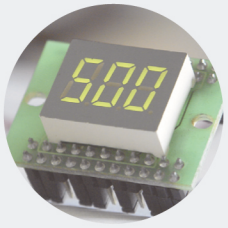
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MESTRADO EM ENGENHARIA ELECTROTÉCNICA - SISTEMAS ELÉCTRICOS DE ENERGIA






TESTES AUTOMÁTICOS DE LOCALIZAÇÃO DE DEFEITOS EM REDES DE DISTRIBUIÇÃO DE ENERGIA

PEDRO MOREIRA NOVAIS

novembro de 2022

POLITÉCNICO
DO PORTO

TESTES AUTOMÁTICOS DE LOCALIZAÇÃO DE DEFEITOS EM REDES DE DISTRIBUIÇÃO DE ENERGIA

Pedro Moreira Novais

Departamento de Engenharia Eletrotécnica

Mestrado em Engenharia Eletrotécnica – Sistemas Elétricos de Energia

2022

Relatório elaborado para satisfação parcial dos requisitos da Unidade Curricular de TEDSEE –Dissertação de Mestrado em Engenharia Eletrotécnica – Sistemas Elétricos de Energia

Candidato: Pedro Moreira Novais, Nº 1160861, 1160861@isep.ipp.pt

Orientação científica: Engª Teresa Alexandra Nogueira, tan@isep.ipp.pt

Empresa: EFACEC

Supervisão: Nuno André Silva, nuno.andre.silva@efacec.com

Departamento de Engenharia Eletrotécnica

Mestrado em Engenharia Eletrotécnica – Sistemas Elétricos de Energia

Agradecimentos

A realização de todo este percurso não seria possível sem o vasto contributo de diversas pessoas, a quem quero demonstrar o meu apreço e reconhecido agradecimento.

Dirijo a primeira palavra aos meus pais e irmão por todo o valioso apoio demonstrado ao longo da minha vida e por me proporcionarem todas as condições necessárias para atingir os meus objetivos.

À minha namorada, Margarida, pela paciência, ajuda, carinho e amor sempre presentes ao longo do meu percurso, e por acreditar em mim em todos os momentos. Por estar sempre do meu lado e inspirar-me todos os dias.

Ao meu orientador na Efacec, Nuno André Silva, por toda a disponibilidade, ensinamentos e dedicação. Ao meu amigo da Efacec, Nuno André Silva, pela amizade, boa disposição, conselhos e bons momentos passados. Representou uma presença e apoio imprescindíveis orientando-me da melhor forma para o meu sucesso e do projeto.

A todos os colegas com quem tive contacto na Efacec, especialmente Estefânia Paulo e José Simões, por todo o apoio, conversas, sugestões, e por terem tornado cada dia mais divertido.

À Engª Teresa Nogueira, a quem expresso a minha admiração, pela contínua disponibilidade e simpatia tanto no desenvolvimento desta dissertação como em todas as unidades curriculares nas quais tive o prazer de ser seu aluno.

Ao ISEP por ter sido a minha segunda casa durante a licenciatura e mestrado, e a todos os docentes presentes no meu percurso universitário, porque de alguma forma me fizeram crescer.

Aos meus amigos, que sabem quem são, por todas as conversas, risos, diversão e aconselhamento. Tornam a vida mais fácil.

O meu sincero obrigado!

Resumo

Com o aumento das linhas de distribuição de energia os operadores da rede sentem uma crescente pressão, imposta tanto pela entidade reguladora como pelos consumidores, para uma maior garantia de abastecimento e melhor qualidade de serviço. Nesse sentido, para o operador, ter uma ferramenta que o auxilie na localização do defeito pode ser crucial no tempo de recuperação do serviço se a mesma funcionar corretamente.

Na atualidade existem aplicações de energia a correr no sistema da Efacec, nomeadamente uma ferramenta que realiza o cálculo da localização de defeitos, no entanto não existe uma grande cobertura em termos de testes automáticos para essas aplicações ou funcionalidades. O objetivo é aumentar essa cobertura para as aplicações de energia e, para isso, iniciou-se o desenvolvimento de testes automáticos na ferramenta de localização de defeitos com o intuito de complementar a *pipeline* de testes automáticos já existentes que é executada diariamente. Estendeu-se assim a cobertura destes testes a uma área mais abrangente do sistema, o que permite por um lado poupar bastante tempo em testagem que seria feita manualmente, e por outro, responder com mais rapidez a possíveis problemas detetados.

Procedeu-se à identificação de uma zona de média tensão da E-REDES onde os testes serão aplicados, definindo as condições iniciais que devem ser garantidas na rede para a elaboração e execução destes testes. Foram desenvolvidos testes com intenção de testar indicadores de defeito, medidores de distância, indicadores de defeito em conjunto com medidores de distância e, por último, como reage o sistema perante cenários incongruentes.

Assim, é possível testar a ferramenta de localização de defeitos diariamente, várias vezes por dia, garantindo o seu correto funcionamento.

Palavras-Chave

Linhas de distribuição de energia, localização do defeito, testes automáticos, indicadores de defeito, medidores de distância

Abstract

With the growing number of energy distribution lines, network operators feel the pressure put by the regulator entity and the consumers, for a bigger supply guarantee and better quality of service. This way, having a tool that helps locating the fault can be crucial in reducing service restoration time.

To this day, there are energy applications running on Efacec's system, in particular there is a tool that calculates the fault location, nevertheless there isn´t a great coverage in terms of automatic tests for these applications. The goal is to enhance that coverage in terms of automatic tests for energy applications and, for that, the development of automatic tests in the fault location tool has started intending to complement the automatic tests pipeline that is executed daily. This way, the coverage of these tests is extended to a wider area of the system, what allows on one side to save lots of time testing manually, and on the other side, to react faster to possible detected problems.

It was identified a medium voltage area in the network where these tests will be applied, defining the initial conditions that must be guaranteed to develop and run these tests. The tests were developed with the intention of testing fault passage indicators, distance measurers, fault passage indicator and distance measures combined, and lastly, how the system react to non-logic scenarios.

This way, it is possible to test the fault location tool daily, more than one time a day, guarantying its correct operation.

Keywords

Energy distribution lines, fault location, automatic tests, fault passage indicators, distance measurers

Índice

Índice de Figuras

Índice de Tabelas

Siglas e Acrónimos

ANFIS – *Adaptive Neuro Fuzzy Inference System*

CC – Curto circuito

CI – *Continuous Integration*

CI/CD – *Continuous Integration and Continuous Delivery*

DevOps – *Development and Operations*

DLL – Defeito Linha – Terra

DLLLT – Defeito Linha – Linha – Linha – Terra

DLLT – Defeito Linha – Linha – Terra

DLT – Defeito Linha - Terra

DMS – *Distribution Management System*

DR – *Datarequest*

EDP – Energias de Portugal

FLF – *Fault Location Finder*

GPS – *Global Positioning System*

GUI – *Guided User Interface*

IDE – *Integrated Development Environment*

IEC – *International Electrotechnical Comission*

IED – *Intelligent Electronic Device*

IPD – Indicador de Passagem de Defeito

IT – *Information Technology*

JS – *Javascript*

LED – *Light Emitting Diode*

LTR – *Left to Right*

MT - Média Tensão

OV – Ondas Viajantes

PL/SQL – *Procedural Language/Structured Query Language*

RMS – *Root Mean Square*

RNA – Redes Neuronais Artificiais

SCADA – *Supervisory Control and Data Acquisition*

SE – Subestação

SQL – *Structured Query Language*

TAP – *Test Anything Protocol*

WKS – *Workstation*

XML – *Extensible Markup Language*

1. INTRODUÇÃO

As redes de distribuição de energia foram alvo de evolução e consequente aumento no número de linhas dedicadas à distribuição de energia. Com isto, a exigência por parte dos consumidores e das entidades reguladoras para um constante aumento da qualidade do serviço, pressionou os operadores de transporte de eletricidade a encontrar soluções para atingir essas expectativas. Daí surgiu a motivação para o desenvolvimento de sistemas de localização de defeito que fossem capazes de obter uma localização precisa para os defeitos que surgissem nas linhas de distribuição, reduzindo o tempo de interrupção do serviço, e aumentando a qualidade no tempo de resposta após defeitos.

1.1. CONTEXTUALIZAÇÃO

São vários os métodos propostos para localização automática dos defeitos, dos quais se destacam os métodos baseados na impedância aparente, os baseados em ondas viajantes e os baseados em inteligência artificial. Assim, na Efacec foi desenvolvida uma ferramenta de localização de defeitos que, como o próprio nome indica, auxilia no reconhecimento do local do defeito. No entanto, esta ferramenta apenas se torna útil caso esteja a funcionar sem erros, uma vez que sendo uma ferramenta automatizada, também está sujeita a falhas que podem levar a que esta ferramenta tenha o efeito contrário ao pretendido, que é a diminuição do tempo de duração da falha. Posto isto, esta ferramenta tem de ser testada de forma a garantir que, quando acontecer um caso real, o resultado retornado pela ferramenta de localização de

defeitos é fiável. Daqui surge a motivação para o desenvolvimento de testes automáticos que sejam capazes de testar autonomamente esta aplicação, garantindo o seu correto funcionamento.

Foram desenvolvidos quatro tipos de testes diferentes incidindo em Indicadores de Passagem de Defeito (IPDs), medidores de distância, IPDs em conjunto com medidores de distância, e testes com cenários incongruentes, de modo a verificar qual a reação do sistema.

O desenvolvimento destes testes foi feito por etapas. Primeiro começou por se idealizar cenários defeito com os equipamentos mencionados anteriormente, tendo estes cenários sido desenvolvidos e executados manualmente numa primeira fase. De seguida, foi estabelecida uma sequência de execução dos testes (correspondentes aos cenários) com o objetivo de automatizar essa sequência, colocando-a a correr de forma automática. Por fim, os testes obtidos foram implementados num bloco de testes automáticos de localização de defeitos que se encontra numa *pipeline* de testes automáticos já existente e que é executada diariamente, 4 vezes por dia.

1.2. OBJETIVOS

O objetivo do projeto é aumentar a cobertura em termos de testes automáticos para as aplicações de energia, nomeadamente para a testagem da ferramenta de localização de defeitos. Desenvolvendo testes automáticos para deteção de erros no funcionamento desta ferramenta, a intenção será a integração destes testes numa *pipeline* de testes automáticos, já existente para outras funcionalidades, que é executada várias vezes por dia, todos os dias, de modo que a ferramenta que auxilia a localização de defeitos seja regularmente testada.

Para isso, dividiu-se o desenvolvimento do trabalho de dissertação em diferentes etapas:

- Identificação de uma zona da rede de média tensão para aplicação dos testes;
- Estudo do funcionamento do sistema, nomeadamente funcionamento da interface, dos elementos da rede que farão parte dos testes e da ferramenta de localização de defeitos.
- Idealização e desenvolvimento de testes de execução manual através de comandos dbgm (comandos que efetuam a simulação da telemetria recebida no terreno) com

manipulação de estado de disjuntores, IPDs, medidores de distância, IPDs juntamente com medidores de distância, e cenários incongruentes. Desenvolvimento em simultâneo de um script de reset para colocar os equipamentos da rede no estado pretendido. Verificação do correto funcionamento dos testes e dos resultados obtidos;

- Desenvolvimento dos testes em *javascript* (JS) para futura automatização da sequência de testes. Verificação, em *logs*, da correta execução destes testes e dos resultados obtidos;
- Estudo da base de dados onde ficam armazenados os resultados destes testes para distinção dos dados necessários de submeter a validações de modo a confirmar que o teste foi bem executado;
- Testagem de *querys* em *PL/SQL* à base de dados armazenada em *SQL Developer*;
- Desenvolvimento de *script* de validações fundamentado nos três pontos anteriores;
- Desenvolvimento de *script* de pré-validação do estado da rede;
- Automatização da sequência "*script reset* – *script* teste – *script* validações" para todos os testes idealizados. Verificação do correto funcionamento e dos resultados esperados;
- Integração dos testes na pipeline de testes automáticos para validação do seu funcionamento.
- Integração de *plug-in* de ficheiros TAP para desenvolvimento/melhoria dos outputs dos testes automáticos para mais fácil e intuitiva análise;

Deste modo, é possível testar a ferramenta de localização de defeitos diariamente, várias vezes por dia, garantindo o seu correto funcionamento.

1.3. ORGANIZAÇÃO DO DOCUMENTO

No Capítulo 1 é feita a introdução ao projeto. Neste capítulo é contextualizado o projeto e a motivação do mesmo, são expostos os objetivos e as etapas efetuadas para os alcançar, e acaba com a organização do presente documento.

O capítulo 2 contém o estado da arte, ou seja, é o capítulo onde é feita a referência ao estado atual de conhecimento sobre o assunto em questão, nomeadamente sistemas de

monitorização e gestão de distribuição utilizados na Efacec, comparação entre testes manuais e testes automáticos, sistemas de deteção de passagem de defeito, métodos de localização de defeito, e classificação e consequências dos defeitos.

O capítulo 3 é onde se dá início ao desenvolvimento do projeto em questão. Nesse capítulo é identificada a zona da rede onde foram atuados os testes de localização de defeitos, é explicada a manipulação dos diferentes equipamentos presentes nessa zona da rede (estados possíveis, significado de cada estado, etc), e são explicados os cenários de testes idealizados, bem como o seu desenvolvimento, execução e resultados obtidos para cada um.

O capítulo 4 representa o maior capítulo da dissertação e é onde estão explicados os desenvolvimentos mais importantes para o sucesso do projeto. É aqui que é feita a explicação da automatização dos testes explicados no capítulo 3, bem como as ferramentas/aplicações utilizadas para tal. Para que os testes possam correr automaticamente, é necessário que sejam também validados automaticamente. Essas validações estão enunciadas neste capítulo, assim como a automatização das mesmas.

Os resultados/produtos finais do projeto estão enunciados no capítulo 5 onde é feita a descrição da *pipeline* de testes automáticos onde os testes de localização de defeitos são inseridos, é explicado o desenvolvimento do bloco de testes a ser inserido na *pipeline*, e é descrita a utilização de uma aplicação não prevista numa fase inicial do projeto de forma a melhorar o *output* obtido dos testes.

No 6º e último capítulo são apresentadas as conclusões retiradas do desenvolvimento do presente projeto, bem como são enunciados alguns possíveis caminhos a serem tomados futuramente para a continuação do desenvolvimento dos testes automáticos de localização de defeitos.

2. DEFEITOS NA REDE DE DISTRIBUIÇÃO

No presente capítulo apresenta-se o estado da tecnologia relacionada com o tópico do projeto em questão. Alguns dos pontos abordados baseiam-se em tecnologias ou elementos utilizados diretamente durante o trabalho, outros em assuntos relativos ao estado atual do tema.

2.1. SISTEMAS DE MONITORIZAÇÃO E GESTÃO DA REDE DE DISTRIBUIÇÃO NA EFACEC

São utilizados maioritariamente dois sistemas responsáveis pela monitorização e pela gestão das redes de distribuição.

2.1.1 *Supervisory Control and Data Acquisition* - SCADA

Supervisory Control and Data Acquisition (SCADA) são sistemas usados para monitorizar e controlar sistemas industriais e infraestruturas, como plantas de energia, distribuição de energia, água, gás, etc, e outros sistemas que são cruciais para a vida moderna [1], recolhendo

e analisando informação em tempo real, facilitando a transmissão de informação útil para controlo de processos, entre estações remotas e o centro de comando.

Os sistemas SCADA tratam um grande volume de dados, no entanto é crucial que se mantenha a sua fácil operação, pelo que a informação obtida a partir da constante monitorização da rede é recolhida nos postos de trabalho e analisada pelos operadores de comando de forma simples e intuitiva, sendo prontamente alertados em caso de anomalias nos processos. Para controlar a rede, estes operadores podem atuar sobre a mesma remotamente a partir dos seus postos de trabalho, possibilitando a ação rápida e direta no sistema, havendo a correção destas anomalias num curto intervalo de tempo.

O ScateX é a versão da plataforma de controlo e gestão de rede mais recente da Efacec. É um sistema SCADA e é o software presente nos postos de trabalho.

O ScateX, além da monitorização e controlo possibilita a gestão de informação, gestão da rede de energia, supervisão técnica e gestão de interrupções, fornece um suporte de decisão aprimorado e acesso à informação rápido e intuitivo. As aplicações de gestão de rede, energia e interrupção determinam a eficiência da rede, e caraterísticas como a possibilidade de expansão, bom desempenho e flexibilidade permitem que este sistema possa ser adaptado e dimensionado consoante as necessidades de diferentes redes. Assim, este software pode ser ajustado para corresponder às necessidades específicas de cada cliente/utilizador [2].

2.1.2 *Distribution Management System* – DMS

Estes sistemas de gestão da rede de distribuição (DMS) surgiram da evolução dos sistemas SCADA, no entanto, ao contrário dos sistemas SCADA que monitorizam equipamentos de uma determinada rede sem noção da interligação destes pontos, os DMS dispõem de informação sobre o modelo de conectividade da rede que controlam.

Os sistemas DMS dispõe, além das funcionalidades dos sistemas SCADA, de informação relativamente ao modelo de conectividade da rede elétrica, o que abre portas a ferramentas que auxiliam no controlo da rede:

- **Análise de Curto Circuito** – Estuda o comportamento da rede elétrica em curto circuito.
- **Previsor de Cargas** – Prevê o consumo de energia de determinados pontos/equipamentos da rede.

- **Alocação de Carga** – Calcula a carga de equipamentos que não estão a ser monitorizados, com base em medidas obtidas e monitorizadas pelo sistema.
- **Fluxo de Cargas** – Usando modelos matemáticos, calcula o fluxo de energia que passa em determinadas zonas da rede.
- **Segurança** – Possibilita a definição de regras de segurança em operações de recuperação e manutenção da rede elétrica. Assegura ainda que estas regras sejam cumpridas.

2.2. TESTES MANUAIS E TESTES AUTOMÁTICOS

No ScateX, aquando da ocorrência de um defeito na rede elétrica nacional, a localização desse defeito é calculada através da ferramenta de localização de defeitos referida anteriormente. Esta ferramenta efetua os cálculos com base na informação passada por indicadores de passagem de defeito, aparelhos de corte, medidores de distância, etc. Uma vez que esta está sujeita a falhas, caso a mesma indique uma localização errada na rede de distribuição, isto pode levar a que o defeito demore mais tempo a ser resolvido por serem concentrados os esforços numa zona "saudável" da rede. Pode ser enviada uma equipa ao local errado; o defeito que está numa zona diferente da calculada pela ferramenta vai continuar a existir e a causar danos os aparelhos de corte (partindo do princípio que estes atuaram no sítio correto) vão continuar em corte e, por isso, a real zona de defeito da rede vai continuar sem energia, etc.

2.2.1 Ferramenta de localização de defeitos

A ferramenta de localização de defeitos (*Fault Location Finder* – FLF) foi uma ferramenta desenvolvida internamente, e implementa uma solução orientada para redes de distribuição de média tensão (MT), que fornece uma lista de possíveis localizações de um defeito [3]. Esta aplicação foi feita para incorporar três tipos de informação: Informação elétrica, corrente de curto-circuito (CC), e os valores de distância ou impedância obtidos pelo aparelho de corte que foi accionado aquando da falha. Se esta informação não estiver disponível, a FLF considera todas as linhas como localizações igualmente prováveis da falha. Além disto, informação relacionada com o estado (aberto/fechado ou ativo/não-ativo) da aparelhagem ou indicadores de passagem de defeito (IPDs) são alvos de um cuidado

especial uma vez que a informação por eles disponibilizada desempenha um papel importante na redução da lista de localizações prováveis do defeito.

Os possíveis *inputs* para esta aplicação são:

- Dados relativamente a aparelhos de corte e estado de interruptores depois da falha;
- Tipo de rede e de dados de entrada;
- Valores dos dados de entrada (distância, impedância e corrente de falha);
- Margem de erro para cálculo da distância máxima e mínima do defeito;
- Percentagem de penalidade para dados de corrente de falha não confiáveis;

Caso seja um sistema de inferência *fuzzy* (abordado mais à frente no documento), há outros possíveis inputs: Condições meteorológicas e sensibilidade das linhas a essas condições; Propensão a erros/perigo das linhas; Possibilidade de estado do tempo normal, tempestade ou variável; Grau de propensão ao perigo e não propensão ao perigo; Lista de nós de referência onde falhas de energia foram reportadas por telefone.

Daqui podem ser retirados também os possíveis *outputs*, divididos em *outputs* gerais e resultados da localização de defeitos. Para os primeiros é retornado sucesso ou erro de código. Para os resultados de localização de defeitos pode ser retornado: a lista de linhas com probabilidade de defeito; lista de equipamentos entre o aparelho de corte até ao segmento de linha com probabilidade de defeito; lista de equipamentos na zona de defeito [3].

2.2.2 Testes manuais na Efacec

Assim, tal como outras ferramentas, esta também tem de ser testada. Estes testes começaram a ser implementados de uma forma manual com uma pessoa destacada para essa função, geralmente um *tester* ou *developer*, e encarregue de testar a ferramenta sempre após uma integração de uma nova versão do sistema.

O processo de teste manual, de forma simplificada, resume-se nos seguintes passos: 1. Definição de casos de estudo/teste; 2. Atualização ou criação dos protocolos de testes manuais; 3. Preparação/Garantia das condições necessárias para execução dos testes; 4. Execução dos testes manuais; 5. Validação manual dos resultados; 6. Registo dos resultados nos protocolos de teste; 7. Simultaneamente com o passo 6 ou no final desse passo, é feito o registo de potenciais problemas identificados na ferramenta *RedMine*, e são encerrados os

processos de problemas (também denominados *issues*) identificados em execuções anteriores e já resolvidos.

O facto de estes testes serem executados de forma manual traz vantagens não só a este processo, mas a todos os processos que são sujeitos a testes, vantagens essas representadas na Figura 1. No entanto, tem também algumas desvantagens associadas, as quais se encontram também enunciadas na mesma figura.

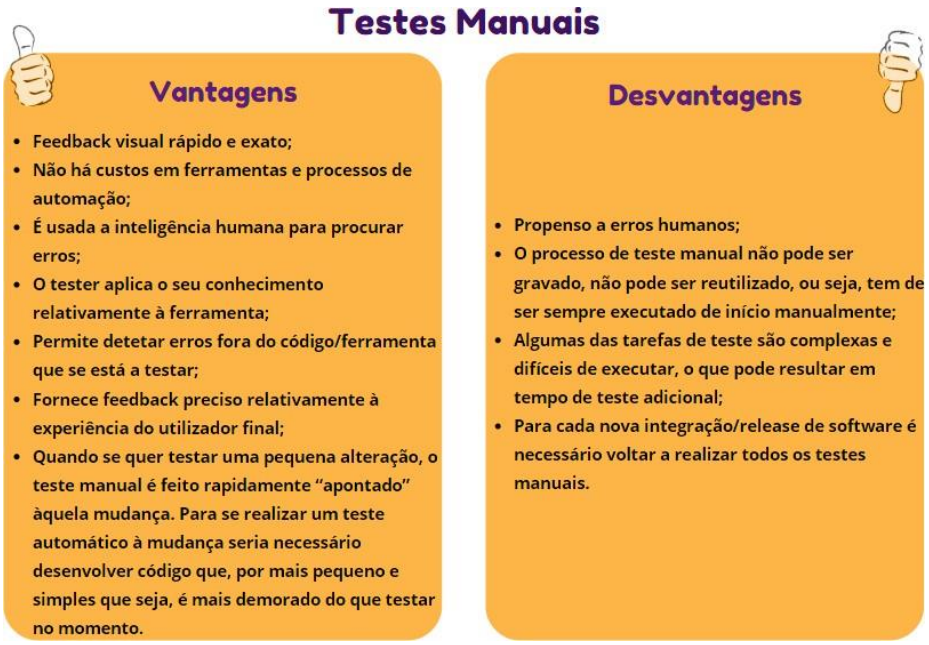


Figura 1 Vantagens e desvantagens dos testes manuais

Com estas vantagens e desvantagens em mente e de forma a acompanhar a evolução, estes testes manuais foram sendo gradualmente implementados de forma automática com recurso a programação maioritariamente em JS.

2.2.3 Testes automáticos na Efacec

Além da automatização dos testes manuais já existentes, foi implementada uma estratégia, de modo que estes testes automáticos se inserissem e substituíssem progressivamente os testes manuais. Esta estratégia passou por cada vez que se criasse uma funcionalidade ou aplicação, se construísse simultaneamente um teste automático para testar a mesma. Deste modo, é possível identificar e corrigir os problemas mais cedo e de forma mais rápida.

Com os testes automáticos, à semelhança dos manuais, vêm também vantagens e desvantagens associadas, as quais estão enumeradas na Figura 2.

Figura 2 Vantagens e desvantagens dos testes automáticos

Atualmente, na Efacec, existem já vários testes automáticos com intenção de testar diferentes ferramentas a atuar na automação de sistemas de energia, no entanto, ainda não existem testes de localização de defeitos, ou seja, testes que validem que a localização de defeitos está a ser bem executada.

2.3. CLASSIFICAÇÃO DOS DEFEITOS

O conceito de defeito define-se como uma situação associada a uma mudança repentina e, por vezes, violenta do comportamento do funcionamento do sistema elétrico de energia, sendo que os defeitos se caracterizam pela sua origem, forma e duração. É uma anormalidade

que resulta na falha do circuito ou de um sistema da rede, requerendo uma quebra de ligação imediata através de aparelhos de corte apropriados [4].

Grande parte dos defeitos que surgem nas redes elétricas estão associados com a segurança e fiabilidade da distribuição de energia onde, se não houver ações de prevenção e proteção, várias falhas podem surgir, nomeadamente: situações de sobrecarga, intensidade da corrente superior ao limite seguro definido, etc. Ou seja, para que as interações desgovernadas entre os diferentes elementos da rede não causem instabilidade à mesma, devem ser assegurados alguns aspetos [5]:

- Balanço energético entre o consumo e a produção, para que não seja prejudicada a frequência da rede por desequilíbrios entre estes dois elementos;
- Não ultrapassar limites térmicos impostos devido ao aquecimento dos elementos da rede pela passagem de corrente.
- Seguir o critério "n - 1" que se baseia numa tentativa de prever e planear o inesperado, uma vez que quando há uma falha na rede elétrica esta se propaga rapidamente. Este é um dos princípios básicos aplicados para a segurança e fiabilidade de uma rede, e diz que se um componente falhar numa rede que opera no máximo nível de distribuição, a segurança desta rede deve ser, ainda assim, garantida, não existindo interrupções na distribuição nem a expansão da falha a outras zonas da rede [6].

A instabilidade é uma ocorrência que advém de incumprimento de (pelo menos) um destes aspetos, e que pode levar a *blackout* – interrupção total do fornecimento de energia elétrica –, que é a consequência mais séria uma vez que afeta todos os elementos da rede elétrica (com exceção dos *Uninterrupable Power Supplies* (UPS) que são equipamentos que fornecem energia de emergência quando há uma falha na rede elétrica que impede a sua alimentação) [7].

Os defeitos podem ser permanentes ou momentâneos. Os defeitos permanentes requerem uma intervenção humana, no entanto os defeitos momentâneos são erradicados pela religação automática das proteções, ou seja, abrindo o disjuntor da linha com defeito, e religando esta linha após o tempo de isolamento, confirmando assim que o defeito foi resolvido. É estimado que, em redes aéreas, 80% a 90% dos defeitos sejam de natureza momentânea, por serem normalmente causados por causas naturais (por exemplo uma

tempestade), enquanto nas redes subterrâneas aproximadamente 100% dos defeitos registados são de natureza permanente, por serem geralmente causados pelo rompimento ou corte de um cabo [8].

Há essencialmente dois tipos de defeito que podem acontecer e ser encontrados em sistemas de distribuição de energia, as falhas em série e falhas de derivação ou falhas *shunt* [9]. Há ainda um terceiro tipo de defeito, denominado defeito combinado, que é caraterizado por uma junção das duas falhas mencionadas inicialmente, ou seja, um defeito paralelo ocorre em simultâneo com um defeito série [10].

Estes defeitos podem então ser divididos por:

- Defeitos por circuito aberto ou defeito em série;
- Defeitos por curto-circuito ou defeito paralelo (*shunt*);
- Defeito combinado.

2.3.1 Defeito em série

Segundo a *International Electrotechnical Comission* (IEC) é um defeito no qual a impedância de cada uma das três fases não é igual, o que normalmente é causado por uma interrupção numa ou duas fases [11], ou seja, quando as linhas têm impedância de série desiquilibrada. Representa um condutor aberto e ocorre quando uma rede do sistema de energia tem uma linha partida ou impedância numa ou mais linhas havendo deslastre de carga. Estes defeitos têm caraterísticas como a subida de tensão, e redução de corrente nas fases com o defeito.

2.3.2 Defeito *shunt*

A IEC define um defeito *shunt* (ou defeito paralelo) como "um defeito caraterizado pelo fluxo de corrente entre duas ou mais fases ou entre fase(s) e a terra à frequência do sistema de energia associado" [12]. Os sistemas de distribuição experienciam regularmente este tipo de falhas paralelas. Relés *phase-overcurrent* e relés *ground-overcurrent* são comummente usados para detetar e isolar o circuito com defeito num sistema de distribuição. A caraterística a destacar neste tipo de defeito é o aumento da corrente e a redução no valor da tensão, ao contrário do que se vê nas falhas em série.

De entre os *shunt faults*, os diferentes defeitos são ainda mais categorizados dividindo-se também por defeitos simétricos e defeitos assimétricos. Quando o curto-circuito afeta as três fases este é considerado um defeito simétrico – neste caso, apesar do defeito, as três fases

mantêm-se equilibradas. Quando pelo menos uma fase não é afetada o defeito considera-se assimétrico – existe uma assimetria entre o valor da corrente das diferentes fases do sistema. A maior parte dos defeitos afeta apenas duas fases, inserindo-se desta forma nos defeitos assimétricos [13].

Para uma linha trifásica há os seguintes tipos de falhas *shunt* [14]:

2.3.2.1 Defeito Linha – Terra (DLT)

Também conhecido como defeito de curto circuito, ocorre quando uma fase da linha de transmissão estabelece contacto com a terra ou com o neutro. Dentro dos possíveis motivos para este acontecimento encontram-se ventos fortes, queda de árvores, tempestados, ou outros acontecimentos geralmente de origem natural.

70% dos defeitos de shunt são classificados como DLT que é simultaneamente o tipo de defeito menos grave. Os três tipos de defeitos de CC estão representados na Figura 3.

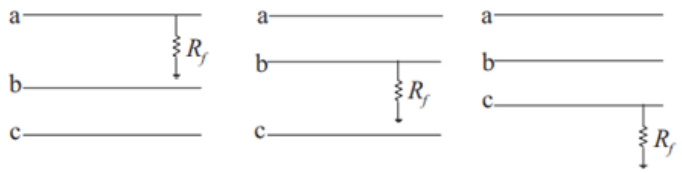


Figura 3 Representação gráfica de defeito linha – terra nas diferentes linhas (adaptado de [14]**).**

2.3.2.2 Defeito Linha – Linha (DLL)

Pode ser provocado quando dois condutores são curto circuitados e pode acontecer tanto em cabos aéreos como em transmissão subterrânea. Uma das caraterísticas do DLL é a grande magnitude da impedância de falha que pode variar amplamente, sendo difícil prever um limite inferior e superior. 15% das falhas na rede são consideradas DLL. Na Figura 4 estão representados os defeitos de linha a linha num sistema trifásico.

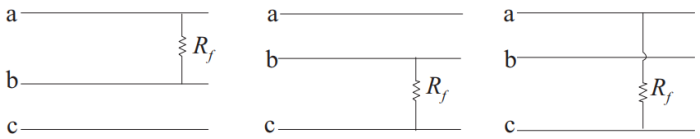


Figura 4 Representação gráfica de defeito linha – linha nas diferentes linhas (adaptado de [14]**).**

2.3.2.3 Defeito Linha – Linha – Terra (DLLT)

Ocorre quando, por exemplo, uma árvore caída conecta duas fases com a terra. Um DLLT é uma falha séria porque resulta numa assimetria significante que pode vir a tornar-se numa falha trifásica se não for resolvida num certo período de tempo – a este tipo de desenvolvimento de uma falha mais simples para uma falha mais complexa dá-se o nome de "falha em desenvolvimento" (*developing fault*), definida pelo IEC como uma falha de isolamento que pode começar com um defeito de fase-terra ou fase-fase e desenvolver-se para um defeito com duas ou três fases envolvidas [15]. 10% das falhas são DLLT. A Figura 5 representa este tipo de defeito envolvendo fases diferentes.

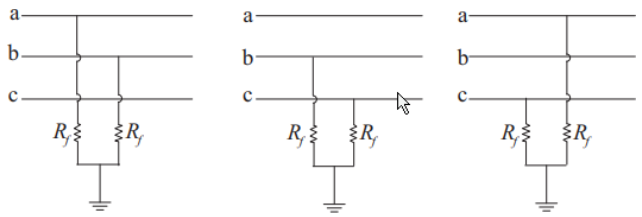


Figura 5 Representação gráfica de defeito linha – linha – terra (adaptado de [14]**).**

2.3.2.4 Defeito Linha – Linha – Linha – Terra (DLLLT)

Também é conhecido como falha simétrica. Pode ocorrer com a falha de um equipamento, queda de torre ou com a conexão de uma linha com as restantes fases. Num cenário real o DLLLT não existe regularmente, representando apenas 5% das falhas. É o tipo de defeito mais perigoso, e é causado por um CC em simultâneo entre as três fases e a terra. Neste erro os níveis de tensão igualam a zero e a corrente de falha é altíssima. Este defeito está representado na Figura 6.

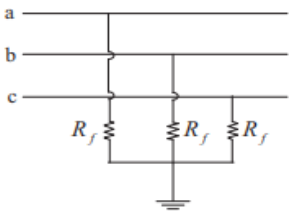


Figura 6 Representação gráfica de defeito linha – linha – linha – terra (adaptado de [14]**).**

Na Tabela 1 estão representados os defeitos paralelos por curto-circuito (*shunt faults*) categorizados por percentagem de ocorrência e, simultaneamente, pela gravidade do

respetivo defeito [14]. Esta gravidade do defeito pode ser expressa em função da magnitude da corrente de falha e do potencial de, a partir deste defeito, resultarem danos.

Tabela 1 Ocorrência e severidade de cada defeito (adaptado de [14]).

Tipo de Defeito	Ocorrência (%)	Severidade
Linha - Terra	75 - 80	Muito pouco severo
Linha - Linha	10 - 15	Pouco severo
Linha - Linha - Terra	5 - 10	Severo
Linha - Linha - Linha - Terra	2 - 5	Muito severo

Anteriormente foram enunciados alguns motivos e acontecimentos que levam à ocorrência destas falhas, no entanto estas também podem ser de origem material, ou seja, podem surgir a partir de falhas nos materiais que compõem a rede de distribuição sem que nada externo (como fatores naturais) as despoletem. Na Tabela 2 estão representados os elementos do sistema de distribuição nos quais é mais comum originarem este tipo de defeitos, e respetivas percentagens [14].

Tabela 2 Percentagem de ocorrência de defeito em cada elemento do sistema (adaptado de [14]**).**

Elemento do Sistema	Percentagem de Ocorrência (%)
Transformadores	10
Linhas aéreas	50
Cabos subterrâneos	9
Aparelhagem	12
Relés de PTs, equipamento de controlo, etc	12
Geradores	7

2.4. CONSEQUÊNCIAS DOS DEFEITOS

Com o intuito de reduzir o impacto dos efeitos na rede elétrica, é comum recorrer ao critério de segurança já mencionado anteriormente "n - 1", o qual consiste em manter o sistema em funcionamento de modo a não provocar falhas para os clientes/consumidores, mesmo que um elemento da rede deixe de operar corretamente. Ainda assim, mesmo recorrendo a esta solução, em casos mais graves podem existir consequências significativas provocadas por estas falhas.

Há quatro repartições feitas para as consequências destes defeitos [16]:

Consequências ocultas: por definição são defeitos sem impacto direto, mas que expõem a organização a múltiplas falhas e com possíveis consequências graves.

Consequências operacionais: quando a produção é afetada como uma consequência de uma falha, seja a nível de qualidade do produto, serviço ao cliente ou custos operacionais bem como no custo adicional de correção do erro.

Consequências de segurança e ambientais: se a falha colocar alguém em perigo de vida ou de lesão, significa que este defeito teve consequências de segurança. A ocorrência de falhas pode ainda causar choques elétricos a pessoas, cuja gravidade irá depender da corrente e da tensão no local da falha, podendo levar a lesões graves permanentes ou até mesmo morte [17]. As consequências ambientais surgem quando a falha origina violações de padrões ambientais definidos a nível regional, nacional ou internacional.

Consequências não operacionais: falhas das quais a consequência das mesmas não origina problemas de segurança nem produção e apenas envolve o custo direto de correção do defeito.

As consequências dos defeitos podem ainda ser divididas por dois critérios distintos, originando consequências que surgem no início do defeito (por exemplo, quebra de isolamento) e consequências que dependem da duração do defeito. A partir do instante em que uma falha se inicia, esta vai causar algumas consequências iniciais, que são consequências cuja gravidade ou existência não depende da duração da falha e, por isso, é difícil (senão impossível) fazer algo para as impedir/minimizar. No entanto, o sistema pode ser protegido contra consequências adicionais (consequências subsequentes) que, estas sim, dependem da duração da falha. Quanto mais rápida for eliminada a falha, menores serão os danos provocados. Estas consequências podem ser categorizadas da seguinte forma:

2.4.1 Consequências gerais

As consequências gerais dividem-se por sua vez em duas partes, as forças mecânicas e o stress térmico.

2.4.1.1 Forças mecânicas

Para condutores paralelos em sistemas monofásicos ou trifásicos, a força imposta num dos condutores é proporcional ao quadrado da corrente de pico. Posto isto, quando a corrente de curto-circuito é conduzida através das fases, irá haver uma força mecânica sobre os mesmos,

força essa que irá rapidamente crescer com o aumento das correntes de curto-circuito, por ser proporcional ao quadrado desta corrente. No acontecimento de um curto-circuito, a corrente de falha irá conter uma componente contínua em estado decadente que depende do momento de início do defeito. Deste modo, a corrente de falha é mais elevada no momento imediatamente a seguir ao início do defeito e o valor mais elevado da força imposta nos componentes do sistema elétrico é causado pelo valor mais elevado da corrente de pico. O equipamento presente no sistema elétrico é concebido para resistir a correntes de pico até um certo valor, pelo que a IEC publicou uma proposta de cálculo do valor standard da corrente de pico máxima permitida, que iguala este valor máximo a 2.5 vezes o valor eficaz da corrente de curto-circuito

2.4.1.2 Danos térmicos

Devido à resistência inerente aos condutores presentes no sistema elétrico, existem perdas por calor quando a corrente os percorre, perdas essas traduzidas pela conhecida lei de Joule – fenómeno no qual o fluxo de corrente através de uma substância causa a produção irreversível de calor com um valor de potência proporcional em cada ponto, à resistência e ao quadrado da corrente elétrica [18].

Uma vez que, como é enunciado a partir da lei de Joule, as perdas dependem do quadrado da corrente, estas correntes de falha dão origem a um aumento do calor. Dependendo do tamanho e do material do condutor, este aquecimento pode eventualmente levar ao derretimento do mesmo, uma vez que o equipamento presente no sistema elétrico é projetado para tolerar correntes de falha apenas por um certo período de tempo.

2.4.2 Falhas envolvendo arco elétrico

Se a falha ocorrer no exterior ou num ambiente não fechado, a corrente de falha é limitada devido à resistência do arco elétrico reduzindo os possíveis danos, ainda assim, um arco elétrico a arder irá eventualmente causar estragos à linha e a sua corrente de falha terá de ser eliminada. Se a falha acontecer em ambiente fechado, como um cubículo de aparelhagem, a limitação da corrente de falha devido à resistência do arco elétrico é desprezável, logo os estragos causados nesta situação serão acentuados. Primeiramente, a dissipação de energia irá causar o aquecimento do ar dentro do cubículo, a pressão dentro do mesmo irá aumentar

e uma onda de pressão irá atravessar toda a aparelhagem. Entre aproximadamente 7 ms a 12 ms depois do início do arco elétrico, a onda de pressão encontrar-se-á completamente desenvolvida [19], no entanto, os cubículos de aparelhagem modernos encontram-se equipados com válvulas de despressurização que abrem na tentativa de impedir alguns possíveis danos causados por este abrupto aumento de pressão. Se o arco não for interrompido a temperatura irá continuar a aumentar, o que dará origem ao derretimento dos condutores e a uma reparação aprofundada da aparelhagem, ou até mesmo a sua substituição [17].

2.4.3 Custo

As consequências iniciais de um defeito podem levar a, por exemplo, quebras de isolamento num cabo, levando a que o cabo necessite de reparo. Se a falha não for interrompida, o mesmo cabo ficará cada vez mais danificado podendo ser necessário proceder à substituição. Eventualmente a corrente de defeito afetará não só o cabo, mas também o restante equipamento que se encontre no caminho do fluxo desta corrente, como disjuntores, transformadores de potência, ou até mesmo geradores. Por vezes, incêndios resultantes destas correntes de falha resultam em equipamentos totalmente queimados, o que leva a danos irreparáveis [17]. Fazer o reparo ou substituição de um cabo pode não acarretar custos significativos, no entanto se se tratar da substituição de um transformador, aí os custos serão bastante mais avultados.

Outro aspeto inserido no custo reside no facto de que quanto maior forem os danos, mais moroso será o seu processo de reparação antes do sistema de distribuição poder voltar ao seu correto funcionamento. Dependendo do esquema do sistema de distribuição, mais ou menos clientes poderão ficar sem energia durante o tempo de arranjo.

2.5. SISTEMAS DE DETEÇÃO DE PASSAGEM DE DEFEITO

Um indicador de passagem de defeito é um equipamento estrategicamente posicionado ao longo de uma rede de distribuição para captura e estudo de sinais de corrente e tensão, com o intuito de alertar (à distância ou localmente) a ocorrência de uma falha na rede.

Alguns indicativos de existência de defeito podem ser o sobreaquecimento das linhas, o aumento excessivo da corrente ou uma queda de tensão elevada, uma vez que em falha o

valor da corrente e/ou da tensão é muito diferente do valor pretendido/nominal. Na oposição a estas ocorrências, os sistemas de proteção (disjuntores, seccionadores, etc) permitem a deteção de defeitos, desligam os equipamentos em falha de modo a não permitir que os estragos alastrem e limitam o seu impacto na rede elétrica [20]. Aqui, os sistemas de proteção seccionam a zona da rede em defeito com o intuito de o eliminar e, de seguida, tentam religar o sistema. Se após um número pré-determinado de sucessivas tentativas estas não surtem efeito, então conclui-se que o defeito é permanente, correndo o risco de não haver energia a chegar aos consumidores uma vez que o disjuntor de alimentação fica bloqueado. O papel das equipas de manutenção é proceder à localização desta falha, percorrendo inicialmente a linha principal de alimentação, seccionando esta mesma linha em partes através dos equipamentos de proteção a jusante do bloqueio e, com o intuito de religar o sistema, procedem à sua abertura. Quando a religação não tiver sucesso é porque o defeito se encontra a jusante da proteção em questão, no entanto, caso não se encontre o defeito é necessário continuar a procurar. Estas operações de abertura e fecho das proteções são comandadas pelo centro de operação, interligando todas as informações que lhe são transmitidas quer pelos equipamentos de proteção, quer pelas equipas de manutenção. Enquanto o defeito não for encontrado, se a linha principal já tiver sido pesquisada então passa-se à procura nos ramais, até que a falha seja localizada.

Os sistemas de proteção estão a evoluir e, cada vez mais a sua evolução é associada aos indicadores de passagem de defeito (IPDs), com um controlo progressivamente mais automatizado executado à distância, que consequentemente resulta numa diminuição no tempo de falha, uma redução de custos causados pelo tempo em que a rede se encontrou defeituosa, e uma redução de custos com as equipas de manutenção no terreno.

2.5.1 Caraterísticas dos IPDs

Os defeitos podem ter dois tipos de informação associada. A primeira auxilia na localização de defeitos quando os mesmos são permanentes, e a segunda ajuda a entender a origem dos defeitos que não são permanentes, sendo as caraterísticas dos defeitos registadas pelos IPDs. Estes dados contribuem para a melhor manutenção da rede, uma vez que facilitam a localização da zona da rede responsável por um dado número de defeitos [21]. Os indicadores de passagem de defeito devem funcionar assegurando a segurança de pessoas e bens, evitar danos a materiais da rede devido a elevadas temperaturas, e devem assegurar a

continuidade do serviço de energia localizando e eliminando a falha na rede [22]. Para o fazer e cumprir com todas as suas funções, os IPDs equipados na rede devem seguir com um conjunto de caraterísticas, nomeadamente:

- Sensibilidade, para que o aparelho não atue de forma desnecessária;
- Seletividade, de modo a ser possível eliminar apenas a parte em defeito, ou seja, em caso de falha permite apenas eliminar a parte da rede estritamente necessária para que o defeito seja isolado;
- Rapidez, de modo a reduzir as consequências de uma avaria cuja gravidade, normalmente, é proporcional ao tempo.
- Fiabilidade, para que sejam evitados disparos desnecessários e ser garantido o funcionamento adequado dos equipamentos em caso de falha. O estado normal destes equipamentos é não atuado mas em posição de alerta, ou seja, é o estado em que passam grande parte do tempo, mas na ocorrência de uma falha é importante que atuem corretamente;
- Baixo consumo, por razões de ordem económica.

A seletividade e rapidez acabam por ser caraterísticas dos IPDs que se contradizem, uma vez que caso o processo seja muito rápido pode não haver tempo suficiente para ser seletivo. Conciliar todas estas condições dos IPDs torna-se muito complexo uma vez que os pontos de qualidade de serviço frequentemente jogam contra os custos das instalações.

O princípio da deteção de defeitos baseia-se na comparação entre as informações obtidas, que são os dados de entrada dos indicadores, e os limites pré-estabelecidos. Após o tratamento destes dados de entrada, o IPD sinaliza para indicar o estado da zona da rede que está em monitorização. Esta sinalização pode ser feita de forma visual através de LEDs, por exemplo, ou podem ser transmitidos ao centro de controlo segundo a localização do defeito.

2.5.2 Funcionamento dos IPDs

Para bem gerir uma rede elétrica de distribuição é necessário que haja um bom tratamento de ocorrência de defeitos. Os passos a percorrer no tratamento de um defeito são, por ordem cronológica, a deteção da falha, a seleção da zona/ramal de rede em falha e a localização do defeito (de onde se parte para a reconfiguração da rede) [23] .

A deteção deste defeito deve ser o mais rápida possível, visto que os disjuntores serão abertos para desligar a zona defeituosa. A seleção do troço da rede com defeito possibilita isolar o defeito para ser posteriormente resolvido, o que permite também procurar uma solução de alimentação para os consumidores enquanto a falha não é resolvida. Isto permite que o defeito seja localizado mais rapidamente.

A localização exata do defeito na rede torna-se útil na fase seguinte, a fase de reconfiguração da rede, onde a localização do defeito pode ser mais lenta do que a deteção, de modo a privilegiar a precisão da mesma a fim de controlar os interruptores da rede de modo otimizado. Ainda assim, não pode ser deixado de parte o facto de uma localização muito demorada do defeito atrasar a restituição do serviço afetando a qualidade do mesmo e a energia não distribuída.

Os indicadores de passagem de defeito detetam os defeitos analisando os dados obtidos da tensão e da corrente que, quando ocorre um defeito, são analisados de modo a determinar o tipo de defeito. Se esta falha for prejudicial para o sistema, então os aparelhos de corte vão abrir para colocar esta zona da rede isolada. A não sinalização de um IPD é também um dado a considerar, visto que o princípio de funcionamento/utilização dos indicadores de passagem de defeito não direcionais diz que a falha se encontra entre o último IPD atuado e o primeiro não sinalizado, o que está representado na Figura 7.

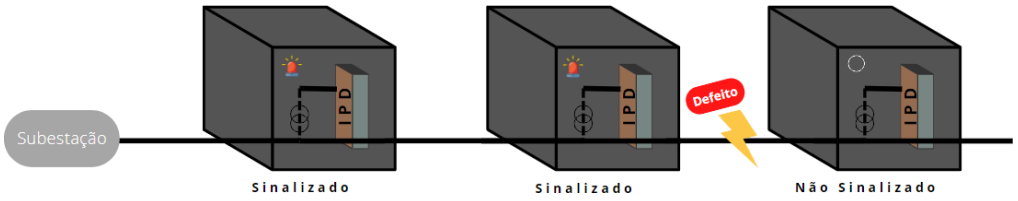


Figura 7 Princípio de utilização de IPDs não direcionais (adaptado de [24]**).**

Quando se abordam os IPD direcionais, esta é baseada num código de cores. Os postos de transformação a jusante e a montante do defeito são sinalizados pelas cores vermelha e verde, respetivamente, se este defeito for monofásico à terra, representado na Figura 8. Caso este defeito envolva mais do que uma fase, os indicadores colocados entre a saída da subestação e o defeito serão os únicos ativados.

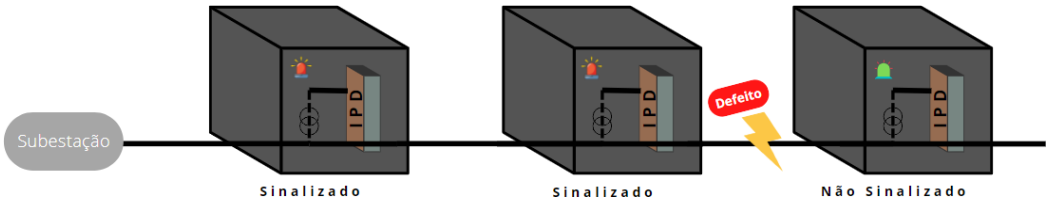


Figura 8 Princípio de utilização de IPDs direcionais (adaptado de [24]**).**

A zona delimitada pelos disjuntores e pelos IPDs é gradualmente reduzida com o intuito de isolar mais precisamente a falha, ou seja, são isolados alguns troços da rede e faz-se o restauro da alimentação do sistema para verificar se o defeito se encontrar naquela zona da rede ou não.

Para recuperar o serviço, repõe-se a alimentação nas zonas saudáveis da rede através do modo de alimentação principal ou através de uma forma de alimentação alternativa/secundária. Sendo que a zona defeituosa da rede está isolada do resto do sistema, uma equipa de manutenção vai proceder à sua reparação que, quando finalizada, permite fechar os interruptores e retomar a configuração inicial da rede.

Estes sistemas de deteção de defeito são importantes para a fiabilidade, qualidade e continuidade do serviço, visto que permitem reduzir o tempo alocado à localização das falhas e, como consequência, o tempo total do defeito na rede. Não obstante às vantagens associadas à redução do tempo de avaria, para os utilizadores e para as companhias elétricas, estas ferramentas têm mais benefícios [25]:

- Otimizar a gestão das equipas de manutenção;
- Reduzir custos de operação;
- Identificar zonas da rede problemáticas;
- Evitar que os equipamentos se deteriorem no decorrer do processo tradicional de localização de defeitos devido ao desgaste provocado pelas religações realizadas sobre defeitos permanentes.

Existem dois tipos de métodos de localização de defeitos a partir de IPDs: reconfigurando a rede ou através do cálculo da distância [22]. No método da reconfiguração da rede, esta possui aparelhos de comutação em pontos estratégicos que, após a deteção de um defeito permanente, permitem a reconfiguração da rede de modo a retomar a distribuição de energia aos consumidores afetados, e colocar a secção com falha sem alimentação (quando

necessário) para se eliminar a causa do defeito e reparar os equipamentos com danos. No entanto, esta pesquisa pode ser muito morosa, podendo levar algumas horas, por ser feita através de um método de tentativa – erro, o que aumenta os estragos dos materiais afetados. O método de localização de defeitos baseado no cálculo da distância do defeito obtém uma localização precisa da falha através do cálculo da distância entre o ponto de referência, geralmente representado pelo barramento da subestação, e o defeito.

2.6. MÉTODOS DE LOCALIZAÇÃO DE DEFEITO

As companhias elétricas adotaram vários métodos aplicados à deteção e localização de defeitos, métodos esses que vão desde a implementação de indicadores de passagem de defeitos nas redes de distribuição (abordados anteriormente), à adoção de metodologias analíticas apoiadas por SCADA. Estes métodos analíticos podem ser distribuídos por três grupos:

- Métodos baseados na impedância;
- Métodos baseados na propagação de ondas (*travelling-waves methods*);
- Métodos baseados em inteligência artificial (*knowledge-based methods*).

2.6.1 Métodos baseados em impedância fasorial

Quando se dá um curto-circuito numa determinada linha, esta traduz-se por um decréscimo no valor da tensão e um aumento na corrente, o que consequentemente origina uma queda do valor da impedância obtida a partir de cada um dos terminais da linha. A esta impedância determinada por estas medidas de corrente e tensão denomina-se impedância aparente [26].

Os métodos fundamentados no cálculo da impedância aparente são os mais usados por serem de fácil implementação e com soluções de baixo custo (quando comparados com outros métodos como o método *travelling-waves*) [14]. Assumem o pressuposto de que a impedância de uma parte da linha defeituosa está diretamente relacionada com a distância à falha nessa mesma linha e, sendo o comprimento da linha desde o ponto de referência até local da falha proporcional à impedância medida, ao conhecer a impedância da linha por unidade de comprimento, a distância da falha é facilmente obtida [27]. Estes algoritmos fazem uso de caraterísticas da linha – indutância, resistência, etc por unidade de

comprimento – e da corrente e tensão de uma ou mais extremidades da linha, para que se torne possível calcular a distância de uma das extremidades ao defeito. Podem ainda ser divididos em duas subcategorias: métodos de uma só extremidade ou método de duas extremidades, em que no primeiro caso se usa apenas uma subestação de tensão e corrente para localizar o defeito, e no segundo usa-se a tensão e a corrente nos dois extremos do sistema de distribuição para identificação da localização da falha [14].

A Figura 9 explica o conceito do método baseado em impedância usando um circuito modelo simples.

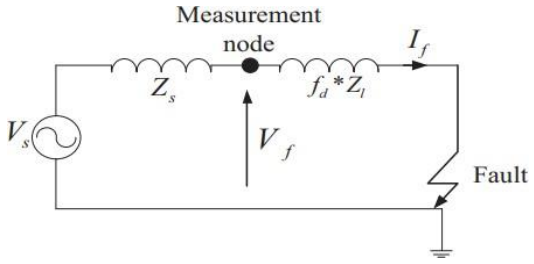


Figura 9 Circuito modelo simples de representação do método baseado em impedância aparente [14]**.**

V_f e I_f correspondem à tensão e à corrente, respetivamente, durante o defeito, Z_l é a impedância da linha por unidade de comprimento, f_d é a distância da falha desde o nó de referência (*Measurement node*), V_s é a tensão da fonte e Z_s é a impedância da fonte. Baseando-nos na lei de Ohm, a corrente e a tensão desde o nó de medida pode ser usada para encontrar a distância do defeito usando a equação 1 [14].

$$f_d = \frac{V1}{(If * Zl)} \quad (1)$$

São implementadas diferentes abordagens para encontrar a distância do defeito considerando os tipos de defeitos existentes e as diferentes resistências de falha. Métodos como o "Método da componente reativa" [28], "Algoritmo Takagi" [29] e o "Método Girgis" [30] foram desenvolvidos no passado com o objetivo de estimar a localização da falha em sistemas de distribuição radiais.

2.6.1.1 Algoritmos baseados em medidas de uma extremidade

Os métodos de localização de defeitos baseados em medidas de uma só extremidade estimam a localização de falhas considerando um *feeder* de distribuição ou transmissão a partir de uma das extremidades [28]. São obtidas as formas de onde de corrente e tensão durante a falha numa extremidade da linha, e são usadas para determinar a impedância aparente entre a localização da falha e essa extremidade.

As vantagens de usar métodos de uma extremidade refletem-se por serem métodos diretos de implementar, resultam em estimativas de localização de defeitos razoavelmente boas, e necessitam de dados e informação provenientes de apenas um extremo da linha. Os seus algoritmos são relativamente simples e não necessitam de um canal de comunicação remoto para recolha de dados, no entanto a precisão da estimativa é afetada pelo efeito combinado da resistência da falha e carga, erros de modelação, não-homogeneidade do sistema, e medições imprecisas [31].

2.6.1.2 Algoritmos baseados em medidas de duas extremidades

Os algoritmos de duas extremidades necessitam de dados de forma de onda capturados em ambos os extremos de uma linha aérea e são mais adequados para localizar falhas em redes de sistemas de transmissão onde as medições de vários monitores podem estar disponíveis. O princípio de localização de defeitos deste método é semelhante ao dos métodos de uma só extremidade, ou seja, usando a tensão e a corrente durante a falha para calcular a impedância aparente desde o nó de medida até à falha. São usadas medidas adicionais da extremidade remota da linha de transmissão, para eliminar quaisquer erros de reatância causados por corrente de carga, resistência de falha, ou por uma não homogeneidade do sistema e, por isso, são mais precisos do que os métodos baseados em apenas uma extremidade. Um canal de comunicação transmite os dados de um aparelho eletrónico inteligente (IEDs – *intelligent electronic devices*) para o outro. Alternativamente, os dados recolhidos por ambos os IEDs podem ser reunidos e processados numa localização central. Métodos como o "Método sincronizado de duas extremidades" [32], "método dessincronizado de duas extremidades" [33] e o "método dessincronizado só de corrente de duas extremidades" [34] são exemplos de algoritmos baseados em medidas provenientes de dois terminais da linha de distribuição.

2.6.2 Métodos baseados em ondas viajantes (*travelling-waves*)

Na ocorrência de um defeito dá-se a formação de uma onda que "viaja" em direção aos dois terminais da linha em questão, com uma velocidade próxima à velocidade da luz [35].

Para se representar a evolução direcional e temporal de ondas que se propagam entre terminais de uma determinada linha utiliza-se um diagrama denominado diagrama de Bewley que, não só, mas em específico neste método se torna útil [36].

À semelhança dos métodos baseados na impedância aparente, também nos métodos das *travelling-waves* se faz a distinção entre método das Ondas Viajantes (OV) nos dois extremos e métodos das OV num só extremo. Estes distinguem-se pelo cálculo do tempo de chegada da onda viajante a ambos os extremos da linha de distribuição, no caso do método das OV nas duas extremidades, e no tempo de chegada da onda numa extremidade e da sua reflexão no local do defeito, no caso do método das OV numa só extremidade.

No caso do método das OV nos dois terminais, este baseia-se na recolha dos tempos de chegada da onda viajante a ambos os extremos da linha de distribuição, após a ocorrência da falha [37]. A Figura 10 representa o diagrama de Bewley de uma linha de distribuição simples, de comprimento LL, e onde ocorreu um defeito representado por F, com tempos de chegada às extremidades *ts* e *tr*. M e LL-M é a distância de cada terminal ao defeito.

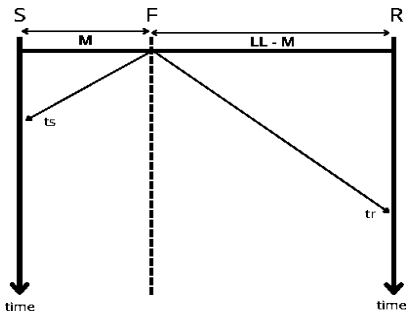


Figura 10 Diagrama de Bewley demonstrativo do método OV de dois extremos (adaptado de [38]**).**

No método das OV de uma extremidade, este assenta na diferença entre o tempo de chegada da primeira onda após o defeito ao terminal onde são feitas as medições, e o tempo de chegada da onda que reflete no local onde surgiu a falha, o que permite o cálculo da localização onde ocorreu o defeito [35]. Recorrendo novamente ao diagrama de Bewley,

representado na Figura 11, é possível observar as três situações distintas: defeito na primeira parte da linha, em que M < LL-M; defeitos a meio da linha, em que M = LL-M; defeitos na segunda metade da linha, em que M > LL-M. Neste último caso (gráfico mais à direita) as reflexões provenientes do terminal oposto da linha chegam primeiro ao local da falha do que as reflexões da extremidade a ser considerada, pelo que o algoritmo identifica em qual dos três cenários nos encontramos tendo em conta que a polaridade da onda viajante que reflete no terminal R é oposta à polaridade da onda que reflete no local onde ocorreu o defeito [39].

O aparecimento de um defeito é detetado calculando o módulo da diferença entre a última amostra da tensão pré-defeito e a amostra da tensão no momento do cálculo, onde, caso este valor seja superior a uma constante na ordem das milésimas é assumida a existência de um defeito.

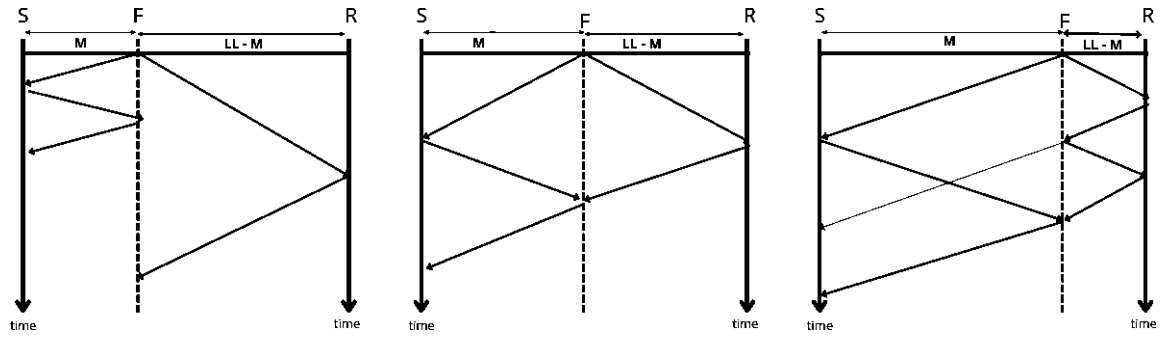


Figura 11 Diagrama de Bewley demonstrativo do método OV de um extremo (adaptado de [38]**).**

2.6.3 Métodos baseados em inteligência artificial

Os métodos baseados em inteligência artificial, também denominados métodos baseados em conhecimento (*knowledge-based methods*) ou métodos baseados em aprendizagem (*learning-based methods*), são outro tipo de métodos utilizados para localizar o surgimento de um defeito nas redes de distribuição.

A incerteza causada por caraterísticas das linhas, como desconhecer a resistência da falha ou o comprimento da linha, acoplado com a estrutura complexa dos sistemas de distribuição tende a tornar a localização dos defeitos a partir dos métodos baseados na impedância aparente e nas ondas viajantes menos precisa. Por isto, os métodos de localização de defeitos *knowledge-based*, nos últimos anos foram alvo de mais atenção por parte dos investigadores.

Estes métodos, por norma, necessitam de dados como o estado dos interruptores da subestação e das linhas de distribuição, condições atmosféricas, medidas das linhas, e informação disponibilizada pelos aparelhos de deteção de defeitos [14]. Em posse dessa informação, esta é analisada recorrendo a inteligência artificial para localizar o defeito. As redes neuronais artificiais (RNA) têm a habilidade inerente de reconhecer padrões e, por isso, são uma possibilidade na localização de defeitos na rede elétrica, onde o *input* de treino pode chegar a partir de medidas como a corrente, tensão, estado de um disjuntor, etc, e o *output* seria a localização da falha [40]. Têm uma fase de treino *offline* onde são empregues grandes conjuntos de cenários reais ou cenários simulados, funcionando como amostras para o treino da rede neuronal, os tempos de execução são curtos, e têm a capacidade de generalizar [40], o que é importante visto que uma das dificuldades encontradas nos métodos que utilizam RNAs é o facto de não haver um guia bem definido que indique o número ideal de camadas (entre a camada de *input* e a camada de *output*) nem do número de neurónios em cada uma destas camadas [41].

Existem ainda os sistemas de inferência *fuzzy* que são sistemas baseados na lógica *fuzzy* e na teoria de conjuntos fuzzy/de difusão – desenvolvida de modo a ter capacidade de lidar com problemas relacionados com julgamentos subjetivos, ambíguos e imprecisos, conseguindo quantificar a face linguística das preferências e informação disponível para tomadas de decisão em grupo ou individuais [42]. A logica *fuzzy* faz uso de termos linguísticos, o que torna a relação entre *inputs* e *outputs* mais fácil de existir. É uma hipótese científica que incorpora a parte ambígua da informação, mas simultaneamente emoldurando uma ideia ou relevância, tornando este método fundamentalmente uma forma de retratar a incerteza.

Por fim, existem os métodos *neuro-fuzzy*, nomeadamente o método *Adaptive Neuro Fuzzy Inference System* (ANFIS) que é um método híbrido – composto por duas ou mais técnicas de modelação, com objetivo de atingir maior capacidade de aprendizagem, interpretação, generalização e estimativa [43] –, que concilia conceitos provenientes das redes neuronais artificiais e da lógica *fuzzy*.

3. TESTES EXECUTADOS MANUALMENTE

No presente capítulo encontra-se descrita a zona da rede escolhida para testar a ferramenta de localização de defeitos. Foi com base nesta que os cenários de teste foram idealizados, os quais também se encontram devidamente explicados neste capítulo. Estes foram, primeiramente, executados de forma manual e individual com o intuito de validar o cálculo da zona provável de defeito e para verificar que os resultados obtidos correspondiam aos resultados esperados para cada cenário.

São ainda descritos os equipamentos da rede manipulados, o modo de execução dos testes, e os resultados obtidos nestes testes.

3.1. ESCOLHA DA ZONA DA REDE

Para poderem ser desenvolvidos testes de localização de defeitos, é necessário primeiramente escolher uma zona da rede a ser testada. Para o projeto em questão foi escolhida uma zona da rede com as caraterísticas elétricas da E-REDES, de modo que as mesmas estivessem bem definidas e já devidamente validadas. A subestação (SE) que

contém a zona da rede escolhida, denomina-se “SE ALTO DE SÃO JOÃO” e localiza-se na zona de Coimbra. A localização e representação da rede escolhida estão representadas na Figura 12, onde a mesma foi ampliada.

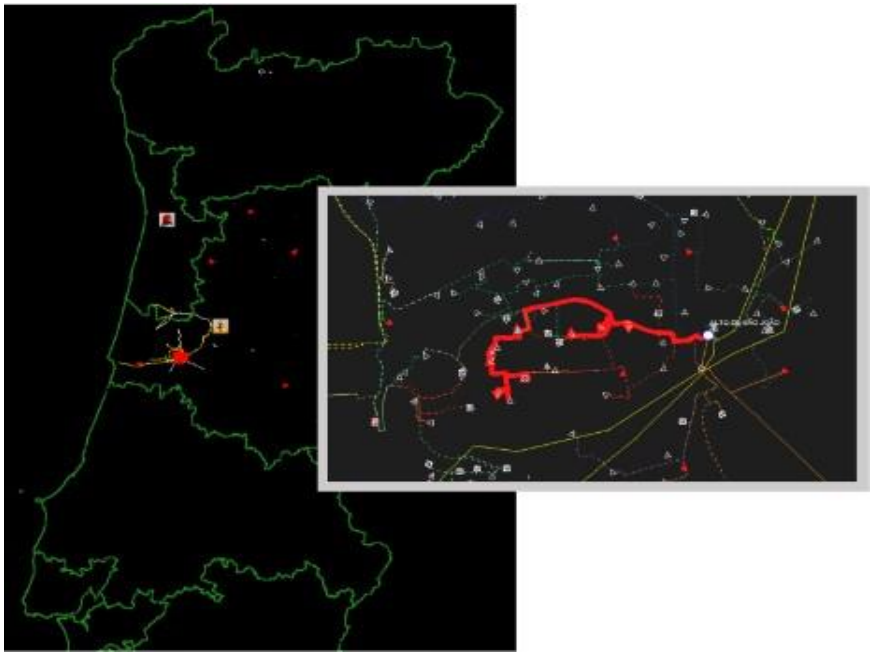


Figura 12 Localização e representação da rede escolhida.

Esta rede foi escolhida em conjunto com profissionais experientes e dominantes desta área. É uma zona com um grau de complexidade adequado para a testagem da FLF, e possui um conjunto de IPDs já configurado, bem como as entidades de medidas necessárias para o desenvolvimento de testes com distâncias de defeito. A Figura 13 representa a SE mencionada anteriormente onde se encontra circundada a vermelho a saída para a zona da rede escolhida denominada “SE ALTO DE SÃO JOÃO – CONTINENTE”, e a Figura 14 representa o esquemático da zona da rede onde vão ser desenvolvidos estes testes. Na Figura 14, além da representação da zona da rede escolhida, estão também circundados a azul os disjuntores da rede que têm IPDs associados, a cor-de-rosa o disjuntor de cabeceira que possui entidades de medida (por exemplo distância de defeito) associadas, e a amarelo a sinalização dos IPDs.

Figura 13 Subestação do alto de São João.

Figura 14 Esquemático da zona da rede escolhida – SE ALTO DE SÃO JOÃO – CONTINENTE.

3.2. MANIPULAÇÃO DE ENTIDADES

Na zona da rede definida para o desenvolvimento e execução dos testes, encontram-se presentes vários equipamentos que foram manipulados para simular defeitos, dos quais se destacam disjuntores, indicadores de defeito e medidores de distância.

Os disjuntores, tanto o disjuntor de cabeceira como os restantes disjuntores posicionados ao longo da zona da rede, têm quatro estados possíveis associados, como se verifica a partir da Figura 15, que representa a janela de imposições possíveis ao estado do disjuntor, no ScateX. Há dois estados de anomalia (estados 0 e 3), o estado desligado (disjuntor aberto, estado 1) e o estado ligado (disjuntor fechado, estado 2). Para o desenvolvimento do projeto foram tidos em conta os últimos dois estados.

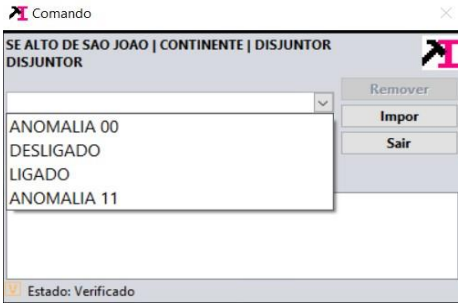


Figura 15 Estados possíveis dos disjuntores da rede.

No caso dos indicadores de defeito, apenas têm dois estados possíveis, também já mencionados anteriormente, representados na Figura 16. O estado normal (estado 0) significa que não detetou qualquer defeito, e o estado atuado (estado 1) é o estado em que um indicador de defeito sinaliza a existência de um defeito.

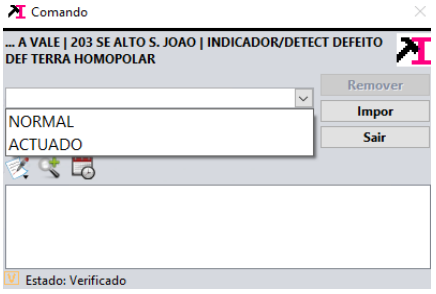


Figura 16 Estados possíveis dos IPDs.

Além destes equipamentos são também manipulados *bits* de qualidade que podem acionar e/ou desativar caraterísticas de cada elemento da rede. *Bits* de qualidade são *bits* que estão associados às diferentes entidades da rede, encontram-se na memória partilhada, e classificam a qualidade da medida/digital/contador que chegou ao centro de comando, bem como identificam comportamentos diferentes no

sistema (por exemplo, quando se altera manualmente o estado de um disjuntor, esta alteração denomina-se imposição, logo o *bit* que aciona a *tag* de imposição vai ser ativado). No desenvolvimento do projeto, foi manipulado um desses *bits*, de modo que dependendo do teste a ser efetuado, uma determinada entidade fosse (ou não) tida em consideração – o *bit* ativado denomina-se "*bit* de *manual* e *auto offscan*". De seguida encontram-se alguns exemplos de *bits* de qualidade, com o mais importante para o projeto sublinhado:

- *Bit* 64 – Acionar (64 64) ou desativar (0 64) o *bit* de falha de comunicações;
- *Bit* 96 – Acionar (96 96) ou desativar (0 96) os *bits* de *auto* e *manual offscan*;
- *Bit* 262144 – Acionar (262144 262144) ou desativar (0 262144) o *bit* de telemetria inválida.

Além das entidades digitais existem também entidades de medidas (analógicas), controlos e contadores, dos quais se destacam as primeiras. Estas foram-nos úteis por permitirem a manipulação de medidores de distância, indicando a este elemento uma distância de erro pretendida para execução de cenários com este tipo de medidores. Nesta rede apenas existem entidades de medida no disjuntor de cabeceira, pelo que estas distâncias de erro são associadas apenas a esse disjuntor.

Como dito anteriormente, os cenários idealizados começaram por ser executados manualmente para se verificar que os resultados obtidos eram os esperados. Para isso, foram elaborados *scripts* dbgm, que são ficheiros de código desenvolvidos a partir de comandos dbgm. Estes comandos dbgm não são considerados uma linguagem de programação propriamente dita, mas são uma ferramenta desenvolvida pela Efacec que simula a receção de telemetria medida pelo terreno, alterando o estado de medidas, digitais, contadores e controlos, bem como os *bits* de qualidade dessas entidades.

3.3. INDICADORES DE DEFEITO

Os indicadores de defeito foram um dos elementos a serem utilizados para a criação de cenários de defeito e testagem da ferramenta de localização de defeitos.

3.3.1 Lógica de funcionamento

A lógica de funcionamento dos indicadores de defeito já foi explicada anteriormente, no entanto a mesma baseia-se na localização da falha entre o último IPD atuado e o primeiro não sinalizado, o que está representado na Figura 7 e Figura 8.Os detetores de defeito presentes na rede elétrica dão-nos informação relativamente à corrente que percorre cada um destes elementos. Estes componentes têm dois estados possíveis, estado "Atuado" e estado "Normal", em que cada um deles nos dá a informação respetiva relativamente à corrente que os percorre. Quando um detetor de defeito se encontra não atuado, este representa o estado ideal e pretendido, uma vez que significa que nenhuma corrente de falha percorreu aquela zona ou aquele equipamento. No entanto, quando o detetor de defeito se encontra atuado, significa que detetou uma corrente de falha a atravessar o componente.

3.3.2 Cenários de teste idealizados

Os *scripts* desenvolvidos manipulando indicadores de defeito têm a intenção de testar a ferramenta de localização de defeitos consoante o funcionamento conhecido e explicado previamente dos IPDs. Para isso foram pensados e desenvolvidos três cenários diferentes utilizando IPDs, com o objetivo de colocar a FLF sobre testes cujo resultado esperado fosse diferente. Os cenários que se seguem resultam de manipulações dos estados destes equipamentos presentes nas linhas de energia em questão, com o intuito de comparar os resultados obtidos com os resultados esperados.

Assumindo que o nome/descritivo dos disjuntores representados e visíveis na Figura 17 dão nome também aos IPDs presentes em cada um deles, segue a associação desses nomes a números inteiros (também representados a azul na imagem) para mais fácil compreensão e identificação ao longo da descrição de cada cenário. Além da associação a números inteiros, é apresentada também (entre parênteses) a *tag* de cada IPD (e disjuntor associado) de modo a facilitar a interpretação de linhas de código enunciadas posteriormente:

- 203 SE ALTO S. JOAO – 1 (FD328A2203)
- 204 PS CBR 0732 – 2 (FD328A2204)

- PT CBR 0565 – 3 (FP32352202)
- PT CBR 0756 – 4 (FP32352201)
- 201 PS CBR 0587 – 5 (FD328A2201)
- PT CBR 0565 – 6 (FP32B02205)

Além desta identificação, na mesma figura estão também representadas as diferentes linhas que compõem a zona da rede em questão. Apesar da sua separação não se encontrar bem definida por barramentos, em alguns casos existe mais do que uma linha entre barramentos. Esses casos estão representados na Figura 17 com as respetivas divisões a cor de laranja.

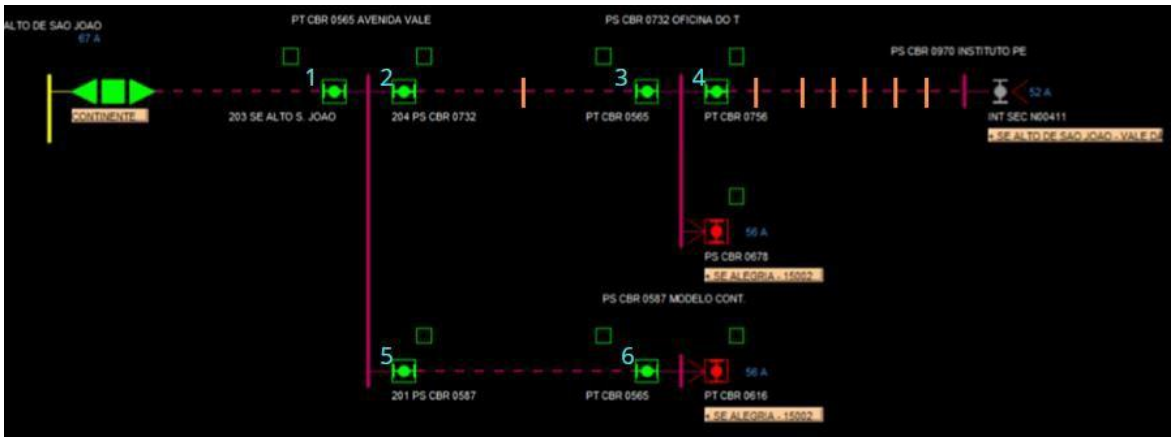


Figura 17 Representação das diferentes linhas da rede escolhida.

3.3.2.1 Cenário 1 – DF1+2

No primeiro cenário testado, foram ativados os IPDs 1 e 2, resultando na Figura 18. Nessa mesma imagem verifica-se a sinalização dos IPDs (círculo amarelo e preto junto aos disjuntores 1 e 2) e a abertura dos dois primeiros disjuntores (os acionados) e do disjuntor de cabeceira. Daqui, e sabendo o princípio de funcionamento dos IPDs, a zona de defeito esperada seria após o disjuntor 2, isto porque, como se sabe, a falha encontra-se entre o último IPD sinalizado e o primeiro IPD não sinalizado – "A não sinalização de um IPD é também um dado a considerar" –, ou seja, neste caso seriam as linhas entre os disjuntores 2 e 3.

Figura 18 Atuação do cenário DF1+2.

A Figura 19 e a Figura 20 representam as *tags* dos elementos atuados, em que o identificador que termina em -IT-DF representa o detetor da falha, o identificador acabado em ITEST é o interruptor de estado (ou disjuntor), e, na Figura 20, o identificador que termina em DDHIF é a sinalização do indicador de defeito. Em ambas as imagens, o estado dos três elementos é o inicial, isto é, antes de ser despoletado o defeito.

Figura 19 Identificador e descritivo de IPD e disjuntor.

Figura 20 Identificador/tag de sinalização de IPD.

Como referido previamente, para manipular estes elementos e executar estes cenários de teste manualmente, recorreu-se a ficheiros de código com comandos dbgm. O *script* deste cenário de teste encontra-se de seguida.

```
wtag dig FSAJO-2212-DJEST 1
wtag dig FSAJO-2212-DJEST 0 96

wtag dig FD328A2203-ITEST 1
wtag dig FD328A2203-ITEST 0 96
wtag dig FD328A2204-ITEST 1
wtag dig FD328A2204-ITEST 0 96

wtag dig FD328A2203-IT-DF 0 96
wtag dig FD328A2203-IT-DF 1
wtag dig FD328A2204-IT-DF 0 96
wtag dig FD328A2204-IT-DF 1

wtag dig FD328A2203-DDHIF 0 96
wtag dig FD328A2203-DDHIF 1
wtag dig FD328A2204-DDHIF 0 96
wtag dig FD328A2204-DDHIF 1
```

Nas duas primeiras linhas representadas é manipulado o estado do disjuntor de cabeceira (*tag* `FSAJO-2212-DJEST`) abrindo o mesmo, e também os seus *bits* de qualidade (desativa os bits de *offscan*). Nas restantes linhas temos a manipulação do estado e dos *bits* de qualidade dos disjuntores, IPDs e respetiva sinalização, abrindo os primeiros e atuando os restantes. A expressão `wtag`, significa *write tag*, ou seja, indica que se pretende escrever naquela *tag*, e `dig` é por se estar a interagir com entidades digitais.

Foi verificado o resultado obtido pela FLF no cálculo da zona provável de defeito, que está representado na Figura 21, comparando-os com os resultados esperados enunciados anteriormente. A linha amarela é apresentada pelo *Guided User Interface* (GUI) e representa a zona provável de defeito calculada, a qual valida a lógica de funcionamento dos IPDs.

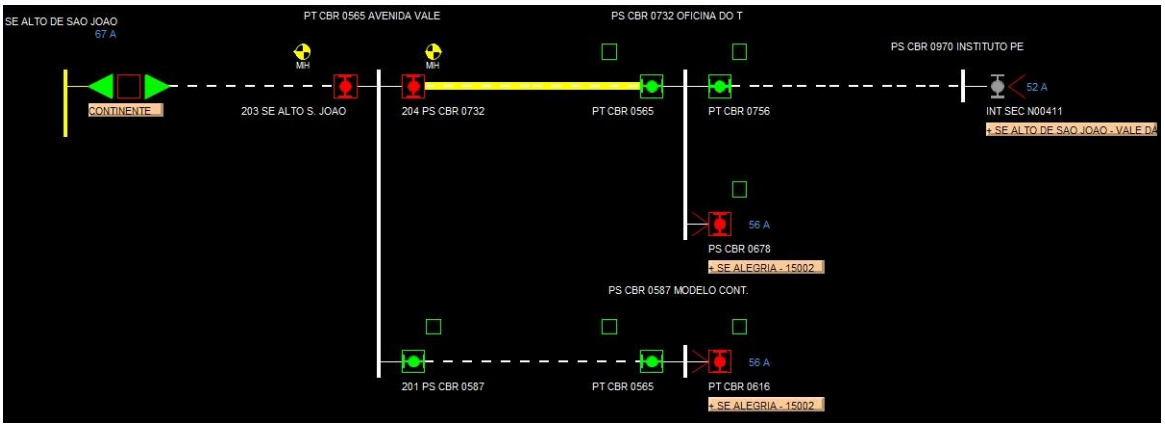


Figura 21 Zona provável de defeito para o cenário DF1+2.

3.3.2.2 Cenário 2 – DF1+2+5

No segundo cenário, atuou-se o indicador de defeito 5, para além dos indicadores 1 e 2 já atuados no cenário anterior, verificando a sua ativação a partir da Figura 22. Uma vez mais, conhecendo o princípio de funcionamento dos IPDs, neste cenário pretendeu-se obter uma zona provável de defeito ambígua, ou seja, não tão explícita e conclusiva, de modo a verificar a sinalização dessa mesma zona pela FLF. Assim, com os IPDs 2 e 5 ativos , espera-se que haja sinalização de dois ramais diferentes como zona provável de defeito – as mesmas linhas referidas no cenário anterior, e a linha entre os disjuntores 5 e 6.

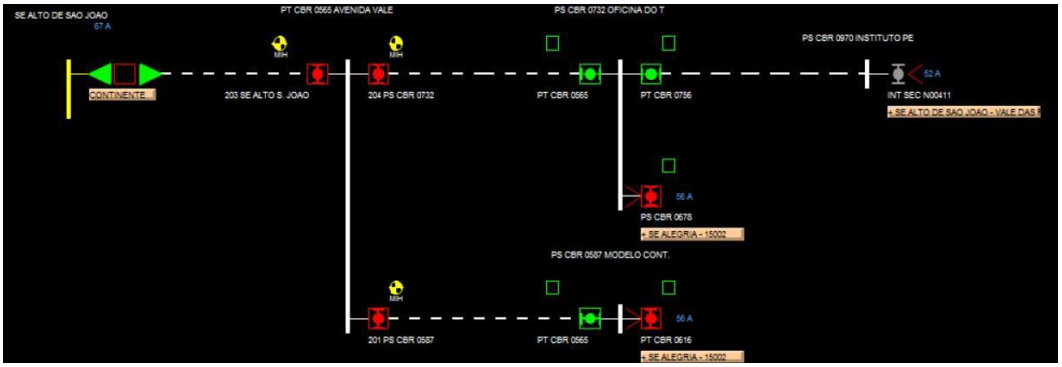


Figura 22 Atuação do cenário DF1+2+5.

Novamente, e seguindo a lógica explicada no cenário anterior, foram abertos os disjuntores em questão e ativados os IPDs associados a esses mesmos disjuntores, bem como a devida sinalização, e foi aberto o disjuntor de cabeceira. Os resultados obtidos foram devidamente analisados e confrontados com os resultados esperados, provando, uma vez mais, o correto funcionamento da ferramenta de localização de defeitos, como se vê a partir da Figura 23.

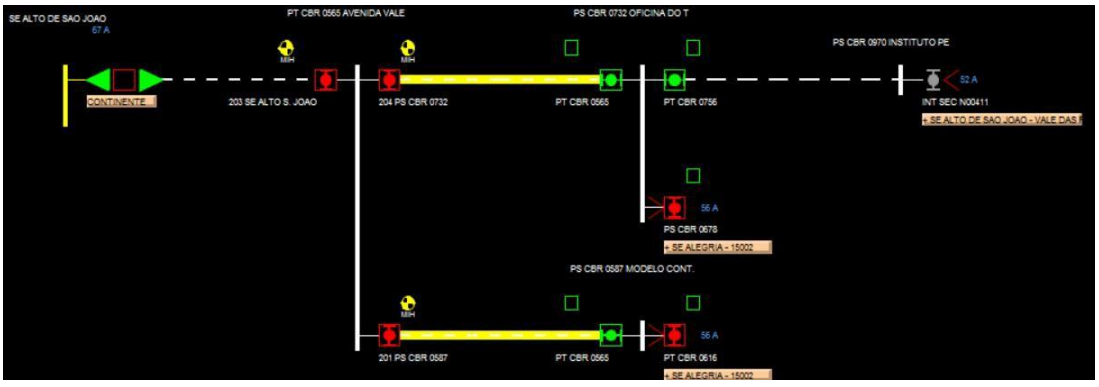


Figura 23 Zona provável de defeito para o cenário DF1+2+5.

3.3.2.3 Cenário 3 – DF1+2+3+4

No terceiro e último cenário testado apenas com IPDs, foram ativados os indicadores de defeito 3 e 4, para além dos dois primeiros. Este cenário foi idealizado de modo que o resultado obtido para zona provável de defeito fossem as 7 linhas presentes a seguir ao disjuntor 4, a última zona daquele ramo onde não está presente qualquer IPD no extremo final do mesmo. Apesar de não haver um IPD no fim dessas linhas, esse é o resultado esperado, por ser o ramo que se segue ao último IPD acionado. Os equipamentos atuados encontram-se representados em conjunto com a zona provável de defeito calculada na Figura 24.

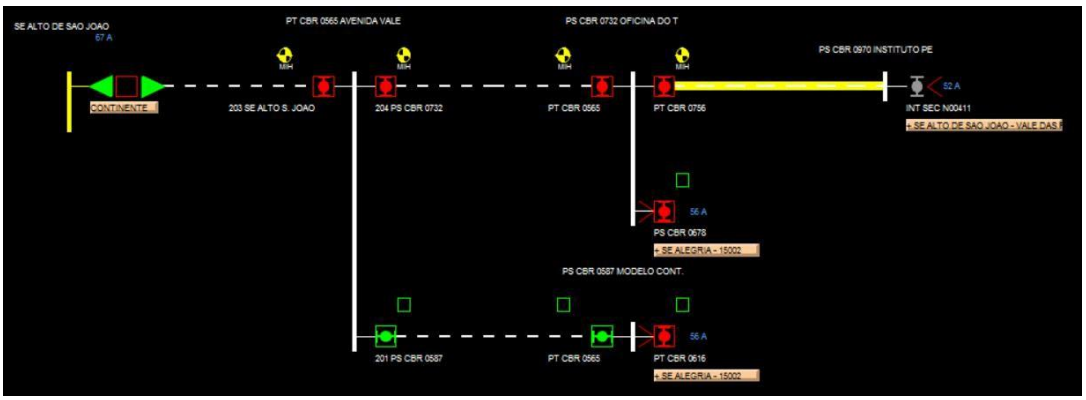


Figura 24 Zona provável de defeito para o cenário DF1+2+3+4.

3.4. MEDIDAS DE DISTÂNCIA

Além dos IPDs, também os indicadores/medidores de distância foram elementos a ser manipulados na criação de novos cenários de defeito para teste da FLF.

3.4.1 Lógica de funcionamento

Os indicadores de distância presentes na rede elétrica dão-nos informação relativamente à distância a que um defeito se encontra do disjuntor em que estão inseridos, com uma margem de erro definida, neste caso de 10%, a mais e a menos. Para os cenários de teste que irão ser enunciados a seguir, esta distância é relativamente ao disjuntor de cabeceira, uma vez que é o disjuntor que tem indicadores de distância associados.

A distância do defeito é uma medida analógica que, após obtida e calculadas as margens de +10% e -10%, nos dá a indicação da(s) zona(s) de defeito mais provável(eis). De salientar que na FLF, quando é detetado um erro numa determinada zona, não é sublinhada apenas a área calculada do defeito, mas sim toda a linha em que essa área se encontra.

3.4.2 Cenários de teste idealizados

Os seis cenários de teste que de seguida são explicados foram desenvolvidos com o intuito de testar a FLF ao nível dos indicadores (ou medidas) de distância e de a submeter a cálculos de diferentes zonas prováveis de defeito. Foram manipuladas a entidade digital que define o estado do disjuntor de cabeceira, a entidade de medida que define a distância (DJDST), e os *bits* de qualidade da entidade DJZRA, que corresponde à resistência do defeito, e da entidade DJZRS que representa a reatância do defeito, de modo que as mesmas não fossem consideradas. Após execução dos cenários foram comparados os resultados obtidos com os esperados.

Nos testes com IPDs abordados no subcapítulo 3.3, os comprimentos de cada linha da rede onde os testes são executados, não são importantes. Isto acontece uma vez que com cenários apenas de IPDs, a FLF não define (por não haver informação suficiente) distâncias mínimas e máximas para a zona provável de defeito, apenas se obtém a informação de quais as linhas com defeito, ou seja, são consideradas as linhas como um todo na seleção da zona provável de defeito. Nos cenários com medidas de distância o resultado obtido pela FLF é mais concreto, sendo calculadas distâncias mínimas e máximas para a zona provável de defeito. Estas distâncias mínimas e máximas vêm em percentagem, isto é, a partir da FLF conhece-se um intervalo de percentagens em que é provável que o defeito se encontre.

Posto isto, e recorrendo novamente à Figura 17, encontram-se listadas em baixo as linhas com o respetivo comprimento. A numeração das linhas foi feita olhando para o esquemático da rede como num sistema de escrita e leitura *left-to-right* (LTR) – da esquerda para a direita e de cima para baixo.

- 1ª linha – 678 metros
- 2ª linha – 617 metros

- 3ª linha – 155 metros
- 4ª linha – 226 metros
- 5ª linha – 11 metros
- 6ª linha – 93 metros
- 7ª linha – 82 metros
- 8ª linha – 295 metros
- 9ª linha – 11 metros
- 10ª linha – 287 metros
- 11ª linha – 349 metros

3.4.2.1 Cenários 1 e 2 – D200 e D500

O primeiro e segundo cenários foram juntos num só capítulo uma vez que os resultados esperados para estes, são os mesmos. Como dito anteriormente, nestes cenários de distâncias o único estado de disjuntor manipulado foi o do disjuntor de cabeceira uma vez que, por ser o disjuntor que recebe a medida de distância de erro, é também o disjuntor que abre para cortar a energia que percorreria o restante da rede. Além do estado desse disjuntor, foram alteradas as entidades analógicas do mesmo, de modo a serem simuladas uma distância de erro de 200 metros (D200) e 500 metros (D500). Para complementar foram ainda desativados os *bits* de qualidade referentes ao *manual* e *auto offscan*.

O estado da rede após a execução do cenário de teste D200 e antes de ser realizado o *trace* dos resultados obtidos está representado na Figura 25, na qual apenas se verifica a abertura do disjuntor de cabeceira e, com o intuito de tornar mais compreensível a entrada da informação relativamente à distância de 200 metros, foi colocada a janela correspondente ao estado das entidades desse mesmo disjuntor, onde se verifica a distância de 200 metros.

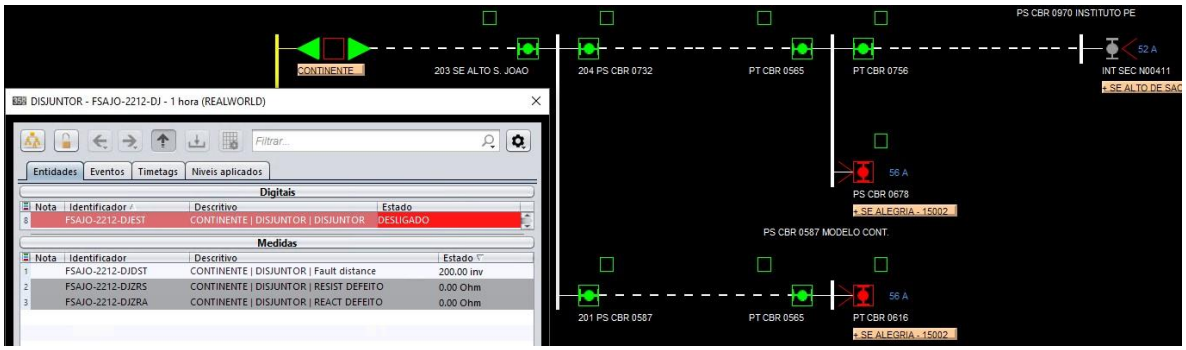


Figura 25 Atuação do cenário D200.

Na figura anterior, além dos 200 metros definidos, verifica-se um sombreado cinzento-escuro nas entidades relativas à resistência e reatância do defeito. Esse sombreado simboliza a ativação dos *bits* de qualidade referidos anteriormente, como se verifica na Figura 26 que representa a legenda (dividida em três devido à sua extensão) dos *bits* de qualidade e seus sombreados. A partir da legenda verifica-se que a ativação do *bit* de *auto offscan* é representada por uma cor lilás com letras brancas e que a ativação do bit de *manual offscan* é sombreada por cinzento-escuro com letras pretas. Na janela com o estado das entidades que se encontra na Figura 25, apenas se visualiza um dos sombreados, uma vez que, como foram acionados simultaneamente (96 96), as cores sobrepõem-se.

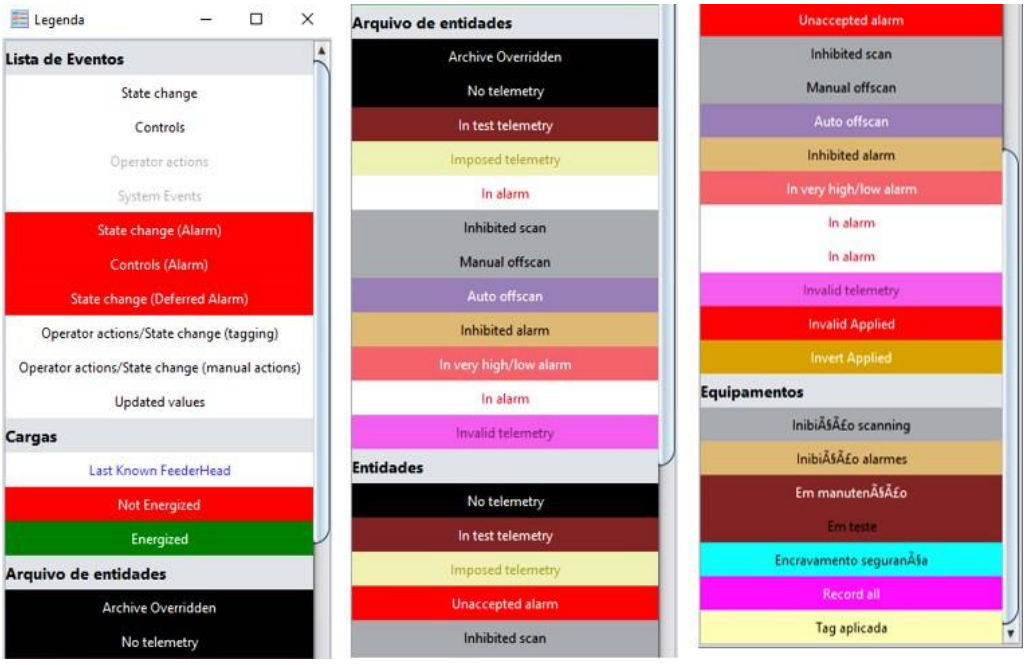


Figura 26 Legenda de *bits* de qualidade.

Para o cenário com a distância de defeito de 200 metros, aplicando as margens de erro de 10%, obtém-se um *range* provável de defeito entre os 180 metros e os 220 metros, o que se encontra representado na Tabela 3. Para o cenário 2, com 500 metros entre o disjuntor de cabeceira e o defeito, novamente aplicando as margens de erro, obtém-se um intervalo entre os 450 metros e os 550 metros, representados na mesma tabela. Estes valores representam os resultados esperados em ambos os casos, assumindo valores em metros.

Tabela 3 Intervalo de distâncias prováveis de defeito para os cenários D200 e D500.

Intervalo do Defeito		
-10%	0	10%
180	200	220
450	500	550

Como enunciado previamente, a FLF assinala as linhas prováveis na sua totalidade. Assim, verifica-se que em ambos os casos, a distância máxima para a zona de defeito não ultrapassa os 550 metros e, sabe-se ainda que o comprimento da primeira linha da zona da rede estudada é de 678 metros, pelo que o *trace* esperado para a zona de defeito deverá salientar a primeira linha da rede. A Figura 27 representa o resultado obtido pelo GUI para os dois cenários de defeito em questão, que confirma a resposta esperada.

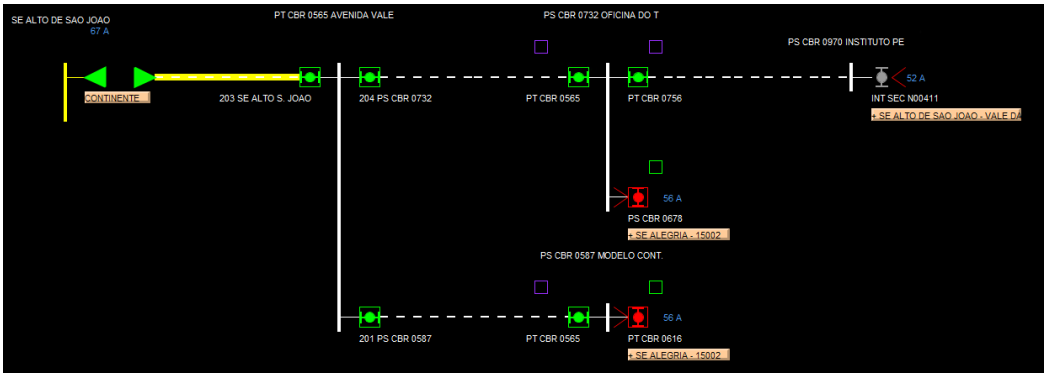


Figura 27 Zona provável de defeito para os cenários D200 e D500.

No entanto, idealmente a zona provável de defeito indicada pelo sistema deverá ser o mais precisa, bem como o mais esclarecedora possível. Para isso, o gestor de defeitos da *workstation* (WKS) não só possibilita a visualização de informação gráfica através do GUI,

como também dá informações mais esclarecedoras relativamente ao defeito, ou seja, fornece dados relativamente a telemetria, e localiza o defeito de forma mais concreta e menos abrangente. Deste modo, é possível analisar nos vários cenários as soluções obtidas e compará-las, uma vez mais, com os resultados esperados calculados previamente de forma manual.

Os dados de telemetria obtidos incluem a data e hora a que o defeito ocorreu, o equipamento defeituoso, o tipo de defeito – que neste caso apenas será um defeito de distância –, o valor do defeito (no caso dos cenários 1 e 2 apresentam 200 metros e 500 metros, respetivamente), a unidade (metro) e o estado do defeito.

Como referido anteriormente, os dados de localização obtidos a partir do gestor de defeitos contêm a identificação da linha e o intervalo de localização do defeito em valores percentuais relativamente à distância. Deste modo, antes da execução dos cenários idealizados foram calculados os valores que serviriam de comparação com os resultados obtidos. Calculados já os intervalos de distâncias presentes na Tabela 3 e conhecidos também os comprimentos das linhas da rede, converteu-se esses mesmos intervalos de distâncias para um intervalo percentual equivalente – para que a medida de comparação com os resultados do gestor de defeitos fosse a mesma.

Utilizando as equações 2 e 3 obteve-se o resultado posteriormente utilizado para validação das soluções obtidas pelo sistema, neste caso calculadas para o teste D200:

$$\%\,Mínima\,(D200) = \frac{Distância\ Minima\ do\ Defeito}{Comprimento\ da\ 1ª\ linha} * 100 = \frac{180}{678} * 100 = 26.55\% \quad (2)$$

$$\%\,Máxima\,(D200) = \frac{Distância\ Máxima\ do\ Defeito}{Comprimento\ da\ 1ª\ linha} * 100 = \frac{220}{678} * 100 = 32.45\% \quad (3)$$

Com o intervalo [26.55, 32.45] (%) calculado restou comparar com os cálculos efetuados pelo sistema, presentes na Figura 28. Nessa figura verifica-se que os valores obtidos coincidem com os valores calculados (considerou-se desprezável a diferença de 0.02% em

ambos os casos). Estes resultados são visualizados no gestor de defeitos do sistema, onde a informação sobre as falhas que ocorrem no sistema fica armazenada.

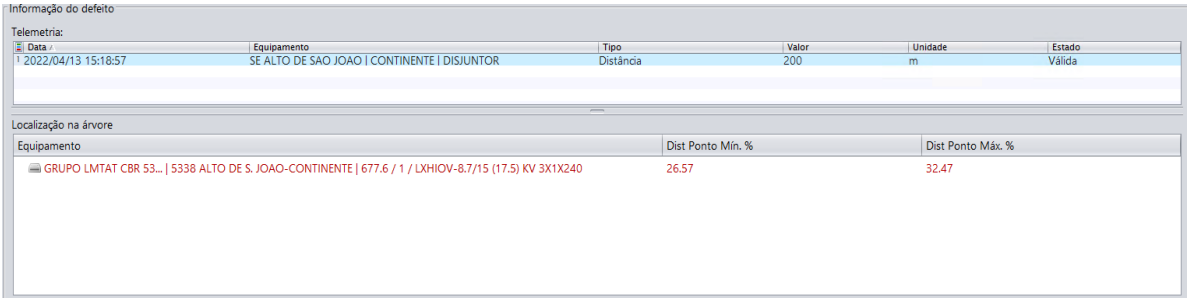


Figura 28 Resultados obtidos no gestor de defeitos para o cenário D200.

A representação gráfica da linha na Figura 27 é um complemento à localização do defeito, no entanto, caso este complemento não existisse, a partir da descrição do equipamento presente na Figura 28 seria possível associar este resultado às propriedades da própria linha, apresentadas na Figura 29, em que a sua descrição coincide com a descrição fornecida pelo gestor de defeitos.

Figura 29 Janela descritiva da linha 1 da rede.

3.4.2.2 Cenário 3 – D670

No terceiro cenário foi idealizado um defeito a surgir a uma distância de 670 metros do disjuntor de cabeceira. Aqui, calculadas as margens de erro, obtém-se um intervalo de distância entre os 603 metros e os 737 metros. Ao contrário dos cenários 1 e 2, a distância de 670 metros é suficiente para ultrapassar a primeira linha da rede, pelo que, a zona provável de defeito incluirá equipamentos a seguir ao barramento “PT CBR 0565 AVENIDA VALE”.

Uma vez que são duas as linhas que se iniciam nesse barramento (no esquema estão representadas uma em cima e uma em baixo), não há informação suficiente para a FLF assinalar apenas uma das duas. Assim, parte do resultado esperado para este defeito seria a sinalização de ambas as linhas. Contudo, uma parte do intervalo de distâncias calculado (603 metros), mantém-se inferior ao comprimento da primeira linha, pelo que também esta linha será assinalada como zona provável de falha. A Figura 30 demonstra a sinalização obtida para essa zona, que confirma o resultado esperado.

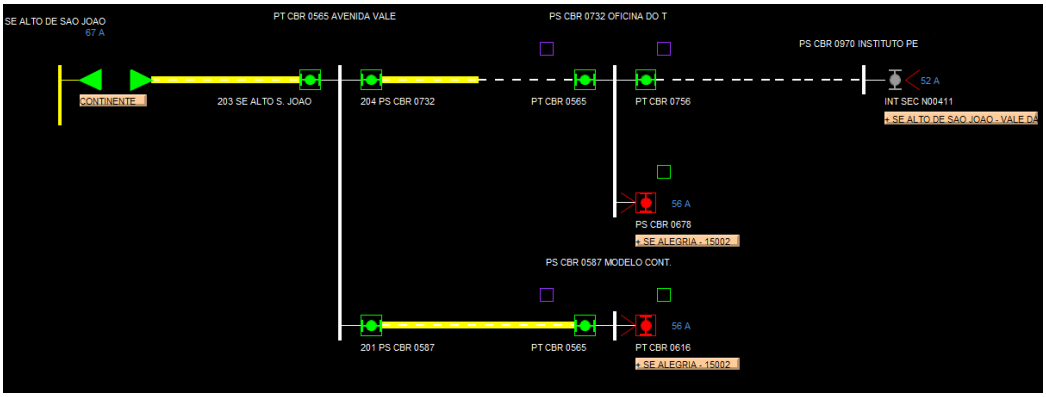


Figura 30 Zona provável de defeito para o cenário D670.

Neste caso temos que o *range* máximo do defeito vai até aos 737 metros, ou seja, 59 metros após a primeira linha. A segunda linha de cima assinalada a amarelo (linha 2) tem um comprimento de 617 metros, pelo que os 59 metros "excedentes" da linha anterior são 9,56% desta linha. No caso da linha em defeito de baixo (linha 11), esta tem 349 metros, pelo que 59 metros representam 16,91% da linha. A distância mínima do defeito são 603 metros, distância essa que ainda se encontra na primeira linha, mais propriamente a 88,94% da linha. Assim, temos os resultados obtidos no gestor de defeitos representados na Figura 31. A primeira linha tem uma área provável de defeito entre os 88,99% e os 100% – ou seja, desde os 603 metros até ao fim da linha – a segunda linha, tem uma área de defeito entre os 0% (início da linha) e os 9,63%, e a linha "de baixo" no esquema tem uma área provável de defeito entre os 0% e os 17,04% do comprimento dessa mesma linha. Uma vez mais, a baixa diferença percentual entre os cálculos e os resultados obtidos foi desprezável tanto neste caso como daqui em diante.

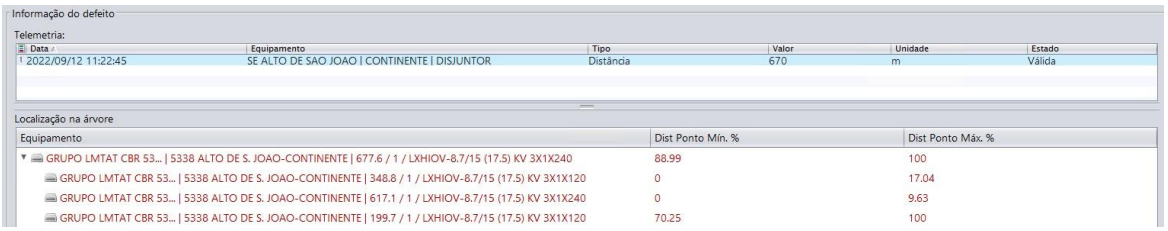


Figura 31 Resultados obtidos no gestor de defeitos para o cenário D670.

Neste cenário surge uma peculiaridade (que volta a surgir em cenários enunciados mais à frente) que é o facto de aparecer uma quarta linha identificada como zona provável de defeito, sem ter qualquer representação gráfica pelo GUI. Essa linha faz parte da subestação "ALTO DE SÃO JOÃO – CONTINENTE", no entanto não aparece na representação do esquemático da zona da rede escolhida, logo é considerada quando surge um defeito mas não é salientada graficamente. Esta linha tem uma outra individualidade, uma vez que tem os seus terminais "ao contrário" das outras linhas abordadas. Isto é, enquanto as linhas referidas anteriormente "iniciam" no "fim" da primeira linha, esta acaba no mesmo ponto que a primeira linha, pelo que tem de ser vista ao contrário. Em tentativa de explicar graficamente foi elaborada a Figura 32, em que a linha que não se encontra representada no GUI está identificada como "linha não representada".

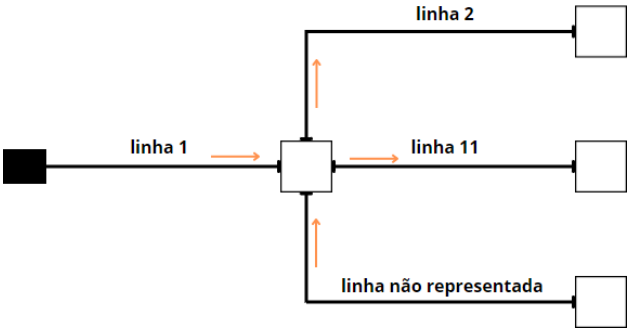


Figura 32 Explicação gráfica da presença e orientação da "linha não representada" na rede.

3.4.2.3 Cenário 4 e 5 – D800 e D1000

Pela mesma razão que os cenários 1 e 2 foram reunidos no mesmo capítulo, também os cenários 4 e 5 vão ser explicados em conjunto. Nestes, a distância de defeito foi novamente aumentada, desta feita para 800 metros e 1000 metros para os cenários 4 e 5, respetivamente. Com este aumento na distância, pretendeu-se excluir a primeira linha da rede, dos resultados.

Calculando o intervalo aglomerado das distâncias (distância mínima de D800 e distância máxima de D1000) para a zona provável de falha, obtém-se um intervalo entre os 720 e os 1100 metros. Com esse intervalo de distâncias sabe-se que a distância mínima é superior ao comprimento da linha 1, mas inferior ao comprimento da linha 1 e linha 2 somados, ou da linha 1 somado ao da linha 11, esperando-se que a zona provável de defeito se inicie na linha 2 ou 11. Seguindo o mesmo raciocínio, espera-se também que a zona máxima provável de falha se encontre numa das linhas mencionadas. Na Figura 33 encontra-se a representação das linhas assinaladas pelo GUI como zonas prováveis de defeito.

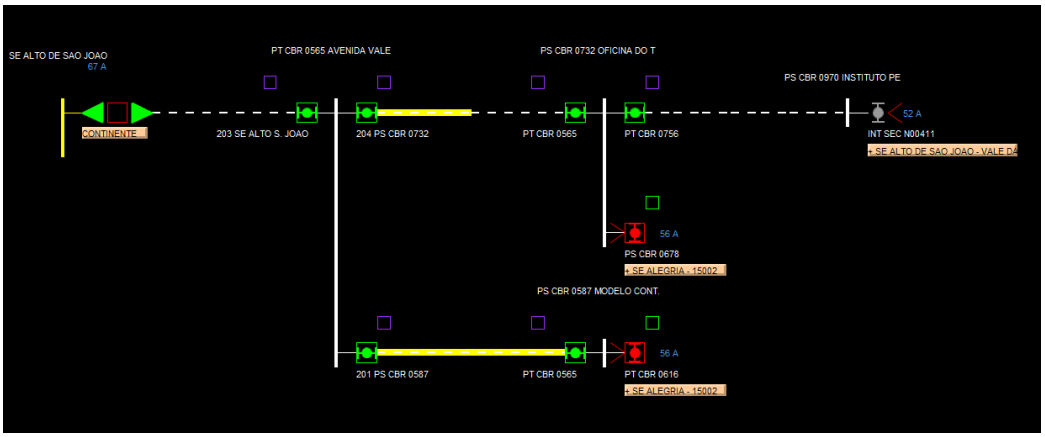


Figura 33 Zona provável de defeito para os cenários D800 e D1000.

Novamente, convertendo os intervalos de distância para percentagens das linhas, esperaram-se os seguintes resultados para o cenário D800:

- [6,81; 32,74] (%) para a linha 2;
- [12,03; 57,88] (%) para a linha 11;
- [1,00; 79,00] (%) para a mesma linha mencionada nos cenários anteriores, não representada no diagrama.

Para o cenário D1000 os intervalos obtidos foram:

- [35,98; 68,40] (%) para a linha 2;
- [63,61; 100] (%) para a linha 11.

Na verdade, os cálculos para o limite máximo da zona de defeito obtido no cenário D1000 para a linha 11 resultaram em 120,92%, no entanto, por não haver continuação da rede, foi considerado que o defeito se estende até ao fim da linha em questão. Na Figura 34 estão representados os resultados obtidos no gestor de defeitos para D800 (em cima) e D1000 (em baixo).

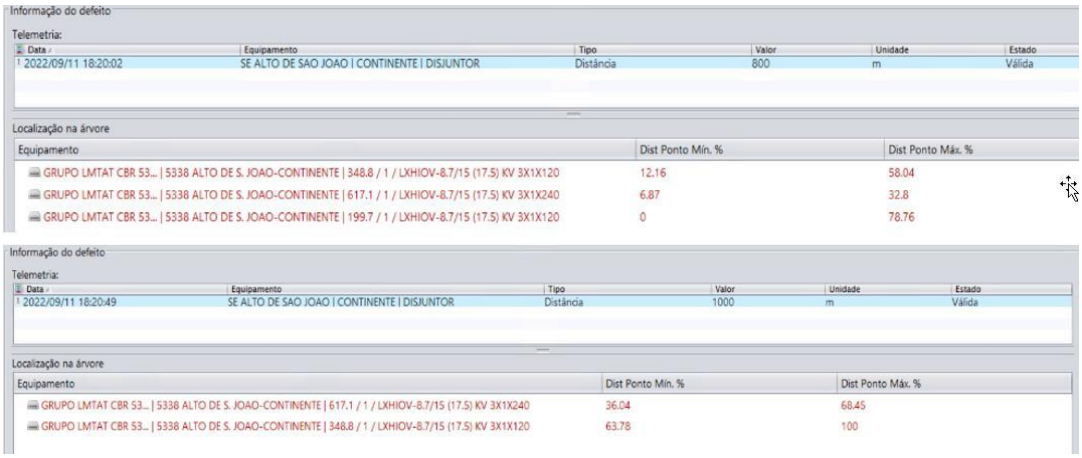


Figura 34 Resultados obtidos no gestor de defeitos para os cenários D800 e D1000.

3.4.2.4 Cenário 6 – D1200

O valor atribuído à distância de defeito no sexto e último cenário estudado apenas com os componentes de medição de distância, foi de 1200 metros. Este valor foi escolhido de forma que a linha 11 deixasse de ser incluída na zona de defeito, mas simultaneamente não ultrapassando o barramento PT CBR 0565.

Efetuando os cálculos necessários para saber que resultados esperar, obteve-se o intervalo de distâncias entre 1080 metros e 1320 metros. O comprimento das linhas, bem como a soma dos mesmos está representado na Tabela 4.

Tabela 4 Dimensão das linhas para cálculo da zona provável de defeito.

<table>
<tr><th></th><th colspan="4">Comprimento das Linhas</th></tr>
<tr><th></th><th>Linha 1</th><th>Linha 2</th><th>Linha 3</th><th>Total</th></tr>
<tr><td>Comprimento Parte Superior do Diagrama</td><td rowspan="2">678</td><td>617</td><td>155</td><td>1450</td></tr>
<tr><td>Comprimento Parte Inferior do Diagrama</td><td>Linha 11
349</td><td>/</td><td>1027</td></tr>
</table>

Conhecido o *range* da zona provável de defeito sabe-se também que o limite mínimo desse intervalo (1080 metros) ultrapassa o comprimento da linha 1 em 402 metros, o que significa que a zona provável do defeito começa a 402 metros após o início da linha 2 (65,15%). Por sua vez, os 1320 metros que definem o fim da zona considerada, ultrapassam a soma dos comprimentos da linha 1 e 2 em 25 metros (104,05%), significando que se inclui parte da linha 3 correspondente a essa distância "excedente". Sabendo que o comprimento da linha 3 é de 155 metros, 16,13% é a percentagem da linha incluída na zona provável de defeito. Estabelecendo a mesma lógica de pensamento para a linha 11, o intervalo percentual obtido está entre 115,19% e 183,95%, o que comprova que esta deixa de fazer parte da zona assinalada. A zona obtida foi a esperada e encontra-se representada na Figura 35, bem como os intervalos percentuais foram os calculados anteriormente e encontram-se assinalados na Figura 36.

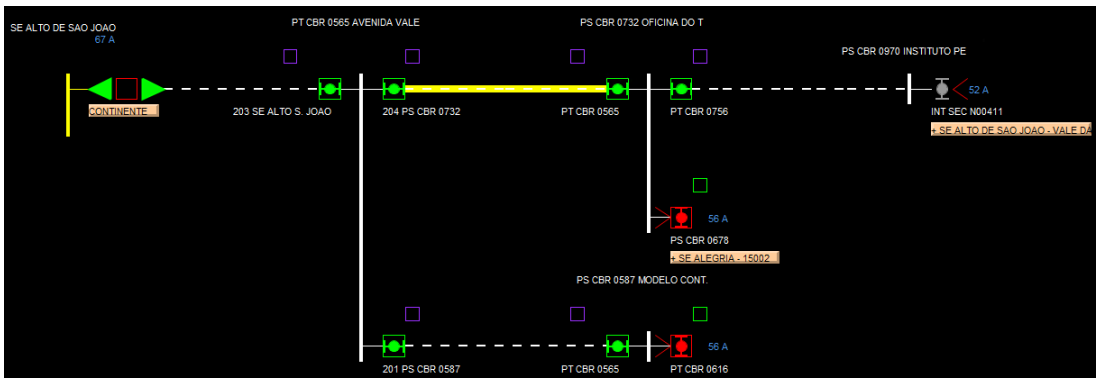


Figura 35 Zona provável de defeito para o cenário D1200.

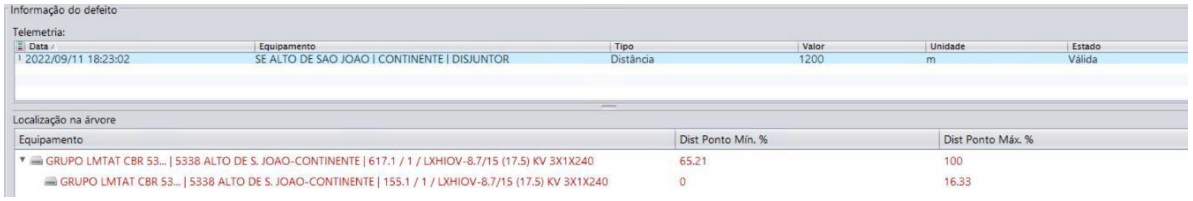


Figura 36 Resultados obtidos no gestor de defeitos para o cenário D1200.

3.5. INDICADORES DE DEFEITO E MEDIDAS DE DISTÂNCIA

Efetuados cenários de defeito com indicadores de defeito e, de seguida, com medidores de distância, foram também idealizados e desenvolvidos cenários de teste utilizando os dois elementos referidos anteriormente e, consequentemente, conciliando as funções de ambos.

3.5.1 Lógica de funcionamento

Quando usados em conjunto, estes dois elementos complementam-se obtendo, normalmente, soluções mais concretas e precisas do que quando usados em separado. No entanto, nesta situação, a ferramenta de localização de defeitos dá prioridade à medida de distância do defeito em função dos indicadores de passagem de defeito. Isto é, os IPDs, quando utilizados em conjunto com medidas de distância, tornam-se úteis a eliminar ramos "a mais" que seriam assinalados no caso de uma distância de defeito com solução ambígua – como é o caso dos cenários 3 (D670) e 4 (D800) em que a zona obtida salienta mais do que um ramo. No entanto, utilizando IPDs e medidores de distância, deixa de se aplicar a lógica de funcionamento dos IPDs em que o defeito se encontra entre o último IPD atuado e o primeiro IPD em estado normal. Aqui, caso a distância de defeito inclua uma linha que se encontre antes do último IPD atuado, essa linha será sinalizada como zona provável de defeito. Aquando da realização do presente projeto esta caraterística foi questionada porque, seguindo a lógica de funcionamento dos IPDs, se um IPD for sinalizado é porque factualmente foi atravessado pela corrente de defeito/detetou o defeito – presumindo o correto funcionamento do IPD –, independentemente da maior precisão dos medidores de distância. Após discussão relativamente ao assunto, chegou-se à conclusão que faria sentido alterar o modo de cálculo da FLF para passar a considerar a lógica de funcionamento dos IPDs, mesmo quando atuados juntamente com indicadores de distância. Essa alteração não foi feita durante o desenvolvimento do projeto, porém foi assumido o compromisso de implementar essa mudança após o desenvolvimento do projeto. Os cenários idealizados e enunciados de seguida têm o intuito de esclarecer o explicado.

3.5.2 Cenários de teste idealizados

Idealizaram-se quatro cenários de defeito utilizando IPDs e medidores de distância, com o intuito de verificar o funcionamento explicado anteriormente. Nestes testes voltaram a ser manipulados os estados dos disjuntores, as medidas de distância, e os bits de qualidade das medidas das devidas entidades. Uma vez mais, após execução dos cenários foram postos lado a lado os resultados obtidos com os resultados esperados.

3.5.2.1 Cenário 1 e 2 – D670 DF1+2 e D670 DF1+5

Nos primeiros dois cenários desenvolvidos utilizando indicadores de defeito e medidas de distância, recorreu-se a um dos cenários anteriormente enunciados desenvolvido apenas com medidas de distância cujo resultado obtido tenha sido ambíguo, e ativaram-se os IPDs e disjuntores que reduzissem a ambiguidade dessa solução. Aqui explicam-se em conjunto os cenários D670 DF1+2 e D670 DF1+5, por terem o mesmo cenário base de distância. Começou por se inserir a distância de defeito de 670 metros no disjuntor de cabeceira, onde na Figura 30 se verifica que o resultado obtido para a zona provável de defeito nesse cenário assinala dois ramos diferentes. Atuando os detetores de defeito e disjuntores 1 e 2 (coincidentemente também foram atuados num cenário explicado anteriormente), o resultado esperado será a exclusão da linha 11, mantendo apenas as linhas superiores no diagrama. Novamente, seguindo a lógica de funcionamento explicada anteriormente, neste caso não é esperado que a linha 1 seja excluída por ser conhecida a prioridade das medidas de distância em relação aos IPDs. A Figura 37 representa, a roxo a zona de defeito utilizando apenas os IPDs mencionados, e a cor de laranja a zona provável de defeito utilizando apenas medidas de distância. Como se verifica, a linha 2 é coincidente a ambos os cenários, pelo que o mais lógico seria mantê-la como zona provável de defeito, no entanto, tendo em conta que no momento de desenvolvimento do projeto é reconhecida maior precisão na localização de defeitos a partir de medidas de distância e, por isso, é desconsiderado o princípio de funcionamento dos IPDs, espera-se que também a linha 1 seja assinalada. A Figura 38 representa o resultado obtido após a execução do cenário, que confirma o esperado.

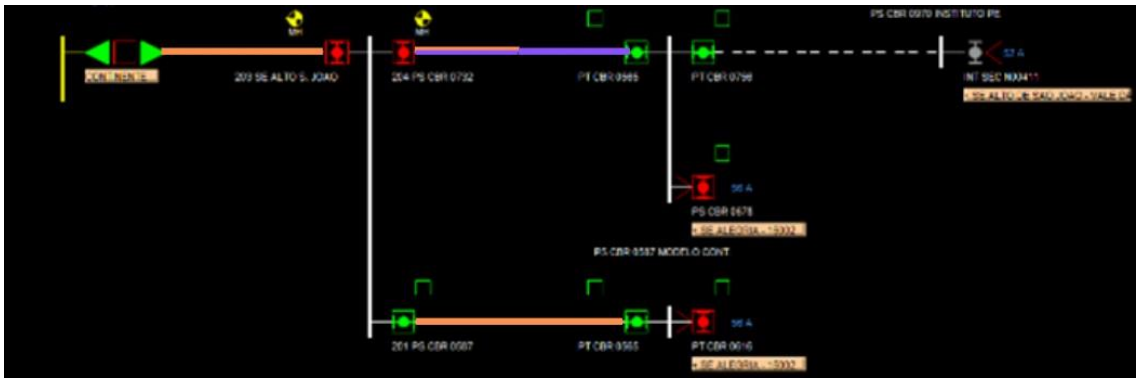


Figura 37 Representação individual das zonas prováveis de defeito para os IPDs 1 e 2 e para a distância de 670 metros, sobrepostas.

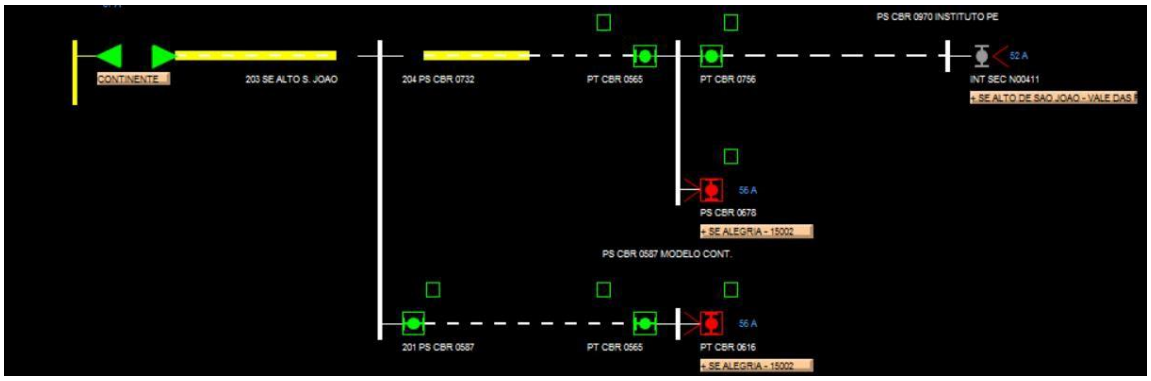


Figura 38 Zona provável de defeito para o cenário D670 DF1+2.

O *script* dbgm desenvolvido para o cenário D670 DF1+2 encontra-se, em parte, representado em baixo, de modo a salientar a manipulação das entidades analógicas e digitais referentes ao estado de disjuntores e IPDs, distância entre o disjuntor de cabeceira e o defeito, e *bits* de qualidade. Estão representadas as linhas de código que manipulam essas entidades.

```
wtag dig FSAJO-2212-DJEST 1
wtag dig FSAJO-2212-DJEST 0 96

wtag anl FSAJO-2212-DJDST 670
wtag anl FSAJO-2212-DJZRA 96 96
wtag anl FSAJO-2212-DJZRS 96 96

wtag dig FD328A2203-ITEST 1
wtag dig FD328A2203-ITEST 0 96
wtag dig FD328A2204-ITEST 1
wtag dig FD328A2204-ITEST 0 96

wtag dig FD328A2203-IT-DF 0 96
wtag dig FD328A2203-IT-DF 1
wtag dig FD328A2204-IT-DF 0 96
wtag dig FD328A2204-IT-DF 1
wtag dig FD328A2203-DDHIF 0 96
wtag dig FD328A2203-DDHIF 1
wtag dig FD328A2204-DDHIF 0 96
wtag dig FD328A2204-DDHIF 1
```

Para o cenário D670 DF1+5, atuando os indicadores de defeito e disjuntores 1 e 5, o resultado esperado é o contrário ao cenário D670 DF1+2, no sentido em que desta feita, a linha excluída da zona de defeito espera-se que seja a linha 2, incluindo novamente a linha 11. À semelhança da Figura 37, foi composta uma imagem com a representação dos resultados da zona provável de defeito utilizando apenas os IPDs mencionados e apenas medidas de distância, representada na Figura 39. Aqui a única linha comum é a linha 11 pelo que esta

seria o resultado lógico, no entanto, o resultado esperado inclui também a linha 1, à semelhança do caso anterior. O resultado obtido está representado na Figura 40 e verifica, uma vez mais, o resultado esperado.

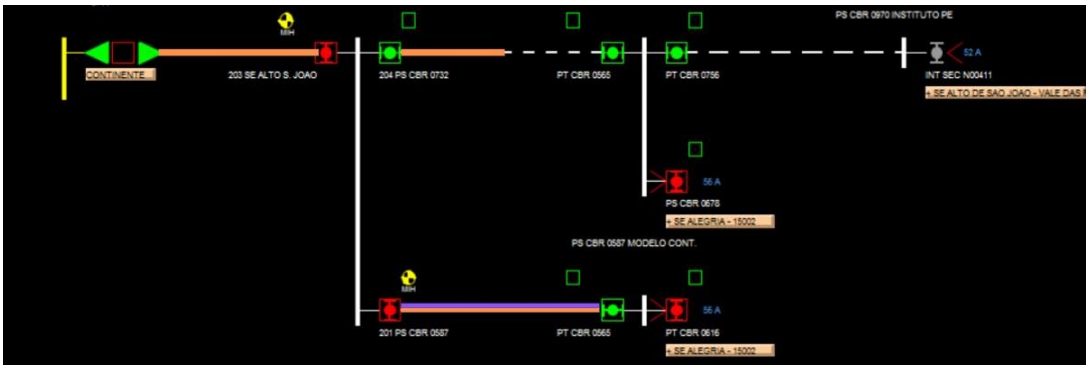


Figura 39 Representação individual das zonas prováveis de defeito para os IPDs 1 e 5 e para a distância de 670 metros, sobrepostas.

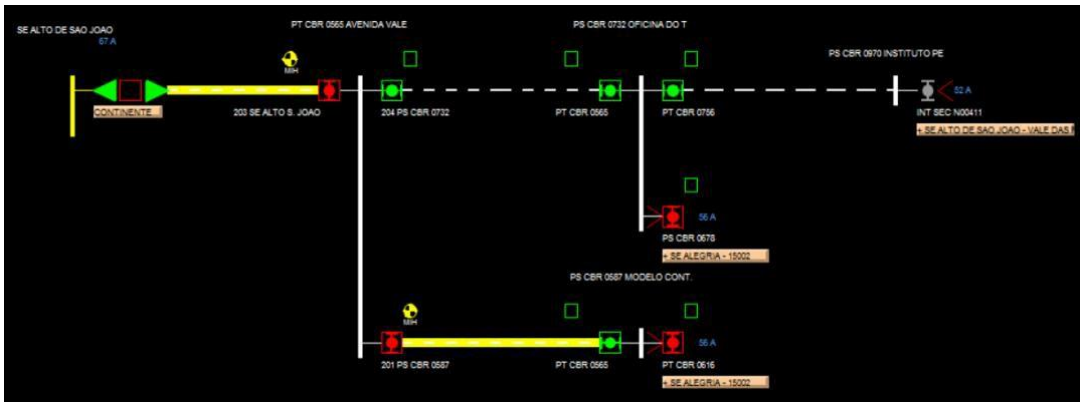


Figura 40 Zona provável de defeito para o cenário D670 DF1+5.

3.5.2.2 Cenário 3 e 4 – D800 DF1+2 e D800 DF1+5

Passando para os dois últimos cenários desenvolvidos utilizando IPDs e medidas de distância, recorreu-se ao cenário D800 onde o resultado obtido foi ambíguo, e ativaram-se os IPDs e disjuntores que reduzissem a ambiguidade dessa solução. Novamente, explicam-se em conjunto os cenários D800 DF1+2 e D800 DF1+5, por terem o mesmo cenário base de distância. Colocando a medida de distância a 800 metros e atuando os IPDs e disjuntores 1 e 2, espera-se que a linha 11 deixe de fazer parte do resultado obtido pela FLF. O resultado esperado é a linha 2, uma vez que agora não há qualquer linha assinalada antes do disjuntor 1. Na Figura 41 temos novamente representada a rede assinalada com os resultados obtidos

para os detetores de defeito e para as medidas de distância, em separado e com cores diferentes, seguida do resultado obtido, na Figura 42.

Figura 41 Representação individual das zonas prováveis de defeito para os IPDs 1 e 5 e para a distância de 670 metros, sobrepostas.

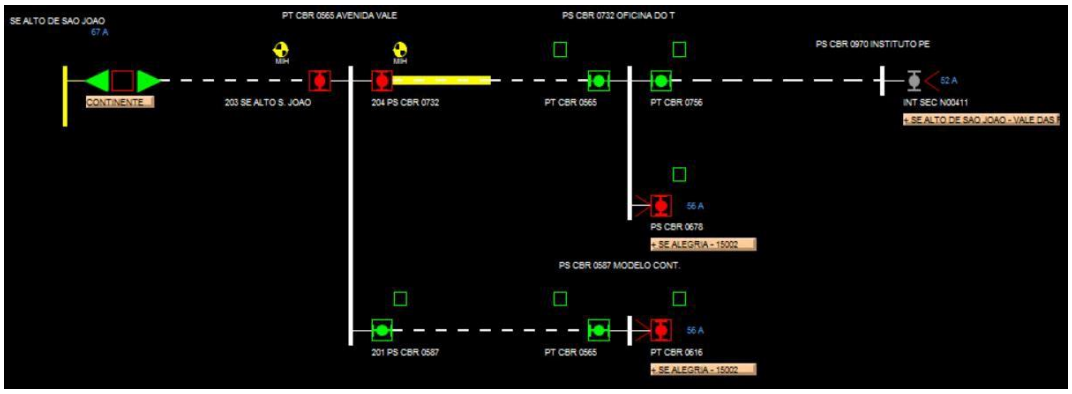


Figura 42 Zona provável de defeito para o cenário D800 DF1+2.

Como se vê a partir da Figura 42, a linha assinalada como linha com defeito foi a única linha que havia sido comum anteriormente, quando os cenários foram executados em separado (o que se verifica na Figura 41), o que confirma o resultado esperado.

No caso do cenário 4, foram atuados os disjuntores e IPDs 1 e 5 em conjunto com a distância de defeito de 800 metros e, desta vez, o resultado esperado seria a linha 11 na sua totalidade. Isto porque a zona comum de defeito entre o cenário D800 e DF1+5 é a linha 11, ou seja, numa representação como a da Figura 41 para este cenário, a linha roxa assinalaria a linha 11. O resultado obtido após execução do cenário de defeito confirma o esperado e está presente na Figura 43.

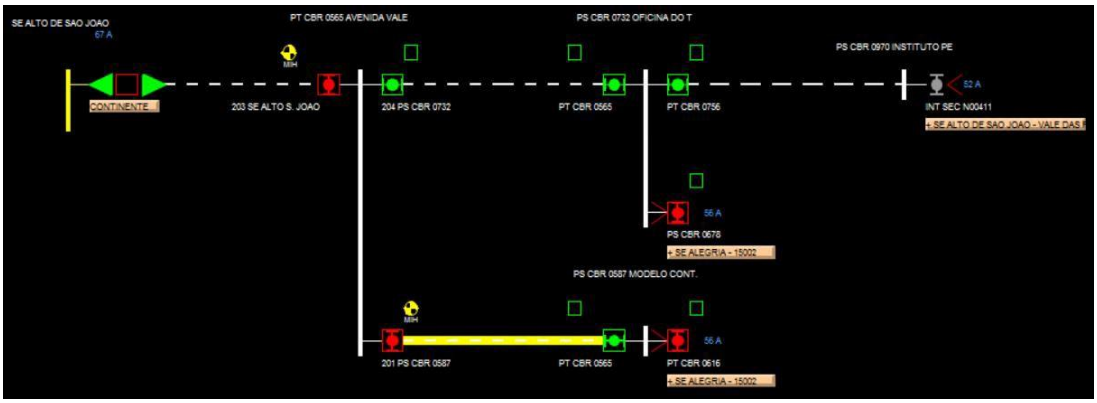


Figura 43 Zona provável de defeito para o cenário D800 DF1+5.

Para estes dois cenários, são também apresentados os resultados obtidos no gestor de defeitos para os intervalos de distância, em percentagens. Visto que apesar de serem utilizados detetores de defeito também são usadas medidas de distância, a margem de erro da distância continua a ser considerada. Nestes dois cenários, dado que a distância de defeito utilizada foi de 800 metros, a margem de erro foi já calculada em 3.4.2.3 aquando da explicação do cenário D800. Assim, a diferença do resultado apresentado pelo gestor de defeitos rege-se pela exclusão da linha 11 e da linha 2 nos cenários D800 DF1+2 e D800 DF1+5, respetivamente. Esta afirmação verifica-se a partir das linhas e percentagens obtidas pelo gestor de defeitos representadas na Figura 44.

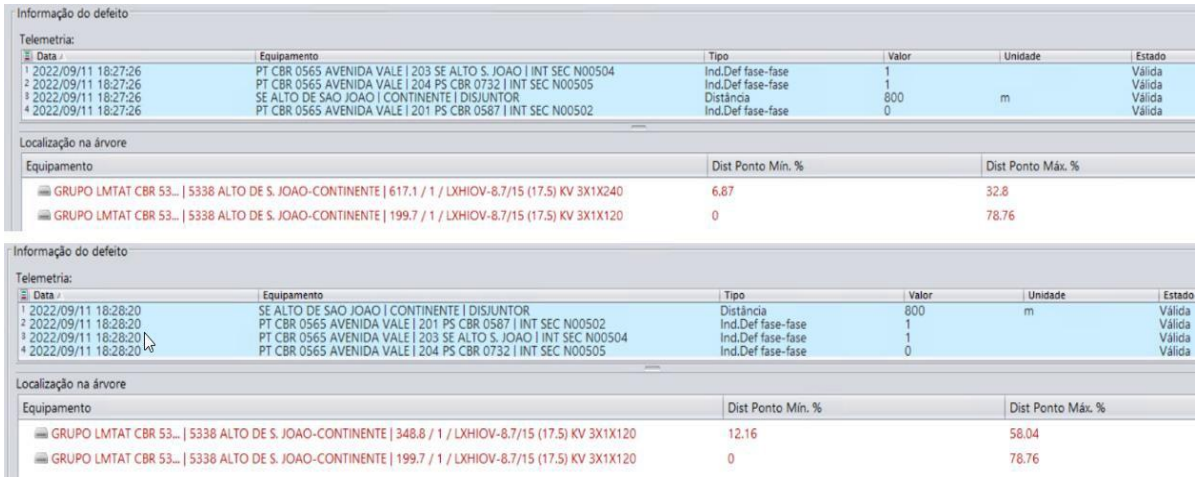


Figura 44 Resultados obtidos no gestor de defeitos para os cenários D800 DF1+2 e D800 DF1+5.

À semelhança do que acontece em cenários anteriores (por exemplo no cenário D800), uma vez mais a linha que não está representada no diagrama volta a surgir nos resultados obtidos, pelo que deve ser considerada como solução.

3.6. CENÁRIOS INCONGRUENTES

Após verificado o correto funcionamento da FLF em cenários com IPDs, cenários apenas de medidores de distância, e cenários que conciliam ambos os equipamentos em conjunto, achou-se por bem submeter a ferramenta de localização de defeitos a cenários cujo resultado não fosse possível (incongruentes).

3.6.1 Lógica de pensamento

O intuito na criação destes cenários incongruentes foi, como dito anteriormente, submeter a FLF a defeitos cujo cálculo da zona provável de defeito não fizesse sentido. Isto é, foram idealizados cenários utilizando, à semelhança do capítulo anterior, IPDs e medidas de distância, desta feita, estes elementos foram utilizados para se contrariarem e não para, em conjunto, tornar os resultados obtidos pela FLF mais concretos. A título de exemplo, suponha-se uma distância de defeito de 2000 metros (ou superior) mas apenas o indicador de defeito 1 atuado e o disjuntor 1 aberto. Neste exemplo sabe-se que a distância é elevada o suficiente para ultrapassar o comprimento da linha 1, no entanto tendo os IPDs e disjuntores 2 e 5 fechados (estado normal) significa que o defeito não passa nem pela linha 2 e seguintes, nem pela linha 11. O objetivo é submeter a FLF a este tipo de cenários de teste, verificando depois os resultados obtidos.

3.6.2 Cenários de teste idealizados

Foram idealizados 4 cenários de defeito incongruentes recorrendo novamente às entidades mencionadas nos outros cenários – IPDs, disjuntores, medidas de distância e *bits* de qualidade.

3.6.2.1 Cenário 1 – INC D500 DF1+2+3+4

No primeiro cenário incongruente, o objetivo foi manipular a medida de distância de modo que esta fosse curta o suficiente para não ultrapassar o comprimento da primeira linha. Conhecendo já duas possíveis distâncias que cumprem com esse requisito, utilizadas nos

cenários de medidores de distância, decidiu-se atribuir 500 metros à distância de defeito. Com o intuito de trazer incongruência ao cenário, os detetores de defeito atuados foram todos os da linha superior, ou seja, os IPDs e disjuntores 1, 2, 3 e 4 que, segundo o seu princípio de funcionamento, indicariam as últimas linhas como zonas prováveis de defeito (desde a linha 4 à linha 10). Assim, a solução indicada pela distância salienta a primeira linha, enquanto a solução indicada pelos IPDs indica as últimas linhas.

Agora sim, faz sentido atribuir mais importância e prioridade às medidas de distância, uma vez que, tendo obtido resultados totalmente opostos significa que ocorreu algum tipo de erro. Partindo do princípio de que os medidores de distância são mais precisos e concretos – que são – espera-se que a FLF, neste caso, realize o cálculo da zona provável de defeito tendo em conta apenas a distância do defeito. No entanto, apenas se espera o dito anteriormente, uma vez que o resultado esperado a partir da medida de distância é uma zona na qual os disjuntores e IPDs presentes também se encontram acionados. Se os disjuntores e IPDs se encontram acionados, é porque um defeito ocorreu na rede, se se encontram acionados também nessa zona e a medida de distância refere essa zona como solução então o mais certo é que o defeito tenha passado nessa zona. O mesmo está representado na Figura 45.

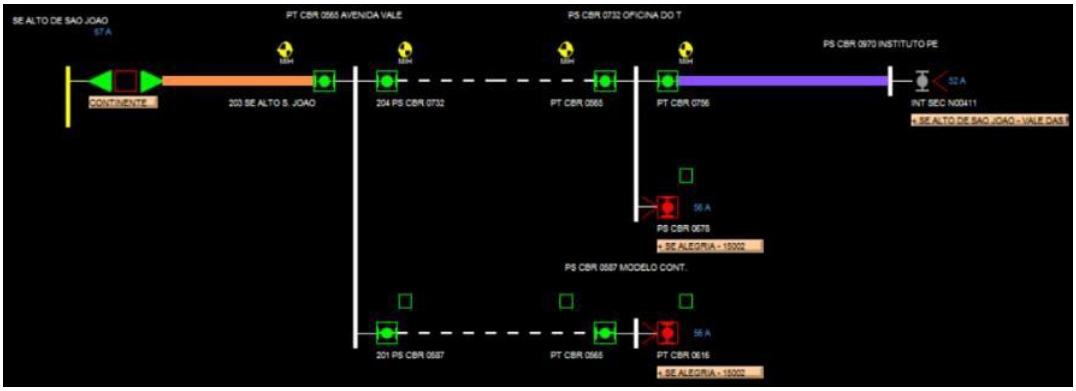


Figura 45 Representação individual das zonas prováveis de defeito para os IPDs 1, 2, 3 e 5 (a roxo) e para a distância de 500 metros (a laranja), sobrepostas.

Assim, o resultado esperado será a sinalização da primeira linha como zona provável de defeito. O resultado obtido encontra-se na Figura 46 onde, para além da zona do defeito, se verificam também os IPDs e disjuntores atuados.

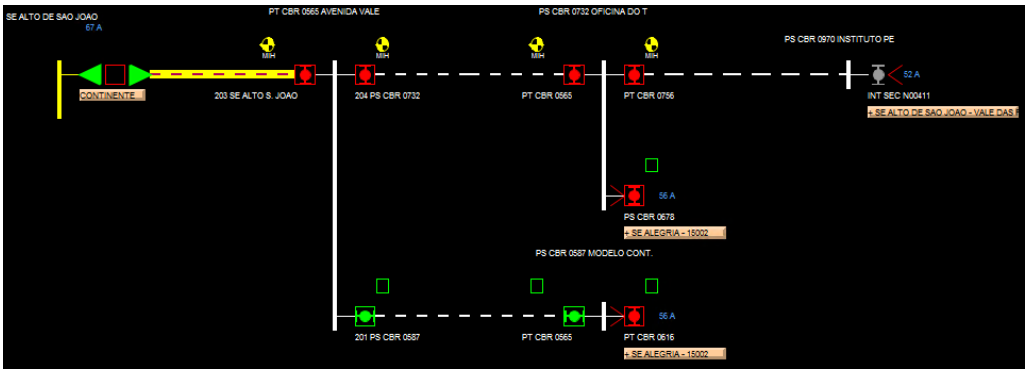


Figura 46 Zona provável de defeito obtida para o cenário INC D500 DF1+2+3+4.

O resultado esperado e obtido para o intervalo de distâncias referentes à zona provável de defeito é o mesmo que o obtido no cenário 2 (D500) e encontra-se representado na Figura 47 que é o resultado obtido a partir do gestor de defeitos.

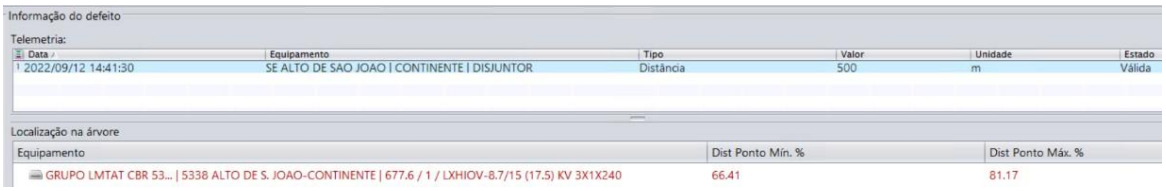
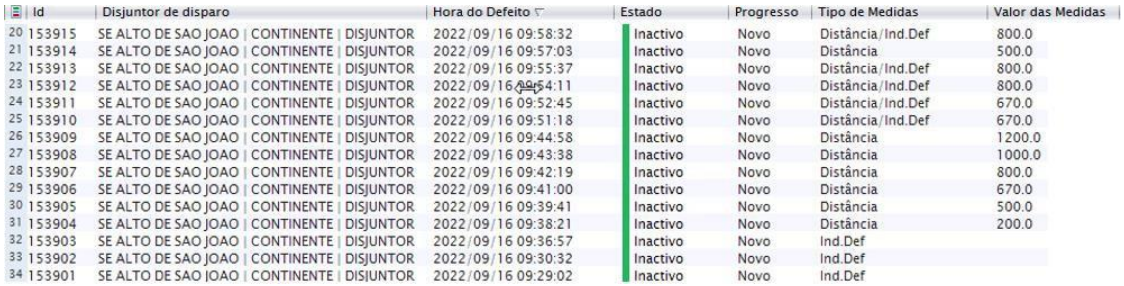


Figura 47 Resultado obtido no gestor de defeitos para o cenário INC D500 DF1+2+3+4.

Algo curioso de salientar é o facto de o próprio gestor de defeitos apresentar uma indicação de que os IPDs não foram tidos em conta neste cenário (verifique-se o tipo de defeito considerado). Prova disso é a Figura 48 que tem apresentado o cenário em questão, todos os cenários anteriores, e o cenário seguinte (INC D800) em forma de lista. Nessa lista, a partir do cabeçalho vê-se que na coluna "Tipo de Medidas", todos os defeitos têm atribuído "Ind.Def" no caso dos primeiros cenários explicados, "Distância" nos segundos, e "Distância/Ind.Def" nos restantes porque tanto nestes últimos como nos cenários incongruentes, recorreu-se aos dois tipos de medidas. No entanto, verifica-se que o cenário que se está a explicar agora – com o Id 153914 – apenas está definido como um cenário de distância. Daqui conclui-se que o sistema não considerou os IPDs acionados.



	Id	Disjuntor de disparo	Hora do Defeito	Estado	Progresso	Tipo de Medidas	Valor das Medidas
20	153915	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:58:32	Inactivo	Novo	Distância/Ind.Def	800.0
21	153914	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:57:03	Inactivo	Novo	Distância	500.0
22	153913	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:55:37	Inactivo	Novo	Distância/Ind.Def	800.0
23	153912	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:54:11	Inactivo	Novo	Distância/Ind.Def	800.0
24	153911	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:52:45	Inactivo	Novo	Distância/Ind.Def	670.0
25	153910	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:51:18	Inactivo	Novo	Distância/Ind.Def	670.0
26	153909	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:44:58	Inactivo	Novo	Distância	1200.0
27	153908	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:43:38	Inactivo	Novo	Distância	1000.0
28	153907	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:42:19	Inactivo	Novo	Distância	800.0
29	153906	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:41:00	Inactivo	Novo	Distância	670.0
30	153905	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:39:41	Inactivo	Novo	Distância	500.0
31	153904	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:38:21	Inactivo	Novo	Distância	200.0
32	153903	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:36:57	Inactivo	Novo	Ind.Def	
33	153902	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:30:32	Inactivo	Novo	Ind.Def	
34	153901	SE ALTO DE SAO JOAO | CONTINENTE | DISJUNTOR	2022/09/16 09:29:02	Inactivo	Novo	Ind.Def	

Figura 48 Registo de defeitos no gestor de defeitos.

3.6.2.2 Cenário 2 – INC D800

No cenário 2 dos cenários incongruentes, o objetivo foi colocar um qualquer valor de distância, mas sem colocar qualquer indicador de defeito ativo. Isto é, criar a existência de um defeito com um determinado valor de distância até ao disjuntor de cabeceira, no entanto colocando todos os IPDs e disjuntores a indicar um estado de rede saudável.

O código desse mesmo cenário está representado de seguida. Escolheu-se uma distância de 800 metros (sem justificação uma vez que poderia ser qualquer distância dentro da rede) e, como dito anteriormente, forçou-se a manter os disjuntores e IPDs fechados/não atuados. Mantendo isto em mente, no excerto do código representado de seguida estão presentes o estado aberto do disjuntor de cabeceira, os seus *bits* de qualidade, as entidades analógicas DJDST, DJZRA e DJZRS, e apenas estão representadas linhas de código referentes a um dos disjuntores e seu IPD associado (últimas seis linhas de código), uma vez que para os outros disjuntores as linhas são idênticas com a exceção da *tag*.

```
wtag dig FSAJO-2212-DJEST 1
wtag dig FSAJO-2212-DJEST 0 96

wtag anl FSAJO-2212-DJDST 800
wtag anl FSAJO-2212-DJZRA 96 96
wtag anl FSAJO-2212-DJZRS 96 96

wtag dig FD328A2203-ITEST 2
wtag dig FD328A2203-ITEST 0 96

wtag dig FD328A2203-IT-DF 0 96
wtag dig FD328A2203-IT-DF 0

wtag dig FD328A2203-DDHIF 0 96
wtag dig FD328A2203-DDHIF 0
```

Neste cenário, a FLF não indica qualquer linha como zona provável de defeito. Isto é, uma vez que, apesar de haver uma distância de erro associada ao disjuntor, visto que os IPDs não detetam qualquer defeito, é porque não houve nenhuma corrente de falha a atravessar os mesmos. Daí ser impossível indicar uma zona provável de defeito. Aqui, não é atribuída qualquer prioridade às medidas de distância.

3.6.2.3 Cenário 3 – INC D1200 DF1+5

No terceiro cenário incongruente, quis-se colocar uma distância cujo valor fosse elevado o suficiente para ultrapassar a primeira linha e, simultaneamente, ultrapassar a linha 11. Assim, o valor escolhido foi 1200 metros, por ter sido executado um cenário com essa distância anteriormente. Sabendo qual a área provável de defeito com essa medida de distância, decidiu-se atuar os IPDs e disjuntores 1 e 5, isto é, os aparelhos cujo resultado para zona provável de defeito seria a linha 11. Por se saber que por o IPD 2 não ter detetado corrente de falha, não será indicada nenhuma das linhas que se seguem ao mesmo como zona provável de defeito, não se espera a indicação de qualquer linha como zona de defeito. Na Figura 49 estão representados os resultados para a medida de distância e para os indicadores de defeito.

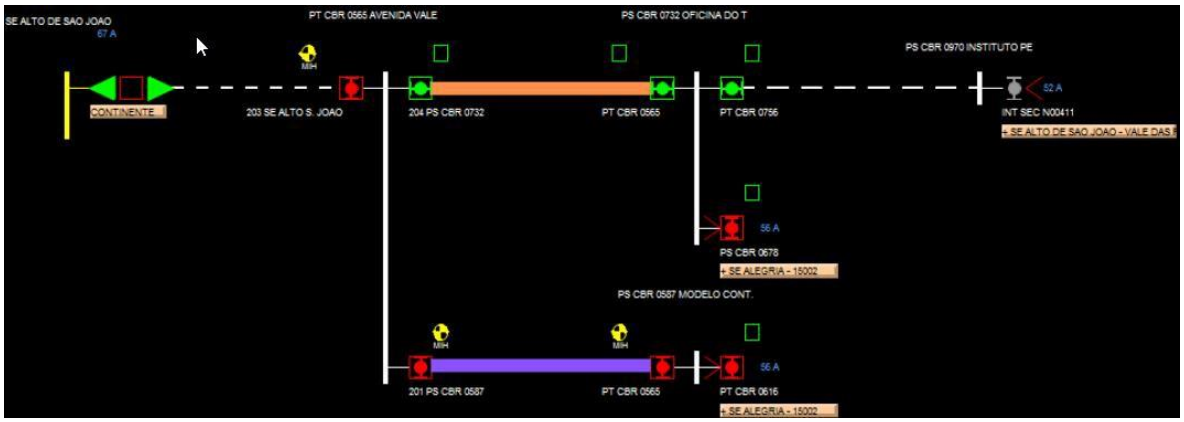


Figura 49 Representação individual das zonas prováveis de defeito para os IPDs 1 e 5 e para a distância de 1200 metros, sobrepostas.

Como se confirma a partir da Figura 49, a zona indicada pelo medidor de distância são as linhas 2 e 3, zona essa que é indicada como estando em estado normal pelos IPDs. Assim, confirma-se que, novamente, não haverá qualquer indicação de zona provável de defeito.

3.6.2.4 Cenário 4 – INC D1780 DF1+4

No quarto e último cenário, colocou-se uma distância elevada o suficiente para que a zona provável de defeito resultante dos cálculos da FLF fosse a última zona da rede (da linha 4 à linha 10). Com essa zona provável de falha, sabe-se que existem quatro IPDs e disjuntores entre o disjuntor de cabeceira e essa zona assinalada. O objetivo passou por acionar apenas dois, de modo a compreender se, havendo IPDs que não detetaram defeito ao longo de um caminho teoricamente defeituoso, esse defeito é calculado seguindo as medidas de distância, ou se a FLF considera o facto desses IPDs "afirmarem" a não passagem de defeito naquelas linhas. Posto isto, atribuiu-se ao defeito uma distância de 1780 metros, e acionaram-se os IPDs e disjuntores 1 e 4. A essa distância de defeito, corresponde o intervalo de distâncias apresentado na Tabela 5, que confirma a localização do defeito a partir da linha 4 até à linha 8.

Tabela 5 Intervalo de distâncias prováveis de defeito para o cenário INC D1780 DF1+4.

Range do Defeito		
-10%	0	10%
1602	1780	1958

Na Figura 50 está representada a zona provável de defeito caso o cenário apenas recorresse à medida de distância de 1780 metros. Calculando as devidas percentagens do intervalo de distâncias da zona de falha utilizando as fórmulas apresentadas anteriormente, obteve-se a Tabela 6 com todas as linhas da provável área de falha representadas.

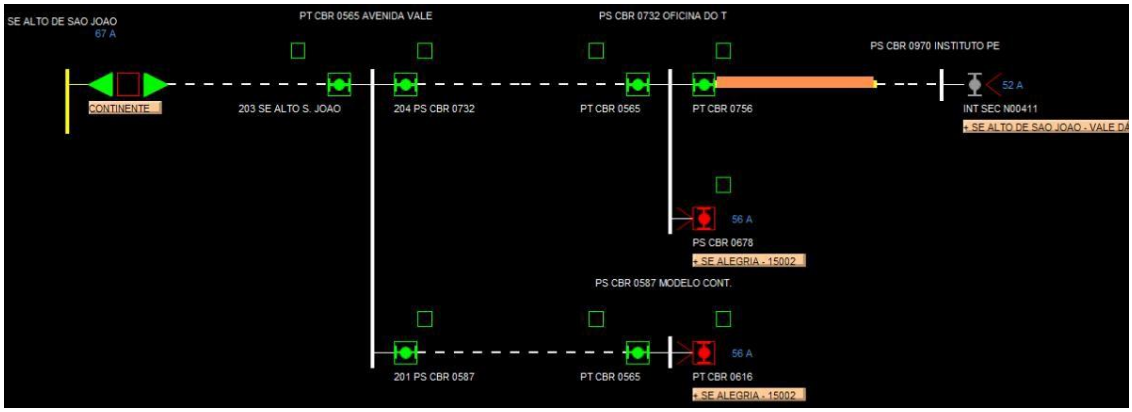


Figura 50 Zona provável de defeito para o cenário INC D1780 DF1+4.

Tabela 6 Percentagem provável de defeito para cada linha assinalada como linha defeituosa no cenário INC D1780 DF1+4.

	%	
Linha 4	62,26	100
Linha 5	0	100
Linha 6	0	100
Linha 7	0	100
Linha 8	0	32,61

Uma vez que os IPDs 2 e 3 se mantêm não atuados não é esperado que seja assinalada qualquer zona provável de falha. Aqui seguiu-se a mesma linha de pensamento que no cenário anterior. Apesar de, neste caso, os IPDs 1 e 4 terem assinalado passagem de defeito, os IPDs intermediários 2 e 3 não foram acionados, pelo que isso indica que a partir do disjuntor 2 não haverá qualquer defeito. Esta foi a lógica de pensamento aplicada e confirmada a partir do resultado obtido que não retornou qualquer zona provável de defeito.

No entanto, no caso deste cenário poderia ter sido diferente. Isto acontece uma vez que para estes casos existe uma "margem de erro" que indica a possibilidade de haver erro nos IPDs em casos como este. Explicando melhor, se essa margem de erro fosse 2, este cenário teria uma zona provável de defeito igual à da Figura 50, uma vez que os dois IPDs não acionados seriam desconsiderados e, como os outros IPDs presentes entre o disjuntor de cabeceira e a zona do defeito estão ativados, a zona de defeito seria assinalada. Se a margem de erro fosse 3, poderia existir ainda mais um IPD presente entre o disjuntor de cabeceira e a zona de defeito que não tivesse sido acionado, além dos dois já considerados. Caso essa margem de erro fosse 1, um dos IPDs não acionados teria de ter sinalizado passagem de defeito para que as linhas 4-8 fossem assinaladas. Neste cenário e nesta rede, a margem de erro considerada foi de 0, pelo que todos os IPDs teriam de ter sinalizado a passagem de erro para que a zona de defeito fosse retornada.

3.7. *SCRIPT* DE *RESET*

Enunciados e explicados os cenários de defeitos para testagem da ferramenta de localização de defeitos, tem também de ser referida a criação de um *script* de *reset* que coloca a rede no estado ideal inicial. O objetivo deste *script* é, como foi referido, colocar a rede num estado ideal para correr os cenários de teste. Assim, o *reset* da zona da rede em questão tem de ser

feito sempre antes de ser corrido um novo cenário de teste e/ou no fim do último cenário de teste para que a rede fique no seu estado normal (energizada). Após ser executado o *script* de *reset*, a rede fica no estado representado anteriormente na Figura 14.

Este *script* coloca as entidades no seu estado inicial que são:

- Disjuntor de cabeceira – Fechado e *bits* de qualidade desativados
- Disjuntores de rede – Fechados e *bits* de qualidade desativados
- Indicadores de passagem de defeito e respetiva sinalização – Não atuados e *bits* de qualidade ativos
- Entidades analógicas (DJDST, DJZRA e DJZRS) – Valor a 0 e *bits* de qualidade desativados

Neste *script* existem ainda dois elementos que nunca são utilizados nem alteram o seu valor, no entanto é feito o seu *reset* apenas por segurança. Estes dois elementos são os disjuntores de fim de linha, que se encontram no fim da linha 11 e no barramento entre as linhas 3 e 4 – são os disjuntores nos barramentos PT CBR 0616 e PS CBR 0676.

Deste modo, executando este *script*, todas as entidades voltam ao estado inicial pré-defeito e pode ser executado qualquer cenário de teste, independentemente dos elementos da rede que vá manipular ou dos estados que irá colocar nos mesmos.

3.8. CENÁRIOS COM RESISTÊNCIA E REATÂNCIA

A FLF tem também capacidade para calcular zonas prováveis de defeito com base em resistências e reatâncias das linhas. No presente projeto não foram desenvolvidos cenários de teste tendo em consideração estes dois elementos elétricos uma vez que não houve tempo para tal, atribuindo prioridade aos tipos de defeito mencionados anteriormente. Contudo, no fim do desenvolvimento e automatização dos cenários de defeito explicados foi abordada esta possibilidade e feito um estudo da rede realizando os cálculos necessários de forma a obter os valores de resistência e reatância de cada linha, que acabaram por não ser utilizados, mas que se encontram representados na Figura 51 e na Tabela 7.

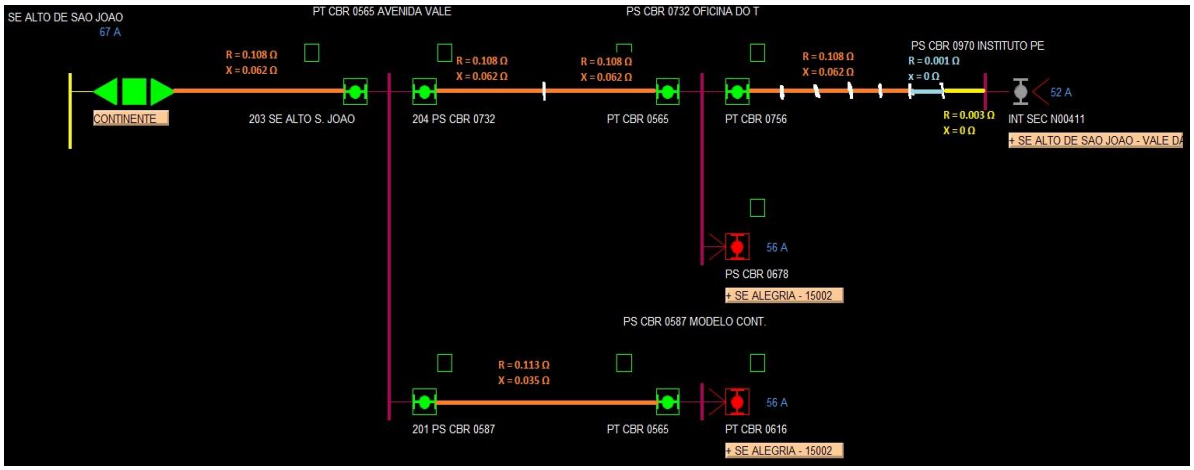


Figura 51 Resistência e reatância das linhas constituintes da zona da rede estudada.

Tabela 7 Parametrização da zona da rede estudada, ao nível da resistência e reatância das linhas.

LINHA	COMPRIMENTO DA LINHA (km)	RESISTÊNCIA (ohm/km)	RESISTÊNCIA TOTAL (ohm)	REATÂNCIA (ohm/km)	REATÂNCIA TOTAL (ohm)
1	0.678	0.160	0.108	0.092	0.062
2	0.617	0.160	0.099	0.092	0.057
3	0.155	0.160	0.025	0.092	0.014
4	0.226	0.160	0.036	0.092	0.021
5	0.011	0.160	0.002	0.092	0.001
6	0.093	0.160	0.015	0.092	0.009
7	0.082	0.160	0.013	0.092	0.008
8	0.295	0.160	0.047	0.092	0.027
9	0.11	0.010	0.001	0.001	0.000
10	0.287	0.010	0.003	0.001	0.000
11	0.349	0.324	0.113	0.101	0.035

4. AUTOMATIZAÇÃO DOS TESTES

Após apresentação dos cenários de teste idealizados para testar o correto funcionamento da ferramenta de localização de defeitos e serem verificados os resultados obtidos em cada um dos defeitos, o passo seguinte foi a automatização dos mesmos. Este capítulo descreve a parte do projeto onde foram dados os passos mais consideráveis em direção ao objetivo final, bem como onde foram encontrados os obstáculos mais significativos do mesmo. É descrita a automatização dos testes e apresentadas ferramentas, programas e linguagens de programação utilizadas para tal. São enunciadas as validações efetuadas em cada cenário e a sua automatização, é inserido um novo *script* para verificação do estado da rede antes da execução dos testes, e é explicada a integração dos *scripts* na máquina para criação do bloco de testes de localização de defeitos.

4.1. CENÁRIOS DE DEFEITO

A automatização dos cenários de teste abordados no capítulo anterior baseou-se em ficheiros de código *javascript*. Cada cenário foi agora implementado em JS, atuando para cada cenário

as entidades mencionadas anteriormente, no entanto, agora desenvolvidos numa aplicação do ScateX denominada “Editor de Mundo Real”. O Editor de Mundo Real é uma aplicação que está disponível na WKS e que permite aos utilizadores fazer alterações na base de dados das máquinas DMS e SCADA, nomeadamente a partir do desenvolvimento de ficheiros de código *javascript* denominados “sequências avançadas”.

4.1.1 Desenvolvimento sequências avançadas

As sequências avançadas são uma ferramenta que permitem a criação de ficheiros em *javascript*, utilizados para configuração e execução de uma série, ou sequência, de operações no sistema. Após a injeção destas sequências avançadas no sistema (este tópico será explicado mais à frente), estas passam a estar presentes numa outra aplicação/ferramenta denominada “Gestor de Sequências de Comando”, onde se efetua a execução das mesmas. Estas duas ferramentas mencionadas correm num componente do sistema ScateX denominado *derivedthread*, que é uma aplicação do sistema da Efacec que tem *interface* direta com o ScateX e, entre outras funcionalidades, tem a capacidade de executar *scripts* desenvolvidos em *javascript*.

4.1.1.1 *Javascript*

A linguagem de programação *javascript* (também conhecida como JS), é uma linguagem interpretada e baseada em objetos com funções de primeira classe. O facto de ser uma linguagem de programação interpretada significa que o código fonte é executado por um programa (interpretador) que, de seguida é interpretado pelo sistema operacional ou processador. Isto significa que o código fonte não é diretamente traduzido pela máquina, mas sim pelo interpretador que lê e executa o código [44]. Uma linguagem de programação baseada em objetos, como é o caso do *javascript*, é uma linguagem que se baseia no conceito de classes – que são pedaços de código simples, reutilizáveis e usados para criar objetos com determinadas caraterísticas – e objetos [45].

4.1.1.2 Editor de Mundo Real

Como referido anteriormente, o editor de mundo real é uma aplicação presente no sistema ScateX e foi nesta aplicação que se criaram os ficheiros de código *javascript* referentes a cada cenário. Na criação de cada um dos *scripts* é necessário indicar o nome do mesmo, o

modo de execução (“Periodicamente” – corre de x em x tempo; “A pedido do utilizador” – corre quando o utilizador executar), e escrever o código do *script*. Estes três parâmetros estão representados na Figura 52. O modo de execução “A pedido do utilizador” foi o escolhido, uma vez que, quando a sequência dos testes estiver estabelecida, vai ser necessário que cada cenário seja capaz de executar o seguinte. Para isso, é necessário que a sequência aguarde por um pedido para ser executada.

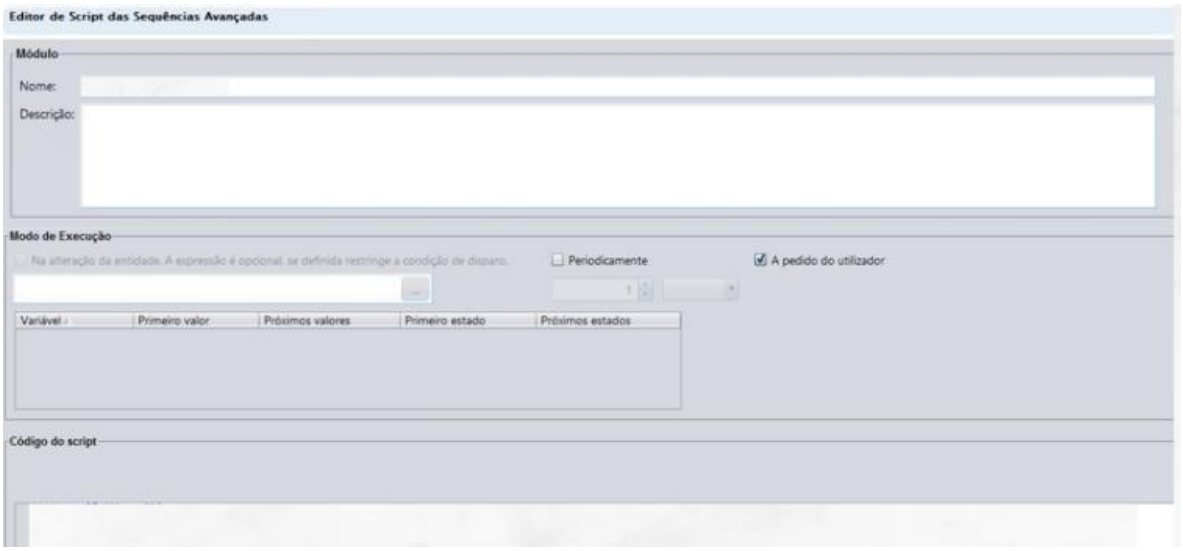


Figura 52 Criação de uma sequência avançada.

Após escrever o código pretendido ou fazer alterações, é necessário gravar o ficheiro e, além disso, realizar uma injeção do novo ficheiro/novas alterações no sistema. Uma injeção de um ficheiro consiste numa alteração imposta nas bases de dados das máquinas DMS e SCADA, e pode ser relativa a diagramas, construção efetuada na rede, dados de equipamentos (alterações, remoções ou criações), ou mesmo para alterar configurações do sistema. Caso esta injeção não seja feita, quando se tentar executar o código podem acontecer duas situações: 1- se o ficheiro for novo e nunca tiver sido injetado, o sistema não o vai reconhecer, ou seja, é como se o ficheiro não existisse; 2- se apenas tiverem sido feitas algumas alterações, quando o ficheiro for executado, este vai correr sem essas alterações. Outra situação que pode ocorrer caso não seja feita a injeção no sistema é o desaparecimento do ficheiro (caso tenha sido criado pela primeira vez) ou das suas alterações (caso sejam apenas alterações a um código já existente) quando as máquinas forem atualizadas.

4.1.1.3 Cenários de teste

Uma vez que os cenários de teste já foram idealizados e implementados em *scripts* dbgm no capítulo anterior, o que está descrito neste subcapítulo é, fundamentalmente, a transformação desses comandos dbgm (ferramenta desenvolvida pela Efacec para simular telemetria do terreno) que constituem os *scripts*, em *javascript*. Isto é, todos os ficheiros de código que constituem os cenários de defeito idealizados, foram agora desenvolvidos em *javascript*, de modo a ser possível, mais à frente, automatizá-los numa sequência de testes. Para explicar a adaptação dessas alterações e manipulações de entidades e estados, são utilizados excertos de código para alguns dos cenários com a devida explicação.

Antes disso, algo comum a todos os cenários é a necessidade de haver essas manipulações de valores e/ou estados, pelo que de seguida segue um excerto de código que serve de código base para o processo de manipulação das entidades. Assuma-se que:

- *tag* – identificação do equipamento (ex: FD328A2203-ITEST);
- *value* – valor que indica o estado a colocar a entidade;
- *load* – variável criada para carregar a entidade passada pela *tag*.

```
var load = JSObjectLoader.loadDigital(tag);
//se "tag" for uma entidade de medida então fica:
//var load = JSObjectLoader.loadAnalog(tag);
if(load.waitForReply(10000) == JSReplyState.TIMEOUT){
   JSLogger.filelog('Equipment not loaded');
}
if(!load.isOk()){
   JSLogger.filelog('Equipment not Ok);

var loadTarget = load.getTarget();
loadTarget.setStatus(JSEntityStatus.ST_M_OFFSCAN);
loadTarget.setStatus(JSEntityStatus.ST_A_OFFSCAN);
//se for para desativar os bits de offscan:
//loadTarget.setStatus(JSEntityStatus.ST_M_OFFSCAN);
//loadTarget.setStatus(JSEntityStatus.ST_A_OFFSCAN);

loadTarget.setValue(value);
loadTarget.write();
```

Neste código são apresentadas algumas palavras reservadas iniciadas por "JS", que representam funções previamente desenvolvidas na Efacec e que, a partir dos nomes são subentendidas as suas aplicações. Inicialmente carrega-se a *tag* da entidade que é guardada em *load*. Aqui é garantido que a entidade é bem carregada através da definição de um tempo

de *timeout* (`load.waitForReply(10000)==JSReplyState.TIMEOUT`) e da verificação do correto funcionamento da função utilizada para tal (`(!load.isOk()`). De seguida, é necessário obter o *target* – que é uma referência à variável, isto é, é o elemento alvo de um evento e permite a reutilização da mesma –, e, nesse *loadTarget* faz-se a escrita do estado dos *bits* de qualidade através da função `setStatus()`, do valor referente ao estado do equipamento utilizando `setValue()` e, por fim, faz-se a escrita de toda esta informação (`write()`).

Como dito previamente, todo este código tem de ser executado para cada entidade que se pretenda alterar. Inicialmente foi o implementado. Em cada momento do código que se pretendia manipular uma entidade, tinha-se estas linhas apresentadas anteriormente, no entanto, após testar para cada cenário o funcionamento dessas linhas, concluiu-se que o código ficava demasiado extenso para cada cenário e demasiado moroso e complicado de analisar. Para solucionar essa questão, desenvolveu-se uma função, denominada `loadEntity`, que assimila todas as instruções necessárias para manipular uma entidade e, deste modo, deixa de ser necessária a escrita de mais de 11 linhas de código, passando apenas a ser feita a chamada à função sempre que se intenciona manipular uma entidade. O código da função está presente de seguida. A função recebe o tipo de entidade que vai ser manipulada, a identificação da mesma (*tag*), o estado que se pretende colocar os *bits* de qualidade, e o valor que neste caso pode representar o estado a colocar na entidade, ou um valor de distância.

Função *loadEntity*

```
function loadEntity(type, tag, status, value){
  if (type == 'digital'){
        var load = JSObjectLoader.loadDigital(tag);
    } else if (type == 'analog'){
        var load = JSObjectLoader.loadAnalog(tag);
    }
  if (load.waitForReply(10000)==JSReplyState.TIMEOUT){
        JSLogger.fileLog('Equipment not Loaded');
  }
  if (!load.isOk()){
        JSLogger.fileLog('Equipment not Ok');
  }
  var loadTarget = load.getTarget();
  JSLogger.fileLog('Status: ' + status);
  if (status != 'undefined'){
   if (status == 'set'){
```

```
    loadTarget.setStatus(JSEntityStatus.ST_M_OFFSCAN);
   }else if (status == 'unset'){
    loadTarget.unsetStatus(JSEntityStatus.ST_M_OFFSCAN
);
   }
  }
  JSLogger.fileLog('Value: ' + value);
  if (value != undefined){   //Se não tiver sido
passado nenhum número no campo do value, este vai ser
undefined, o que quer dizer que este equipamento não é
atuado neste cenário
      loadTarget.setValue(value);
    }
  loadTarget.write();
  return loadTarget;
}
```

A função `loadEntity` tem a capacidade de manipular tanto entidades digitais como de medidas, é utilizada tanto nos cenários de defeito como no cenário de *reset*, e transforma a necessidade de escrever várias linhas de código sempre que se pretende alterar algo numa entidade (que multiplicando pelas várias entidades resulta em ficheiros de código muito extensos), numa necessidade de se escrever apenas uma linha para a chamada a essa função. Posto isto, a criação desta função revelou-se de elevadíssimo valor para o projeto.

Reset

Uma vez que o *script* de *reset* é executado sempre antes de ser executado um cenário de defeito, começa-se por explicar e demonstrar este ficheiro de código em *javascript*. Sabe-se que para colocar todas as entidades no seu estado inicial, significa colocar todos os disjuntores fechados, todos os IPDs e respetiva sinalização não atuados, e todas as entidades de medidas a 0. Além disso, os *bits* de qualidade referentes ao *offscan* automático e manual são desativados para todos os elementos, exceto para os IPDs e sua sinalização, onde nesse caso se mantêm ativos. No caso dos disjuntores de fim de linha o seu estado é aberto com os *bits* de qualidade desativados. Assim, no seguinte código é representada a manipulação dessas entidades, com apenas 1 exemplo para cada tipo de entidade.

```
//Disjuntor de Cabeceira
var handleDJEst = loadEntity("digital", "FSAJO-2212-
DJEST", "unset", 2);

//Medidas de Distância
var handleDJDst = loadEntity("analog", "FSAJO-2212-
DJDST", "undefined", 0);
```

```
//Interruptores de Estado
var handleIT2203 = loadEntity("digital", "FD328A2203-
ITEST", "unset", 2);

//Detetores de Defeito
var handleDF2203 = loadEntity("digital", "FD328A2203-
IT-DF", "set", 0);

var handleHIF2203 = loadEntity("digital", "FD328A2203-
DDHIF", "set", 0);

//Disjuntores de Final de Linha
var handleIT32352203 = loadEntity("digital",
"FP32352203-ITEST", "unset", 1);
```

Sabendo o código presente na função `loadEntity` (21 linhas de código) e o número total de linhas de manipulação de entidades presente no *script* de *reset* (24), é possível afirmar que, com esta função, se reduziu este *script* em mais de 480 linhas de código.

Cenário DF1+2

Para os cenários de defeito, os códigos obtidos são menores por haver necessidade de manipulação de menos entidades, isto é, só são alteradas as entidades que são atuadas no defeito idealizado, mantendo-se todas as outras no estado inicial. Inicialmente, um dos erros cometidos foi a colocação de linhas de código "recolocando" as entidades que não são manipuladas, novamente no estado inicial. Ou seja, em vez de se atuar apenas nas entidades utilizadas em cada cenário, colocou-se também linhas de código a garantir que nenhuma das outras entidades alterava o seu estado. No entanto, a colocação dessas linhas a reafirmar e a impedir a alteração do estado inicial nas entidades não manipuladas, equivalem a enviar falsa telemetria à ferramenta, uma vez que uma alteração numa dessas medidas poderia significar a deteção de um erro. Posto isto, fez-se essa correção retirando essas linhas, e segue um pequeno excerto do código *javascript* do cenário DF1+2.

```
//Disjuntor de Cabeceira
var handleDJEst = loadEntity("digital", "FSAJO-2212-
DJEST", "unset", 1);

//Interruptores de Estado
var handleIT2203 = loadEntity("digital", "FD328A2203-
ITEST", "unset", 1);
var handleIT2204 = loadEntity("digital", "FD328A2204-
ITEST", "unset", 1);

//Detetores de Defeito
```

```
var handleDF2203 = loadEntity("digital", "FD328A2203-
IT-DF", "unset", 1);
var handleDF2204 = loadEntity("digital", "FD328A2204-
IT-DF", "unset", 1);

var handleHIF2203 = loadEntity("digital", "FD328A2203-
DDHIF", "unset", 1);
var handleHIF2204 = loadEntity("digital", "FD328A2204-
DDHIF", "unset", 1);
```

Cenário D800

No cenário D800, o código do cenário de defeito torna-se ainda mais simples e mais curto, uma vez que não é alterado o estado de nenhum disjuntor de linha (apenas do disjuntor de cabeceira) e o defeito é inserido através da manipulação de entidades de medidas.

```
//Disjuntor de Cabeceira
var handleDJEst = loadEntity("digital", "FSAJO-2212-
DJEST", "unset", 1);

//Medidas de Distância
var handleDJDst = loadEntity("analog", "FSAJO-2212-
DJDST", "undefined", 800);
var handleDJZra = loadEntity("analog", "FSAJO-2212-
DJZRA", "set");
var handleDJZrs = loadEntity("analog", "FSAJO-2212-
DJZRS", "set");
```

Como se vê no código, além da alteração de estado do disjuntor de cabeceira, é colocada, como "*value*" a distância do cenário de defeito, e são alterados os *bits* de qualidade de DJZRA e DJZRS.

Cenários de IPDs com medidores de distância e cenários incongruentes

Uma vez que são atuados os dois tipos de entidades nos cenários de IPDs com medidas de distância e nos cenários incongruentes, o código desses cenários de defeito é baseado nos excertos de código apresentados anteriormente para os outros tipos de cenários, pelo que não se considerou importante apresentar um excerto de código para facilitar a compreensão.

4.1.2 Verificação de resultados obtidos

À semelhança do que foi explicado e feito no capítulo 3, em que além da explicação dos cenários de defeito idealizados, foram também verificados os resultados obtidos, também neste capítulo é feita essa verificação. Para garantir que as sequências avançadas de cada

cenário foram bem implementadas, os resultados obtidos na execução de cada uma foram novamente validados e comparados com os resultados obtidos a partir da execução dos *scripts* dbgm. No entanto, na fase do projeto que este capítulo representa, os resultados foram verificados não só a partir do GUI e do gestor de defeitos, mas também a partir da base de dados onde a informação relativamente a cada defeito fica armazenada. Esta base de dados é acessível através do programa *Oracle SQL Developer* que é um ambiente de desenvolvimento integrado (IDE – *Integrated Development Environment*) para trabalhar com a linguagem SQL nas bases de dados da *Oracle*. Isto é, é um programa que reúne caraterísticas e ferramentas de apoio ao desenvolvimento de *software* com o objetivo de agilizar o processo de criação/desenvolvimento. Para isso, acede-se à base de dados preenchendo o formulário de acesso representado na Figura 53. Neste caso, o acesso à base de dados é feito utilizando o ip da máquina DMS do cliente em questão (172.18.214.74), uma vez que é onde é sustentada a base de dados. É inserido o *username* e a *password*, é escolhido o tipo de ligação, o *hostname* (que neste caso é o ip da conexão), e, por último são inseridos o porto e o SID (SID funciona como um nome de código único que identifica a base de dados à qual se vai conectar remotamente).

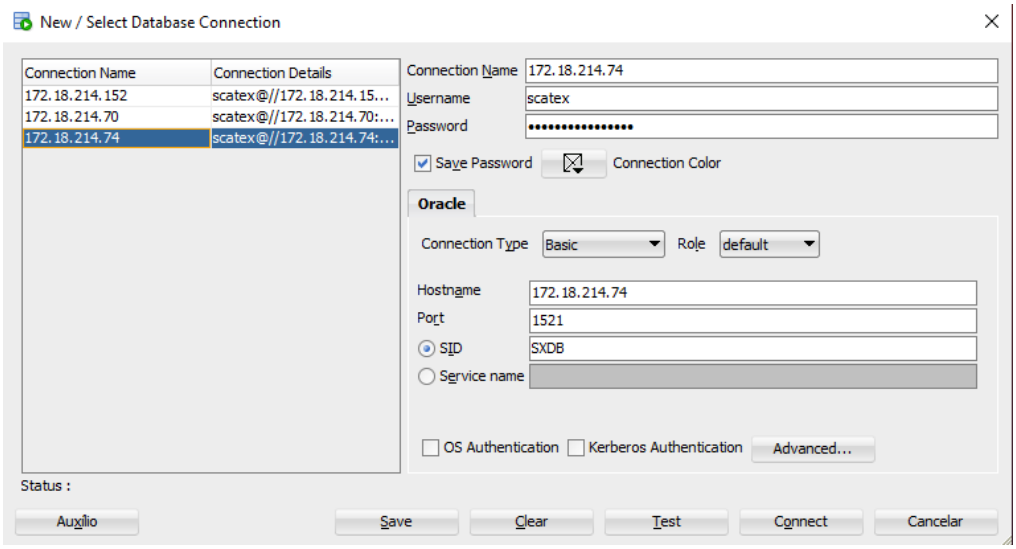


Figura 53 Formulário de acesso a base de dados do SQL Developer, preenchida.

Após ser estabelecida a conexão à base de dados, é necessário aceder à tabela onde é armazenada a informação dos defeitos. Em cada conexão existem diferentes utilizadores, em que cada um geralmente está associado a uma aplicação do ScateX e, por isso, contêm as

tabelas que guardam a informação relativamente a essa aplicação. Como referido previamente, os cálculos das zonas prováveis de defeito são feitos no gestor de defeitos pela FLF. Por sua vez, o gestor de defeitos corre numa aplicação denominada "PAS" que dá também nome ao *user* ao qual se acede para ter acesso às tabelas, o que está representado na Figura 54 (do lado esquerdo). Também na Figura 54, do lado direito, estão representadas as tabelas existentes no utilizador "PAS", das quais estão destacadas a tabela "*FAULT*" e "*FAULTLOCATION*" que é onde está guardada a informação relativamente aos defeitos. Para a verificação dos resultados apenas é necessário recorrer à segunda tabela mencionada, mais especificamente às colunas "*EQUIPMENTUID*", "*LENGTHFRACTIONMIN*" e "*LENGTHFRACTIONMAX*", as quais também são facilmente identificáveis na Figura 54 (novamente do lado direito) em cascata debaixo da tabela "FAULTLOCATION".

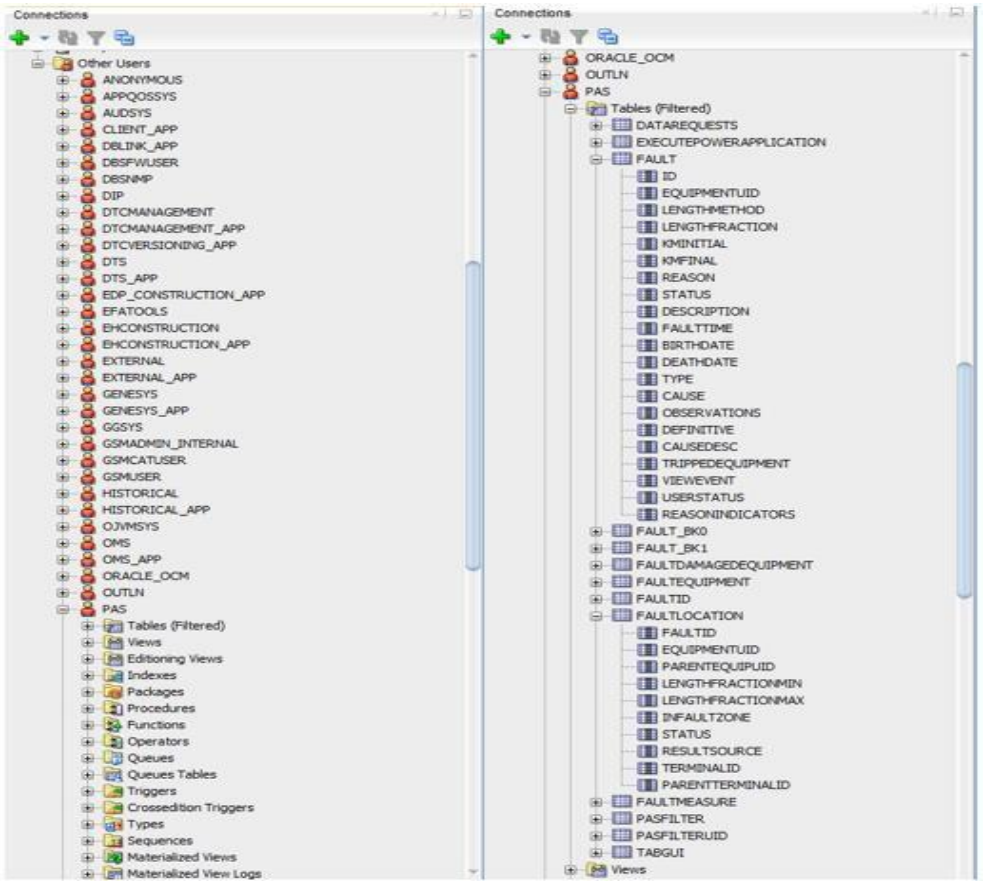


Figura 54 Tabelas do utilizador PAS na base de dados.

Para facilitar a comparação entre os resultados obtidos após execução dos cenários em *scripts* dbgm e os resultados obtidos após execução das sequências avançadas, foi construída

a Tabela 8, com os resultados obtidos na primeira situação. Os cenários de defeito apresentados na mesma apenas representam os testes com medidores de distância e com IPDs em conjunto com medidores de distância, uma vez que são os defeitos que apresentam resultados de distâncias específicos, ou seja, não são assinaladas linhas inteiras (de 0% a 100%). Para os casos em que são selecionadas linhas inteiras (cenários apenas com IPDs), apenas é importante confirmar que são selecionados os equipamentos corretos (a partir do id do equipamento) e, por perfecionismo, confirmar que foram selecionados de 0 a 100%. Na Tabela 8, as distâncias obtidas para cada defeito estão representadas em números decimais, uma vez que é dessa forma que são obtidos e armazenados os resultados na base de dados, logo, são assim apresentados para facilitar a comparação. A cada um dos cenários representados na Tabela 8, está também associado um id de defeito para mais fácil associar à Figura 55 que representa os resultados dos cenários de teste armazenados na base de dados.

Tabela 8 Distâncias prováveis de defeito obtidas em cada cenário de defeito com medidores de distância e medidores de distância com indicadores de passagem de defeito.

ID Defeito	Cenário de Teste	Nº Linha em Falha	ID Linha em Falha	LengthMin	LengthMax
153621	D200	1	(10234,33664359)	0,2657	0,3247
153622	D500	1	(10234,33664359)	0,6643	0,8117
153623	D670	2	(10234,33665961)	0	0,0963
		Não representada	(10234,33666172)	0,7025	1
		1	(10234,33664359)	0,8899	1
		11	(10234,33666169)	0	0,1704
153624	D800	11	(10234,33666169)	0,1216	0,5804
		Não representada	(10234,33666172)	0	0,7876
		2	(10234,33665961)	0,0687	0,3280
153625	D1000	11	(10234,33666169)	0,6378	1
		2	(10234,33665961)	0,3604	0,6845
153626	D1200	2	(10234,33665961)	0,6521	1
		3	(10234,33666164)	0	0,1633
153627	D670 DF1+2	1	(10234,33664359)	0	0,0963
		2	(10234,33665961)	0	0,0963
153628	D670 DF1+5	1	(10234,33664359)	0	0,1704
		11	(10234,33666169)	0	0,1704
153629	D800 DF1+2	Não representada	(10234,33666172)	0	0,787581
		2	(10234,33665961)	0,0687	0,3280
153630	D800 DF1+5	Não representada	(10234,33666172)	0	0,7876
		11	(10234,33666169)	0,1216	0,5804

	FAULTID	EQUIPMENTUID	PARENTEQUIPUID	LENGTHFRACTIONMIN	LENGTHFRACTIONMAX	INFAULTZONE	STATUS	RESULTSOURCE	TERMINALID	PARENTTERMINALID
1	153630	(10234,33666169)	(0,0)	0,121634	0,580416	5	UNKNOWN	0	1	0
2	153630	(10234,33666172)	(0,0)	0	0,787581	2	UNKNOWN	0	2	0
3	153629	(10234,33665961)	(0,0)	0,068742	0,328023	5	UNKNOWN	0	1	0
4	153629	(10234,33666172)	(0,0)	0	0,787581	2	UNKNOWN	0	2	0
5	153628	(10234,33666169)	(10234,33664359)	0	0,17038	3	UNKNOWN	0	1	2
6	153628	(10234,33664359)	(0,0)	0	0,17038	3	UNKNOWN	0	1	0
7	153627	(10234,33665961)	(10234,33664359)	0	0,0962907	3	UNKNOWN	0	1	2
8	153627	(10234,33664359)	(0,0)	0	0,0962907	3	UNKNOWN	0	1	0
9	153626	(10234,33666164)	(10234,33665961)	0	0,163325	3	UNKNOWN	0	1	2
10	153626	(10234,33665961)	(0,0)	0,652125	1	2	UNKNOWN	0	1	0
11	153625	(10234,33665961)	(0,0)	0,360434	0,684536	5	UNKNOWN	0	1	0
12	153625	(10234,33666169)	(0,0)	0,637763	1	2	UNKNOWN	0	1	0
13	153624	(10234,33666172)	(0,0)	0	0,787581	2	UNKNOWN	0	2	0
14	153624	(10234,33665961)	(0,0)	0,068742	0,328023	5	UNKNOWN	0	1	0
15	153624	(10234,33666169)	(0,0)	0,121634	0,580416	5	UNKNOWN	0	1	0
16	153623	(10234,33664359)	(0,0)	0,889932	1	2	UNKNOWN	0	1	0
17	153623	(10234,33665961)	(10234,33664359)	0	0,0962907	3	UNKNOWN	0	1	2
18	153623	(10234,33666169)	(10234,33664359)	0	0,17038	3	UNKNOWN	0	1	2
19	153623	(10234,33666172)	(10234,33664359)	0,702454	1	3	UNKNOWN	0	2	2
20	153622	(10234,33664359)	(0,0)	0,664128	0,811712	5	UNKNOWN	0	1	0
21	153621	(10234,33664359)	(0,0)	0,265651	0,324685	5	UNKNOWN	0	1	0

Figura 55 Resultados obtidos e armazenados na base de dados para cada cenário de defeito com medidores de distância e medidores de distância com indicadores de passagem de defeito.

Como dito anteriormente, a Figura 55 representa a tabela da base de dados denominada "*FAULTLOCATION*" e as colunas a serem analisadas são "*EQUIPMENTUID*", "*LENGTHFRACTIONMIN*" e "*LENGTHFRACTIONMAX*", que contém o id do equipamento em falha, o comprimento mínimo da zona provável de defeito, e o comprimento máximo dessa mesma zona, respetivamente. Além disso, na coluna "*FAULTID*" retira-se também o id da falha para associar ao respetivo defeito na Tabela 8.

Os resultados obtidos não são exatamente iguais devido ao número de casas decimais com que o resultado é retornado num caso e noutro, havendo arredondamentos que alteram ligeiramente o resultado no caso dos resultados obtidos a partir do gestor de defeitos. Ainda assim, os resultados obtidos em ambos os casos são próximos o suficiente para se desprezar as diferenças de valores, concluindo que a implementação das sequências avançadas não só foi feita corretamente e constituem os mesmos cenários de teste, como é possível assumir as mesmas como uma solução viável.

4.2. VALIDAÇÕES AUTOMÁTICAS

Para que a automatização dos testes se torne útil, também ter de ser feita uma validação automática dos resultados. Caso contrário, se a validação de cada teste tiver de ser feita manualmente, as vantagens de automatizar os testes são praticamente anuladas. Para isso, é

preciso saber quais os dados necessários para se ter uma validação completa do cenário de defeito, e ser capaz de aceder à informação dos defeitos armazenada na base de dados.

4.2.1 Informação necessária para validação de cenário de teste

A validação de um cenário reside na confirmação dos resultados obtidos para os equipamentos em falha (linhas em falha), distâncias mínimas (ou distância de início) da zona provável de defeito para cada um desses equipamentos, e distâncias máximas (ou distância de fim) da zona provável de defeito para cada uma dessas linhas. Contudo, para que a validação seja automática, o código *javascript* das validações dos cenários de teste, tem de ser capaz de identificar o cenário em questão e obter essa mesma informação para esse cenário de defeito. Agora sim, tem de ser tida em consideração a tabela "*FAULT*", cuja parte da sua composição, ou parte das colunas que a constituem, estão presentes na Figura 56. Nesta figura estão presentes, pelo menos, todas as colunas desta tabela a que se recorreu para realizar as validações, mas não só.

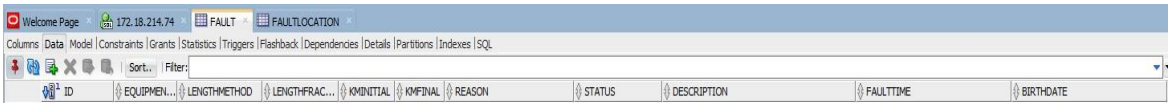


Figura 56 Algumas colunas da tabela "*FAULT*".

As validações de cada teste são executadas sempre no fim de cada cenário de defeito. Assim, para que o *script* de validação de resultados seja capaz de identificar corretamente o cenário de defeito a ser validado, recorre-se às colunas "*BIRHTDATE*", "*DESCRIPTION*" e "*STATUS*". Um defeito pode ter quatro estados: 1- Cálculo, ainda está a realizar o cálculo do defeito; 2- Ativo, o cálculo já foi realizado e o defeito ainda existe na rede; 3- Inativo, depois de ativo o defeito é inativado para deixar de ser considerado; 4- Compatibilidade, que acontece quando o cálculo da zona do defeito não é bem-sucedido e o defeito é passado para um modo de compatibilidade com a rede. Resumidamente, no fim da execução de um cenário de teste o *script*, procura o último registo da base de dados cuja descrição seja igual ao nome da zona da rede escolhida e que esteja ainda no estado ativo. Após encontrar um registo com estas caraterísticas, obtém o id do mesmo, que corresponde ao id da falha imposta ao sistema. Daqui, agora é necessário aceder à informação presenta na segunda coluna, "*FAULTLOCATION*", que tem parte das suas colunas representadas na Figura 57.

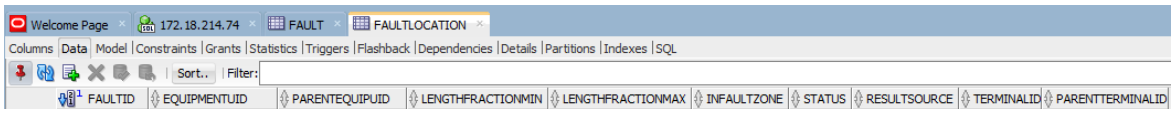


Figura 57 Algumas colunas da tabela "*FAULTLOCATION*".

Como dito anteriormente, o ID do defeito executado é obtido a partir da tabela "FAULT" que, transpondo para a presente tabela, passa a ser denominado "FAULTID". Sabendo o ID da falha, resta verificar o(s) equipamento(s) em falha e respetiva(s) distância(s) da zona provável de defeito obtida. Na Figura 58 está representado o esquema de obtenção dos dados necessários para validação dos resultados, que acabou de ser descrito.

Figura 58 Sequência de obtenção dos dados necessários para validação de resultados.

4.2.2 *Queries* em *PL/SQL*

De modo a testar a obtenção de informação a partir da base de dados, foram testadas algumas *queries* à base de dados. Uma *query* é um pedido específico de dados ou informação feito a uma tabela ou combinação de tabelas de uma base de dados. Estas *querys* foram testadas diretamente no *SQL Developer* utilizando *PL/SQL*.

PL/SQL

O *PL/SQL* é uma linguagem desenvolvida especificamente para "abraçar" declarações *SQL* dentro da sua sintaxe. Um programa *PL/SQL* é compilado através do servidor das bases de

dados *Oracle* e são guardados dentro da base de dados [46]. No *PL/SQL*, temos a adição de muitas construções procedimentais à linguagem *SQL* para ajudar a superar algumas limitações dessa linguagem [47].

Deste modo, as *querys* efetuadas foram com o intuito de obter os dados do defeito mencionados anteriormente (o nome da zona da rede, o último registo, o estado do defeito, etc). Assim, foram criadas a Tabela 9 e Tabela 10, que contêm possíveis dados (os dados inseridos em cada tabela são imaginários apenas para exemplificar) da tabela "*FAULT"* e *"FAULTLOCATION"*, respetivamente, apenas com as colunas utilizadas em cada caso.

Tabela 9 Tabela replicativa da tabela "FAULT" com dados imaginários para exemplificação da interação entre as tabelas "FAULT" e "FAULTLOCATION".

FAULT			
ID	**STATUS**	**DESCRIPTION**	**BIRTHDATE**
153495	CALCULATION	SE ALTO DE SAO JOAO | CONTINENTE	09.09.2022, 00:01:00
153494	ACTIVE	SE ALTO DE SAO JOAO | CONTINENTE	09.09.2022, 00:02:00
153493	ACTIVE	SE ALTO DE SAO JOAO | CONTINENTE	09.09.2022, 00:05:00

Tabela 10 Tabela replicativa da tabela "FAULTLOCATION" com dados imaginários para exemplificação da interação entre as tabelas "FAULT" e "FAULTLOCATION".

FAULTLOCATION			
FAULTID	**EQUIPMENTUID**	**LENGTHFRACTIONMIN**	**LENGTHFRACTIONMAX**
153495	(10234, 33664485)	0,156748	0,321856
153494	(10234, 33664359)	0,664128	0,811712
153493	(10234, 33664726)	0,458756	0,654723

Imaginemos que as três linhas com dados imaginários das tabelas anteriores, representam informação relativamente a três defeitos diferentes e que o objetivo é obter informação relativamente ao último defeito atuado e calculado. Para se obter a informação do último defeito, mais propriamente o seu ID, utiliza-se a função `max(birthdate)`, no entanto, como se sabe pretende-se que este defeito tenha acontecido na zona da rede em estudo, logo utilizou-se a seguinte linha de código:

```
SELECT id FROM PAS.FAULT WHERE description = 'SE
ALTO DE SAO JOAO | CONTINENTE' AND birthdate =
(SELECT max(birthdate) FROM PAS.FAULT);
```

O que é feito na linha de código apresentada é a seleção do ID a partir da tabela "FAULT" que se encontra no utilizador "PAS", onde a descrição tem o nome da rede em estudo e o início desse defeito ("nascimento") é correspondente ao último defeito a ter acontecido nessa mesma tabela (se tem a *birthdate* mais recente é porque foi o último registo). Para essa linha de código, e observando a Tabela 9 e a Tabela 10, o ID devolvido seria o 153495. No entanto sabe-se que não é esse o defeito pretendido, uma vez que esse defeito ainda se encontra no estado de cálculo. Posto isto, acrescentaram-se mais duas condições à linha de código anterior. A primeira refere que o estado do defeito deve ser "*ACTIVE*" e, caso apenas se colocasse esta condição, no caso destas tabelas, já se obteria o resultado pretendido (ID = 153494). Contudo, imagine-se que no defeito acabado de executar e do qual se pretende obter informação ocorre algum tipo de erro e o mesmo não fica registado na tabela (ou seja, a linha relativa ao ID 153494 não se encontra na tabela). Nesse caso, a linha de código apresentada anteriormente e já com a condição de estado do defeito, devolveria o ID 153493, o que representaria o retorno de informação errada e, quando fosse utilizado para aceder à informação da coluna "*FAULTLOCATION*" iriam ser obtidos dados correspondentes a outro defeito. Para corrigir/evitar situações desse tipo, acrescentou-se uma segunda condição que restringe o nascimento do defeito aos últimos três minutos. Isto é, o ID retornado tem de ser relativo a um defeito que, além de todas as outras condições mencionadas anteriormente, não tenha acontecido há mais de três minutos. Daí resultou a linha de código representada de seguida.

```
SELECT id FROM PAS.FAULT WHERE description = 'SE
ALTO DE SAO JOAO | CONTINENTE' AND birthdate =
(SELECT max(birthdate) FROM PAS.FAULT WHERE
description = 'SE ALTO DE SAO JOAO | CONTINENTE'
```

```
AND
(efatools.time_tools.date_to_ntime(SYS_EXTRACT_UT
C(systimestamp))-
efatools.time_tools.date_to_ntime(birthdate))<180
AND status = 'ACTIVE'));
```

Nesta linha, obtém-se o ID do último defeito ocorrido, com a descrição "SE ALTO DE SAO JOAO | CONTINENTE", com o estado ativo e que não aconteceu há mais de três minutos. A função `efatools.time_tools.date_to_ntime` converte o formato da data instantânea do sistema e o formato da data da coluna "*BIRTHDATE*" para o mesmo formato (segundos), para que possam ser comparadas e subtraídas entre si, obtendo quanto tempo passou desde o acontecimento do defeito até ao momento atual. De seguida, refere-se que essa subtração deverá ser menor do que 180 segundos (três minutos). Daqui o resultado obtido seria o ID 153494. Na Figura 59 estão representados os resultados obtidos para as *queries* indicadas acima, em que, cada linha de resultado apresentado na janela de baixo na figura, representa a informação retornada por cada uma das linhas de código da janela de cima na figura.

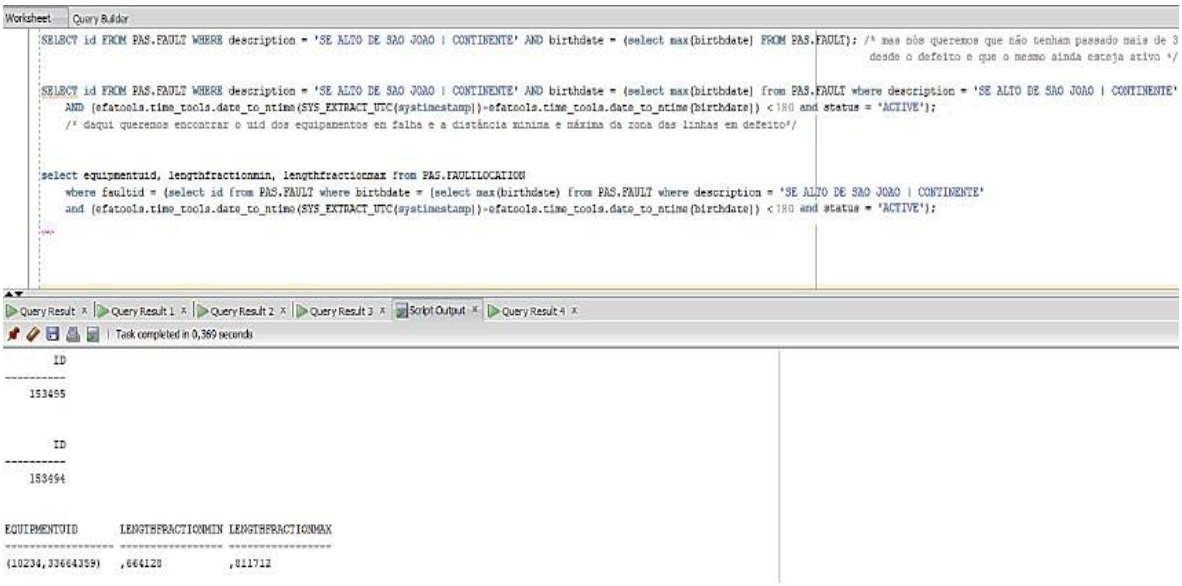


Figura 59 Resultados obtidos a partir das *queries* efetuadas no *SQL Developer*.

Com a informação obtida a partir das *queries*, confirma-se que é possível fazer pedidos à base de dados desta forma e obter a informação pretendida para realizar as verificações necessárias dos resultados obtidos após a execução de um cenário.

4.2.3 *Queries* em *Javascript*

Sabendo que a partir das *queries* testadas e executadas anteriormente é possível ter acesso à informação necessária para realizar todas as validações de um cenário de defeito, resta agora converter essas mesmas *queries* para *javascript*. Melhor dito, as *queries* executadas serão exatamente as mesmas, no entanto, o que é necessário é ser capaz de as enviar diretamente para a base de dados através de código *javascript*. Assim, primeiramente é necessário estabelecer a ligação à base de dados em questão. Para isso, e utilizando uma função interna (`JSDatabaseManager.createConnection`), utilizou-se a seguinte linha de código para criar a conexão à base de dados, em que os campos a sublinhado são campos que tiveram de ser preenchidos também quando se acedeu à mesma através do formulário de acesso presente na Figura 53.

```
var dataBaseConnection =
JSDatabaseManager.createConnection("jdbc:oracle:t
hin:@172.18.214.74:1521:SXDB", "scatex", "****");
```

Tendo esta ligação estabelecida, resta ser capaz de enviar as *queries* com os pedidos de informação às tabelas da base de dados. Inicialmente, quando se começou a desenvolver o código para obtenção dessa informação, todos os pedidos eram feitos individualmente. Isto é, primeiro fazia-se o pedido do ID com todas as condições descritas anteriormente, de seguida o pedido dos equipamentos em falha numa *query* dedicada a esse pedido, posteriormente uma outra *query* apenas para a distância mínima da zona de defeito dos equipamentos obtidos e, por fim, uma nova *query* com o pedido da distância máxima para os mesmos equipamentos. Testando dessa forma o resultado obtido foi o intendido, no entanto, resultou num código extenso e difícil de analisar, pelo que os pedidos de informação relativamente ao equipamento em falha, distância mínima e distância máxima foram todos feitos utilizando a mesma *query*. Isto é possível uma vez que se trata de informação relativa ao mesmo defeito e que se encontra na mesma linha (porque o ID da falha é o mesmo). Como dito anteriormente, o código da *query* é exatamente idêntico, uma vez que este vai ser passado e executado na base de dados. Deste modo, para que fosse possível essa lógica de funcionamento recorreu-se novamente a uma função já existente, `dataBaseConnection.executeQuery("código da query")`, que executa a *query* colocada entre aspas e retorna *null* em caso de não ter sido bem-sucedida ou retorna um

"recurso" que pode ser utilizado para obter a informação necessária. Explicando melhor o significado de "recurso" na última frase, quando numa *query* se utiliza "*SELECT*", o resultado retornado não é objetivamente o pedido feito, mas sim uma resposta que tem de ser passada a uma função cujo objetivo seja lidar com esse tipo de respostas com o intuito de obter o resultado pretendido [48] (a função utilizada será abordada oportunamente mais à frente). Isto é, no caso da *query* executada para obter o ID da falha, o resultado dessa *query* recebido pelo código *javascript* não é o ID propriamente dito, mas sim um recurso que pode ser manipulado para chegar a essa informação. Para isso, a função utilizada para obter os resultados da *query* foi a função `iterator()`, que como o próprio nome indica obtém o iterador com os dados pedidos à base de dados. Um iterador (ou *iterator*) é um objeto que define e sabe como aceder a uma sequência e, potencialmente, retornar um valor. Especificamente, um *iterator* é um objeto que usando o método *next()* implementa o protocolo dos iteradores [49]. Esse método retorna um objeto com duas caraterísticas: 1- o próximo valor na sequência do *iterator*; 2- *true* se o último valor da sequência já tiver sido utilizado [50]. Posto isto, em conjunto com a função mencionada anteriormente, foi também utilizado o método *next()* para ser possível obter e armazenar os dados pedidos. Assim, segue o código desenvolvido para obter o ID da falha que serve de exemplo ao explicado.

```
//QUERY
var getId =
dataBaseConnection.executeQuery("select id from
PAS.FAULT where birthdate = (select
max(birthdate) from PAS.FAULT where description =
'SE ALTO DE SAO JOAO | CONTINENTE' and
(efatools.time_tools.date_to_ntime(SYS_EXTRACT_UT
C(systimestamp))-
efatools.time_tools.date_to_ntime(birthdate)) <
300 and status = 'ACTIVE')");

//OBTENÇÃO DO ID
if(getId!=null){
    iterator = getId.iterator();
    if (iterator.hasNext()){
        var next = iterator.next();
        idfound = next[0];
        JSLogger.fileLog('ID Found: ' + idfound);
    }
} else {
         JSLogger.fileLog('ID not found');
}
```

Uma vez que quando se faz o pedido para obter o ID, apenas é necessário guardar um elemento, sabe-se que este se encontra na primeira posição do iterador, *next[0]*. Dessa forma, apenas é necessário guardar esse valor (neste caso na variável *idfound*) e, por isso, é o que é feito no código apresentado anteriormente. A Figura 60 representa a resposta da *query* antes de passar pela função e métodos explicados (primeira linha da figura) e a informação obtida depois de passar por esses processos (na segunda linha).

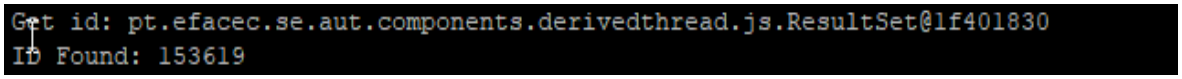


Figura 60 ID obtido a partir de *query* à base de dados.

No entanto, a função *iterator()*, além de armazenar dados individuais como é o caso anterior, também é capaz de armazenar informação em tabelas quando assim o é necessário. Exemplo disso é o segundo pedido feito à base de dados, em que numa só *query* são pedidos três elementos que têm de ser armazenados num iterador em forma de tabela. Nesse caso, a utilização da função *next()* terá de ser feita de forma a armazenar toda a informação obtida a partir da *query*. Segue o código desenvolvido para essa situação.

```
var getEquipments =
dataBaseConnection.executeQuery("select
equipmentuid, lengthfractionmin,
lengthfractionmax from PAS.FAULTLOCATION where
faultid = '" + idfound + "'");
JSLogger.fileLog('Get Equipments: ' +
getEquipments);

//OBTENÇÃO DA INFORMAÇÃO PRETENDIDA
if(getEquipments!=null){
    iterator2 = getEquipments.iterator();
    var i = 0;
    while (iterator2.hasNext()){
        var next2 = iterator2.next();
        equipments[i] = next2[0];
        locatimin[i]= next2[0, 1];
        locatimax[i]= next2[0, 2];
        i++;
    }
} else {
    JSLogger.fileLog('Equipments not found');
}
```

Como foi enunciado anteriormente e se verifica no código, foi feito um pedido para obtenção de três dados: *equipmentuid*, *lengthfractionmin* e *lengthfractionmax* da tabela *"FAULTLOCATION*", utilizando o ID encontrado na *query* anterior (*idfound*). É utilizada novamente a função *iterator()* para resgatar os dados do pedido e é seguida pela utilização do método *next()*, desta feita de uma forma diferente. Visto que a partir de um só cenário de defeito podem ser obtidos vários equipamentos em falha, significa que a um só ID de falha podem corresponder também vários equipamentos em falha com as respetivas distâncias mínima e máxima. Usando o método *next()* obtém-se as diferentes linhas da tabela e, por isso, é utilizado um ciclo *while()* para que todas as linhas do iterador sejam percorridas. Tendo a questão das linhas do iterador solucionada, resta solucionar também a questão das colunas, uma vez que em cada linha haverá três colunas correspondentes aos três pedidos de informação efetuados na *query*. Nesse caso, a manipulação do iterador foi feita da forma tradicional de manipular um *array*, em que na posição [*i*, *j*], *i* corresponde à linha – que será sempre 0 uma vez que a linha é avançada a partir do método *next()* – e *j* corresponde à coluna. Para armazenar a informação obtida foram criados três vetores para cada um dos tipos de dados pedidos, e para que haja capacidade de armazenar toda a informação no caso de se tratar de mais do que um equipamento em falha. Deste modo, supondo a existência de mais do que um equipamento em falha, todos eles, bem como as suas distâncias mínima e máxima de falha, ficam armazenados em posições semelhantes nos vetores criados para o efeito.

4.2.4 Construção do *script*

O desenvolvimento e verificação do funcionamento das *queries* foi um passo importante na construção do script das validações dos resultados, no entanto constitui apenas uma parte do mesmo. Após ser obtida a informação necessária para poder verificar os resultados, é também necessário passar os resultados esperados para o código de modo a ter o termo de comparação com a informação recebida através das *queries* e, além disso, é necessário programar quais as comparações a ser feitas e como verificar a informação. Os resultados esperados para cada cenário foram passados diretamente e individualmente, da forma que se vê no excerto de código que se segue. Nesse excerto apenas foram colocados os casos dos primeiros dois cenários para exemplificar como é feita a passagem dos resultados esperados, uma vez que para o resto dos cenários de teste é feito de igual forma. Assuma-se que todas

as variáveis apresentadas de seguida foram devidamente inicializadas e que a variável *value*, que será explicada mais à frente, serve para identificar qual o cenário a validar.

```
if (value == 1){ //DF1+2
    JSLogger.fileLog('Corre cenario 1');
    cenario = ['(10234,33666164)',
'(10234,33665961)'];
    distMin = ['0', '0'];
    distMax = ['1', '1'];
    t1 = 2;
    v1 = 4;
} else if (value == 2){ //DF1+2+3+4
    JSLogger.fileLog('Corre cenario 2');
    cenario = ['(10234,33478170)',
'(10234,33477717)', '(10234,33483960)',
'(10234,33483958)', '(10234,33616679)',
'(10234,33588486)', '(10234,33477716)',
'(10234,33483959)'];
    distMin = ['0', '0', '0', '0', '0', '0', '0',
'0'];
    distMax = ['1', '1', '1', '1', '1', '1', '1',
'1'];
    t1 = 8;
    v1 = 16;
```

Como se vê a partir do código apresentado, a variável "cenario", que é um *array*, guarda os equipamentos que se espera que a falha se encontre em cada cenário, e as variáveis "*distMin*" e "*distMax*" contêm as distâncias mínima e máxima, respetivamente, de cada equipamento armazenado em "cenario". Por último, "t1" representa o número de equipamentos em falha, e "v1" representa o número de distâncias obtidas (tanto mínimas como máximas, daí ser o dobro do número de equipamentos), no entanto, estas duas variáveis ganharão contexto mais à frente. Dependendo do cenário de defeito a ser verificado, as variáveis armazenam os valores respetivos a cada cenário, naturalmente. Conhecidos os resultados esperados por parte do *script*, e definidas as *queries* a ser executadas para obter os resultados obtidos em cada cenário, resta programar o processo de validação dos dados.

Para a comparação e validação da informação, começou por se obter e comparar o comprimento (usando a função `length`) do vetor que contém os dados previamente conhecidos (vetor "cenario") com o do vetor que armazena os dados obtidos (vetor "*equipments*"). Caso este comprimento seja diferente, significa que se obteve um número de equipamentos em falha diferente e, por isso, sabe-se à partida que os resultados obtidos

depois da execução do cenário não estão corretos porque obteve ou mais ou menos equipamentos do que os esperados. Nesse caso, o resto do *script* não é executado. Caso se confirme o mesmo número de elementos nos dois *arrays*, ambos são percorridos utilizando dois ciclos `for`, com o intuito de comparar todas as posições de um vetor com todas as posições do outro vetor. Isto é feito uma vez que o vetor que guarda os resultados obtidos após a execução de um teste, pode armazenar os mesmos numa ordem diferente cada vez que o mesmo teste é executado. Ao realizar estas comparações, sempre que é encontrado um equipamento igual em ambos os vetores, o código subtrai o valor 1 a "t1", o que significa que um dos "t1" equipamentos esperados em falha foi encontrado. De seguida, antes de se avançar de posição nos vetores que estão a ser comparados, aproveita-se o facto de se saber a posição de cada vetor em que os equipamentos são os mesmos e comparam-se também as distâncias mínima e máxima desse mesmo equipamento. Ou seja, imagine-se que "y" é a posição de um elemento no vetor "cenario" e "u" é a posição de um elemento no vetor "*equipments*", sabendo também que a cada equipamento numa determinada posição de um vetor, corresponde uma distância mínima e uma distância máxima de defeito na mesma posição mas do vetor correspondente, pode ser assim validado se as distâncias obtidas foram as corretas. Primeiramente é feita a validação da distância mínima de defeito, seguida, da validação da distância máxima obtida. Validações essas sujeitas a uma margem de erro, ou seja, o valor obtido para as distâncias mínima e máxima pode não ser sempre exatamente igual, havendo ligeiras variações desprezáveis, contudo, para um *script* de validações automáticas, qualquer diferença, por mais pequena que seja, representa um novo número e não o que se esperava. Posto isto, foi introduzida uma margem de erro de 0.1%. Caso a distância mínima e a distância máxima sejam as esperadas, subtrai-se 1 a "v1" em cada um dos casos (quando ambas as distâncias são validadas decrementa-se "v1" duas vezes). Imagine-se que uma das distâncias encontradas não corresponde ao valor esperado. Para que se torne fácil de analisar esse erro, foram criados dois vetores específicos para armazenar dados de erros. O primeiro vetor, `equipError[]`, guarda o equipamento em que foi encontrado o erro de validação, o segundo vetor, `distError[]`, armazena qual o erro encontrado na validação. Exemplificando, imagine-se que a distância mínima encontrada para o equipamento (10234,33666169) que se encontra na posição "u" não foi a esperada, nesse caso guarda-se em `equipError[h]` esse mesmo equipamento e, em

`distError[h]` coloca-se a mensagem de erro "DISTANCIA MINIMA DE EQUIPAMENTO ERRADA" ("h" define a posição dos novos vetores). Esta lógica de funcionamento encontra-se representada no seguinte excerto de código. Todos os vetores e variáveis aí presentes foram devidamente inicializados anteriormente. Nesse código, os *arrays* `distMin[]` e `locatimin[]` correspondem aos vetores das distâncias mínimas esperadas e distâncias mínimas obtidas, respetivamente, e a variável "h" é incrementada sempre após a existência de um erro de validação de dados, para que a informação de um possível erro seguinte não se sobreponha à informação já presente nesses vetores.

```
if(distMin[y] <=
(locatimin[u]+locatimin[u]*0.001)&& distMin[y] >=
(locatimin[u]-locatimin[u]*0.001)){
     JSLogger.fileLog("Equipment '" + cenario[y]
+ "' minimum distance found: '");
     v1 = v1 - 1;
} else {
     distError[h] = "DISTANCIA MINIMA DE
EQUIPAMENTO ERRADA";
     equipError[h] = equipments[u];
     h = h + 1;
}
```

Desta forma, as validações são feitas totalmente de forma automática. Sempre que se verifica um equipamento previsto, "t1" é decrementado, e o mesmo acontece sempre que é encontrada uma distância esperada, "v1" é decrementado. Assim, o grande objetivo das validações é que as variáveis "t1" e "v1" cheguem ao fim do código a 0, o que significa que todos os dados esperados foram corretamente identificados. Se assim não for, caso "t1" não guarde 0, é impressa uma mensagem de erro referindo que não foram encontrados os equipamentos esperados e, nesse caso, não é sequer feita a validação de "v1". Caso "t1" seja igual a 0, faz-se a validação de "v1" que, caso não tenha alcançado 0 é impressa também uma mensagem de erro salientando que as distâncias não foram encontradas corretamente. Além disso, no caso de "v1" ser diferente de 0 são impressos os vetores `equipError[]` e `distError[]`, para se verificar quais os erros obtidos.

4.3. CONSTRUÇÃO DA SEQUÊNCIA AUTOMATIZADA DE TESTES

Após ser desenvolvido o *script* que faz as validações automáticas dos dados obtidos, para que os testes sejam completamente automáticos, é necessário automatizar a interação entre

os diferentes *scripts* de defeito, *reset* e validações. Para isso, cada *script* tem de saber qual o *script* que vai correr a seguir e tem de ter a capacidade de o chamar para executar. Na Figura 61 está representado um esquema representativo da sequência automatizada e da interação entre os diferentes ficheiros *javascript*. O primeiro *script* a ser executado deverá ser o de *reset*, de seguida corre o primeiro cenário de teste, acabando com as validações desse mesmo cenário. No fim das validações, o objetivo é que se inicie um novo ciclo igual ao descrito, desta vez com o cenário de defeito seguinte, repetindo este ciclo até o último cenário de defeito e respetivas validações serem executadas.

Figura 61 Sequência automatizada de um ciclo de testes de localização de defeitos.

Para que isto fosse possível foi necessário ter em consideração dois aspetos muito importantes:

1. A chamada do *script* a ser executado a seguir;
2. Identificação do cenário a validar por parte do *script* de validações.

4.3.1 Chamada do script seguinte

A chamada de *scripts*, foi feita recorrendo a funções *javascript* já existentes na Efacec. Sabendo o nome da sequência avançada que queremos chamar, aplica-se a função que carrega o módulo dessa sequência avançada, e guarda-se o resultado (que é o próprio módulo) numa variável criada para o efeito. Neste caso decidiu fazer-se todas as chamadas aos diferentes *scripts* dentro dos cenários de teste, isto é, ao executar um determinado cenário de teste, este, antes de executar o defeito propriamente dito, faz a chamada ao *script* de *reset*.

Esta chamada é executada através do código apresentado de seguida em que "Pedro_Reset" foi o nome atribuído à sequência avançada de *reset*. Após carregar o módulo da sequência avançada, o mesmo é executado a partir da função `reset.runByName()`, onde poderiam ser passados parâmetros que neste caso não foram necessários.

```
var reset =
JSObjectLoader.getModule("Pedro_Reset");

if (reset == null) {
    JSLogger.fileLog('FAILED to get module' +
reset + 'due an error');
    Assert.fail();
}
reset.runByName(null, null);
```

O excerto de código apresentado faz a chamada do *script* de *reset*, no entanto o mesmo código aplica-se (naturalmente alterando o nome do módulo carregado) para a execução do *script* das validações e dos cenários de defeito. Após a chamada do *reset*, o *script* tem de aguardar que o mesmo seja totalmente executado para depois inserir o defeito na rede. Após inserir o defeito na rede, o próprio *script* tem de esperar que o cálculo da zona provável de defeito seja efetuado para no fim fazer a chamada ao *script* das validações. Uma vez que o *reset* e os restantes cenários de defeito são ficheiros de código que manipulam o estado da rede, é necessário aguardar algum tempo para que as alterações impostas por cada um desses *scripts* seja assumida e recebida pela rede. Além de esperar que essas alterações sejam recebidas pela rede, no caso dos cenários de defeito é necessário aguardar pelo cálculo da zona de defeito (como foi referido anteriormente). De modo a garantir que estes tempos de espera são cumpridos, foram inseridos *sleeps*. Estes *sleeps*, ou pausas, são facilmente inseridos utilizando `JSPause.sleep(10000)`, em que neste exemplo foram utilizados 10000 milissegundos, ou 10 segundos. No entanto, uma vez que se está perante testes automáticos cujo objetivo é serem aplicados ao sistema de vários clientes da Efacec, o tempo destas pausas pode variar de cliente para cliente, uma vez que o sistema pode demorar mais ou menos tempo a assumir alterações e/ou a realizar os cálculos para obtenção da zona de defeito. Para solucionar isso, criaram-se variáveis no ficheiro que contém os parâmetros da aplicação/componente onde as sequências avançadas são executadas – o *derivedthread*. Estas variáveis foram inicializadas com um determinado valor, aplicado ao cliente em que os testes foram criados, mas desta forma, quando houver a necessidade de alterar o valor

destes *sleeps* basta realizar a alteração nesse ficheiro em vez de alterar em todos os *scripts* em que são utilizados. O ficheiro dos parâmetros chama-se *derivedthread_parameters.xml* e é desenvolvido em *XML* (*Extensible Markup Language*) que, apesar de não ser propriamente uma linguagem de programação, é definida como *markup language* – é uma linguagem que, a partir de *tags*, serve para definir a estrutura, a formatação e/ou a relação entre partes de um documento [51], neste caso o *XML* é utilizado para guardar e transmitir dados. Um outro elemento utilizado, neste caso ao longo do *script* de validações, que também se decidiu inicializar desta forma foi a margem de erro nas distâncias mínima e máxima obtidas para a zona provável de falha. Neste cliente, como se mencionou anteriormente, definiu-se 0,1%, no entanto poderá haver clientes com margens menores ou maiores. Assim, estes três elementos foram inicializados nesse ficheiro de parâmetros e estão representados na Figura 62.

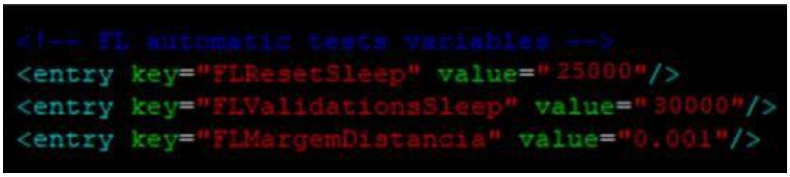


Figura 62 Inicialização de variáveis diretamente no ficheiro de parâmetros do *derivedthread*.

Para ser possível utilizar estes parâmetros ao longo dos *scripts*, os mesmos têm de ser chamados utilizando `JSParameters.getParameter("Nome do Parâmetro")`, e podem ser guardados numa variável que irá guardar o valor do parâmetro em questão. Assim, imagine-se que se guardou o valor de "*FLResetSleep*" na variável "*ResetSleep"*. Para utilizar esse valor, chama-se a função mencionada anteriormente, mas com essa variável no lugar do argumento, ou seja, `JSPause.sleep(ResetSleep)`.

O tempo de pausa denominado *"FLResetSleep*" é utilizado no fim da execução do código de *reset*, e o outro tempo de pausa é utilizado no fim de ser inserido o defeito na rede, antes de serem iniciadas as validações. Assim, cada ciclo de teste demora 50 segundos a ser executado. Tendo em conta que são 17 testes, o tempo obtido para a execução dos 17 ciclos foi à volta dos 15,4 minutos. No entanto, antes de se conseguir reduzir o tempo de execução dos testes para 15,4 minutos, havia surgido um problema, uma vez que começou por se aplicar *sleeps* que cumprissem as caraterísticas originais do cliente em questão originando

um tempo de execução dos testes superior a 30 minutos, o que era muito elevado. Após análise de um especialista, chegou-se a uma solução que envolveu alterar o valor de um parâmetro definido do lado do produto, que influencia no tempo de cálculo da zona de defeito. Este parâmetro correspondente ao tempo de espera que o sistema aguarda antes de efetuar o pedido de cálculo à ferramenta de localização de defeitos, que serve para aguardar pela estabilização das medidas quando existem alterações. Contudo, verificou-se que este tempo de espera podia ser diminuído, não tendo sido necessário alterar caraterísticas do cliente. Diminuindo este tempo, teve de ser certificado que os cálculos da zona de defeito se mantinham corretos e que tinham tempo de ser executados, pelo que após essa verificação, obteve-se o tempo de execução do ciclo completo dos 17 testes mencionado anteriormente.

4.3.2 Identificação do cenário de teste a validar

Como já foi referido previamente, para que o *script* de validação dos dados obtidos funcione corretamente, este tem de ser capaz de identificar qual o cenário de defeito que correu em último lugar e que pretende validar. Caso contrário, iria estar a validar um determinado cenário de defeito, com dados de outro cenário diferente. Para isso, foi utilizada uma entidade de medida cujo valor é alterado pelo cenário de defeito a ser executado. Esta entidade já existia no sistema, no entanto não era utilizada em nenhum processo e a alteração do seu valor não tem qualquer efeito, pelo que foi aproveitada para este caso. Após ser executado um defeito, este altera o valor desta entidade para o número do seu cenário (por exemplo, o cenário 2 coloca o valor 2) para que, de seguida, ao serem feitas as validações, o *script* tenha acesso a esse valor e tome conhecimento de qual o cenário que vai validar. O valor do cenário é colocado na entidade recorrendo à função `loadEntity` explicada anteriormente, e é obtido nas validações através da utilização de três funções: 1º- carrega-se a entidade utilizando `handle=JSObjectLoader.loadAnalog("FSAGUI5TP1-RTTOM")`, em que a *tag* da entidade fica entre parênteses; 2º- obtém-se o *target* da variável *handle* utilizando a função `t = handle.getTarget()` já explicada ao longo da dissertação; 3º- por fim obtém-se o valor desse *target* utilizando `value=t.getValue()`. A partir daqui, com o valor da entidade, é possível identificar corretamente qual o cenário a validar. A Figura 63 representa o valor da entidade em questão aquando da execução do cenário 1 (cenário

representado na parte superior da imagem) e do cenário 17 (parte inferior da imagem), onde se verifica a alteração no seu valor identificando o cenário atuado.

Figura 63 Alteração no valor da entidade de identificação do cenário executado.

Tendo a sequência de testes automatizada, após testar a mesma verificou-se o correto funcionamento de todos os *scripts*. Até aqui, a informação obtida após a execução dos testes é analisada no *log* do módulo respetivo a cada sequência, sendo que o *log* mais importante de analisar é o do módulo das validações, uma vez que é lá que está presente a informação relativamente aos dados obtidos em cada ciclo de testes. Na Figura 64 está presente a informação obtida após a execução do cenário de defeitos DF1+2 e obtenção da informação esperada (teste bem-sucedido) e na Figura 65 verifica-se o *log* de um cenário que encontrou um erro na informação obtida a partir da base de dados. Neste dois *logs* podem ser obtidos todos os dados necessários relativamente à execução do cenário. Nessas figuras, observam-se ainda os parâmetros "*Get id*" e "*Get Equipments*" com uma linha pouco usual associada.

Essa linha representa o resultado de uma *query* antes de serem aplicadas as funções para extrair a informação que daí advém (o anteriormente denominado “recurso”).

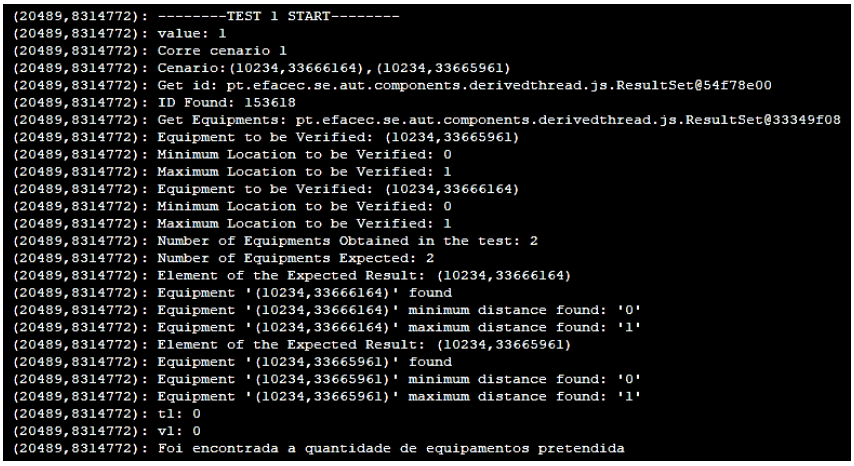


Figura 64 *Log* do cenário DF1+2 após uma correta execução do mesmo.

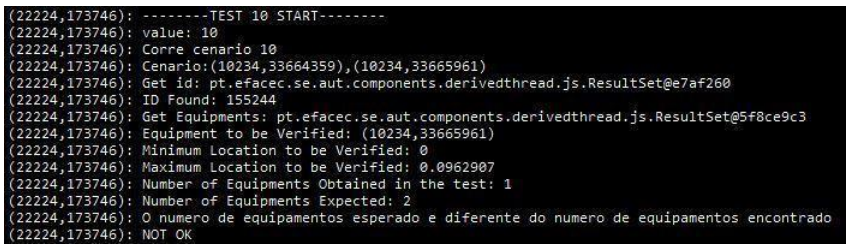


Figura 65 *Log* do cenário D670 DF1+2 após uma execução com erros.

4.4. *SCRIPT* DE PRÉ-REQUISITOS DA REDE

Após a criação dos *scripts* de *reset*, de validação de dados e dos cenários de defeitos a testar, a experiência da empresa conduziu a que fosse desenvolvido um *script* que, antes de executar os ciclos de teste, verificasse o estado da rede e que este era o ideal para avançar com os testes automáticos. Esta sugestão surgiu uma vez que caso o estado da rede não seja o ideal e os testes automáticos sejam executados, é possível que os resultados não sejam os esperados e que o resultado de todos eles possa estar errado, isto é, que após a validação dos resultados obtidos estes sejam diferentes dos esperados. Caso isso acontecesse, o que se iria concluir seria uma falha no cálculo da zona provável de defeito, o que iria ser uma conclusão errada, e a deteção da real causa do erro poderia levar muito tempo a ser detetada uma vez que as atenções iriam estar a ser orientadas para uma falsa causa. Com este *script*, caso a

rede não esteja no estado ideal, os testes automáticos são executados de qualquer maneira, uma vez que pode acontecer de serem bem executados, mas: 1- é obtida a informação de que a rede não se encontrava/encontra no estado ideal, o que tem de ser entendido e solucionado o motivo; 2- caso os testes não tenham o sucesso esperado, sabe-se que o mais provável é ter sido essa a causa.

4.4.1 Pedido de informação – *datarequest*

A base da obtenção dos pré-requisitos da rede é um pedido de informação, denominado *datarequest* (DR), onde podem ser enviados vários pedidos, denominados eventos, consoante a informação pretendida. Neste caso, a informação que se pretende obter é relativa ao estado da rede, mais propriamente, a zona da rede que está a ser testada. No entanto, para que se consiga compreender os dados obtidos a partir desse pedido, é necessário conhecer os possíveis estados da rede relevantes para o estudo em questão:

- *Live* – É o estado ideal da rede que significa que esta se encontra totalmente energizada;
- *Faulted* – Corresponde a um estado de falha da rede em que a mesma se encontra simultaneamente energizada e ligada à terra;
- *Earthed* – Neste estado a rede encontra-se diretamente ligada à terra;
- *Dead* – Como o próprio nome indica, neste estado a rede encontra-se "morta" e sem qualquer energia.

De salientar que o estado ideal da rede é *Live*, no entanto, para que os testes automáticos sejam executados corretamente e sem nenhuma implicação basta que a rede esteja energizada, ou seja, pode encontrar-se também no estado *Faulted*, ainda que não seja a situação ideal. Conhecendo os possíveis estados da rede começa por se testar o *datarequest* recorrendo a uma aplicação denominada *event_simulator*. Para enviar um pedido de informação deste tipo, é necessário recorrer ao evento "*GetEquipmentDynamics*" que obtém várias informações relativamente ao equipamento pretendido. Neste evento é necessário enviar o Uid do equipamento, ou equipamentos, dos quais se pretende obter informação.

Para o *script* que se pretende desenvolver é importante saber o estado da primeira linha da zona da rede escolhida, uma vez que conhecendo o estado desse equipamento, o estado das restantes linhas será o mesmo (a não ser que haja falhas nos aparelhos de corte, uma vez que todos eles serão colocados no seu estado inicial com o *reset*). A janela desta aplicação está representada do lado esquerdo da Figura 66 onde o Uid presente e enviado no *datarequest* é o da linha 1. Pressionando "*Send* DR" nessa janela, surge a janela pop-up da direita na mesma figura, com dois campos a ser preenchidos relativos aos IDs do pedido universal e do pedido propriamente dito – veja-se o pedido universal como um grupo que contém vários pedidos possíveis, isto é, dentro de um pedido universal pode haver vários pedidos possíveis, algo que não acontece neste caso – que nesta situação será o mesmo.

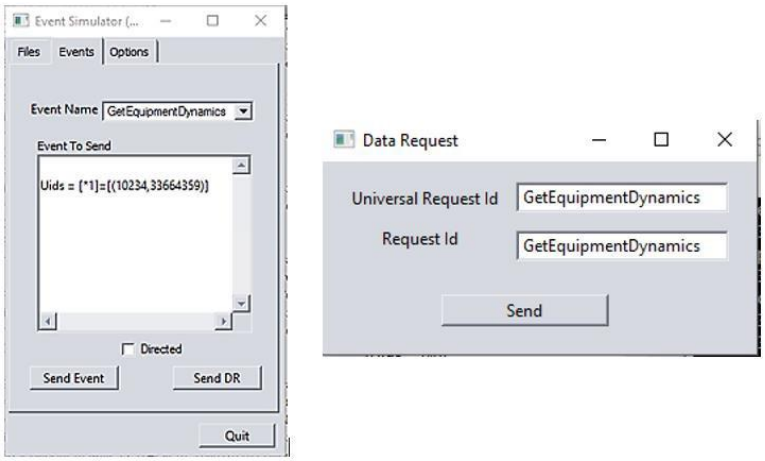


Figura 66 Janelas de envio de datarequest.

Após o envio deste pedido, a informação requisitada é escrita em *log* – que é um processo de registo de informação relativamente a determinados eventos – que pode ser consultado utilizando um editor de texto que, neste caso, foi o *Notepad++*. Aqui surge o primeiro problema relacionado com a criação deste *script*, mais propriamente, com o pedido de informação que tem de ser feito. Ao analisar a informação obtida a partir do *datarequest* verificou-se que o estado da rede estava *Faulted*, ou seja, a rede estava em falha quando se esperava o estado *Live*. Esta informação pode ser analisada a partir da Figura 67, onde está presente parte do *log* deste *datarequest*. A figura foi cortada de modo que o texto presente na mesma seja legível. Neste *log* verifica-se que o parâmetro "*isFaulted*" se encontra a 1, o que significa que esse é estado da linha naquele momento (quando um dos possíveis estados

se encontra a 0 significa que não é o estado atual). Na Figura 68 verifica-se o estado falhado da linha a partir do painel de caraterísticas da mesma.

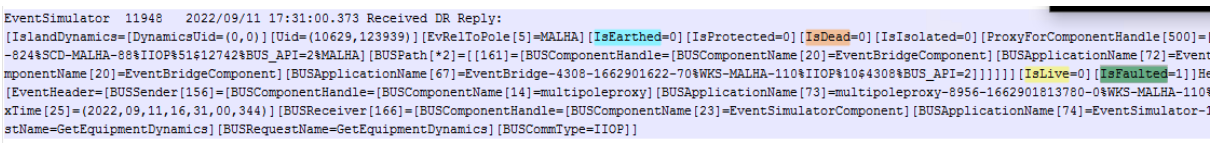


Figura 67 *Log* do pedido *datarequest*.

Figura 68 Janela de caraterísticas da linha 1 da rede com representação do estado de energização da mesma.

Como dito anteriormente, este estado encontrado não tem qualquer influência nos testes idealizados, no entanto, para que o *script* de pré-requisitos cumpra a sua função, a rede tem de estar *Live*. Desta forma, o passo seguinte foi tentar perceber o porquê da linha estar em falha, de modo a resolver o problema. Quando se explicaram os diferentes estados da rede, explicou-se também que quando uma linha está falhada significa que a mesma se encontra simultaneamente energizada e ligada à terra. Para descobrir o local da ligação da linha à terra utilizou-se uma funcionalidade do ScateX denominada "*Trace* às terras" que, aplicada a uma linha, salienta a mesma até ao local da ligação à terra. Com essa linha salientada, consultou-se o geográfico da rede – que mostra todas as linhas da rede de distribuição de energia de forma a conseguir perceber qual a linha identificada pelo trace. Na Figura 69 está presente a representação do geográfico, com a linha assinalada pelo *trace* a vermelho e, na extremidade inferior da linha assinalada, encontra-se a zona da rede em estudo a azul-turquesa. Verifica-

se, na extremidade oposta à rede testada, que a linha vermelha termina numa outra SE denominada “RELVINHA”, que está ampliada.

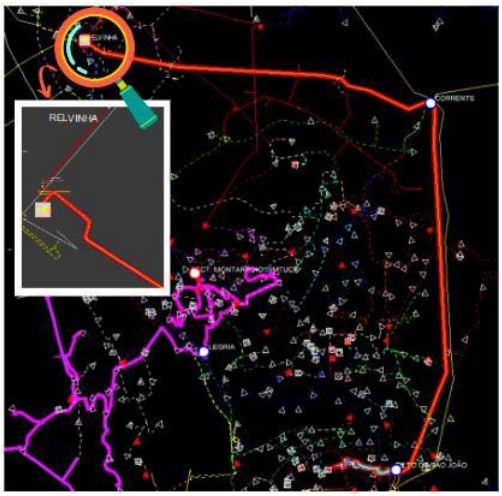


Figura 69 Geográfico da rede com representação da linha com ligação à terra.

Saber que a ligação à terra está estabelecida numa outra SE é um passo importante, mas não é ainda informação suficiente para resolver o problema exposto anteriormente, pelo que foi necessário analisar o esquemático dessa SE, que está representado na Figura 70. Nessa figura, vêm-se quatro disjuntores assinalados a vermelho pelo *trace* às terras, sendo que um deles estabelece ligação a uma terra (o que está ampliado).

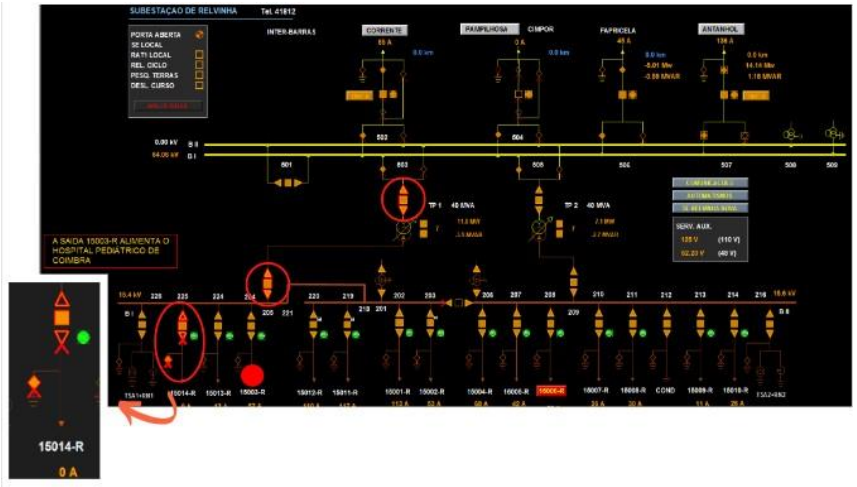


Figura 70 Esquemático da subestação de Relvinha com *trace* de ligação à terra e representação do disjuntor que estabelece essa mesma ligação.

A solução para a linha deixar de estar falhada parece evidente: abrir o condutor de modo que este deixe de estabelecer ligação à terra e a rede passe ao estado ideal. No entanto, esta solução não pode ser aplicada uma vez que se aquele disjuntor se encontra fechado, é porque o cliente configurou desse modo e, tratando-se de configurações do cliente, estas não podem ser alteradas. Ainda assim, de modo a confirmar e a testar se o disjuntar era a causa da falha da linha, este foi aberto manualmente e a energização da rede foi verificada, uma vez mais, a partir do painel de caraterísticas da mesma, na aba de energização. Na Figura 71 verifica-se que com o disjuntor aberto a rede realmente passa a estar energizada e no estado ideal.

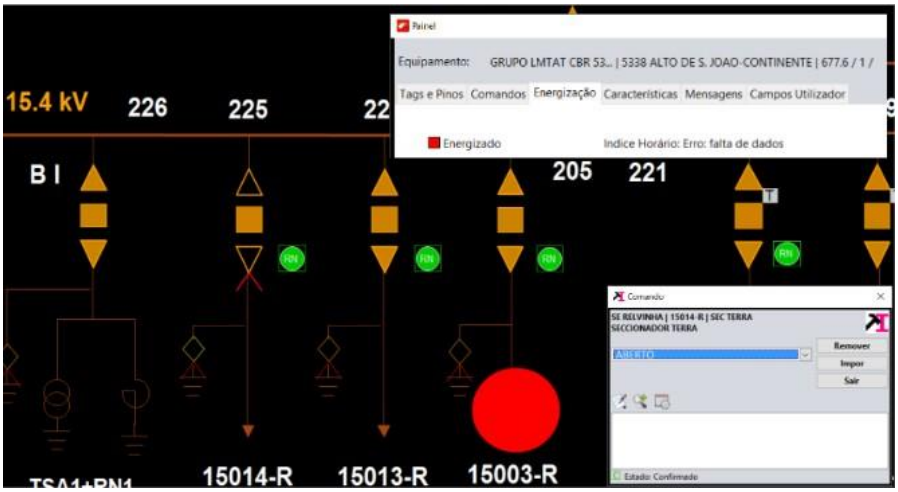


Figura 71 Energização da linha 1 após abertura do disjuntor-terra.

Se o *script* de pré-requisitos cumprir o seu objetivo sem ser aplicada qualquer solução a este problema, este não vai verificar que a rede se encontra no estado *Live*, mas sim *Faulted*. A solução encontrada baseia-se num *workaround* de modo a não alterar permanentemente o estado do disjuntor indo contra as caraterísticas impostas pelo cliente. Uma vez que o *script* de pré-requisitos vai ser desenvolvido em *javascript* à semelhança dos restantes *scripts*, a solução encontrada envolve alterar o estado do disjuntor durante o *datarequest* feito ao sistema, repondo o estado inicial do mesmo após receção da informação pretendida. Sabe-se que a única razão para a rede não estar no estado ideal é o facto desse disjuntor se encontrar fechado (o que foi verificado e demonstrado anteriormente a partir da Figura 71), pelo que abrindo-o deverá resultar na mudança do estado da rede para *Live*. Caso mesmo assim o estado da rede recebido seja outro, significa que a rede está com um novo problema não resultante de caraterísticas do cliente e que tem de ser resolvido.

4.4.2 Desenvolvimento do script

Como foi referido, a solução encontrada para que o disjuntor ligado à terra não mantivesse a rede falhada (estado *faulted* explicado anteriormente), foi a sua abertura antes do pedido de informação e o seu fecho a seguir ao mesmo. Essa alteração do seu estado foi novamente feita recorrendo à função `loadEntity`. Para ser possível enviar o equipamento pretendido no *datarequest*, tem de ser criada uma variável utilizando a função `EFAStringArrayValue()` – já existente e que, de uma forma resumida, cria um vetor de *strings* – para que o valor enviado seja reconhecido pelas restantes funções. Para colocar a *tag* do equipamento dentro do vetor criado (o vetor criado tem o nome de *keys*), é necessário recorrer a `insertValue` e, o valor a inserir dentro do vetor tem de ser definido como um *EFAStringValue*, ou seja, `keys.insertValue(new EFAStringValue("(10234,33664359)"))`.

Para o envio do *datarequest* e confirmação da informação obtida foi criada uma função, `checkLineState`, que recebe como parâmetro o Uid do equipamento (que neste caso é a variável *keys*). A função começa por criar um evento `GetEquipmentDynamics`, onde define como valor do parâmetro Uids (rever Figura 66) a variável *keys*, isto é, a *tag* da linha da qual se quer saber o estado. De seguida é criado o pedido *datarequest* propriamente dito, com o preenchimento da janela *pop-up* do *event_simulator* mencionada anteriormente com os IDs do pedido universal e do pedido propriamente dito, e com o envio do evento acabado de criar. Depois da criação do pedido, este *datarequest* é inserido em *handle* e, por fim são requisitados os seus eventos. Serve de demonstração o código apresentado de seguida que contêm desde a criação do evento, até à obtenção dos eventos pretendidos do *datarequest*.

```
var event = new Event("GetEquipmentDynamics");
event.setValue("Uids", Uid);

var dataRequest = new
DataRequest("GetEquipmentDynamics",
"GetEquipmentDynamics", event);

var handle = JSBus.pushDataRequest(dataRequest,
true);
var eventsArray = handle.getEvents();
```

Tendo o pedido de informação sido enviado e a informação registada em *log*, é necessário fazer com que o *script* tenha a capacidade de obter a informação pretendida e verificar o

estado da linha. A informação obtida a partir do *datarequest* é organizada da seguinte forma:

é retornado um vetor com diferentes eventos em que cada um desses eventos contém um outro vetor com diferentes caraterísticas, ou seja, essas caraterísticas estão armazenadas num vetor dentro de um vetor. O evento do qual pretendemos obter informação encontra-se na quarta posição do (primeiro) vetor e chama-se "*IslandDynamics*". Dentro desse evento encontram-se os estados da rede que é a informação pretendida e útil para o *script* e da qual se pretende obter o valor (se um determinado estado está a 0 ou a 1). Dessa forma, sabendo que o evento *IslandDynamics* se encontra na quarta posição do vetor `eventsArray`, para obter o valor que se pretende de cada estado da rede utiliza-se a função `getValue`, com o nome do parâmetro que se pretende a ser passado como argumento, ou seja:

```
var isL = eventsArray[3].getValue("IsLive");
var isD = eventsArray[3].getValue("IsDead");
var isE = eventsArray[3].getValue("IsEarthed");
var isF = eventsArray[3].getValue("IsFaulted");
```

O valor de cada um dos parâmetros é armazenado na variável respetiva e resta verificar o estado da rede. Para isso utilizou-se uma estrutura de condição `if…else` para verificar se a rede não está no estado ideal e, caso se verifique, foi inserida uma outra estrutura de condição `if…else` para verificar todos os restantes estados. Os outros estados apenas são verificados caso o estado *isLive* não se verifique. Foi colocado um excerto da parte do código descrita em que verifica o estado *isLive* diferente de 1 (ou seja, não ativo) e a verificação do estado *isDead* que é feita da mesma forma que para os outros estados.

```
if (isL != 1) {
   JSLogger.fileLog("A ilha nao se encontra
energizada");
   if (isD == 1) {
      JSLogger.fileLog("Estado: Dead");
      JSLogger.fileLog("IsDead: " + isD);
   }
```

É após estas validações que o disjuntor que estabelecia ligação à terra é colocado no seu estado inicial. Tendo o *script* de verificação de pré-requisitos completo surge o segundo problema relacionado com o desenvolvimento do mesmo. No momento da execução do *script* é obtido um erro que refere o não reconhecimento de um tópico denominado "*Dynamics*". Na Efacec o modelo de dados define a existência de tópicos, eventos e atributos. Cada tópico possui um ou mais eventos definidos e, que por sua vez possui uma série de atributos. Neste caso, o tópico *Dynamics* é necessário para que se possa enviar um

datarequest ao sistema sobre o evento "*GetEquipmentDynamics*". Para isso, acrescentou-se o *Dynamics* aos restantes tópicos num ficheiro já existente e cuja função é de armazenar os tópicos a serem reconhecidos/importados no componente *derivedthread*. A inserção deste tópico diretamente nessa lista traz um novo problema, uma vez que o facto de passar a estar presente nesse ficheiro significa que vai ser recebida mais informação e que o componente *derivedthread* poderá ficar bloqueado devido a falta de espaço (além da informação relativamente ao *datarequest* que se pretende enviar, este tópico "abre portas" a mais informação inútil e que anteriormente o *derivedthread* não recebia). A solução que se desenvolveu neste caso foi, à semelhança da abertura e fecho do disjuntor-terra, fazer a subscrição e retirar a subscrição do tópico, respetivamente antes e depois do pedido de informação ser enviado. Para que isso seja possível, é necessário fazer a publicação do tópico seguido da sua subscrição e, depois de obtida a informação pretendida, retirar a publicação e subscrição do mesmo, como se vê no excerto de código.

```
JSBus.publish("Dynamics");
JSBus.subscribe("Dynamics");
/*Obtenção da informação pretendida*/
JSBus.unpublish("Dynamics");
JSBus.unsubscribe("Dynamics");
```

Desta forma, a verificação dos pré-requisitos da rede fica completa e funcional, e a informação obtida a partir deste *script* pode ser verificada no *log* do módulo da sequência avançada, como se vê na Figura 72. Foram escritas para *log* várias informações durante a execução do *script* de modo que, caso o código não fosse bem executado, saber até onde este tinha corrido. Verificam-se informações como o Uid enviado, o evento impresso na sua totalidade para confirmar que foi bem enviado, o tamanho do vetor recebido com os diferentes eventos, etc. Nesta figura, o *log* apresentado foi cortado de modo a tornar o texto presente no mesmo legível, e verifica-se a partir das últimas duas linhas do mesmo que a linha se encontrava energizada e no estado ideal.

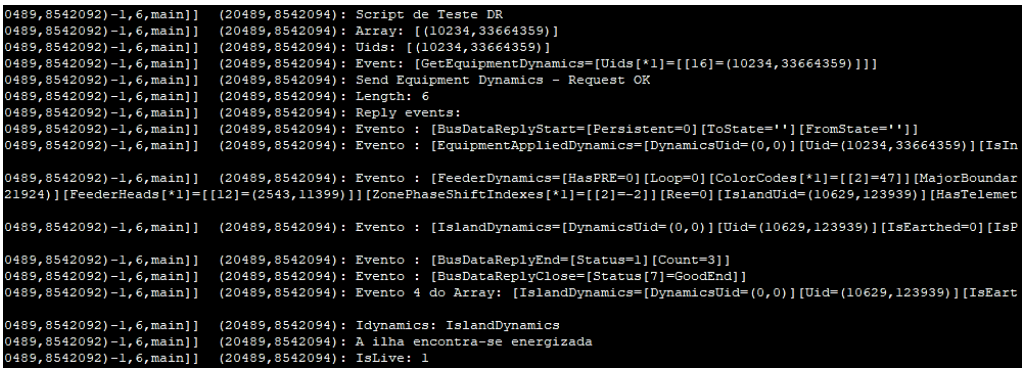


Figura 72 *Log* do módulo da sequência avançada correspondente ao *script* de pré-requisitos da rede, onde a mesma se encontra energizada.

4.5. INTEGRAÇÃO DOS *SCRIPTS* NA MÁQUINA

Antes de ser criado o *job* de testes automáticos relativo à localização de defeitos, os *scripts* desenvolvidos anteriormente para os cenários de defeito, *reset*, validações e pré-requisitos, têm de estar presentes na máquina (em pastas próprias) e não em formato de sequências avançadas no editor. Para que se possa avançar para a criação do *job* na *pipeline* de testes automáticos, primeiramente é necessário colocar todos estes *scripts* na máquina, nomeadamente na pasta de testes, onde vão servir de *template* para a criação de novas sequências avançadas. De seguida, são desenvolvidos os ficheiros de configuração que fazem uso desses *templates* na criação das novas sequências avançadas mencionadas, diretamente no gestor de sequências de comando, isto é, as sequências deixam de ter sido criadas no editor de mundo real e passam a ser criadas pela máquina através destes ficheiros de configuração e carregadas diretamente na base de dados. De modo a executar automaticamente estas novas sequências, são criados um ficheiro *python* que faz a chamada das mesmas pela ordem pretendida, e um ficheiro executável cujo objetivo é "lançar" o *script python*. Neste capítulo são criados esses ficheiros necessários para que o *job* possa ser criado. Na Figura 73 está representada a sequência de interação dos ficheiros mencionados anteriormente, isto é, os ficheiros de código JS são utilizados como *template* nos ficheiros de configuração para a criação de novas sequências avançadas, que por sua vez vêm a sua ordem de chamada/execução definida no ficheiro *python*, que por último é chamado a executar pelo ficheiro FL2.sh.

Figura 73 Interação dos diferentes ficheiros criados na máquina para execução dos testes automáticos.

4.5.1 *Templates javascript* e ficheiros de configuração

Os ficheiros de código *javascript* que neste passo são inseridos na pasta de testes da máquina em nada diferem, no que diz respeito ao código, das sequências avançadas, contudo foi acrescentado "*template.js*" no fim do nome de cada um destes *scripts*. Estes ficheiros têm de estar presentes nesta pasta para que possam servir de referência aos ficheiros de configuração.

Um ficheiro de configuração define parâmetros, opções, definições e preferências aplicadas a sistemas operativos, dispositivos de infraestrutura de aparelhos e aplicações num contexto de *information technology* (*IT*) [52]. No caso deste projeto, nos ficheiros de configuração são definidas, como dito anteriormente, as condições/referências para a criação de novas sequências avançadas, desta feita diretamente na base de dados da máquina em questão. Neste ficheiro, entre outras configurações, é definido o nome (da nova sequência avançada) que irá ser usado pelo ficheiro *python* para executar a sequência avançada respetiva, e é definido também qual o ficheiro *javascript* a utilizar como *template* na criação da nova sequência avançada. Na Figura 74 está representado o ficheiro de configuração do cenário de defeito D1000, onde na primeira linha é definido o nome entre parênteses retos, e de seguida é definido o que vai ser criado, que neste caso é um módulo de uma sequência avançada. Além disso, o ficheiro é configurado como *template* com uma breve descrição, e com o nome do ficheiro a ser tido em conta como *template*, bem como a sua localização.

```
[FL2_D1000]
stype = module
istemplate = True
description = Fault location 2 D1000
template_name = fl2.d1000.template.js
package = tests.fl2
```

Figura 74 Ficheiro de configuração do módulo da sequência avançada do cenário D1000.

4.5.2 *Ficheiro python e ficheiro de execução*

Como foi referido anteriormente, o ficheiro *python* é onde fica definida a ordem de execução das novas sequências avançadas, uma vez que é aqui que as mesmas são chamadas a correr. Este ficheiro *python* é necessário uma vez que para a execução dos testes automáticos já existentes foi criada uma *framework* nesta linguagem de programação. Tendo os nomes das sequências avançadas ficado definidos nos ficheiros de configuração, são esses os nomes a ser inseridos na chamada de cada um deles. Para isso, foi usada uma função na qual se passam três argumentos: o primeiro argumento é o nome do ficheiro que queremos executar, o segundo, que no nosso caso não foi passado, pode ser uma variável, um valor, ou algo que se pretenda passar ao ficheiro que se está a chamar, ou seja, é um argumento "livre", e o terceiro é um valor de *timeout*, ou seja, caso esse tempo seja ultrapassado a função interrompe a sua execução. Na Figura 75 está representado parte do ficheiro de código *python*, em que se verifica a chamada do *script* de pré-requisitos, seguido do primeiro ciclo de testes, equivalente ao cenário DF1+2, e a chamada do ciclo de testes seguinte que inclui o cenário DF1+2+3+4.

```
def run(self):
        rc = self.run_test_from_script('FL2_PRE_REQ', None, 300)
        rc = self.run_test_from_script('FL2_RESET', None, 300)
        rc = self.run_test_from_script('FL2_DF1e2', None, 300)
        rc = self.run_test_from_script('FL2_VALIDATIONS', None, 300)
        rc = self.run_test_from_script('FL2_RESET', None, 300)
        rc = self.run_test_from_script('FL2_DF1e2e3e4', None, 300)
        rc = self.run_test_from_script('FL2_VALIDATIONS', None, 300)
```

Figura 75 Representação de uma parte do ficheiro *python* que realiza a chamada para execução das sequências avançadas.

O ficheiro executável é um *shell script* – é composto por uma lista de comandos que são executados pela *Unix shell* (que é um interpretador de linhas de comando) –, tem a extensão ".sh", e serve, como o próprio nome diz, para executar o ficheiro *python*. Este é o ficheiro colocado como referência na criação do *job* dos testes automáticos, isto é, é o ficheiro chamado a executar pelo *build* do *job*.

5. RESULTADOS TANGÍVEIS

Ao longo de toda a dissertação foram mencionados resultados obtidos e objetivos cumpridos, no entanto todos esses se tratam de passos importantes em direção a um grande objetivo final, a integração de testes automáticos de localização de defeitos na *pipeline* de teste já existente. No início da tese foram definidos vários passos a serem executados até finalizar o projeto, com o intuito de ser desenvolvido um *job* de testes automáticos para testagem da ferramenta de localização de defeitos na rede de distribuição de energia. Todos os passos definidos foram cumpridos com sucesso e os resultados objetivos do projeto são definidos neste capítulo.

5.1. *PIPELINE* DE TESTES AUTOMÁTICOS

A arquitetura dos testes automáticos baseia-se em *builds* automáticos. Um *build* automático é o resultado da execução única de um *job* [53], isto é, sempre que o *Jenkins* executa um *job*/projeto, este compila a configuração do mesmo de modo a executar os *steps* definidos. A cada execução deste *job* chama-se "*build*" [54]. Após a execução (com sucesso) dos *builds* automáticos, é feita a instalação de uma nova versão do sistema que, por sua vez, em caso de sucesso é seguida pela execução da *Pipeline* de testes automáticos. Tanto o *build*, como a instalação da nova versão e os testes automáticos correspondem a *jobs* do *Jenkins* – tarefas automatizadas para atingir determinados objetivos, por exemplo a testagem de aplicações e/ou ferramentas. Esta sequência de projetos está definida na Figura 76.

Figura 76 Sequência de processos até despoletar a *pipeline* de testes automáticos.

Uma *pipeline* de *Development Operations* (*DevOps*) define-se como um conjunto de ferramentas e processos automatizados que permitem o trabalho coeso entre profissionais de desenvolvimento e operações. Tipicamente inclui automação de *builds*/integração contínua (*continuous integration – CI*), testes de automação, validação e comunicação de resultados. A palavra "contínua" é vista como uma caraterística diferenciadora de uma *pipeline* de *DevOps*, uma vez que inclui integração e entrega contínuas – *Continuous Integration and Continuous Delivery* (*CI/CD*) é um método desenvolvido para entrega de produtos/aplicações com frequência aos clientes, utilizando a automação nos processos de desenvolvimento dos mesmos. Com o *CI/CD* podem ser solucionados rapidamente problemas originados pela integração de novos códigos por ser aplicada a monitorização contínua das aplicações, incluindo em etapas de teste e integração [55] –, e *feedback* e operações contínuos. Em vez de testagem ou desenvolvimentos singulares, a *pipeline* corre continuamente. Uma *Jenkins Pipeline*, por sua vez, é um conjunto de *plug-ins* que suportam a implementação e integração de *pipelines* de *CI/CD* no *Jenkins*.

Como dito anteriormente, a execução dos testes automáticos faz recurso a uma *framework* em *python* criada internamente, e no final de cada processo são enviadas notificações de sucesso/insucesso para o email dos desenvolvedores/responsáveis de cada processo e para canais específicos do *Slack* – aplicação de mensagens para empresas que conecta as pessoas à informação.

A execução, acompanhamento e o sucesso/insucesso dos testes automáticos podem ser monitorizados através da *pipeline* de testes automáticos, representada na Figura 77, que é despoletada tipicamente após a atualização automática e periódica dos servidores e *workstation*, mas também pode ser executada através de um pedido do utilizador. A *pipeline* de testes automáticos existente encontra-se em parte representada na Figura 77, uma vez que neste momento existem 20 *jobs* de testes automáticos pelo que não seria legível caso estivessem todos presentes na figura.

Figura 77 Representação de parte da *pipeline* dos testes automáticos.

Os *jobs* são executados em sequência/cascata, em que cada um destes corresponde a um teste automático. O objetivo do projeto é a integração do *job* de testes automáticos de localização de defeitos nesta *pipeline* para que também este seja executado de forma automática e periódica e, no fim, serem validados os resultados. Dependendo da execução e dos resultados obtidos, cada *job* é devidamente sinalizado, seguindo as descrições presentes na Figura 78.

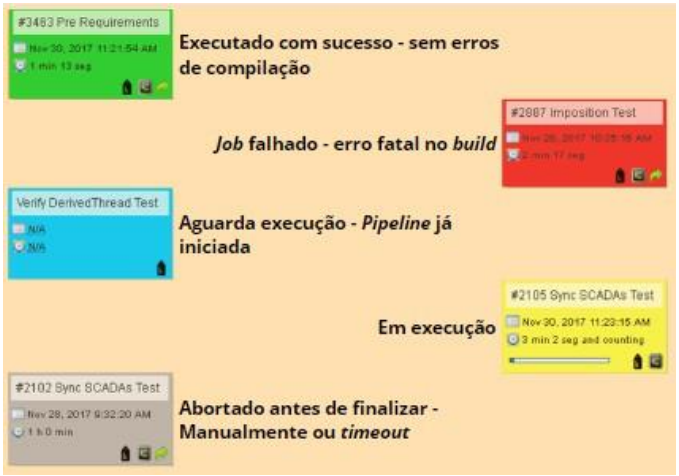


Figura 78 Significado das cores de cada *job* da *pipeline*.

5.2. *JOB* DE LOCALIZAÇÃO DE DEFEITOS

A criação do *job* de testes automáticos de localização de defeitos foi um processo relativamente fácil por já existirem outros *jobs* na *pipeline*. Dessa forma, é possível fazer a criação do novo *job* a partir de um anterior e escolher qual o local onde se quer colocar os novos testes automáticos. No *build* de um teste, de forma resumida e superficial, é necessário desenvolver um código que transmita a informação relativamente ao local onde se encontra o ficheiro executável do teste e local onde serão escritos os resultados. Tendo em conta que, como foi referido anteriormente, o *job* da localização de defeitos foi criado a partir de um outro *job*, este código apenas necessitou de ser adaptado para o ficheiro executável e para o

ficheiro de resultados relativamente à localização de defeitos, pelo que não foi efetuado desenvolvimento de código, mas sim adaptação do mesmo. As adaptações efetuadas foram as mencionadas e estão sublinhadas na Figura 79 que representa a consola do *build* onde os testes de localização de defeitos são chamados a executar. Nessa mesma figura, estão sublinhados dois ficheiros denominados *"faultlocation.tap"* cujo objetivo do mesmo será abordado e devidamente explicado mais à frente.

Figura 79 Comando de execução do *build* dos testes automáticos.

O *job* da localização de defeitos foi colocado após o *job* "*Alarms Test"* – que testa a geração de alarmes quando se altera o estado em entidades específicas – e antes do "*Generate Outages Test*" – onde se realiza um disparo num equipamento de estado bem conhecido, e valida a criação da *tag* de *switch* e a geração do *outage* (interrupção) associado à mesma. Para que fosse reconhecido pelo sistema o sucesso ou insucesso de cada cenário de defeito e, por consequência, do *job* na sua totalidade, foram colocadas duas funções em partes críticas dos *scripts*: `mark_as_ok()` e `mark_as_failed()`. A primeira marca o sucesso de um determinado *script* e a segunda tem a função contrária de assinalar um erro e marcar a falha do *script* numa determinada parte do código. Posto isto, na Tabela 11 e na Tabela 12 estão representados os locais onde foram inseridas as funções que assinalam insucesso e sucesso, respetivamente. Cada uma destas funções foi inserida de modo a identificar cada *script* como bem-sucedido ou falhado.

Tabela 11 Locais de utilização na função mark_as_failed().

Função	*Script*	Onde foi colocada a função
`mark_as_failed()`	Pré-Requisitos	Quando não é possível alterar o estado do disjuntor-terra
		Quando a linha está no estado *Dead*
		Quando a linha está no estado *Faulted*
		Quando a linha está no estado *Earthed*
	Validações	Quando não é possível estabelecer ligação à base de dados
		Quando não é encontrado nenhum ID de falha
		Quando não são obtidos equipamentos a partir da *query*
		Quando as distâncias obtidas estão erradas
		Quando os equipamentos obtidos não são os esperados
		Quando a quantidade de equipamentos obtidos é diferente da quantidade de equipamentos esperados

Tabela 12 Locais de utilização na função mark_as_ok().

Função	*Script*	Onde foi colocada a função
`mark_as_ok()`	Pré-Requisitos	Quando a linha se encontra energizada
	Validações	Quando se verifica a mesma quantidade de equipamentos obtidos e esperados
		Quando as distâncias se verificam as esperadas

Caso estas funções não fossem utilizadas, a *pipeline* iria identificar o *job* sempre como bem-sucedido uma vez que, apesar de não ser enviada a informação definindo o sucesso/insucesso, não seria recebido qualquer sinal de erro, logo seria assumido o seu sucesso. Com as funções colocadas nos sítios descritos anteriormente, o *job* foi inserido e testado na *pipeline*, onde correu com sucesso, o que está representado na Figura 80.

Figura 80 *Job* de testes automáticos de localização de defeitos inserido na *pipeline* de testes automáticos.

5.3. *PLUG-IN* DE RESULTADOS *TAP*

O *output* do *job* de localização de defeitos pode ser observado, como demonstrado anteriormente, a partir do *log* criado aquando da sua execução. Ainda que o log agora seja analisado a partir do jenkins a informação nele contida é semelhante à apresentada na Figura 64 e Figura 65. No entanto, o objetivo é tornar a análise desse *log* tão desnecessária quanto possível por ser um processo moroso e difícil de analisar, substituindo-o por uma análise mais gráfica, intuitiva, rápida e simples. Aqui surge o *plug-in* "*TAP*", que significa *Test Anything Protocol*, e cujo objetivo é esse mesmo. Para aplicar o *plug-in* ao *job* em questão, na consola do *build* é definida a escrita de resultados do mesmo no ficheiro *faultlocation.tap*, mencionado anteriormente e sublinhado na Figura 79. De seguida, na configuração do *job* existe um campo denominado "*Post-build Actions*", onde podem ser configurados os resultados *TAP* indicando o caminho para os resultados do *job*. O descrito está representado na Figura 81.

Figura 81 Aplicação do *plug-in TAP* aos testes automáticos de localização de defeitos.

Utilizando este *plug-in*, o resultado obtido a partir dos testes é mais gráfico e garante que a informação necessária para uma primeira análise aos resultados (normalmente não é necessário haver uma segunda análise, apenas em caso de um erro que não seja percetível a partir dos resultados *TAP*) é reportada para ser facilmente analisada sem distrações devido a informação excessiva. Os resultados *TAP* obtidos após execução do *job* estão representados na Figura 82.

	Number	Description
	1	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 1 correct validations
	2	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 2 correct validations
	3	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 3 correct validations
	4	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 4 correct validations
	5	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 5 correct validations
	6	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 6 correct validations
	7	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 7 correct validations
	8	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 8 correct validations
	9	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 9 correct validations
	10	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 10 correct validations
	11	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 11 correct validations
	12	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 12 correct validations
	13	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 13 correct validations
	14	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 14 correct validations
	15	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 15 correct validations
	16	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 16 correct validations
	17	FL2_VALIDATIONSEQUIPMENTS AND DISTANCES | Test 17 correct validations

Figura 82 Representação dos resultados obtidos, utilizando o *plug-in TAP*, após a execução dos testes automáticos de localização de defeitos.

Como se vê na figura, todos os testes tiveram sucesso com as validações, no entanto, caso algum deles tivesse falhado, a cor da coluna da esquerda seria vermelha na linha do teste em questão, como se vê na **Figura 83**.

	9	FL2_VALIDATIONS- EQUIPMENTS AND DISTANCES | Test 9 - FL2_D1200 correct validations
	10	FL2_VALIDATIONS- INCORRECT_DISTANCES | Test 10 - FL2_D670_DF1e2 incorrect validations | Distâncias dos equipamentos encontradas não estão corretas.
	11	FL2_VALIDATIONS- INCORRECT_DISTANCES | Test 11 - FL2_D670_DF1e5 incorrect validations | Distâncias dos equipamentos encontradas não estão corretas.
	12	FL2_VALIDATIONS- EQUIPMENTS AND DISTANCES | Test 12 - FL2_D800_DF1e2 correct validations

Figura 83 Representação do resultado de testes de localização de defeitos, recorrendo ao *plug-in TAP*, executados com e sem erros.

Desta forma, a análise dos resultados torna-se mais simples, eficaz e intuitiva, diminuindo o tempo e a quantidade de informação a ser filtrada aquando da análise.

6. CONCLUSÕES

6.1. ANÁLISE CONCLUSIVA

O presente trabalho foi realizado com o intuito de testar de forma automática a ferramenta de localização de defeitos desenvolvida pela Efacec. O desenvolvimento de testes automáticos que visam garantir o correto funcionamento dessa ferramenta reveste-se de particular importância na medida em que esta é a única aplicação e o único auxílio que o operador da rede tem na procura pela localização de uma falha nas linhas de distribuição. Por isto, o seu correto funcionamento deve ser garantido evitando que um possível erro na mesma resulte num efeito contrário ao pretendido, que é a redução do tempo de interrupção do serviço, e o aumento da qualidade do tempo de resposta após defeitos, resultando também numa qualidade superior do serviço.

A realização deste projeto exigiu uma constante superação de conhecimentos, aptidões e obstáculos, para que fosse possível atingir todos os objetivos definidos e implementar melhorias e caraterísticas não previstas, mas que acrescentam valor intrínseco ao bloco de testes desenvolvido. Como dito, os objetivos deste trabalho foram atingidos na sua plenitude, tendo sido possível ainda implementar melhorias significativas como foi o caso da utilização do *TAP plug-in* para obtenção de um *output* mais visual e intuitivo, e o desenvolvimento de um *script* de pré-requisitos da rede para conhecer o estado da mesma antes da execução dos testes automáticos.

O facto de o projeto proposto ser de elevada importância e utilidade para o futuro da empresa, levou a um interesse e exigência pessoal ainda maiores na concretização do mesmo. A vontade de desenvolver um projeto eficiente, útil e que acrescentasse valor ao dia a dia da empresa, e em especial à unidade de negócio do ASE, foi a principal motivação perante as adversidades encontradas ao longo do estágio.

O desenvolvimento do projeto de dissertação numa empresa com o histórico sucesso, valores e anos de experiência da Efacec foi simultaneamente entusiasmante e desafiante, resultando num elevado grau de aprendizagem e desenvolvimento de *soft skills*. A possibilidade de interação diária com os sistemas da Efacec tornou o projeto ainda mais interessante e aprazível, bem como a utilização de linguagens de programação com as quais não existia grande familiaridade (*PL/SQL, javascript,* comandos dbgm) tornou o projeto mais complexo e desafiante. O reconhecimento e o interesse no potencial destes testes automáticos por parte dos colaboradores da empresa foram também um fator motivador e que elevou a responsabilidade no desenvolvimento dos mesmos.

Uma vez que o projeto levou à interação com várias equipas do ASE, houve uma necessidade de adaptação e esforço de todas as partes, promovendo a organização, comunicação e brio ao longo de todo o desenvolvimento.

Para a realização do projeto ser bem-sucedida foi necessário, numa fase inicial, um período de aprendizagem sobre o sistema e sobre a ferramenta de localização de defeitos. Saber como aceder às ferramentas e aplicações do sistema tornou-se quase tão importante como aprender quais os princípios de funcionamento dos IPDs e dos medidores de distância.

A divisão no processo de desenvolvimento dos testes, começando por *scripts* dbgm e evoluindo progressivamente até à solução obtida, foi importante para encarar o desenvolvimento do projeto como algo gradual que simultaneamente permitiu visualizar resultados adaptados à fase em que o projeto se encontrava.

O aparecimento de certas adversidades, como o tempo elevado na execução dos testes, a rede encontrar-se constantemente no estado *faulted*, o *datarequest* não ser reconhecido devido à falta do tópico *Dynamics*, e a sobrecarga de informação no *derivedthread* pela inserção desse mesmo tópico, foram imprevistos de diferentes complexidades mas que exigiram a combinação de caraterísticas como astucia e capacidade de superação com

aprendizagem e conhecimento técnico, de modo a que fossem desenvolvidas soluções para cada um desses obstáculos.

Atualmente os testes automáticos desenvolvidos são executados diariamente, 4 vezes por dia, cumprindo a intenção de testar e monitorizar o funcionamento da ferramenta de localização de defeitos.

Em suma, o trabalho realizado vai ao encontro da proposta e expectativas formuladas pela Efacec, resultando numa contínua testagem da ferramenta de localização de defeitos de modo a garantir o seu correto funcionamento ou detetar erros na mesma tão rápido quanto possível, alertando para a necessidade da sua correção.

6.2. PERSPETIVAS FUTURAS

Esta secção da dissertação aparece em jeito de concordância com o termo japonês *kaizen*, que significa "continua a melhorar" com o intuito de transmitir a ideia de melhoria contínua [56] e de que tudo, assim como o projeto desenvolvido, tem margem de evolução. Posto isto, seguem-se possíveis trabalhos futuros que não foram implementados, mas que podem dar seguimento à evolução dos testes automáticos de localização de defeitos.

O primeiro desenvolvimento a fazer no projeto poderia ser a implementação de *debug levels*, ou seja, níveis de quantidade de informação a ser escrita para *log* aquando da execução dos testes automáticos. Neste desenvolvimento, o objetivo seria que a informação disponível nos *logs* após execução dos testes fosse variável consoante o *debug level*, ou seja, quanto maior for o *debug level*, mais informação estará presente no *log*. Desta forma, visto que grande parte das vezes não é necessário consultar os *logs* porque se obtém a informação necessária a partir dos resultados *TAP*, este nível poderia manter-se baixo de modo que a informação ocupasse pouco espaço, mas quando fosse necessário aumentar-se-ia o nível de modo a consultar mais informação.

Outro desenvolvimento a ser feito poderia ser o desenvolvimento de novos testes, nomeadamente utilizando resistência e reatância, para que outras formas de cálculo da zona provável de defeito fossem também testadas, alargando desta forma a cobertura dos testes automáticos de localização de defeitos. Este ponto foi abordado para que pudesse ser desenvolvido ainda no contexto do presente projeto, no entanto, devido a condicionantes que

ultrapassam os envolvidos no projeto (nomeadamente falta de tempo, responsabilidades profissionais, etc) não foi possível o seu desenvolvimento. Nesse sentido, este tópico foi abordado no capítulo 3.8 onde estão presentes os valores de resistência e reatância de cada linha representados na Figura 51 e na Tabela 7.

Referências Bibliográficas

[1] M. K. R. M. Amit Shlomo, "Temporal pattern-based malicious activity detection in SCADA systems," *Elsevier,* vol. 102, p. 17, 2020.

[2] M. e. E. E. S. Efacec Energia, "ScateX + Network Manager," [Online]. Available: http://www.efacec.pt/en/wp-content/uploads/2016/10/CS256I1502B1.pdf. [Acedido em 15 July 2022].

[3] A. Moreira, I. Franchin, J. Pereira e M. Matos, "Fault Location Finder Specification," INESC PORTO, Efacec Engenharia, Porto, 2012.

[4] IEC, "International Electrotechnical Vocabulary, IEV Ref 448-13-02: Power System Protection," International Electrotechnical Comission , 12 1995. [Online]. Available: https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=448-13-02. [Acedido em 14 08 2022].

[5] P. M. C. Torres, "Metodologias de resolução de congestionamentos : rede ibérica de transporte de electricidade," Instituto Superior Técnico, Lisboa, 2004.

[6] "Netz Entwicklungs Plan," Netz Entwicklungs Plan, 2019. [Online]. Available: https://www.netzentwicklungsplan.de/en/node/621. [Acedido em 26 07 2022].

[7] C. C. J. L. A. Júlio S. Martins, "Qualidade de energia elétrica," em *3º Congresso Luso-Moçanbicano de Engenharia – CLME'2003*, Maputo, Moçambique, 2003.

[8] C. Puret, "MV public distribution networks throughout the world," *Schneider Electric, Cahier Technique,* vol. 155, p. 28, 1992.

[9] T. Gonen, Electrical Power Transmission System: Analysis and Design, California, Sacramento, USA: CRC Press, 2014.

[10] IEC, “International Electrotechnical Vocabulary, IEV Ref 448-13-07: Power System Protection,” International Electrotechnical Comission, [Online]. Available: https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=448-13-07. [Acedido em 29 Janeiro 2022].

[11] IEC, “International Electrotechnical Vocabulary, IEV Ref 448-13-06: Power System Protection,” International Electrotechnical Comission, [Online]. Available: https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=448-13-06. [Acedido em 29 Janeiro 2022].

[12] IEC, “International Electrotechnical Vocabulary, IEV Ref 448-13-05: Power System Protection,” International Electrotechnical Comission, [Online]. Available: https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=448-13-05. [Acedido em 28 Janeiro 2022].

[13] M. Bolotinha, Distribuição de Energia Elétrica em Média e Baixa Tensão, Lisboa: engebook, 2019.

[14] H. M. H. I. S. S. Gururajapathy, “Fault location and detection techniques in power distribution systems with distributed generation: A review,” *Renewable and Sustainable Energy Reviews,* p. 10, 2017.

[15] IEC, “International Electrotechnical Vocabulary, IEV Ref 448-13-12: Power System Protection,” International Electrotechnical Comission, [Online]. Available: https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=448-13-12. [Acedido em 29 Janeiro 2022].

[16] D. M. G. J. C. W. P. A. S. Michael D. Fontaine, “IEEE / ESW Applying Reliability Centered Maintenance (RCM) to Electrical Equipment Critical to Worker Safety,” em *IEEE IAS Electrical Safety Workshop*, Dallas, TX, USA, 2013.

[17] D. Prajapat, “Faults and Effects in Electrical Power System,” Madhav University.

[18] IEC, “International Electrotechnical Vocabulary, IEV Ref 121-12-76: Power System Protection,” International Electrotechnical Comission, [Online]. Available: https://www.electropedia.org/iev/iev.nsf/display?openform&ievref=121-12-76. [Acedido em 30 01 2022].

[19] L. Gauffin, “Design of personlly safe 1 - 72.5kV switchgear rooms: Physical characteristicas and design aspects,” ABB Distribution, Västerås, 1988.

[20] T. J. D. J. Lewis Blackburn, Protective Relaying: Principles and Applications, CRC Press; 4th edition, 2014.

[21] H. F. W. R. K. A. A. T. J. Y. Tang, “Fault indicators in transmission and distribution systems,” em *International Conference on Electric Utility Deregulation and Restructuring and Power Technologies*, London, UK, 2000.

[22] J. P. S. D. R. G. Baker, “Performance of on-line fault distance monitor for distribution cable circuits,” em *IEEE/PES Transmission and Distribution Conference and Exposition. Developing New Perspectives*, Atlanta, GA, USA, 2001.

[23] T. Welfonder, “Localisation de défauts monophasés dans les réseaux de distribution à neutre compensé,” Thèse de Doctorat de l’Institut National Polytechnique de Grenoble, 2013.

[24] B. R. D. P. M. F. B. S. N. H. Christophe Andrieu, “Fault detection, analysis and diagnostics in high-DG distribution systems,” CRISP, 2004.

[25] D. J. Krajnak, “Faulted circuit indicators and system reliability,” em *Rural Electric Power Conference*, 2000.

[26] M. A. A. Md Shafiullah, “A Review on Distribution Grid Fault Location Techniques,” *Electric Power Components and Systems,* p. 19, 2017.

[27] L. K. K. K. Amir Farughian, “Review of methodologies for earth fault indication and location in compensated and unearthed MV distribution networks,” *Electric Power Systems Research*, p. 8, 2018.

[28] IEEE, “IEEE Guide for Determining Fault Location on AC Transmission and Distribution Lines,” em *IEEE Std C37.114-2014 (Revision of IEEE Std C37.114-2004)*, vol. 10, New York, IEEE, 2015, pp. 1-76.

[29] Y. Y. M. Y. R. K. T. M. T. Takagi, “Development of a New Type Fault Locator Using the One-Terminal Voltage and Current Data,” *IEEE Transactions on Power Apparatus and Systems*, Vols. %1 de %2PAS-101, nº 8, pp. 2892 - 2898, 1982.

[30] C. M. F. D. L. L. A. A. Girgis, “A fault location technique for rural distribution feeders,” *IEEE Transactions on Industry Applications*, vol. 29, nº 6, pp. 1170-1175, 1993.

[31] D. C. K. Zimmerman, “Impedance-based fault location experience,” *58th Annual Conference for Protective Relay Engineers*, pp. 211-226, 2005.

[32] B. Fleming, “Elemento direcional de impedância de seqüência-negativa,” em *10th Annual ProTest User Group Meeting*, Pasadena, California, 1998.

[33] J. J. I. E. R. Murari Mohan Saha, Fault Location on Power Networks,1st edition, Springer Publishing Company, Incorporated, 2009.

[34] J. B. R. G. B. D. A. Tziouvaras, “New multi-ended fault location design for two- or three-terminal lines,” em *2001 Seventh International Conference on Developments in Power System Protection (IEE)*, Amsterdam, Netherlands, 2001.

[35] A. Guzmán, B. Kasztenny, Y. Tong e M. V. Mynam, “Accurate and economical traveling-wave fault locating without communications,” em *2018 71st Annual*

Conference for Protective Relay Engineers (CPRE), College Station, Texas, USA, 2018.

[36] B. Datta e S. Chatterjee, "A literature review on use of Bewley's lattice diagram," em *2012 1st International Conference on Power and Energy in NERIST (ICPEN)*, Nirjuli, India, 2012.

[37] B. Kasztenny, A. Guzmán, M. V. Mynam e T. Joshi, "Locating faults before the breaker opens - Adaptive autoreclosing based on the location of the fault," em *2018 71st Annual Conference for Protective Relay Engineers (CPRE)*, College Station, Texas, USA, 2018.

[38] A. M. G. Godinho, "Localização de Defeitos em Linhas Aéreas de Transporte e Distribuição de Energia," Faculdade de Ciências e Tecnologia, Lisboa, 2019.

[39] J. Chang, "Single Ended Traveling Wave Based Fault Location Using Discrete Wavelet Transform | Theses and Dissertations - Electrical and Computer Engineering," University of Kentucky, Kentucky, 2014.

[40] S. J. A. E. E. B. A. Bahmanyar, "A comparison framework for distribution system outage and fault location methods," *Electric Power Systems Research*, p. 16, 2017.

[41] A. Cichocki e R. Unbehauen, Neural Networks for Optimization and Signal Processing, Nova Iorque: John Wiley & Sons, 1993.

[42] H. Bon-gang, "Chapter 3 - Methodology," em *Performance and Improvement of Green Construction Projects*, Management Strategies and Innovations, 2018, pp. 15-22.

[43] I. N. d. Silva, D. H. Spatti e R. A. Flauzino, Redes Neurais Artificiais: para engenharia e ciências aplicadas, São Paulo: Artliber Editora, 2010.

[44] “FreeCodeCamp,” 10 Janeiro 2020. [Online]. Available: https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/. [Acedido em 05 Setembro 2022].

[45] E. Doherty, “Educative,” 15 Abril 2020. [Online]. Available: https://www.educative.io/blog/object-oriented-programming. [Acedido em 05 Setembro 2022].

[46] Oracle, “Oracle,” Oracle, [Online]. Available: https://www.oracle.com/pt/database/technologies/appdev/plsql.html. [Acedido em 07 09 2022].

[47] V. Martins, “PL/SQL: saiba como funciona e quando usar!,” Be Trybe, 18 08 2022. [Online]. Available: https://blog.betrybe.com/linguagem-de-programacao/pl-sql-tudo-sobre/. [Acedido em 06 09 2022].

[48] “mysql_query,” php, [Online]. Available: https://www.php.net/manual/en/function.mysql-query.php. [Acedido em 10 09 2022].

[49] “Iterators and generators,” mdn web docs _, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Iterators_and_Generators. [Acedido em 10 09 2022].

[50] “Iteration protocols,” mdn web docs_, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Iteration_protocols#the_iterator_protocol. [Acedido em 10 09 2022].

[51] T. E. o. E. Britannica, “markup language,” Encyclopedia Britannica, 21 Julho 2021. [Online]. Available: https://www.britannica.com/technology/markup-language. [Acedido em 10 09 2022].

[52] S. J. Bigelow, “configuration file,” TechTarget , Setembro 2018. [Online]. Available: https://www.techtarget.com/searchitoperations/definition/configuration-file. [Acedido em 12 09 2022].

[53] Jenkins, “Glossary,” Jenkins, [Online]. Available: https://www.jenkins.io/doc/book/glossary/. [Acedido em 12 09 2022].

[54] I. Chubb, “Jenkins Terms Defined and Demystified,” Inedo, 8 Março 221. [Online]. Available: https://blog.inedo.com/jenkins/definitions. [Acedido em 12 09 2022].

[55] R. Hat, “CI/CD: integração e entrega contínuas,” Red Hat, 12 Abril 2018. [Online]. Available: https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd. [Acedido em 12 09 2022].

[56] K. Institute, “What is Kaizen,” Kaizen Institute, [Online]. Available: https://www.kaizen.com/what-is-kaizen.htm. [Acedido em 13 09 2022].