INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Sistema de gestão de micro-redes através do protocolo IEC61850

JOÃO PAULO PINTO MOREIRA
Outubro de 2020

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Sistema de gestão de micro-redes através do protocolo IEC61850

João Paulo Pinto Moreira

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

Orientador: Paulo Gandra Sousa

Porto, 12 de outubro de 2020

Resumo

Com o decorrer dos anos, os sistemas de gestão de redes têm vindo a evoluir junto com a rede de energia, aumentando cada vez mais a complexidade na gestão e manutenção dos mesmos. Nos tempos mais recentes tem existido um aumento acentuado em recursos de energia, distribuída em redes de baixa tensão, como por exemplo: sistemas de micro e mini geração fotovoltaica, sistemas de armazenamento de baterias ou estações de carregamento de veículos elétricos, estes com características muito próprias de ligação com a rede geral de energia, podendo mesmo, em certos casos, serem geridos de forma independente.

Aqui surge o conceito de micro-redes, conceito que responde a estes novos casos de uso cada vez mais comuns através da operação de redes em pequena escala.

Este conceito é estudado nesta dissertação, realizando uma comparação entre o estado atual da rede e um conceito de micro-redes, sendo usado o protocolo IEC61850 como via para a realização de um sistema de gestão de micro-redes que seja flexível e capaz de ser integrado com vários sistemas da empresa, como por exemplo, o sistema de gestão de rede SCADA.

Palavras-Chave:

Gestão de redes centralizada

Micro-redes

Protocolo de comunicação IEC61850

Abstract

Over the years, network management systems have evolved along with the power network, increasing their complexity in their management and maintenance. In more recent times there has been a major increase in energy resources, low voltage distributed networks, such as: micro and mini photovoltaic generation systems, battery storage systems or charging stations for electric vehicles, This characteristics can be very singular on this type of connection with the general energy network and even in some cases they can be managed independently.

Then, the concept of micro-networks surges, which is a concept that responds to these new, increasingly common use cases through the operation of small-scale network management.

This dissertation targets the comparison between the current state of the network and a concept of micro-networks, using the IEC61850 protocol as a way to create a micro-network management system that is flexible and capable of being integrated on company systems, such as the SCADA network management system.

Keywords:

Central grid management system

Microgrids

IEC61860 communication protocol

Agradecimentos

Em primeiro lugar, quero agradecer à minha família e à minha namorada que sempre me apoiou em tudo o que foi necessário. Ao meu orientador Dr. Paulo Sousa Gandra, pela disponibilidade e compreensão na orientação que prestou ao longo da elaboração do presente documento e, por fim, aos meus amigos.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ADMS	*Advanced document management system*
DMS	*Document management system*
IED	*Intelligent electronic device*
JSON	*Javascript object notation*
MMS	*Manufactoring message specification*
OMS	*Order management system*
SCADA	Sistema de supervisão e aquisição de dados
SCL	*Substation Configuration language*
SQL	*Structured query language*
SV	*Sample variable*
TCP	*Transmission control protocol*
UDP	*User datagram protocol*
URL	*Uniform resource locator*
XML	*Extensible markup language*

1 Introdução

1.1 Apresentação da Empresa

A Efacec é uma empresa portuguesa com quase 70 anos de existência e presença em mais de 65 países, sendo que se foca, principalmente, nas áreas da energia, mobilidade e automação.

Primeiramente, no âmbito da energia, a empresa disponibiliza transformadores, sistemas de aparelhagem elétrica, bem como vários serviços de automação.

Já na parte da mobilidade elétrica, é de sublinhar o desenho e a assemblagem de carregadores elétricos.

Por último, quanto à automação, realça-se a aposta na investigação do desenvolvimento de soluções para a produção, transmissão, distribuição e utilização de energia elétrica, com especial destaque no domínio das Redes Inteligentes. Esta divisão, desde a sua génese até aos dias de hoje, conta com cerca de 70 investigadores/técnicos a tempo inteiro afetos às seguintes áreas tecnológicas:

- Automação de Sistemas:
    - Investigação industrial;
    - Desenvolvimento de produtos SCADA/DMS/OMS e *Demand Response*;
    - Desenvolvimento de produtos de controlo, automação e operação;
- Inversores Solares;
- Sistemas de armazenamento elétrico;
- Projetos espaciais.

Por fim, é importante salientar que a Efacec se encontra dividida em diversas unidades de negócio com o propósito de proporcionar, a cada uma destas, a oportunidade de se poderem focar inteiramente na sua respetiva área de negócio (Efacec, 2020).

1.2 Enquadramento

Com a crescente integração de recursos de energia, distribuída em redes de baixa tensão, como por exemplo: sistemas de micro e mini geração fotovoltaica, sistemas de armazenamento de baterias ou estações de carregamento de veículos elétricos, surge a necessidade de desenvolver soluções que permitam uma transição sustentável dos sistemas elétricos de energia, tendo em vista a sua descarbonização e a sua excelência operacional.

Atualmente, as micro-redes constituem uma das soluções em crescimento, sendo que esta é vista como uma forma de garantir uma gestão integrada destes recursos de forma automática e cómoda, permitindo altas partilhas de energia renováveis, bem como uma elevada qualidade de serviço e energia (Efacec, 2017).

Embora a empresa, no momento, já possua, no seu portefólio de produtos, soluções para a gestão de redes elétricas, estas quando são aplicadas a uma micro-rede podem não responder de forma eficaz, pois estes dispositivos mais "*smart*" respondem através de protocolos não explorados. Estas comunicações também fogem da norma empresarial, uma vez que estão espalhados por vários locais, sendo assim importante a segurança destas comunicações.

Este será o sistema utilizado para a prova de conceito, sendo fulcral o estudo de uma forma de criar este novo componente de maneira mais dinâmica e com capacidade de escalabilidade, podendo, assim, ser replicada à *posteriori* por outros componentes, que venham a ser desenvolvidos no futuro.

1.3 Motivação

Devido a uma maior integração de recursos de energia distribuída em redes de baixa tensão, surge a necessidade da criação de uma aplicação que seja capaz de centralizar todos os recursos associados a uma micro-rede.

Como principal motivação, emerge a possibilidade de acrescentar ao portefólio atual da empresa a capacidade de comunicar com estes novos dispositivos, usando o protocolo IEC61850 para as suas diversas soluções internas e a potencial segregação por áreas dos vários dispositivos.

A empresa, atualmente, não possui a capacidade de ter um produto que consiga de forma centralizada servir um número variado de clientes, sendo assim também parte integrante desta dissertação o estudo de uma solução que suprima este problema e sirva de ponto de partida para o futuro.

1.4 Objetivos

A realização deste trabalho tem vários propósitos, tanto a curto como a longo prazo, para o desenvolvimento da empresa, no sentido de ir ao encontro das rápidas mudanças que se verificam no mercado da energia.

Deste modo, a presente dissertação tem como finalidade a obtenção da capacidade de passar a poder falar com dispositivos através do protocolo IEC61850 e usufruir de todas as vantagens adjacentes a estes, realçando que, de momento, tal não é possível em nenhum outro produto. Esta solução terá de comunicar com potenciais milhares de dispositivos e ser capaz de os separar por redes distintas, tendo sempre como foco a segurança nestas comunicações.

Sendo assim, pretende-se com a realização desta dissertação, criar um sistema *midleware* que seja capaz de gerir múltiplas micro-redes com aparelhos IEC61850 associadas a vários clientes. Para tal, pretende-se realizar um estudo de quais a melhores tecnologias que podem garantir a sua implementação, tendo sempre em conta a sua manutenção e escalabilidade a longo prazo.

1.5 Planeamento

Efetivamente, o planeamento do desenvolvimento da dissertação será dividido em 5 fases.

Numa primeira fase, será analisado qual o problema que esta dissertação pretende resolver, com uma análise do contexto em que esta se encontra.

Posteriormente, numa segunda fase, será realizado um estudo atual do mercado existente e o que este tem para oferecer em termos de aplicações que consigam resolver o problema descrito; será realizada, também, uma análise da viabilidade da implementação do mesmo, bem como quais as principais vantagens e fragilidades que este sistema poderá encontrar.

Numa terceira fase, será desenvolvida uma proposta arquitetural, que responda ao problema descrito, desenvolvendo todos os documentos necessários para a sua manutenção e escalabilidade.

Adicionalmente, a quarta fase consistirá na implementação de tudo o que foi descrito na fase de desenho (terceira fase).

Por fim, na última parte, será necessário olhar para o que foi realizado e fazer uma avaliação imparcial do projeto, o que foi possível de retirar do mesmo, se esta solução corresponde aos objetivos iniciais e quais os próximos passos que esta aplicação poderá tomar.

Para a realização destes processos, será adotada uma metodologia ágil, que permita à empresa ter um conhecimento mais rápido do que se encontra a ser desenvolvido, podendo, assim, realizar alguma reformulação, caso necessário, dentro de um menor período de tempo e de uma forma menos dolorosa.

Também serão realizadas reuniões periódicas com o orientador, de modo a que este possa, de forma mais dinâmica, acompanhar o caminho que se encontra a ser percorrido.

1.6 Estrutura do documento

Este documento encontra-se dividido em múltiplos capítulos, cada um referente a uma parte do desenvolvimento e com a sua respetiva importância para o resultado final.

Os capítulos presentes são os seguintes:

- **Introdução**
    - Neste capítulo é apresentada a empresa, o enquadramento do projeto, qual a sua motivação e os seus objetivos;
- **Contexto**
    - Neste capítulo é apresentada uma contextualização de vários temas que sejam relacionados com o projeto e considerados relevantes para a compreensão do produto que se pretende desenvolver;
- **Estado da Arte**
    - Neste capítulo é realizada uma avaliação das diversas alternativas que respondam aos requisitos do sistema, através da viabilidade e usabilidade de cada uma.
    - Também são apresentados um estudo de todas as tecnologias necessárias para o desenvolvimento e um levantamento de requisitos, e são apresentados todos os métodos, técnicas e tecnologias utilizadas ao longo do projeto;
- **Análise de Valor**
    - Neste capítulo são descritos os métodos e as técnicas usadas para realizar a análise de valor do projeto, para além das suas partes interessadas.
- **Desenho**
    - Neste capítulo são apresentadas as decisões arquiteturais e os seus impactos para o desenvolvimento do projeto.
- **Implementação**
    - Neste capítulo são apresentados todos os pormenores técnicos considerados relevantes para o sucesso da implementação e manutenção do projeto.
- **Experimentação e Avaliação**
    - Neste capítulo são apresentados todos os testes realizados à solução desenvolvida, onde é avaliado o sucesso na implementação;
- **Conclusões**
    - Neste capítulo são apresentadas todas as deduções retiradas do desenvolvimento e implementação do projeto, bem como o que poderá surgir deste desenvolvimento.

2 Contexto

Na parte do contexto, será realizada uma explicação de todos os temas necessários para a completa compreensão do problema que é proposto realizar durante esta dissertação.

Serão explicados os vários conceitos necessários para uma boa análise posterior da ideia e dos possíveis caminhos que poderão ser tomados no sentido da resolução do problema.

2.1 Gestão de controlo de rede centralizada

Atualmente, os sistemas de gestão de redes (ADMS) são produtos extremamente complexos, com muitas dependências entre os vários componentes, tornando assim muito complicada a sua aplicação a pequenos contextos. São exemplos destes últimos, casos de redes empresarias ou redes militares isoladas. Para além disso, é quase impossível esta aplicação quando se fala em condomínios (conceito de micro-rede).

Os ADMS estão instalados no cliente, geralmente numa sala de comandos de forma isolada e sem grande capacidade de escalonamento e expansão. Adicionalmente, é de sublinhar o facto de a atualização para uma versão mais recente ser sempre um processo demorado e penoso.

Este sistema está dividido nas seguintes três componentes:

- **SCADA**

Um sistema de supervisão e aquisição de dados (SCADA) é um sistema de *software* com o propósito de monitorizar e supervisionar as variáveis e os recursos de um sistema de controlo.

Este recolhe dados de diversos parâmetros como a temperatura, pressão, peso, tensão, entre outros, de vários recursos registados no sistema e produz gráficos de sinóticos para supervisão, gerando alarmes de dados que não estejam de acordo com o configurado, e exibindo relatórios através dos históricos de alarmes, valores recolhidos e comandos enviados para os recursos.

Este também tem a capacidade da gestão de utilizadores e acessos dos mesmos às várias funcionalidades do sistema.

Este sistema é dividido em cinco níveis bem definidos:

Nível 0. Contém os recursos físicos da estação que medem os vários parâmetros de análise de controlo;

Nível 1. Contém os módulos de (I/O) e os seus processadores elétricos.

Nível 2. É composto por polos de computadores de supervisão que agregam vários valores dos parâmetros definidos no nível 0 e disponibilizados no nível 1.

Nível 3. É o nível onde é realizado o controlo e monitorização da produção.

Nível 4. Onde são geridos, de um modo geral, os vários controlos de acordo com a agenda da entidade responsável. (Efacec, 2018)

- ***Distribution Management System* (DMS)**

Um DMS é um sistema que aglomera um conjunto de aplicações para o controlo e monitorização de uma rede de distribuição, de forma eficiente e fiável. Atua como suporte à decisão do operador que se encontre na sala de controlos.

Efetivamente, tem acesso a dados em tempo real e tenta produzir o máximo de informação que tenha disponível. Estes sistemas podem variar conforme a região demográfica, dificultado a comutação entre eles (Efacec, 2018).

- ***Outage Management System* (OMS)**

O OMS é o sistema que garante o controlo de interrupções e auxilia no restauro das mesmas.

Para além disso, é responsável por gerar alarmes e manter um histórico do estado da rede ao longo da sua atividade.

Estes componentes encontram-se a correr num ou mais servidores (sala de controlo) e comunicam através de uma rede física dedicada (Efacec, 2018).

2.2 Micro-rede

2.2.1 Definição

O conceito de micro-rede tem vindo a ser definido como uma solução tecnológica que assegura a operação de redes em pequena escala e que apresenta um conjunto de características próprias, tais como:

- O isolamento em chamadas “ilhas” que não têm qualquer contacto com a rede, podendo estas gerir-se a elas próprias de forma parcial ou total;
- Capacidade de assegurar o abastecimento de estabelecimentos prioritários, como por exemplo, hospitais, de modo a garantir a continuidade de serviços mínimos quando não existe ligação com a rede principal.

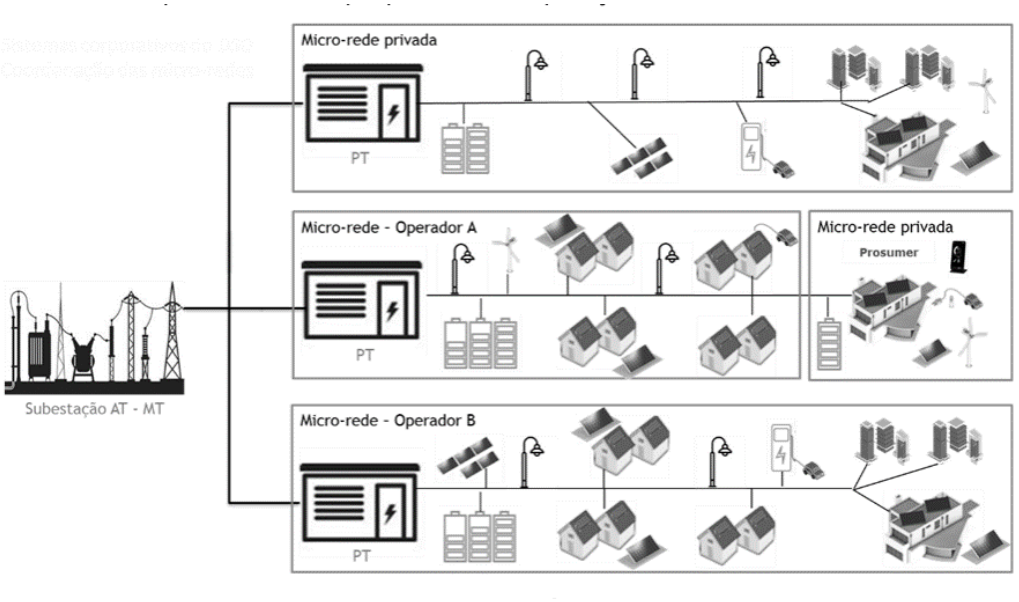

Figura 1 – Conceito de micro-rede

Como pode-se verificar na Figura 1, a rede pode ser dividida em várias micro-redes e estas podem serem controladas por entidades diferentes. A título de exemplo, uma pode ser um condomínio privado e as outras duas podem ser geridas por operadores diferentes, embora pertençam à mesma entidade, gerando assim um sistema de hierarquia. As três micro-redes na Figura 1 têm capacidade de armazenamento de baterias e assim a possibilidade de se desligarem definitivamente ou temporariamente da rede elétrica.

Estas redes, se estiverem ligadas com a rede principal, a ligação é realizada apenas num ponto, de fácil controlo. Caso estas redes não possuam qualquer ligação com a rede principal, são consideradas redes isoladas (*off-grid*), tratando-se de uma situação muito comum em locais remotos com grandes dificuldades da rede elétrica lá chegar, quer por questões físicas, quer mesmo políticas.

As micro-redes estão cada vez mais a assumir um papel fulcral, devido à sua menor complexidade e à sua grande eficácia na poupança de energia. Consequentemente, são de sublinhar os custos associados à sua gestão e manutenção (Efacec, 2017).

2.2.2 Componentes Básicos

Estas redes são compostas por vários componentes que podem ser divididos em três categorias: recursos de consumo, que só retiram energia da rede; recursos de produção, que produzem energia para a rede; e recursos mistos, como as baterias de *storage,* que tanto podem estar a sugar energia da rede, como a fornecê-la para a mesma.

2.2.3 Como se gere uma micro-rede

A gestão de uma micro-rede pode variar caso esta esteja ou não ligada à rede principal e aos componentes que a constituem e consoante a sua finalidade. Por exemplo, uma rede abastecida apenas por painéis fotovoltaicos geralmente não será sustentável por si só.

A estrutura de gestão de uma micro-rede é composta pelas seguintes funcionalidades:

- Uma DMS que seja responsável pela deteção de flutuações ou falhas na rede;
- Um sistema SCADA ou semelhante que controle e supervisione com aquisição de dados dos vários componentes do sistema;
- Sistemas de provisionamento, de modo a prever as flutuações da rede para que esta seja melhor gerida, com os menores desperdícios de energia possível. Com a utilização destas funcionalidades, é possível gerir de forma eficaz uma micro-rede ou um conjunto de micro-redes (Efacec, 2017).

2.3 Comparação

Num mercado cada vez mais distribuído e progressivamente com mais dispositivos, que necessitam de supervisão e controlo em áreas mais pequenas, como condomínios (por exemplo, carregadores ou painéis fotovoltaicos), a solução de micro-redes vem resolver as incapacidades dos sistemas tradicionais se adaptarem a esta nova realidade. Embora estas micro-redes ainda ofereçam um menor conjunto de funcionalidades, as mesmas podem ser consideradas não essenciais para o bom funcionamento desta pequena rede. Adicionalmente, esta segregação permite um controlo mais apertado e a possibilidade de o mesmo poder passar a ser realizado por pessoas menos especializadas na área.

Em contrapartida, a distribuição dos recursos por vários locais traz problemas de segurança e potenciais perdas de dados, problemas estes que não se encontram nos sistemas tradicionais.

Para além disso, os sistemas tradicionais, como também estão muito próximos da ligação física, conseguem ter, em teoria, menores valores de tempos entre o acontecimento e o alerta, que é despoletado no sistema.

Com todos estes pontos enunciados, conclui-se que, no caso de se tratar de uma rede extensa e sem capacidade de ser dividida, o sistema atualmente implementado consegue dar uma resposta mais segura e fiável que a sua separação em micro-redes. Porém, quando se trata de redes mais concretas e facilmente controláveis, a arquitetura das micro-redes apresenta-se com maior flexibilidade de previsão e elevada adaptação aos vários cenários, sendo que o seu custo muito mais reduzido resulta como uma mais valia.

3 Estado da Arte

3.1 Protocolo IEC61850

O protocolo IEC61850 reúne um conjunto de normas a serem usadas para a comunicação entre entidades e dispositivos inteligentes. Este conjunto de regras permite que equipamentos de diferentes marcas consigam comunicar entre si de forma clara e transparente.

Este protocolo está divido em diversos tipos de comunicações a serem trocadas, tendo cada um apenas uma tarefa:

Protocolo *Generic Object-Oriented Substation Event* (GOOSE):

As mensagens do tipo GOOSE são do tipo *multicast*. São responsáveis apenas pelo tráfego de mensagens que informam sobre qualquer atuação de qualquer proteção ou sinal digital. São as mensagens mais rápidas do sistema, devido a serem assentes em *user datagram protocol* (UDP), com a desvantagem de não terem qualquer tipo de validação e confirmação que chegam onde é suposto. Para suprimir esta contrariedade, o mesmo pacote é enviado várias vezes ao longo de um período de tempo.

Protocolo *manufacturing message specification* (MMS):

O protocolo envia mensagens do tipo *unicast*, via *transmission control protocol* (TCP), a um consumidor apenas. Estas mensagens são usadas para troca de informações apenas com o intuito de análise e pequenos controlos, devido à sua inferior velocidade comparada com o protocolo GOOSE.

Protocolo *Sampled Variable* (SV):

O protocolo SV é responsável pelo tráfego das leituras analógicas através de *multicast.* Usa uma arquitetura de publicador-subscritor para transmitir dados aos seus subscritores (clientes).

Para além das comunicações, este protocolo também define a configuração do sistema através de um ficheiro do tipo SCL, usando XML como linguagem estrutural.

Dentro deste ficheiro, existe a configuração dos vários recursos e estes são compostos da seguinte forma:

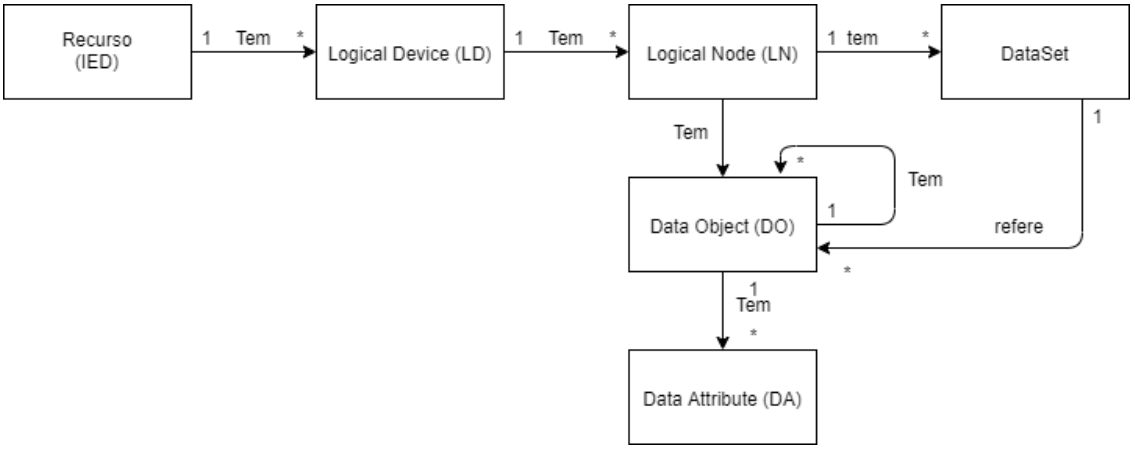

Figura 2 – Estrutura de um recurso no ficheiro SCL

Como mostra na Figura 2, um recurso é composto por vários LD e estes têm diversos LN que integram variados DO e estes podem ter outros DO ou DA. Os LN têm associados *datasets*, que são um conjunto de referências para os DO.

Existe também a possibilidade da criação de *reports* associados a LN, estes *reports* trabalham como um sistema de publicador-subscritor, onde pode-se escolher quais os parâmetros que fazem despoletar o registo de uma nova leitura, como por exemplo, periodicamente, sempre que o valor muda ou sempre que a qualidade se altera (Efacec, 2019).

3.1.1 Alternativas

Para a implementação, irão ser procuradas as bibliotecas existentes no mercado que implementem o protocolo IEC61850 e serão avaliadas quais destas melhor respondem aos requisitos que se pretendem desenvolver.

Foram encontradas três bibliotecas que poderão interessar utilizar, sendo que estas serão descritas em baixo:

Tabela 1 – Comparação de bibliotecas que implementem o protocolo IEC61850

Biblioteca	Descritivo
OpenIEC61850	A OpenIEC61850 é uma biblioteca desenvolvida em Java que implementa o protocolo IEC61850 tanto em modo MMS cliente como servidor. Esta biblioteca disponibiliza um *plugin* maven e grandle como também a possibilidade de utilizar e modificar o código fonte. Para a utilização desta biblioteca é necessária uma licença de Apache 2.0 (GmbH, 2020).
LibIEC61850	A biblioteca LibIEC61850 faz parte de um projeto que pretende disponibilizar várias funções no ramo da energia. Esta biblioteca está disponível em C e C# e fornece um cliente com as interfaces

Biblioteca	Descritivo
	MMS, GOOSE e *Sample Values* (Zillgith, 2018).
SmartGridware 61850	A SmartGridware 61850 é um conjunto de *APIs* desenvolvidas em Java para a utilização do protocolo IEC61850 como cliente e como servidor. É pago por instalação e esta necessita de ser paga para poder ter acesso às funcionalidades (Monfox, 2018).
MicroWorks IEC61850	Biblioteca que implementa IEC61850 Cliente, Servidor GOOSE e *Sample Values* (MicroWorks, 2019).
Xelas 61850 Toolkits	Biblioteca completa que oferece toda a stack IEC61850 (Software, 2004).

3.1.1.1 Fatores de escolha

Consideram-se três fatores para a escolha de uma biblioteca a ser usada:

- **Usabilidade**
  - Fator que necessita de estar escrito na linguagem usada pela empresa, que é Java.
- **Qualidade**
  - Consideram-se a disponibilidade do código, exemplos e testes presentes, um fator de escolha.
- **Modularidade**
  - Disponibiliza as funcionalidades necessárias para a realização do projeto.
  - Tem capacidade de se adaptar às alterações e às expansões que possam surgir no futuro.

Fazendo uso destes pontos, pode ser gerada uma matriz comparativa entres os fatores, de modo a avaliar o grau de importância de cada um:

Tabela 2 – Análise comparativa de fatores de escolha

	Usabilidade	Qualidade	Modularidade
Usabilidade	1	3/4	1/2
Qualidade	1/4	1	1/2
Modularidade	2	2	1

Através da Tabela 2 normalizada, obtém-se o grau de prioridade de cada fator:

Tabela 3 – Tabela de fatores de escolha normalizada

	Usabilidade	**Qualidade**	**Modularidade**	**Prioridade**
Usabilidade	3/13	1/5	1/4	23%
Qualidade	4/13	4/15	1/4	27%
Modularidade	6/13	8/15	1/2	50%

De seguida, expõe-se a análise realizada a cada biblioteca segundo os critérios descritos:

Tabela 4 – Análise das bibliotecas segundos os valores identificados (GmbH, 2020; MicroWorks, 2019; Monfox, 2018; Software, 2004; Zillgith, 2018)

Biblioteca	**Usabilidade**	**Qualidade**	**Modularidade**
OpenIEC61850	Java	Código aberto e bem documentado com testes.	Implementa o lado cliente e o lado servidor, para além disso, está disponível em *plugin*.
LibIEC61850	C#	Código aberto, mas sem qualquer explicação.	Tem de ser compilado para poder ser usado no projeto, requer um passo extra de criar uma interface.
SmartGridware 61850	Java	Boa documentação e tutoriais, mas pago.	Implementa o lado do cliente e do servidor, para além disso, está disponível através de *APIs*.
MicroWorks IEC61850	C#	Boa documentação e tutoriais, mas pago.	Implementa GOOSE e *Sample Values*, falta Polling, só implementa o cliente.
Xelas 61850 Toolkits	Java	Sem qualquer informação, é preciso pagar para mais informações.	Refere implementar a *stack* completa, mas sem mais informações.

Com estas informações, pode agora avaliar-se cada biblioteca contra os fatores de escolha, para observar qual a melhor em cada critério.

Usabilidade:

Tabela 5 – Análise comparativa segundo o fator de usabilidade

Biblioteca	OpenIEC61850	LibIEC61850	SmartGridware 61850	MicroWorks IEC61850	Xelas 61850 Toolkits	Prioridade Relativa
OpenIEC61850	1	2	1	2	1	25%
LibIEC61850	1/2	1	1/2	1	1/2	13%
SmartGridware 61850	1	2	1	2	1	23%
MicroWorks IEC61850	1/2	1	1/2	1	1/2	13%
Xelas 61850 Toolkits	1	2	1	2	1	25%

Qualidade:

Tabela 6 – Análise comparativa segundo o fator de qualidade

Biblioteca	OpenIEC61850	LibIEC61850	SmartGridware 61850	MicroWorks IEC61850	Xelas 61850 Toolkits	Prioridade Relativa
OpenIEC61850	1	3/2	3/2	5/4	2	27%
LibIEC61850	0,6	1	1/2	1/2	2	16%
SmartGridware 61850	0,6	2	1	1/2	2	23%
MicroWorks IEC61850	0,8	2	1	1	2	24%
Xelas 61850 Toolkits	1/2	1/2	1/2	1/2	1	11%

Modularidade:

Tabela 7 – Análise comparativa segundo o fator de modularidade

Biblioteca	OpenIEC61850	LibIEC61850	SmartGridware 61850	MicroWorks IEC61850	Xelas 61850 Toolkits	Prioridade Relativa
OpenIEC61850	1	2	1	1	2	25%
LibIEC61850	1/2	1	1/2	1/2	2	15%
SmartGridware 61850	1	2	1	1	2	25%
MicroWorks IEC61850	1	2	1	1	2	25%
Xelas 61850 Toolkits	1/2	1/2	1/2	1/2	1	11%

Depois de feita a comparação de cada um dos requisitos pelas várias bibliotecas, podem juntar as prioridades relativas e ficar com a ordem pela qual será adotada uma biblioteca:

Tabela 8 – Tabela final de comparação de fatores de escolha

Biblioteca	Usabilidade	Qualidade	Modularidade	Prioridade
OpenIEC61850	25%	27%	25%	25%
LibIEC61850	13%	16%	15%	14%
SmartGridware 61850	25%	23%	25%	24%
MicroWorks IEC61850	13%	24%	25%	20%
Xelas 61850 Toolkits	25%	11%	11%	16%

Da análise das várias bibliotecas, podem ordenar-se as mesmas por importância:

1. OpenIEC61850;
2. SmartGridware 61850;
3. MicroWorks IEC61850;
4. LibIEC61850;
5. Xelas 61850 Toolkits.

Com base nos resultados obtidos, será utilizada a biblioteca OpenIEC61850, pois é a que obteve melhor classificação na pesquisa realizada.

3.2 Gestão de micro-redes

3.2.1 MicroGrid Plus System

A Asea Brown Boveri (ABB) apresenta uma solução, designada Microgrid Plus System, que representa um sistema de controlo centralizado, capaz de operar todos os recursos presentes na micro-rede. Este sistema inclui controlos específicos para a gestão de redes com gerações híbridas (renováveis e tradicionais) que tem como funcionalidade principal a estabilização da rede e a integração de renováveis, garantindo simultaneamente uma reserva girante e secundária. Adicionalmente, este sistema integra algoritmos de otimização com vista à maximização do peso das energias renováveis no *mix* energético (ABB, 2017).

3.2.2 Spectrum Power Microgrid Management

Spectrum Power Microgrid Management (MGMS) é um sistema fornecido pela Siemens que permite adicionar às funcionalidades típicas de SCADA ferramentas específicas para o controlo e supervisão de micro-redes em modo de operação isolado ou conectado à rede principal. Este sistema permite ainda a inclusão de um conjunto de funcionalidades avançadas para otimizar a operação do sistema. Entre estas funcionalidades são de destacar a capacidade de previsão de produção e o consumo até um máximo de sete dias, o controlo de tensão e frequência, o deslastre de cargas para manter a estabilidade da rede, e o despacho ótimo de centrais baseado em custo económico e de emissões e em ferramentas de participação em mercados (Siemens, 2017).

3.2.3 Microgrid Management System

A Schneider Electric apresenta a solução Microgrid Management System (SE, 2020) que se divide em duas funções:

- Microgrid Control, que tem como principal função assegurar a estabilidade da rede e a segurança do abastecimento em modo de operação isolado ou conectado à rede principal.
- EcoStruxure Demand Side Management, que contém ferramentas baseadas em soluções *cloud* para otimização do desempenho da micro-rede com base em critérios de custos de energia e produtividade. Conta ainda com a capacidade de previsão de consumos e produção de Distributed Energy Resources (DER).

3.2.4 Análise comparativa

Da análise realizada, conclui-se que estas soluções são desenvolvidas com o intuito de fazer face a situações de natureza particular e são vistas de forma isolada, na medida em que as micro-redes não são consideradas como um ativo capaz de interagir e prestar flexibilidade a outros ativos presentes na sua envolvência. Deste modo, obrigam a um desenvolvimento específico, caso a caso, seguindo uma filosofia do tipo *tailor made*, na medida em que implicam um rigoroso estudo e planeamento de todos os ativos a serem incluídos, por forma a satisfazer a operação global do sistema. Por fim, referir que nenhuma solução menciona a capacidade de comunicar com aparelhos através do protocolo IEC61850.

Assim, este projeto pretende distinguir-se dos demais, com a criação de um sistema *midleware* que sirva de ponto de partida para uma gestão distribuída, não tendo como vista um cliente ou uma arquitetura em específico, mas um aglomerado das melhores funcionalidades disponibilizadas pelos concorrentes.

4 Análise de Valor

Neste capítulo, será realizada uma pesquisa de várias alternativas tecnológicas que existem e conseguem responder a este problema nas várias áreas em que este se propõe inserir.

A análise de valor tem como responsabilidade efetuar um estudo de como realizar um aumento de valor de um produto ou de um serviço com o mínimo custo possível, sem sacrificar a qualidade. Tem de ser realizado de forma sistemática, organizada e formal.

De modo a aumentar o valor de um produto ou de um serviço, deve-se ter em conta três fontes de valor: em primeiro, o valor funcional do mesmo; em segundo, qual o valor que o cliente dá aos atributos do produto ou serviço; por fim, o terceiro é o valor de mercado que o produto ou serviço poderá ter. Com a análise destes três pontos, poder-se-á ter uma visão sobre quais serão os melhores passos a tomar no sucesso do produto ou serviço.

O processo de inovação é dividido em três componentes: O *Fuzzy Front End*, o *New Process Development e* comercialização (thombremahesh, 2009).

4.1 Modelo de análise

O *Fuzzy Front End* é o processo de análise mais adequado para projetos mais experimentais, dos quais não se tem uma definição concreta, tanto da comercialização do mesmo, como das suas expetativas de retorno. Geralmente, surge como acrescento a um projeto já existente, pretendendo cobrir uma falha de negócio ou tecnológica.

Este é o início do desenvolvimento de produtos, que consiste na identificação do problema e a captação de ideias ou oportunidades. Esta fase tem como características uma maior nuvem de incerteza e uma falta de estrutura que visam a identificação das necessidades do público alvo.

A fase seguinte já requer que as ideias estejam bem estruturadas e os seus objetivos claros, de modo a se realizar o desenvolvimento do produto.

Por fim, a comercialização é a fase onde é introduzido o produto no mercado.

O modelo de *New Concept Development Model* é a representação das principais atividades do *Fuzzy Front End*. Este consiste em três componentes:

- **Motor:** Mecanismo que impulsiona as atividades de liderança e estratégia de negócio da organização;
- **Cinco Processos:** São os processos que compõem o centro do modelo, que têm em conta a visão geral, a estratégia pretendida e a sua motivação;

- **Fatores de Influência:** Variáveis não controláveis de origem interna ou externa ao projeto que afetam a sua inovação. Estes podem ser de origem política, legal, económica ou tecnológica.

Como se pode ver na Figura 3, o modelo apresenta duas opções de entrada e apenas uma de saída. Este processo inicia-se pela identificação de uma oportunidade ou pelo surgimento de uma ideia (Koen et al., 2002).

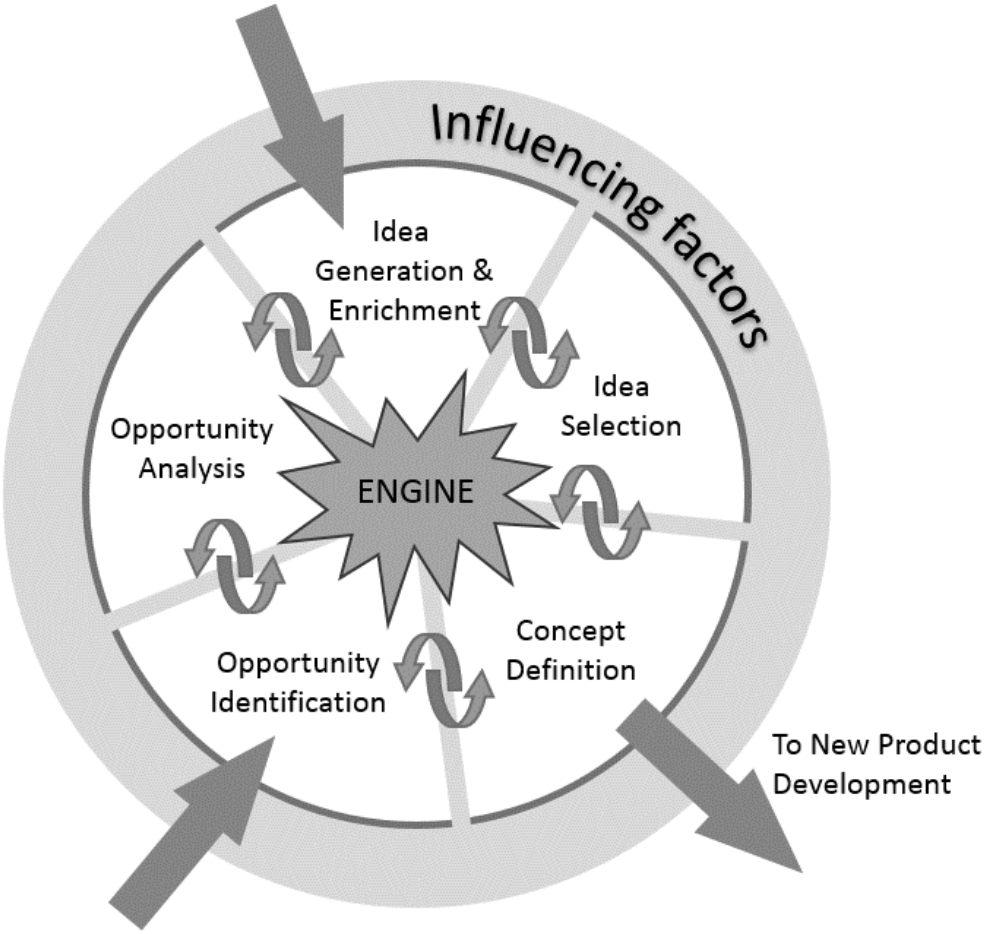

Figura 3 – Imagem ilustrativa do modelo New Concept Development

4.1.1 Identificação da Oportunidade

A identificação da oportunidade é o pontapé de saída para o modelo *New Concept Development Model*, onde é identificada a oportunidade que se pretende explorar. No caso da presente dissertação, emerge como grande oportunidade o surgimento recente do conceito de micro-rede e a sua integração no mercado elétrico, dando assim a oportunidade do seu estudo no sentido da empresa se focar neste novo ramo.

O surgimento do conceito de micro-redes motiva a empresa à adoção de um novo componente que seja capaz de, utilizando o protocolo IEC61850, realizar uma pesquisa pelos recursos existentes numa sub-rede, tornando-se apto para os mapear e agrupar por cliente. Para além disso, poder ler e escrever valores para o mesmo. É também pretendido que este novo componente seja altamente escalável.

4.1.2 Análise da oportunidade

Neste passo, é realizada a validação da viabilidade da ideia identificada. Pode ser necessária a obtenção de mais informações, de modo a obter uma decisão final com um elevado nível de confiança.

De facto, foi analisado que a falta de capacidade da empresa de comunicar com dispositivos que utilizem o protocolo IEC61850, abrindo a oportunidade para o estudo desta nova área e, assim, criar um sistema capaz de suprimir a necessidade para os sistemas já existentes, sem deixar de pensar num futuro em que este sistema poderá ser migrado para abordagens mais ágeis.

4.1.3 Enriquecimento da ideia

Esta fase corresponde ao momento em que a ideia começa a ganhar forma, sendo esta aperfeiçoada até se estar contente e confiante com a mesma, embora esteja sempre aberta a possibilidade da mesma sofrer alterações com a introdução de novos fatores de influência.

A ideia surge da vontade de romper com o processo de desenvolvimento existente na empresa, que tem alguns problemas identificados, e procurar uma nova metodologia de desenvolvimento de *software* que permita não só a integração desta nova aplicação, como também uma futura migração de outros projetos para esta nova forma, caso esta prove ser mais valiosa que a atual.

4.1.4 Seleção da ideia

A seleção da ideia é o momento onde se assume qual o caminho que se pretende tomar com base nos processos anteriores. Aqui procura-se ter a solução mais vantajosa para todas as partes interessadas na ideia proposta ou nas suas alternativas.

O objetivo desta dissertação é a criação de uma aplicação que prove a viabilidade de um futuro investimento na área de micro-redes, bem como o estudo de uma outra forma de o desenvolvimento de *software* ser realizado.

4.1.5 Definição do Conceito

O momento da definição do conceito corresponde à fase final da formalização da ideia já escolhida. Tem de se basear na potencialidade de mercado, nas necessidades reconhecidas, nas condições de investimento e no risco face aos concorrentes identificados.

A realização desta aplicação tem como pilares o recente conceito de micro-redes e a fase imatura das soluções encontradas no mercado atual, podendo assim distinguir-se dos processos atuais menos focados nesta área, como o caso da aplicação SCADA Web presente na empresa.

4.2 Análise Funcional

A proposta de valor é a forma como a organização representa os benefícios de um produto para os seus clientes. Esta tem de apresentar de forma clara e imediata os benefícios do produto, descrevendo como este se diferencia da concorrência e a razão pela qual os clientes devem optar por esse produto.

4.2.1 Modelo FAST

O diagrama FAST é um modelo muito útil para a descrição das funcionalidades do sistema, assim como para criar um pensamento crítico. Este modelo pode ser extremamente proveitoso para pessoas que não contenham conhecimentos técnicos sobre a ideia que se encontra a ser analisada (thombremahesh, 2009).

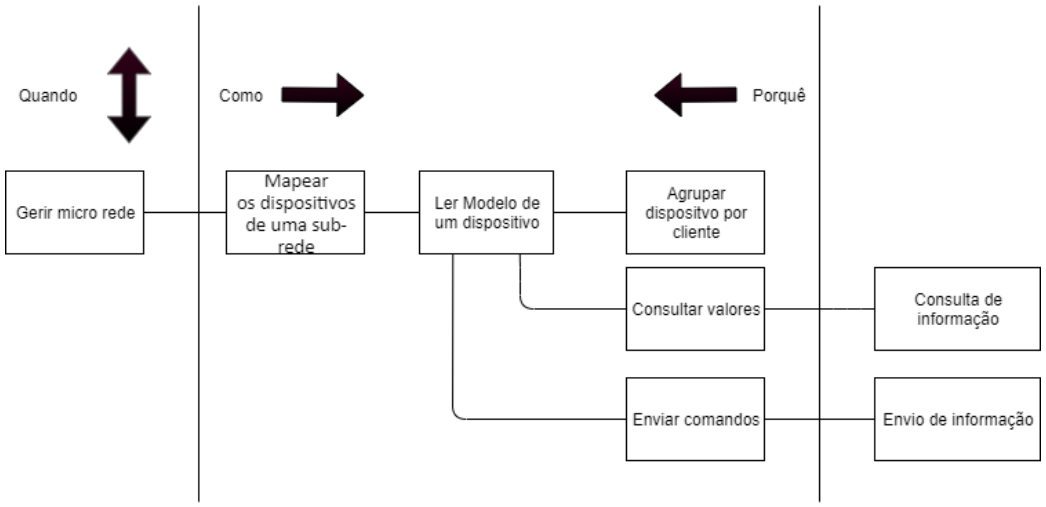

Figura 4 – Modelo FAST da análise de valor

Com a interpretação da Figura 4, pode verificar-se que, da esquerda para a direita, começa-se com uma funcionalidade e, percorrendo esta, chegar-se-á à questão de como esta é alcançável.

No modelo podem observar-se as seguintes funcionalidades:

- Mapear os dispositivos de uma sub-rede;
- Ler o modelo de um dispositivo;
- Agrupar dispositivos por cliente;
- Consultar valores;
- Enviar comandos.

Contrariamente, este pode ser interpretado no sentido inverso, da direita para a esquerda. Neste caso, o modelo serve para questionar a finalidade da funcionalidade.

Aqui pode-se, por exemplo, saber que a consulta de valores terá a finalidade de mapear e registar o recurso numa micro-rede e associá-la a um cliente.

Também deste modelo podem retirar-se quais as partes interessadas no sistema:

- Empresa:
    - A empresa constitui uma parte interessada, pois esta poderá acrescentar ao seu portefólio a capacidade de comunicar com recursos que utilizem o protocolo IEC61850.
- Clientes:
    - Neste caso os clientes serão representados por todas as aplicações que farão uso deste novo sistema.
- Utilizador:
    - Serão os operadores do sistema que, com esta implementação, não vão precisar de registar todos os recursos manualmente e poderão apenas criar uma sub-rede, tendo esta a capacidade de autodescoberta.

Daqui pode identificar-se rapidamente quem tirará mais partido de cada funcionalidade da solução proposta.

Assim, podem comparar-se as funcionalidades entre elas e verificar qual a ordem de importância para o sistema, como realçado no gráfico em baixo.

Tabela 9 – Comparação de funcionalidade com as outras funcionalidades

	Registar uma sub-rede	**Mapear os recursos de uma sub-rede**	**Agrupar recursos por cliente**	**Consultar valores**	**Enviar comandos**	Relevância
Registar uma sub-rede		+	+	+	+	4
Mapear os recursos de uma sub-rede	-		+	+	+	3
Agrupar recursos por cliente	-	-		-	-	0
Consultar valores	-	-	+		+	2
Enviar comandos	-	-	+	-		1

Posto isto, podem mencionar-se as funcionalidades pela sua ordem de prioridade:

1. Registar uma sub-rede;
2. Mapear os recursos de uma sub-rede;
3. Consultar valores;
4. Enviar comandos;
5. Agrupar recursos por cliente.

4.2.2 Modelo Canvas

O modelo de Canvas é uma ferramenta que permite desenvolver modelos de negócio, de forma estratégica, proporcionando às empresas um mapa visual do projeto que se pretende desenvolver. Este modelo é constituído pelas seguintes áreas:

- **Clientes:**
  - Nesta área é descrito para quem se vai criar o valor e quais os clientes com mais relevância para o projeto.
- **Proposta de valor:**
  - Nesta área são evidenciadas quais as mais-valias do projeto e quais dos problemas que foram identificados se pretendem suprimir.
- **Canais de distribuição:**
  - Nesta área são identificadas quais as melhores formas de atingir o(s) mercado/clientes pretendido(s), sendo que também se analisa quais as metodologias utilizadas neste momento. Para além disso, é estudado qual o canal com melhor relação qualidade/preço e que melhor se adapte às rotinas do mercado/cliente alvo.
- **Relação com os clientes:**
  - Nesta área descreve-se qual o tipo de relação existente no momento, qual o valor desta e se esta se adequa ao novo modelo. Caso esta não corresponda às perspetivas ou não exista de todo, refere-se qual a melhor forma para criar esta ponte entre o projeto e os seus clientes.

- **Fontes de receita:**
    - Nesta área realiza-se um estudo de qual o valor e a forma que os clientes estão dispostos a pagar, bem como qual o valor *target* que se encontra neste momento no segmento de mercado em questão.
- **Atividades chave:**
    - Nesta área identificam-se quais a atividades em que se considera que o projeto se vai distinguir dos demais, quais os canais de entrega, as relações com os clientes e quais as fontes de receita.
- **Recursos Chave:**
    - Nesta área são colocados quais os recursos mais valiosos do projeto, sejam eles físicos, intelectuais, humanos e/ou financeiros.
- **Parceiros chave:**
    - Nesta área são identificados os parceiros chave e os fornecedores chave que permitem uma redução do risco de incerteza.
- **Estrutura de custos:**
    - Nesta área apresentam-se quais são os custos mais importantes presentes na solução do projeto e quais as atividades chave mais caras.

(Pitchspot, 2019)

A aplicação do modelo Canvas ao projeto atual pode ser observada na imagem em seguida:

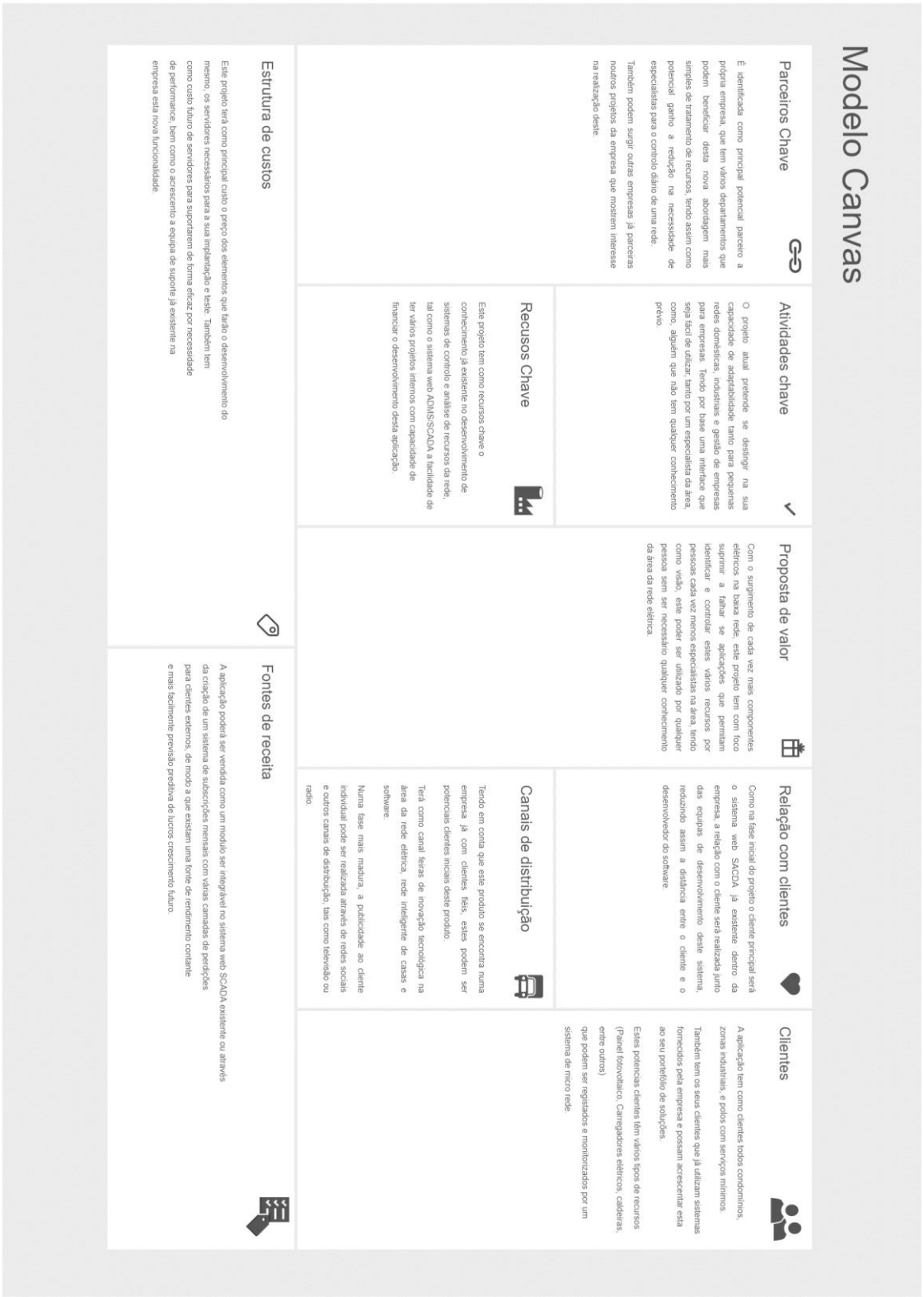

Figura 5 – Modelo Canvas do projeto

5 Análise e Desenho

De forma a que o sistema seja implementado, é necessário planear a sua execução por etapas, com o objetivo de definir como será realizada a solução para os problemas levantados anteriormente. Para isso, será apresentada uma visão geral do problema descrito com recurso à notação UML.

No sentido de satisfazer as necessidades descritas, é fulcral gerir os acessos a dados por vários clientes; podem existir dispositivos (IED) registados em micro-redes de clientes diferentes, sendo da responsabilidade da nossa implementação, a segregação dos dados.

Ter-se-á de lidar com diversos dispositivos e estes podem ficar com uma conexão presa ao sistema, caso seja pedido para atualizar um valor de um *report* periodicamente, ou sempre que exista uma alteração na qualidade. Para isso, é usado um sistema de *sockets* com publicador-subscritor, tornando, assim, obrigatório o conhecimento de que conexões se encontram ligadas e, caso estas caiam, o voltar a repor do estado das mesmas.

Existe uma limitação em que os dispositivos só permitem que uma conexão seja feita de cada vez. Assim sendo, este terá de se adaptar no caso de dois clientes realizarem dois pedidos diferentes ao mesmo dispositivo, através da criação de uma nova conexão ou da reutilização de uma previamente aberta, sendo do mesmo também a responsabilidade de redirecionar os dados para os clientes certos.

5.1 Requisitos não funcionais

Por conseguinte, os requisitos não funcionais identificados são:

Escalabilidade: A solução tem de ser capaz de escalar conforme o número de cliente e, consequentemente, de micro-redes que forem inseridos no sistema.

Desempenho: No desenvolvimento da aplicação móvel deverão ter-se em consideração aspetos como o tempo de resposta e consumo de recursos e de hardware.

Acessibilidade: A solução tem de ser capaz de ter o mínimo tempo de *downtime* nas ligações realizadas com os vários dispositivos. Também tem de garantir que está o máximo de tempo disponível para outros serviços que a pretendam utilizar.

Segurança: Uma vez que a aplicação irá manipular um conjunto de dados sensíveis que poderão estar distribuídos por vários clientes, no seu desenvolvimento deverão ser implementados mecanismos de segurança que tanto impeçam a fuga de informação e a manipulação indevida por terceiros, como garantam a segurança nas comunicações aos dispositivos.

Legais: A aplicação deverá atender ao Regulamento Europeu para a Proteção de Dados.

5.2 Requisitos funcionais

Os requisitos funcionais de um sistema vão de encontro com as necessidades reportadas pelo cliente, referindo-se às funcionalidades que o sistema deverá apresentar.

Neste sistema foram identificados os seguintes casos de uso (UC):

5.2.1 Registar um novo cliente no sistema

Pré-requisitos:

- O sistema tem de ter um administrador já registado no sistema.

Descrição:

Para o registo de um novo utilizador, o administrador terá de fornecer ao sistema o *username* e a *password*. Este verifica se o utilizador não é duplicado, encripta a *password* e guarda na base de dados, retornando a informação do sucesso da operação.

5.2.2 Registar uma micro-rede associada a um cliente

Pré-requisitos:

- O cliente tem de estar pré-autorizado a usar o sistema.

Descrição:

No processo de registar uma micro-rede, o cliente tem de fornecer um nome para a micro-rede e um conjunto de dispositivos, compostos por endereço e porta, que serão registados na micro-rede. O sistema valida os dados e guarda-os na base de dados, respondendo que o registo foi aceite.

De seguida, o sistema vai por todos os dispositivos registados na micro-rede, e que ainda não tenham sido associados a outra micro-rede do mesmo cliente, e tenta realizar uma ligação com os mesmos, atualizando os dados da base de dados para consulta posterior por parte do sistema do cliente.

Diagrama de sequência:

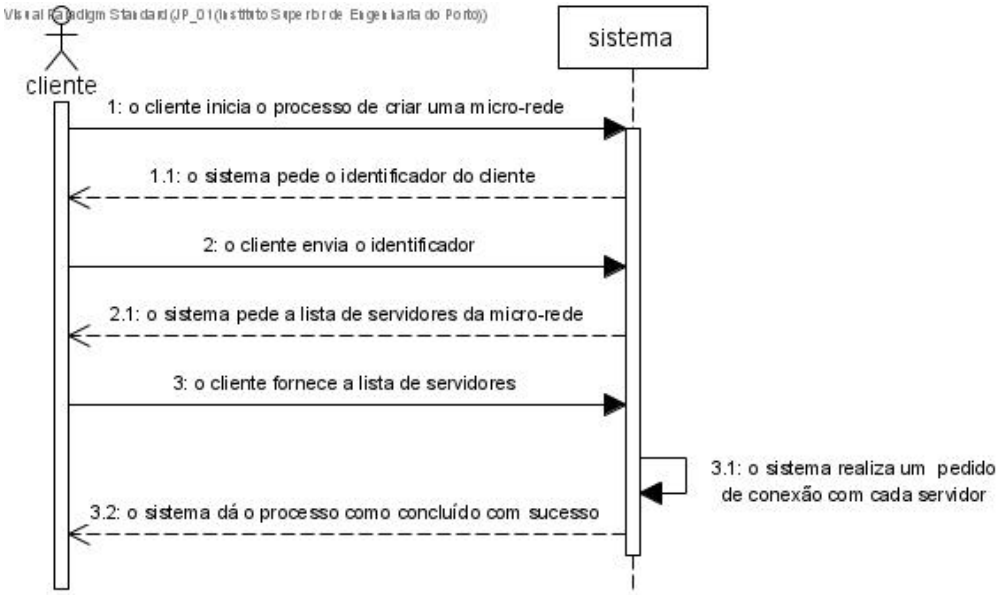

Figura 6 – Diagrama de sequência do UC 1

5.2.3 Associar uma lista de dispositivos a uma micro-rede

Pré-requisitos:

- O sistema do cliente tem de estar pré-autorizado a usar o sistema;
- O sistema tem de ter criado pelo menos uma micro-rede associada a um cliente.

Descrição:

O cliente fornece o id da micro-rede e um conjunto de dispositivos, compostos por endereço e porta, que serão registados na micro-rede. O sistema valida os dados e guarda-os na base de dados, respondendo que o registo foi aceite.

De seguida, o sistema vai pelos dispositivos registados na micro-rede e tenta realizar uma ligação com os mesmos, atualizando os dados da base de dados para consulta posterior por parte do sistema do cliente.

5.2.4 Consultar o estado dos dispositivos registados numa micro-rede

Pré-requisitos:

- O sistema do cliente tem de estar pré-autorizado a usar o sistema;
- O sistema tem de ter criado pelo menos uma micro-rede associada a um cliente;
- O sistema tem de ter acabado de processar os pedidos de conexão aos dispositivos registados na micro-rede.

Descrição:

O sistema do cliente tem acesso ao estado de cada dispositivo enviando um pedido de leitura com o identificador do cliente e da micro-rede. O sistema devolve uma lista com os estados do dispositivo, que podem ser os seguintes:

- Criado – Ainda não foi realizado nenhum pedido de conexão;
- Conectado – Encontra-se com ligação ao dispositivo;
- Desligado – Já esteve ligado, mas a ligação foi quebrada por alguma razão;
- Não encontrado – Tentou realizar a ligação sem sucesso.

5.2.5 Realizar um pedido de leitura da árvore IEC61850 de um dispositivo

Pré-requisitos:

- O sistema do cliente tem de estar pré-autorizado a usar o sistema;
- O sistema tem de ter criado pelo menos uma micro-rede associada a um cliente.

Descrição:

Neste caso de uso, o cliente realiza um pedido para o serviço realizar uma leitura da árvore de um dispositivo. Para isso, o sistema precisa de receber o identificador do cliente, da micro-rede e do dispositivo pretendido, valida a existência do dispositivo na base de dados e informa que o pedido foi aceite. De seguida, o sistema contacta a instância responsável pelo dispositivo e realiza uma leitura da árvore de atributos. Caso o dispositivo já possua uma árvore registada anteriormente, esta é substituída pela nova.

Diagrama de sequência:

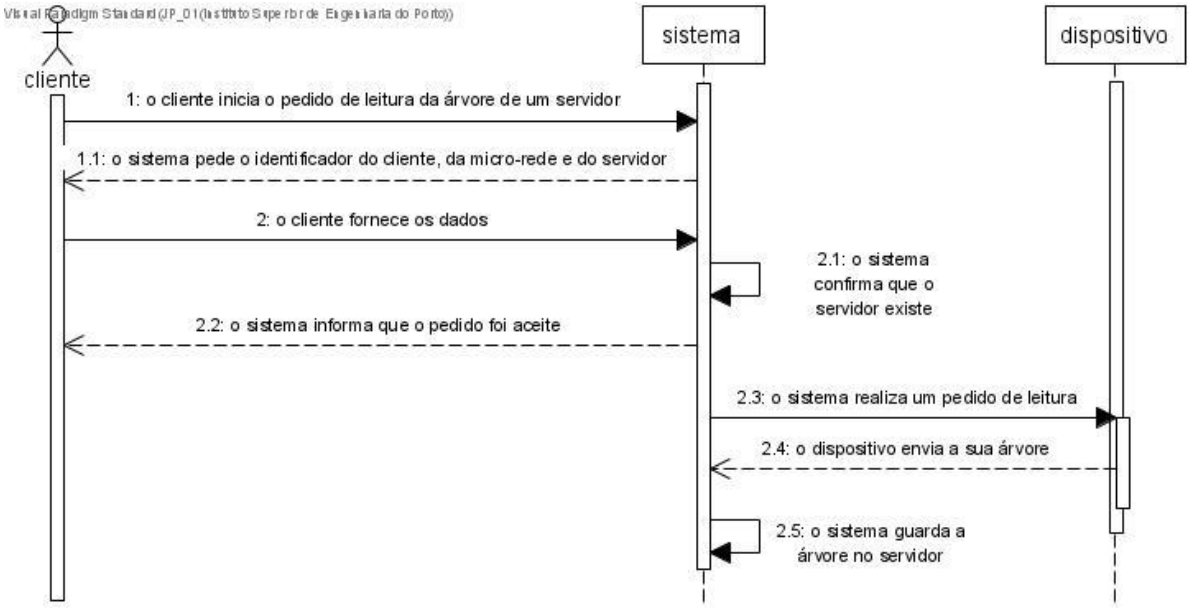

Figura 7 – Diagrama de sequência do UC 3

5.2.6 Leitura do valor de atributo associado a um dispositivo

Pré-requisitos:

- O sistema do cliente tem de estar pré-autorizado a usar o sistema;
- O sistema tem de ter criado pelo menos uma micro-rede associada a um cliente;
- O servidor tem de ter uma árvore de atributos.

Descrição:

Para realizar uma leitura de um atributo associado a um dispositivo, o cliente tem de fornecer ao sistema o dispositivo e a localização do atributo segundo a estrutura IEC61850. De seguida, o sistema contacta a instância responsável pelo dispositivo e realiza uma leitura do valor pedido e guarda-o na base de dados.

5.2.7 Enviar um comando para um dispositivo

Pré-requisitos:

- O sistema do cliente tem de estar pré-autorizado a usar o sistema;
- O sistema tem de ter criado pelo menos uma micro-rede associada a um cliente.

Descrição:

Para a escrita de um atributo associado a um dispositivo, o cliente tem de fornecer ao sistema o dispositivo e a localização do atributo segundo a estrutura IEC61850. De seguida, o sistema contacta a instância responsável pelo dispositivo e realiza uma leitura do valor pedido.

5.3 Diagrama de Domínio

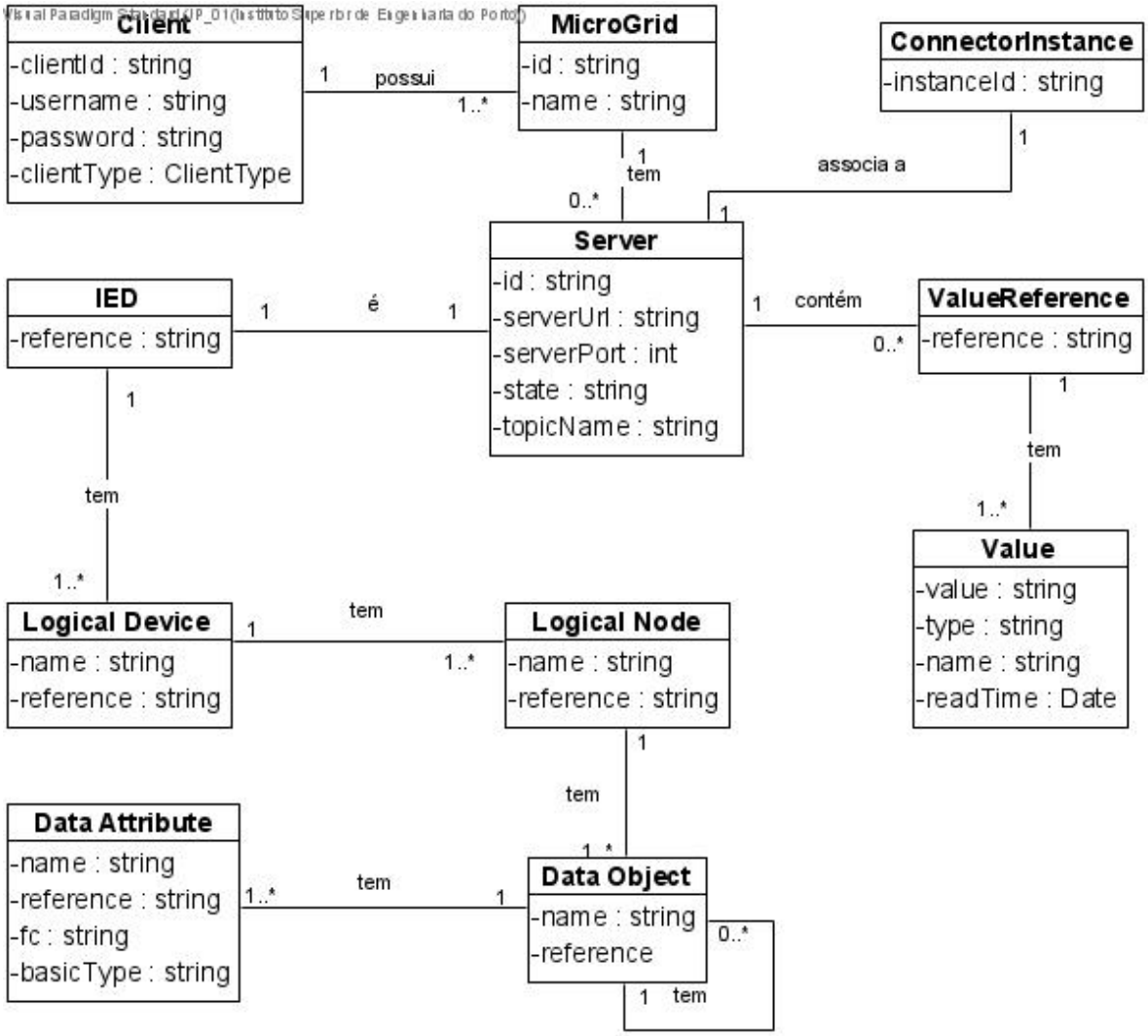

Figura 8 – Diagrama de Domínio

Pelo diagrama de domínio, é possível identificar as seguintes entidades:

Client

Representa a entidade que guarda os dados dos clientes que podem interagir com o sistema. Estes são identificados por um *username* e uma *password*.

Existem dois tipos de utilizadores:

- Administrador: É responsável pela aplicação geral, tem a capacidade de criar ou remover outros clientes, bem como manipular os dados referentes a estes;

- Cliente-Utilizador: Este tipo de utilizador apenas pode gerir micro-redes que sejam dele. Não consegue ter acesso a outras redes nem a outros valores que não estejam na gama de servidores registados por ele.

MicroGrid

É a representação lógica de um conjunto de servidores num determinado local. Tem um nome para a identificar e um conjunto de servidores.

Server

Corresponde à entidade responsável por guardar os dados referentes a um aparelho IEC61850 físico (IED). É composto por um URL e uma porta.

Este também tem o nome do tópico para onde o sistema comunica os pedidos de leitura ou escrita a serem realizados ao IED.

ConnectorInstance

É a entidade que serve para a recuperação do estado por parte das várias instâncias que sejam responsáveis pela comunicação com os aparelhos. Tem uma ligação de um para um com o servidor.

IED

Corresponde à representação lógica do aparelho IEC61850, tem a referência do mesmo e uma árvore de atributos composta pela arquitetura IEC61850 (Logical Device →Logical Node → Data Object → Data Object ou Data Atribute). Todas as entidades da árvore têm um nome e uma referência.

Value Reference

É a entidade que guarda os valores que são lidos de um atributo da árvore representada pelo IED. Contém a sua referência na árvore e o valor que é representado por um tipo de valor, o próprio valor e uma data de leitura.

5.4 Padrões Significativos

Um padrão consiste numa solução que consiga resolver de forma consistente um problema comum a vários contextos da fase de desenvolvimento de *software*. (Martin, 2016)

Nesta secção serão apresentados os padrões que foram considerados para a realização da dissertação.

5.4.1 Troca de mensagens de forma assíncrona

A troca de mensagens de forma assíncrona consiste numa forma de troca de mensagens entre duas aplicações sem que estas precisem de se conhecer, reduzindo assim o acoplamento entre aplicações na solução. Baseia-se no padrão de *Message Queueing Publish/Subscribe*. As mensagens são compostas por um cabeçalho e um corpo onde é enviada a informação.

Existem duas abordagens para a sua implementação:

- **Comunicação de mensagens de destinatário único:**

  Nesta abordagem, embora vários consumidores possam subscrever-se, a comunicação realiza-se ponto a ponto, em a mensagem é entregue a um dos consumidores e esta é lida apenas uma vez.

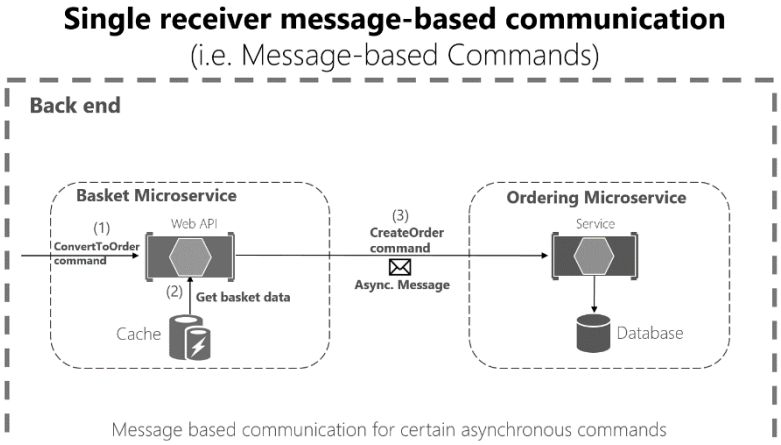

Figura 9 – Apenas um consumidor a receber uma mensagem assíncrona

- **Comunicação de mensagens de vários destinatários:**

  Nesta abordagem, o mecanismo de publicação realiza uma publicação por todos os clientes que subscrevam a um tema de forma tipo *Broadcast*. Esta forma permite uma maior flexibilidade.

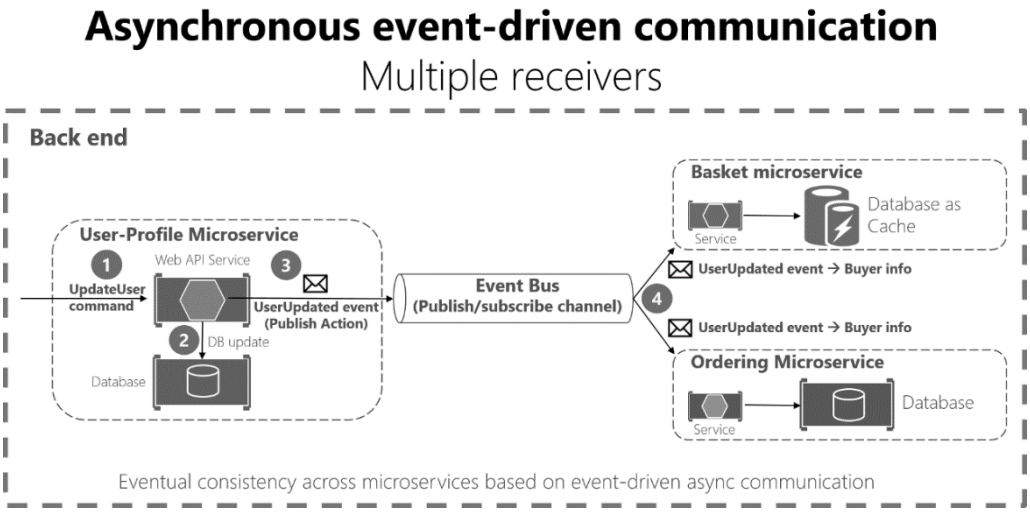

Figura 10 – Vários consumidores a receber uma mensagem assíncrona

Assim, com estas abordagens permitem tanto uma grande flexibilidade como um baixo acoplamento entre os vários serviços, mas têm como contrapartida a consistência de dados, que terá de seguir uma abordagem de consistência eventual (Microsoft, 2018).

Este padrão foi considerado útil para a comunicação entre as instâncias de sistemas diferentes, pois oferece vantagem em comparação com pedidos web RESTS, que são assíncronos por definição e obrigam a que a ligação com o servidor esteja ativa para poder realizar com sucesso a operação pretendida, sendo assim mais lentos e codependentes.

5.4.2 Consistência Eventual

A separação de responsabilidades por várias aplicações e as comunicações entre estas a serem realizadas de forma assíncrona cria um problema no que à consistência de dados diz respeito, que, ao contrário de uma aplicação monolítica, em que os dados estão guardados e o seu estado é perfeitamente conhecido por todo o sistema, estes podem estar em estados diferentes em aplicações diferentes, não permitindo em momento algum ter certeza que o estado de objeto é igual por todas as aplicações que usem o mesmo. Segundo o princípio da consistência eventual, o sistema acredita que os seus dados serão mais tarde ou mais cedo uniformizados e estarão no mesmo estado pelas várias aplicações (Fowler, 2005).

Devido ao requisito de rapidez de resposta, o conceito de consistência eventual responde de forma perfeita ao requisito.

5.4.3 Representational State Tranfer

O *Representational State Transfer* (REST) é um estilo arquitetural para a representação uniforme de comunicação entre sistemas sem que seja necessário manter o estado dos

recursos. Esta forma permite que os clientes mantenham um alto desacoplamento entre eles. Neste estilo, o cliente realiza pedidos de leitura ou escrita de um recurso do servidor. Estes pedidos são compostos por uma *Header* – onde o cliente passa os detalhes do pedido –, um *body* – que pode conter a informação do recurso a ser criado ou alterado –, e o verbo – que indica a Ação a realizar (GET, POST, PUT, PATH ou DELETE) –. Este pedido é realizado a um caminho próprio do recurso(Fielding, 2000).

Foi escolhido REST como interface para o cliente pelo facto de oferecer uma linguagem muito clara e fácil de ser replicada por qualquer sistema que pretenda realizar pedidos ao sistema.

5.4.4 JWT Token

O *Json Web Token* (JWT) é um padrão que permite a comunicação segura através do envio de objetos com o formato JSON de forma compacta e segura entra duas aplicações.

Este padrão segue uma norma que permite que os dados nele contidos possam ser validados a qualquer momento, desde que se tenha a chave com que inicialmente foram cifrados (M. Jones, 2015).

O *token* produzido é composto por três partes:

- ***Header****:* A *header* indica o tipo de *token* e o algoritmo utilizados na criptografia usada para a sua assinatura.
- ***Payload****:* Contém o JSON com as várias informações que sejam necessárias. Normalmente é a informação do utilizador autenticado.
- ***Signature*****:** Este campo tem como objetivo garantir a integridade do *token*.

Foi optado por este formato de token pela facilidade de enviar variadas características sobre o utilizador que pede o token de forma cifrada e com apenas acesso de leitura por entidades externas.

5.4.5 Contentores

A tecnologia de contentores consiste num mecanismo de criar um pedaço de uma unidade de *software* que permita que uma aplicação possa ser corrida de forma rápida e sem depender do sistema onde se encontra. Este *package* contém o código e as configurações necessárias para correr, para além das ferramentas e livrarias do sistema (Docker, 2020a).

Utilizado pela capacidade de encapsular o sistema onde corre a assim aumentar a rapidez e reduzir problemas no momento de testes e implantação do sistema.

5.5 Arquitetura

A arquitetura pretende responder aos objetivos do projeto, tendo em conta tanto os requisitos funcionais como os não funcionais.

Para isso, este projeto terá de ter em vista o facto de o sistema ter de responder a vários clientes, estes com potenciais milhares de aparelhos IEC61850 (servidores) que têm de ser conectados e cuja conexão será mantida ao longo do tempo. Também terá de ter em conta que estes pedidos poderão demorar um tempo considerável a serem respondidos por parte do aparelho.

De modo a lidar com os milhares de processos que poderão vir a surgir por parte dos clientes, será favorecida a utilização de abordagens mais dinâmicas que permitam a escalabilidade de componentes individuais, como o componente de registo de micro-redes e o componente de gestão de servidores IEC61850. Para tal serão criados dois componentes distintos:

- **Gestão de micro-redes:**
  - Componente que será responsável pela gestão de utilizadores e micro-redes. Este componente é a porta de entrada para os pedidos que sejam realizados, permitindo assim a abstração do estado, do número ou da localização do servidor a que se pretende realizar a leitura/escrita.
- **Gestão de Servidores IEC61850:**
  - Componente que tem a responsabilidade de assegurar a comunicação com os vários IED que forem inseridos no sistema, através do protocolo IEC61850, para além de ter de guardar os servidores (IEDs) pelos quais é responsável, pois, no caso de falha, terá de conseguir reestabelecer o seu estado.

Daqui é possível retirar a vista lógica do sistema:

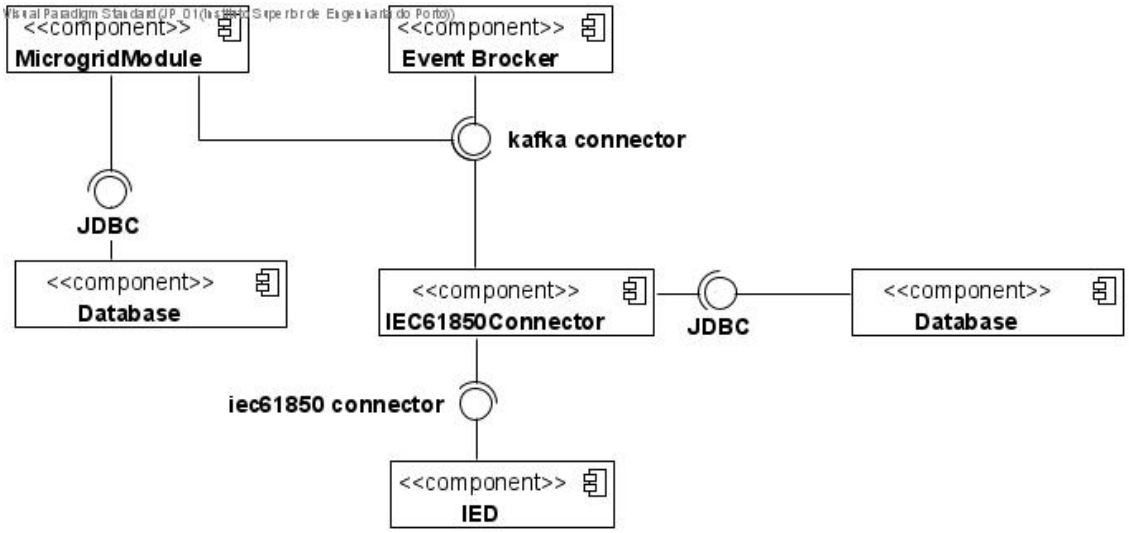

Figura 11 – Vista lógica do sistema

Como se pode ver na Figura 11, tem-se como centro os dois componentes a serem desenvolvidos (MicrogridModule e IEC61850Connector). Cada um terá uma base de dados MYSQL para persistência de dados e estes comunicarão entre si através de um *event broker* que, neste caso, será Apache Kafka. Para além destes componentes comuns, o componente MicrogridModule disponibilizará uma interface REST para a realização de pedidos e o componente IEC61850Connector terá uma biblioteca para comunicar com os servidores, usando o protocolo IEC61850.

De salientar que esta separação nesta arquitetura introduz o problema de Consistência Eventual, que se baseia na assunção de que as alterações realizadas a um recurso poderão levar a inconsistências de informação dentro do sistema, mas que este será capaz de as normalizar num determinado ponto do tempo futuro, tal como supramencionado.

Por fim, para suprimir o facto de os pedidos aos aparelhos IEC61850 serem bastante demorados, a comunicação pela aplicação será feita através da troca de mensagens assíncronas, que consiste no armazenamento de eventos num *event broker* e assim poder saber o estado do sistema em qualquer momento do tempo, apenas realizando as somas das alterações até aí efetuadas. Deste modo, atinge um menor acoplamento e uma maior capacidade de escalabilidade em função do número de servidores IEC61850.

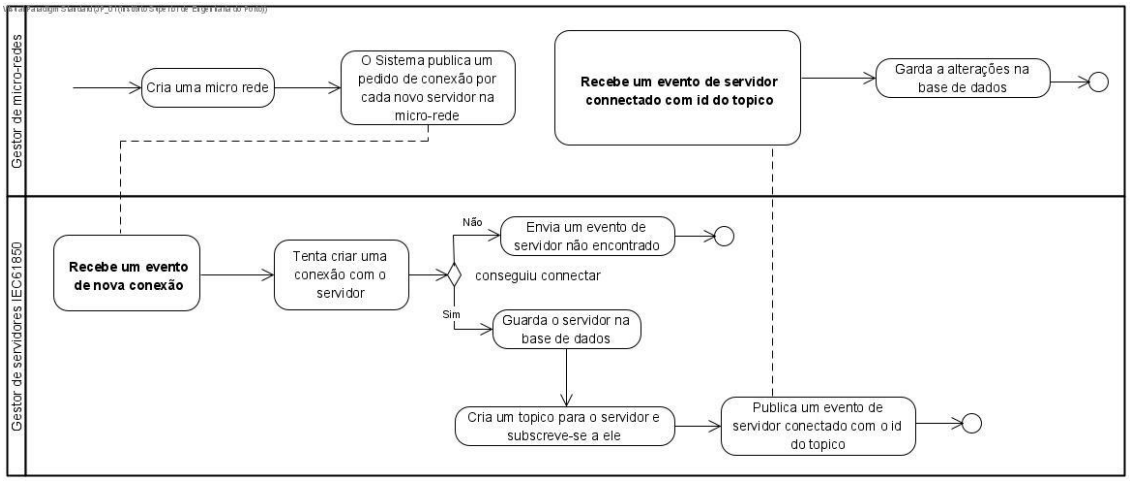

Figura 12 – Fluxo de criação de uma micro-rede

Como se vê na Figura 12 – Fluxo de criação de uma micro-rede, a criação de uma nova micro-rede leva à publicação de um pedido de conexão por cada novo servidor que for registado. De seguida, uma instância do sistema gestor de servidores IEC61850 recebe a mensagem e tenta estabelecer conexão com o aparelho. Se esta não conseguir, publica um evento a fazer notar isso, caso contrário, se a ligação for bem sucedida, esta é guardada na base de dados a ligação, sendo criado um tópico próprio para que os pedidos a este aparelho passem por esta nova via, e é publicado um evento com o resultado da operação e o id do tópico para o qual os pedidos devem ser direcionados. Por fim, uma instância recebe o evento de sucesso e guarda junto ao servidor o id do tópico a este associado.

Assim, sempre que seja necessário realizar um pedido de leitura de árvore, ou a leitura ou a escrita de um valor, é publicado no tópico próprio e a instância responsável pelo tópico (servidor) irá receber e tratar o pedido, voltando a responder por este tópico.

6 Implementação

Este capítulo incide sobre a forma como a arquitetura proposta foi desenvolvida e implementada.

Foi escolhida a *framework* Spring boot, que é uma *framewrok* para o desenvolvimento de micro-serviços. Foi escolhida pelo facto de oferecer uma interface flexível e de rápido desenvolvimento para pequenos projetos em Java (Walls, 2016), linguagem adotada pela empresa. Esta é uma *framework* alternativa à Java Enterprise Edition, que é a atualmente adotada pela empresa.

Para a troca de mensagens foi escolhido o Apache Kafka, plataforma de *streaming* de dados com capacidade tanto para criar filas de eventos como para propagar eventos por um conjunto de serviços (Foundation, 2017). Dado ser uma ferramenta conhecida na empresa, facilita a sua utilização em detrimento de outras, como por exemplo Rabbit MQ., sendo esta a razão pela qual foi escolhida.

Para o *storage* de dados foi utilizado MYSQL, sistema de base de dados *OpenSource* que também é utilizado pela empresa como principal meio de guardar informação.

Para a gestão do *deployment* recorreu-se à tecnologia de containers, nomeadamente ao Docker, permitindo que o processo de implantação fosse simples e sem necessitar de configuração de máquinas (Docker, 2020b).

6.1 Visão Geral

A implementação fixou-se no desenvolvimento de dois serviços distintos, o MicroGridModule e o IEC61850Connector.

Estes serviços partilham do código de ligação com o Apache Kafka, bem como os eventos que são trocados entre eles. Estes eventos são enviados com recurso a um conjunto de classes descritas na Figura 13 em baixo, que são responsáveis por encapsular o envio dos eventos acima descritos para o *event brocker*.

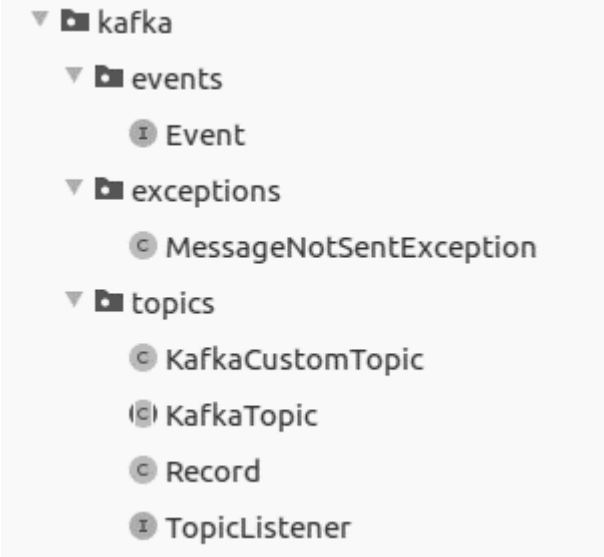

Figura 13 – Classes de encapsulamento da comunicação com o Apache Kafka

Aqui, todas as mensagens enviadas para um tópico são do tipo *Record*.

```
/**
 * Sends the record event to the kakfa topic
 * @param record record with event
 * @return true if the operation was successfully done, false otherwise
 */
public boolean sendMessage(Record<E> record) {
    String eventKey = record.getEventType() + "-" + record.getEventKey();
    String topic = getTopicName();
    boolean sent = false;
    try {
        Future<?> send = template.send(topic, eventKey, record.toJson());
        template.flush();
        sent = send.isDone();
    } catch (JsonProcessingException ex) {
        LOGGER.error("Error in converting the record with key " + record.getEventKey() + " to json", ex);
    }
    return sent;
}
```

Figura 14 – Método sendMessage presente nos projetos

Também foi criada a classe *KafkaCustomTopic*, que permite a subscrição através do *TopicListener* e o envio de eventos em *runtime*.

Esta classe é extremamente útil, pois só se sabe o *topic* para o qual se tem de comunicar com um servidor quando este for dado como conectado, tendo de ser assim criada a ligação em *runtime*.

Para além das classes de configuração e utilização do Apache Kafka, as duas aplicações também partilham os eventos que são trocados entre elas:

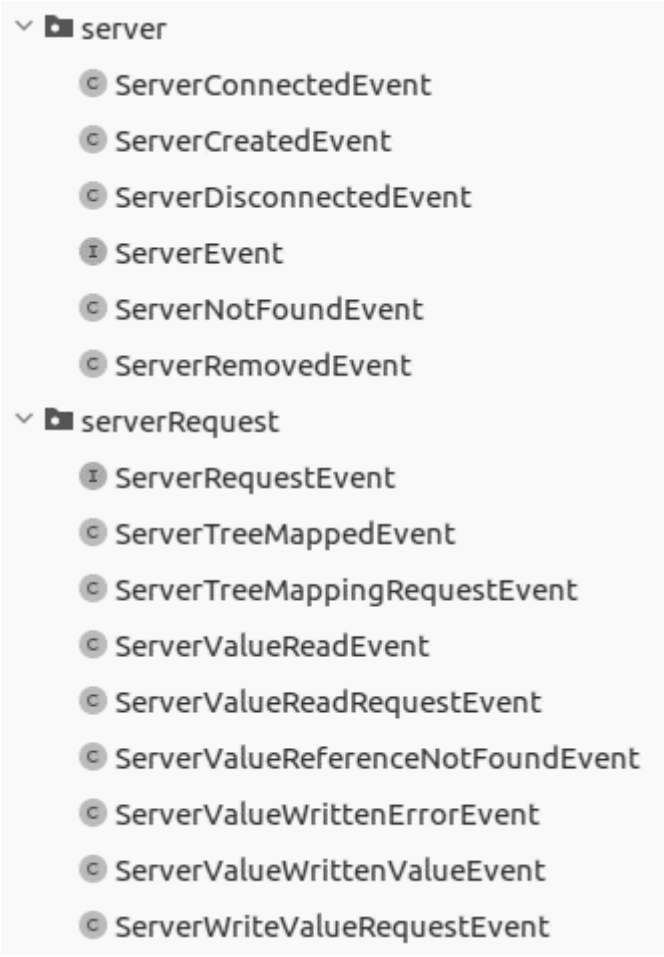

Figura 15 – Eventos trocados entre aplicações

Como se pode ver pela Figura 15, os eventos foram divididos em duas áreas:

Eventos para pedidos ao conector:

Aqui encontram-se todos os pedidos que serão realizados ao servidor. Estes eventos têm sempre “Request” no seu nome e são utilizados através do tópico próprio de cada servidor, que foi gerado no momento em que foi criada a conexão com o servidor.

Eventos de estado do servidor:

No Apache Kafka existe um tópico onde são colocados todos os eventos gerais da micro-rede, como mostra na pasta “serverstatus” na Figura 15. Todos os eventos são de carácter geral e podem ser consumidos por qualquer instância que esteja disponível.

Quanto à implantação, ambos os projetos usam contentores, nomeadamente, docker e docker-compose. Cada projeto tem um ficheiro docker (Figura 16), e na raiz dos projetos existe um ficheiro docker-compose.yml (Figura 17) onde são registadas as dependências de cada projeto, bem como as suas próprias configurações de *build*. Esta abordagem permite que o sistema e as suas dependências arranquem usando apenas uma linha código (Figura 18) podendo ainda ser manipulado o número de instâncias de cada serviço.

```
FROM openjdk:8-jdk-alpine as build
WORKDIR /app
# Copy maven executable to the image
COPY mvnw .
COPY .mvn .mvn
# Copy the pom.xml file
COPY pom.xml .
# Build all the dependencies in preparation to go offline.
RUN ./mvnw dependency:go-offline -B
COPY src src
# Package the application
RUN ./mvnw package -DskipTests
RUN mkdir -p target/dependency && (cd target/dependency; jar -xf ../*.jar)

FROM openjdk:8-jre-alpine
ARG DEPENDENCY=/app/target/dependency

# Copy project dependencies from the build stage
COPY --from=build ${DEPENDENCY}/BOOT-INF/lib /app/lib
COPY --from=build ${DEPENDENCY}/META-INF /app/META-INF
COPY --from=build ${DEPENDENCY}/BOOT-INF/classes /app

ENTRYPOINT ["java","-cp","app:app/lib/*","pt.efacec.iec61850connector.Iec61850connectorApplication"]

EXPOSE 9092
```

Figura 16 – Exemplo de ficheiro docker do serviço IEC61850Connector

```
version: '3'
services:
    # mysql
    mysql-microgrids: <7 keys>
    # mysql
    mysql-connections: <7 keys>
    # kafa image
    zookeeper: <7 keys>
    kafka: <8 keys>
    # Services
    microgrids_module: <5 keys>
    iec61850connector: <6 keys>
networks:
    default:
        external: false
    kafka_network:
    mysql_network:
volumes:
    mysql-mg-data-volume:
    mysql-co-data-volume:
    zk-data-volume:
    zk-datalog-volume:
```

Figura 17 – Visão geral do ficheiro de configuração do docker-compose

```
docker-compose up --scale iec61850connector=2
```

Figura 18 – Comando para correr as aplicações usando docker-compose

O comando presente na Figura 18 serve para iniciar o arranque das instâncias presentes no ficheiro docker-compose.yml, que por sua vez arrancará duas instâncias da aplicação IEC61850.

```
microgrids_module:
    build: microgridsmodule/
    ports:
        - 9095:9091
    networks:
        - kafka_network
        - mysql_network
    depends_on:
        - mysql-microgrids
        - kafka
    links:
        - mysql-microgrids
        - kafka
iec61850connector:
    build: iec61850connector/.
    ports:
        - 9090-9092:9090
    expose:
        - 9000-9010
    networks:
        - kafka_network
        - mysql_network
    depends_on:
        - mysql-connections
        - kafka
    links:
        - kafka
        - mysql-connections
```

Figura 19 – Parte dos serviços dentro do ficheiro de configuração do docker-compose

Na porção de código na Figura 19, é possível ver que ambos os projetos têm dependência do Apache Kafka e ambos têm a sua própria base de dados, mas não dependem um do outro, pelo que se podem arrancar por ordem aleatória, não tendo a necessidade de ter os dois serviços a funcionar para que se possa responder parcialmente aos pedidos que forem chegando ao sistema.

Com a finalidade de melhor descrever cada aplicação, de seguida estas serão abordadas em separado.

6.2 MicrogridModule

Esta é a aplicação pela qual são realizados os pedidos externos ao sistema, através de uma interface REST disponibilizada. Esta interface tem a gestão de clientes, a gestão de micro-redes e todos os pedidos de leitura e escrita a um servidor de uma micro-rede.

Disponibiliza um *endpoint* de *login* que retorna um JWT *Token* com a informação de login do cliente que pretende interagir com a aplicação.

```
{
    "tokenType": "Bearer",
    "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJTY2FkYSIsImlhdCI6MTU5NDU2NDEzNiwiZXhwIjoxNTk0NTY0NzM2LCJyb2xlIjoiUk9MRV9BRE1JTiJ9.
        7gkDMDnx4lWRZizszQ0lvx64iPized7CHTn-t4W_QDW0Uo1VFzba_cRl3hDpil9GhJWwvjlHoFoKFkvGqa50Og",
    "expiresIn": 599952
}
```


Figura 20 – Exemplo de JWT Token retornado no momento de login

Para lidar com os vários pedidos que são realizados, a aplicação está dividida em serviços que são responsáveis apenas por uma área funcional.

Figura 21 – Lista de serviços do projeto MicroGridModule

Pela Figura 21, é possível visualizar os seguintes serviços:

***ClientService*:**

Neste serviço podem ser criados novos clientes ou remover acessos aos já existentes. Estes processos apenas podem ser executados pelo utilizador com o papel de administrador. Também tem a responsabilidade pela autenticação dos utilizadores que pretendam aceder aos recursos do serviço.

***MicroGridService*:**

Serviço responsável pela gestão das micro-redes. Este serviço é ainda responsável pelas chamadas CRUD de uma micro-rede e pelos seus servidores.

Produz o seguinte evento:

- *ServerCreatedEvent*: Quando é criada ou atualizada uma micro-rede, este serviço publica um evento por cada novo servidor criado.

***ServerService*:**

Este serviço é responsavel pelas chamadas CRUD dos servidores associados a uma ou mais micro-redes. Também é neste serviço que se pode ver e comutar o estado de um servidor.

Produz o seguinte evento:

- *ServerTreeMappingRequestEvent*: Tem como parâmetro o id do servidor. Este evento é publicado quando o utilizador envia um pedido de leitura da árvore de atributos de um servidor.

***ServerListenerService*:**

Serviço responsável por subscrever ou cancelar a subscrição de um tópico que esteja associado a um servidor.

```
/**
 * Starts a new listening thread on the server topic event
 * @param server server to kafka subscribe
 */
public void startListening(@NotNull  Server server) {
    if(server == null)
        throw new NullPointerException("Server is null");

    if(server.getKafkaTopic().isEmpty()) {
        LOGGER.debug("Server with id " + server.getId() + " has not topic to subscribe, doing nothing");
        return; //Don't have topic to connect
    }

    KafkaCustomTopic<ServerRequestEvent> topic = new KafkaCustomTopic<>(servers, groupId, server.getKafkaTopic());
    if (topicList.contains(topic)) {
        LOGGER.debug("Topic " + topic.getKafkaTopicName() + "is already subscribed, doing nothing");
        return;
    }

    topic.startListeningOnTopic(new TopicListener() {
        @Override
        public void onReceivedRecord(ConsumerRecord<String, String> record) {
            LOGGER.info("Received record from topic "+ topic.getKafkaTopicName());
            ObjectMapper mapper = new ObjectMapper();
            try {
                JsonNode jsonNode = mapper.readTree(record.value());
                String eventType = jsonNode.get("eventType").asText();
                handler.handles(record.topic(), eventType, record.value());
            } catch (JsonProcessingException e) {
                LOGGER.error("Error processing incoming event " + e.getMessage());
            }

        }
    });
    this.topicList.add(topic);
    LOGGER.debug("Start listening on topic " +  topic.getKafkaTopicName());
}
```

Figura 22 – Método para subscrever a um tópico

Como se pode observar pela Figura 22, o método cria uma ligação com o tópico que esteja associado ao servidor e, quando este recebe um evento, redireciona para o *handler* correspondente. Todos os tópicos são guardados numa lista em *runtime*.

***ValueReferenceService*:**

Serviço que realiza os pedidos de leitura ou escrita de uma referência num determinado servidor de uma micro-rede.

Produz os seguintes eventos:

- *ServerValueReadRequestEvent*: Este evento é publicado quando o utilizador pretende saber o valor atualizado de um atributo no servidor. É enviado no evento o id do servidor e o atributo do qual se pretende saber o valor. Este evento não tem em conta a árvore de atributos que está associada ao servidor em questão, passando para o utilizador a responsabilidade de que o atributo exista na árvore presente no equipamento (servidor);
- *ServerWriteValueRequestEvent*: Este é publicado quando o utilizador pretende atualizar o valor de um atributo no servidor. No evento publicado é enviado o atributo e o novo valor, para além de ser produzido um id único para o pedido, sendo este utilizado mais tarde para consultar o resultado da operação. Tal como no evento anterior, não é validado se o atributo existe no servidor.

Para além dos serviços esta aplicação tem duas classes nas quais os eventos subcritos são processados:

ServerRequestsEventHandler:

Aqui são processados todos os eventos que sejam relacionados com o estado de um servidor:

- *ServerConnectedEvent*: Evento recebido quando é realizada ou recuperada a conexão com um servidor. Este evento atualiza o estado para conectado em todas as micro-redes a que este esteja inserido.
- *ServerNotFoundEvent*: Evento recebido quando é realizado um pedido de conexão sem sucesso com o servidor. Este evento coloca o estado como “Not found” no servidor correspondente.
- *ServerDisconnectedEvent*: Evento recebido quando existe uma quebra de ligação com um servidor. Este evento coloca o estado do servidor como “Disconnected”.

ServerStatusEventServiceHandler:

Já nesta classe são processados todos os eventos relacionados com pedidos a um servidor:

- *ServerTreeMappedEvent*: Evento recebido quando uma nova árvore de atributos associada a um servidor é publicada. Aqui é chamado o serverService para substituir/criar a árvore no servidor correspondente.
- *ServerValueReadEvent*: Evento recebido quando é publicada uma nova lista de valores de um atributo na árvore de um servidor. Nesta situação, o sistema cria uma nova entrada para cada valor com a referência do servidor e o valor recebido.
- *ServerValueReferenceNotFoundEvent*: Evento recebido quando o sistema IEC61850 tenta ler uma lista de valores de um atributo que não existe no servidor, isto pode acontecer caso a árvore já não corresponda à árvore no equipamento ou o utilizador tenha pedido um atributo que não exista mesmo no servidor.

- *ServerValueWrittenValueEvent*: Evento recebido quando a alteração de um valor de um atributo é executado. Aqui o sistema regista na base o sucesso da operação.
- *ServerValueWrittenErrorEvent*: Evento recebido quando a alteração de um valor não foi realizada com sucesso, pelo facto de o atributo ser *read-only* ou a árvore de atributos não permita alterações. O evento é composto pelo id do pedido e o erro correspondente.

De notar, também, que a aplicação é criada inicialmente com um utilizador configurável, com o papel de administrador, que será o único com capacidade de criar outros utilizadores.

6.3 IEC61850Connector

Esta aplicação, ao contrário da anterior, não disponibiliza interface REST, apenas comunica através de eventos. Tem a responsabilidade de lidar com todos os pedidos relacionados com um servidor, sendo agnóstico do conceito de micro-redes.

Tem uma pequena base de dados onde guarda o seu id, que é criado na primeira execução e guardado nas propriedades. Este é usado para se associar aos servidores pelo qual é responsável.

É também no momento de falha e reinício que faz uso deste id, para consultar quais os seus servidores e restabelecer a conexão com os mesmos.

Este projeto faz uso do módulo OpenIEC61850, como se pode ver na Figura 23.

```
<dependency>
        <groupId>com.beanit</groupId>
        <artifactId>openiec61850</artifactId>
        <version>1.8.0</version>
</dependency>
```

Figura 23 – Excerto do pom.xml do projeto IEC61850Connector

Para esta ligação foram criadas várias classes, como mostra a Figura 23, de modo a encapsular e facilitar a comunicação.

Figura 24 – Classes de ligação com a biblioteca IEC61850

De destacar a classe ClienteService (Figura 25), que contém todos os métodos necessários para a comunicação com um aparelho IEC61850, como criar conexão, ler ou escrever valores e consultar a árvore de atributos do servidor. Também foram implementados outros métodos que possam ser úteis no futuro, tais como criar DataSets e subscrever aos mesmos para obter valores em função de alterações de estado no servidor.

```
public interface ClientService {
    boolean connect(ClientListenerHandler handler);
    boolean disconnect();
    boolean isConnected();
    boolean isDataModeLoaded();
    boolean subscribe(String reportName, String dataSet, SubscribeRCBOptionalParameters parameters);
    boolean subscribe(String reportName, String dataSet);
    boolean unsubscribe(String reportName);
    List<BasicDataAttribute> readValue(String propertyName);
    boolean writeValue(String propertyName, String value);
    ServerModel getModel();
    IED getIED();
    boolean loadModel();
    boolean createDataSet(String name, String[] childFcModelReferences);
    boolean canDeleteDataSet(String name);
    boolean deleteDataSet(String name);
    String getServerId();
    String getAddress();
    int getPort();
    List<DataSet> getDataSets();
}
```

Figura 25 – Interface ClientService

O *package value* é dedicado a escrever valores no servidor, pois estes podem ser *ReadOnly* e não permitirem ser alterados, para isso foi utilizado o padrão *Strategy* para, conforme o FC (indicador do tipo de atributo), saber qual a forma de escrever o valor.

Este serviço recebe e processa vários eventos que são publicados pelo serviço MicroGridManagement tais como:

- *ServerTreeMappingRequestEvent*: Tem como parâmetro o id do servidor. Neste evento é realizado um pedido de leitura da árvore de atributos e publicado o evento *ServerTreeMappedEvent* com o id recebido e a árvore correspondente.
- *ServerValueReadRequestEvent*: Tem como parâmetro o id do servidor e o atributo que se pretende saber o valor. Aqui é pedido ao servidor o valor do atributo e devolvido através do evento *ServerValueReadEvent* ou, caso o atributo não exista, é publicado o evento *ServerValueReferenceNotFoundEvent* a informar o insucesso da operação.
- *ServerWriteValueRequestEvent*: Tem como parâmetros o id do pedido, o id do servidor e o atributo com o valor que se pretende escrever. É realizado um pedido de escrita do valor no atributo e enviado o evento *ServerValueWrittenValueEvent* com o id de pedido que foi executado com sucesso. Caso este falhe, é antes enviado o evento *ServerValueWrittenErrorEvent* com o erro que ocorreu.

7 Experimentação e Avaliação

Para a realização desta dissertação, é preciso definir grandezas de teste, hipóteses e metodologias de avaliação.

7.1 Emulador de Aparelho IEC61850

Devido à pandemia que surgiu este ano, tornou-se impossível realizar testes com aparelhos que comuniquem por IEC61850 reais. Foi, por isso, implementada a emulação de comunicação com o aparelho de modo a poder realizar estes testes.

Este projeto permite arrancar vários aparelhos na máquina local em simultâneo, lendo a árvore de atributos de um ficheiro próprio de configuração.

```
private ServerSap startServer(String modelPath, int port) {
    try {
        List<ServerModel> serverModels = SclParser.parse(modelPath);
        ServerModel serverModel = serverModels.get(0);
        ServerSap server = new ServerSap(port, 0, null, serverModel, null);
        server.startListening(new ServerEventListener() {

            @Override
            public List<ServiceError> write(List<BasicDataAttribute> arg0) {
                return null;
            }
            @Override
            public void serverStoppedListening(ServerSap arg0) {}
        });
        return server;

    } catch (IOException e) {
        System.err.println("IO Exception creating the server " + e.getMessage());
    } catch (SclParseException e) {
        System.err.println("Error parsing the scl file " + e.getMessage());
    }
    return null;
}
```

Figura 26 – Método para arrancar um servidor IEC61850

Para facilitar a dinâmica, foi também criada uma interface gráfica para gerir o número de servidores arrancados que existem a cada momento.

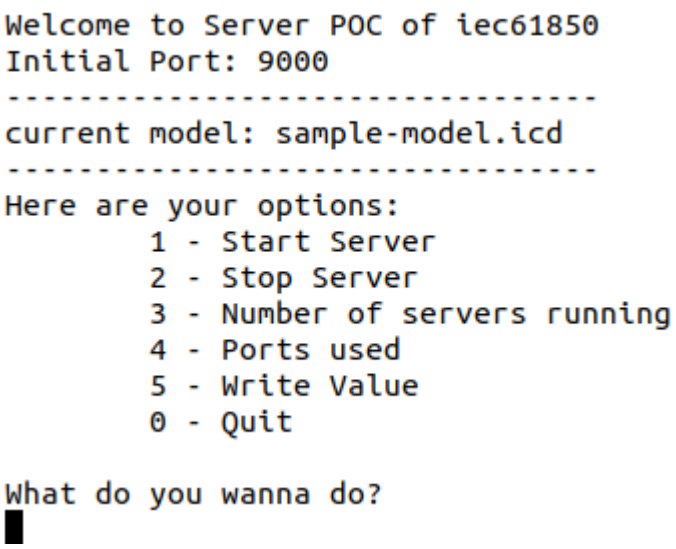

Figura 27 – Interface gráfica do emulador de dispositivos IEC61850

7.2 Indicadores a avaliar

Tendo em conta que o desenvolvimento deste projeto consiste na criação de uma plataforma de comunicação para milhares de dispositivos com bastantes restrições, existem dois potenciais indicadores a avaliar:

- Qualidade da implementação – A qualidade da implementação é crucial para tornar viável a aceitação da solução.
- Escalabilidade – Pelo facto de esta solução tanto poder estar a comunicar com um ou milhares de dispositivos.

7.3 Hipóteses

Uma hipótese consiste numa afirmação que se quer corroborar através de testes estatísticos. As grandezas identificadas foram as seguintes:

1. Sucesso de 100% dos testes – Esta hipótese, se verificada, garante que a solução implementada tem qualidade para ser considerada viável;
2. É capaz de responder com sucesso a uma amostra considerável dos dispositivos, cumprindo com os restantes testes. – Pode ser considerado parcialmente aprovado, caso esta seja capaz de servir um cliente;
3. Consegue segregar dados pelos vários clientes que se registem no sistema.

7.4 Exemplos de experiências

É possível realizar vários testes, a fim de garantir o sucesso do nosso sistema, através de alterações de estados em sistema externos. Aqui seguem alguns dos possíveis cenários que serão tidos em conta nesta fase:

- Testar o comportamento do sistema perante a falha de comunicação temporária de um dispositivo;
    - O sistema terá de ser capaz de recuperar a ligação com o dispositivo de forma automática, assim que seja possível.
- Testar o comportamento do sistema perante a falha de comunicação temporária de um cliente;
    - O sistema terá de guardar temporariamente os dados/resultados de ações que tenham sido executadas na ausência do cliente.
- Verificar qual o comportamento do sistema perante a quebra de ligação a uma micro-rede (vários dispositivos);
    - O sistema terá de ser capaz de recuperar a ligação com esta micro-rede e ligar com os dispositivos de forma automática, assim que seja possível.
- Verificar o comportamento do sistema se dois clientes realizarem o mesmo pedido de consulta ao mesmo tempo;
    - O resultado da consulta terá de ser distribuído pelos dois clientes;
- Testar qual o comportamento perante um cenário de concorrência no envio de comandos para um dispositivo;
    - O sistema terá de enviar dois pedidos, sendo o último a prevalecer.

7.4.1 Metodologias de avaliação

As metodologias de avaliação consistem nas formas pelas quais se pode avaliar as hipóteses acima descritas.

7.4.1.1 Testes unitários

Os testes unitários são validações que pretendem testar pequenas porções de código e verificar se este cumpre o propósito para o qual foi desenvolvido.

Para isso, foi seguida a estratégia de programação orientada a testes (TDD), que pode ser resumida a “testa duas vezes, codifica uma”. Antes da criação de cada teste, segundo Hamil, cada pessoa é convidada a pensar bem no que é pretendido garantir de cada funcionalidade e

qual o critério de sucesso para a mesma (Hamill, 2004). Como é seguida uma metodologia de TDD, estes inicialmente falham, dado que a funcionalidade a ser testada ainda não foi implementada. Esta metodologia tem como vantagem o facto de permitir uma maior certeza de que o que é implementado cumpre com a funcionalidade pretendida, e não se desvia em momento algum do que é pretendido.

Foi utilizada a *framework* de testes JUnit, neste caso JUnit4, e, onde os pedidos a serviços externos que não pertenciam ao que era pretendido testar, estes foram simulados com recurso à *framework* Mockitos.

```
@Mock
MicroGridRepository microGridRepository;

@Mock
MicroGridQueryServiceImpl microGridService;

@Mock
PasswordEncoder passwordEncoder;

@Mock
KafkaCustomTopic<ServerRequestEvent> topic;

@InjectMocks
ServerCommandServiceImpl service;

@BeforeEach
public void initMocks() {
    MockitoAnnotations.initMocks(this);
}
```

Figura 28 – Inicialização de serviços utilizando Mockito

```
Server server = new Server( url: "localhost",  port: 9090);
String serverId = server.getId();
String microgridId = "1";
Client client = new Client( username: "test",  password: "");
List<Server> serverList = new ArrayList<>();
serverList.add(server);
when(microGridRepository.findById(microgridId)).thenReturn(Optional.of(new MicroGrid(client,  name: "testGrid", serverList)));
```

Figura 29 – Exemplo de um teste unitário utilizando JUnit e Mockito

A Figura 29 é um exemplo da utilização da Mockito em parceria com o JUnit para realizar um teste unitário.

Optou-se também por não testar Gets e Sets, devido a estes apenas se limitarem a colocar ou a disponibilizar um atributo de uma classe. Pode-se assim considerar irrelevante para a qualidade gerar esses testes da aplicação.

Figura 30 – Exemplo de pasta de testes no projeto MicroGridsModule

Foram realizados 84 testes unitários, tendo sido atingida uma taxa de cobertura de 100% das classes de serviços do projeto.

7.4.1.2 Testes de integração

Os testes de integração são responsáveis por averiguar se os módulos de *software* desenvolvidos de forma independentes funcionam corretamente em conjunto, testando também a integração com interfaces externas.

Para este processo foram utilizados os testes unitários, mas os pedidos simulados foram substituídos por pedidos reais.

7.4.1.3 Testes funcionais

Os testes funcionais têm como objetivo compreender o comportamento da aplicação, à medida que os clientes realizam pedidos à mesma, simulando cenários semelhantes aos de produção e identificando possíveis problemas na solução.

Os testes foram realizados com o recurso ao Postman, onde foram simulados alguns cenários como:

- Registar um utilizador:
    - Tem de retorna 200 OK com o id utilizador criado

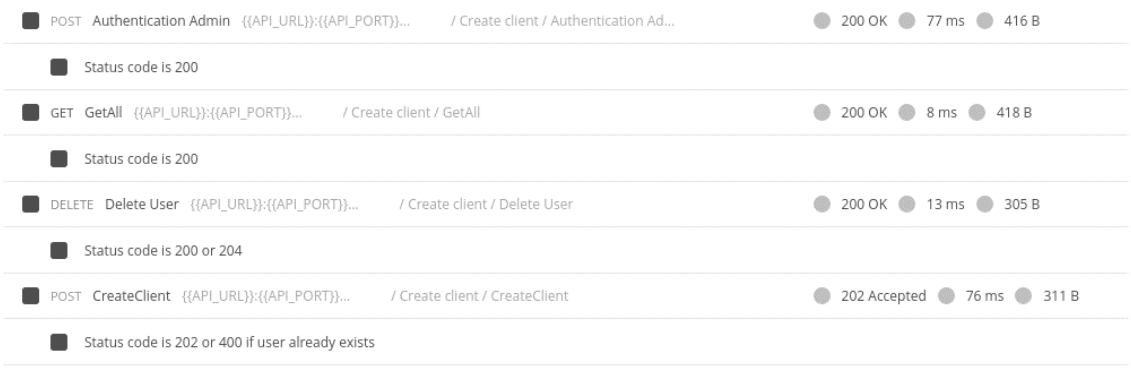

Figura 31 – Testes para gerir um utilizador no postman

- Criar uma micro-rede:
    - O recurso tem de retornar o id da rede criada com o código 202 ACCEPTED.

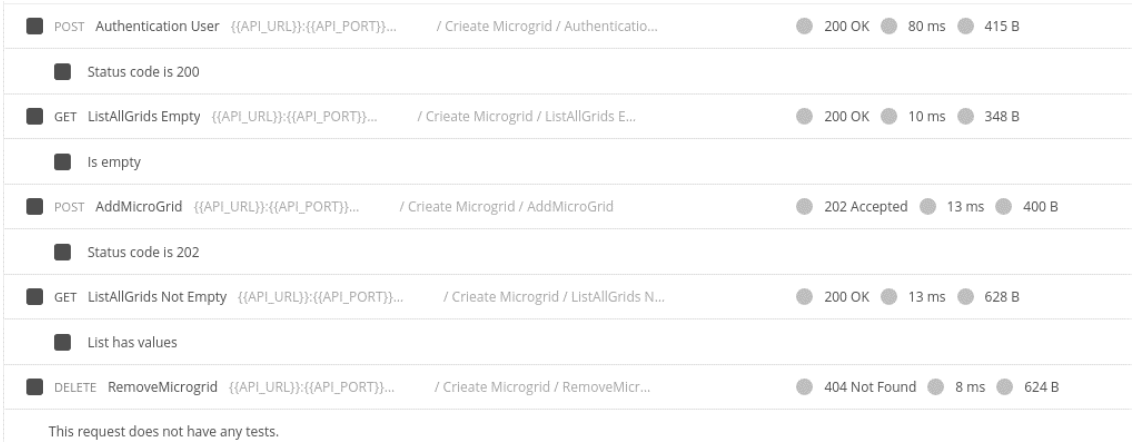

Figura 32 – Testes para criar um micro-rede no postman

- Listar servidores de uma micro-rede:
    - É esperada uma descrição da micro-rede junto dos servidores da mesma e, caso os servidores tenham a sua árvore de valores lida, esta também é incluída.
- Realizar um pedido de leitura da árvore de valores de um servidor dentro de uma micro-rede:
    - O retorno tem de ser 202 ACCEPTED, caso a micro-rede e o servidor existam, ou 400 BAD REQUEST, no caso contrário.
- Realizar um pedido de leitura do valor de um atributo da árvore:
    - O retorno terá de ser 202 ACCEPTED, caso a micro-rede e o servidor existam, ou 400 BAD REQUEST, no caso contrário.

- Consultar o valor de um atributo da árvore:
    - O retorno terá de ser uma lista com os valores lidos, ou uma lista vazia, caso não tenham sido encontrados nenhuns valores para o atributo.
- Realizar um pedido de escrita num atributo da árvore:
    - O retorno terá de ser 202 ACCEPTED, caso a micro-rede e o servidor existam, ou 400 BAD REQUEST, no caso contrário.

7.4.1.4 Testes de carga

Este tipo de teste é usado para verificar qual o limite de dados que consegue ser processado pelo sistema implementado. É essencial para validar a estabilidade do sistema quando sujeito a picos de carga.

Será utilizado o *software* JMeter para realizar estes testes de *performance* no sistema implementado.

Para manter a equidade durante os testes, estes foram todos corridos na mesma máquina que tem como configurações:

Tabela 10 – Especificações do computador utilizado nos testes

CPU	RAM	Sistema Operativo
Intel Core i7-8750H 2.2 GHz	32GB	Ubuntu 20.04

Foi criada uma lista de testes que foram corridos contra o sistema com uma instância de cada aplicação, e, de seguida, novamente *versus* o sistema, mas com uma instância da MicroGridModule e cinco instâncias da IEC61850Module. Nestes testes foi realizado um pedido de autenticação, de seguida um pedido de criação de uma micro-rede com 5 servidores IEC61850 e por fim um pedido de leitura da micro-rede criada.

Inicialmente foram realizados pedidos utilizando 10 utilizadores com um *ramp-up* de 10 segundos.

Tabela 11 – Resultados testes de performance com 10 utilizadores

Instâncias	Tipo de operação	Tempo médio de resposta (ms)	Throughput (kb/s)
1 instância MicrogridManagement	Criar micro-rede	8	19,7
1 instância IEC61850	Consultar micro-rede	372	19,5

Instâncias	Tipo de operação	Tempo médio de resposta (ms)	Throughput (kb/s)
1 instância MicrogridManagement	Criar micro-rede	6	28,7
5 instâncias IEC61850	Consultar micro-rede	229	28,7

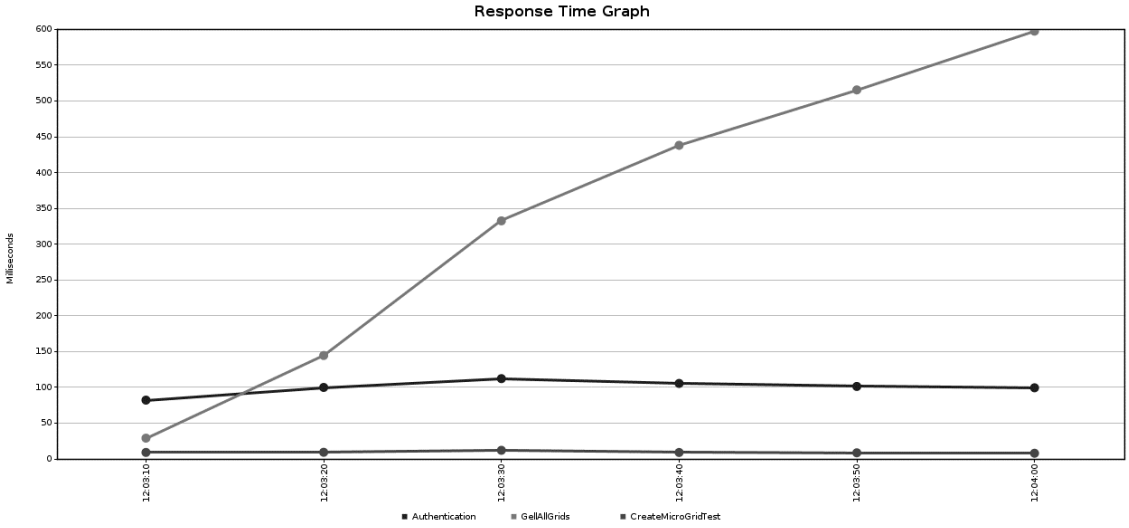

Figura 33 – Gráfico com pedidos a 10 utilizadores com 1 instância por milissegundo

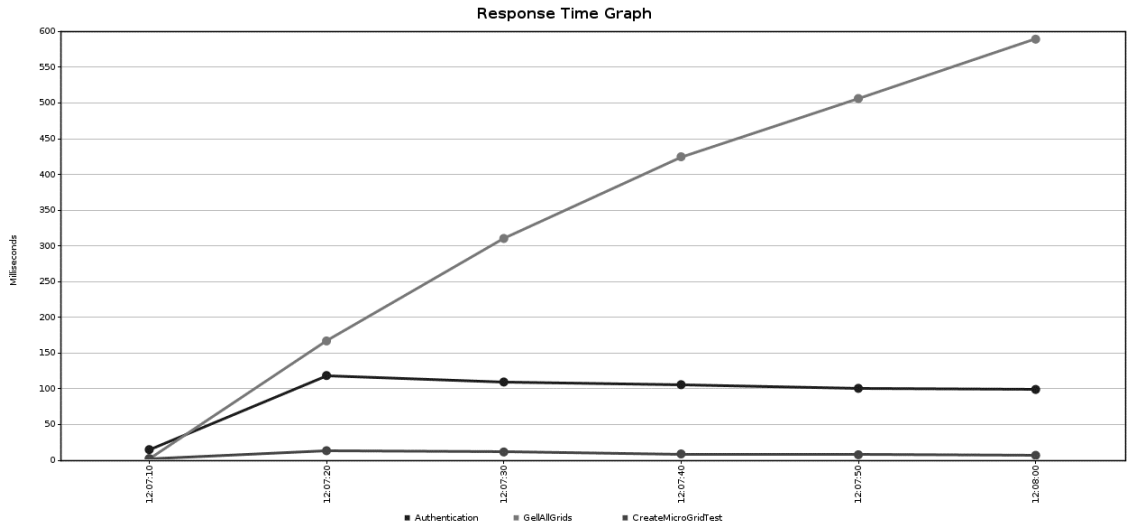

Figura 34 – Gráfico com pedidos a 10 utilizadores com 5 instâncias por milissegundo

Aqui são testados os pedidos de 1000 utilizadores a criar e consultar micro-redes com servidores associados durante um minuto:

Tabela 12 – Resultados testes de performance com 1000 utilizadores

Instâncias	Tipo de operação	Tempo médio de resposta (ms)	Throughput (kb/s)
1 instância MicrogridManagement **1 instância IEC61850**	Criar micro-rede	23637	12,8
	Consultar micro-rede	24025	14,9
1 instância MicrogridManagement **5 instâncias IEC61850**	Criar micro-rede	4387	75,1
	Consultar micro-rede	4508	80,2

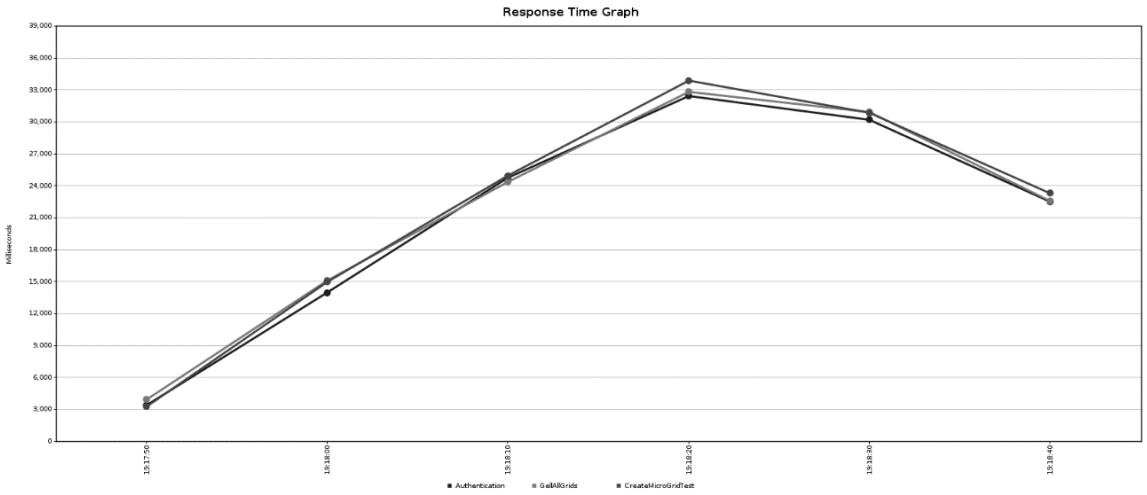

Figura 35 – Gráfico com pedidos a 1000 utilizadores com 1 instância por milissegundo

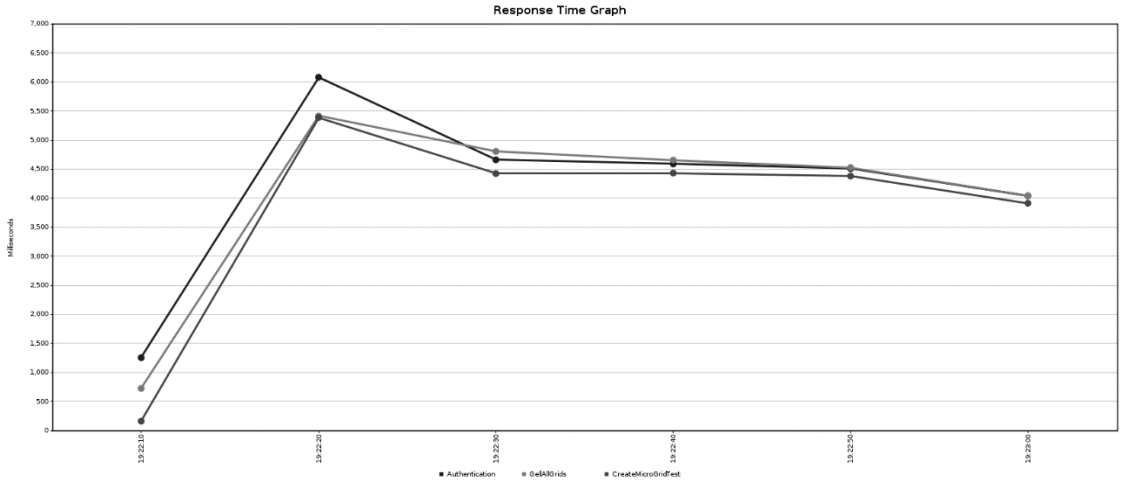

Figura 36 – Gráfico com pedidos a 1000 utilizadores com 5 instâncias por milissegundo

Pela tabela anterior é possível ver que em casos de extrema necessidade, a capacidade de instâncias vai ter de aumentar em função do número de utilizadores e qual o seu propósito, pois uma consulta de valores é muito menos dispendiosa que uma escrita.

Juntando a Tabela 11 e a Tabela 12 pode-se obter uma tabela de comparação seguinte, onde é calculada a variação do tempo de resposta com valor inicial uma instância e valor final as cinco instâncias. Também é calculada a variação do *throughput* com valor inicial cinco instâncias e valor final uma instância.

Tabela 13 – Análise comparativa de testes de carga

Instâncias	**Tipo de operação**	**Variação de tempo de resposta (%)**	**Variação do Throughput (%)**
Em 10 utilizadores, 1 instância *versus* 5 instâncias	Criar micro-rede	-25%	45,7%
	Consultar micro-rede	-38,4%	47,2%
Em 1000 utilizadores, 1 instância *versus* 5 instâncias	Criar micro-rede	-81,4%	486,7%
	Consultar micro-rede	-81,2%	438,3%

Comparando os valores obtidos pode ser concluído que a utilização de um sistema com múltiplas instâncias responsáveis por vários servidores consegue uma melhor *performance* geral em relação a uma abordagem se de uma instância se tratasse apenas. Esta evidência torna-se maior quando se aumenta o número de instâncias e de utilizadores.

7.4.1.5 Testes manuais

Estes são testes que abrangem tudo aquilo que não é possível realizar de forma automática, geralmente acompanhados por um documento de teste onde são explicados quais os passos a executar e qual o resultado esperado.

Devido à pandemia e à introdução do teletrabalho não foi possível realizar qualquer tipo de teste manual com aparelhos que comuniquem por IEC61850 físicos. Estes testes foram realizados recorrendo ao emulador desenvolvido em 7.1.

Foram realizados os seguintes testes:

Realizar uma falha de comunicação temporária num dispositivo:

Neste teste foi criada uma micro-rede com ligação a um servidor IEC61850. De seguida, foi desligada a conexão e verificou-se se esta era refletida pela aplicação, posteriormente, voltou-se a ligar o servidor IEC61850 e confirmou-se que este voltava a aparecer como *online.*

Criar uma micro-rede com um servidor, mas sem instâncias IEC61850Connector a correr:

Neste teste foi verificado que não existem instâncias IEC61850Connector a correr. Ulteriormente, foi realizado um pedido de criação de uma micro-rede e verificou-se que esta informava o sucesso do pedido. Foi arrancada uma instância IEC61850Connector e verificou-se que era criada ligação que os servidores registados anteriormente.

Simular uma falha temporária de uma instância IEC61850Connector:

Neste teste foi criada uma micro-rede com ligação a um servidor IEC61850. Posteriormente, foram desligadas as instâncias IEC61850Connector e foi realizado um pedido ao servidor, onde se verificou que a aplicação informa que o pedido está a ser processado. De seguida, foram ligadas novamente as instâncias, tendo-se verificado que as ligações com os servidores foram repostas e que o pedido foi respondido.

Remover uma micro-rede sem instâncias IEC61850Connector ligadas:

Neste teste foi confirmado que as ligações com os servidores IEC61850 estavam ativas, e, subsequentemente, foram desligadas as instâncias IEC61850Connector que mantinham estas ligações com os servidores. Foi realizado um pedido de remoção das micro-redes que continham estes servidores, arrancaram-se novamente as instâncias IEC61850Connector verificando-se que estas eliminavam do seu registo de conexões ativas os servidores removidos.

8 Conclusões

A avaliação da solução é uma das etapas cruciais no ciclo de desenvolvimento de uma aplicação. Apesar de existir sempre exigência ao longo do processo de desenvolvimento, algumas das grandezas apenas são possíveis de avaliar nesta fase mais avançada do projeto.

Nesta fase, é apresentada a avaliação da solução desenvolvida a nível técnico, é descrito se os objetivos foram cumpridos, sendo analisado o resultado das experiências e dos testes realizados.

8.1 Resposta ao problema

A realização desta dissertação teve como objetivos o estudo e implementação de um sistema que fosse capaz de traduzir o protocolo IEC6150 de forma fácil e rápida, com a capacidade de expansão no futuro. Para isso foi realizada a pesquisa de várias bibliotecas e foi escolhida a que melhor respondia ao problema segundo os critérios propostos. Foi também realizado um estudo de uma arquitetura que permitiu o fácil desenvolvimento e o acrescentar de funcionalidades que podem vir a ser necessárias não só agora, como no futuro.

Pelas razões supramencionadas a realização da dissertação, numa perspetiva global, cumpriu com os objetivos inicialmente colocados.

8.2 Resultados

Através dos resultados obtidos, pode-se afirmar que o sistema consegue falar de uma forma eficaz com aparelhos através do protocolo IEC61850 e também se encontra capaz de ser expandido para milhares de dispositivos e clientes de forma fácil e segura.

8.3 Trabalho futuro e limitações

8.3.1 Melhorias

O Sistema desenvolvido apenas implementa uma pequena parte do protocolo IEC61850; nesse seguimento, o sistema atual encontra-se preparado para expandir e agregar outras funcionalidades, tais como as subscrições a *datasets* ou o desmantelamento da árvore de atributos em algo mais “inteligente” e de melhor análise por parte dos outros sistemas.

Conquanto ainda não esteja implementado, já se pensou na hipótese de os sistemas poderem ser divididos seguindo o padrão de programação *Command Query Responsibility Segregation*

(CQRS). Este padrão separaria as leituras das escritas, dando assim liberdade de cada aplicação ser otimizada de uma forma mais focada e centralizada ao tipo de pedido.

8.3.2 Limitações

Infelizmente, no início do ano atual (2020), surgiu uma pandemia que obrigou a que todos os integrantes da empresa passassem para trabalho remoto, e mais tarde para LayOff. Esta adversidade tornou impossível o acesso a aparelhos físicos para a realização de testes ao longo do desenvolvimento, e foi ainda mais limitativo na parte final, em que era pretendido realizar vários testes manuais e pequenas demonstrações, que tiveram de passar pela simulação destes aparelhos.

Neste momento o número de instâncias tem de ser configurado por cliente, não sendo possível refletir dinamicamente as alterações de necessidades que possam vir a surgir ao longo da sua utilização, obrigando a uma intervenção manual sempre que se queira alterar o número de instâncias a correr.

8.4 Considerações finais

Esta dissertação permitiu conhecer um novo protocolo de comunicação entre aparelhos que até agora ainda não pertencia ao portefólio da empresa.

Para além disso, fomentou também a pesquisa de novas formas de desenvolver *software* e a pesquisa de novas ferramentas e *frameworks* que facilitam a vida, tanto de quem está a desenhar a arquitetura, como de quem está a desenvolver uma funcionalidade, ou mesmo, de quem está a testar essa funcionalidade.

O facto de poder sair da zona de conforto e poder testar novas formas de responder a um problema foi a motivação que levou à realização desta dissertação, o que abriu portas a que num futuro próximo, estas novas ferramentas e formas de desenvolver possam vir a ser analisadas e alargadas a toda a empresa.

9 Referências

ABB. (2017). Microgrid System Plus. Retrieved from http://new.abb.com/microgrids/ouroffering/microgrid-plus-system

Docker. (2020a). What is a Container? Retrieved from https://www.docker.com/resources/what-container

Docker. (2020b). Why Docker. Retrieved from https://www.docker.com/why-docker

Efacec. (2017). DEMOCRAT DEMOnstrator of a miCro grid integRAting sTorage.

Efacec. (2018). ScateX# Technical Specification – Functional Description.

Efacec. (2019). IEC61850 Standard Introduction Fundamentals.

Efacec. (2020). Quem somos. Retrieved from https://www.efacec.pt/quem-somos/

Fielding, R. T. (2000). Representational State Transfer (REST). Retrieved from https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Foundation, A. S. (2017). Apache Kafka. Retrieved from https://kafka.apache.org/intro.html

Fowler, M. (2005). Microservice Trade-Offs. Retrieved from https://martinfowler.com/articles/microservice-trade-offs.html

GmbH, B. (2020). OpenIEC61850. Retrieved from https://www.beanit.com/iec-61850/

Hamill, P. (2004). Unit Test Frameworks. (O'Reilly).

Koen, P. A., Ajamian, G., Boyce, S. D., Clamen, A., Fisher, E. S., Fountoulakis, S. G., . . . Seibert, R. (2002). *Fuzzy Front End : Effective Methods , Tools , and Techniques.*

M. Jones, J. B., Ping Identity, Microsoft, N. Sakimura, NRI. (2015). JSON Web Token (JWT).

Martin, R. C. (2016). Clean Architecture.

Microsoft. (2018). Asynchronous message-based communication. Retrieved from https://docs.microsoft.com/pt-br/dotnet/architecture/microservices/architect-microservice-container-applications/asynchronous-message-based-communication

MicroWorks, T. (2019). *IEC 61850 Edition 1 & 2*. Retrieved from

Monfox. (2018). SmartGridware® Solution Suite. Retrieved from http://www.smartgridware.com/products.html

Pitchspot, T. (2019). The Business Model Canvas Explained. Retrieved from https://medium.com/pitchspot/the-business-model-canvas-explained-1f5b76207f7f

SE. (2020). Microgrid applications. Retrieved from https://www.se.com/ww/en/work/solutions/microgrids/

Siemens. (2017). Management System.

Software, X. (2004). 61850 PROTOCOL STACK LIBRARIES. Retrieved from http://xelasenergy.com/products_/61850-protocol-libraries/

thombremahesh. (2009). Value Engineering And Value Analysis. *SlideShared.* Retrieved from https://pt.slideshare.net/thombremahesh/value-engineering-and-value-analysis

Walls, C. (2016). *Spring Boot in Action*: Hanning.

Zillgith, M. G., MZ Automation. (2018). libIEC61850 Documentation. Retrieved from https://libiec61850.com/libiec61850/documentation/