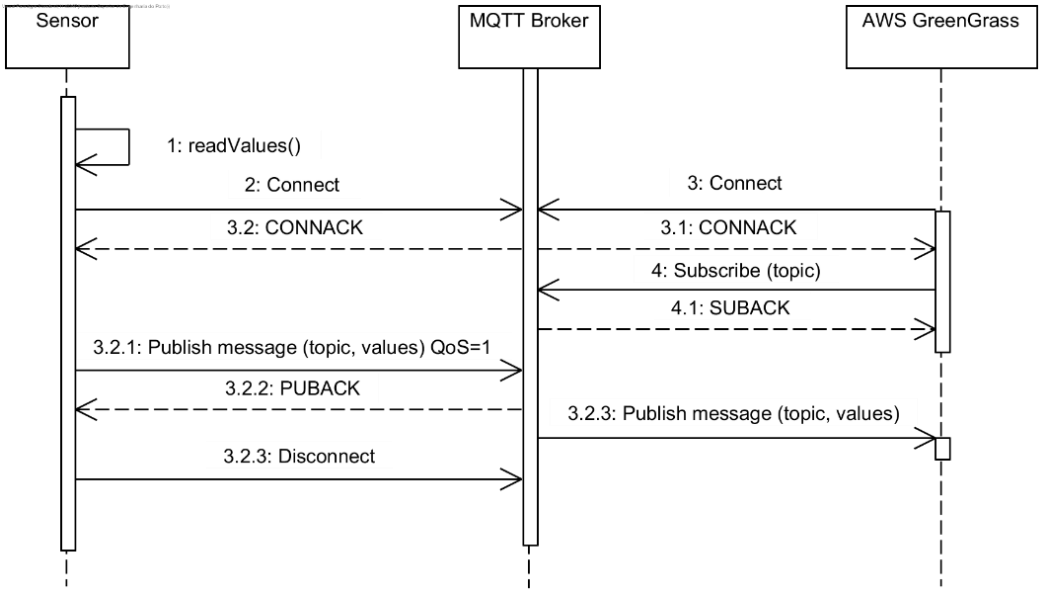


Figura 27 - Diagrama de sequência leitura e envio de valores

Na definição dos tópicos é necessário algum critério. A Figura 28 representa uma possibilidade do pode ser a constituição dos tópicos.

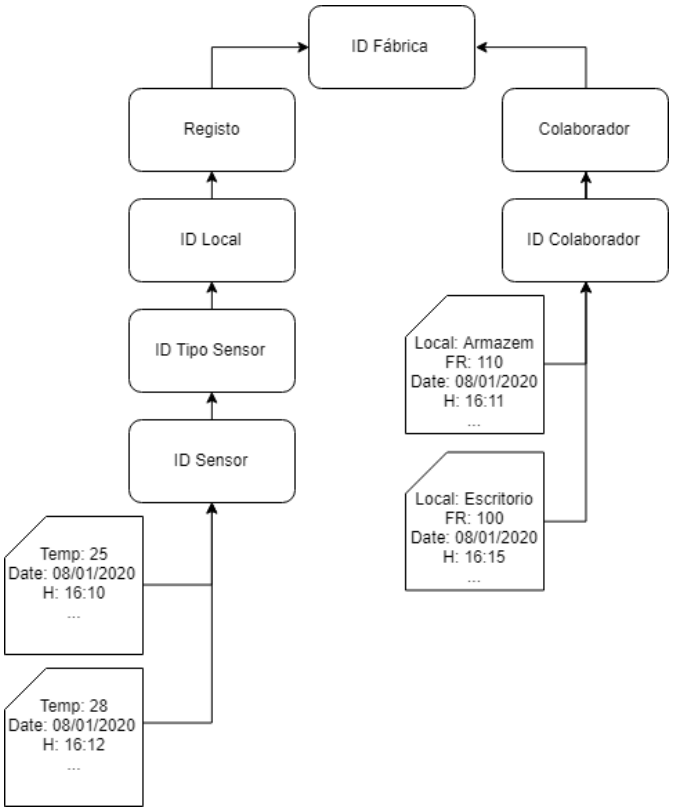


Figura 28 - Definição Tópicos MQTT

Exemplificando o processo mostrado na Figura 28 e tendo em consideração os seguintes pontos:

- **Fábrica** – Maré;
- **Locais** – Armazém, Produção e Escritórios;
- **Tipos de sensores** – Qualidade de ar, temperatura, radioatividade;
- **ID dos Sensores** – SensorIDAr, SensorIDTemp, SensorIDRad;
- Cada local possui um sensor de cada tipo;

Para este caso de uso, as mensagens relativas aos registos dos sensores seriam publicadas nos seguintes tópicos (Apenas é apresentado um exemplo para cada local):

- **Maré**/**Registo**/Armazém/QualidadeAr/SensorIDAr
- **Maré**/**Registo**/Produção/Temperatura/SensorIDTemp
- **Maré**/**Registo**/Escritório/Radioatividade/SensorIDRad

Nestes tópicos, a secção a negrito é a única que se mantém constante. Os três campos seguintes podem variar conforme o local, tipo e ID de sensor. Neste caso específico, haveriam nove tópicos possíveis de receberem mensagens. As mensagens publicadas neste tópico possuem dados como: valor lido, data e hora da leitura, unidade de leitura, entre outras possibilidades.

Para a localização dos trabalhadores, o tópico é mais simples, apresentando apenas qual o ID do trabalhador:

- **Maré**/**Trabalhador**/ID

A mensagem a ser publicada pode conter dados como: Local onde se encontra, data e hora, Frequência cardíaca, entre outros.

A receção das mensagens é da responsabilidade do *gateway*, neste caso o AWS GreenGrass. Para que as mensagens sejam recebidas é necessário que este subscreva todos estes tópicos relativos à fábrica. A forma mais eficiente para subscrever todos os tópicos relativos a uma fábrica é subscrever o seguinte tópico:

- IDFábrica/**#**

Através do uso do auxiliar '#' é possível subscrever todos os subtópicos que possuam como primeiro identificador o ID da fábrica, independentemente do número de níveis que possam ter.

Depois da receção destas mensagens é necessário analisar os dados. Esta informação, quando analisada através das funções *Lambda* poderá despoletar um conjunto de ações, a definir conforme o negócio. Entre as possíveis ações tem-se:

- Registar dados na base de dados **AWS DynamoDB**;
- Chamada de outra função *Lambda*;
- Republicar dados para outro tópico MQTT;
- Envio de notificações *push*;
- Etc.

Sendo o **AWS GreenGrass** um serviço semelhante ao **AWS IoT Core**, todas estas funções podem também ser executadas na *cloud*.

Além do tratamento dos dados lidos, o **AWS GreenGrasss**, é também responsável por manter uma réplica virtual do dispositivo, o *shadow*. Este guarda no próprio *gateway* um registo do estado do sensor assim como os valores mais recentes lidos.

Assim sendo, a Figura 29 mostra o fluxo da informação entre os componentes que estarão implantados no **Raspberry Pi**.

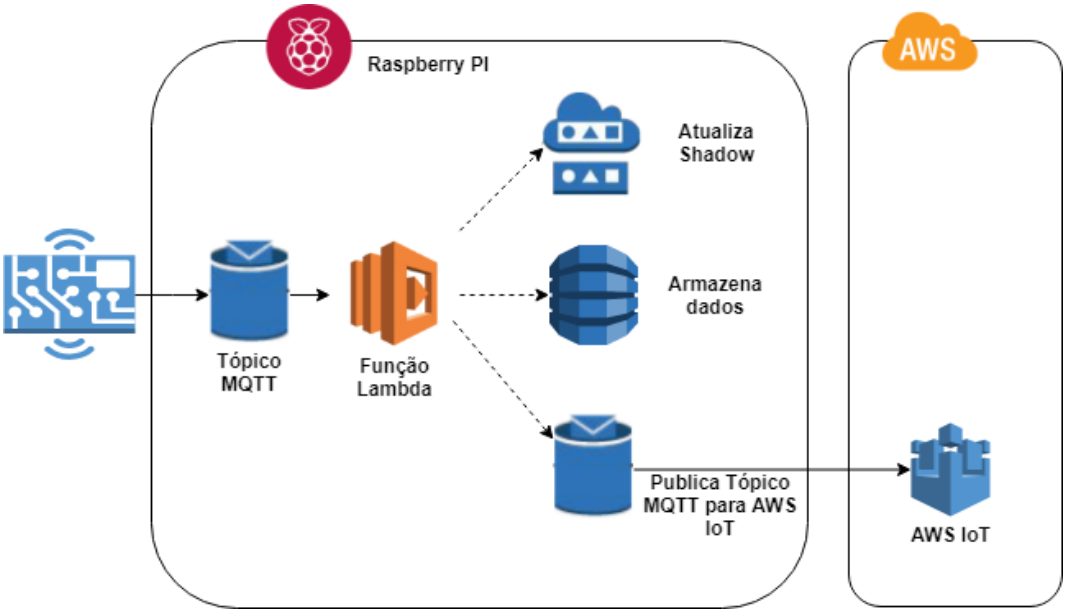


Figura 29 - Fluxo de dados Raspberry Pi

Observando a Figura 29 é possível observar as capacidades das funções **lambda**. Através da sua configuração é possível realizar diversas ações consoante a necessidade do cliente e do seu negócio.

Além desta solução foi também criada uma solução alternativa presente no Anexo G.

4.4 Base de dados

Tendo em consideração a arquitetura apresentada, existem diversas possibilidades para armazenamento de dados. O AWS fornece opções diferentes, deixando ao programador a escolha entre o uso de bases de dados *NoSQL* ou o uso do serviço especializado e *serverless* como DynamoDB e Amazon S3:

- **Amazon DynamoDB –** Facilmente integrável, permite armazenar e gerir dados. Acarreta um custo por leitura/Escrita. É uma base de dados de elevada performance que garante a segurança e replicação de dados, evitando perdas ou acessos não permitidos a estes;

- **Amazon S3 –** Este é um serviço de armazenamento de dados que oferece escalabilidade, elevada disponibilidade, segurança e *performance*. Este serviço apenas permite guardar dados em ficheiros. Apresenta-se como a solução mais acessível apesar da limitação;
- **Cassandra –** Utilizando o serviço *Amazon EC2 (Elastic Compute Cloud*) é possível utilizar a cassandra como base de dados principal. Este serviço oferece segurança e capacidade de expansão automática conciliada com a grande capacidade de armazenamento de dados, já demonstrado, do Cassandra. Como ponto negativo para o uso desta tecnologia é o preço associado ao uso de EC2. É necessário comprar um servidor que, mesmo em *idle*, acarreta um custo. Posto isto, o uso de cassandra torna-se mais complicado e só é usado caso exista uma necessidade especifica para tal.

Na solução apresentada foi decidido usar a base de dados inerente ao AWS**, DynamoDB**. Esta escolha deve-se à fácil integração no sistema, o seu custo reduzido e pela sua capacidade de armazenamento e escalabilidade.

4.5 Organização de Sensores e Dados

Durante a execução deste projeto, foi considerada a possibilidade do uso de uma quantidade de sensores significativa. Este facto faz com que seja necessária uma atenção à forma como é feita a organização destes. O **AWS IoT Core** oferece a possibilidade de criação de grupos de dispositivos. Cada grupo pode possuir até 200 dispositivos e pode ser composto por outros grupos (hierarquia máxima de 10 grupos).

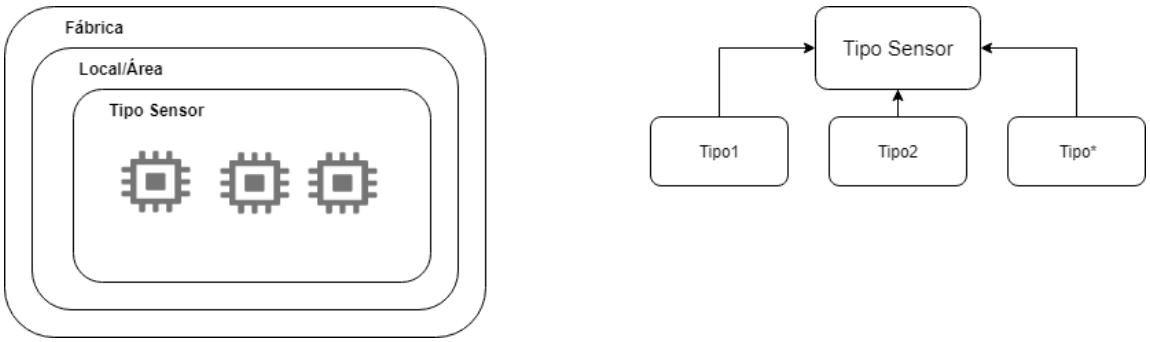


Figura 30 - Definição de Grupos de Dispositivos

A Figura 30 mostra a forma como se procede para a definição de grupos. O grupo com o identificador da fábrica é definido. Este, será composto por diversos grupos que especificam uma área geográfica da planta da própria Fábrica. Já esta será composta por diferentes grupos de dispositivos, consoante a necessidade da área. Através desta composição torna-se mais fácil identificar onde estão implantados os dispositivos e, caso seja necessário realizar algum tipo de ação sobre estes, evitam-se intervenções sobre dispositivos errados.

O estado dos sensores na *cloud* será atualizado sempre que exista uma mudança neste, através do uso do *device shadow* do próprio **AWS GreenGrass**. A informação é também atualizada no serviço *shadow* do **AWS IoT Core**, ficando esta informação disponível para consumo do cliente.

Para isso, o sensor, publica num tópico específico, o seu estado e outras informações necessárias (Ex: Estado da bateria, caso se aplique) ficando então registado o estado *reportado*. O utilizador, tendo acesso a estes dados, caso queira, pode efetuar alterações (ex: desativar um sensor). Para isso irá preencher um campo denominado de estado **desejado**. O sensor, ao receber este campo, irá efetuar as mudanças necessárias para que o **reportado** se assimile ao **desejado**.

O estado relativo ao *gateway* é também guardado na *cloud* através do serviço de *shadow* oferecido pelo **AWS IoT Core**. O processo é semelhante ao já descrito para os sensores.

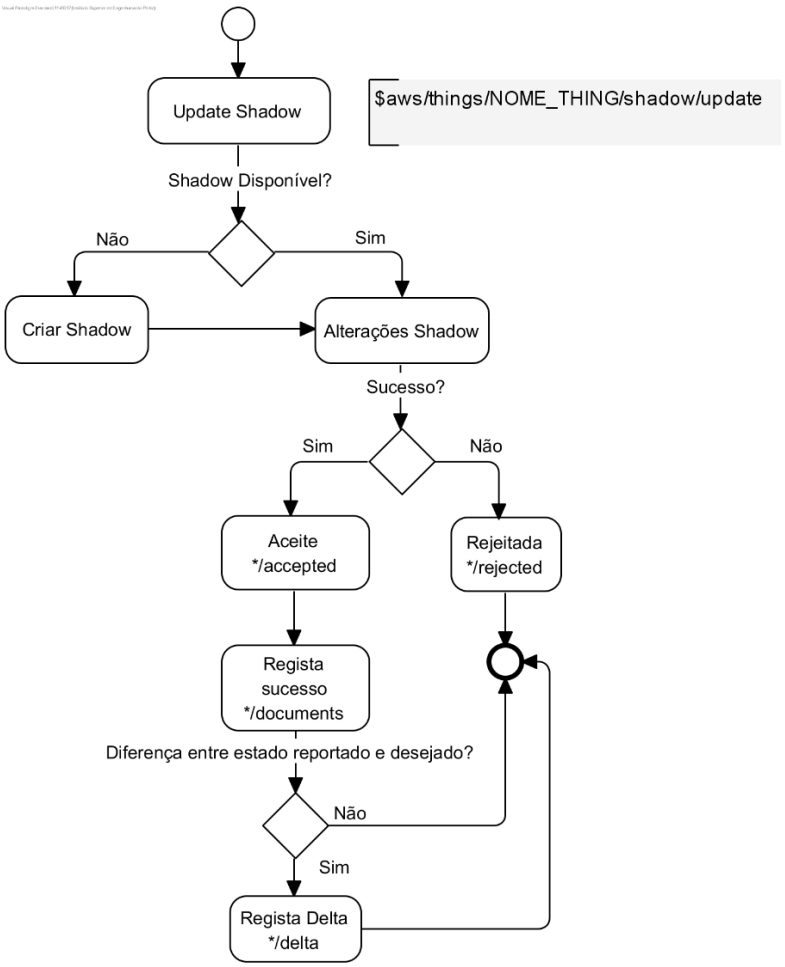


Figura 31 - Diagrama de atividades *Update Shadow*

A Figura 31 representa a sequência de atividades necessárias aquando de uma atualização do *shadow* de um certo dispositivo. Para cada sensor, existe um tópico específico representado na figura e que assume a seguinte composição:

- $aws/things/**Nome_Thing**/shadow/update;

O campo a negrito é especificado com o nome dado ao sensor na plataforma da AWS. Depois de requisitado um novo *update* ao *shadow* é verificado se existe sucesso ou não. Para ambos os casos, existe o envio de mensagens para o tópico respetivo, adicionando ao tópico original

/accepted ou **/rejected**. Se a alteração for aceite, é então publicado no tópico **/delta** os valores diferenciadores entre o estado reportado e o estado desejado que terá como função uniformizar ambos.

A segurança é também um fator importante aquando do desenvolvimento de uma solução IoT. A solução oferecida pele *Amazon* garante a segurança dos dados através da encriptação TLS. Relativamente aos dispositivos, cada um destes apresenta um certificado próprio, gerado no momento do seu registo. Este certificado, permite, no momento da ligação do dispositivo à rede, a sua autenticação. Para efeitos de autorização, é usado a Política do grupo onde o dispositivo se encontra. Aquando da criação de um novo grupo de dispositivos, é associado a este uma nova política que pode ser usada para autorização de dispositivos. Sendo assim, a autorização e autenticação dos dispositivos é representada pela Figura 32.

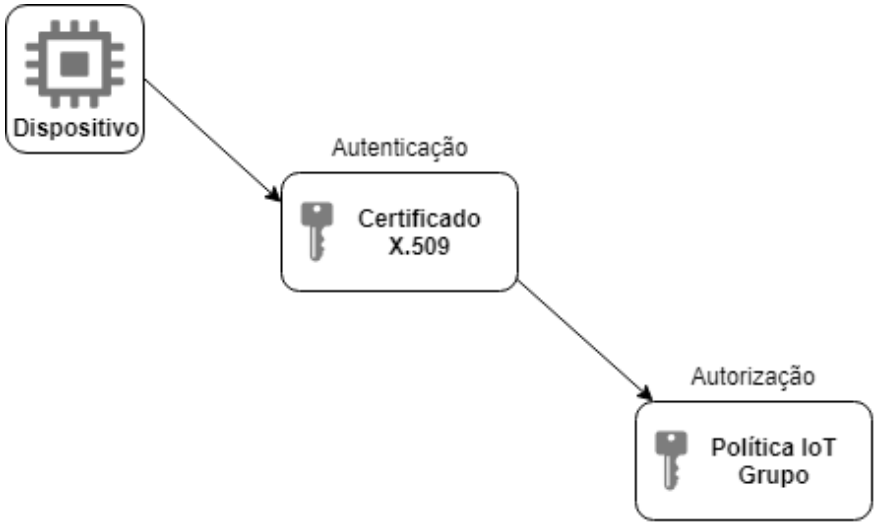


Figura 32 - Autenticação e Autorização

4.6 Web app

A solução, da maneira apresentada usa apenas serviços disponibilizados pela Amazon. Para aumentar o valor da solução é necessário criar uma interface ao cliente final que responda às suas necessidades. Para isso é necessário criar uma *web app*. Sendo o seu foco o uso em computadores e *browsers* é necessário também ter em consideração possuir compatibilidade para uso em dispositivos móveis.

Os utilizadores de sistemas serão divididos em três grupos de utilizador:

- **SuperAdmin** - Administrador de sistema. Acede a todas as páginas e pode efetuar qualquer operação;
- **CompanyAdmin** – Administrador de empresa. Acede apenas aos dados associados à sua empresa. Pode efetuar qualquer operação sobre os dados da sua empresa;

- **CompanyUser** – Utilizador com menos privilégios. Apenas lhe é permitido aceder a dados das instalações associadas a si.

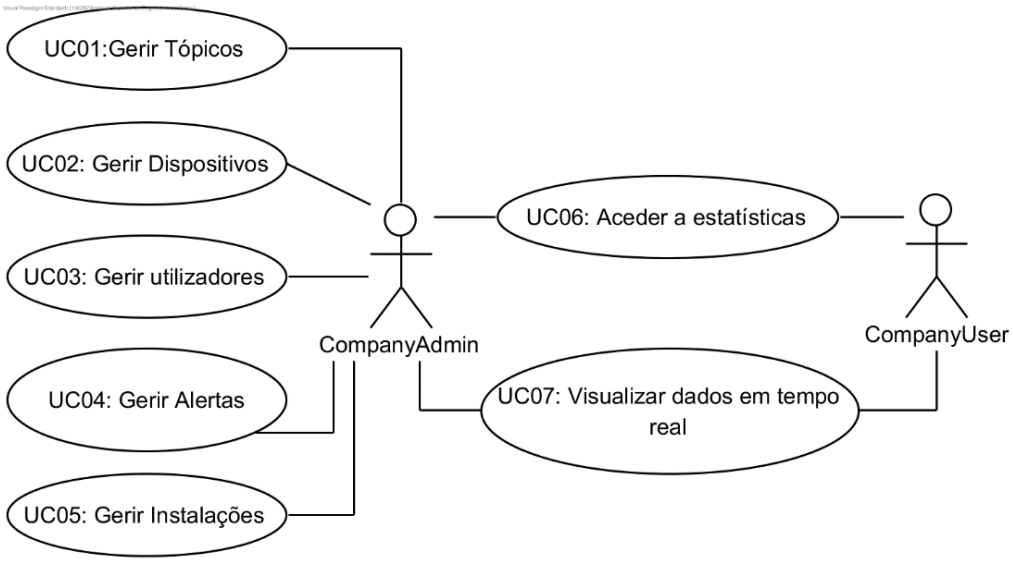


Figura 33 - Diagrama de casos de uso Cliente

A Figura 33 representa quais os casos de uso possíveis para os dois tipos de cliente final do sistema. Através de uma plataforma customizável à sua empresa, o cliente (**companyUser** ou **companyAdmin**) terá de efetuar *login* para ter acesso aos dados das suas instalações. Os dados, em tempo real, poderão ser apresentados através de tabelas, gráficos, imagens, entre outros, conforme a necessidade e vontade do cliente. Além destes, o utilizador terá também acesso a dados estatísticos. A plataforma é também usada para monitorização e gestão dos próprios sensores. Caso o utilizador necessite, por motivos financeiros ou outros, pode ativar/desativar sensores de forma remota. O utilizador pode também definir métricas para os valores obtidos nos sensores.

A Figura 34 apresenta os casos de uso do administrador do sistema. Este, tem acesso a todo o sistema e é responsável por preparar o mesmo para um cliente. Além das funcionalidades em conjunto com o cliente, o administrador tem também o papel de fazer a gestão de utilizadores e empresas.

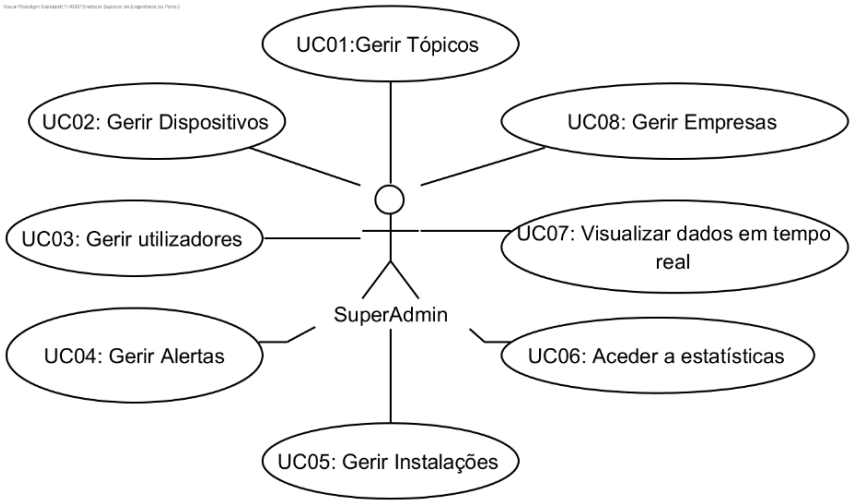


Figura 34 - Diagrama de casos de uso Administrador

Este serviço fica alojado nos servidos da Abaco e será desenvolvido em *.Net*, seguindo o caminho dos projetos já existentes de forma a poder ter continuidade e suporte por parte de outros trabalhadores.

4.7 Requisitos Não Funcionais

Como requisitos não funcionais para o sistema tem-se:

- **Portabilidade –** Visto que o cliente apenas terá acesso à *web app* é necessário garantir o seu funcionamento em qualquer dispositivo independentemente da sua capacidade ou do *browser* que estiver a ser usado;
- **Fiabilidade –** O sistema deve apresentar sempre valores reais e corretos. Só assim poderá ser possível obter a confiança dos utilizadores. Para isso, é necessária uma atenção redobrada na escolha de *hardware*, nomeadamente sensores, que serão responsáveis por fazer as leituras;
- **Segurança –** A segurança é um fator decisivo na implementação da solução. Sendo um sistema que estará em funcionamento 24 por dia, 365 dias por ano, é necessário preparar o mesmo para evitar qualquer tipo de ataque;
- **Disponibilidade –** O serviço deve estar disponível a qualquer altura. Posto isto, é expectável que o sistema possua *uptime*[5] superior a 98% e um *recovery time objetive* (RTO) inferior a uma hora;
- **Facilidade de uso –** Pretende-se que o sistema seja *user-friendly, responsive*[6] e intuitivo;
- **Requisitos Legais** (Confidencialidade de dados);
- **Definição de idioma –** A *web app* deve estar disponível em diversos idiomas.

[5] Tempo de atividade de um sistema
[6] *Web app* adapta o *layout* a qualquer dispositivo automaticamente;

O acesso à *web app* necessitará de credenciais que serão, numa primeira fase, concedidas pela Abaco e que poderão, mais tarde, ser alteradas.

4.8 Modelo de dados

Figura 35 - Modelo de Dados *Web app*

A Figura 35 representa o modelo de dados encontrado para responder a todas as necessidades do sistema. De forma a melhor entender o diagrama, é de seguida apresentado cada modelo:

- ***Site –*** (Instalação) Representa um local físico de uma empresa especifica. Caracterizado por um identificador único, nome, descrição e um identificador da instalação no sistema *Kafka*;

- ***Company*** **–** (Empresa) Representa um cliente do sistema. Apenas os *superadmins* podem criar instâncias deste modelo. Caracterizado por um identificador, nome, email, *website* e código;
- ***ASPNetUser*** **–** Utilizador do sistema. Está associado a uma empresa através do identificador desta. É caraterizado pelo seu ID, *username*, nome, *token e* identificador do utilizador do sistema *kafka*. Cada utilizador pode estar associado a diversas instalações. Sendo uma relação de muitos para muitos é criado o modelo ***ASPNetUserSites*****;**
- **URL –** Modelo usado para armazenar URL's dos serviços a usar pela *web app*. Caracterizado pelo nome, *link* e empresa a que queres associar esse mesmo *link*;
- ***Topic*** **–** (Tópico) Modelo referente ao sistema *Kafka*. Representado por um ID, nome, descrição e identificador de instalação;
- ***DeviceType*** **–** (Tipo de Dispositivo) Modelo de caracterização de dispositivo. Possui um identificador e nome (Ex: ESP32, ESP8266, …);
- ***Device*** **–** Dispositivo de emissão de leituras. Possui um identificador próprio, o identificador do seu tipo (*DeviceType*), local, identificador de instalação e tópico associados a este;
- **Sensor –** Cada dispositivo pode possuir *n* sensores. Cada um é caracterizado pelo seu ID, descrição e unidade de leitura;
- **Alert –** Os utilizadores podem criar alertas para cada sensor de um dispositivo específico. Este alerta é caracterizado por um ID, descrição, valor mínimo e máximo, data de início e data de fim;
- ***TriggeredAlert*** **-** (Alerta Ativo) Representa um alkerta ativo para um determinado sensor. Possibilita a amostra do mesmo na *web app* de forma dinâmica. Caracterizador por um identificador, uma data de início e uma data final sendo que esta pode ser *null* até que os valores voltem a ser dentro dos limites;
- ***Action*** **–** Modelo que identifica as possibilidades de ações aquando de um registo de um valor anómalo. Apenas os *superadmins* conseguem efetuar ações sobre este modelo;
- ***ActionToAlert*** **–** Modelo intermédio que identifica o que cada alerta deve acionar quando ativado.

Sendo este o componente que estará em contacto direto com os clientes é necessário garantir máxima eficácia e eficiência na sua execução. A personalização será também importante e irá melhorar a relação entre os clientes e a empresa.

4.9 Custos

De forma a conseguir responder à questão **Q7**, depois de desenvolvido a arquitetura é possível ter uma perceção mais real daquilo que será o custo associado à execução do sistema.

Na simulação, a seguir apresentada, tem-se em consideração os seguintes fatores:

- Os sensores ligam-se todos ao mesmo *Gateway* sendo que este processa os dados localmente, enviando-os de dois em dois minutos para a *cloud*;
- 1000 dispositivos com ligação 24/7 durante 12 Meses;
- 10 AWS GreenGrass *Cores* (100 dispositivos por *Core*);
- Cada dispositivo atualiza o *shadow* uma vez por dia (sensores e *gateways*);
- O servidor AWS usado situa-se em Frankfurt, Alemanha;

A Tabela 10 representa os valores dos serviços disponíveis no AWS (Preços consultados a 03/01/2019).

Tabela 10 - Registo de Preços em Dólares Americanos

Preços Frankfurt	
AWS GreenGrass	0,220
Cores	10
AWS IoT Core	
Ligação	0,096
Mensagens	1,200
Shadow	1,500
Rules	0,180
AWS Device Managment	
Registo	0,120
Indexação	2,700
Updates	0,035
AWS IoT Analytics	
Data Processing	0,200
Data Storage	0,030
Queries	6,500
Análise customizada	0,360

O componente AWS Lambda não é registado devido ao seu nível *free tier*. Para todos os clientes, o AWS oferece, por mês, o primeiro milhão de solicitações assim como 400.000 GB-Segundo de tempo de computação.

A Tabela 11 apresenta os custos relativos ao serviço de *gateway* e *IoT Core.* O cálculo relativo ao AWS GreenGrass é feito através do número de *Cores* que interage com a *cloud* por mês, assumindo um valor, ao final de 12 meses de 23,2€. Relativamente ao componente *IoT Core,* o seu custo é definido pelo número de ações sendo que os valores aqui apresentados podem variar significativamente. Neste caso, apenas os *gateways* comunicam com o *IoT Core*. Os 10 *cores* comunicam de dois em dois minutos com a *cloud* enviando mensagens que não ultrapassam os 5KB. Assim, ficamos com um valor de cerca de 3,9€ para este serviço.

Tabela 11 - Custo AWS GreenGrass e IoT Core

AWS Anual	
AWS GreenGrass	23,232
AWS IoT Core	
Ligação	0,252
Mensagens	3,154
Shadow	0,553
Rules Triggered	0,237
Actions Executed	0,237
Total $	4,432
Total €	3,900

O serviço AWS IoT Analytics (Tabela 12) apresenta um valor final de 25,16€ através dos seus 4 componentes: processamento de dados, armazenamento, *queries* de pesquisa e Análise de dados. Neste caso assumimos que cada dispositivo envia 10MB de dados por mês (10.000MB por mês). Sendo o preço calculado através da quantidade de informação em GB é necessário transformar a quantidade total em GB, ficando assim com cerca de 9.8 GB de informação por mês. É sobre este valor que são depois calculados os valores para os quatro diferentes processos.

Tabela 12 - Custo AWS IoT Analytics

AWS IoT Analytics	
Data processing	23,437
Data Storage	3,516
Queries	0,744
Análise	0,900
Total $	28,597
Total €	25,165

A Tabela 13 representa os valores relativamente ao registo, indexação e *updates* de dispositivos. Para o primeiro campo, são cobrados cerca de 0,10€ por cada 1000 registos. Tendo esta simulação 1000 dispositivos, ficamos com o valor final igual. Os valores seguintes são apenas opcionais e são relativos a indexações e *updates* que seja necessário serem realizados. No caso dos *updates*, caso fosse necessário realizar esta ação a todos os dispositivos teríamos um valor final de 30,8€.

Tendo estes dados em consideração para a simulação em questão, e sem valores relativos à base de dados, este sistema teria um custo anual de aproximadamente 58,40€ sem *updates* realizados.

Tabela 13 - Custo AWS Device Managment

AWS Device Managment		
	Dólar	**Euro**
Registo	0,120	0,106
Indexação	8,100	7,128
Updates	35,00	30,800

Tabela 14 - Custo AWS DynamoDB

A Tabela 14 representa os custos associados ao uso de uma base de dados AWS DynamoDB.

Base de Dados AWS DynamoDB		
Escrita	1,525	/1000000
Leitura	0,305	/1000000
Armazenamento	0,306	>25GB Por mês
Backup Continuo	0,245	/GB
Backup por Request	0,122	/GB
Restauro de tabela	0,184	/GB

A não inclusão dos serviços *lambda* é importante para esta secção. A quantidade de execuções varia bastante conforme os tipos de negócios e é difícil de fazer uma previsão exata do valor que pode vir a assumir. Por outro lado, tendo um número relativamente alto de execução grátis, assume-se que só começará a ser cobrada quando existir um número elevado de dispositivos a comunicar com a *cloud*.

No Anexo A encontra-se uma figura ilustrativa do ficheiro criado para execução da simulação aqui mencionada.

4.10 Implementação

Definida toda a arquitetura necessária e os detalhes dos componentes é necessário a implementação destes. De forma a melhor demonstrar alguns detalhes deste processo encontra-se no Anexo H um resumo do trabalhado tido em conta. Neste, encontram-se dados sobre todos os componentes do sistema de forma sequencial começando pelos **sensores** (Anexo H.1), passando para o ***gateway*** (Anexo H.2), **AWS IoT** (Anexo H.3), **AWS DynamoDB** (Anexo H.4), **AWS API Gateway** (Anexo H.4) e por último, ***web app*** (Anexo H.5 até H.10).

4.11 Sumário

O capítulo de desenvolvimento da solução teve como principal objetivo preparar todo o sistema e arquitetura a desenvolver. Com o trabalho, neste Capítulo, desenvolvido foi possível responder de forma completa à questão **Q6**, ficando definido o uso dos serviços AWS nos diferentes componentes. Assim, a arquitetura fica dividida em quatro partes: ESP32 responsáveis por emitir dados, *gateway* que usa o componente AWS GreenGrass, responsável por fazer de intermediário entre dispositivos e *cloud* e finalmente o AWS IoT presente na *cloud*. Este último inclui também os componentes responsáveis por armazenamento e análise de dados. O último componente da arquitetura é a *web app* que permite a gestão do sistema em si. Definida a arquitetura foi então desenvolvido o sistema. Este desenvolvimento permitiu complementar ou consolidar algumas das respostas às questões. Entre estas pode-se evidenciar a questão **Q2.** Através dos testes e simulações realizadas foi possível perceber de que forma os dados podem e devem ser enviados entre dispositivos e sensores. Estes testes acabaram por ser decisivos melhorando a forma como estes dados seriam enviados. Além desta, através da análise das funcionalidades do AWS IoT é possível consolidar a resposta à questão **Q5** referente à notificação dos gestores. Através da definição de regras é possível notificar qualquer sujeito via *email* com tempos de resposta bastante baixos. Esta resposta é testada no Capítulo de Avaliação que se segue. Por fim, este Capítulo, responde também à questão **Q7** relativa aos custos. Tendo a arquitetura definida é possível simular quais os custos associados. Porém, a resposta a esta questão não pode ser definida como certa ou errada pois existe uma serie de fatores que podem influenciar o mesmo.

Contudo, o objetivo principal deste capítulo seria cobrir todo o processo desde o levantamento de requisitos até à implementação do sistema em si. Através das diversas respostas atingidas foi possível desenvolver um sistema eficaz e fluido.

5 Experimentação e Avaliação

No capítulo de experimentação e avaliação é apresentada uma descrição das experiências e formas de avaliação submetidas à solução final. Entre estas podem-se incluir: grandezas para avaliação de trabalho (tempo, satisfação do utilizador, disponibilidade, etc.); metodologias de avaliação (Grupos de testes/controlo, etc.) e Tipos de testes (Estatísticos, etc.).

5.1 Experimentação

De modo a testar a viabilidade da solução no mundo real, foram implantados sensores e um *gateway* numa fábrica cliente (simulada internamente na empresa Abaco) e a partir daí, aperfeiçoado o modelo de negócio. Esta implantação tinha como objetivo aperfeiçoar tanto a gestão dos dados como o próprio *frontend* da *web app*.

A simulação foi efetuada apenas com um dispositivo emissor de dados (**ESP32**) sendo que estes dados são simulados. O dispositivo comunica com o serviço ***gateway*** emitindo dados variáveis que mais tarde são enviados para o **AWS IoT** de forma a serem avaliados e armazenados. O sistema é representado na Figura 36.

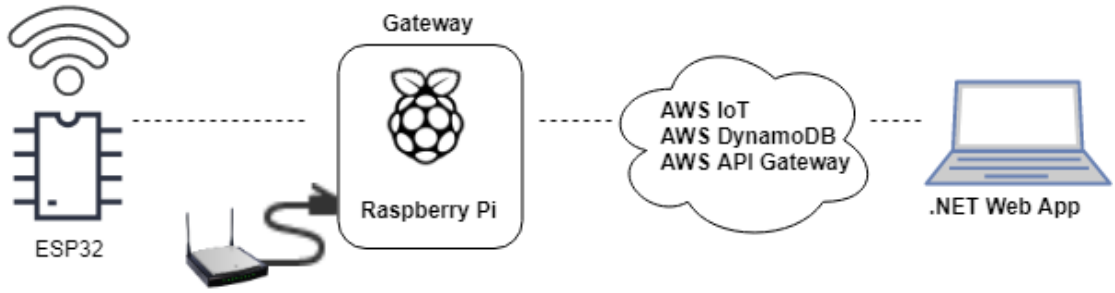


Figura 36 - Esquema de Simulação

De forma a validar o sucesso ou não do sistema, na Tabela 15 estão definidas as métricas tidas em consideração durante a execução da simulação assim como os resultados esperados. Cada ponto da tabela possuiu uma carga de testes diferentes e em condições diferentes.

Tabela 15 – Definição de métricas

Métrica	Resultado esperado
1. O *gateway* responde a valores anómalos?	Resposta entre 2 a 4 segundos.
2. Os dados são enviados para a *cloud* nos tempos definidos para o utilizador.	Validar
3. Não existe perda de dados quando enviados para a *cloud.*	Validar
4. O sistema notifica o utilizador/utilizadores em menos de 15 segundos desde o acontecimento.	Validar
5. O *gateway* envia valores guardados quando repõe ligação à rede.	Validar
6. O cliente tem acesso aos dados atuais na *web app.*	Validar
7. A *web app* responde às necessidades dos clientes e apresenta os valores atuais.	Validar

5.1.1 Métrica 1

De forma a responder a este requisito foram realizados testes consecutivos em diferentes períodos diários para averiguar a capacidade de resposta do *gateway*. No total, foram efetuados 50 testes que comprovam a eficiência do dispositivo na resposta de leituras dos valores. Os testes foram realizados nos intervalos de maior consumo de rede (11:00h – 14:00h/ 16:00h – 18:00h) sendo realizados em conjuntos de 10 por hora. Desta forma é possível testar também a reação do sistema a possíveis falhas de ligação.

Os testes foram realizados com as seguintes condições:

- Um dispositivo (ESP32) ligado por *Wifi* através do protocolo 802.11 b/g/n;
- Raspberry Pi 2 a servir de dispositivo *gateway* com ligação por cabo à rede;
- Os dados anómalos são enviados para o tópico **Abaco/Data/A01** que reenvia para o ESP32 de forma a estes os replicar**;**
- O dispositivo emite uma mensagem MQTT com os dados recebidos. Este passo marca o início do registo temporal;
- O *gateway* analisa os dados e emite mensagem de erro para a *cloud;*
- A mensagem é recebida no AWS IoT Core*.* Paragem do registo temporal.

Inicialmente, depois de uma breve reunião com os responsáveis, foi dado um período de dois a quatro segundos como aceitável para resposta à deteção de um valor anómalo. Após a realização dos testes averiguou-se que estes valores dificilmente iriam ser atingidos. Dos 50

testes realizados, 48 obtiveram resposta com tempo inferior a um segundo e dois com resposta inferior a dois segundos. Os resultados a este teste podem ser encontrados nos Anexo E. Através dos resultados obtidos neste teste é possível considerar o mesmo como válido e aceite.

5.1.2 Métrica 2

O teste de ligação entre o dispositivo *gateway* (Raspberry Pi 2) e a *cloud* (AWS IoT) revelou alguns resultados interessantes e que acabaram por ter um papel importante em algumas decisões. Estes testes foram realizados com as seguintes condições:

- Dispositivo ESP32 a emitir dados para o *gateway;*
- *Gateway* armazena os dados através dos lambdas já definidos, emitindo-os após X segundos para a *cloud;*
- É considerado o tempo de receção na *cloud* como fator de decisão.

Assumindo estas condições, foram realizados testes para quatro intervalos de tempo diferentes. Para cada intervalo foram realizados 10 testes, no período entre as 12:00h e as 13:00h:

- **Intervalo de 60 segundos:**

Tabela 16 - Teste ligação *gateway-cloud* 60s

Teste	Tempo	Resultado	%
1	60	61	101,67
2	60	60	100,00
3	60	60	100,00
4	60	43	71,67
5	60	60	100,00
6	60	42	70,00
7	60	60	100,00
8	60	41	68,33
9	60	60	100,00
10	60	60	100,00
		54,7	-8,83

A Tabela 16 apresenta os resultados dos 10 testes realizados com um intervalo de 60 segundos entre envio de dados. Estes podem ser representados através do gráfico representado na Figura 37.

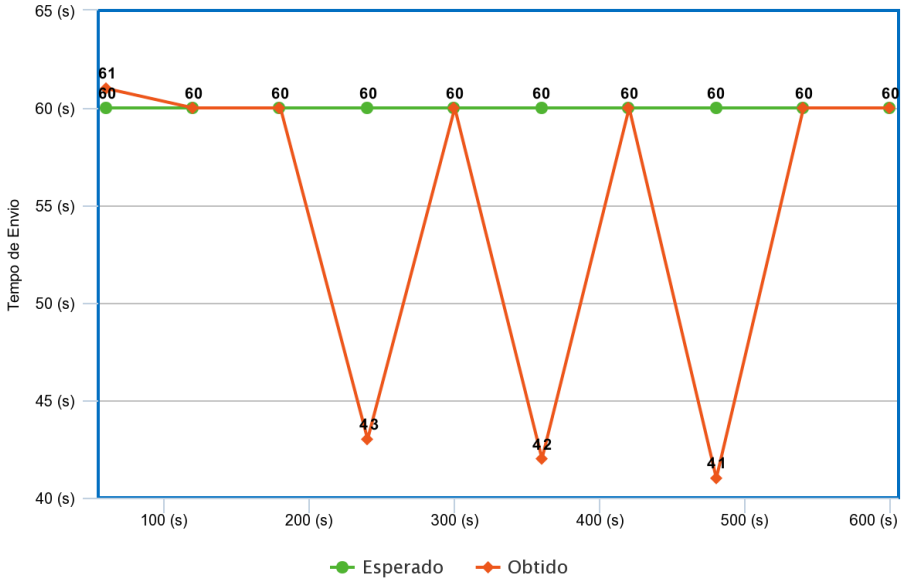


Figura 37 - Gráfico Linhas - Teste ligação *gateway-cloud* 60s

Analisando os dados é possível notar que a média de tempos acabou por ser mais baixa que o esperado apresentando um valor de **54,7** segundos que representa um valor aproximadamente **8.8%** inferior ao esperado. Este foi o primeiro teste a ser realizado e os dados mostraram-se algo curiosos o que criou a necessidade de efetuar testes com diferentes valores à procura de uma percentagem de diferença menor;

- **Intervalo de 90 segundos:**

Tabela 17 - Teste ligação *gateway-cloud* 90s

Teste	Tempo	Resultado	%
1	90	95	105,56
2	90	101	112,22
3	90	101	112,22
4	90	88	97,78
5	90	101	112,22
6	90	87	96,67
7	90	102	113,33
8	90	100	111,11
9	90	100	111,11
10	90	101	112,22
		97,6	8,44

Assumindo um intervalo de 90 segundos entre envio de dados acabou-se por verificar um efeito contrário ao teste anterior. Estes dados estão representados graficamente na Figura 38.

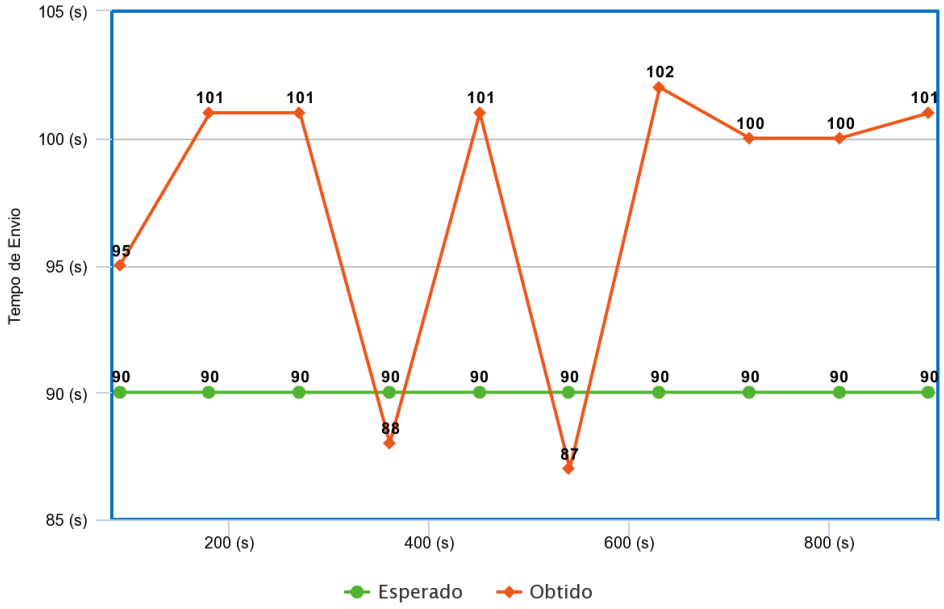


Figura 38 - Gráfico Linhas - Teste ligação *gateway-cloud* 90s

Através dos dados da Tabela 17 é possível verificar que a média de intervalos foi cerca de **8.4%** a mais que o valor esperado, que significa numa média de **97,6** segundos entre dados. Esta sequência de valores revelou-se bastante incoerente e assumiu uma percentagem de desvio relativamente alta para aquilo que seria expectável. Devido a este facto foi necessário proceder a outros testes semelhantes à procura de um desvio menor.

- **Intervalo de 120 segundos:**

Tabela 18 - Teste ligação *gateway-cloud* 120s

Teste	Tempo	Resultado	%
1	120	101	84,17
2	120	102	85,00
3	120	102	85,00
4	120	101	84,17
5	120	100	83,33
6	120	101	84,17
7	120	101	84,17
8	120	101	84,17
9	120	102	85,00
10	120	101	84,17
		101,2	-15,67

O teste com 120 segundos entre envio de dados teve resultados algo curiosos como é possível verificar na Figura 39.

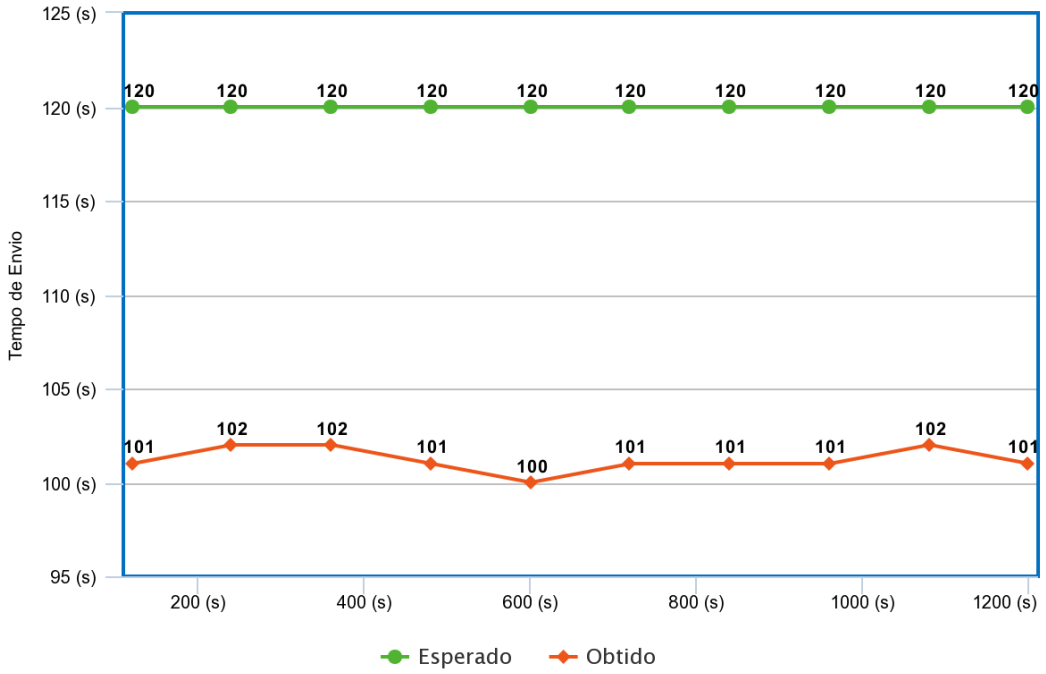


Figura 39 - Gráfico Linhas - Teste ligação *gateway-cloud* 120s

Analisando os dados da Tabela 18, é possível averiguar que todos os testes realizados acabaram por ter um valor próximo dos 100 segundos. Este resultado acabou por se resumir num valor cerca de **15%** mais baixo do que o esperado. Inicialmente este seria o último teste a realizar nesta métrica, porém, devido aos resultados demonstrados, foi necessário realizar um teste equivalente com o valor neste obtido.

- **Intervalo de 100 segundos:**

Tabela 19 - Teste ligação *gateway-cloud* 100s

Teste	Tempo	Resultado	%
1	100	101	101
2	100	101	101
3	100	100	100
4	100	101	101
5	100	102	102
6	100	101	101
7	100	102	102
8	100	100	100
9	100	102	102
10	100	101	101
		101,1	1,1

Efetuado o teste com 100 segundos de intervalo, o resultado foi o esperado tendo em conta os testes anteriores. Neste caso os intervalos de dados correspondem ao esperado havendo

apenas uma margem de **1.1%** de diferença. Estes valores são representados na Figura 40 (A escala usada para este gráfico difere da escala usada para os gráficos anteriores).

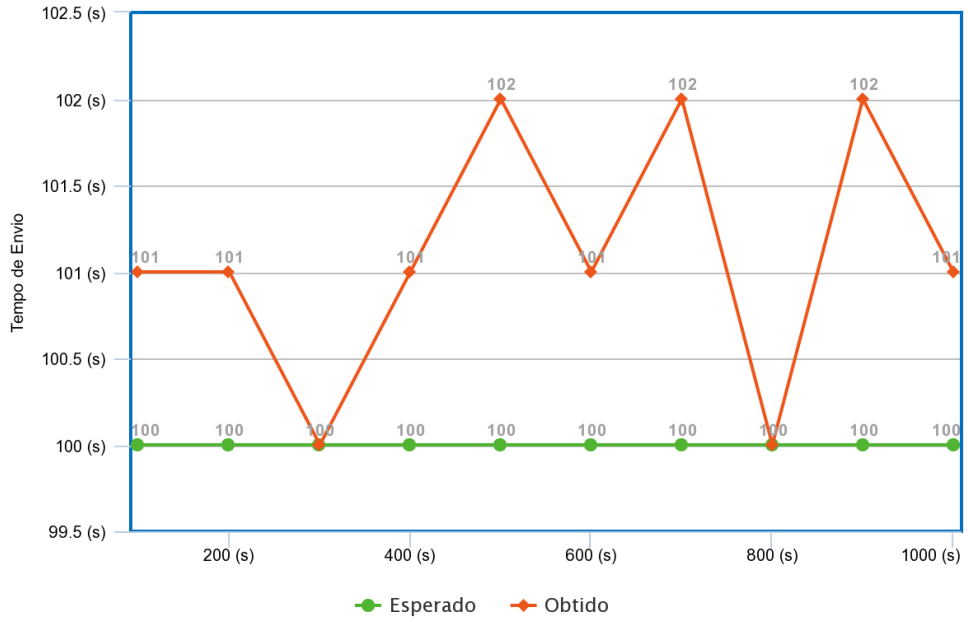


Figura 40 - Gráfico Linhas - Teste ligação *gateway-cloud* 100s

Analisados os dados no geral verificou-se que poderia haver algum problema no desenvolvimento da função **UploadFromSQLite** (Anexo H.2). Após algum estudo foi considerado que esta variação de dados pode estar ligada à função *sleep* da biblioteca *time*. Através da sua documentação é possível notar que podem existir diferenças tanto positivas como negativas com o valor esperado. Os valores de suspensão podem ser menores que o esperado quando é emitido um sinal que termina a função *sleep* ou maiores que o esperado quando existe o agendamento de qualquer outra atividade no sistema [61].

O valor de envio de dados, inicialmente, estaria preparado para qualquer valor que o cliente necessitasse. Porém, depois de analisados os resultados destes testes foi decidido adiar este caso de uso, apesar de já implementado e pronto a funcionar, até que seja possível tornar este processo mais consistente. Além disto, inicialmente, o valor por omissão para este envio seria de 120 segundos entre mensagens, este valor foi mais tarde alterado para 100 de forma a ser mais constante.

5.1.3 Métrica 3

De forma a conseguir testar a métrica **3**, foi realizado um conjunto de testes que, de forma incremental, compara o número de dados gravados no dispositivo *gateway* com o número de dados armazenados na base de dados **DynamoDB**. Para cada teste efetuado, é realizado previamente a limpeza da base de dados **DynamoDB** e da base de dados do próprio *gateway.* Para armazenar o número de mensagens guardadas no *gateway* foi criada um base de dados auxiliar que armazena a contagem.

Os testes têm como referência temporal as iterações da função **uploadFromSQL**. O primeiro teste ocorreu durante duas iterações, sendo este valor incrementado nos restantes cinco testes. Importa referir que os valores armazenados não apresentam uma relação linear crescente pois a função responsável pelo envio de dados está constantemente a correr pelo que não é possível sincronizar a mesma com o início do envio dos dados por parte do ESP32. Neste caso, este facto não altera a viabilidade dos resultados.

Tabela 20 - Teste perda dados *gateway-cloud*

Teste	Iterações	DBLocal	Cloud	%
1	2	12	12	100
2	3	13	13	100
3	4	18	18	100
4	5	24	24	100
5	6	29	29	100

Analisando a Tabela 20 é possível concluir que, em funcionamento normal do *hardware*, não existe qualquer perda de dados a partir do momento em que estes são armazenadas no dispositivo *gateway* até ao seu envio para a *cloud*.

5.1.4 Métrica 4

Para definir esta métrica foi necessário averiguar, com os responsáveis pelo projeto, qual o valor limite máximo de resposta. Posto isto, e tendo em consideração diversos clientes com diferentes tipos de ambientes, foi definido que o valor máximo de resposta seria de 15 segundos.

Para obter resposta a este ponto existe um diverso número de fatores a ter em conta. A eventualidade de alertar um utilizador em menos de 15 segundos é possível quando este possui uma subscrição de *email*, assumindo que os serviços de notificação da *Amazon* funcionam de forma correta. Quando usado a *web app,* este valor pode variar entre um segundo a três minutos. Isto deve-se ao intervalo de envio de dados entre *gateway* e *cloud* (100 segundos) e à taxa de atualização dos dados na *web app* (60 segundos). Apesar de ser possível alterar o valor de atualização, por defeito, este encontra-se com um minuto, pelo que, no pior caso, o alerta só será criado, aproximadamente, três minutos depois do acontecimento.

Considerando que a subscrição do *email* está ativa, foram realizados 10 testes que cronometram o envio de *email* após a receção de uma mensagem referente à deteção de um valor anómalo.

Tabela 21 - Teste Resposta após anomalia

Teste	Tempo(s)
1	2,65
2	2,07
3	2,31
4	5,20
5	2,12
6	3,99
7	9,06
8	2,68
9	2,22
10	2,01
	3,431

Analisando a Tabela 21 é possível observar que o sistema é bastante rápido na resposta a leituras anómalas. Realizados 10 testes é possível concluir que com uma média de **3,43** segundos e um valor máximo de **9,06 segundos** de resposta entre acontecimento e receção de *email,* o sistema está apto no ponto de avaliação em questão. Para esta conclusão assume-se o funcionamento normal do serviço de notificação da *Amazon*.

5.1.5 Métrica 5

O facto de existir verificação do estado de ligação (Anexo H.2) permite garantir que não existe perda de dados entre *gateway* e *cloud* aquando de uma falha da rede. De forma a testar isto, foram realizados diversos testes onde a rede (cablada) era desligada do *gateway* durante cerca de 2 minutos. Quando reposta a mesma, todos os dados ainda guardados eram então enviados através da função ***uploadFromSQLite***.

5.1.6 Métrica 6 e 7

Para as métricas **6** e **7** foi elaborado um questionário que avalia tanto o conteúdo e funcionamento do sistema assim como a interface de modo a obter alguns dados relativos a satisfação do cliente, adequação funcional, eficiência, usabilidade, entre outras. O questionário tem também o intuito de receber opiniões dos utilizadores sobre os processos que realizou. Assim é possível identificar falhas ou melhorias a realizar no sistema. No Anexo F apresentam-se as páginas do inquérito realizado. Os resultados do inquérito são apresentados no Subsecção 5.2.2.

5.1.7 Outros Testes

Além dos testes já referidos, houve também necessidade de testar a conexão entre o microcontrolador (ESP32) e o *gateway*. Apesar de este valor ser configurável, é necessário testar a confiabilidade na ligação entre as entidades. Foram então realizados dois testes distintos com tempos de intervalo de 10 e 15 segundos. Para cada valor, realizaram-se 10 iterações, tendo como referência a execução de nove iterações completas da função **uploadFromSQLite** (10 segundos espera 90 mensagens, 15 segundos espera 51). As tabelas que se seguem apresentam os resultados obtidos nos testes.

Tabela 22 - Teste ligação ESP32 - *gateway* 10s

Teste	Esperado	Obtido	%
1	90	72	80,00
2	90	71	78,89
3	90	72	80,00
4	90	72	80,00
5	90	73	81,11
6	90	72	80,00
7	90	70	77,78
8	90	71	78,89
9	90	71	78,89
10	90	70	77,78
		71,40	79,33

Tabela 23 - Teste ligação ESP32 - gateway 15s

#Teste	Esperado	Obtido	%
1	54	51	94,44
2	54	51	94,44
3	54	51	94,44
4	54	50	92,59
5	54	49	90,74
6	54	50	92,59
7	54	48	88,89
8	54	50	92,59
9	54	48	88,89
10	54	51	94,44
		49,90	92,41

O microcontrolador, estando ligado por *wifi*, apresenta para os dois testes, um valor de mensagens enviadas de **79** e **92%** quando comparado com o valor esperado. Estas falhas devem-se a súbitas desconexões do ESP32 à rede, que, consequentemente, obrigam à desconexão também do *gateway*. É importante notar que o microcontrolador, como o nome indica, é um sistema com algumas limitações capacitativas que pode ser a origem das falhas. Verificadas estas falhas houve também o estudo do tempo de reposição da ligação ao *gateway* após falha na mesma. Este resultou numa média de **6** segundos a repor a ligação o que significa, que, no pior caso, apenas uma mensagem é perdida. Apesar da possibilidade de esta falha ser insignificante para o sistema no geral, é um ponto que pode ser melhorado de forma a evitar tempos de resposta acima do esperado.

Considerando os dados apresentados, foi definido, como valor por defeito, o uso de **15** segundos para envio de mensagens no microcontrolador.

5.2 Avaliação

Realizados os diferentes testes ao sistema é necessário efetuar uma avaliação formal do mesmo de forma incisiva e imparcial. Nesta secção é efetuada essa mesma avaliação tendo em conta

um conjunto diferente de grandezas. Assim, esta secção, inicia-se com a apresentação destas grandezas e de seguida é feita a relação das mesmas aos testes realizados. Por fim é realizada a avaliação em si onde são comparados os resultados obtidos com os resultados esperados.

5.2.1 Grandezas

Tendo em conta o sistema desenvolvido, assumiu-se uma avaliação dividida em três visões diferentes: solução, utilização e suporte de aplicação.

5.2.1.1 Solução

Na visão da solução averigua-se a qualidade geral de todo o sistema. Esta análise é feita através da análise do benefício que a solução poderá trazer à empresa cliente. Esta é feita através do estudo de:

- **Satisfação do cliente –** o *feedback* do cliente é fundamental para execução e manutenção de um Sistema com sucesso;

5.2.1.2 Utilização

Na visão da utilização do sistema procede-se à avaliação do uso do sistema. Para avaliação deste campo são usadas as normas ISO/IEC 25010 [62] que estrutura um processo de avaliação da qualidade de um produto de *software*. Esta norma apresenta oito métricas de classificação:

- **Adequação funcional –** Avalia a forma como o sistema responde às necessidades quando submetido a condições especificas. Averigua se um sistema responde a todas as funções necessárias, se fornece os resultados corretos nessas funções e, finalmente, se a forma como as funções são realizadas facilitam na execução da mesma;
- **Eficiência –** Relaciona a *performance* do sistema com a quantidade de recursos que usa. Averigua se os tempos de resposta estão em conformidade com os requisitos do sistema, se o uso de recursos não excede o necessário e se a capacidade máxima do sistema está em conformidade com os requisitos;
- **Compatibilidade –** Mede o grau com que o sistema consegue comunicar com terceiros. Apresenta duas características: coexistência (Mede a forma como o sistema funciona quando executado num sistema partilhado) e interoperabilidade (De que forma dois sistemas conseguem comunicar e partilhar informações entre si);
- **Usabilidade –** Mede de que forma um sistema pode ser usado por utilizadores específicos para realizar ações especificas e chegar a objetivos específicos de forma eficiente. Avalia a forma como os clientes reconhecem se o sistema traz valor ao negócio, a facilidade de aprendizagem de uso, a operabilidade do sistema, a acessibilidade (diferentes utilizadores conseguem usar o sistema), a estética da interface e, finalmente, a proteção contra os erros cometidos pelos utilizadores;
- **Confiabilidade –** Avalia até que ponto um sistema pode ser usado para efetuar funções especificas, sobre condições especificas e durante um período de tempo específico. Os componentes analisados são: Maturidade (operabilidade do sistema sobre uso normal), disponibilidade, tolerância a falhas e recuperabilidade em caso de falha;

- **Segurança –** Analisa como o sistema protege a informação de outros sistemas ou utilizadores. É avaliada a confidencialidade, integridade da informação e autenticidade do sistema;
- **Manutenção –** Analisa de que forma o sistema está pronto para receber atualizações para melhoramentos ou correções. Os campos a ser avaliados são: Modularidade do sistema, usabilidade, analisabilidade (capacidade de prever possíveis alterações ou identificar falhas), modificabilidade (capacidade de receber modificações de forma estável) e testabilidade;
- **Portabilidade –** Avalia a capacidade de um sistema ser transferido de um local físico (*hardware*) para outro. Neste campo são analisados: adaptabilidade (eficácia e eficiência da mudança de ambiente) e estabilidade.

5.2.1.3 Suporte

Sendo a configuração inicial do sistema feita pela equipa de desenvolvimento existe uma responsabilidade acrescida de estabelecer um ambiente de trabalho estável e seguro para os clientes. Além desta primeira configuração, a manutenção e reparação do sistema é também um fator importante na relação com o cliente e torna-se num componente fulcral para o negócio. Para esse efeito é necessário que se definam algumas métricas de forma a assegurar a qualidade do trabalho. Assim sendo definem-se como métricas de controlo de qualidade de suporte:

- **Análise de Falhas –** O serviço disponibilizado deve possuir o menor número de falhas possível. Aquando de uma falha deve haver um registo provisório e alerta à equipa. O objetivo final será de compreender e resolver no menor espaço de tempo possível as falhas;
- **Capacidade de configuração inicial –** A configuração inicial é o passo mais importante na solução visto que é a face do projeto que será entregue ao cliente. De forma a garantir que este processo é bem feito será desenvolvido um ou vários documentos do protocolo a ter aquando desta configuração, explicitando quais os passos a tomar.

5.2.2 Metodologia de Avaliação

Apresentadas as experiências e as grandezas de avaliação, é possível, neste momento, relacionar ambos de forma a avaliar o sistema no geral. Posto isto:

5.2.2.1 Visão da solução

Neste ponto é importante receber o *feedback* dos clientes ou possíveis utilizadores do sistema. Neste caso, o uso do questionário permite receber essa mesma opinião e alterar o sistema conforme a necessidade dos clientes, melhorando campos como usabilidade e *performance*. Estas respostas são expostas na **visão de suporte**.

5.2.2.2 Visão de utilização

Esta é uma visão mais prática que as outras duas. Esta avaliação cai, maioritariamente, sobre o funcionamento interno da aplicação. De forma a efetuar os testes necessários para todos os campos compreendidos nesta visão, foram realizados não só os testes já demonstrados, assim

como testes unitários ao sistema que verificam a consistência dos algoritmos criados. Além destes, o sistema foi também testado pelos dois *developers* responsáveis de forma intensiva na procura de erros, falhas, melhorias, entre outras.

Dos campos compreendidos nesta visão pode-se especificar:

- **Adequação funcional –** Os testes unitários assim como as simulações permitiram verificar este ponto;
- **Eficiência –** Neste ponto os testes residiram sobre as ligações feitas aos serviços externos. Neste caso foram realizados testes sobre a API do AWS com o intuito de perceber a sua eficiência, tentando diminuir tempos de espera;
- **Compatibilidade –** Houve um esforço extra por parte da equipa de desenvolvimento para criar uma interface compatível com diversos equipamentos. Para isso foi criada uma interface responsiva que reage prontamente às alterações de tamanho e forma. Para testes neste ponto foi usado o *browser Google Chrome* que permite simular o uso da aplicação *web* em diversos dispositivos móveis (Ver Figura 54);
- **Usabilidade –** Em termos de usabilidade, as respostas ao inquérito permitiram obter algum *feedback* importante;
- **Confiabilidade –** A avaliação a este ponto é bastante complicada. Só quando efetuada a simulação num ambiente real é que se terá uma maior perceção da reação do sistema à carga a que será submetido;
- **Segurança –** O tópico de segurança é assegurado pelos serviços ligados ao .NET da *Microsoft* já apresentados;
- **Manutenção –** O desenvolvimento do sistema foi realizado tendo em conta possíveis futuras alterações ao mesmo. Assim sendo, e de forma a facilitar trabalho futuro, todo o código foi comentado e criado de forma modular;
- **Portabilidade –** Neste ponto não foram realizados quaisquer tipos de testes.

5.2.2.3 Visão de suporte

Nesta visão é necessário ter capacidade de receber reclamações e pedidos especiais dos clientes. Para isso, numa primeira fase, através do questionário, é possível descrever os pontos fulcrais a necessitar de melhorias ou alterações. Este questionário apresentou 17 respostas às questões relativas à satisfação e usabilidade do sistema. A análise aos resultados permitiu melhor alguns pontos identificando onde haveria mais falhas ou dificuldades. Como resultados a destacar do inquérito realizado tem-se:

- Durante a execução do guião, todos os utilizadores responderam com **não** à questão "Durante o guião alguma das operações não correu da forma esperada?";
- Entre as operações menos intuitivas destaca-se a alteração de *password* e a criação de alerta;
- Numa escala de 1 a 5, 56.3% (nove respostas) dos utilizadores responderam com quatro ao nível de satisfação do sistema no geral. seis responderam com cinco e um respondeu com três;

Os resultados podem ser consultados na integra no Anexo I.

5.3 Sumário

O Capítulo de **Experimentação e Avaliação** mostra os testes realizados ao sistema desenvolvido. Estes revelaram-se bastante importantes na conceção do sistema obrigando a algumas alterações que o tornaram mais eficaz e fiável. Este Capítulo teve como objetivo responder parcialmente à questão **Q8**. Através dos testes realizados é possível averiguar que esta resposta é afirmativa: Dispositivos IoT permitem reduzir o número de acidentes. A resposta é complementada no Capítulo de conclusões que se segue.

6 Conclusões e trabalho futuro

Neste último Capítulo são apresentadas as conclusões adquiridas através do estudo e trabalho realizado. Nestas conclusões são também referenciadas as respostas às questões identificadas aquando da definição dos objetivos. De seguida são mencionados os tópicos que ficaram pendentes e que num futuro poderiam ser implementados.

6.1 Síntese

O tópico de SST é, atualmente, algo a ter em conta no dia a dia das empresas. Apesar disto, os acidentes continuam a acontecer e por vezes, além da perda material, existe também a perda de vidas humanas. De forma a evitar acontecimentos que possam gerar este tipo de perdas, existem, já, algumas soluções que, com base no uso de sensores, permitem fazer a gestão ambiental de qualquer área de trabalho, fazendo uma constante análise do risco a que está associada. Tendo isto em conta, a empresa Abaco Consultores, projetou uma solução neste campo, que, associada ao seu serviço *InCloud for SafeMed* (*Software* de gestão de SST nas empresas), permitiria fazer uma relação entre dados clínicos dos trabalhadores e ambientais.

Dado então o problema e o contexto, o objetivo do projeto seria desenvolver um sistema que usasse dispositivos IoT para monitoramento de áreas de empresas com ambientes fabris. Este monitoramento permitiria ter, em tempo real, dados relativos a diversos fatores ambientais como temperatura ou humidade, de forma a conseguir reduzir ou até evitar situações de risco. Inicialmente o projeto seria integrado no *Incloud for SafeMed* mas, mais tarde, percebeu-se que numa primeira fase seria apenas feito um sistema independente de forma a poder ser instalado em empresas que não fossem clientes do serviço de SST. Tendo o objetivo definido foram então construídas oito questões a responder durante o estudo e desenvolvimento da solução.

Depois de realizado um estudo sobre as diversas tecnologias associadas, optou-se por criar um sistema que incluísse os serviços da AWS. Este sistema, estaria dividido em quatro partes. Na primeira parte, **sensores**, foi utilizado um microcontrolador ESP32, que, ligando-se à rede, tem a capacidade de enviar mensagens ao segundo componentes, **Gateway**. Este, terá como base

um Raspberry Pi que terá implantado no seu sistema o componente **AWS Greengrass**, responsável pela gestão de dispositivos e mensagens de forma local. Este, é também responsável pelo envio das mensagens para o terceiro componente, a ***cloud***. Nesta, são apenas usados serviços AWS, nomeadamente **AWS IoT**, **AWS DynamoDB**, **AWS API Gateway**, entre outros. Como último componente, tem-se a **web app**. Aplicação responsável por permitir aos clientes a gestão de todos os dispositivos e consulta dos dados.

6.2 Objetivos Alcançados

Depois de implementado o sistema foi possível realizar os testes de forma a responder às questões criadas aquando da definição dos objetivos. Assim conseguiu-se perceber se os objetivos propostos foram alcançados.

6.2.1 Questão Q1

A questão **Q1** foi a primeira a ser definida e tinha como objetivo reunir todas as tecnologias que poderiam estar presentas na solução final de forma a serem estudadas e validadas. Depois de realizado um estudo superficial foi possível averiguar que seria necessário estudar:

- **IoT** – Tendo a solução base em dispositivos IoT, era necessário averiguar como funcionam os sistemas que englobam estes dispositivos. Entre estes estudos pode-se realçar: protocolos de comunicação existentes; plataformas computacionais que sirvam de base; sensores disponíveis;
- **Big Data** – Tratando-se de sistemas IoT era certa a produção massiva de informação. Tendo isto em mente era imprescindível realizar um estudo relativo a bases de dados capazes de gerir esta afluência;
- **Plataformas de gestão** – Depois de emitidos os dados por parte dos dispositivos, era necessário haver algum tipo de gestão. Para isso teriam de ser estudadas quais as plataformas existentes capazes do mesmo. Neste campo, houve um estudo tanto de plataformas *open-source* como plataformas alojadas na *cloud*.

A resposta a esta questão pode ser encontrada no Capítulo 2 sendo também complementada no Capítulo 3.

6.2.2 Questão Q2

A questão **Q2**, respondida no Capítulo 3, aborda a forma como podem ser obtidos os dados de um determinado local.

Para responder a esta questão foi necessário, numa primeira fase, compreender que tipo de plataformas computacionais teriam a capacidade de emitir dados. Este estudo iniciou-se com as placas Arduino que rapidamente foram descartadas devido a não se conseguirem ligar à rede de forma independente. De seguida foram estudados os microcontroladores ESP8266 e ESP32.

Ambos possuem a capacidade de ligação à rede e, sendo o último mais potente, foi a escolha para a emissão de dados. Este microcontrolador, estando ligado a diversos sensores poderia emitir dados relativos a qualquer área necessitando apenas de uma fonte de alimentação. O ESP32 emite dados em *JSON* que identificam dados do microcontrolador em si e dados sensoriais lidos. Esta configuração pode ser encontrada no Anexo H.1.

6.2.3 Questão Q3

Relacionada com a questão anterior, a **Q3** tinha como objetivo identificar que tipos de dispositivos estariam disponíveis no mercado. Para resposta a esta questão foi identificada uma lista de alguns sensores disponíveis assim como preços de aquisição destes. Esta lista pode ser encontrada na Secção 3.4.3.

6.2.4 Questão Q4

Depois de enviados os dados por parte do **microcontrolador** é necessário receber e tratar os mesmos. Esta gestão pode ser feita tanto no local de emissão através de um *gateway* como na *cloud* caso a emissão de dados seja direta. Para o primeiro caso, foi usado a ferramenta **AWS GreenGrass** instalada num Raspberry Pi. Esta, permite receber os dados dos sensores, criar regras de valores limites e enviar os mesmos para a *cloud* de forma a serem armazenados de forma permanente. Através do uso de um *gateway* é possível reduzir custos, delimitando quando devem ser enviadas as leituras para a nuvem de forma a serem consultadas.

A resposta a esta questão pode ser encontrada com mais detalhe no Capítulo 3 sendo complementada no Capítulo 4.

6.2.5 Questão Q5

Aquando da ocorrência de uma leitura anómala deveria ser iniciado um processo de notificação dos responsáveis pela fábrica. Para isso, através das ferramentas do **AWS IoT** foi possível criar um sistema de e*mail* que, aquando da deteção de uma anomalia, notificasse os responsáveis. Este processo mostrou-se bastante eficaz apresentando tempos de resposta média abaixo dos 5 segundos desde a deteção da anomalia. Além deste sistema, a própria *web app* identificava as situações, porém fora dos valores limites definidos.

6.2.6 Questão Q6

A questão **Q6** relativa à arquitetura tinha como objetivo definir quais os componentes a adotar para desenvolver o sistema. Esta encontra-se respondida no Capítulo 4 onde é demonstrado todo o processo de criação arquitetural da solução. Esta acabou por ser realizada usando serviços oferecidos pela *Amazon* nomeadamente: **AWS GreenGrass**, **AWS IoT**, **AWS DynamoDB**,

AWS API *gateway*, entre outros, em interação com um microcontrolador **ESP32** e uma ***web app*.**

6.2.7 Questão Q7

Sendo que a arquitetura usa serviços da *Amazon* é necessário ter em conta os custos que pode vir a ter. A questão **Q7** respondida na secção 4.9 tinha como objetivo tentar simular aquilo que seria o preço que a arquitetura adotada poderia ter. Porém, a resposta a estão questão não pode ser dada de forma precisa e correta devido à variação dos preço conforme o uso dos serviços.

6.2.8 Questão Q8

A questão final **Q8** é a questão fulcral da dissertação aqui apresentada. Tem como objetivo final, averiguar se o sistema desenvolvido consegue, de facto, minimizar o número de acidentes de trabalho. De forma a responder a esta questão foram realizados diversos testes que averiguaram a forma como o sistema respondia à deteção de anomalias. Estes testes mostraram a rapidez e eficácia do sistema aqui desenvolvido e, de certa forma, comprovam que o sistema poderia evitar e, consequentemente, reduzir o número de acidentes em ambientes fabris.

No entanto é preciso ter em conta que os testes realizados foram feitos em ambientes simulados diferentes daquilo que poderiam ser na vida real. Para responder de forma convicta a esta questão seria necessário efetuar testes em condições mais similares àquilo que seria o ambiente de execução do sistema.

Porém, e concluindo, atendendo à análise e testes realizados ao projeto, pode-se concluir que a deteção de valores anómalos de forma preventiva por parte de dispositivos inteligentes, pode reduzir o número de acidentes em ambientes fabris.

6.3 Limitações e Trabalho futuro

Durante a implementação do sistema houve alguns fatores que acabaram por limitar os testes e execução das simulações. Estes, foram planeados tendo em conta que haveria uma série de microcontroladores e sensores disponíveis para os mesmos. Porém, no momento da sua realização, por alguma alteração de interesse por parte dos responsáveis, acabou por não haver essa possibilidade, o que acabou por limitar tanto na qualidade como na quantidade dos testes, impossibilitando alguns conclusões.

Sendo este apenas uma primeira iteração do projeto existe alguma margem de progressão. Devido a alguns problemas não foi possível a inclusão de tecnologias de *machine learning* como

foi projetado. No futuro seria interessante incluir este serviço de forma a tentar prever algumas situações que possam ser sazonais.

Além disto podem também ser realizados mais testes em ambientes mais próximos daquilo que será o ambiente final. Nestes testes devem ser usados sensores físicos acoplados aos microcontroladores evitando a emissão de dados simulados como foi realizado até então.

Referências

[1] R. v. d. Meulen, “Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016,” Gartner, 7 Fevereiro 2017. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016. [Acedido em 2 Novembro 2018].

[2] IDC, “Data Growth, Business Opportunities, and the IT Imperatives,” Abril 2014. [Online]. Available: https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm. [Acedido em 2 Novembro 2018].

[3] B. Marr, “Big Data: 20 Mind-Boggling Facts Everyone Must Read,” Forbes, 30 Setembro 2015. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/#5bcd011e17b1. [Acedido em 2 Novembro 2018].

[4] NewGen apps, “8 Uses, Applications, and Benefits of Industrial IoT in Manufacturing,” 15 Dezembro 2017. [Online]. Available: https://www.newgenapps.com/blog/8-uses-applications-and-benefits-of-industrial-iot-in-manufacturing. [Acedido em 5 Novembro 2018].

[5] i-Scoop, “The Industrial Internet of Things (IIoT): the business guide to Industrial IoT,” 2016. [Online]. Available: https://www.i-scoop.eu/internet-of-things-guide/industrial-internet-things-iiot-saving-costs-innovation/. [Acedido em 5 Novembro 2018].

[6] Direção-Geral de Saúde, “Acidentes de Trabalho,” Serviço Nacional de Saúde, 2010. [Online]. Available: https://www.dgs.pt/saude-ocupacional/doencas-profissionais-e-acidentes-de-trabalho/acidentes-de-trabalho.aspx. [Acedido em 4 Novembro 2018].

[7] Gabinete de Estratégia e Planeamento, “Acidentes de trabalho Estatisticas,” 23 Outubro 2018. [Online]. Available:

http://www.gep.msess.gov.pt/estatistica/acidentes/at2016sint.pdf. [Acedido em 2 Novembro 2018].

[8] Autoridade para as Condições de Trabalho, "Acidentes de trabalho tipo," 2016. [Online]. Available: http://www.act.gov.pt/(pt-PT)/Campanhas/Campanhas%20a%20decorrer/CampanhaIbericadePrevencaodeAcidentesdeTrabalho/acidentestrabalhotipo/Paginas/default.aspx. [Acedido em 4 Novembro 2018].

[9] Abaco Consultores, "inCloud for SafeMed," 2018. [Online]. Available: https://abaco.consulting/en/Product/incloud-for-safemed/. [Acedido em 4 Dezembro 2018].

[10] J. Montezinho e S. Almeida, Janeiro 2019. [Online]. Available: https://expressodasilhas.cv/economia/2019/01/12/estamos-prontos-para-a-quarta-revolucao-industrial/61822. [Acedido em 19 Maio 2019].

[11] C. Ribas, "Jornal Económico," 2017. [Online]. Available: https://jornaleconomico.sapo.pt/noticias/nao-aprovar-industria-4-0-a-quarta-revolucao-industrial-182746. [Acedido em 16 Fevereiro 2019].

[12] V. C. Pinheiro, "Indústria 4.0 a Quarta Revolução industrial," 2016. [Online]. Available: http://www.poci-compete2020.pt/destaques/detalhe/Industria_4ponto0. [Acedido em 19 Fevereiro 2019].

[13] IoTConnect, "Iot-Enabled Smart Connetced Worker," 2018. [Online]. Available: https://www.iotconnect.io/smart-iot-connected-worker-solutions.html. [Acedido em 14 Dezembro 2018].

[14] IoTConnect, "IoTConnect.io," 2018. [Online]. Available: https://help.iotconnect.io/2018/08/21/hello-world/?section=modules-5. [Acedido em 20 Fevereiro 2019].

[15] Avnet, "Employee monitoring solution to increase safety," 2018. [Online]. Available: https://www.avnet.com/wps/portal/us/solutions/iot/cloud-and-digital-services/digital-solutions/smart-connected-worker/. [Acedido em 14 Dezembro 2018].

[16] C. M. Vigna, "Fuzzy Front End: Linha de Frente da Inovação," 8 Julho 2015. [Online]. Available: https://www.ibm.com/developerworks/community/blogs/tlcbr/entry/mp238?lang=en. [Acedido em 28 Novembro 2018].

[17] P. A. Koen, G. M. Ajamian, S. Boyce, A. Clamen, E. Fisher, S. Fountoulakis, A. Jonhson, P. Puri e R. Seibert, "Fuzzy Front End: Effective Methods, Tools, and Techniques," 2002. [Online]. Available: http://www.stevens-tech.edu/cce/NEW/PDFs/FuzzyFrontEnd_Old.pdf. [Acedido em 27 Novembro 2018].

[18] P. A. Koen, "Understanding the Front End: A common Language and Structured Picture," 25 Maio 2004. [Online]. Available: https://www.kip.zcu.cz/kursy/imi/ERASMUS/KoenMay26UnderstandFrontEndPDMA_01.pdf. [Acedido em 28 Novembro 2018].

[19] P. A. Koen, H. M. J. Bertels e E. Kleinschmidt, “Managing the Front End of Innovation,” 2014.

[20] Carlos Reis, “Cadeia de Valor”.

[21] T. Woodall, “Conceptualising 'Value for the Customer': An Atttributal, Structural and Dispositional Analysis,” 2003.

[22] A. Lindgreen e F. Wynstra, “Value in business markets: What do we know? Where are we going?,” 2005.

[23] D. A. Aaker, Building Strong Brands, The Free Press, New York, 1996.

[24] NOS, “Empreendorismo,” [Online]. Available: http://www.nos.pt/empresas/repositorio-informacao/criar-uma-empresa/guias-teoricos/Pages/business-model-canvas.aspx. [Acedido em 12 Novembro 2018].

[25] M. Rouse, “IoT Analysis Guide,” 2016. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT. [Acedido em 31 Outubro 2018].

[26] RedPixie, “The World of Data: 32 Essential stats about Big Data and IoT,” [Online]. Available: https://www.redpixie.com/hubfs/Red%20Pixie%20-%20The%20World%20of%20Data%20infographic%20Black%20and%20Blue.pdf. [Acedido em 29 Outubro 2018].

[27] CriticalSoftware, “Internet of things,” [Online]. Available: https://www.criticalsoftware.com/pt/what-we-do/internet-of-things. [Acedido em 31 Outubro 2018].

[28] K. K. Patel e S. M. Patel, “IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges,” India, 2016.

[29] M. Rouse, “IoT Security,” 2016. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/IoT-security-Internet-of-Things-security. [Acedido em 31 Outubro 2018].

[30] Oracle, “What is Big Data?,” 2017. [Online]. Available: https://www.oracle.com/big-data/guide/what-is-big-data.html. [Acedido em 29 Outubro 2018].

[31] MongoDB, “Introduction to MongoDB,” 2008. [Online]. Available: https://docs.mongodb.com/manual/introduction/. [Acedido em 29 Outubro 2018].

[32] MongoDB, “Internet of Things,” 2015. [Online]. Available: https://www.mongodb.com/use-cases/internet-of-things. [Acedido em 30 Outubro 2018].

[33] Apache, “Cassandra,” 2009. [Online]. Available: http://cassandra.apache.org/. [Acedido em 30 Outubro 2018].

[34] Datastax, “NoSQL Apache Cassandra database for Internet of Things, sensor data, and time series,” [Online]. Available: https://academy.datastax.com/use-cases/internet-of-things-time-series. [Acedido em 30 Outubro 2018].

[35] IBM, “Apache Hbase,” 2018. [Online]. Available: https://www.ibm.com/analytics/hadoop/hbase. [Acedido em 30 Outubro 2018].

[36] Amazon, “Amazon DynamoDB,” 20 Outubro 2018. [Online]. Available: https://aws.amazon.com/dynamodb/. [Acedido em 30 Outubro 2018].

[37] DB-Engines, 30 Outubro 2018. [Online]. Available: https://db-engines.com/en/system/Amazon+DynamoDB%3BCassandra%3BHBase%3BMongoDB. [Acedido em 30 Outubro 2018].

[38] AdaFruitLearning, “HTTP,” 14 Dezembro 2017. [Online]. Available: https://learn.adafruit.com/alltheiot-protocols/http. [Acedido em 29 Outubro 2018].

[39] OASIS, “MQTT Version 3.1.1,” 29 Outubro 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf. [Acedido em 29 Outubro 2018].

[40] K. Pepples, “Internet of things MQTT Quality of Service Levels,” 20 Abril 2015. [Online]. Available: https://dzone.com/articles/internet-things-mqtt-quality. [Acedido em 26 Outubro 2018].

[41] A. Stanford-Clark e H. L. Truong, “MQTT For Sensor Networks (MQTT-SN),” 14 Novembro 2013. [Online]. Available: http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf. [Acedido em 29 Outubro 2018].

[42] CoAP, “CoAP,” 2016. [Online]. Available: http://coap.technology/. [Acedido em 29 Outubro 2018].

[43] T. Jaffey, “MQTT and CoAP, IoT Protocols,” 2014. [Online]. Available: https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php. [Acedido em 29 Outubro 2018].

[44] A. C. Ajit e K. N. Mininath, “Secure CoAP Using Enhanced DTLS for Internet of Things,” 2014.

[45] C. Gündoğan, P. Kietzmann, M. Lenders, H. Petersen, T. C. Schmidt e M. Wählisch, “NDN, CoAP, and MQTT: A Comparative,” 28 Setembro 2018. [Online]. Available: https://arxiv.org/pdf/1806.01444.pdf. [Acedido em 29 Outubro 2018].

[46] Arduino, “What is Arduino?,” [Online]. Available: https://www.arduino.cc/en/Guide/Introduction. [Acedido em 2 Novembro 2018].

[47] Espressif, “ESP32 Series,” 2019. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Acedido em 10 Junho 2019].

[48] Raspberry Pi, “Raspberry Pi,” [Online]. Available: https://www.raspberrypi.org/. [Acedido em 19 Dezembro 2018].

[49] ThingsBoard, “What is ThingsBoard?,” [Online]. Available: https://thingsboard.io/docs/getting-started-guides/what-is-thingsboard/. [Acedido em 6 Dezembro 2018].

[50] ThingsBoard, “What is ThingsBoard IoT Gateway?,” [Online]. Available: https://thingsboard.io/docs/iot-gateway/what-is-iot-gateway/. [Acedido em 6 Dezembro 2018].

[51] Mainflux, “WHAT IS MAINFLUX,” [Online]. Available: https://www.mainflux.com/. [Acedido em 6 Dezembro 2018].

[52] K. I. Plataform, “Kaa concepts,” [Online]. Available: https://docs.kaaiot.io/DOC/docs/current/Kaa-concepts/. [Acedido em 12 Fevereiro 2019].

[53] Amazon, “AWS IoT,” 2015. [Online]. Available: https://aws.amazon.com/iot/. [Acedido em 12 Dezembro 2018].

[54] Google, “Google Cloud Iot,” 2017. [Online]. Available: https://cloud.google.com/solutions/iot/. [Acedido em 12 Dezembro 2018].

[55] Google, “Cloud IoT Core,” 2017. [Online]. Available: https://cloud.google.com/iot-core/. [Acedido em 12 Dezembro 2018].

[56] Google, “Cloud Pub/Sub,” 2017. [Online]. Available: https://cloud.google.com/pubsub/. [Acedido em 12 Dezembro 2018].

[57] Microsoft, “Azure IoT Hub,” 2016. [Online]. Available: https://azure.microsoft.com/en-us/services/iot-hub/. [Acedido em 12 Dezembro 2018].

[58] Microsoft, “Azure IoT Edge,” 2016. [Online]. Available: https://azure.microsoft.com/en-us/services/iot-edge/. [Acedido em 12 Dezembro 2018].

[59] Microsoft, “Azure IoT Central,” 2016. [Online]. Available: https://azure.microsoft.com/en-us/services/iot-central/. [Acedido em 12 Dezembro 2018].

[60] Microsoft, “Azure IoT Solution Accelerator,” 2016. [Online]. Available: https://azure.microsoft.com/en-us/features/iot-accelerators/. [Acedido em 12 Dezembro 2018].

[61] Python, “time — Time access and conversions,” 2019. [Online]. Available: https://docs.python.org/3/library/time.html#time.sleep. [Acedido em 2019 Maio 31].

[62] ISO25000, “ISO/IEC 25010,” 2011. [Online]. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010. [Acedido em 29 Janeiro 2019].

[63] Mongoose, “Mongoose,” 2019. [Online]. Available: https://mongoosejs.com/. [Acedido em 26 Maio 2019].

[64] RestSharp, “RestSharp,” 2019. [Online]. Available: http://restsharp.org/. [Acedido em 28 Maio 2019].

[65] “Accidents at work statistics,” Eurostat, [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php/Accidents_at_work_statistics . [Acedido em 2 Novembro 2018].

[66] D. Hughes e C. Don, “Turning new Product development into a Continuous Learning.”.

[67] DeviceHive, “How to choose your IoT Plataform - Should you Go Open-Source?,” 24 Outubro 2017. [Online]. Available: https://medium.com/iotforall/how-to-choose-your-iot-platform-should-you-go-open-source-23148a0809f3.

[68] T. Hoff, “How Big Is A Petabyte, Exabyte, Zettabyte, Or A Yottabyte?,” 11 Setembro 2012. [Online]. Available: http://highscalability.com/blog/2012/9/11/how-big-is-a-petabyte-exabyte-zettabyte-or-a-yottabyte.html. [Acedido em 9 Novembro 2018].

[69] JWT, “Introduction to JSON Web Tokens,” [Online]. Available: https://jwt.io/introduction/. [Acedido em 8 Fevereiro 2019].

[70] E. Mosquitto, “Eclipse Mosquitto,” 2018. [Online]. Available: https://mosquitto.org/. [Acedido em 2 Novmebro 2018].

Anexos

Anexo A – Custos AWS

Número de Dispositivos	**1000**
Número Horas Diárias	**24**
Meses de Atividade	**12**
Dias de Atividade	**365**
Ligação Local-Cloud (Minutos)	**2**
Tempo de Ligação (Minutos)	**1**

Preços Frankfurt	
AWS GreenGrass	**0,22**
Cores	10
AWS IoT Core	
Ligação	**0,096**
Mensagens	**1,2**
Shadow	**1,5**
Rules	**0,18**
Device Managment	
Registo	**0,12**
Indexação	**2,7**
Updates	**0,035**
IoT Analytics	
Data Processing	**0,2**
Data Storage	**0,03**
Queries	**6,5**
Analise customisada	**0,36**

Dolar para Euro	0,88

AWS Anual	
GreenGrass	**23,232**
IoT Core	
Ligação	**0,252288**
Mensagens	**3,1536**
Shadow	**0,552975**
Rules Triggered	**0,23652**
Actions Executed	**0,23652**
Total $	**4,431903**
Total €	**3,900075**

Device Managment		**Euro**
Registo	**0,12**	**0,1056**
Indexação	**8,1**	**7,128**
Updates	**35**	**30,8**

IoT Analytics	
Data processing	23,4375
Data Storage	3,515625
Queries	0,743866
Análise	0,9
Total $	**28,59699**
Total €	**25,16535**

Base de Dados		
Amazon DynamoDB		
Escrita	**1,525**	/1000000
Leitura	**0,305**	/1000000
Armazenamento	**0,306**	>25GB Por mês
Backup Continuo	**0,2448**	/GB
Backup por Request	**0,1224**	/GB
Restauro de tabela	**0,1836**	/GB

Anexo B – Dados Simulação

Simulação

Inventário

- Raspberry Pi 3 b+
- 4x nodemcu ESP8266 (5,70€- https://mauser.pt/catalog/product_info.php?products_id=84577) = 22,8€
- 4x sensores Temperatura e humidade (8,95€ - https://www.electrofun.pt/sensores-arduino/sensor-humidade-temperatura-dht22) (https://www.amazon.es/DollaTek-Digital-Temperatura-Humedad-reemplazar/dp/B07DK4M1PS?SubscriptionId=AKIAIJYBMV6KUA55XTDA&tag=haggle-web-es-21&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B07DK4M1PS) = 15,99€
- 4x sensor ruido (4,70€ - https://www.electrofun.pt/sensores-arduino/sensor-de-som-para-arduino) = 18,8€
- 4x Sensor qualidade do ar (4,30€ - https://www.electrofun.pt/sensores-arduino/sensor-de-gas-mq135) (https://www.amazon.es/AZDelivery-135-Gas-sensor-Calidad-Arduino/dp/B07CNNKQ5R/ref=sr_1_2?ie=UTF8&qid=1549879732&sr=8-2&keywords=mq-135) = 11,99€
- 4x Sensor de chama ou fogo (3,30€ - https://www.electrofun.pt/sensores-arduino/modulo-sensor-de-chama-fogo) = 13,2€

Total: 82,78€

1º NodeMCU - Sala Edge
- DHT22;
- Sensor ruido;
- MQ-135;
- Sensor Chama;

2º NodeMCU - Sala
- DHT22;
- Sensor ruido;
- MQ-135;
- Sensor Chama;

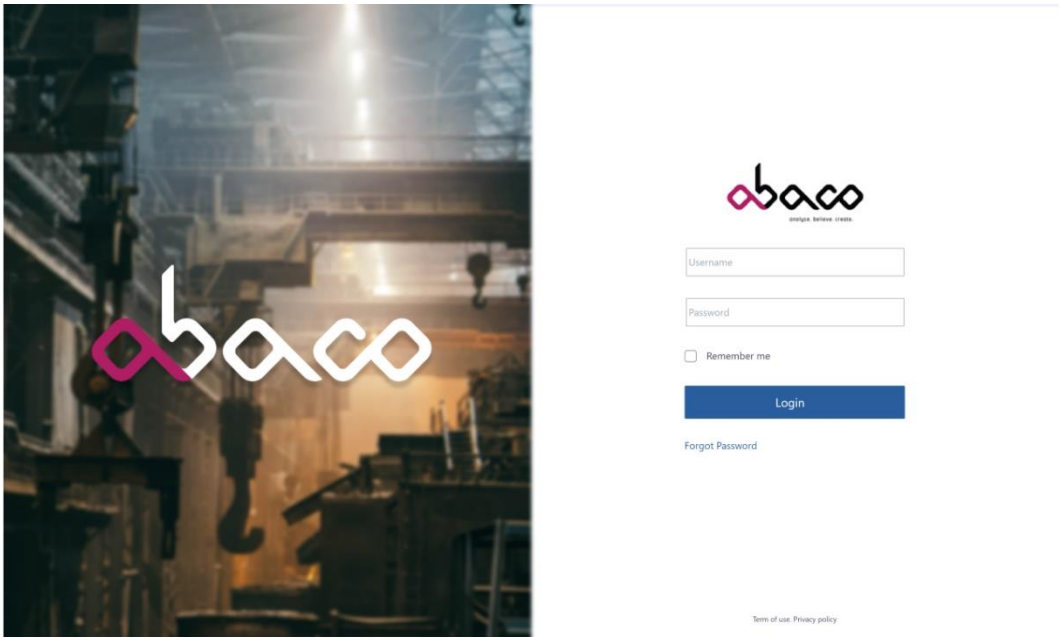
3º NodeMCU - Sala
- DHT22;
- Sensor ruido;
- MQ-135;
- Sensor Chama;

4º NodeMCU – Sala de refeições
- Sensor ruido;
- Sensor Chama;
- M-135

Raspberry Pi 3b+ - Sala Edge
- Web app:
  - Definir métricas;
  - Definir métodos de aviso;
  - Definir tempos de envio;
- *Lambdas;*
- Shadow;
- Verificar Segurança de dados;
- ...

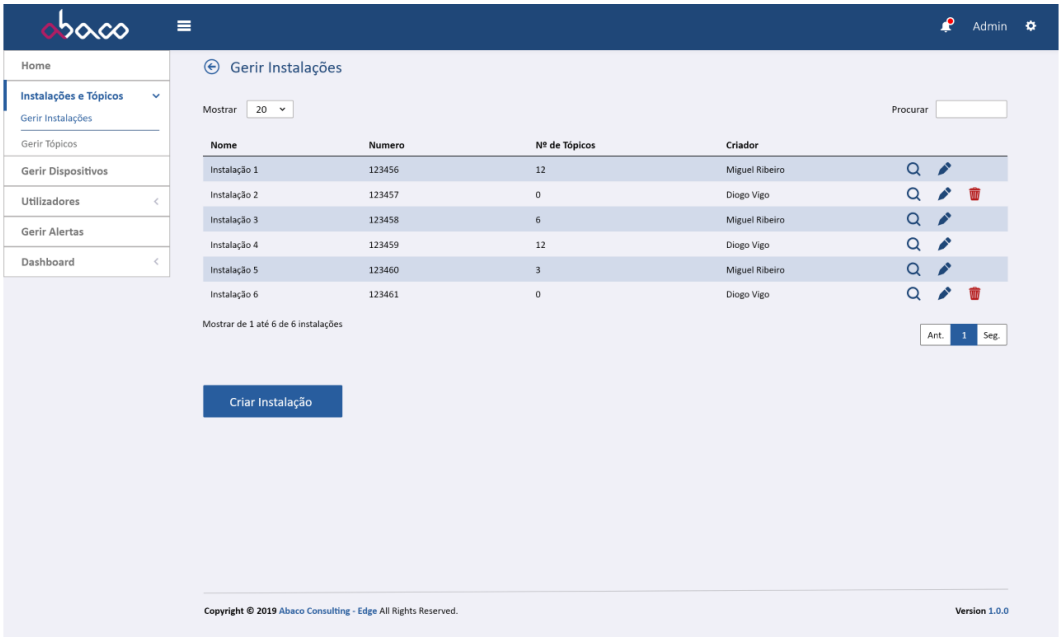
Anexo C – Mockups Web App

Login:

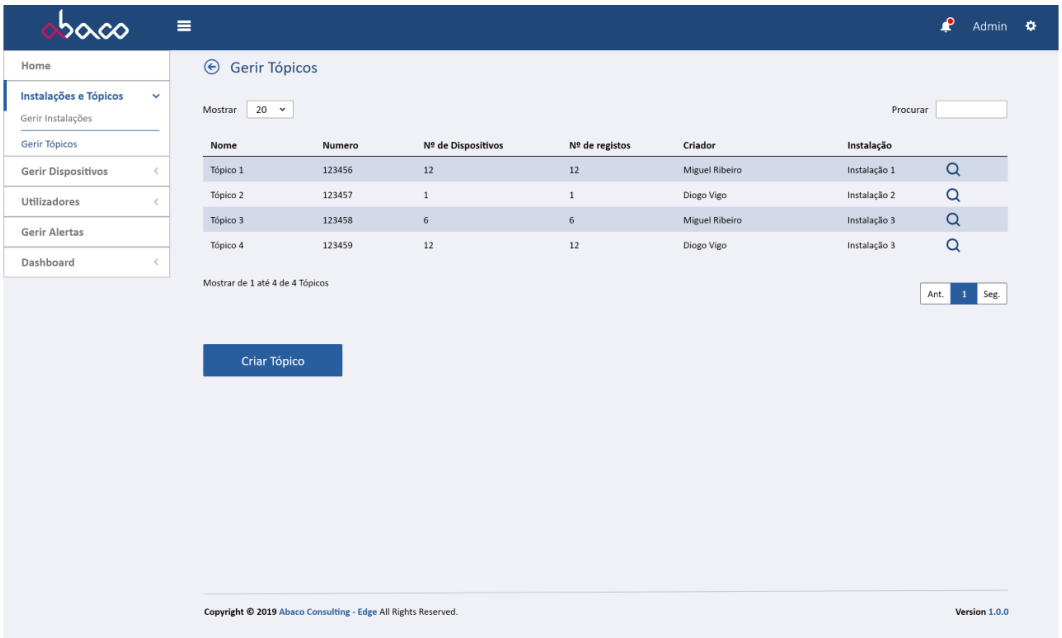


Home:

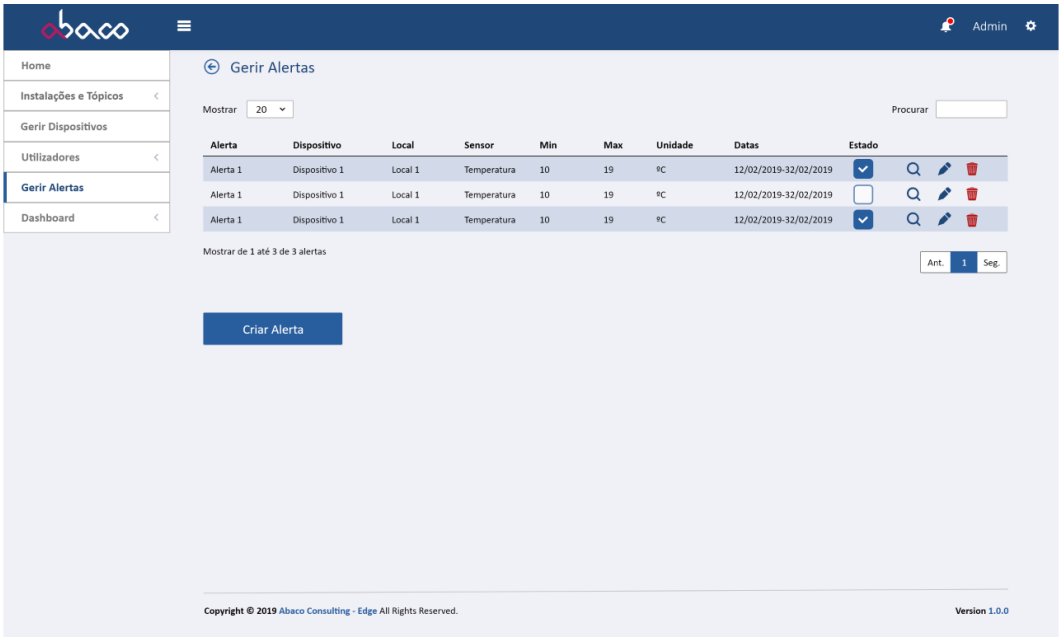
Gerir Instalações:



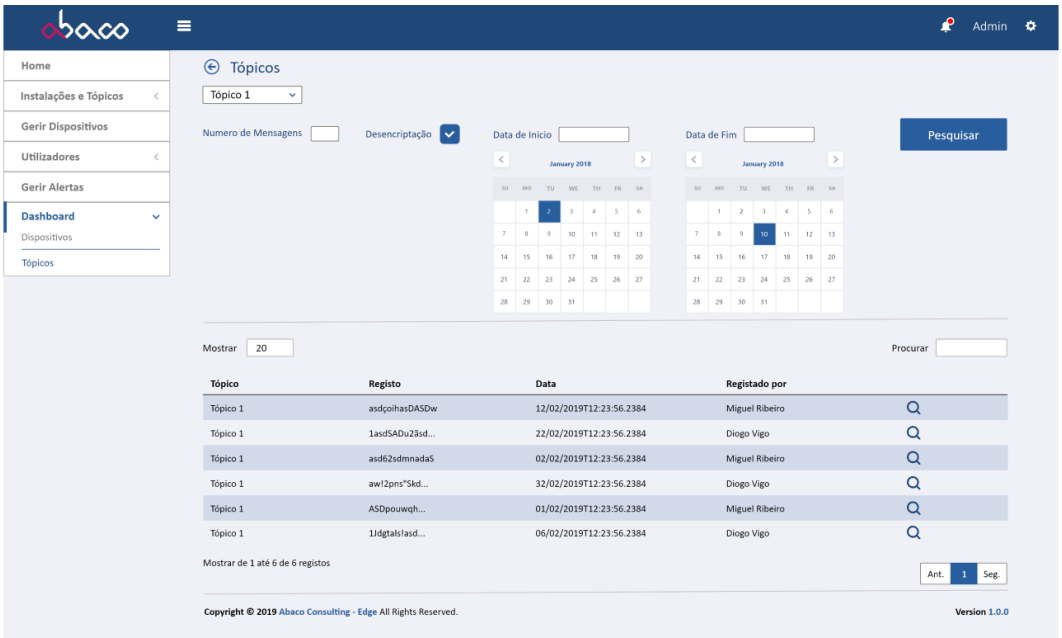
Gerir tópicos:



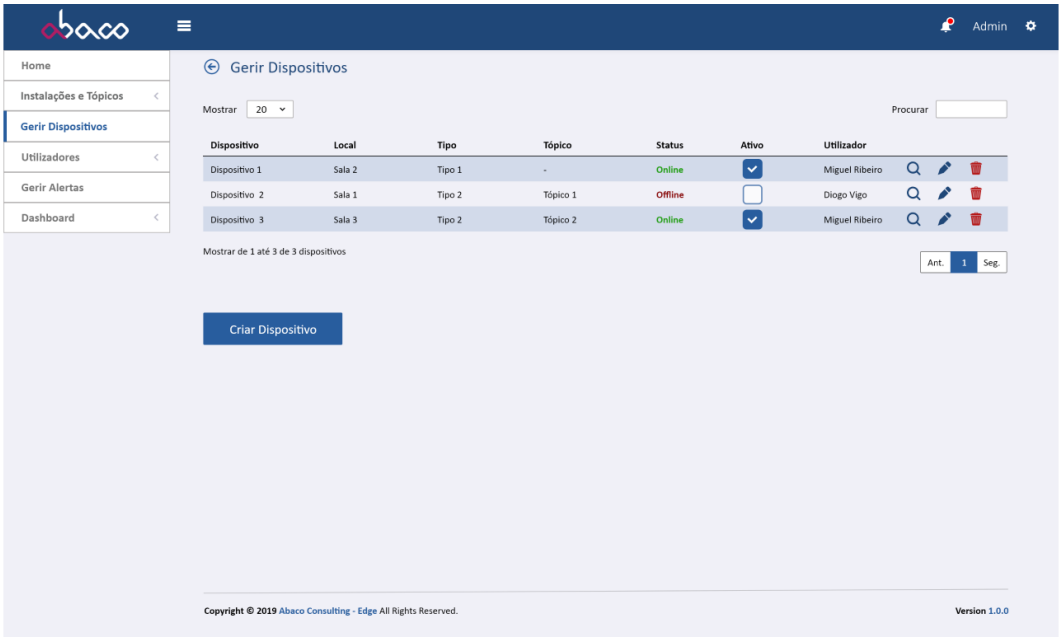
Gerir Alertas:



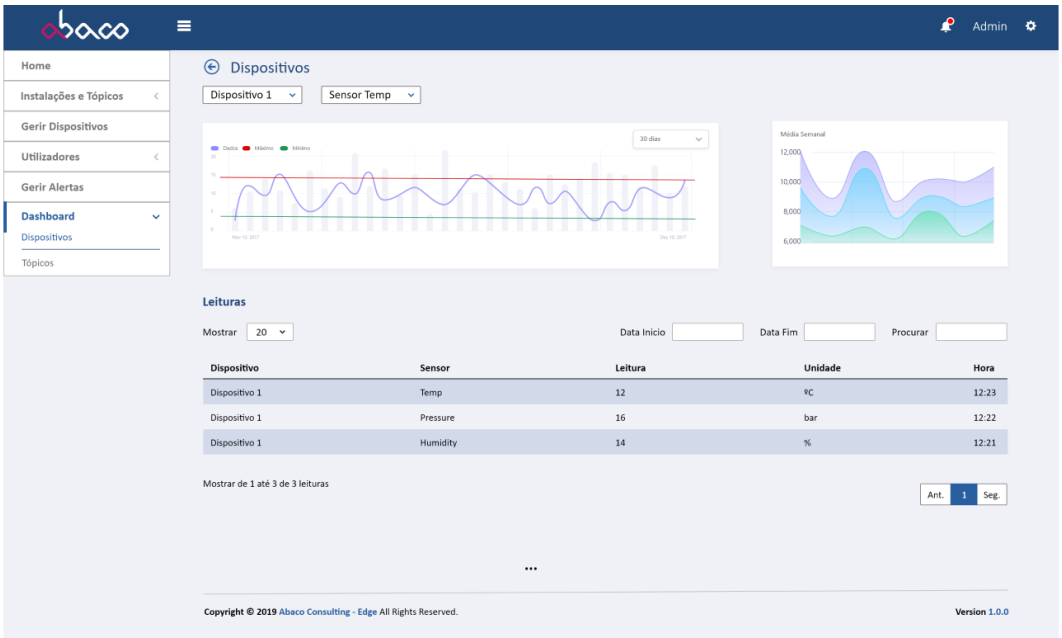
Dashboard Tópicos:



Gerir Dispositivos:

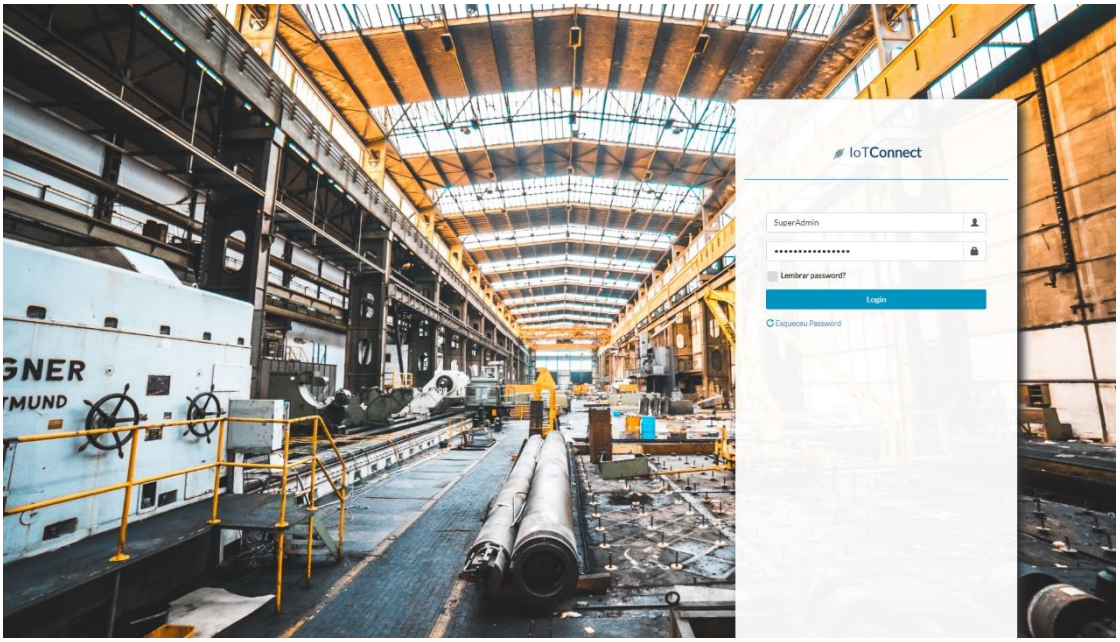


Dashoboard Dispositivos:

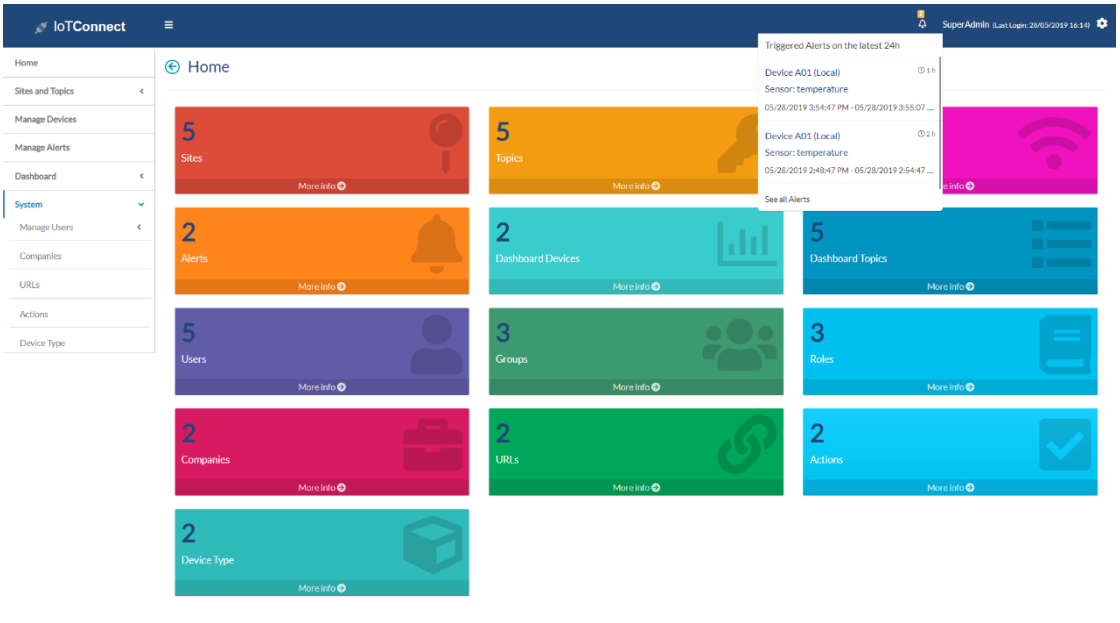


Anexo D - Interfaces *Web App*

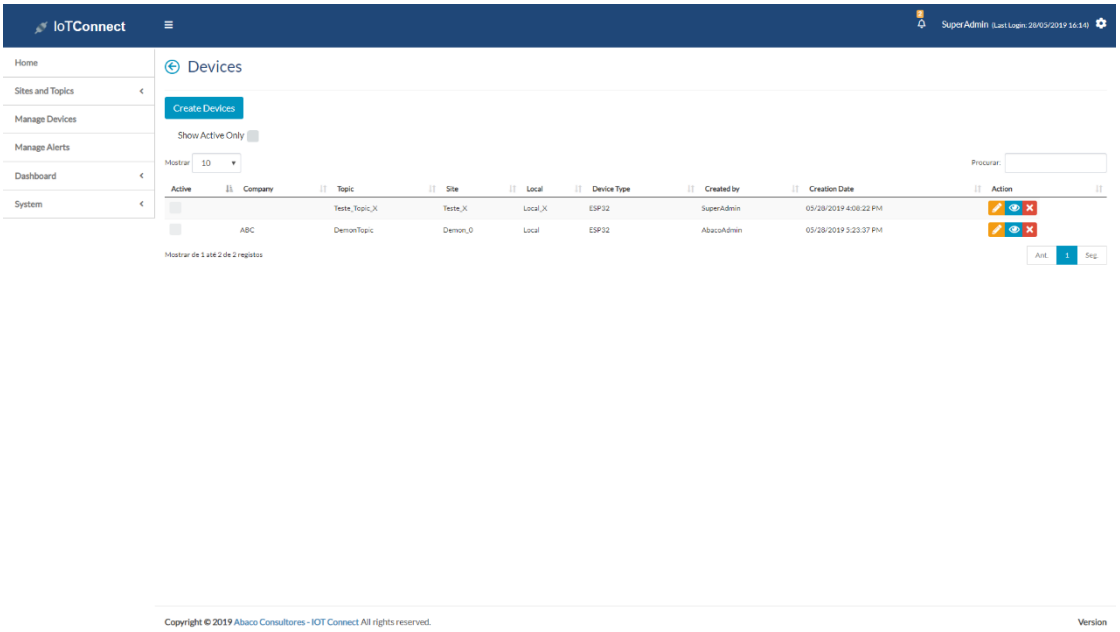
Login:



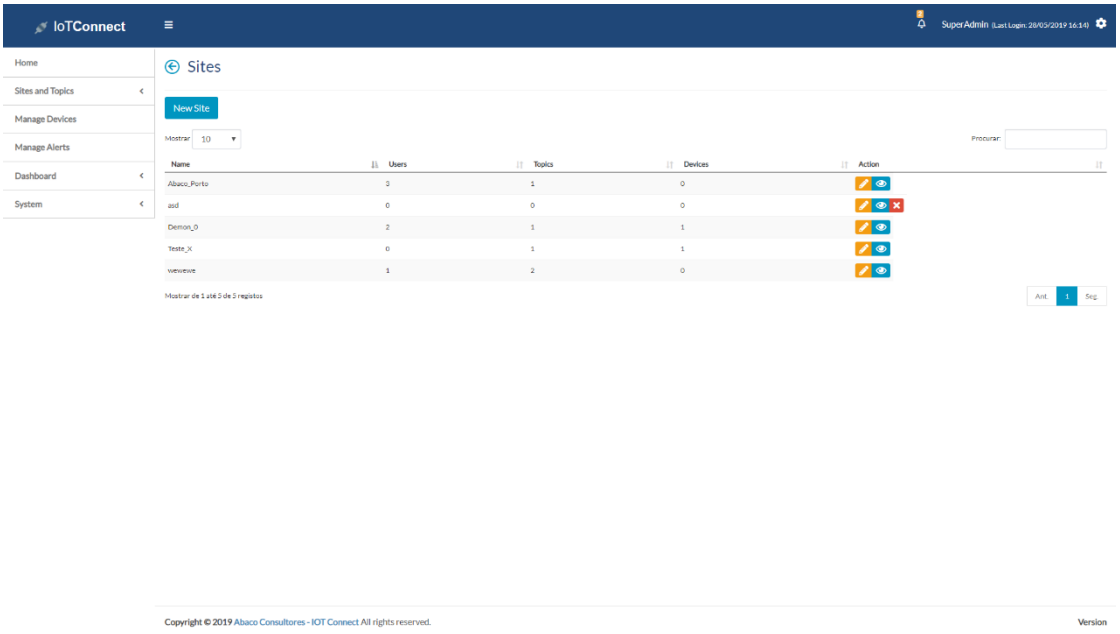
Home (Com alertas ativos):



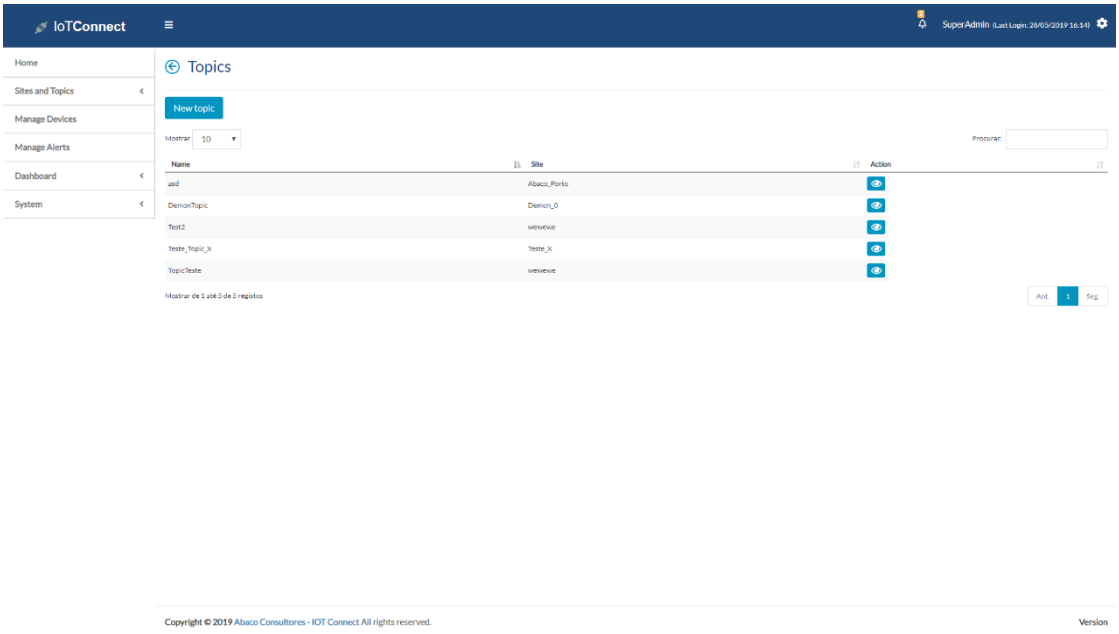
Gestão de dispositivos:



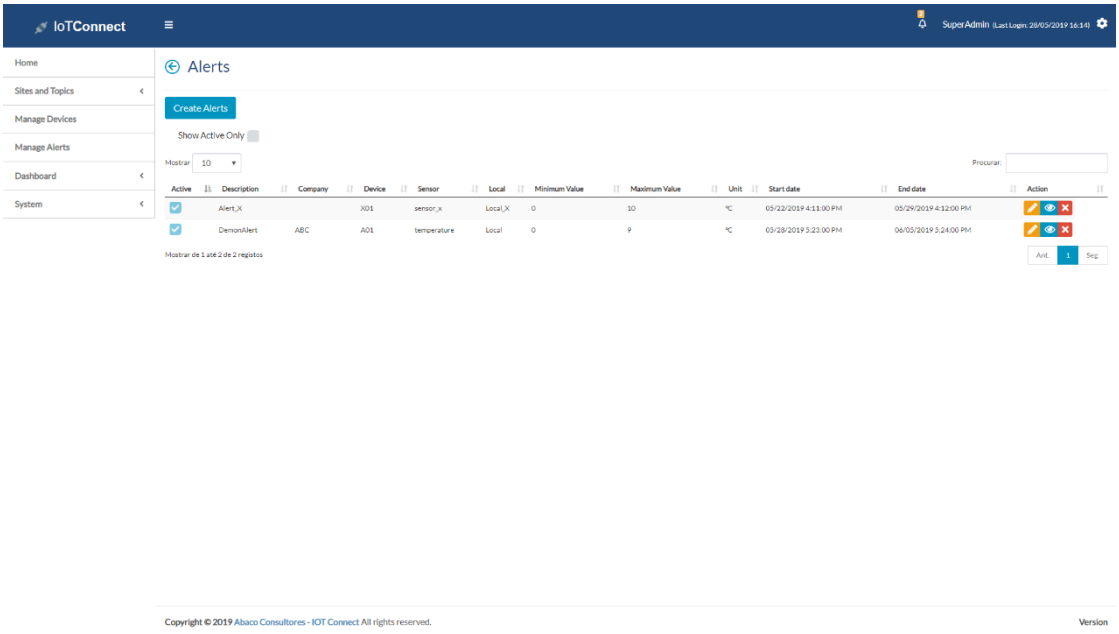
Gestão de instalações:



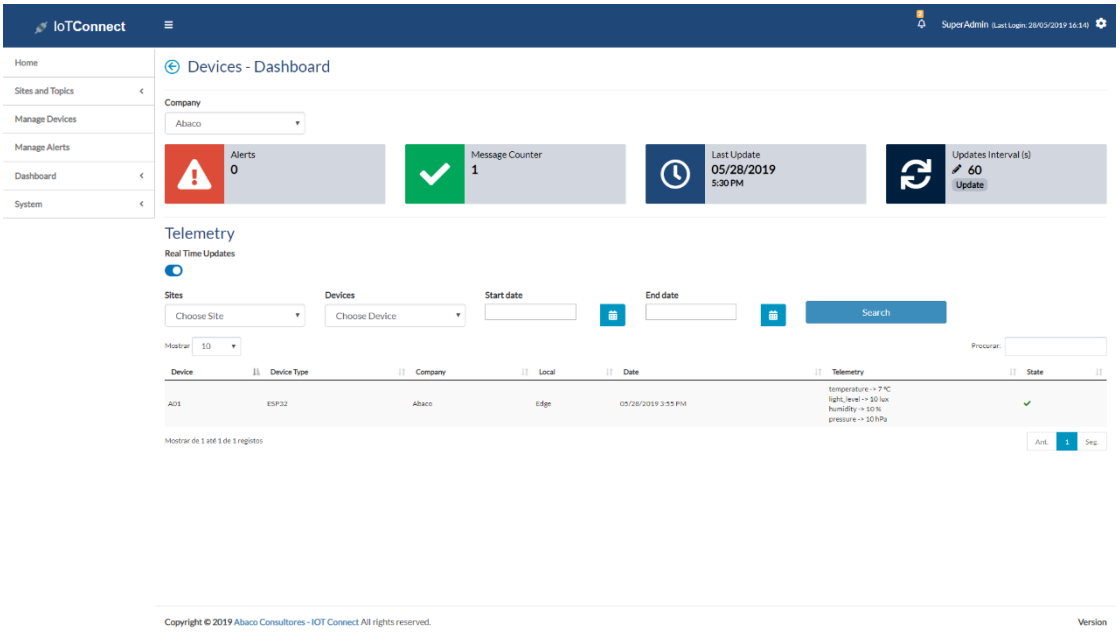
Gestão de tópicos:



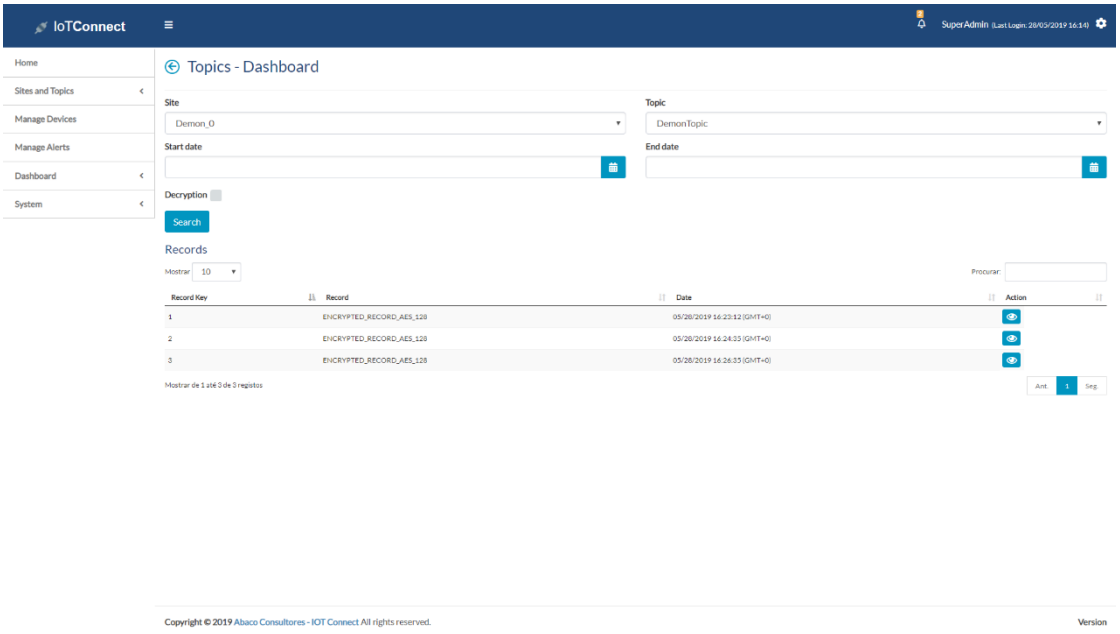
Gestão de alertas:



Dashboard de dispositivos:



Dashboard tópicos:

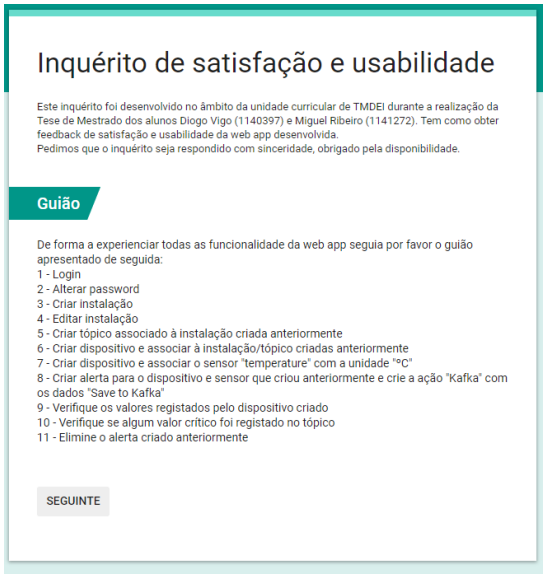


Anexo E - Dados dos testes de resposta ao Gateway

Teste tempo de resposta gateway			
#Teste	Tempo (s)	Intervalo(s)	Resultado
1	<1	2->4	Correto
2	<1	2->4	Correto
3	<1	2->4	Correto
4	<1	2->4	Correto
5	<1	2->4	Correto
6	<1	2->4	Correto
7	<1	2->4	Correto
8	<1	2->4	Correto
9	<1	2->4	Correto
10	<1	2->4	Correto
11	<1	2->4	Correto
12	<1	2->4	Correto
13	<1	2->4	Correto
14	1.3	2->4	Correto
15	<1	2->4	Correto
16	<1	2->4	Correto
17	<1	2->4	Correto
18	<1	2->4	Correto
19	<1	2->4	Correto
20	<1	2->4	Correto
21	<1	2->4	Correto
22	<1	2->4	Correto
23	<1	2->4	Correto
24	<1	2->4	Correto
25	<1	2->4	Correto
26	<1	2->4	Correto
27	<1	2->4	Correto
28	<1	2->4	Correto
29	<1	2->4	Correto
30	<1	2->4	Correto
31	1.7	2->4	Correto
32	<1	2->4	Correto
33	<1	2->4	Correto
34	<1	2->4	Correto
35	<1	2->4	Correto
36	<1	2->4	Correto
37	<1	2->4	Correto
38	<1	2->4	Correto
39	<1	2->4	Correto
40	<1	2->4	Correto

Teste tempo de resposta gateway			
#Teste	Tempo (s)	Intervalo(s)	Resultado
41	<1	2->4	Correto
42	<1	2->4	Correto
43	<1	2->4	Correto
44	<1	2->4	Correto
45	<1	2->4	Correto
46	<1	2->4	Correto
47	<1	2->4	Correto
48	<1	2->4	Correto
49	<1	2->4	Correto
50	<1	2->4	Correto

Anexo F - Inquérito de satisfação



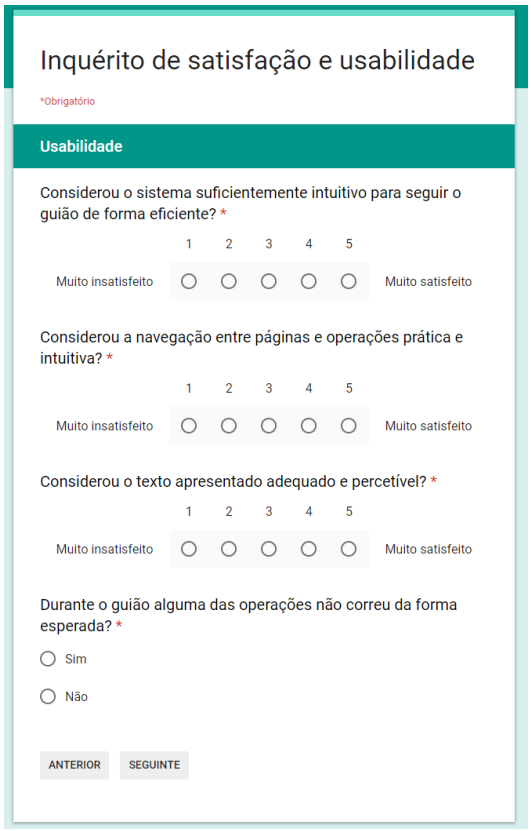
Inquérito de satisfação e usabilidade

Este inquérito foi desenvolvido no âmbito da unidade curricular de TMDEI durante a realização da Tese de Mestrado dos alunos Diogo Vigo (1140397) e Miguel Ribeiro (1141272). Tem como obter feedback de satisfação e usabilidade da web app desenvolvida.
Pedimos que o inquérito seja respondido com sinceridade, obrigado pela disponibilidade.

Guião

De forma a experienciar todas as funcionalidade da web app seguia por favor o guião apresentado de seguida:
1 - Login
2 - Alterar password
3 - Criar instalação
4 - Editar instalação
5 - Criar tópico associado à instalação criada anteriormente
6 - Criar dispositivo e associar à instalação/tópico criadas anteriormente
7 - Criar dispositivo e associar o sensor "temperature" com a unidade "°C"
8 - Criar alerta para o dispositivo e sensor que criou anteriormente e crie a ação "Kafka" com os dados "Save to Kafka"
9 - Verifique os valores registados pelo dispositivo criado
10 - Verifique se algum valor crítico foi registado no tópico
11 - Elimine o alerta criado anteriormente

SEGUINTE



Inquérito de satisfação e usabilidade

*Obrigatório

Usabilidade

Considerou o sistema suficientemente intuitivo para seguir o guião de forma eficiente? *

	1	2	3	4	5	
Muito insatisfeito	○	○	○	○	○	Muito satisfeito

Considerou a navegação entre páginas e operações prática e intuitiva? *

	1	2	3	4	5	
Muito insatisfeito	○	○	○	○	○	Muito satisfeito

Considerou o texto apresentado adequado e percetível? *

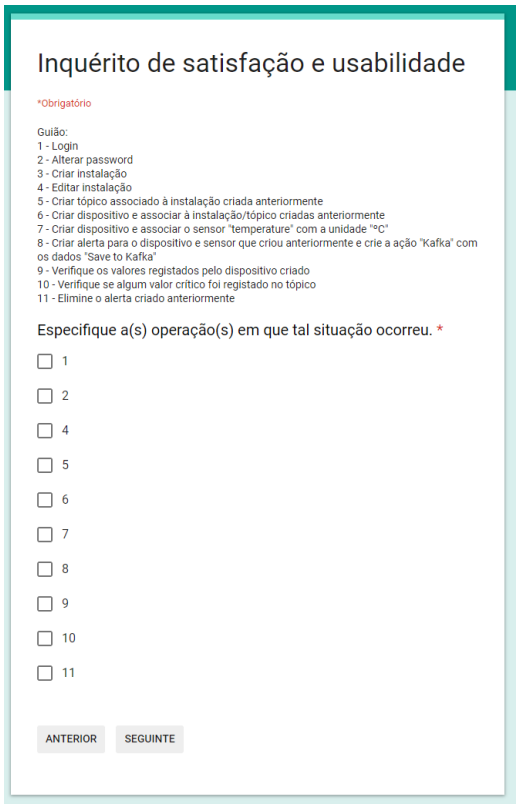
	1	2	3	4	5	
Muito insatisfeito	○	○	○	○	○	Muito satisfeito

Durante o guião alguma das operações não correu da forma esperada? *

○ Sim

○ Não

ANTERIOR SEGUINTE



Inquérito de satisfação e usabilidade

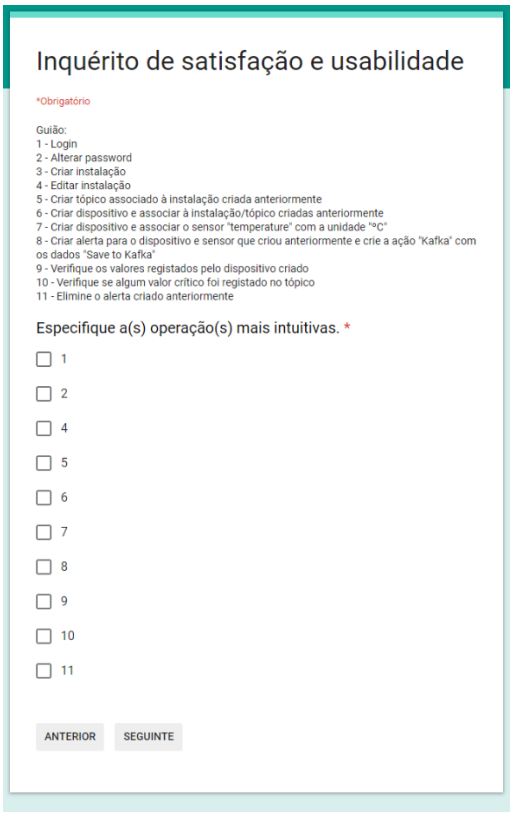
*Obrigatório

Guião:
1 - Login
2 - Alterar password
3 - Criar instalação
4 - Editar instalação
5 - Criar tópico associado à instalação criada anteriormente
6 - Criar dispositivo e associar à instalação/tópico criadas anteriormente
7 - Criar dispositivo e associar o sensor "temperature" com a unidade "°C"
8 - Criar alerta para o dispositivo e sensor que criou anteriormente e crie a ação "Kafka" com os dados "Save to Kafka"
9 - Verifique os valores registados pelo dispositivo criado
10 - Verifique se algum valor crítico foi registado no tópico
11 - Elimine o alerta criado anteriormente

Especifique a(s) operação(s) em que tal situação ocorreu. *

☐ 1
☐ 2
☐ 4
☐ 5
☐ 6
☐ 7
☐ 8
☐ 9
☐ 10
☐ 11

ANTERIOR SEGUINTE



Inquérito de satisfação e usabilidade

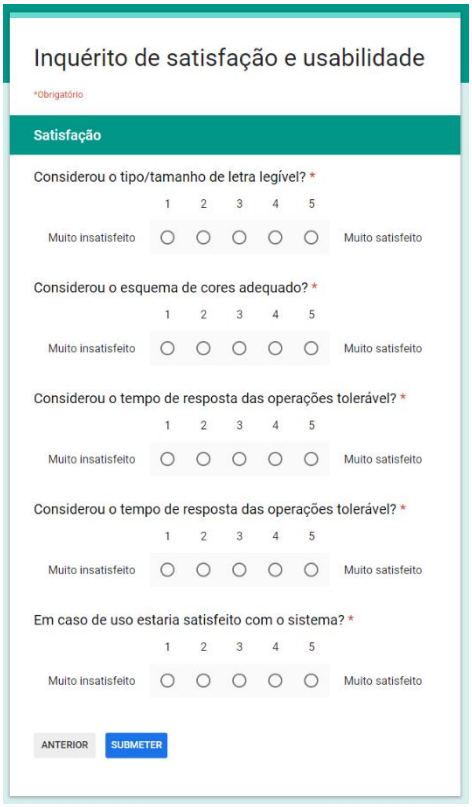
*Obrigatório

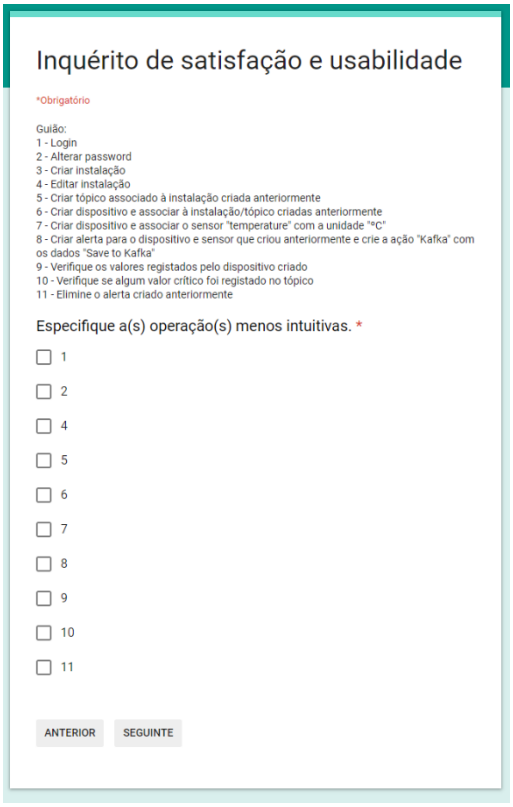
Guião:
1 - Login
2 - Alterar password
3 - Criar instalação
4 - Editar instalação
5 - Criar tópico associado à instalação criada anteriormente
6 - Criar dispositivo e associar à instalação/tópico criadas anteriormente
7 - Criar dispositivo e associar o sensor "temperature" com a unidade "°C"
8 - Criar alerta para o dispositivo e sensor que criou anteriormente e crie a ação "Kafka" com os dados "Save to Kafka"
9 - Verifique os valores registados pelo dispositivo criado
10 - Verifique se algum valor crítico foi registado no tópico
11 - Elimine o alerta criado anteriormente

Especifique a(s) operação(s) mais intuitivas. *

☐ 1
☐ 2
☐ 4
☐ 5
☐ 6
☐ 7
☐ 8
☐ 9
☐ 10
☐ 11

ANTERIOR SEGUINTE





Anexo G – Arquitetura Alternativa

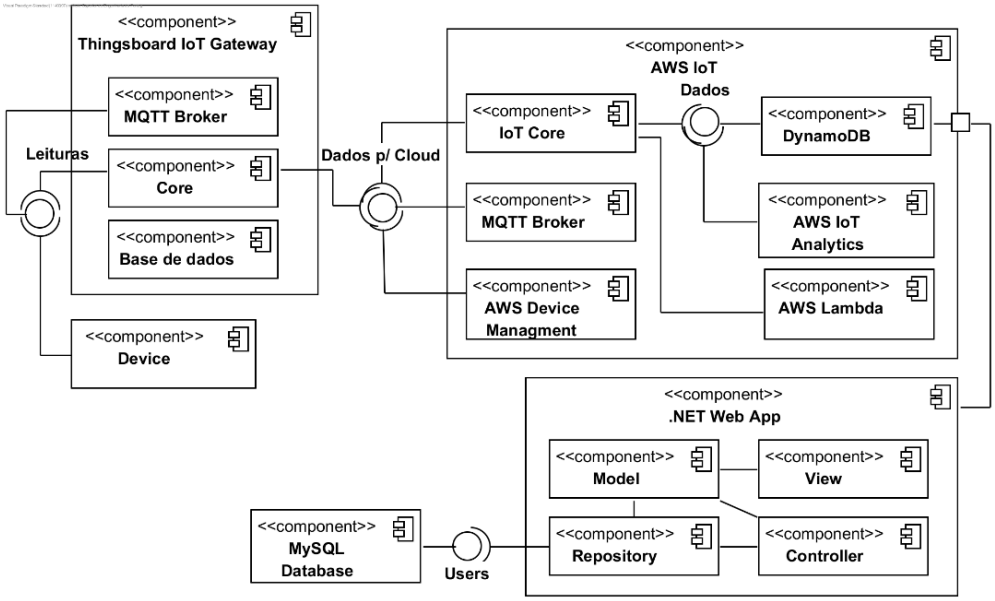


Figura 41 - Diagrama de componentes - Alternativa 1

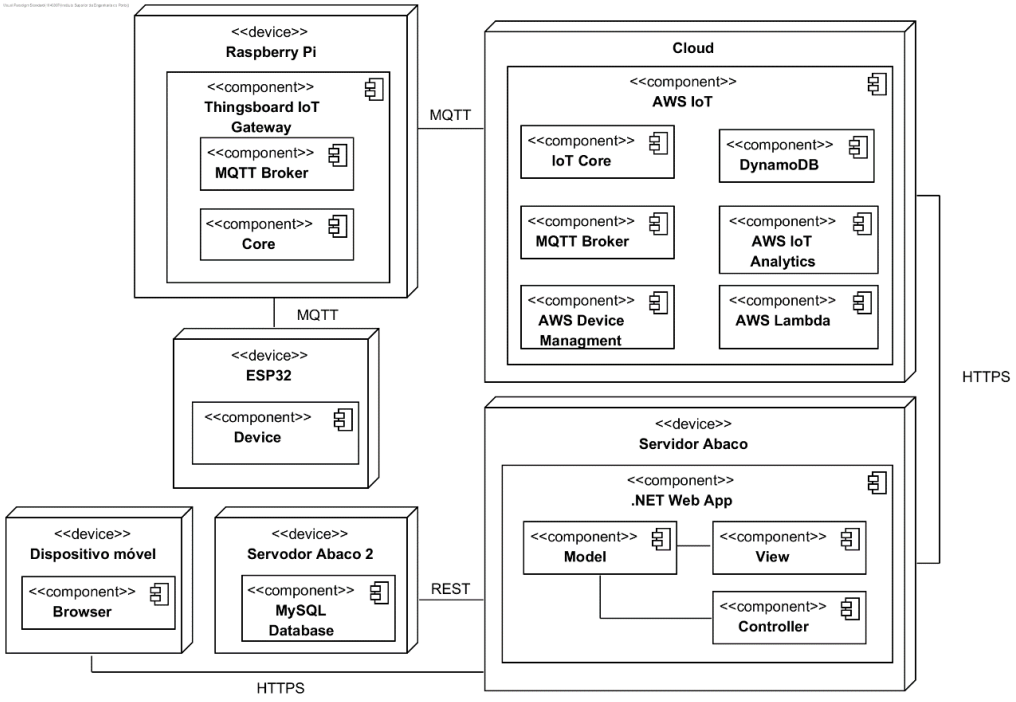


Figura 42 - Diagrama de *Deployment* - Alternativa 1

A arquitetura é bastante similar à já apresentada. Esta, difere-se pelo uso de um *gateway open source* oferecido pela *ThingsBoard*, já apresentada neste documento. O planeamento arquitetural desta solução deve-se caso seja necessário reduzir custos. Usando um serviço *open source* poderá ser feito algum controlo sobre os custos, evitando o uso de mais um componente *Amazon*. Esta solução apresenta, no *gateway*, uma abordagem mais limitada quanto as suas funções. Neste caso, o *Thingsboard gateway,* apenas é usado para fazer a ligação a outros serviços e reenviar mensagens. Apresentando menor capacidade, esta solução foi descartada sendo usada a arquitetura composta apenas pelos serviços AWS.

Anexo H – Implementação da Solução

Este Capítulo tem como objetivo sintetizar todo o trabalho desenvolvido para chegar à solução final. Este, inclui todo o processo desde o desenvolvimento dos componentes até à execução da simulação final. O Capítulo está dividido em cinco Subcapítulos que mostram o trabalho desenvolvido nos diferentes componentes do sistema. A ordem de apresentação segue a ordem de transferência dos dados do sistema proposto.

H.1 Sensores

Estes são os dispositivos responsáveis pela leitura e emissão de dados dentro da sua área de atuação. Durante o desenvolvimento do trabalho foram usados dois microcontroladores denominados de ESP32 e ESP8266. Na solução apresentada, cada um destes dispositivos será

colocado num local estratégico e emitirá dados relativos a esse mesmo local. De forma a ser possível identificar o local em si, são atribuídos aos dispositivos valores específicos para os seguintes campos:

- ***CompanyID*** – Identifica qual a empresa cliente (ex: “Abaco”);
- ***LocalId*** – Identificador do local a receber o dispositivo (ex: “Sala Edge”);
- ***DeviceType*** – Sendo dispositivos diferentes, é importante identificar o seu tipo (ESP32 ou ESP8266);
- ***DeviceID*** – Identificar do dispositivo em si (ex: “A01”);

Tendo estes campos preenchidos, é então possível a criação do *JSON* a enviar para o dispositivo *gateway*. Este é composto pela seguinte estrutura:

```
{
  "Topic": "Abaco/Telemetry/Edge/ESP32/A01",
  "device": {
            "active": "True",
            "localID": "Edge",
            "companyID": "Abaco",
            "deviceType": "ESP32",
            "deviceID": "A01",
  },
  "telemetry": [
        {"sensor": "temperature", "unit": "ºC", "value": 10},
        {"sensor": "light_level", "unit": "lux", "value": 10},
        {"sensor": "humidity", "unit": "%", "value": 10},
        {"sensor": "pressure", "unit": "hPa", "value": 10}
 ],
  "system": {
  "battery": "34",
  "uptime": 20.813364,
  "free_ram": 31192
  }
}
```

Listagem de Código 1- JSON de emissão dos dispositivos

Além dos dados identificadores do dispositivo e dos valores lidos pelos sensores, são também incluídos dados de sistema e o tópico para qual a mensagem é publicada. De forma a implementar este código nos sistemas, foi usado o sistema operativo Mongoose [63] que permite a configuração dos dispositivos e a sua ligação à *cloud* e ao próprio *gateway.* Outra mais valia deste sistema é a capacidade de permitir fazer o *update* dos dispositivos sem a necessidade de estar no mesmo local físico através do OTA (*over-the-air) updates.*

A mensagem, depois de criada e transformada em JSON é enviada para o *gateway* com QoS igual a 1. Este processo é demonstrado na listagem de código que se segue.

```
1.  let messageJSON = JSON.stringify(message);
2.  let ok = MQTT.pub(topic, messageJSON, 1);
3.  print('Published: ', ok ? 'yes' : 'no');
```

Listagem de Código 2 - Publicação de Mensagem

Este processo, de envio de menagem, repete-se a cada 10 segundos através do uso da função *Timer*:

```
1.  Timer.set(10000, true, poll_sensors, null);
```

De forma a alterar o tempo de envio de leituras cada dispositivo subscreve ao tópico *Abaco/Update/IDdispositivo*. Deste modo, o *gateway* consegue enviar informação para que este intervalo de tempo seja alterado conforme a necessidade do cliente. Além desta subscrição, cada dispositivo possui também a possibilidade de publicar leituras com valores recebidos. Desta forma é possível forçar uma leitura anómala. O intuito desta subscrição é estritamente para testes.

H.2 Gateway

Como dispositivo de *gateway* foi usado um Raspberry Pi 2. Este serviu de *host* para a tecnologia AWS GreenGrass, já no documento mencionada. Esta instalação é relativamente simples quando seguido o tutorial da própria Amazon.

O uso deste equipamento deve-se, não só à capacidade de resposta mais rápida, mas também o controlo de custos para envio de mensagens para a *cloud*. Assim sendo, este dispositivo é responsável por receber informação dos microcontroladores através do seu *MQTT Broker*. Aquando da chegada desta informação, esta é analisada e armazenada. Caso se detete algum tipo de anomalia nos valores é então, de imediato, enviada uma mensagem para a *cloud* com o conteúdo da leitura realizada pelo ESP.

Para obter este processo é necessário o uso das funções *Lambda* fornecidas pela *amazon.* Neste caso, foi usada a linguagem *Python* para o seu desenvolvimento. Depois de concluídas, estas são introduzidas no *gateway* e é definida a forma de ativação de cada uma. Este processo de configuração é realizado através da consola do AWS IoT com recurso a subscrições, como

demonstrado na Figura 43. Estas apresentam uma fonte, destino, e o tópico de comunicação entre ambos.

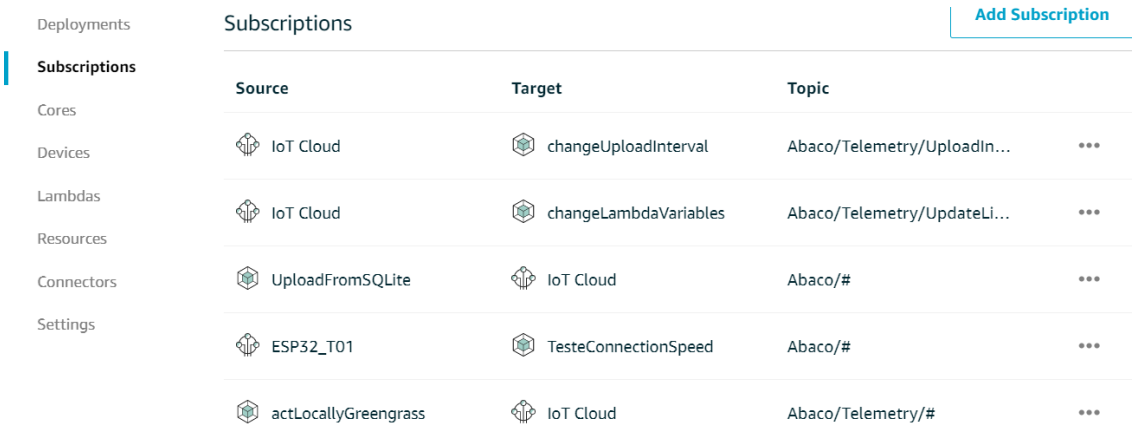


Figura 43 - Painel de configuração das funções lambda no AWS GreenGrass

Conforme estas configurações é então criado um fluxo de dados. A Figura 44 representa as funções *lambda* encarregues de receber e tratar os dados.

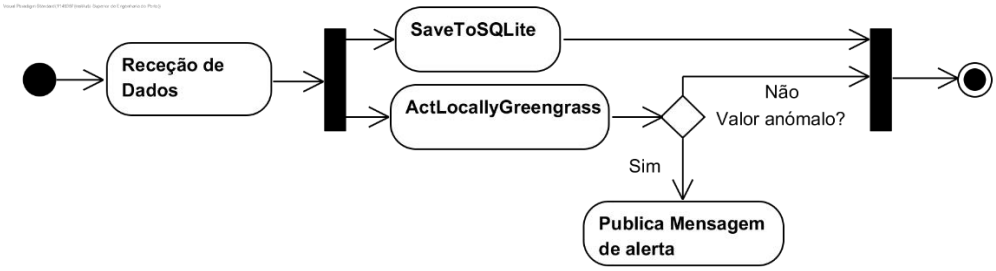


Figura 44 - Fluxo de dados AWS GreenGrass – Receção e tratamento

A receção dos dados é feita pelo *Broker* do AWS GreenGrass. Conforme o tópico de chegada da informação, podem ser ativadas determinadas funções:

- **SaveToSQLite** – Esta função tem como objetivo o armazenamento dos dados na base de dados *SQLite3* instalada no Raspberry Pi. Quando ativada, esta é responsável por mapear os dados recebidos de forma a poderem ser guardados no ficheiro especificado. Este ficheiro foi criado numa pasta especifica criada no *root* do Raspberry denominada de *GG_BD*. Depois da sua criação foi necessário atribuir as permissões de leitura e escrita necessárias. Para que o processo possa ser realizado com sucesso é necessário especificar, nas configurações, que a função não deve correr dentro de um *container*.

O processo inicia-se com o mapeamento da informação executado da seguinte maneira:

```
1.  def lambda_handler(event, context):
2.      # create Json
3.      topicReceived = event['Topic']
4.      data= {}
5.      data['Topic'] = topicReceived
6.      data['Device'] = event['device']
7.      data['DeviceID'] = data['Device']['deviceID']
8.      data['Telemetry'] = event['telemetry']
9.      data['System'] = event['system']
10.     data['Date'] = calendar.timegm(time.gmtime())
```

Listagem de Código 3 - Mapeamento de dados - SaveToSQLite

O JSON recebido apresenta a estrutura já apresentada e é enviado através do parâmetro *event.* À informação enviada é adicionada uma data que representa a marca temporal da leitura no resto do sistema. Depois de mapeada esta informação é feita, então, a inserção da nova mensagem na base de dados. Neste caso, é usado o ficheiro *telemetry.db*.

- **ActLocallyGreenGrass** – Este *lambda* tem como objetivo verificar se os valores obtidos estão em conformidade com os limites inseridos pelo cliente. Para isso, à semelhança da função anterior, é necessária que esta não corra em nenhum *container* de modo a ter acesso ao ficheiro de dados *variables.db*. Este tem como conteúdo duas tabelas. Para esta função é usada apenas a tabela *limits* que, como o nome indica, define quais os limites para as leituras dos sensores. Esta tabela apresenta seis campos de entrada: identificador do dispositivo, identificador do sensor, valor mínimo e máximo, data de início e data de fim. A Listagem de Código 4 apresenta o JSON exemplificativo. Este será também usado mais tarde para criação de regras.

```
1.  {
2.      "deviceId": "ESP32_A01",
3.      "sensor":"temperature",
4.      "min":7,
5.      "max": 15,
6.      "begin":1553770112,
7.      "end": 1585392512
8.  }
```

Listagem de Código 4 - JSON Limites - ActLocallyGreenGrass

A função inicia-se com a verificação da existência de limites:

```
# Gets called on any device sending data
# Checks if that deviceID has any rules set
def lambda_handler(event, context):
    conn = sqlite3.connect(sqlite_file)
    c = conn.cursor()
    c.execute('SELECT * FROM limits WHERE deviceId= "' + event['device']
       ['deviceID']+ '"')
    rows = c.fetchall()
    if(len(rows)>0):
        checkLimits(rows, event, c)

    conn.commit()
    conn.close()
```

Listagem de Código 5 - Verificação de alerta - ActLocallyGreenGrass

Através do identificador do dispositivo (presente no JSON recebido) é possível verificar se existem regras definidas para o dispositivo em questão. Caso se confirme é então feita a verificação da leitura. Esta verificação começa por identificar o valor lido para o sensor alvo do alerta. Depois de identificado o valor é necessário fazer a validação temporal. Se a data de leitura for maior que a data final ou menor que a data inicial o processo termina, caso contrário, verifica-se se o valor é maior que o máximo definido ou menor que o mínimo.

Quando se verifica um valor anómalo, é, de imediato, enviada a mensagem para a *cloud*. Esta é enviada para o tópico definido no JSON do dispositivo adicionado os valores "/{sensorAnómalo}/Exception" através da seguinte subscrição:

Tabela 24 - Configuração subscrição de envio de mensagens - ActLocallyGreenGrass

Fonte	Destino	Tópico
ActLocallyGreenGrass	cloud	"Abaco/Telemetry/#"

(Ex: "Abaco/Telemetry/Edge/ESP32/A01/temperature/Exception").

Depois de armazenadas e verificadas as leituras dos dispositivos, é necessário o envio destas para a *cloud*.

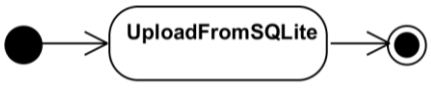


Figura 45 - Fluxo de dados AWS GreenGrass – Envio de leituras

A Figura 45 apresenta a função responsável pelo envio dos dados.

- **UploadFromSQLite** - Função responsável por recolher os dados armazenados e enviar os mesmo, de forma agrupada, para a *cloud.* É configurada como uma função *long lived* pois não tem um responsável pela sua ativação.

Apresenta uma única subscrição no sentido função, *cloud* através do tópico "Abaco/#". A definição de tempo de ativação é da responsabilidade do cliente. Caso não seja especificado o seu valor, é usado o valor predefinido de 100 segundos de intervalo.

```
1. def starter():
2.     while True:
3.         upload = getTimeValue()
4.         if is_connected("www.google.com"):
5.             getData()
6.         time.sleep(upload)
7.
8. starter()
```

Listagem de Código 6 - Função inicial UploadFromSQLite

A Listagem de Código 6 apresenta a função inicial chamada aquando da ativação do *lambda*. Depois desta é, numa primeira fase, verificada se existe algum valor para o tempo entre envio de mensagens.

```
1. def getTimeValue():
2.     conn = sqlite3.connect(sqlite_Time)
3.     c = conn.cursor()
4.     #Retreive data
5.     c.execute('SELECT time FROM uploadValue WHERE component = "gg_upload
   Batch"')
6.     value = c.fetchall()
7.     if not value:
8.         return 100
9.     upload = value[0][0]
10.       if upload is None:
11.           return 100
12.       return upload
```

Listagem de Código 7 - Estabelecer intervalo de tempo - UploadFromSQLite

Para que seja possível estabelecer o intervalo de tempo desejado pelo utilizador é necessário verificar se a base de dados *variables.db* contém, na tabela *uploadValue,* um dado de entrada com a descrição "gg_uploadBatch". Este identificador permite verificar se existe um valor desejado pelo cliente. Caso exista é então retornado, passando este a ser o intervalo entre chamadas.

```
1. def is_connected(hostname):
2.   try:
3.     host = socket.gethostbyname(hostname)
4.     # connect to the host
5.     s = socket.create_connection((host, 80), 2)
6.     return True
7.   except:
8.      pass
9.   return False
```

Listagem de Código 8 - Verificar ligação à rede

Depois de verificado este valor, é verificada se existe ligação à rede de forma a garantir o envio dos dados. A Listagem de Código 8 exibe a função *is_connected* que tenta fazer uma ligação, neste caso ao *google.com*, retornando o resultado da mesma.

Tendo ligação à rede é, então, realizada a componente principal do *lambda*, a recolha e envio dos dados para a *cloud*. Para isso, são recolhidos todos os valores guardados previamente no ficheiro *telemetry.db* e apagados do mesmo. Depois da recolha, estes são enviados para a *cloud* com o tópico "Abaco/GroupTelemetry" tendo, entre leituras, o caracter "@" a separar.

As três funções *lambda* apresentadas são as responsáveis pela receção, tratamento e envio de dados entre dispositivos e *cloud* passando pelo *gateway*. Além destas, foram também criadas duas funções extra:

- **ChangeLambdaVariables** – Recebendo um JSON igual ao apresentado na Listagem de Código 4, esta função é responsável pelo seu armazenamento no ficheiro *variables.db* na tabela *limits.* Antes do armazenamento, verifica se já existe alguma regra criada com o mesmo identificador de dispositivo e sensor, caso exista, esta é substituída com os novos valores.
  A sua subscrição apresenta o seguinte formato:

  Tabela 20 - Configuração subscrição definição de limites - ChangeLambdaVariables

Fonte	Destino	Tópico
IoT Cloud	changeLambdaVariables	"Abaco/Telemetry/UploadLimits"

- **ChangeUploadInterval** – Função responsável por armazenar no ficheiro *variables.db* (tabela *uploadValue*) o valor que define o tempo entre envio de dados. Recebe em JSON o valor identificado como *time* e armazena-o substituindo ou não o valor que poderá já ter sido introduzido. Este valor é armazenado em conjunto com a chave *gg_uploadBatch* de forma a ser identificado noutras funções que dele necessitem.
  A sua subscrição apresenta o seguinte formato:

  Tabela 21 - Configuração subscrição definição de valor de *upload*

Fonte	Destino	Tópico
IoT Cloud	changeLambdaVariables	'Abaco/Telemetry/UploadInterval"

H.3 AWS IoT

Este é o componente central de todo o sistema. Além da responsabilidade pela receção dos dados este componente é também onde é feita toda a configuração do sistema. É através deste

que é feita a gestão dos dispositivos e do próprio *gateway*, nomeadamente a definição de subscrições.

Alem destas funções, é também através do AWS IoT que é possível fazer o tratamento de dados aquando da receção, neste caso dos dados agrupados. Para isso foi criada uma regra denominada de saveToDynamoBatch.

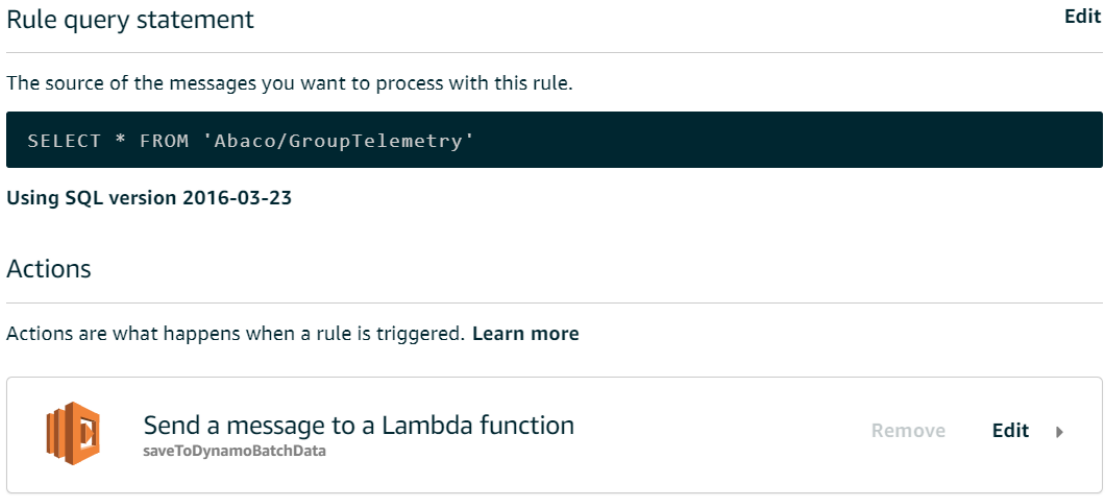


Figura 46 - Configuração saveToDynamoBatch

Como é possível observar através da Figura 46, esta regra é acionada sempre que existe algum tipo de mensagem no tópico específico "Abaco/GroupTelemetry". Acionando a regra, a mensagem é então redirecionada para a função *lambda* com o mesmo nome.

Esta função apresenta a seguinte ordem de execução:

```
1.  def lambda_handler(event, context):
2.      data = event['Data']
3.      array = data.split("@")
4.      getData(array)
```

Listagem de Código 9 - Função inicial - SavetoDynamoBatchData

Como é possível observar na, o *lambda* começa por fazer a separação das diferentes leituras através do caracter "@" introduzido pela função demonstrada na Secção H.2. Estando as leituras separadas e armazenadas na variável *array*, é chamada a função *getData(array)*.

```
1.  def getData(array):
2.      for value in array:
3.          if value:
4.              reading= json.loads(value.replace("'","\""))
5.              saveToDynamoDB(reading)
```

Listagem de Código 10 - Função getData - SaveToDynamoBatchData

Esta função (Listagem de Código 10) verifica se cada um dos objetos em *array* é valido. Caso o seja são então substituídos os caracteres ‘ por “ de forma a corresponder ao formato necessário no ambiente *lambda.* De seguida é chamada a função *savetoDynamoDB.*

```
1.  def saveToDynamoDB(reading):
2.      reading = decimalAux(json.loads(json.dumps(reading)))
3.      table.put_item(Item = {
4.          "DeviceID": reading["Device"]["deviceID"],
5.          "Date": reading["Date"],
6.          "payload": reading})
```

Listagem de Código 11 - Função SaveToDynamoDB - SaveToDynamoBatchData

Após o tratamento dos dados *float*, onde são transformados em *double* de forma a manter a sua composição, a função representada na Listagem de Código 11, guarda os valores de forma permanente no serviço AWS DynamoDB.

Para que seja possível esta transição de dados entre AWS GreenGrass, AWS IoT e AWS DynamoDB é necessário a criação de diversos *roles* que possuam na sua composição as políticas de acesso exigidas para as operações. Para isso, foi usado o serviço AWS IAM que permite a criação e atribuição de *Roles* com determinadas políticas para os diversos componentes. Além da praticidade e facilidade de uso deste serviço, apresenta-se também como um serviço grátis e que invariavelmente acrescenta valor e segurança ao sistema no geral.

H.4 AWS DynamoDB e AWS Gateway API

A criação da tabela no serviço AWS DynamoDB é feita de forma bastante simples. A sua composição apresenta três campos de entrada:

- *DeviceID* – Como o nome indica, é o identificador do dispositivo que emitiu os dados. Esta é considerada a chave primária de cada registo da tabela;
- *Date* – Referencia temporal, no formato *Unix*[7], guardado como um inteiro e que permite a ordenação dos registos.
- *Payload* – Neste campo é incluída toda a mensagem armazenada no *gateway* tendo como forma o JSON apresentado na Listagem de Código 1.

Assumindo que os registos possuem todos este formato, é esperado que cada um possua um tamanho de aproximadamente 365 *bytes.* A tabela que se segue representa os custos que podem vir a ser associados a este consumo tendo em consideração o tamanho de cada mensagem.

[7] Referência temporal em segundos desde 1 de janeiro de 1970 (Ex: 1559209191 = 30/05/2019 9:30)

Tabela 25 - Custos AWS DynamoDB específico

Quantidade Registos	Tamanho (bytes)	Preço
1	365	0
2 739 726	1 GB	0.306$
68 493 150	25 GB	Grátis por mês

Atraves da sua análise e tendo em conta que os primeiros 25 GB (*gigabites*[8]) de dados por mês são gratuitos, é possivel armazenar cerca de 68,493,150 mensagens. Este valor permite o envio e armazenamento de 2,283,105 mensagens diárias. Caso sejam ultrapassados estes valores, é então cobrado cerca de 27 cêntimos por GB.

Depois de feita a configuração e armazenamento dos dados no AWS DynamoDB é necessário criar forma de aceder aos dados sem necessitar de usar a plataforma oferecida pela *Amazon*. Para isso é usada a ferramenta AWS API *Gateway*.

Após a criação de um novo *role* com autorização para aceder à tabela criada, foi então criada uma nova API. A sua configuração foi relativamente simples, sendo criados cinco pedidos que atuam diretamenta na base de dados (Figura 47).

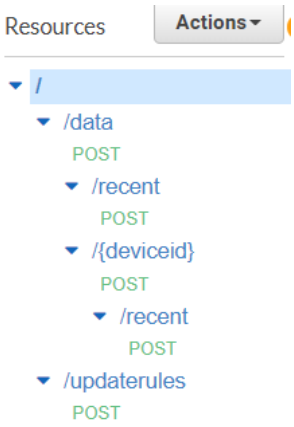


Figura 47 - Configuração AWS API Gateway

[8] 1 GB = 1,000,000,000 bytes

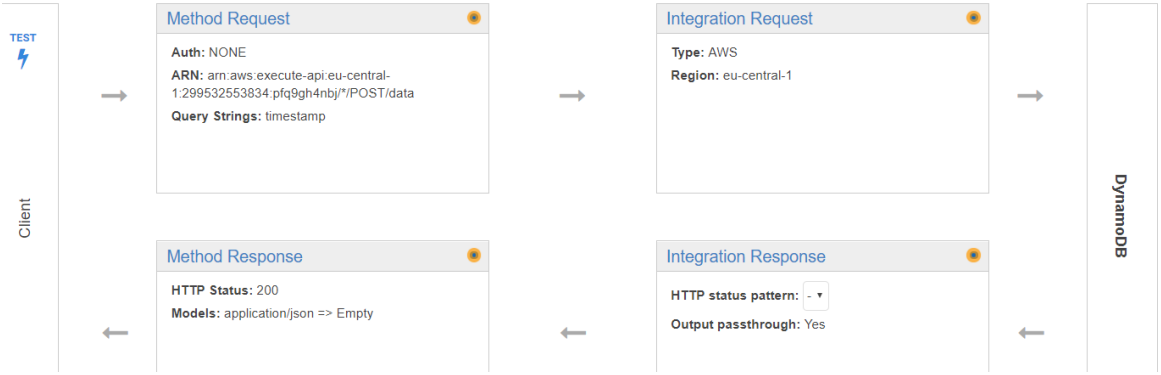


Figura 48 - Estrutura de pedido no *AWS API Gateway*

Como se pode observar na Figura 48, cada método é composto por duas ações de pedido e duas ações de resposta:

- *Method Request* – Área onde é feita a configuração da interface do pedido. Inclui-se a autorização e gestão de dados de entrada;
- *Integration Request* – Área onde é especificada a forma como o *API Gateway* vai interagir com o alvo do seu pedido;
- *Integration Response* – Depois de realizado o pedido, esta secção é encarregue de especificar a forma como a resposta deve ser mapeada;
- *Method Response* – Área onde se permite especificar os códigos de resposta suportados para o pedido em específico.

Todos os métodos criados, que são de seguidas apresentados, seguem a estrutura referida na Figura 48, diferindo apenas na forma como é configurado e realizado o seu pedido.

- **/Data** – Método *POST* que permite obter todos os registos da base de dados, independentemente do dispositivo ou data em que foram inseridos. A sua configuração, representada na Figura 49 é relativamente simples:

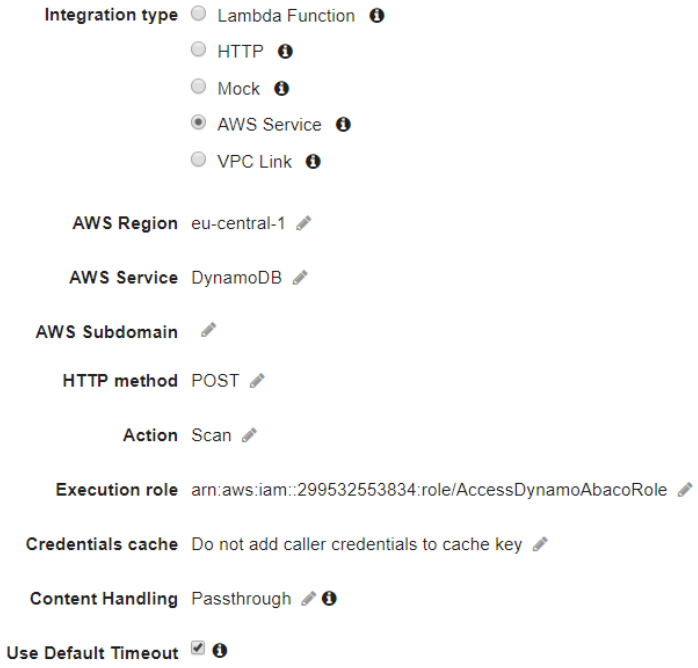


Figura 49 – Configuração Método /data

Começa-se por definir o serviço a usar (AWS Service*)* definindo qual e onde se encontra o mesmo (*eu-central-1, DynamoDB*). De seguida define-se o tipo de pedido e ação que este método vai realizar à tabela (*POST, Scan*). A configuração termina com a atribuição do *role* previamente criado e pela criação do *template* de dados a obter. Para este ultimo passo é necessário criar o mesmo através do componente *Mapping Templates* assumindo a seguinte forma:

```
1. {
2.     "TableName": "AbacoTelemetry_F",
3.     "ReturnConsumedCapacity": "TOTAL"
4. }
```

Listagem de Código 12 - Mapping Template - /data

Desta forma é possivel obter todos os registos na tabela *AbacoTelemetry_F*, incluindo na sua resposta um campo que totaliza o numero de leituras enviadas. Após a realização do pedido, é recebida uma resposta que apresenta o *template* da mensagem com o formato proveniente do *DynamoDB*. Este formato não é de fácil compreensão e de forma a torná-lo mais simples e intuitivo é necessário mapear esses dados para um formato à nossa escolha. Para isso, é novamente criado um *mapping template* representado pela Listagem de código que se segue:

```
#set($inputRoot = $input.path('$'))
[
#foreach($elem in $inputRoot.Items)
{
    "Date": $elem.Date.N,
    "DeviceID": "$elem.DeviceID.S",
    "payload" : {
        "Date": $elem.payload.M.Date.N,
        "DeviceID": "$elem.payload.M.DeviceID.S",
        "Topic" : "$elem.payload.M.Topic.S",
        "Device" : {
           "Active": $elem.payload.M.Device.M.active.BOOL,
           "CompanyID": "$elem.payload.M.Device.M.companyID.S",
           "DeviceID" : "$elem.payload.M.Device.M.deviceID.S",
           "DeviceType" : "$elem.payload.M.Device.M.devicetype.S",
           "LocalID": "$elem.payload.M.Device.M.localID.S"
        },
        "System" : {
            "battery": "$elem.payload.M.System.M.battery.S",
            "free_ram" : $elem.payload.M.System.M.free_ram.N,
            "uptime": $elem.payload.M.System.M.uptime.N
        },
        "Telemetry" : [
        #foreach($aa in $elem.payload.M.Telemetry.L)
            {
            "sensor":"$aa.M.sensor.S",
            "unit": "$aa.M.unit.S",
            "value": $aa.M.value.N
            }#if($foreach.hasNext),#end
            #end]
    }
}#if($foreach.hasNext),#end

#end]
```

Listagem de Código 13 - Mapping Template resposta /data

Importa referir que este mapeamento é repetido em todos os pedidos realizados através da *API Gateway* de forma a poderem ser tratados, mais tarde, pela *Web app* de maneira igual.

- **/Data/Recent –** Extensão do pedido anterior, este, permite receber todos os dados guardados na base de dados a partir de uma terminda linha temporal.
  Para que seja possivel esta ação, aquando da chamada da função deve ser enviado no corpo do pedido um JSON com o seguinte formato:

```
{
    "timestamp" : 1556702102
}
```

Listagem de Código 14 - JSON pedido /data/recent

Caso o pedido apresente este conteúdo (Listagem de Código 14) é então possível iniciar o pedido sem si. Para isso é necessário referir, através do *mapping template*, a necessidade de restringir os dados a receber. A Listagem de Código 15 representa o formato desse pedido:

```
{
    "TableName": "AbacoTelemetry_F",
    "ExpressionAttributeNames": {
      "#c" : "Date"
   },
    "ExpressionAttributeValues":
        {
        ":val": {"N":"$input.path('$.timestamp')"}
        },
    "FilterExpression": "#c >= :val"
}
```

Listagem de Código 15 - Formato pedido /data/recent

Caso a tabela apresente valores com o parâmetro *Date* maior que a inserida pelo utilizador, estes são então enviados na resposta do pedido, após o seu mapeamento para o formato já apresentado.

- **/Data/{DeviceId} –** Método que filtra os resultados por ID do dispositivo. O identificador do dispositivo a restringir é incluído no URL do pedido. O pedido a realizar difere dos anteriores já mencionados, sendo, neste caso, usado a ação de *Query*. Mais uma vez, a restrição do pedido acontece no *mapping template* que apresenta a seguinte forma:
  A resposta, em caso de sucesso, é de seguida mapeada e enviada.

```
{
    "TableName": "AbacoTelemetry_F",
    "ExpressionAttributeValues":
        {
        ":val": {"S":"$input.params('deviceid')"}
        },
    "KeyConditionExpression": "DeviceID = :val"

}
```

Listagem de Código 16 - Pedido /data/{deviceid}

- **/Data/{DeviceId}/Recent –** Método que estende o anterior e que pemite a filtragem de dados por identificador e por data, tanto de fim como de início. As variáveis que delimitam as restrições temporárias são enviadas através do corpo do pedido (Listagem de Código 17) enquanto que o identificador do dispositivo é inserido no próprio URL.

```
{
    "valBegin" : 1558948502,
    "valEnd" : 1559300650
}
```

Listagem de Código 17 - JSON Pedido /data/{deviceid}/recent

O pedido a realizar necessita de obter todas as três variáveis de forma a restringir os resultados. Para isso, é feito o mapeamento, demonstrado na Listagem de Código 18. No caso de o pedido obter resultados, é então feito o mapeamento e enviadas as respostas.

```
{
    "TableName": "AbacoTelemetry_F",
    "ExpressionAttributeNames": {
      "#c" : "Date"
   },
    "ExpressionAttributeValues":
        {
        ":device" : {"S":"$input.params('deviceid')"},
        ":valBegin": {"N":"$input.path('$.valBegin')"},
        ":valEnd":{"N":"$input.path('$.valEnd')"}
        },
    "FilterExpression": "DeviceID = :device and #c BETWEEN :valBegin AND :valEnd"
}
```

Listagem de Código 18 - Mapeamento /data/{deviceid}/recent

- **/UpdateRules** – Este pedido é específico para criação de regras no dispositivo *gateway*. A sua execução difere das anteriores pois não necessita do mapeamento de dados. O alvo deste pedido é também diferente. Enquanto que os já apresentados tem como alvo uma tabela do serviço *DynamoDB*, este pedido tem como alvo final, despoletar uma função *lambda*. Porém a sua configuração é semelhante, diferindo apenas o serviço a usar e o *role*. Não havendo mapeamento, esta função foi configurada de forma a reencaminhar o corpo do pedido que recebe para a função *lambda*.

```
{
  "Topic": "Abaco/Telemetry/UpdateLimits",
  "payload": {
    "deviceId": "ESP32_A01",
    "sensor": "temperature",
    "min": 7,
    "max": 15,
    "begin": 1553770112,
    "end": 1585392512
  }
}
```

Listagem de Código 19 - Pedido /updaterules

A Listagem de Código 19 representa esse mesmo corpo. Esta mensagem é depois reencaminhada para a função *lamda APIGreenGrassConnector.* Esta é responsável por publicar no AWS IoT a mensagem definida como ***payload*** no tópico ***Topic***. Por sua vez, a mensagem é enviada para o *gateway* despoletando as funções associadas ao tópico. Através deste fluxo de dados, é possível para o cliente definir novas regras ou alterar o tempo de *upload* através das funções demonstradas anteriormente.

H.5 Web App

A criação dos componentes previamente apresentados necessita de ter um componente alvo, que possa usar as funcionalidades disponibilizadas. Para isso, e tendo em conta o *design* apresentado, foi então desenvolvida uma *web app* que conseguisse responder a todos os requisitos necessários tendo em conta os fatores expostos na secção 4.7. Importante realçar que este componente do projeto foi realizado em conjunto com o meu colega de estágio Miguel Ribeiro com o nº mecanográfico 1141272 pelo que poderá haver secções onde serão apresentados os mesmos dados.

Processo de desenvolvimento

O processo de desenvolvimento iniciou-se com a criação de *mockups* relativos às interfaces necessárias. Este processo tinha como objetivo permitir desenvolver e aperfeiçoar o modelo de dados e também a apresentação aos responsáveis pelo projeto de modo a obter a sua validação (*Mockups* apresentados no Anexo C).

O desenvolvimento iniciou-se com base num projeto fornecido pela empresa. Este projeto apresentava, já, algumas bases relativas a autorização e autenticação de utilizadores. Depois de examinado o código em si foi iniciado o processo de limpeza do mesmo, sendo retiradas partes que não interessariam. Deste passo em diante foi então iniciado o processo do desenvolvimento, tendo sempre em conta os requisitos abordados.

Autenticação e Autorização

A autenticação e autorização de utilizadores na *web app* é realizada através do uso de *ASP.NET Identity*. Esta é uma ferramenta *Microsoft* que permite a criação e gestão de utilizadores e gestão de acessos dos mesmos.

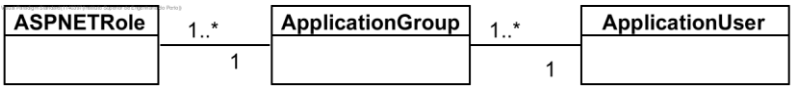


Figura 50 - Diagrama classes autorização e autenticação

Como é possível observar na Figura 50 cada utilizador do sistema é considerado um *ApplicationUser* e tem associado a si um ou mais *ApplicationGroups*. Já estes tem associados um ou mais *ASPNETRoles*, que são responsáveis pela autorização durante o uso do serviço.

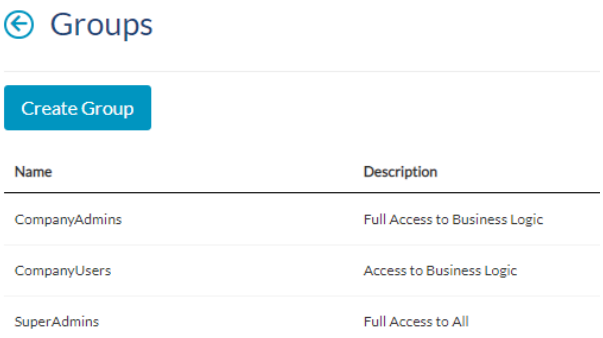


Figura 51 - *ApplicationGroups* da solução

A Figura 51 representa os grupos criados por defeito na *web app* demonstrada. A cada grupo está associada um *ASPNETRole* com o mesmo nome do grupo:

- *SuperAdmin* – *role* com maior quantidade de permissões. Tem acesso a todos os dados e pode efetuar qualquer operação CRUD *(Create, read, update, delete*). Único tipo de utilizador que pode criar outros utilizadores de sistema;
- *CompanyAdmin* – utilizador responsável pelo *CRUD* dos dados relativos à empresa a que este está associado;
- *CompanyUser* – utilizador que apenas pode consultar dados relativos à empresa a que está associado.

Sendo assim, aquando da criação de um novo utilizador, é necessário referir que tipo de *role* é que este vai possuir e qual a empresa em que os vai poder aplicar. A Figura 52 apresenta a interface completa do diálogo de criação de utilizador.

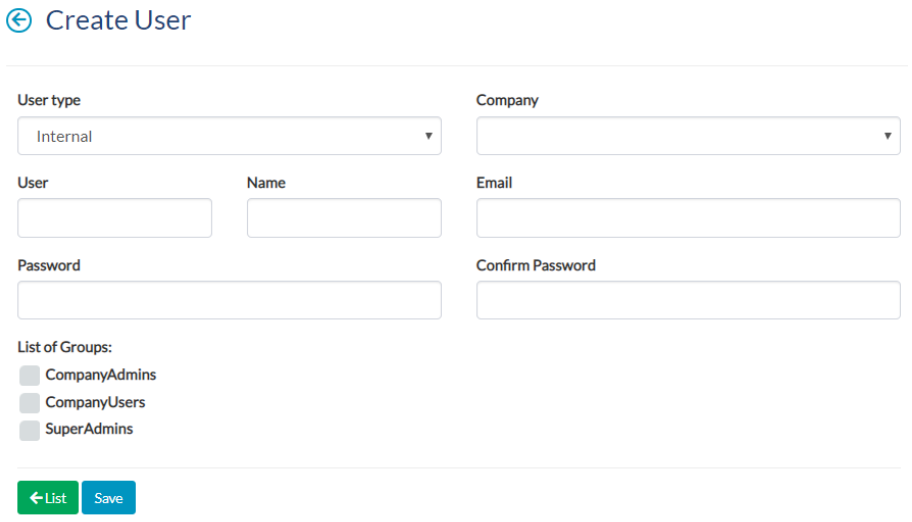


Figura 52 - Interface criação de utilizador

Estando os utilizadores associados a *roles* é então possível fazer a restrição de uso do serviço através da mesma. Usando a notação `[Authorize]` é possível restringir que tipo de utilizador pode ter acesso a funções especificas.

O uso desta ferramenta permite também a restrição a nível visual das interfaces. Apenas os utilizadores que estejam associados ao grupo *SuperAdmin* tem acesso ao menu de sistema que permite a execução de diversas operações que não estão relacionadas com o modelo de negócio (Figura 53).

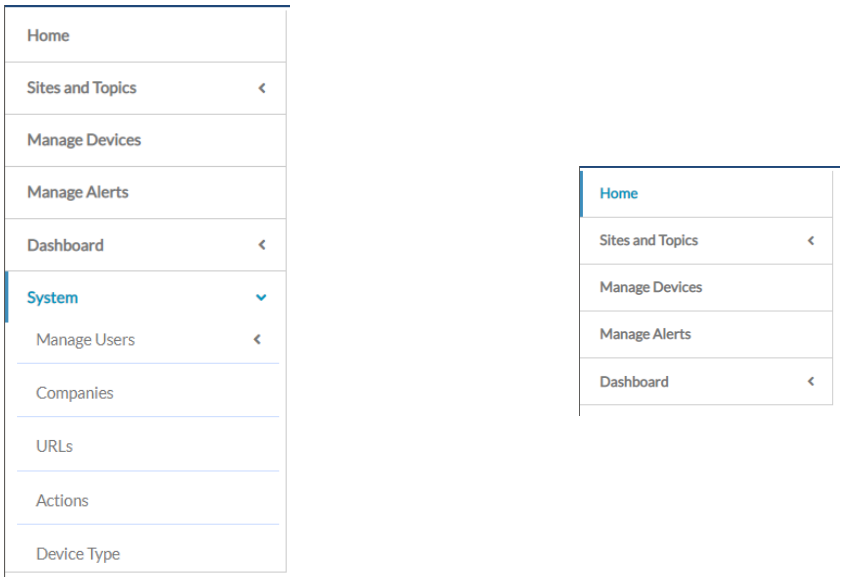


Figura 53 - Comparação Menus entre *SuperAdmin* e outros.

H.6 Controllers e Views

Seguindo o modelo MVC e usando *Entity Framework* foi possível criar as *views* e *controllers* necessários assim como o mapeamento destes à base de dados.

As *views* foram todas alteradas de forma a serem mais intuitivas e fáceis de usar. Estas alterações tiveram como base a ferramenta *AdminLTE* que, construída sobre *bootstrap*, permite a criação de uma interface responsiva, reutilizável e mais intuitiva através de diversos componentes. O uso de *JQuery* e *Javascript* é também importante no desenvolvimento da interface. O uso destas duas linguagens, em conformidade com o HTML das *views*, permite obter uma interface mais fluida sem necessidade de atualizações constantes à página para apresentação de novos dados.

De forma a tentar exibir mais e melhor informação ao utilizador da *webapp* foram também desenvolvidas diversos *ViewModels* que permitem uma troca de dados entre *controller* e *views mais completa e eficaz.*

Alterando as *views* é também necessário adaptar os *controllers*. Estes sofreram algumas alterações não só a nível de ligação com a interface, mas também alterações relacionadas com autorização de utilizadores.

Todas os *controllers* criados foram também alterados relativamente à sua herança. Previamente estes herdavam a classe *Controller* sendo esta alterada para *BaseController*. Esta

última permite ao utilizador ter uma interface com a linguagem a que o seu *browser* está associada. Através dos *cookies* é possível obter o idioma que o utilizar usa e alterar os recursos a usar, mantendo a sua preferência.

H.7 Base de dados

A tecnologia escolhida para armazenamento de dados foi *SQL server*. Esta, pela sua capacidade e integração com as tecnologias escolhidas mostrou-se como a escolha óbvia para este papel. O mapeamento dos modelos com a base de dados, como já referido, é da responsabilidade da tecnologia *Entity Framework*, ferramenta esta que se apresenta como uma parte integrante da *framework .NET.* Aliando esta tecnologia com a metodologia *Code First Migrations* foi-nos possível desenvolver os modelos de negócio de forma dinâmica e simples sem a preocupação de desenvolver e manter código *SQL* para instanciação e gestão da base de dados.

A capacidade de triagem de dados é também uma mais valia desta tecnologia. Através do uso de *queries* foi-nos possível identificar determinados tipos de dados, facilitando o acesso a estes e aos dados a este associados.

De forma a controlar os dados e associações criadas foi também utilizada a ferramenta *SQL Server management* que permite a ligação à mesma e consulta e alteração de dados, opção importante aquando da realização de testes.

H.8 Ligações externas

Para que a *web app* acrescente valor a todo o sistema é necessário que esta apresenta alguns dados armazenados noutros locais que não a base de dados desta, nomeadamente no AWS DynamoDB. Para isso foi então criada uma classe de ligação a esta, que, através do URL da API criada permite aceder de forma relativamente simples a todos os dados.

De forma a manter o código fácil de compreender, eficaz e eficiente foi utilizada a ferramenta *RestSharp* [64] desenvolvida para conceber pedidos REST.

Para cada pedido apresentado na Secção H.4 foi então desenvolvida a função correspondente que permitiria a sua chamada. O pedido é feito de forma relativamente simples começando-se por definir o cliente e pedido como mostra a Listagem de Código 20 referente ao pedido **/data/{deviceid}/recente.** Realço a inclusão do identificador do dispositivo no URL do pedido a realizar, só assim será possível obter uma resposta correta da API.

```
1. //Setup
2.    string url = getAPIGatewayEndPoint(db);
3.    var client = new RestClient(url);
4.    var request = new RestRequest("data/" + DeviceId + "/recent", Method.POST);
```

Listagem de Código 20 - Configuração pedido RestSharp

De seguida cria-se os cabeçalhos e o corpo do pedido como representado na Listagem de Código 21. Este pedido necessita de obter dados de data de início e de fim no JSON a enviar, dados estes que são criados e armazenados na variável *postData.*

```
1. // Headers
2.   request.AddHeader("Accept", "application/json");
3.   request.AddHeader("Content-Type", "application/json");
4.
5. // Request Body
6.    dynamic data = new JObject();
7.    int epochBegin = (int)(dateBegin - new DateTime(1970, 1, 1)).TotalSeconds;
8.    int epochEnd = (int)(dateEnd - new DateTime(1970, 1, 1)).TotalSeconds;
9.    data.valBegin = epochBegin;
10.   data.valEnd = epochEnd;
11.   string postData = ((JObject)data).ToString();
12.   request.AddParameter("undefined", postData, ParameterType.RequestBody);
```

Listagem de Código 21 - Definição de cabeçalhos e corpo de pedido

Tendo estes componentes definidos e populados é então realizado o pedido em si e analisada a resposta. Caso o pedido tenha tido sucesso é então retornado o código *200* e é recebido o *JSON* de resposta. Caso contrário é mostrado o erro ao utilizador. Este processo é demonstrado na Listagem de Código 22.

```
1.  // Execute Request
2.     IRestResponse response = client.Execute(request);
3.     var content = response.Content;
4.
5.     if (response.StatusCode == HttpStatusCode.OK)
6.     {
7.       var jsonObject = JArray.Parse(content);
8.       return jsonObject;
9.     }
10.     else
11.    {
12.    if (!string.IsNullOrEmpty(content) && !string.IsNullOrWhiteSpace(content))

13.    {
14.      var jsonObject = JArray.Parse(content);
15.      var errorMessage = (string)jsonObject["message"];
16.    }
17. }
```

Listagem de Código 22 - Realização do pedido e tratamento da respostaListagem

Este processo repete-se para todos os pedidos podendo ou não, estes, possuir definição de corpo de pedido.

H.9 Interface gráfica

A interface gráfica era um componente bastante importante para que a *web app* pudesse ter a utilidade e sucesso que dela era esperada. Para isso, o seu desenvolvimento foi sempre realizado tentando manter a simplicidade e praticidade da mesma. Foi com esta ideologia que foram desenvolvidas as várias páginas de *mockups* já apresentadas. Estas, possuem bastantes tabelas e este componente acabou por ser bastante importante na globalidade da *web app*. Através do uso de componentes *DataTables* foi possível criar uma interface simples, com informação adequada e completa sem aumentar a complexidade da mesma. Além disto, as interfaces foram alteradas de forma a usar *div* com a class *row* sendo que cada uma destas possui uma largura com 12 unidades. A gestão destas unidades foi feita pelos desenvolvedores conforme a necessidade da página ou elemento.

Este desenvolvimento teve também em conta os requisitos não funcionais apresentados na secção 4.7. A portabilidade do sistema foi um fator importante e tido em consideração. Para isso foram realizados constantes testes à responsividade das interfaces de forma a garantir todas as funcionalidades da mesma. As figuras que se seguem representam alguns desses testes e adaptações realizados.

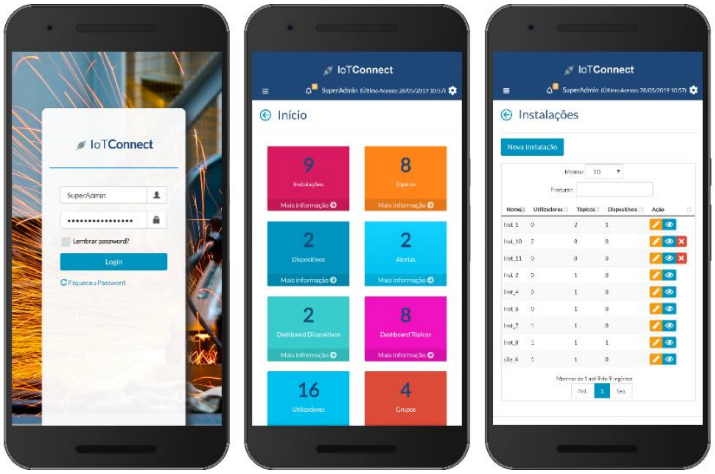


Figura 54 – Demonstração do uso da *web app* em dispositivos móveis

De forma a melhor demonstrar o trabalho efetuado para obter este resultado, em Anexo 0, apresentam-se algumas das interfaces das páginas mais relevantes para o sistema.

H.10 RGPD e Logs

A segurança e registo de ações foi também um ponto de interesse durante o desenvolvimento do sistema. De modo a obter e armazenar alguns dados importantes, todos os modelos criados herdam a classe *Audit*.

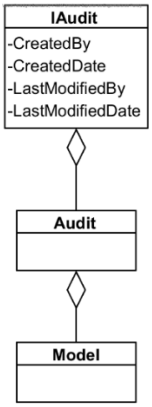


Figura 55 - Classe *Audit*

Como é possível notar na Figura 55, é desta forma, possível registar a data e responsável de criação assim como data e responsável pela última alteração sendo estes valores armazenadas de seguida.

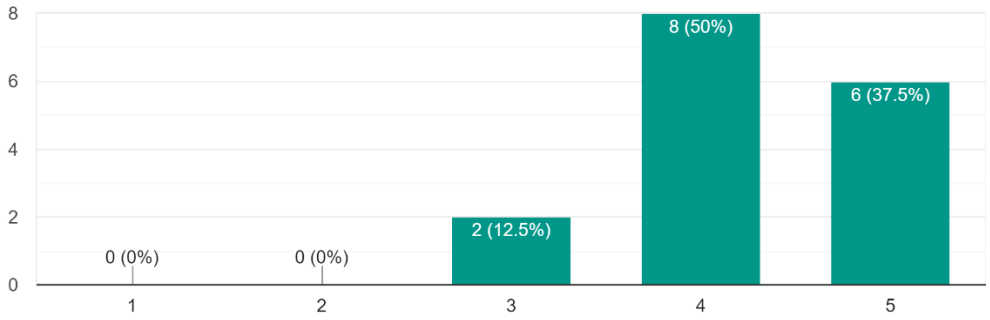
Além deste registo, o sistema apresenta também a classe *RGPDLogon* que permite efetuar o registo de acessos ao sistema. Aquando de um novo *login/logout* é feito um registo na base de dados onde é identificado o tipo de ação (*login ou logout*), o identificador do utilizador e o seu nome de utilizador, a data do momento da ação e, finalmente, o IP e *port* de acesso. Desta forma é possível manter algum registo de atividades dentro da própria *web app*.

Anexo I – Respostas Inquérito

I.1 Questões de usabilidade

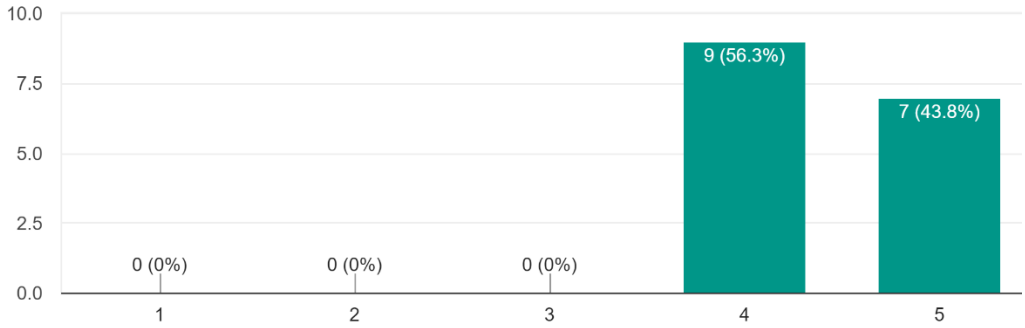
Considerou o sistema suficientemente intuitivo para seguir o guião de forma eficiente?

16 responses



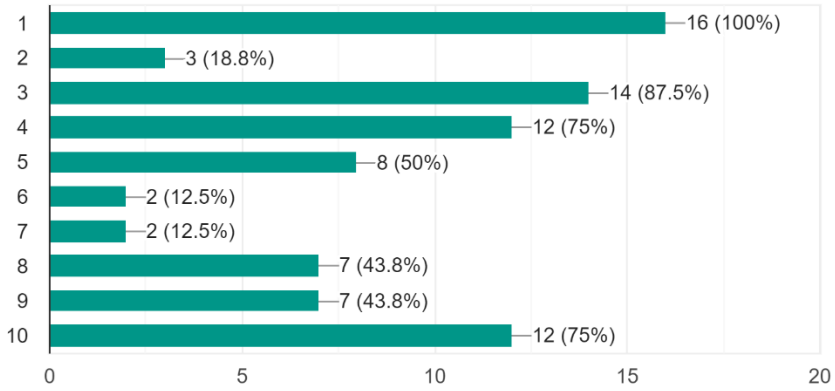
Considerou a navegação entre páginas e operações prática e intuitiva?

16 responses



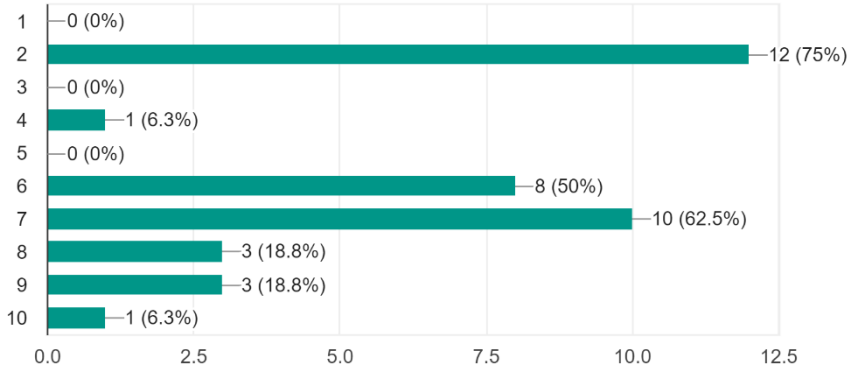
Especifique a(s) operação(s) mais intuitivas.

16 responses



Especifique a(s) operação(s) menos intuitivas.

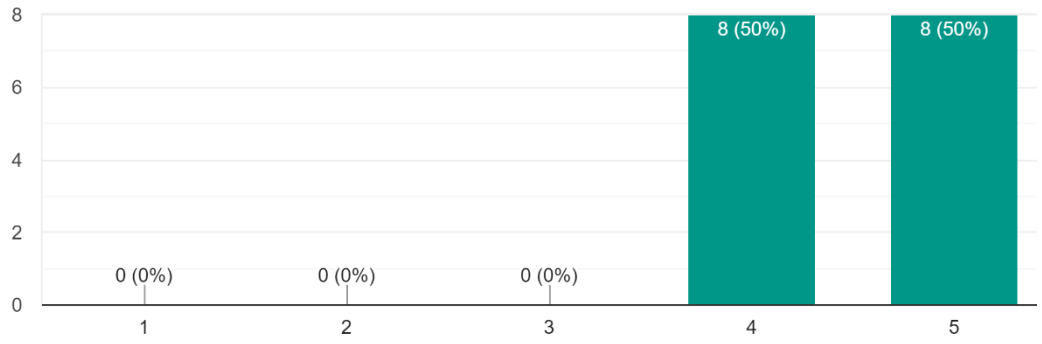
16 responses



I.2 Questões de satisfação

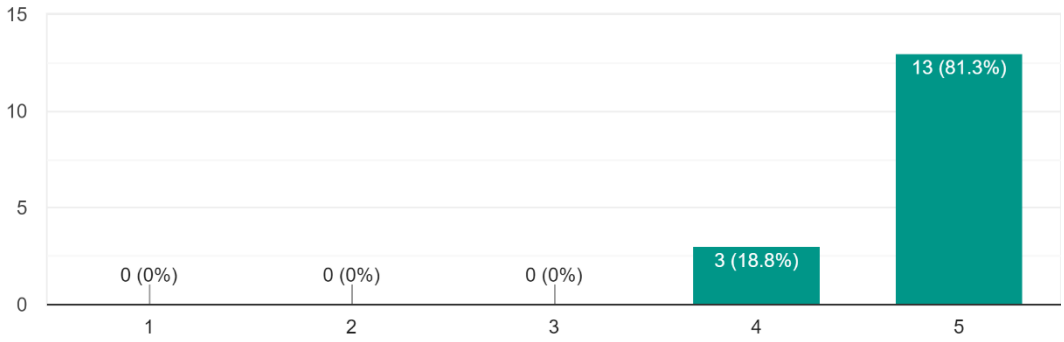
Considerou o tipo/tamanho de letra legível?

16 responses



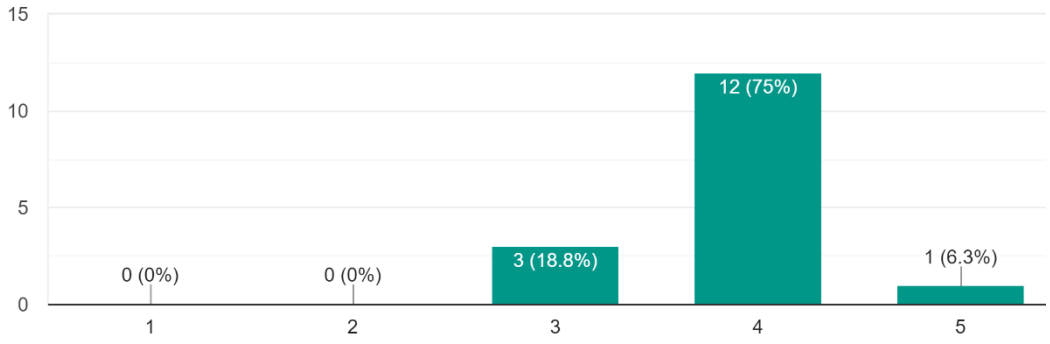
Considerou o esquema de cores adequado?

16 responses



Considerou o tempo de resposta das operações tolerável?

16 responses



Em caso de uso estaria satisfeito com o sistema?

16 responses