



Universidade do Minho
Escola de Engenharia

Adriano Eduardo Magalhães Oliveira

Evolução de um sistema de gestão de alarmes

Evolução de um sistema de gestão de alarmes

Adriano Eduardo Magalhães Oliveira

UMinho | 2023

Outubro de 2023



Universidade do Minho
Escola de Engenharia

Adriano Eduardo Magalhães Oliveira

Evolução de um sistema de gestão de alarmes

Relatório de Trabalho de Dissertação de Mestrado Integrado
em Engenharia e Gestão de Sistemas de Informação

Trabalho efetuado sob a orientação de
Professor Filipe Miguel Lopes Meneses

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos. Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



**Atribuição-NãoComercial-Compartilhalgual
CC BY-NC-SA**

<https://creativecommons.org/licenses/by-nc-nd/4.0/>

AGRADECIMENTOS

Realizar esta dissertação foi uma experiência desafiadora e divertida para mim. Apesar das dificuldades e obstáculos que encontrei adorei os momentos de diversão e de sucesso que compensaram completamente todo o esforço.

Gostava de começar por agradecer ao meu orientador Professor Filipe Meneses que disponibilizou tempo e esforço para analisar, criticar de forma construtiva e ajudar a desenvolver esta dissertação.

Gostava também de agradecer aos meus colegas do estágio que me receberam da melhor forma, proporcionaram apoio e conhecimento e mantiveram sempre uma opinião crítica em relação ao meu trabalho. Um agradecimento especial ao meu supervisor Bruno Rebelo que se mostrou sempre disponível para me ajudar e ao meu amigo Marco Rosa que me orientou em questões de melhores práticas de desenvolvimento e proporcionou excelentes momentos de alegria.

Agradeço muito aos meus pais por todo o apoio que me deram, eles que muitas vezes tiveram de abdicar de bens materiais e de tempo livre para que conseguisse chegar onde cheguei, estarei eternamente grato pelo vosso esforço e continuarei a aproveitar toda a motivação que me deram.

Agradeço também à Flávia por ter superado este desafio comigo e por me ter ajudado a distrair mesmo quando era mais difícil.

Por último agradeço a todos os meus amigos, muito obrigado por contribuírem para a minha felicidade e por me ajudarem a evoluir, tentarei sempre contribuir para que também vocês sejam motivados nesse sentido.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio, nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Evolução de um sistema de gestão de alarmes

A evolução das redes de telecomunicações, nomeadamente o seu crescimento e o aumento da sua complexidade tem representado um grande desafio para as empresas deste setor. Gerir estas redes para garantir a sua disponibilidade e a qualidade dos serviços torna-se cada vez mais difícil. Devido à sua complexidade a gestão das redes é normalmente dividida por diferentes empresas que desempenham funções complementares e cujos sistemas necessitam de comunicar para assegurar esta gestão. Assim, a interoperabilidade entre os sistemas assume um papel crucial neste domínio.

Nesta dissertação foi realizada uma prova de conceito onde se implementou uma *Open API* (especificação de comunicação) da *TM Forum* num sistema de gestão de alarmes de uma fornecedora de serviços de telecomunicações. Esta prova de conceito foi realizada com o objetivo de demonstrar a utilidade da adoção de especificações e de normas para facilitar a compreensão da informação entre os diferentes sistemas e tornar a sua integração num processo mais simples e rápido.

Com base na prova de conceito foi desenvolvida uma método que tem como objetivo simplificar a implementação das *Open APIs* da *TM Forum*, incentivando as fornecedoras de serviços de telecomunicações a adotar especificações de comunicação de uma forma global e a melhorar a eficiência da gestão de redes através da colaboração.

Por último, são apresentadas conclusões em relação à utilidade da *API* implementada para a fornecedora de serviços de telecomunicações, destacando a sua capacidade de proporcionar uma alternativa simples e eficaz de integração do seu sistema. Além disso são identificadas perspetivas de trabalho futuro relativo à necessidade de evolução das especificações de comunicação e à necessidade de continuar a promover a interoperabilidade entre sistemas de gestão de redes.

Palavras-chave: gestão de rede; gestão de falhas; interoperabilidade; telecomunicações.

ABSTRACT

Evolution of an alarm management system

Telecommunication networks' evolution, especially their growth and increasing complexity, has presented a significant challenge for the companies in this sector. Managing these networks to ensure availability and service quality has become increasingly difficult. Due to this process complexity, it's typically distributed among different companies' systems with complementary roles. Therefore, interoperability between systems plays a crucial role in this domain.

This dissertation presents a proof of concept involving the implementation of an Open API (a communication specification) from the *TM Forum* into a telecommunications services provider's alarm management system. This proof of concept aims to underscore the importance of adopting specifications and standards, making information exchange between different systems more comprehensible and streamlining the integration process for increased efficiency.

Derived from the proof of concept, a methodology has been created to assist other telecommunications services providers in implementing *TM Forum's* Open APIs. This methodology encourages them to embrace communication specifications globally and enhance network management efficiency through collaborative efforts.

Finally, conclusions regarding the practicality of the developed solution for the telecommunications services supplier are presented, along with future perspectives on the evolution of communication specifications and the continued necessity to promote interoperability among network management systems.

Keywords: fault management; interoperability; network management; telecommunications.

ÍNDICE

1.	Introdução	1
1.1	Contexto e motivação do trabalho	1
1.2	Objetivos	2
1.3	Metodologia e abordagem iterativa	3
1.4	Estrutura	5
2.	Gestão de redes.....	6
2.1	Gestão de falhas e de alarmes	7
2.2	Sistema de gestão de alarmes da prova de conceito.....	9
2.2.1	Arquitetura e funcionamento.....	9
2.2.2	Aplicação	10
2.2.3	API de gestão de alarmes	12
2.3	Programa de <i>Open APIs</i> da <i>TM Forum</i>	13
2.4	Outras alternativas à solução proposta	15
3.	Planeamento da implementação	18
3.1	Verificação de compatibilidade	18
3.1.1	Recolha dos recursos de apoio à implementação e análise inicial.....	18
3.1.2	Levantamento de pedidos e funcionalidades obrigatórias de implementar.....	20
3.1.3	Comparação de pedidos e mapeamento de atributos	20
3.1.4	Conclusões acerca da compatibilidade.....	21
3.2	Lista de ferramentas	21
4.	Implementação da <i>Open API</i> no sistema	24
4.1	Criação de um novo serviço no sistema	24
4.2	Análise e configuração inicial dos testes de conformidade automáticos (<i>CTK</i>)	25
4.3	Desenvolvimento do pedido <i>GET</i> – Consulta de alarmes.....	28

4.4	Desenvolvimento do pedido <i>GET</i> – Consulta de alarme pelo <i>id</i>	32
4.5	Implementação da funcionalidade de seleção	34
4.6	Implementação da funcionalidade de filtragem	38
4.7	Desenvolvimento do pedido <i>POST</i> – Criação de alarme	42
4.8	Desenvolvimento do pedido <i>PATCH</i> – Alteração de um alarme	47
4.9	Implementação de valores enumerados	53
4.10	Validação final da <i>API</i> através dos testes de conformidade automáticos (<i>CTK</i>)	59
5.	Adoção da <i>Open API</i>	61
5.1	Apresentação da <i>API</i> desenvolvida	61
5.1.1	Pedidos disponibilizados pela <i>API</i>	62
5.1.2	Autenticação necessária para a execução de pedidos	63
5.1.3	Pedidos <i>GET</i> – Consulta de alarmes e consulta de alarme pelo <i>id</i>	63
5.1.4	Pedido <i>POST</i> – Criação de alarme	66
5.1.5	Pedido <i>PATCH</i> – Alteração de alarme	68
5.2	Testes de desempenho da <i>API</i>	70
5.2.1	Teste de desempenho da <i>API</i> desenvolvida (<i>TMF642</i>)	71
5.2.2	Teste comparativo entre a <i>API</i> desenvolvida e a <i>API</i> existente no sistema	73
5.2.3	Conclusões acerca do desempenho	75
5.3	Requerimento e obtenção do certificado de implementação	75
6.	Método de implementação	77
7.	Conclusões	81
	Referências	83
	Apêndice I: Comparação de respostas do pedido de consulta de alarme	87
	Apêndice II: Mapeamento de atributos entre o sistema e a <i>API</i>	90
	Apêndice III: Problemas encontrados no mapeamento de valores dos atributos enumerados	97

Apêndice IV: Mapeamento de valores dos atributos enumerados	100
Apêndice V: Resultados finais dos testes automáticos (<i>CTK</i>).....	109
Resultados <i>CTK</i> do pedido <i>GET</i> – Consulta de alarmes	109
Resultados <i>CTK</i> do pedido <i>GET</i> – Consulta de alarme pelo <i>id</i>	110
Resultados <i>CTK</i> da funcionalidade de seleção	111
Resultados <i>CTK</i> da funcionalidade de filtragem	114
Resultados <i>CTK</i> do pedido <i>POST</i> – Criação de alarme.....	119
Resultados <i>CTK</i> do pedido <i>PATCH</i> – Alteração de alarme.....	120

LISTA DE ABREVIATURAS E SIGLAS

<i>API</i>	<i>Application Programming Interface</i>
<i>CRUD</i>	Operações de criar, consultar, alterar e eliminar (<i>Create, Read, Update, Delete</i>)
<i>CTK</i>	<i>Conformance toolkit</i>
<i>gRPC</i>	<i>Google Remote Procedure Call</i>
<i>HTTP</i>	<i>Hypertext Transfer Protocol</i>
<i>JDBC</i>	<i>Java Database Connectivity</i>
<i>KB</i>	<i>Kilobytes</i>
<i>NGMN</i>	<i>Next Generation Mobile Networks Alliance</i>
<i>OSS</i>	<i>Operation Support System</i>
<i>REST</i>	<i>Representational state transfer</i>
<i>RPC</i>	<i>Remote Procedure Call</i>
<i>SQL</i>	<i>Structured Query Language</i>
<i>TM Forum</i>	<i>TeleManagement Forum</i>
<i>TMF642</i>	<i>API de gestão de alarmes da TM Forum</i>

LISTA DE FIGURAS

Figura 1: Metodologia <i>Design Science Research</i> (adaptado de Peffers et al., 2014)	3
Figura 2: Ciclo de desenvolvimento da <i>Scrum Framework</i> (Scrumorg, 2020)	4
Figura 3: Arquitetura do sistema de gestão de alarmes.	10
Figura 4: Janela de alarmes ativos do sistema.	10
Figura 5: Filtragem de alarmes no sistema.....	11
Figura 6: Seleção de alarmes no sistema.	11
Figura 7: Lista de ações sobre alarmes do sistema.	12
Figura 8: Arquitetura do sistema com a implementação da <i>TMF642</i>	15
Figura 9: Tabela de recursos da <i>TMF642</i> (TM Forum, 2023).....	19
Figura 10: Criação do módulo da <i>TMF642</i>	25
Figura 11: Estrutura da <i>CTK</i>	26
Figura 12: Configuração do <i>CTK</i> para o pedido <i>GET</i> – Consulta de alarmes.....	27
Figura 13: Resultado inicial da <i>CTK</i>	27
Figura 14: Análise inicial dos resultados esperados no pedido <i>GET</i> – Consulta de alarmes.	29
Figura 15: Diagrama de classes e de fluxo de informação do pedido <i>GET</i> – Consulta de alarmes.....	30
Figura 16: Comparação entre as respostas do pedido <i>GET</i> – Consulta de alarmes.	31
Figura 17: Diagrama de classes e de fluxo de informação do pedido <i>GET</i> – Consulta de alarme pelo <i>id</i>	32
Figura 18: Comparação entre as respostas do pedido <i>GET</i> – Consulta de alarme pelo seu <i>id</i>	34
Figura 19: Identificação de atributos obrigatórios para a funcionalidade de seleção.....	35
Figura 20: Diagrama de classes e de fluxo de informação do pedido <i>GET</i> – Consulta de alarmes com a funcionalidade de seleção de atributos.....	36
Figura 21: Teste manual do pedido <i>GET</i> – Consulta de alarmes com a funcionalidade de seleção de atributos (<i>select fields</i>).....	37
Figura 22: Identificação de atributos obrigatórios para a funcionalidade de filtragem.....	38
Figura 23: Diagrama de classes e de fluxo de informação do pedido <i>GET</i> – Consulta de alarmes com a funcionalidade de filtrar atributos.	39
Figura 24: Teste manual do pedido <i>GET</i> – Consulta de alarmes com a funcionalidade de filtragem de atributos (<i>fields filter</i>).....	41

Figura 25: (Continuação) Teste manual do pedido <i>GET</i> – Consulta de alarmes com a funcionalidade de filtragem de atributos (<i>fields filter</i>).....	41
Figura 26: Diagrama de classes e de fluxo de informação do pedido <i>POST</i> – Criação de alarme.....	44
Figura 27: Comparação entre as respostas do pedido <i>POST</i> – Criação de alarme.....	46
Figura 28: Configuração do ficheiro <i>config.json</i> para teste automático do pedido <i>POST</i> – Criação de alarme.	47
Figura 29: Identificação de atributos obrigatórios do pedido <i>PATCH</i> – Alteração de alarme.	48
Figura 30: Distinção entre a criação e alteração de alarmes no sistema	49
Figura 31: Diagrama de classes e de fluxo de informação do pedido <i>PATCH</i> – Alteração de alarme... ..	50
Figura 32: Comparação entre as respostas do pedido <i>PATCH</i> – Alteração de alarme.....	52
Figura 33: Diagrama de classes e de fluxo de informação do mapeamento de valores enumerados. ..	56
Figura 34: Comparação entre as respostas do pedido <i>POST</i> – Criação de alarme com os valores enumerados alinhados com a <i>TMF642</i>	58
Figura 35: Atualização dos valores enumerados no ficheiro de configurações (<i>config.json</i>) da <i>CTK</i>	59
Figura 36: Resultado global das <i>CTK</i>	60
Figura 37: Lista de pedidos disponibilizados na <i>TMF642</i>	62
Figura 38: Autenticações disponíveis para efetuar pedidos na <i>TMF642</i>	63
Figura 39: <i>GET</i> – Consulta de alarmes – <i>path</i> e parâmetros disponíveis.....	64
Figura 40: <i>GET</i> – Consulta de alarme pelo <i>id</i> – <i>path</i> e parâmetros disponíveis	65
Figura 41: <i>GET</i> – Consulta de alarme(s) – exemplo de resposta de sucesso	65
Figura 42 : <i>POST</i> – Criação de alarme – exemplo de pedido.	66
Figura 43: <i>POST</i> – Criação de alarme – exemplo de resposta de sucesso.	67
Figura 44: <i>PATCH</i> – Alteração de alarme – exemplo de pedido	68
Figura 45: <i>PATCH</i> – Alteração de alarme – exemplo de resposta de sucesso.....	69
Figura 46: Grupos de teste criados no <i>JMeter</i>	71
Figura 47: Configurações dos testes do <i>JMeter</i>	71
Figura 48: Gráfico de tempos de resposta da <i>TMF642</i>	72
Figura 49: Gráfico da evolução dos tempos de resposta ao longo da execução do teste da <i>TMF642</i>	73
Figura 50: Gráfico comparativo dos tempos de resposta.....	74
Figura 51: Gráfico comparativo da evolução dos tempos de resposta ao longo do tempo de execução do teste.....	75
Figura 52: Obtenção do certificado de implementação	76

Figura 53: Método de implementação de <i>Open APIs</i> da <i>TM Forum</i>	77
Figura 54: Resultado <i>CTK</i> – consulta de alarmes	109
Figura 55: Resultado <i>CTK</i> – consulta de alarme pelo <i>id</i>	110
Figura 56: Resultado <i>CTK</i> – seleção do atributo <i>alarmRaisedTime</i>	111
Figura 57: Resultado <i>CTK</i> – seleção do atributo <i>id</i>	111
Figura 58: Resultado <i>CTK</i> – seleção do atributo <i>probableCause</i>	112
Figura 59: Resultado <i>CTK</i> – seleção do atributo <i>sourceSystemId</i>	112
Figura 60: Resultado <i>CTK</i> – seleção do atributo <i>state</i>	113
Figura 61: Resultado <i>CTK</i> – filtragem do atributo <i>alarmRaisedTime</i>	114
Figura 62: Resultado <i>CTK</i> – filtragem do atributo <i>id</i>	115
Figura 63: Resultado <i>CTK</i> – filtragem do atributo <i>probableCause</i>	116
Figura 64: Resultado <i>CTK</i> – filtragem do atributo <i>sourceSystemId</i>	117
Figura 65: Resultado <i>CTK</i> – filtragem do atributo <i>state</i>	118
Figura 66: Resultado <i>CTK</i> – Criação de alarme	119
Figura 67: Resultado <i>CTK</i> – Alteração da <i>probableCause</i> de um alarme	120
Figura 68: Resultado <i>CTK</i> – Alteração da <i>perceivedSeverity</i> de um alarme	120
Figura 69: Resultado <i>CTK</i> – Alteração do <i>alarmType</i> de um alarme.....	120
Figura 70: Resultado <i>CTK</i> – Alteração do <i>state</i> de um alarme	121

LISTA DE TABELAS

Tabela 1: Lista de ferramentas usadas na implementação.....	22
Tabela 2: Descrição do diagrama do pedido <i>GET</i> – Consulta de alarmes.	30
Tabela 3: Descrição do diagrama do pedido <i>GET</i> – Consulta de alarme pelo <i>id</i>	33
Tabela 4: Descrição do diagrama do pedido <i>GET</i> – Consulta de alarme com funcionalidade de seleção de atributos.	36
Tabela 5: Descrição do diagrama do pedido <i>GET</i> – Consulta de alarmes com a funcionalidade de filtragem de atributos.	40
Tabela 6: Mapeamento de atributos obrigatórios para o pedido <i>POST</i> – Criação de alarme.....	43
Tabela 7: Descrição do diagrama do pedido <i>POST</i> – Criação de alarme.	44
Tabela 8: Descrição do diagrama do pedido <i>PATCH</i> – Alteração de alarme.	50
Tabela 9: Identificação de problemas e respetivas soluções para o mapeamento de valores enumerados.	53
Tabela 10: Descrição do diagrama de mapeamento de valores enumerados.....	56
Tabela 11: Resultados do teste de desempenho da <i>TMF642</i>	72
Tabela 12: Resultado do teste de desempenho comparativo.....	74
Tabela 13: Comparação de respostas dos pedidos de consulta de alarme	88
Tabela 14: Mapeamento de atributos entre o sistema e a <i>TMF642</i>	91
Tabela 15: Problemas encontrados no mapeamento de valores dos atributos enumerados.....	98
Tabela 16: Mapeamento de valores dos atributos enumerados.....	101

1. INTRODUÇÃO

A realização desta dissertação resultou de um desafio lançado por uma empresa fornecedora de serviços de telecomunicações que, tendo necessidade de continuar a melhorar os seus produtos e serviços, procura explorar novas oportunidades tecnológicas.

A empresa propõe o estudo de uma solução disponibilizada pela *TM Forum (TeleManagement Forum)* que promove a interoperabilidade entre sistemas, através da adoção de métodos de comunicação normalizados e do alinhamento com normas de caráter global. O seu principal objetivo é evoluir o seu sistema de gestão de alarmes e obter uma certificação que permita atrair novos clientes que procurem implementações mais rápidas e com reduzidos custos.

Neste capítulo são apresentados o [contexto e a motivação associados ao trabalho realizado](#), são identificados os [principais objetivos](#), a [metodologia adotada](#) e é realizada uma [descrição geral da estrutura do documento](#).

1.1 Contexto e motivação do trabalho

As redes de telecomunicações têm uma elevada importância na atualidade. Estas fornecem um mecanismo de comunicação que permite efetuar tarefas comuns e essenciais do dia a dia (Flood, 1997), como enviar e receber mensagens, utilizar serviços de localização, partilhar informação, participar em eventos online, assistir a conteúdos de multimédia, etc. Para as organizações, este mecanismo também é muito significativo, pois grande parte dos seus processos depende da comunicação em rede. Alguns exemplos da sua utilidade são garantir a segurança, monitorizar equipamentos, efetuar transações, emitir certificados, entre muitos outros.

Apesar da facilidade e conveniência com que as redes de telecomunicações são utilizadas, nem sempre corre tudo bem durante a comunicação. Há quantidades enormes de dados a ser transmitidos simultaneamente e naturalmente ocorrem falhas ou anomalias nas transmissões (Douvinet, 2018). As falhas das redes podem ser causadas por uma grande diversidade de fatores, como por exemplo erros no processamento de informação, defeitos no funcionamento de um aparelho, condições adversas que danificam materiais ou até mesmo ataques informáticos. Assim, é essencial que as empresas de telecomunicações possuam soluções para controlar as falhas nas comunicações, evitando que a rede deixe de operar eficientemente ou haja uma perda (indisponibilidade) de serviços (Liang et al., 2019).

Assegurar a operabilidade das redes é uma tarefa extremamente complexa que é geralmente distribuída por diferentes sistemas com funções complementares (Jager et al., 2008). À medida que as redes evoluem e tornam-se mais complexas surge a necessidade de melhorar a eficiência da gestão das mesmas, através da evolução dos sistemas que asseguram esta gestão (Xie et al., 2019). Um dos principais pontos chave para a adaptação à evolução das redes é a interoperabilidade entre os sistemas (Bhat & Alqahtani, 2021).

Nesta dissertação é realizada uma prova de conceito num sistema de gestão de alarmes pertencente a uma fornecedora de serviços de telecomunicações, um sistema que desempenha uma função complementar da gestão de redes (Altice Labs, 2020). A fornecedora identificou a necessidade de evoluir o sistema através da adoção de uma especificação de comunicação para facilitar a sua integração por parte dos seus clientes, que são maioritariamente operadoras de telecomunicações que pretendem monitorizar, resolver e reduzir o impacto das falhas de comunicação nas suas redes. O principal objetivo desta implementação é conseguir reduzir o esforço e o tempo despendidos neste processo.

1.2 Objetivos

O primeiro objetivo é estudar um programa de *Open APIs* disponibilizado pela *TM Forum* e reunir informação geral sobre a sua utilidade para melhorar a interoperabilidade entre sistemas do setor das telecomunicações.

O segundo objetivo é proceder à implementação da *Open API* no sistema, através do desenvolvimento de uma nova *API* que segue as regras da especificação. Esta nova *API* deve permitir aos clientes entender a sua informação mais facilmente e simplificar a sua integração noutros sistemas.

O terceiro objetivo é comprovar que a solução implementada possui os requisitos mínimos para ser utilizada pelos clientes da fornecedora de serviços de telecomunicações.

O quarto objetivo é obter um certificado emitido pela *TM Forum* que comprove a correta implementação da *Open API*.

O último objetivo é, com base na implementação efetuada, desenvolver um método que permita a outras fornecedoras de telecomunicações adotar mais facilmente as *Open APIs* da *TM Forum*, fornecendo conselhos gerais sobre as fases envolvidas e o fluxo de desenvolvimento.

1.3 Metodologia e abordagem iterativa

A metodologia que será utilizada para a investigação e desenvolvimento do projeto é a *Design Science Research (DSR)*. Esta fornece orientações sobre as fases que constituem um projeto de investigação e o seu principal objetivo é assegurar a produção de conhecimento na área das ciências e tecnologias através da criação de novos artefactos (vom Brocke et al., 2020).

A *Design Science Research* permite agilizar o desenvolvimento de um projeto no domínio dos sistemas de informação, resultando na obtenção de uma solução fundamentada e planeada e com a possibilidade de melhoria contínua durante o desenvolvimento.

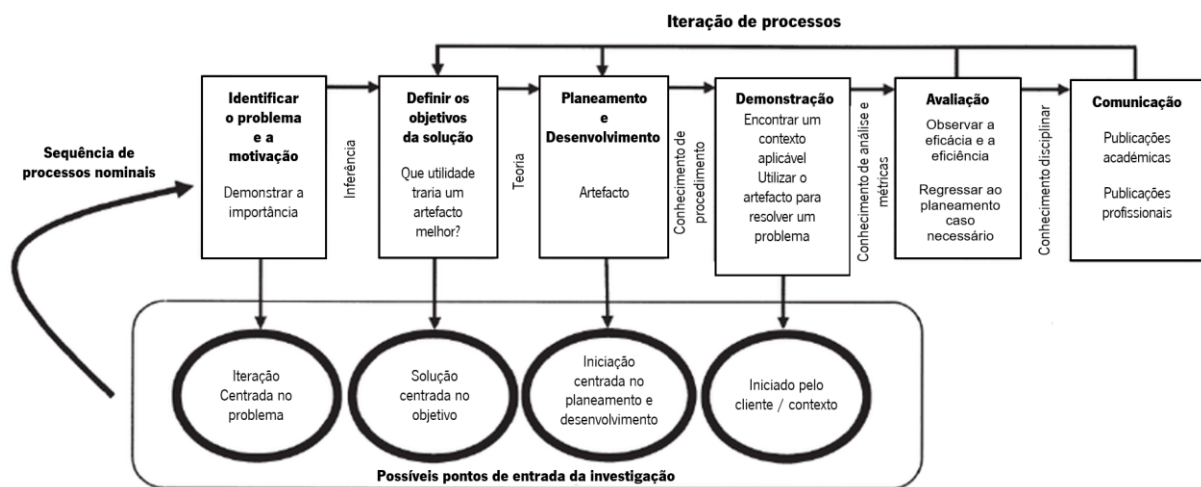


Figura 1: Metodologia *Design Science Research* (adaptado de Peffers et al., 2014)

Na presente dissertação são incluídos os processos da metodologia DSR, ilustrados na Figura 1, da seguinte forma:

As atividades de **identificação do problema, motivação do projeto e definição de objetivos** são apresentadas neste primeiro capítulo e são pormenorizadas no [capítulo 2](#). No segundo capítulo é feito um estado da arte relativo à gestão de redes e à gestão de falhas que serve para contextualizar o problema e apresentar a sua importância. Este consiste numa revisão da literatura sobre estes domínios, onde são apresentadas definições e os contributos tecnológicos para a sua transformação e melhoria gradual. Além disso são expostas limitações existentes e as suas necessidades de evolução.

O processo de **planeamento** é apresentado no [capítulo 3](#) e **desenvolvimento** da solução é documentado no [capítulo 4](#).

A **demonstração** e a **avaliação** são apresentadas no [capítulo 4](#), à medida que é demonstrado o desenvolvimento, e no [capítulo 5](#), onde é apresentada a solução obtida e uma avaliação final da mesma.

Por fim, a **comunicação** é apresentada no [capítulo 7](#), onde são apresentadas as conclusões e resultados do projeto.

Abordagem iterativa para o desenvolvimento do software:

De acordo com Rajib Mall (2018), um projeto de desenvolvimento de *software* pode ser classificado em diferentes classes, mediante as suas características e finalidade. O presente projeto é classificado como desenvolvimento de um serviço de *software*, uma vez que se destina a ser implementado numa base de *software* já desenvolvido e irá acrescentar funcionalidades ao mesmo. Como o novo *software* tem de ser ligado à base existente, este irá seguir uma estrutura idêntica à adotada no sistema atual que se baseia no processamento de alarmes como objetos, ou seja, será utilizada uma abordagem orientada a objetos (*Object-Oriented Software Development*) (Mall, 2018).

Relativamente ao desenvolvimento do software será utilizada a abordagem iterativa Scrum, esta será utilizada uma vez que é utilizada pela restante equipa de desenvolvimento da fornecedora de serviços de telecomunicações. Um diagrama ilustrativo da abordagem é representado na Figura 2. A sua adoção irá permitir dividir a implementação em pequenas partes (*sprints*) que vão sendo validadas ao longo do período de desenvolvimento através de revisões (*sprint review*). Isto permite que o *software* seja concebido de forma gradual e que haja um acompanhamento ativo da sua evolução. Este acompanhamento ativo facilita a deteção de desalinhamentos ou problemas e a sua correção (Schwaber & Sutherland, 2020).

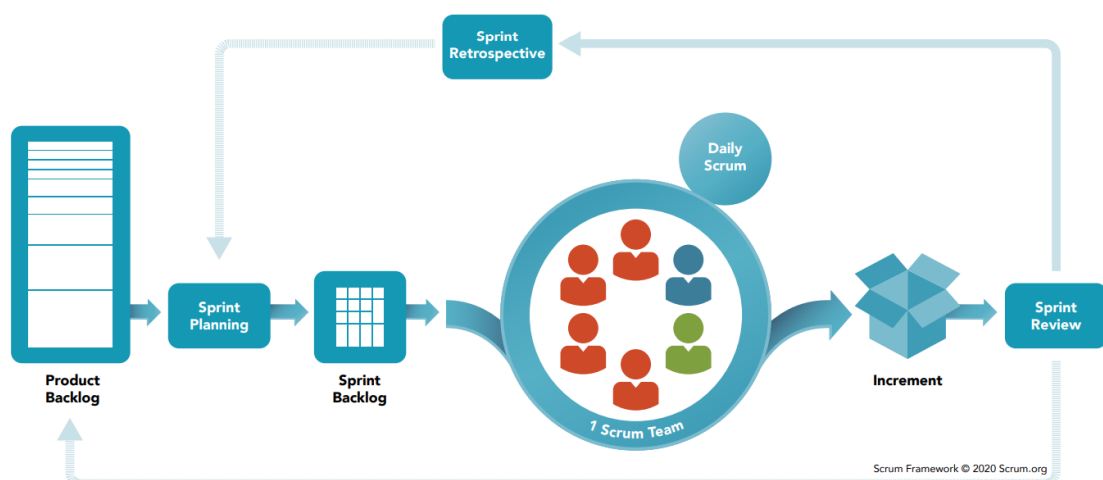


Figura 2: Ciclo de desenvolvimento da *Scrum Framework* (Scrumorg, 2020)

1.4 Estrutura

Este documento está estruturado da seguinte forma:

No [capítulo 2](#) apresenta-se uma revisão de literatura sobre a gestão de redes e da gestão de falhas. Neste capítulo também é descrito o sistema de gestão de alarmes utilizado na prova de conceito e as principais operações de gestão de alarmes disponibilizadas pelo mesmo. Por último é apresentada a *Open API* que será implementada no sistema e possíveis alternativas à mesma.

No [capítulo 3](#) é realizado o planeamento da implementação da solução no sistema de gestão de alarmes. Neste capítulo é inicialmente feita uma verificação de compatibilidade, para se verificar se o sistema possui os recursos necessários para a implementação da *Open API*. Ainda neste capítulo são apresentadas as ferramentas utilizadas para o desenvolvimento da nova solução.

No [capítulo 4](#) é descrito o processo de implementação da *Open API*, nomeadamente, as fases envolvidas no desenvolvimento e os detalhes sobre a forma como a nova solução foi ligada ao sistema existente.

No [capítulo 5](#) é feita uma demonstração da solução obtida, são representados os pedidos e funcionalidades disponibilizadas pela mesma e exemplos de interações. Neste capítulo são também documentados os testes de desempenho efetuados à nova solução para comprovar que a mesma possui os requisitos mínimos estabelecidos pela fornecedora de serviços de telecomunicações. Por último é demonstrada a obtenção do certificado de conformidade da *Open API* emitido pela *TM Forum*.

No [capítulo 6](#) é apresentado um método de implementação das *Open APIs* da *TM Forum* que foi desenvolvido com base na prova de conceito realizada.

Por fim, no [capítulo 7](#), são apresentadas as conclusões sobre a implementação e o trabalho futuro.

2. GESTÃO DE REDES

As redes de telecomunicações têm registado uma evolução notória ao longo dos últimos anos. À medida que se tornam mais complexas, acresce a necessidade de desenvolver novas soluções que permitam facilitar a gestão das mesmas (Gutierrez-Estevez et al., 2019). Para facilitar esta tarefa são utilizados sistemas de suporte à operação que permitem monitorizar e gerir os seus recursos de forma a assegurar a operabilidade das mesmas (Misra, 2004).

Os sistemas de suporte à operação utilizados são geralmente integrados em atividades complementares da gestão de redes (Tawalbeh et al., 2017). De uma forma geral a gestão das redes está dividida em atividades como a gestão de desempenho, configurações, contabilidade, segurança e de falhas (Jager et al., 2008), (Gorod et al., 2007). Estas atividades são distribuídas por diferentes sistemas que necessitam de comunicar entre si para complementar a gestão das redes.

O notável crescimento das redes e o aumento da sua complexidade tem influenciado a evolução destes sistemas, torna-se cada vez mais difícil gerir os seus recursos e existe uma necessidade crescente de reduzir o esforço humano e os custos associados ao mesmo. Para dar resposta a este problema, os sistemas têm evoluído, no sentido de automatizar os processos e aumentar a escalabilidade dos serviços, tornando-se cada vez mais autónomos (Tawalbeh, 2020).

À medida que as redes e as tecnologias evoluem, os sistemas de suporte à operação devem moldar-se a essa evolução, acrescentando novas funcionalidades, sem comprometer os seus utilizadores. Apesar do esforço por implementar múltiplos sistemas de suporte à operação que suportam a gestão das redes, a interoperabilidade entre os mesmos ainda representa uma grande barreira para a eficiência desta gestão. Existe uma necessidade crescente de facilitar a comunicação entre os sistemas e desta forma conseguir simplificar a comunicação entre processos de gestão (Misra, 2004), (Tawalbeh, 2020).

No sentido de endereçar a melhoria da comunicação entre sistemas, a abordagem de “ecossistema” tem aumentado a sua popularidade. Este termo descreve a interdependência e necessidade de evolução comum aos sistemas de gestão de redes. A base desta abordagem é facilitar a interação entre componentes que realizam tarefas interdependentes. Para tal é essencial que sejam entendidas as tarefas específicas de cada uma das componentes e de que forma a sua informação é consumida nas restantes componentes do ecossistema (Yamakami, 2010).

A adoção de normas de comunicação representa uma grande importância para o desenvolvimento destes ecossistemas, pois permite facilitar a perceção da sua informação e comunicação (Gutierrez-

Estevez et al., 2019). Nesta dissertação é apresentada uma iniciativa que tem como objetivo incentivar essa adoção, para melhorar a interoperabilidade entre os sistemas. Mais detalhes sobre a mesma serão apresentados na secção [Programa de Open APIs da TM Forum](#).

2.1 Gestão de falhas e de alarmes

A gestão de falhas é um processo crucial para a deteção de anomalias no funcionamento de uma rede (Sturm et al., 2017). Este processo inclui a deteção, o isolamento e a resolução de problemas através do rastreamento de falhas (Baras et al., 1997). As falhas podem ser definidas como eventos ou mensagens de erro enviadas pelos dispositivos ligados à rede (Asres et al., 2021).

As redes de telecomunicações possuem milhares de aparelhos interligados que estão constantemente a emitir mensagens de erro e de estado. Estas mensagens contêm informações importantes sobre a origem e causas de determinados problemas (Asres et al., 2021).

Os sistemas de gestão de alarmes permitem captar, tratar e armazenar informação sobre as falhas de uma rede. Estas informações são guardadas dentro de alarmes que são notificações sobre a ocorrência de uma falha de comunicação e contêm detalhes descritivos sobre a mesma. Através destes sistemas a informação sobre as falhas é traduzida para uma linguagem perceptível e apresentada de forma centralizada numa interface, facilitando a identificação de problemas e a atuação para resolução dos mesmos (Hajela, 1996), (Salau et al., 2019).

Um exemplo prático de utilidade deste sistema acontece quando um aparelho envolvido na comunicação fica danificado e deixa de funcionar corretamente. Sempre que possível o aparelho emite um alerta que é captado pelo sistema e é apresentado aos utilizadores do sistema, como por exemplo uma operadora de telecomunicações que quer assegurar a operabilidade da sua rede. Através da interface do sistema a operadora identifica a causa da falha e a localização do aparelho danificado e envia uma equipa de manutenção qualificada para resolver o problema.

Uma das grandes complexidades da gestão de falhas é distinguir a importância dos eventos e encontrar a causa dos problemas de comunicação. Os fornecedores de serviços de comunicações necessitam de detetar a causa de uma falha rapidamente, para assegurar o funcionamento correto do serviço que foi comprometido. Esta tarefa de procura pode tornar-se demorada devido à grande quantidade de dados que têm de ser analisados (Baras et al., 1997).

O uso de tecnologias de informação para melhorar a identificação e o reporte de erros e falhas é essencial para as organizações (Salau et al., 2019). Os sistemas têm evoluído no sentido de facilitar as

tarefas de gestão, investindo em funcionalidades que permitem automatizar e reduzir a complexidade da procura pela causa das falhas. Alguns exemplos destas funcionalidades são a filtragem e a correlação de alarmes (Hasan et al., 1999).

A funcionalidade de filtragem facilita a procura de alarmes pois permite visualizar alarmes que ocorreram em determinadas horas, locais, sistemas e muitos outros detalhes. Esta funcionalidade é apresentada na descrição do [Sistema de gestão de alarmes da prova de conceito](#).

A funcionalidade de correlação permite associar alarmes a uma causa através da geração automática de relações. Os alarmes armazenados num sistema de gestão de alarmes são submetidos a um conjunto de verificações e são categorizados em grupos relacionados. Os alarmes relacionados são identificados na interface facilitando a procura pela origem do problema através da pesquisa imediata por alarmes com características idênticas.

Além destas evoluções, também tem sido investigada a aplicação de sistemas de inteligência artificial (*machine learning*) à gestão de alarmes, no sentido de ajudar os operadores a identificar os problemas, através de predições. Por exemplo a aplicação de tecnologias de diagnóstico que permitem monitorizar as falhas, o desempenho e a degradação, fornecendo um diagnóstico preditivo sobre as causas das falhas (Yang et al., 2019).

Outro cenário em que a predição de acontecimentos tem sido investigada é para o processo de emissão de *tickets* de problema (*trouble tickets*). Este processo é procedente à identificação da causa de um problema de comunicação e consiste na emissão de uma mensagem (*ticket*) com a identificação da falha que causou o problema de comunicação e informação útil para a resolução do mesmo (Asres et al., 2021).

Apesar da evolução notória dos sistemas de alarmes, a crescente adesão a tecnologias como o *5G* é previsto um aumento notório do tráfego de dados na rede, da intensificação das comunicações e uma necessidade cada vez maior de criar e melhorar os mecanismos de gestão de redes (Xie et al., 2019).

Ainda existe uma grande limitação relativamente à automação da comunicação entre os sistemas de gestão da rede. Estes necessitam de comunicar entre si devido a cumprirem funções complementares (Jager et al., 2008). Neste sentido existe uma procura pela evolução dos sistemas para melhorar a interoperabilidade dos mesmos e permitir que toda a gestão da rede seja otimizada (Tawalbeh, 2020).

2.2 Sistema de gestão de alarmes da prova de conceito

O sistema de gestão de alarmes utilizado na prova de conceito é composto por:

- uma infraestrutura de captação e armazenamento de falhas de rede;
- uma aplicação *web* de visualização e gestão de alarmes;
- uma *API* de gestão de alarmes (para ligação a outros sistemas).

Para esclarecer melhor a constituição e funcionamento do sistema é inicialmente apresentada a [arquitetura](#), de seguida a [aplicação](#) e por fim a [API](#) e as suas funcionalidades.

2.2.1 Arquitetura e funcionamento

A arquitetura e funcionamento são representados na Figura 3. Os eventos de rede são inicialmente enviados por uma grande diversidade de dispositivos que utilizam diferentes protocolos de comunicação. Estes eventos são recebidos num *gateway* que possui entradas para os diferentes tipos de protocolos. De seguida são encaminhados para um adaptador (*protocol adapter*), responsável por processar e transformar o seu conteúdo num formato compatível com o sistema.

Após serem processados pelo *protocol adapter*, os eventos são redirecionados para dois locais, diretamente para a *API* interna (*JDBC*) e para um filtro que seleciona eventos com características idênticas. Os eventos filtrados são direcionados para uma componente de correlação que agrupa os eventos propícios a estar relacionados e envia a informação sobre as correlações igualmente para a *API* interna.

A *API* interna (*JDBC*) consiste num conjunto de classes e interfaces escritas em Java que envia instruções *SQL* para a base de dados. Esta é responsável por criar os alarmes e inseri-los na base de dados, além disso recebe e adiciona a informação sobre as correlações aos respetivos alarmes.

Por fim a *API* interna também é utilizada para interagir com a interface do utilizador e com a *API* de acesso externo, permitindo efetuar pedidos *CRUD* no armazenamento do sistema.

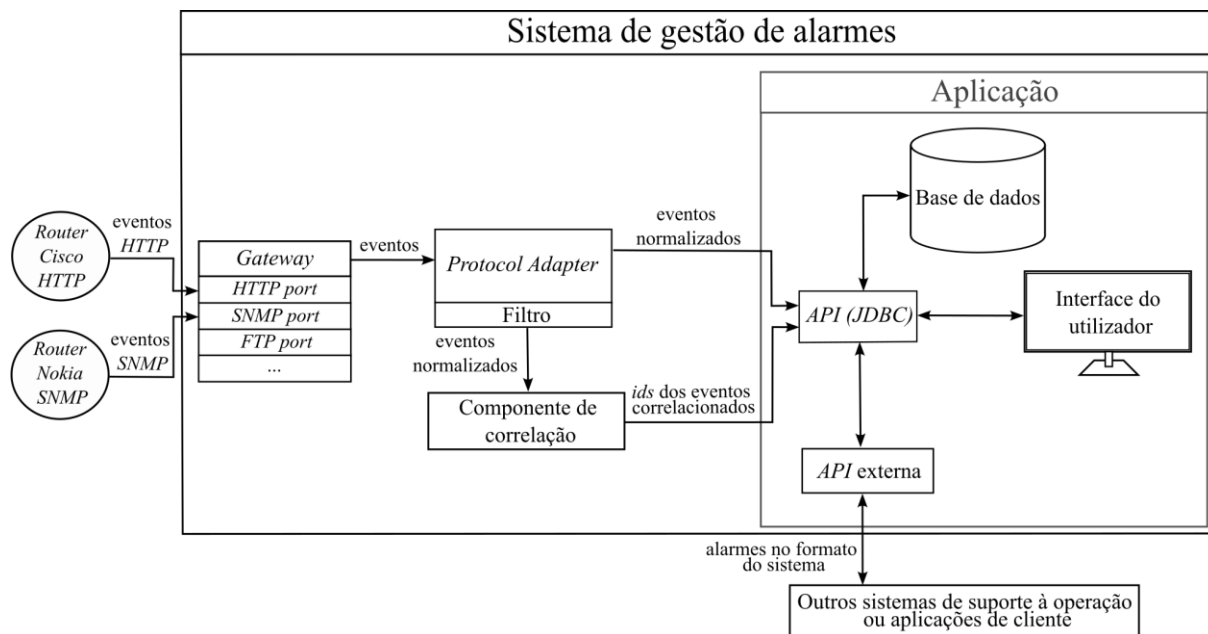


Figura 3: Arquitetura do sistema de gestão de alarmes.

2.2.2 Aplicação

A aplicação do sistema centraliza a informação relevante sobre os eventos de falha detetados na rede. Através da sua interface o utilizador consegue visualizar a lista de alarmes guardados na base de dados e informações importantes acerca de cada alarme, como a sua origem, detalhes sobre o problema, o sistema onde foi detetado, identificação do aparelho, entre outros. Através destas informações os clientes conseguem detetar os problemas mais facilmente e atuar de forma mais rápida, enviando equipas devidamente qualificadas para efetuar as manutenções.

O processo de gestão de alarmes é feito através da janela de alarmes ativos, representada na Figura 4. Nesta janela consta a lista de alarmes registados na base de dados do sistema.

ações	detalhes	#	registo	últ. alteração	subsistema	tecnologia	tipo entidade	entidade	problema	TTK
		190	2023/02/08 15:46:32	2023/02/08 23:39:27	ALMMON	SYS	Probe	subsystem: AMDEV	Probable Inactive	
		288	2023/02/08 15:44:47	2023/02/08 23:38:37	LIMIAR	SYS	Monitor	blade-web trap (10.112.100.38:161, 1.3.6.1.4.1...	Test SNMP fail	
		188	2023/02/08 15:46:32	2023/02/08 23:38:28	ALMMON	SYS	Probe	subsystem: SUB5804	Probable Inactive	
		96	2023/02/08 15:44:51	2023/02/08 23:37:57	ALMMON	SYS	Monitor	maquina_xx (10.10.10.10)	Test Ping fail	
		119	2023/02/08 18:01:27	2023/02/08 23:37:27	ALMMON	SYS	Probe	subsystem: ALMMGR_TEST	Probable Inactive	
		75	2023/02/08 17:27:26	2023/02/08 23:37:26	ALMMON	SYS	Probe	subsystem: A56	Probable Inactive	
		1	2023/02/08 17:43:30	2023/02/08 17:43:30	COUNTERS_TE...	COUNTERS_TE...	COUNTERS...	COUNTERS_TEST	Testing Counters Module - ALMGR-3392	
		1	2023/02/08 17:34:02	2023/02/08 17:34:02	ALMMGR_TEST	ALMGR-4468-2	ALMMGR_T...	ALMMGR_TEST	Testing Alarm Manager - ALMGR-4468	
		1	2023/02/08 17:34:01	2023/02/08 17:34:01	ALMMGR_TEST	ALMGR-4468-1	ALMMGR_T...	ALMMGR_TEST	Testing Alarm Manager - ALMGR-4468	
		18	2023/02/08 15:44:47	2023/02/08 16:28:17	AWS	SYS	Monitor	blade-core trap (10.112.100.38:161, sysDescr)	Test SNMP fail	
		1	2023/02/08 16:23:36	2023/02/08 16:23:36	SUB4008	ALMMGR_TEST1	ALMMGR_T...	ALMMGR_TEST	Testing Alarm Manager - ALMGR-4008	
		1	2023/02/08 16:23:28	2023/02/08 16:23:28	SUB4001	ALMMGR_TEST1	ALMMGR_T...	ALMMGR_TEST	Testing Alarm Manager - ALMGR-4001	
		1	2022/02/04 15:46:00	2023/02/08 16:15:08	ALMMON	SYS	Probe	TEST_ALMGR-1084	Testing Alarm Manager - specificProblem	

Figura 4: Janela de alarmes ativos do sistema.

Essencialmente a aplicação disponibiliza ao utilizador três tipos de operações, a filtragem de alarmes, a seleção e a execução de ações sobre alarmes.

A filtragem de alarmes, ilustrada na Figura 5, consiste em procurar alarmes através de valores dos seus atributos, como por exemplo pesquisar todos os alarmes detetados num subsistema e/ou que ocorreram num determinado intervalo de tempo.

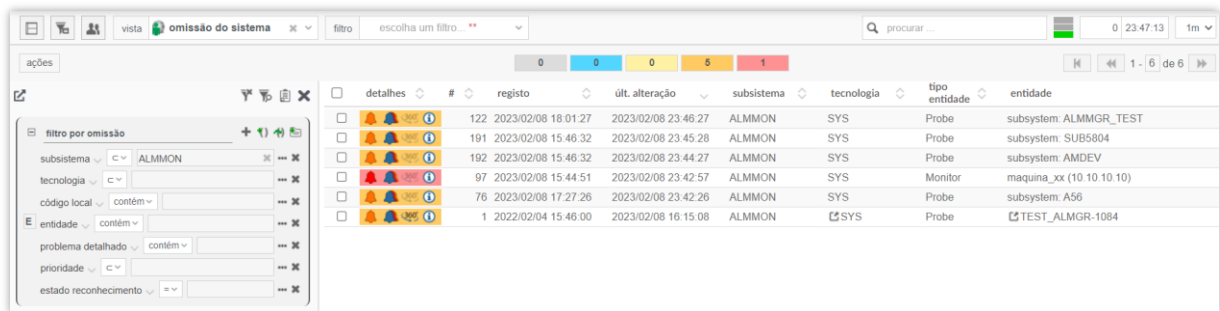


Figura 5: Filtragem de alarmes no sistema.

A seleção de alarmes, representada na Figura 6, serve para selecionar conjuntos de alarmes, é uma operação que normalmente sucede a filtragem e precede a execução de uma ação sobre um grupo de alarmes.

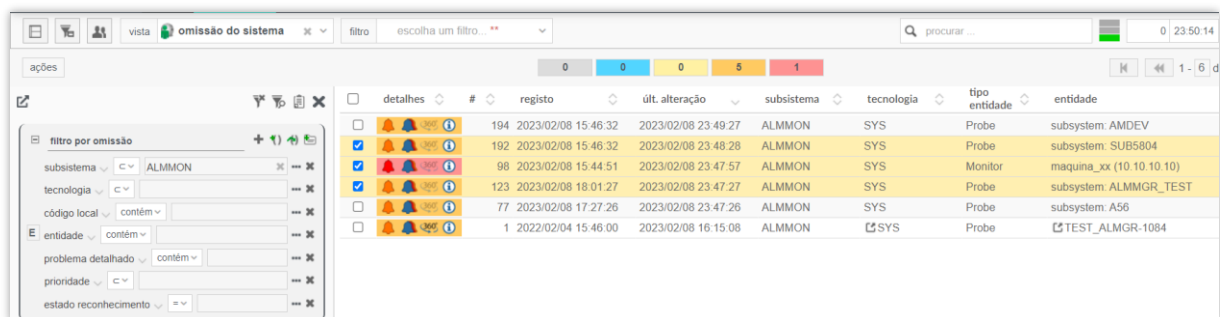


Figura 6: Seleção de alarmes no sistema.

A ação sobre alarmes, apresentada na Figura 7, permite mudar o estado de alarmes e facilitar a sua organização. Os alarmes selecionados podem ser submetidos a uma lista variada de ações, como reconhecer, inibir, engavetar, arquivar, entre outros.

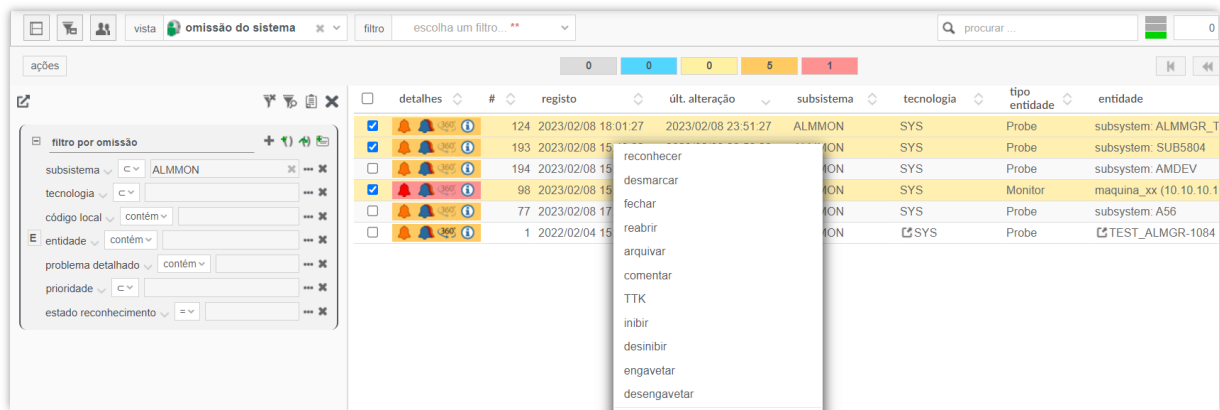


Figura 7: Lista de ações sobre alarmes do sistema.

Estas três operações permitem aos utilizadores da aplicação monitorizar e gerir mais facilmente os alarmes da rede.

2.2.3 API de gestão de alarmes

O sistema possui uma API do tipo *REST* que usa o protocolo *HTTP (Hypertext Transfer Protocol)* para comunicar. Esta API é normalmente utilizada por clientes da fornecedora de serviços de telecomunicações que pretendem dar continuidade à informação retirada do sistema de gestão de alarmes, como por exemplo transmitir informação sobre os alarmes ativos para sistemas de gestão de configurações de rede.

As operações que disponibiliza para gerir alarmes são:

- i. *POST* – Consulta de alarmes;
- ii. *GET* – Consulta de alarmes;
- iii. *GET* – Consulta de alarme pelo *id*;
- iv. *POST* – Executar ações sobre alarmes.

De seguida é apresentada a descrição de cada pedido:

- i. *POST* – Consulta de alarmes: permite consultar a lista de alarmes ativos existentes no sistema.

Este pedido inclui funcionalidades como:

- filtrar alarmes através da inserção de uma *query* no *body* do pedido (*filter*),
- selecionar atributos (*fields*),
- ordenar alarmes (*sort*),
- estabelecer limites para a quantidade de alarmes (*offset* e *limit*),

- selecionar alarmes de acordo com o seu estado (*inhibited, maintenance, shelved, visibility*) - os diferentes estados dos alarmes estão relacionados com ações sobre alarmes, estes permitem classificar os alarmes para facilitar a sua análise.
- ii. GET – Consulta de alarmes: à semelhança do *POST*, também permite consultar a lista de alarmes ativos, a principal diferença entre estes pedidos é o tipo de método usado e o filtro de alarmes no *POST* ser enviado através do body, enquanto no *GET* é enviado como um parâmetro, necessitando que o mesmo seja enviado de forma codificada (*url encoded*).
 - iii. GET – Consulta de alarme pelo id: permite obter a informação de um único alarme. As funcionalidades deste pedido incluem a seleção de atributos (*fields*) e do estado do alarme (*inhibited, maintenance, shelved e visibility*).
 - iv. POST – Executar ações sobre alarmes: permite alterar um alarme em específico ou grupos de alarmes, facilitando o processo de organização e priorização dos mesmos. As possibilidades de ações incluem:
 - Reconhecer (*acknowledge*),
 - Encerrar (*close*),
 - Arquivar (*archive*),
 - Adicionar comentários (*comment*),
 - Associar a *tickets* de problema (*associate to trouble ticket*),
 - Inibir (*inhibit*),
 - Engavetar (*shelve*)
 - e as respetivas ações contrárias.

2.3 Programa de *Open APIs* da *TM Forum*

A *TM Forum (TeleManagement Forum)* é uma associação industrial que se baseia na colaboração entre empresas para facilitar a comunicação entre fornecedores de serviços de telecomunicações e os seus clientes. Esta associação é constituída por mais de 850 empresas que trabalham em conjunto para evoluir tecnologicamente.

Os membros desta associação colaboram entre si ao partilhar experiências e conhecimentos para resolver desafios complexos, criar serviços inovadores e promover o desenvolvimento tecnológico sustentável. A *TM Forum* pretende, desta forma, disponibilizar um ambiente aberto e de suporte que

ajuda os fornecedores de serviços de comunicações a transformar-se e a acompanhar a evolução da era digital (TM Forum, 2021).

A *TM Forum* possui um programa de *Open APIs*, uma iniciativa que apela à facilidade na conectividade, interoperabilidade e portabilidade entre diferentes sistemas. O programa é constituído por 60 *APIs* que são utilizadas por 2300 organizações. Uma das grandes vantagens da implementação das *Open APIs* é o facto de serem compatíveis com diferentes tipos de tecnologias e disponibilizarem de forma aberta as funcionalidades dos seus serviços, facilitando notoriamente o tempo e custo de integração (TM Forum, 2022a).

Solução proposta – API de gestão de alarmes (TMF642):

Uma das *APIs* disponibilizadas pela *TM Forum* é a de Gestão de Alarmes (*TMF642*), concebida para permitir aos fornecedores de serviços de comunicações disponibilizar uma interface de gestão de falhas que possa ser utilizada tanto para cenários simples de monitorização de alarmes como para situações mais complexas que envolvam a comunicação entre diferentes sistemas de suporte à operação. Para tal as especificações da *API* seguem um conjunto de recomendações e normas que permitem melhorar a interoperabilidade entre os sistemas, incluindo as recomendações *ITU*, alinhamento com a *NGMN*, *ETSI*, entre outros (TM Forum, 2022c).

O objetivo ao implementar a API de gestão de alarmes (*TMF642*) é criar um ponto de acesso aberto que permite a clientes da empresa efetuar operações de criação, leitura e alteração de alarmes. A arquitetura do sistema de gestão de alarmes com a implementação da API aberta é representada na Figura 8.

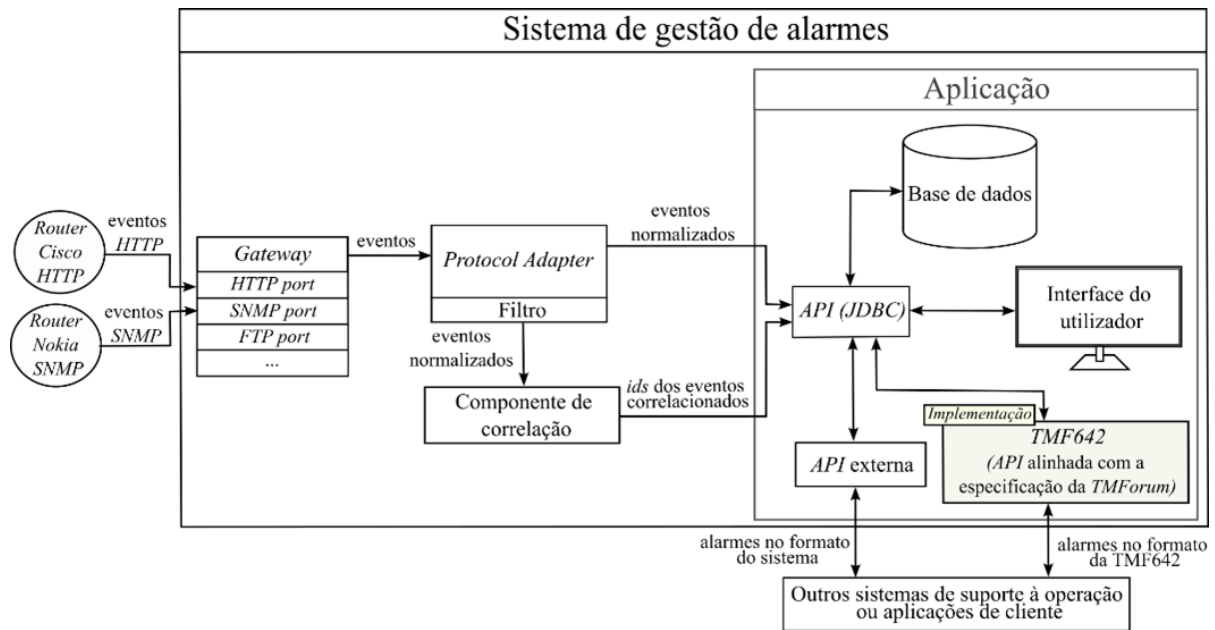


Figura 8: Arquitetura do sistema com a implementação da *TMF642*.

Como se pode observar através da Figura 8, esta nova alternativa de comunicação será utilizada pelos clientes da fornecedora de telecomunicações para aceder à informação e funcionalidades do sistema. Esta *API* irá fornecer informação sobre os alarmes do sistema num formato normalizado, alinhado com a especificação e com as normas que consta materializar.

2.4 Outras alternativas à solução proposta

Para o estudo de alternativas possíveis à *API* de gestão de alarmes da *TM Forum* foi feita uma pesquisa por soluções idênticas no contexto de gestão de falhas e alarmes em redes de telecomunicações. Devido à *TMF642* ser uma especificação aceite de forma global e ao facto de estar alinhada com as principais normas de gestão de alarmes, não foi possível encontrar uma alternativa semelhante à mesma que sirva para o mesmo propósito. Apesar disto, foram identificadas algumas soluções alternativas à sua implementação. As opções consideradas são as seguintes:

- Desenvolvimento de uma *REST API* independente, baseada nas principais normas de gestão de alarmes em redes de telecomunicações, tais como:
 - *3GPP TS 32.111-2* (3GPP, 2022b);
 - *ITU-T X.733* (ITU, 1992);
 - *ITU-T M.3100* (ITU, 2005).

Comparativamente à abordagem da implementação da *TMF642*, isto permite que a *API* seja desenvolvida de forma mais livre, adequando-se mais facilmente ao sistema para o qual está a ser desenvolvida e permite à empresa escolher as normas que considera mais relevantes de serem implementadas conforme as necessidades dos seus clientes.

Por outro lado, tem a desvantagem de não estar completamente alinhada com uma especificação global do serviço de comunicação, reduzindo a probabilidade de interoperabilidade com outros sistemas.

- Desenvolvimento de uma *API* que além de implementar as principais normas de gestão de alarmes, utilize outro tipo de estrutura e/ou protocolos de comunicação. Além das *REST APIs*, tal como é sugerido pela especificação da *TM Forum*, existem outros tipos de soluções que permitem interagir com o sistema. Dadas as características do sistema de gestão de alarmes, algumas das opções analisadas foram:

- *gRPC API* – A *gRPC* é uma tecnologia que implementa o modelo *RPC* e que utiliza o protocolo de comunicação *HTTP 2.0*. Esta solução apresenta uma forma mais simples e direta de definir procedimentos remotos comparativamente à abordagem *REST*. Aspectos como os tipos de métodos *HTTP* usados são pré-definidos no código da *API*, sendo somente necessário invocar os procedimentos, juntamente com os parâmetros necessários, para interagir com o sistema (Nally M., 2020).

Algumas das principais vantagens deste tipo de solução são o facto de, por utilizar *HTTP 2.0* permitir comunicação em tempo real e tornar os serviços mais escaláveis (manter um tempo de resposta estável independentemente do aumento dos utilizadores).

Por outro lado, tem a desvantagem de ser mais complexo de desenvolver e usar, pois exige a configuração de *protocol buffers* para se adaptar a diferentes formatos de mensagens e interfaces. Além disso como utiliza o protocolo *HTTP 2.0* inclui uma lista mais reduzida de *browsers* que o suportam.

- *GraphQL API* – A *GraphQL* é uma linguagem de processamento e execução de *queries* para *APIs* que foi desenvolvida para proporcionar uma alternativa à *REST*. Esta foi desenvolvida no intuito de melhorar a eficiência de acesso aos dados e reduzir a complexidade de acesso aos mesmos, especialmente em casos que se pretende aceder a informações de diferentes recursos (*resources*). Isto permite que a execução de uma única *query* substitua a execução de vários pedidos *REST* e a extração manual da informação pretendida (Quiña-Mera et al., 2023).

Conforme as características apresentadas esta é uma estrutura que tem ganho bastante popularidade nos últimos anos, uma vez que permite reduzir o tempo despendido em juntar informação e evitar o envio e receção de dados que não são inteiramente necessários, aumentando a eficiência da comunicação entre sistemas.

Alguns aspetos menos positivos da *GraphQL* são o facto de ser uma estrutura mais recente, para a qual existe menos suporte comparativamente à *REST* e além disso pelo facto de implementar as *queries* obriga os utilizadores da *API* a ter conhecimentos adicionais sobre a linguagem e como extrair as informações pretendidas.

3. PLANEAMENTO DA IMPLEMENTAÇÃO

Este capítulo é relativo ao planeamento da implementação da *Open API* no sistema de gestão de alarmes.

As secções apresentadas no mesmo são:

- [Verificação de compatibilidade](#) – onde se verifica se o sistema possui os recursos necessários para a implementação da solução;
- [Lista de ferramentas](#) – a descrição das ferramentas utilizadas para a implementação e a sua utilidade no contexto do projeto.

3.1 Verificação de compatibilidade

A verificação de compatibilidade com a especificação envolveu o estudo dos recursos de apoio fornecidos no site da *TM Forum* em paralelo com a compreensão do funcionamento do sistema. Desta forma foi possível ter uma melhor perceção da possibilidade de implementar a nova solução e de que forma se poderia proceder à sua ligação ao sistema.

Esta secção inclui:

- [a descrição dos recursos disponibilizados pela *TM Forum* para facilitar a implementação;](#)
- [o levantamento de pedidos e funcionalidades obrigatórias de implementar;](#)
- [comparação e mapeamento de atributos de alarme entre a *Open API* e o sistema;](#)
- [conclusões acerca da compatibilidade.](#)

3.1.1 Recolha dos recursos de apoio à implementação e análise inicial

Foi feita uma pesquisa pelo site da organização (TM Forum, 2021), onde foram identificados documentos e ferramentas importantes para a implementação da solução, disponibilizados na tabela de *APIs* do programa de *Open APIs* (TM Forum, 2023). O conteúdo da tabela é apresentado na Figura 9.







TM Forum Open APIs	Document Number	Swagger (Apache 2.0 or RAND)	API User Guide / Specification (RAND)	Conformance Profile (RAND)	CTK	Sample Implementation Code	Postman Collection	Release	Swagger Version	Publication Date	ODA Domain
Alarm Management API The Alarm Management API applies lessons that were learned in previous generations of similar APIs that were implemented in the Telecommunication industry, starting from ITU recommendations, TM Forum OSS/J, MTOSI and TIP interfaces, NGMN alignment initiative between 3GPP and TM Forum interfaces, and the more recent ETSI work on requirements for NFV interfaces.	TMF642							21.5.0	v4.0.0	09-Jul-2021	Resource

Figura 9: Tabela de recursos da *TMF642* (TM Forum, 2023)

O principal documento utilizado para a implementação foi o guia de utilizador da *API* de gestão de alarmes (*Alarm Management API User Guide / Specification*) que contém toda a informação sobre as especificações, arquitetura e normas utilizadas nos pedidos (TM Forum, 2022c). Este documento foi utilizado essencialmente para:

- Entender os recursos/objetos envolvidos na *Open API* e os atributos que os descrevem, ou seja, neste caso, os atributos obrigatórios para o objeto alarme;
- A forma como eram enviados os pedidos e as respostas esperadas.

Adicionalmente também foi essencial o uso do *kit* de testes automáticos (*CTK – Conformance Test Kit*). Este *kit* consiste numa verificação automática que é feita para confirmar o alinhamento da *API* com as especificações e dar algumas informações acerca de funcionalidades que estejam a falhar (TM Forum, 2022b). Além de permitir testar a solução, permitiu complementar a documentação da *API* que não era muito clara em alguns aspetos obrigatórios da implementação (como a identificação de atributos obrigatórios nas respostas da *API*).

É importante ter presente que a *CTK* representa uma forma complementar de verificar o funcionamento da solução, pois devido a ser um *kit* automático não consegue assegurar totalmente o alinhamento e funcionamento correto da *API*. Para tal foi necessário complementar o mesmo com testes manuais de verificação de funcionamento e a comparação com as respostas do guia de utilizador.

A *CTK* é explicada com mais detalhe na fase de [análise e configuração inicial dos testes automáticos](#) da implementação.

3.1.2 Levantamento de pedidos e funcionalidades obrigatórias de implementar

A análise dos recursos mencionados na secção anterior permitiu concluir que a *API* a desenvolver teria de implementar os seguintes pedidos:

- Consulta de alarmes;
- Consulta de um alarme pelo *id*;
- Criação de um alarme;
- Alteração de atributos de um alarme.

Além disso a implementação de funcionalidades de:

- Filtragem de alguns atributos do alarme;
- Seleção de atributos de resposta.

Comparativamente à *API* já existente no sistema, a *TMF642* acrescenta os pedidos obrigatórios de criação e de alteração de alarmes (exigidos pela especificação). Devido a estes pedidos corresponderem a operações normalmente executadas por administradores (não eram até ao momento disponibilizadas em interfaces de acesso externo), foi necessário consultar a opinião da equipa de desenvolvimento do sistema para confirmar a possibilidade de implementação das mesmas. Após confirmação, deu-se continuidade ao processo de verificação.

3.1.3 Comparação de pedidos e mapeamento de atributos

Para a comparação dos pedidos e atributos entre o sistema e a *Open API* procedeu-se a uma comparação entre a atual *API* utilizada no sistema e a especificação da *TM Forum*. Foram utilizados exemplos fornecidos no:

- *User guide* da *TMF642* (TM Forum, 2022c);
- Documentação da *API* do sistema.

Através dos exemplos consultados foi criada uma tabela, apresentada no [Apêndice I](#) que permite visualizar os pedidos lado a lado. Ao analisar a estrutura dos pedidos foi possível identificar atributos com nomes e valores idênticos. Adicionalmente, foi consultada a descrição de cada atributo em ambas as documentações.

Com base nestas informações foi realizado o mapeamento dos atributos, representado no [Apêndice II](#). Este consiste numa tabela de correspondência entre os atributos do sistema e da especificação, juntamente com as descrições fornecidas e o tipo de dados esperados em cada atributo.

3.1.4 Conclusões acerca da compatibilidade

A *API* especificada pela *TM Forum* tem muitas semelhanças com a *API* utilizada atualmente pelo sistema e com a estrutura dos alarmes do sistema. A principal diferença entre as duas abordagens é a nomenclatura dos atributos e dos valores enumerados. Do lado do sistema esta nomenclatura segue uma abordagem mais prática, com nomes predominantemente abreviados e mais difíceis de entender, enquanto do lado da *TM Forum* segue uma abordagem mais alinhada com as normas *ITU-T X.733* (ITU, 1992) e *3GPP TS 32.111-2* (3GPP, 2022a), promovendo uma melhor percepção do significado de cada atributo presente nos pedidos.

A comparação dos pedidos, demonstrada nos apêndices [I](#) e [II](#), permitiu verificar que há possibilidade de implementar a *TMF642* no sistema, pois existe correspondência para todos os atributos exigidos pela especificação.

Também foi possível concluir que existe uma diferença notória na dimensão das respostas, a *API* do sistema possui uma quantidade muito superior de atributos em cada alarme que não são mencionados na documentação da *TMF642*.

Apesar disso, a *TMF642* tem a vantagem de ser extensível, o que significa que além dos atributos obrigatórios de serem implementados permite adicionar novos, adequando-se a sistemas mais complexos.

3.2 Lista de ferramentas

Para o desenvolvimento da parte prática do projeto e para testar a conformidade do mesmo foi necessário recorrer a algumas ferramentas que permitiram facilitar a sua execução. A lista de ferramentas utilizadas, a sua descrição e utilidade são apresentadas na Tabela 1.

A lista de ferramentas foi determinada pela fornecedora de serviços de telecomunicações, não são apresentadas alternativas ou justificações de escolha pois eram as ferramentas amplamente adotadas pela equipa de desenvolvimento.

Tabela 1: Lista de ferramentas usadas na implementação

Ferramenta	Descrição	Utilidade
<i>Postman</i>	<p>Uma plataforma que permite construir e testar <i>APIs</i>. Fornece uma interface que permite criar pedidos para vários tipos de protocolos, como <i>HTTP</i>, <i>REST</i>, <i>SOAP</i>, <i>GraphQL</i> e <i>WebSockets</i>. Além disso permite facilitar a gestão dos pedidos através de grupos (coleções), detetar automaticamente a linguagem das respostas e possui suporte para protocolos de autenticação como o <i>OAuth 1.2/2.0</i> (Postman, 2023).</p> <p>Página oficial: https://www.postman.com/</p>	<p>Efetuar testes de pedidos das <i>APIs</i>, tanto da existente no sistema como da solução desenvolvida</p>
<i>IntelliJ</i>	<p>O <i>IntelliJ</i> é um editor de código normalmente utilizado para desenvolvimento em <i>Java</i>. Inclui extensões e funcionalidades que permitem facilitar o desenvolvimento, detetar erros (<i>debugging</i>) e promover o trabalho colaborativo (IntelliJ, 2021).</p> <p>Página oficial: https://www.jetbrains.com/idea/</p>	<p>O <i>IntelliJ</i> foi utilizado para analisar e editar o código da <i>TMF642</i>. Além disso foi utilizada a sua extensão do <i>Maven</i> para compilar e guardar (<i>commit</i>) o código.</p>
<i>Jenkins</i>	<p>Servidor de automação que suporta a construção, preparação de ambientes e automação de projetos (Jenkins, 2023).</p> <p>Página oficial: https://www.jenkins.io/</p>	<p>Utilizado para executar diariamente os testes automáticos da <i>TMF642</i> (<i>CTK</i>), para facilitar o acompanhamento constante da <i>API</i> e publicar o código no ambiente de desenvolvimento.</p>
<i>Swagger</i>	<p>Ferramenta de simplificação de desenvolvimento de <i>APIs</i>. Inclui a <i>OpenAPI Specification</i>, uma</p>	<p>Desenvolvimento da documentação da <i>API</i>.</p>

	<p>norma que sugere as melhores práticas para a representação de <i>APIs</i> (Swagger, 2023).</p> <p>Página oficial: https://swagger.io/</p>	
<i>Docker</i>	<p>Ferramenta de desenvolvimento, entrega e execução de aplicações em contentores (<i>containers</i>). Permite reduzir a necessidade de configuração de infraestruturas ao abstrair as configurações juntamente com as aplicações (Docker, 2022).</p> <p>Página oficial: https://www.docker.com/</p>	<p>Conversão da documentação da <i>API</i> do formato <i>json</i> para uma página do tipo <i>html</i>.</p>
<i>Apache JMeter</i>	<p>Ferramenta de medição de desempenho de software. Permite testar aplicações e funcionalidades, incluindo pedidos <i>HTTP</i>. Inclui vários gráficos e tabelas que permitem facilitar a análise dos resultados.</p> <p>Página oficial: https://jmeter.apache.org/</p>	<p>Utilizado para executar testes de desempenho à solução implementada e comparar o seu funcionamento com a <i>API</i> que já existia no sistema.</p>

4. IMPLEMENTAÇÃO DA OPEN API/NO SISTEMA

O processo de implementação foi faseado, sendo a parte de desenvolvimento intercalada com a verificação de conformidade através da execução de testes manuais e dos testes automáticos (CTK) fornecidos pela *TM Forum*.

A implementação foi dividida nas seguintes fases:

- [Criação de um novo serviço no sistema;](#)
- [Análise e configuração inicial dos testes de conformidade automáticos \(CTK\);](#)
- [Desenvolvimento do pedido *GET* – Consulta de alarmes;](#)
- [Desenvolvimento do pedido *GET* – Consulta de um alarme pelo seu id;](#)
- [Implementação da funcionalidade seleção de atributos;](#)
- [Implementação da funcionalidade de filtragem de alarmes;](#)
- [Desenvolvimento do pedido *POST* – Criação de alarme;](#)
- [Desenvolvimento do pedido *PATCH* – Alteração de um alarme;](#)
- [Implementação de valores enumerados;](#)
- [Validação final da *API* através dos testes de conformidade automáticos \(CTK\).](#)

4.1 Criação de um novo serviço no sistema

Inicialmente, para a implementação da *API* da *TM Forum* foi criado um novo módulo localizado dentro dos serviços *REST* do sistema. A estrutura deste módulo foi criada com base nos restantes módulos dentro da diretoria *REST*, seguindo uma arquitetura idêntica. O módulo criado foi denominado de acordo com o código da respetiva *API*, a *TMF642*, e a sua localização e estrutura é representada na Figura 10. No interior do módulo foi criada uma classe do tipo interface (*TMFService*) para conter as interfaces de cada um dos pedidos da *API* e informações como o tipo de pedido, o *path* e os parâmetros. Foi também criada uma classe de implementação (*TMFServiceImpl*) destinada a conter a lógica para cada um dos pedidos e a invocação de métodos internos do código do sistema. Além disso, foi criada uma classe opcional (*TMFExceptionEnum*) para enumerar as exceções da *API*.

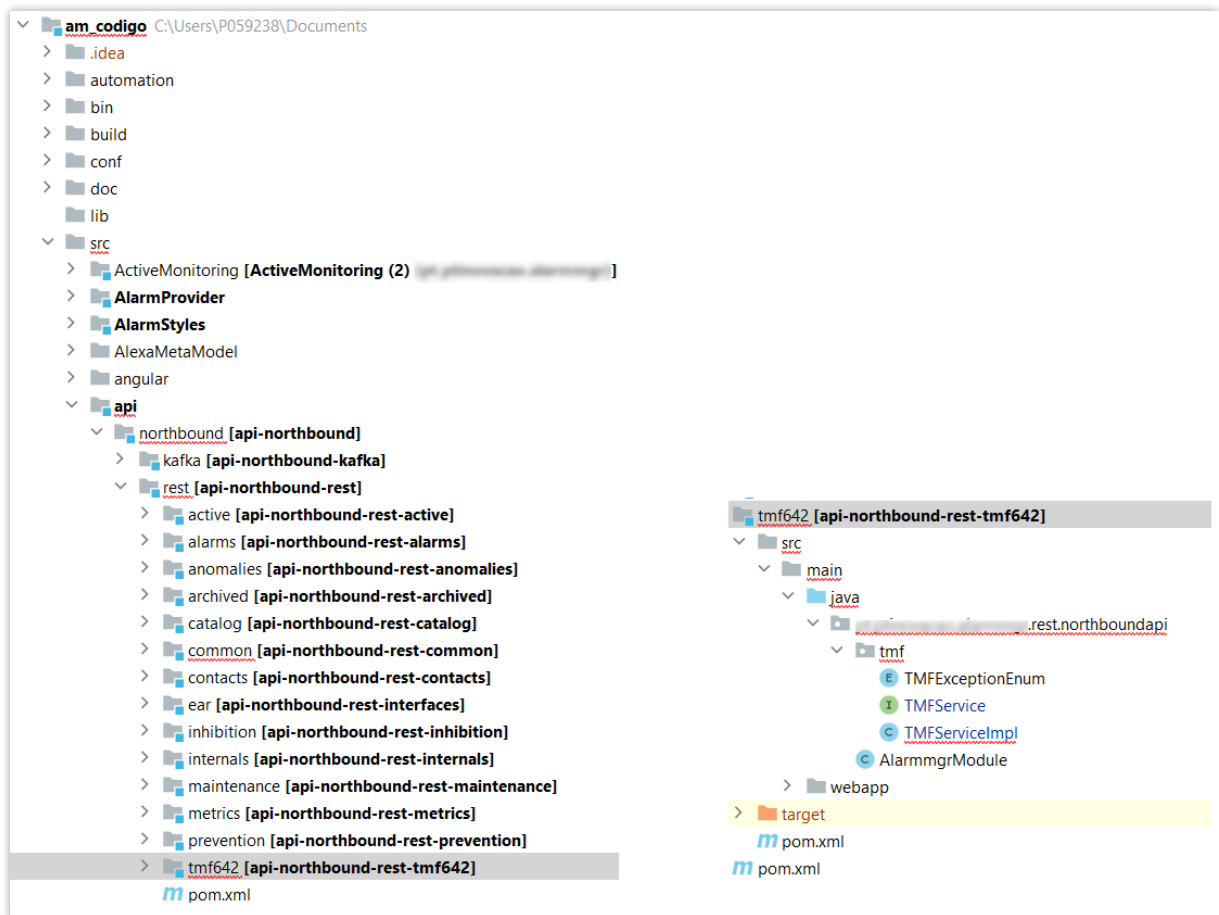


Figura 10: Criação do módulo da *TMF642*.

Na fase inicial de desenvolvimento da solução foi necessário entender como funcionava o fluxo de informação no sistema. Era essencial criar um novo serviço sem interferir com os serviços existentes e, no entanto, tentar reutilizar o máximo de recursos possível. Para tal foi necessário analisar o funcionamento da *API* já existente e entender como interagira com o restante código do sistema, permitindo clarificar as alterações necessárias e a melhor forma de iniciar o desenvolvimento da nova solução.

Devido aos pedidos *GET* serem mais simples, uma vez que já existiam na *API* do sistema, estes foram os primeiros pedidos a ser implementados.

4.2 Análise e configuração inicial dos testes de conformidade automáticos (CTK)

A constituição do *kit* de testes automáticos (CTK) disponibilizado pela *TM Forum* pode ser observado através da Figura 11. Este é constituído por:

- Um ficheiro de configurações (*config.json*) – contém as informações necessárias para o envio dos pedidos, nomeadamente o endereço raiz da *API (root path)*, os dados de autenticação e um *payload* com dados para a criação de um alarme;
- Uma *script* (neste caso foi utilizada a *Windows-PowerShell-RUNCTK*) – que utiliza o ficheiro de configurações (*config.json*) para efetuar testes aos pedidos;
- Um ficheiro *html* e um *json* idênticos (*results.html* e *results.json*) – gerados automaticamente após a *script* de teste ser executada e que contêm os resultados dos testes.

Name	Date modified	Type	Size
conformance	17/02/2022 14:03	File folder	
ctk	20/02/2023 11:58	File folder	
config	29/03/2023 19:22	JSON File	2 KB
htmlResults	29/03/2023 19:22	Brave HTML Docu...	398 KB
jsonResults	29/03/2023 19:22	JSON File	7 023 KB
LICENSE	18/02/2022 15:32	File	2 KB
Mac-Linux-RUNCTK.sh	18/02/2022 15:32	SH File	1 KB
README	18/02/2022 15:32	MD File	2 KB
Windows-Bat-RUNCTK	18/02/2022 15:32	Windows Batch File	1 KB
Windows-PowerShell-RUNCTK	18/02/2022 15:32	Windows PowerS...	1 KB

Figura 11: Estrutura da *CTK*

Inicialmente foi necessário proceder a algumas alterações ao ficheiro de configurações da *CTK (config.json)* para que os testes fossem direcionados para o *endereço do serviço* e possuísem a autenticação necessária. Na Figura 12 é possível observar do lado esquerdo o ficheiro original e do lado direito as alterações realizadas.

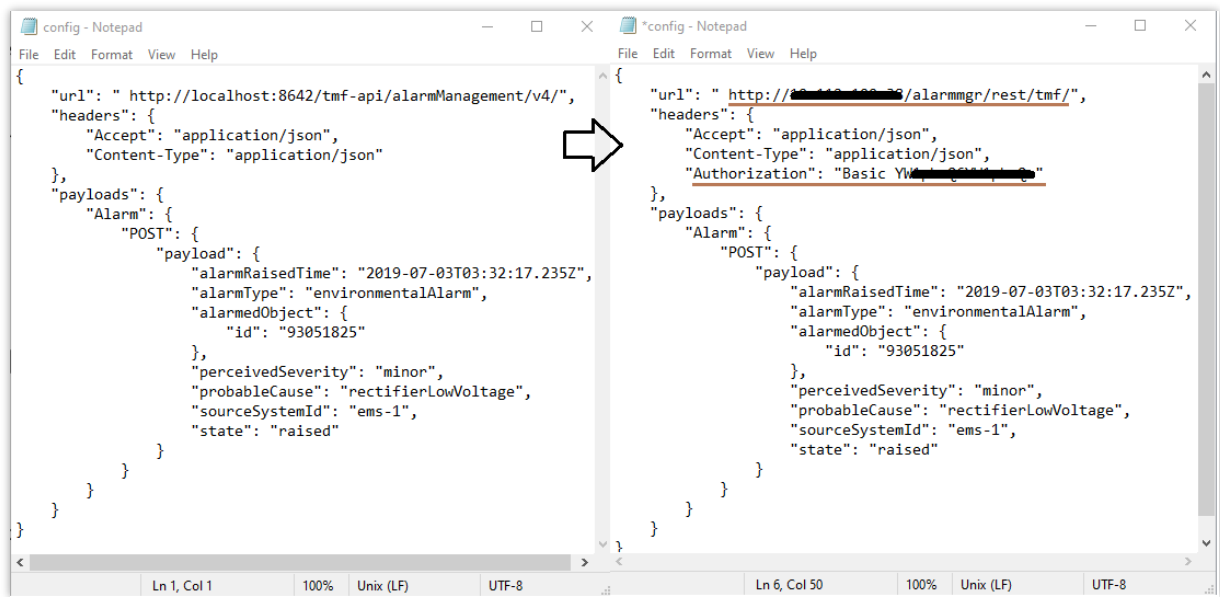


Figura 12: Configuração do *CTK* para o pedido *GET* – Consulta de alarmes.

Os resultados globais da *CTK* podem ser observados na Figura 13. Esta contem:

- **Test Scripts** - a quantidade de pedidos executados (correspondem às operações obrigatórias de implementar);
- **Assertions** - pequenas confirmações de alinhamento com as especificações (como a presença de atributos obrigatórios).

Newman Report		
Collection	CTK-Alarm-4.0.0	
Time	Fri May 05 2023 09:18:39 GMT+0100 (Hora de verão da Europa Ocidental)	
Exported with	Newman v4.6.1	
	Total	Failed
Iterations	1	0
Requests	22	0
Prerequisite Scripts	0	0
Test Scripts	22	20
Assertions	20	19
Total run duration	3.9s	
Total data received	1.46KB (approx)	
Average response time	72ms	
Total Failures	39	

Figura 13: Resultado inicial da *CTK*.

Como é possível observar através da Figura 13, quase todos os testes falharam, pois, ainda não tinha sido realizada a implementação de qualquer pedido. Ao longo da implementação dos pedidos o conteúdo

da *CTK* foi analisado para assegurar que só eram desenvolvidas as funcionalidades necessárias. O resultado final da *CTK* é apresentado [na fase de validação final da API](#).

Nota importante:

Nas fases de desenvolvimento de um pedido/funcionalidade (por exemplo: [Desenvolvimento do pedido GET – Consulta de alarmes](#)) são apresentados links para os resultados finais da *CTK* em relação ao pedido desenvolvido. Estes resultados servem para o leitor verificar que o pedido/funcionalidade desenvolvido foi aprovado pelos testes automáticos da *TM Forum*, no entanto os resultados só foram obtidos no fim da implementação de todos os pedidos e funcionalidades, ou seja, não representam o resultado do teste na altura de desenvolvimento do mesmo.

No capítulo do método de implementação, na fase [5](#) (Realização de testes automáticos da *TM Forum*) é explicado com mais detalhe o motivo dos testes não demonstrarem imediatamente o funcionamento de um pedido

4.3 Desenvolvimento do pedido *GET* – Consulta de alarmes

O primeiro pedido a ser criado foi o de consulta de alarmes. Inicialmente procedeu-se à análise dos resultados esperados na *CTK*, representados na Figura 14. Tal como se pode observar a partir da mesma, é de esperar que o pedido contenha para cada alarme listado os atributos obrigatórios:

- *alarmRaisedTime*;
- *href*;
- *id*;
- *probableCause*;
- *sourceSystemId*;
- *state*.

/Alarm			
Description	This operation search for the created Alarm		
Method	GET		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm		
Mean time per request	1079ms		
Mean size per request	23.75KB		
Total passed tests	430		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200 or 206	0	1
	Instance has all mandatory attributes	0	33
	Response has <u>alarmRaisedTime</u> attribute	0	33
	Response has <u>href</u> attribute	0	33
	Response has <u>id</u> attribute	0	33
	Response has <u>probableCause</u> attribute	0	33
	Response has <u>sourceSystemId</u> attribute	0	33
	Response has <u>state</u> attribute	0	33
	Body includes value held on alarmRaisedTime	0	33
	Body includes value held on href	0	33
	Body includes value held on id	0	33
	Body includes value held on probableCause	0	33
	Body includes value held on sourceSystemId	0	33
	Body includes value held on state	0	33

Figura 14: Análise inicial dos resultados esperados no pedido *GET* – Consulta de alarmes.

Inicialmente foi necessário criar uma classe de alarme chamada ***TMFAlarm.java***, com os atributos indicados na especificação, utilizada para apresentar os alarmes aos utilizadores da *TMF642* no formato da especificação. A classe e os respetivos atributos que a constituem são apresentados na Figura 15.

Além disso, para ligar a *TMF642* ao sistema foram criadas funções de conversão, para mapear os atributos entre a classe *TMFAlarm.java* e o restante código da aplicação. A ligação que é feita entre a *API*, a nova classe de alarme e o restante código de processamento de alarmes é representada pela Figura 15 e a Tabela 2 explica a sua numeração.

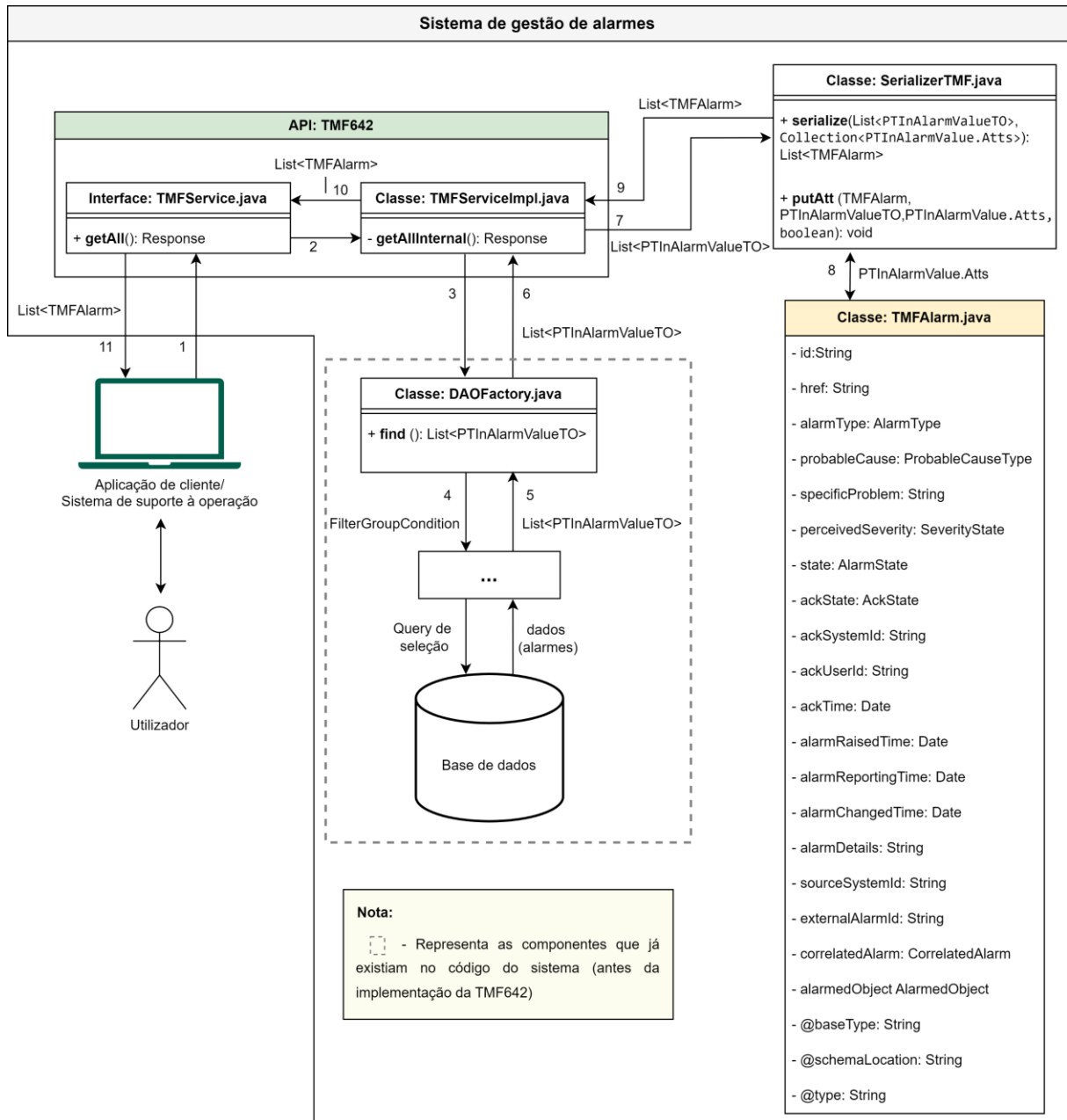


Figura 15: Diagrama de classes e de fluxo de informação do pedido *GET* – Consulta de alarmes.

Tabela 2: Descrição do diagrama do pedido *GET* – Consulta de alarmes.

Nº	Descrição
1	O utilizador envia um pedido do tipo <i>GET</i> para o <i>endpoint</i> - <i>https://<server_ip>/alarmmgr/rest/tmf/alarm</i>
2	O pedido é enviado para o método <i>getAllInternal()</i> , responsável por invocar os restantes métodos para a obtenção da resposta do pedido.
3, 4	Inicialmente invoca o <i>find()</i> que por sua vez invoca outras funções internas que fazem uma busca pelos alarmes ativos registados na base de dados.

5, 6	A base de dados devolve a lista de alarmes ativos que são convertidos no formato <i>List<PTInAlarmValueTO></i> e devolvidos à função <i>getAlIInternal()</i> .
7	A <i>getAlIInternal()</i> invoca o <i>serialize()</i> que percorre a lista de alarmes no formato <i>PTInAlarmValueTo</i> e para cada um deles invoca o método <i>putAtt()</i> .
8	O método <i>putAtt()</i> percorre todos os atributos de um alarme (<i>PTInAlarmValue.Atts</i>) e faz a conversão de valores do formato <i>PTInAlarmValueTo.java</i> para <i>TMFAlarm.java</i> , permitindo criar alarmes de acordo com as especificações da <i>TMF642</i> .
9	Após percorrer toda a lista de alarmes o <i>serialize()</i> devolve a lista de alarmes convertidos ao método <i>getAlIInternal()</i> . O método <i>getAlIInternal()</i> converte a lista de alarmes no formato <i>JSON</i> .
10, 11	Por sua vez o <i>getAlIInternal()</i> encaminha esta lista de alarmes convertida para a interface através da <i>Response</i> e é finalmente enviada para a aplicação do cliente ou a outro sistema de suporte à operação.

Após o desenvolvimento do pedido foi feita uma verificação de conformidade do seu funcionamento. Para tal procedeu-se à comparação entre a resposta obtida num pedido e a resposta apresentada na documentação da *TMF642*, representada na Figura 16.

Pedido da API em desenvolvimento	Exemplo de pedido do user guide da TMF642
<p>GET <code>http://.../alarmmgr/rest/tmf/alarm...</code></p> <p>Params • Authorization • Headers (8) Body Pre-request Script Tests Settings</p> <p>Body Cookies (1) Headers (12) Test Results</p> <p>Pretty Raw Preview Visualize JSON</p> <pre> 1 { 2 3 "id": "41967207", 4 "href": "/alarmmgr/rest/tmf/alarm/41967207", 5 "state": "Open", 6 "alarmType": "Communications", 7 "perceivedSeverity": "Minor", 8 "probableCause": "Def", 9 "specificProblem": "ps=3,sl=1,in=10", 10 "alarmedObjectType": "NePowerSupply", 11 "alarmedObject": { 12 "id": "tmf_moi_03", 13 "href": "alarmmgr/rest/tmf/alarm/tmf_moi_03" 14 }, 15 "sourceSystemId": "ems-1", 16 "alarmRaisedTime": "2023-02-20T14:32:17Z", 17 "alarmReportingTime": "2023-04-16T00:02:44Z", 18 "alarmChangedTime": "2023-05-05T00:02:13Z", 19 "ackUserId": "null", 20 "ackState": "Unacknowledged", 21 "correlatedAlarm": {}, 22 "@baseType": "Alarm", 23 "@type": "Alarm", 24 "@schemaLocation": "/alarmmgr/repository/docs/tmf642/alarm-schema.json" 25 }, </pre>	<p>GET /tmf-api/alarManagement/v4/alar Accept: application/json</p> <p>Response</p> <p>200</p> <pre> [{ "id": "8675309", "href": "/alarManagement/v4/alar/8675309", "@baseType": "Alarm", "@type": "Alarm", "@schemaLocation": "/alarManagement/v4/schema/Alarm.schema.json", "externalAlarmId": "5551212", "state": "updated", "alarmType": "environmentalAlarm", "perceivedSeverity": "major", "probableCause": "rectifierLowVoltage", "specificProblem": "ps=3,sl=1,in=8", "alarmedObjectType": "Rectifier", "alarmedObject": { "id": "93051825", "href": "/resourceInventoryManagement/v4/resource/93051825" }, "sourceSystemId": "ems-1", "alarmDetails": "voltage=95", "alarmRaisedTime": "2019-07-03T03:32:17.235Z", "alarmReportingTime": "2019-07-03T03:32:17.552Z", "alarmChangedTime": "2019-07-03T03:32:24.715Z" }] </pre>

Figura 16: Comparação entre as respostas do pedido *GET* – Consulta de alarmes.

O resultado final do pedido ao ser submetido à *CTK* pode ser verificado no Apêndice V, em [Resultados CTK do pedido GET – Consulta de alarmes](#), onde é possível observar que todos os alarmes detetados possuem os atributos obrigatórios determinados pela *TM Forum*.

4.4 Desenvolvimento do pedido *GET* – Consulta de alarme pelo *id*

Esta implementação foi relativamente simples, uma vez que é muito idêntica à da consulta de alarmes. A única diferença é que o utilizador tem de acrescentar o *id* do alarme ao *path* do pedido. Este *id* é utilizado para fazer uma busca pelo alarme à base de dados. O diagrama apresentado na Figura 17 representa de que forma é processado o pedido e a Tabela 3 descreve a numeração da figura.

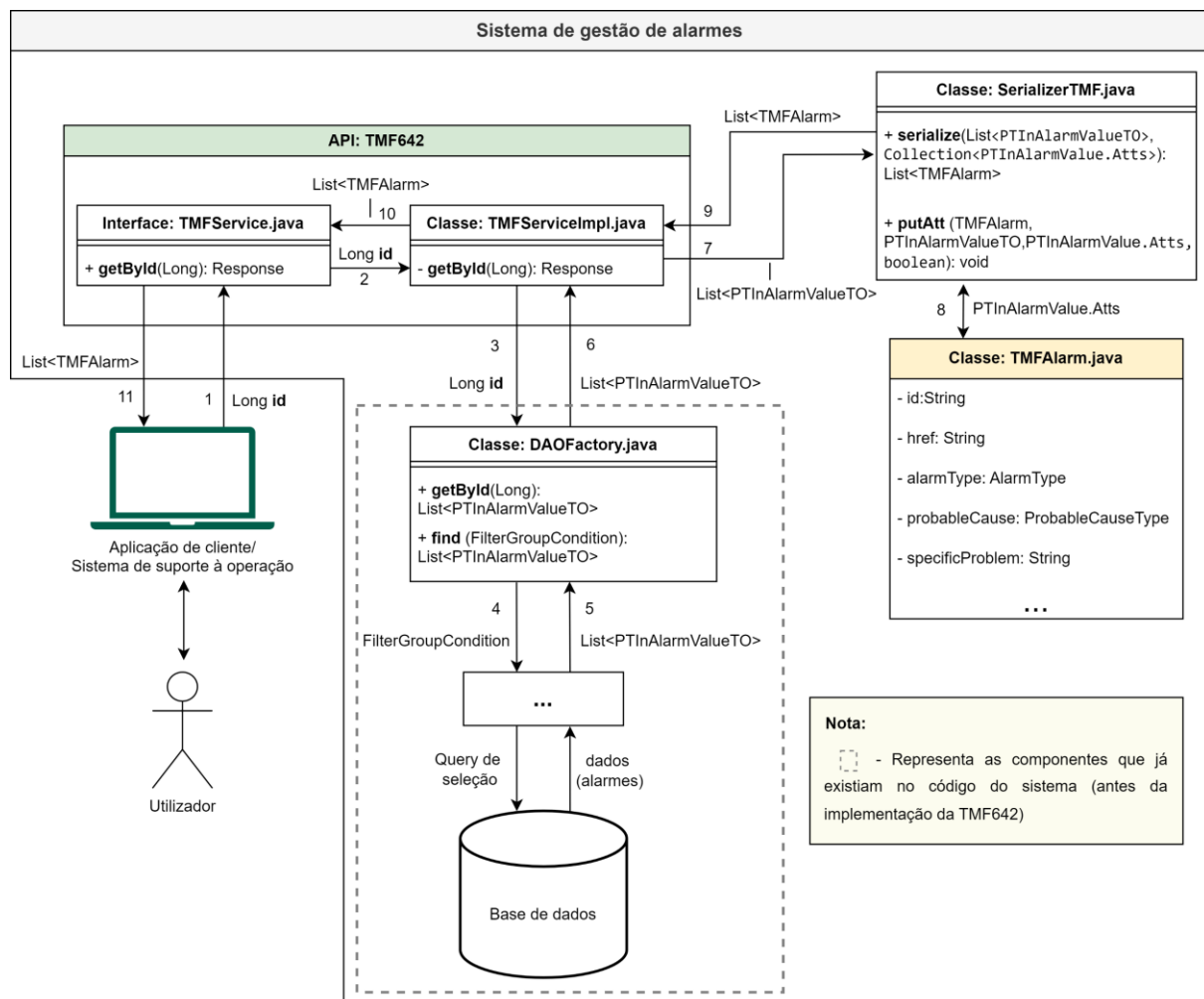


Figura 17: Diagrama de classes e de fluxo de informação do pedido *GET* – Consulta de alarme pelo *id*.

Tabela 3: Descrição do diagrama do pedido *GET* – Consulta de alarme pelo *id*.

Nº	Descrição
1	É enviado um pedido do tipo <i>GET</i> com o <i>id</i> do alarme que se quer receber na resposta: <i>https://<server_ip>/alarmmgr/rest/tmf/alarm/<id></i>
2	O pedido, juntamente com o <i>id</i> , é encaminhado para o método <i>getByld()</i> , responsável por invocar os restantes métodos necessários para a obtenção do alarme.
3	Começa por invocar o <i>getByld()</i> da classe <i>DAOFactory.java</i> que constrói uma condição para filtrar o alarme pelo seu <i>id</i> (<i>FilterGroupCondition</i>). De seguida invoca o <i>find()</i> que recebe a condição de procura.
4	O <i>find()</i> envia a <i>condição</i> para outros métodos internos que eventualmente convertem-na numa <i>query</i> e enviam-na para a base de dados;
5, 6	O alarme devolvido pela base de dados é enviado para o método <i>getByld()</i> no formato <i>List<PTinAlarmValueTo></i> (uma lista com um único alarme).
7	O <i>getByld()</i> invoca o método <i>serialize()</i> que recebe o alarme e passa-o ao <i>putAtt()</i> .
8	O método <i>putAtt()</i> preenche os atributos da classe <i>TMFAlarm.java</i> .
9	A <i>serialize()</i> devolve ao <i>getByld()</i> uma lista de objetos <i>TMFAlarm</i> que contém um único alarme. Para finalizar o <i>getByld()</i> converte o alarme da lista de alarmes num objeto <i>JSON</i> .
10	A lista é encaminhada para o método <i>getByld()</i> da interface <i>TMFService.java</i> .
11	O alarme procurado é enviado na <i>Response</i> para a aplicação do cliente ou para outro sistema de suporte à operação.

Após o desenvolvimento do pedido foi realizada uma comparação entre a resposta obtida ao utilizar a *API* e a resposta fornecida na documentação da *TMF642*. A Figura 18 ilustra esta comparação. Através da mesma pode-se confirmar que o conteúdo da resposta consiste num único alarme, cujo *id* corresponde ao *id* inserido no *path* do pedido.

Pedido da API em desenvolvimento	Exemplo de pedido do <i>user guide</i> da TMF642
<pre>GET http://192.168.1.100:8080/alarmmgr/rest/tmf/alarm/41967207</pre>	<pre>GET /tmf-api/alarmManagement/v4/alarm/8675309 Accept: application/json</pre>
<pre>{ "id": "41967207", "href": "/alarmmgr/rest/tmf/alarm/41967207", "state": "Open", "alarmType": "Communications", "perceivedSeverity": "Minor", "probableCause": "DeF", "specificProblem": "ps=3,sl=1,in=10", "alarmedObjectType": "NePowerSupply", "alarmedObject": { "id": "tmf_moi_03", "href": "alarmmgr/rest/tmf/alarm/tmf_moi_03" }, "sourceSystemId": "ems-1", "alarmRaisedTime": "2023-02-20T14:32:17Z", "alarmReportingTime": "2023-04-16T00:02:44Z", "alarmChangedTime": "2023-05-05T00:02:13Z", "ackUserId": "null", "ackState": "Unacknowledged", "correlatedAlarm": {}, "@baseType": "Alarm", "@type": "Alarm", "@schemaLocation": "/alarmmgr/repository/docs/tmf642/alarm-schema.json" }</pre>	<pre>200 { "id": "8675309", "href": "/alarmManagement/v4/alarm/8675309", "@baseType": "Alarm", "@type": "Alarm", "@schemaLocation": "/alarmManagement/v4/schema/Alarm.schema.json", "externalAlarmId": "5551212", "state": "updated", "alarmType": "environmentalAlarm", "perceivedSeverity": "major", "probableCause": "rectifierLowVoltage", "specificProblem": "ps=3,sl=1,in=8", "alarmedObjectType": "Rectifier", "alarmedObject": { "id": "93051825", "href": "/resourceInventoryManagement/v4/resource/93051825" }, "sourceSystemId": "ems-1", "alarmDetails": "voltage=95", "alarmRaisedTime": "2019-07-03T03:32:17.235Z", "alarmReportingTime": "2019-07-03T03:32:17.552Z", "alarmChangedTime": "2019-07-03T03:32:24.715Z", "ackState": "unacknowledged" }</pre>

Figura 18: Comparação entre as respostas do pedido *GET* – Consulta de alarme pelo seu *id*.

O resultado final dos testes de conformidade *CTK* deste pedido são representados no Apêndice V, em [Resultados CTK do pedido GET – Consulta de alarme pelo id](#), onde se verifica que a resposta do pedido contém todos os atributos obrigatórios exigidos pela *TM Forum*.

4.5 Implementação da funcionalidade de seleção

Inicialmente foi feita uma análise dos resultados esperados na *CTK* para verificar os atributos obrigatórios e a forma como eram selecionados. Estes detalhes são apresentados na Figura 19.

<u>/Alarm?fields=alarmRaisedTime</u>	
Description	This operation filter a Alarm
Method	GET
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?fields=alarmRaisedTime
<u>/Alarm?fields=id</u>	
Description	This operation filter a Alarm
Method	GET
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?fields=id
<u>/Alarm?fields=probableCause</u>	
Description	This operation filter a Alarm
Method	GET
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?fields=probableCause
<u>/Alarm?fields=sourceSystemId</u>	
Description	This operation filter a Alarm
Method	GET
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?fields=sourceSystemId
<u>/Alarm?fields=state</u>	
Description	This operation filter a Alarm
Method	GET
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?fields=state

Figura 19: Identificação de atributos obrigatórios para a funcionalidade de seleção.

Apesar da *CTK* só esperar que os atributos indicados na Figura 19 sejam selecionáveis, a funcionalidade de seleção já existia no código do sistema, portanto, foi mais simples fazer a implementação da mesma para todos os atributos do alarme, tal como na *API* já existente.

Para tal, só foi necessário acrescentar um parâmetro opcional em ambos os pedidos *GET*, este recebe uma *string* com a lista de atributos que o utilizador pretende ver. O restante código de processamento da *string* e de preenchimento dos atributos contidos na mesma já existia internamente.

É possível observar como foi acrescentada a funcionalidade de seleção de atributos através da Figura 20 e da descrição da sua numeração na Tabela 4.

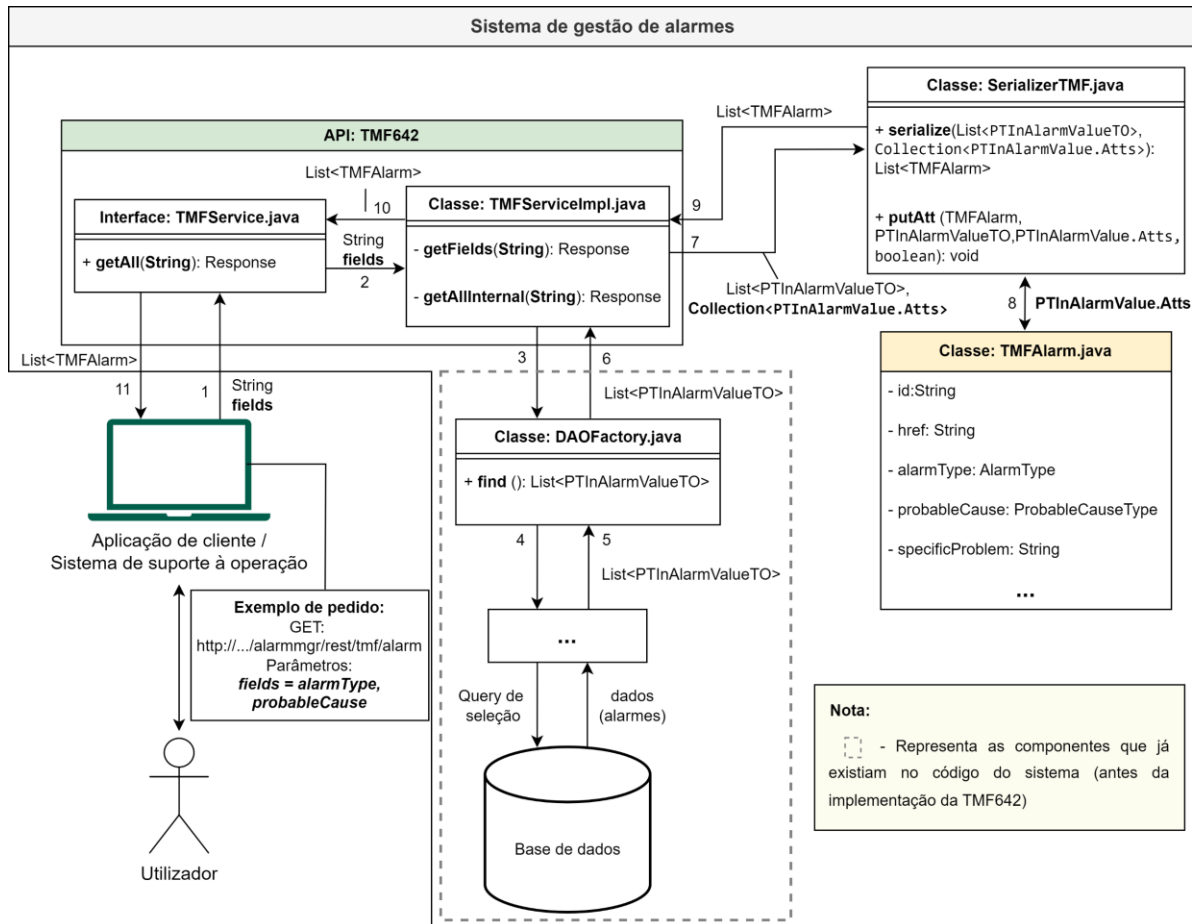


Figura 20: Diagrama de classes e de fluxo de informação do pedido *GET* – Consulta de alarmes com a funcionalidade de seleção de atributos.

Tabela 4: Descrição do diagrama do pedido *GET* – Consulta de alarme com funcionalidade de seleção de atributos.

Nº	Descrição
1	Inicialmente o utilizador envia um pedido do tipo <i>GET</i> onde inclui nos parâmetros do pedido o campo “fields” com a lista dos atributos que pretende visualizar na resposta. Por exemplo: <i>https://<server_ip>/alarmmgr/rest/tmf/alarm?fields=alarmType,probableCause;</i>
2	A <i>String</i> de <i>fields</i> recebida na interface é enviada para o método <i>getAllInternal()</i> ;
3, 4, 5, 6	Tal como na consulta de alarmes é feita uma busca à base de dados por todos os alarmes ativos que são agrupados na lista <i>List<PTInAlarmValueTo></i> . De seguida é invocado o método <i>getFields()</i> que faz uma conversão da <i>string</i> de <i>fields</i> recebida, transformando-a numa lista de atributos (<i>Collection<PTInAlarmValue.Atts></i>);

7	O valor da <i>Collection<PTInAlarmValue.Atts></i> é devolvido ao <i>getAllInternal()</i> e por sua vez invoca o método <i>serialize()</i> com a lista de alarmes e a lista de atributos selecionados como parâmetros;
8	O método <i>serialize()</i> invoca a <i>puttAtt()</i> que preenche os atributos obrigatórios (<i>id</i> e <i>href</i>) e os restantes atributos que constam na lista <i>Collection<PTInAlarmValue.Atts></i> ;
9	A lista de alarmes com os atributos selecionados é devolvida ao método <i>getAllInternal()</i> . Numa fase final este método converte a lista de alarmes para o formato <i>JSON</i> .
10, 11	O objeto é devolvido através da <i>Response</i> à interface <i>TMFService.java</i> e é enviada à aplicação do cliente ou a outro sistema de suporte à operação.

Após o desenvolvimento da funcionalidade de seleção de atributos foi feita uma confirmação do seu funcionamento através da execução de pedidos. Exemplos de pedidos efetuados podem ser observados na Figura 21.

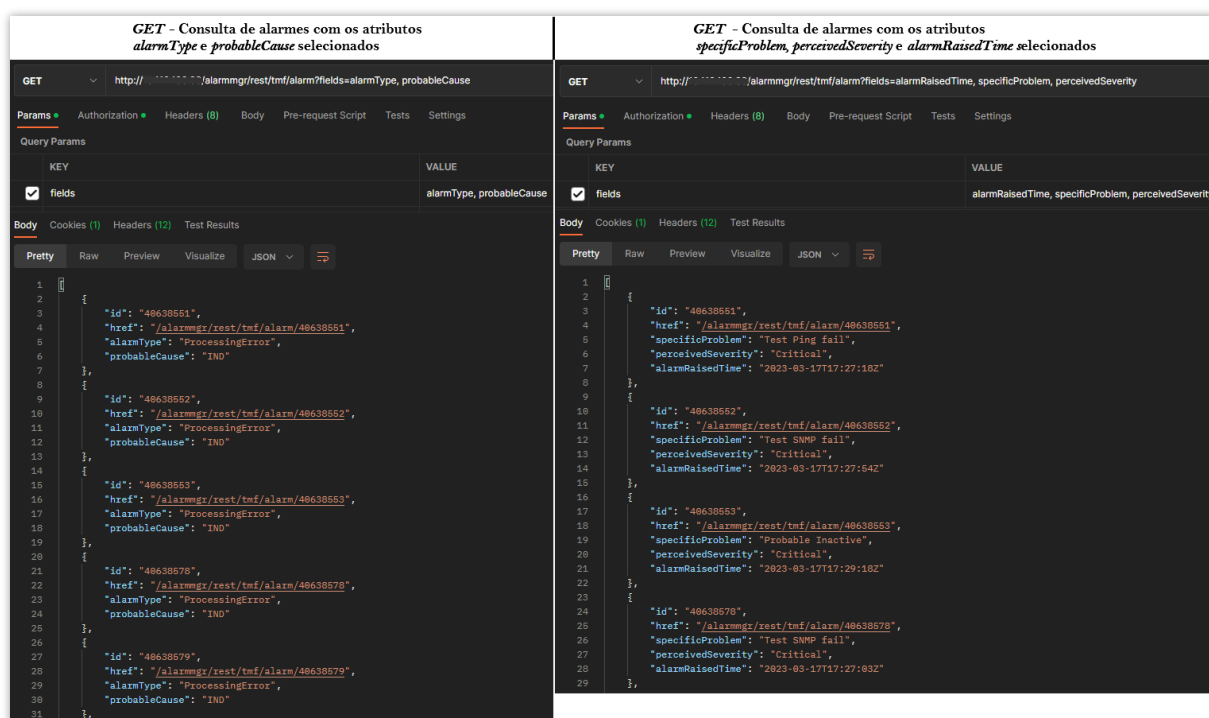


Figura 21: Teste manual do pedido *GET* – Consulta de alarmes com a funcionalidade de seleção de atributos (*select fields*).

O resultado final dos testes automáticos da funcionalidade de seleção pode ser observado no Apêndice V, em [Resultados CTK da funcionalidade de seleção](#), através do mesmo é possível verificar que todos os atributos especificados pela *TM Forum* (apresentado na Figura 19) foram incluídos nesta funcionalidade.

4.6 Implementação da funcionalidade de filtragem

A funcionalidade de filtragem permite obter alarmes de acordo com valores dos seus atributos. Por exemplo, obter os alarmes com o atributo *state* igual a “*Open*” ou um *sourceSystemId* igual a “*ems-1*”.

Inicialmente foi feita uma análise dos resultados esperados na *CTK*, para se averiguar a forma como a *TM Forum* pretendia que esta funcionalidade fosse implementada e os atributos que tinham de ser incluídos na mesma. O conteúdo analisado da *CTK* é apresentado na Figura 22.

/Alarm?alarmRaisedTime='{{ALARMRAISEDTIMEAL01}}'	
Description	This operation filter a Alarm
Method	GET
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?alarmRaisedTime='{{ALARMRAISEDTIMEAL01}}'
/Alarm?id={{IDAL01}}	
Description	This operation filter a Alarm
Method	GET
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?id={{IDAL01}}
/Alarm?probableCause={{PROBABLECAUSEAL01}}	
Description	This operation filter a Alarm
Method	GET
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?probableCause={{PROBABLECAUSEAL01}}
/Alarm?sourceSystemId={{SOURCESYSTEMIDAL01}}	
Description	This operation filter a Alarm
Method	GET
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?sourceSystemId={{SOURCESYSTEMIDAL01}}
/Alarm?state={{STATEAL01}}	
Description	This operation filter a Alarm
Method	GET
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?state={{STATEAL01}}

Figura 22: Identificação de atributos obrigatórios para a funcionalidade de filtragem.

De acordo com as especificações da *TMF642*, os atributos obrigatórios para a funcionalidade de filtragem são:

- *id*;
- *alarmRaisedTime*;
- *state*;
- *sourceSystemId*;
- *probableCause*.

Contrariamente à *API* já existente que envia os filtros no formato de uma *query*, a *TM Forum* pretendia que os mesmos fossem enviados através de um parâmetro opcional, no formato exemplificado na Figura 22. Para tal foi necessário acrescentar parâmetros opcionais ao pedido, nomeadamente um parâmetro para cada um dos atributos filtráveis. Estes parâmetros são convertidos internamente numa *query*, o formato consumível pelo código do sistema. Para se entender o funcionamento dos filtros do sistema e as conversões necessárias foi utilizado um decodificador online para passar os filtros do formato *json* (formato do sistema) para *URL encoded* (formato utilizado na especificação).

Na Figura 23 é possível observar o diagrama que ilustra o processamento dos filtros no sistema e a Tabela 5 explica a sua numeração. Neste caso é utilizado o exemplo de um filtro para o atributo *perceivedSeverity*.

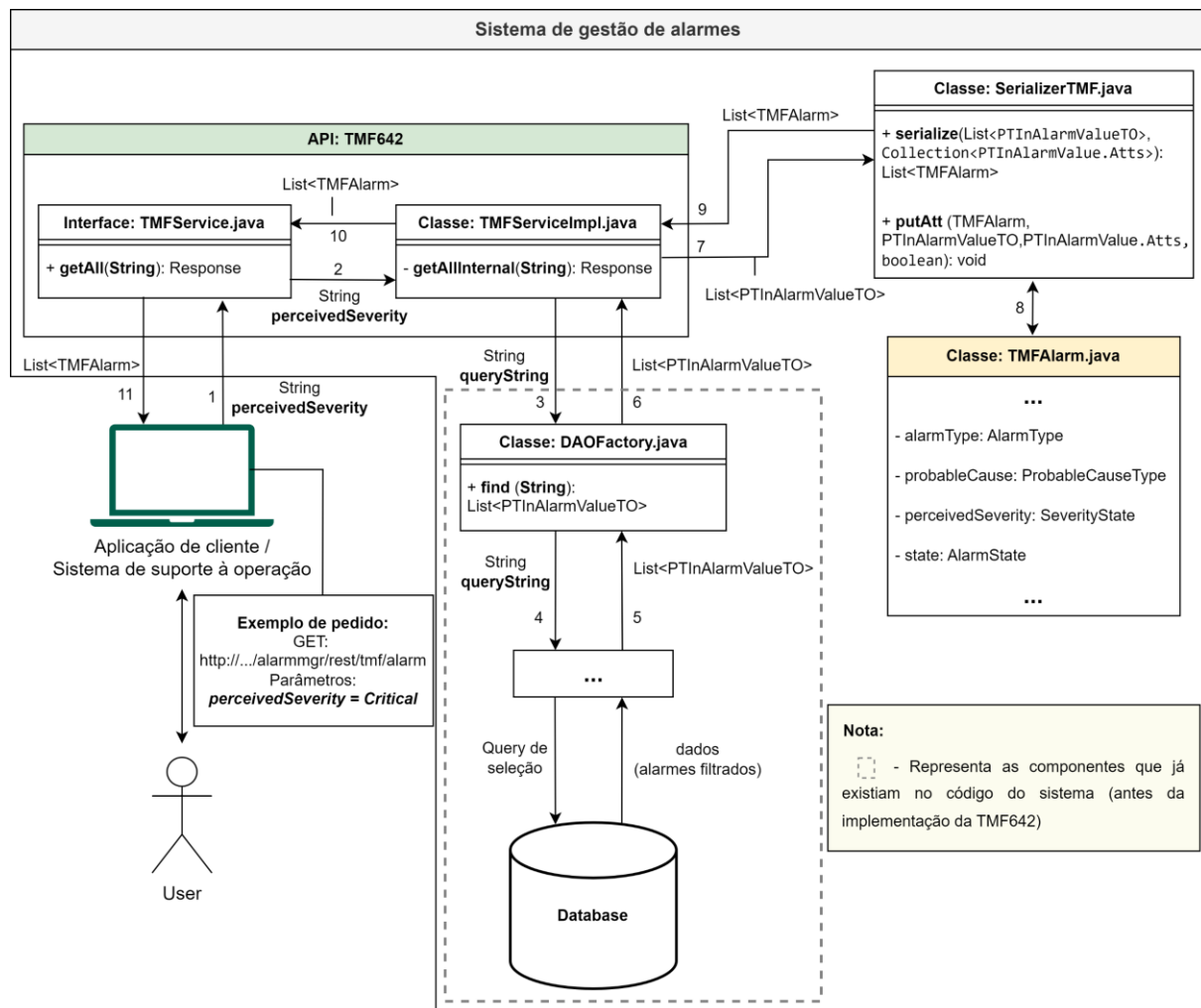


Figura 23: Diagrama de classes e de fluxo de informação do pedido *GET* – Consulta de alarmes com a funcionalidade de filtrar atributos.

Tabela 5: Descrição do diagrama do pedido *GET* – Consulta de alarmes com a funcionalidade de filtragem de atributos.

Nº	Descrição
1	Inicialmente o utilizador envia um pedido do tipo <i>GET</i> para o <i>endpoint</i> - <i>https://<server_ip>/alarmmgr/rest/tmf/alarm</i> e acrescenta aos parâmetros (<i>params</i>) do pedido o nome do atributo e o valor. Por exemplo: <i>https://<server_ip>/alarmmgr/rest/tmf/alarm?perceivedSeverity=critical</i>
2	A variável <i>state</i> recebida na interface é encaminhada para o método <i>getAllInternal()</i> , onde é convertida numa <i>query</i> de procura à base de dados.
3	O <i>getAllInternal()</i> invoca o método de procura dos alarmes <i>find()</i> e envia a <i>query</i> como parâmetro.
4	O método <i>find()</i> invoca um conjunto de classes que eventualmente utiliza a <i>query</i> para fazer a busca dos alarmes correspondentes na base de dados.
5, 6	Após a busca na base de dados, uma lista do tipo <i>List<PTInAlarmValue></i> que contém somente os alarmes cujo atributo " <i>perceivedSeverity</i> " tem valor igual a " <i>critical</i> " é devolvida ao método <i>getAllInternal()</i> ;
7, 8, 9	É realizado o preenchimento dos atributos do <i>TMFAlarm</i> e é criada a lista de alarmes <i>List<TMFAlarm></i> .
10, 11	A lista de alarmes correspondentes à procura é convertida para <i>JSON</i> e entregue na <i>Response</i> do pedido.

Após o desenvolvimento da funcionalidade de filtragem de atributos foi feita uma confirmação do seu funcionamento, através da execução de pedidos. Nestes verificou-se a presença exclusiva de alarmes com os valores inseridos. Alguns exemplos dos pedidos efetuados podem ser observados na Figura 24 e Figura 25.

Os resultados finais dos testes automáticos da funcionalidade de filtragem podem ser observados no Apêndice V, em [Resultados CTK da funcionalidade de filtragem](#), através dos mesmos é possível verificar que todos os atributos especificados pela *TM Forum* (apresentado na Figura 19) foram incluídos nesta funcionalidade.

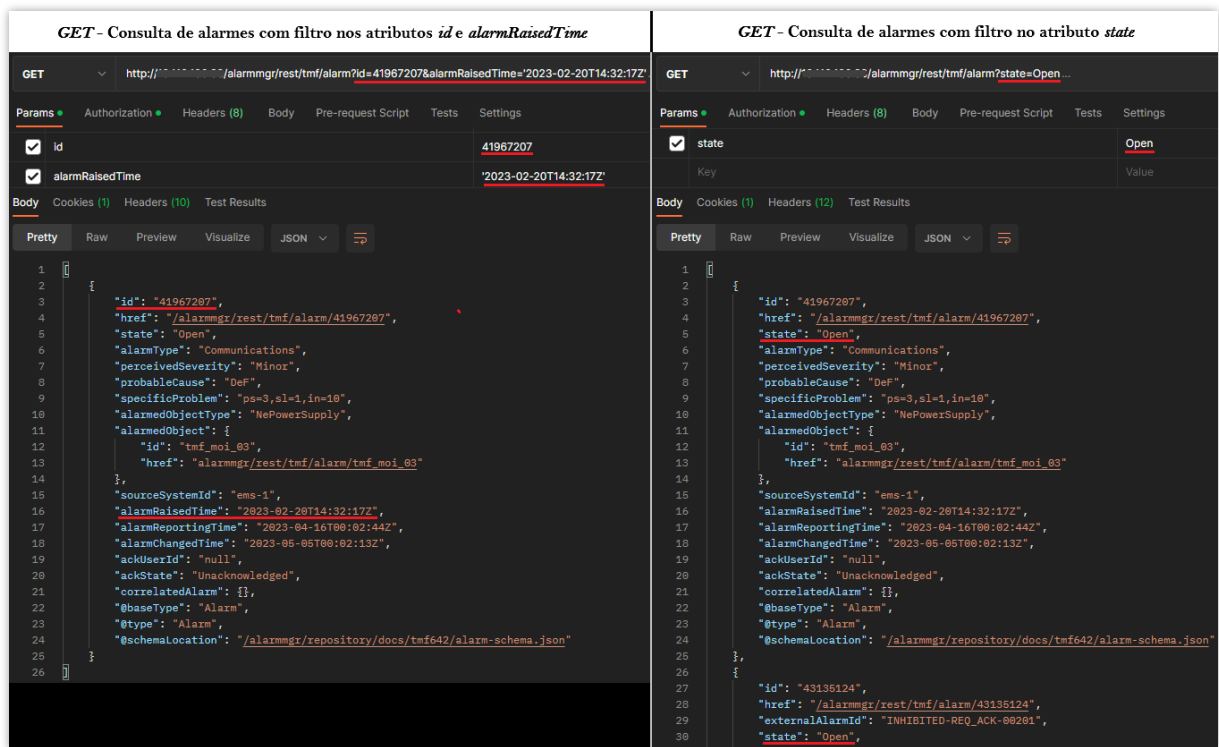


Figura 24: Teste manual do pedido *GET* – Consulta de alarmes com a funcionalidade de filtragem de atributos (*fields filter*).

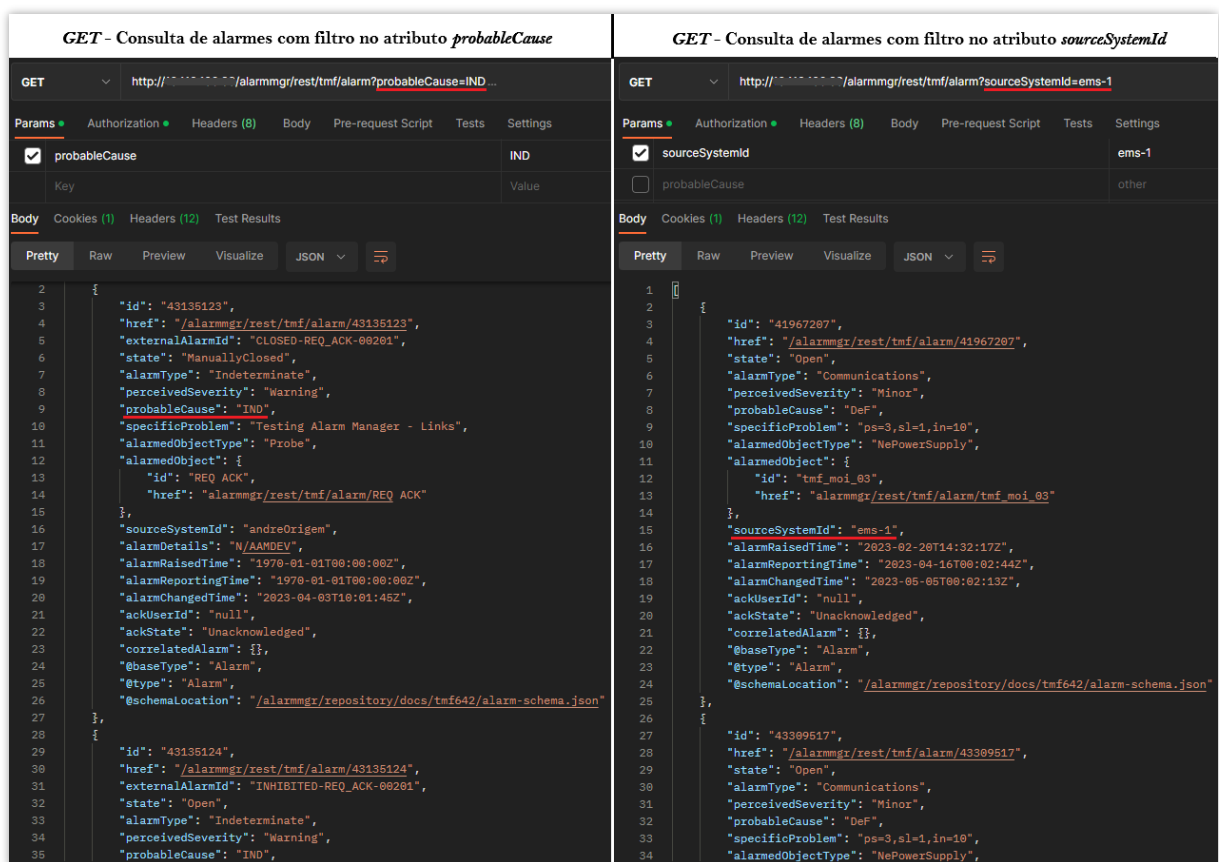


Figura 25: (Continuação) Teste manual do pedido *GET* – Consulta de alarmes com a funcionalidade de filtragem de atributos (*fields filter*).

4.7 Desenvolvimento do pedido *POST* – Criação de alarme

O pedido *POST* foi mais complexo de desenvolver, dado que foi necessário entender de que forma a *API* poderia criar alarmes de acordo com os requisitos de criação do sistema.

Inicialmente procedeu-se à identificação dos atributos obrigatórios para a criação de alarmes na especificação, representados na parte 1 da Tabela 6.

De seguida procedeu-se à identificação dos atributos obrigatórios do sistema, para tal foi necessário entender de que forma eram criados os alarmes internamente, através de análise do código. Através desta análise concluiu-se que o sistema associa duas chaves a um alarme no momento em que é criado:

- Uma chave de entidade (*managed_entity_key*) – com características relativas ao sistema de deteção do alarme que inclui os atributos obrigatórios:
 - *subsystem*;
 - *dom*;
 - *phost*;
 - *pdir*;
 - *moi*;
 - *lcode*;
 - *techn*.
- Uma chave de alarme (*alarm_detail_key*) – com características relativas aos detalhes do alarme que inclui os atributos obrigatórios:
 - *pcause*;
 - *type*;
 - *prob*.

Com base na identificação dos atributos obrigatórios tanto da especificação como do sistema construiu-se a Tabela 6, onde se juntaram todos os atributos obrigatórios para a criação de alarmes:

- Na parte 1 foram adicionados os atributos obrigatórios da especificação e foi feita a correspondência para os atributos existentes no sistema de gestão de alarmes (com ajuda da tabela do [Apêndice II](#));
- Na parte 2 foram adicionados os atributos obrigatórios exclusivos do sistema. Devido a nem todos existirem na especificação, os atributos em falta foram adicionados à mesma seguindo uma regra

de nomenclatura idêntica à convencionada pela *TM Forum* e tirando partido da sua possibilidade de extensão.

Tabela 6: Mapeamento de atributos obrigatórios para o pedido *POST* – Criação de alarme.

Criação de alarme	
Sistema de gestão de alarmes	API da <i>TM Forum</i> (<i>TMF642</i>)
1. Atributos obrigatórios do sistema e da <i>TMF642</i>	
<i>type</i>	<i>alarmType</i>
<i>sev</i>	<i>perceivedSeverity</i>
<i>pcause</i>	<i>probableCause</i>
<i>moiextid</i>	<i>alarmedObject</i>
<i>origin</i>	<i>sourceSystemId</i>
<i>st</i>	<i>state</i>
<i>raised</i>	<i>alarmRaisedTime</i>
2. Atributos obrigatórios exclusivos do sistema (adicionados à <i>TMF642</i>)	
<i>subsystem</i>	<i>subsystem</i>
<i>dom</i>	<i>domain</i>
<i>phost</i>	<i>probeHost</i>
<i>pdir</i>	<i>probeDirectory</i>
<i>lcode</i>	<i>localCode</i>
<i>moi</i>	<i>alarmedObject</i>
<i>tech</i>	<i>technology</i>
<i>prob</i>	<i>specificProblem</i>

O pedido de criação consiste em inserir um novo alarme na base de dados, e de seguida realizar uma busca pelo mesmo para ser devolvido na resposta do pedido (comprovativo de criação).

É possível observar a atualização da classe ***TMFAlarm.java*** (onde foram acrescentados os novos atributos obrigatórios) e a forma como o pedido foi implementado através do diagrama da Figura 26 e da Tabela 7 que explica a sua numeração.

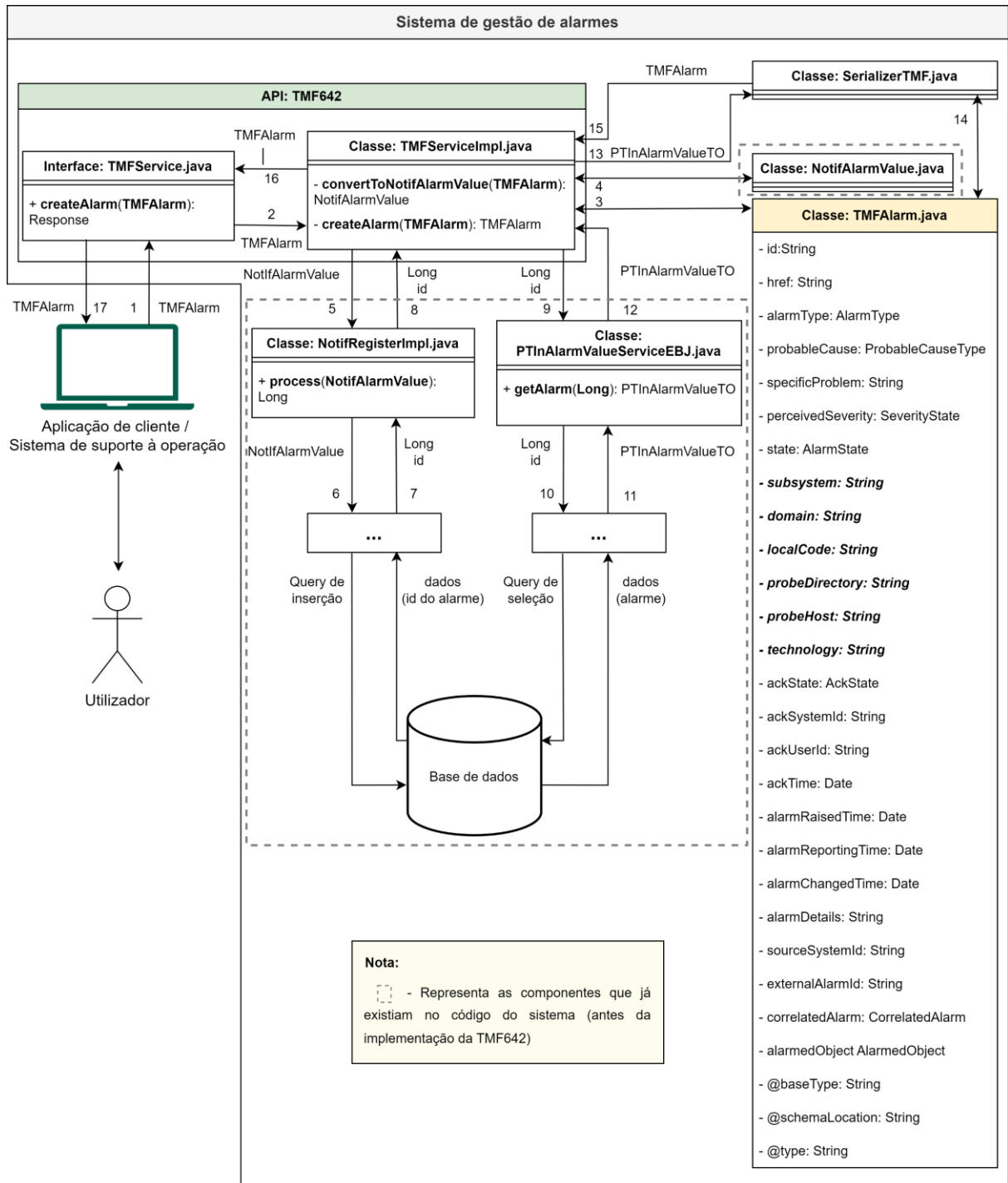


Figura 26: Diagrama de classes e de fluxo de informação do pedido *POST* – Criação de alarme.

Tabela 7: Descrição do diagrama do pedido *POST* – Criação de alarme.

Nº	Descrição
1	O utilizador envia um pedido do tipo <i>POST</i> para o <i>endpoint</i> – <i>https://<server_ip>/alarmmgr/rest/tmf/alarm</i> . No body do pedido inclui um objeto <i>json</i> com a estrutura da classe <i>TMFAAlarm.java</i> que contém os atributos obrigatórios

	apresentados na Tabela 7. Um exemplo de preenchimento do <i>body</i> é ilustrado na Figura 27.
2	O <i>TMFAlarm</i> recebido na interface é encaminhado para o método <i>createAlarm()</i> , responsável por fazer todas as invocações para a criação do alarme e eventual devolução do mesmo na resposta do pedido.
3, 4	Começa por invocar o método <i>convertToNotifAlarmValue()</i> que faz a conversão do alarme do tipo <i>TMFAlarm</i> num alarme que possa ser consumido pelos métodos de criação internos (<i>NotifAlarmValue.java</i>). Para isto utiliza os métodos <i>getters</i> da classe <i>TMFAlarm.java</i> (representado pelo número 3) e os <i>setters</i> da classe <i>NotifAlarmValue.java</i> (representado pelo número 4).
5	De seguida invoca o método <i>process()</i> , com o <i>NotifAlarmValue</i> como parâmetro. Este é o método interno responsável pela criação dos alarmes.
6	O <i>process()</i> faz algumas invocações a métodos de outras classes até eventualmente enviar uma <i>query</i> de inserção do alarme na base de dados.
7, 8	Após a inserção do alarme, é devolvido o <i>id</i> do alarme criado ao método <i>createAlarm()</i> .
9, 10	Este invoca o <i>getAlarm()</i> com o <i>id</i> do alarme como parâmetro, para fazer uma busca à base de dados pelo alarme criado.
11, 12	A base de dados devolve o alarme criado que é retornado à função <i>createAlarme()</i> no formato <i>PTInAlarmValue.java</i> .
13, 14, 15	Para que o alarme seja devolvido na <i>response</i> do pedido é novamente convertido para um alarme do tipo <i>TMFAlarm</i> , através dos métodos pertencentes à classe <i>SerializerTMF.java</i> , representados na figura do pedido <i>GET – Consulta de alarmes</i> (Figura 17, números 7, 8 e 9).
16, 17	Por fim, o alarme do tipo <i>TMFAlarm</i> é convertido para o formato <i>JSON</i> na classe <i>TMFServiceImpl.java</i> . O objeto é devolvido à interface e é enviado através da <i>Response</i> para uma aplicação de cliente ou outro sistema de suporte à operação.

Após o desenvolvimento do pedido foram executados testes manuais para verificar o seu alinhamento com as especificações da *TM Forum*. Através da Figura 27 é possível observar a comparação entre um pedido enviado a partir do *Postman* e o pedido exemplo fornecido na documentação.

Pedido da API em desenvolvimento	Exemplo de pedido do <i>user guide</i> da TMF642
<p>POST <code>http://10.10.10.10/alarmmgr/rest/tmf/alarm</code></p> <p>Params Authorization Headers (10) Body Pre-request Script Tests Settings</p> <p>none form-data x-www-form-urlencoded raw binary GraphQL JSON</p> <pre> 1 { 2 "alarmType": "Communications", 3 "alarmedObject": { 4 "id": "1234" 5 }, 6 "perceivedSeverity": "Warning", 7 "probableCause": "CaSF", 8 "sourceSystemId": "ems-1", 9 "state": "Open", 10 "alarmedObjectType": "NePowerSupply", 11 "specificProblem": "tmf specific description 65", 12 "domain": "AMDEV", 13 "subsystem": "tmf_sub", 14 "localCode": "tmf_195", 15 "probeDirectory": "tmf_dir", 16 "probeHost": "tmf_host", 17 "technology": "SYS" 18 } </pre>	<p>POST /tmf-api/alarmManagement/v4/alarm Content-Type: application/json</p> <pre> { "externalAlarmId": "5551212", "state": "raised", "alarmType": "environmentalAlarm", "perceivedSeverity": "minor", "probableCause": "rectifierLowVoltage", "specificProblem": "ps=3,sl=1,in=8", "alarmedObjectType": "NePowerSupply", "alarmedObject": { "id": "93051825", "href": "/resourceInventoryManagement/v4/resource/93051825" }, "sourceSystemId": "ems-1", "alarmDetails": "voltage=204", "alarmRaisedTime": "2019-07-03T03:32:17.235Z", "alarmReportingTime": "2019-07-03T03:32:17.552Z" } </pre>
<p>Body Cookies (1) Headers (10) Test Results</p> <p>Pretty Raw Preview Visualize JSON</p> <pre> 1 { 2 "id": "43337021", 3 "href": "/alarmmgr/rest/tmf/alarm/43337021", 4 "state": "Open", 5 "alarmType": "Communications", 6 "perceivedSeverity": "Warning", 7 "probableCause": "CaSF", 8 "specificProblem": "tmf specific description 65", 9 "subsystem": "tmf_sub", 10 "domain": "AMDEV", 11 "localCode": "tmf_195", 12 "probeDirectory": "tmf_dir", 13 "probeHost": "tmf_host", 14 "technology": "SYS", 15 "alarmedObjectType": "NePowerSupply", 16 "alarmedObject": { 17 "id": "1234", 18 "href": "alarmmgr/rest/tmf/alarm/1234" 19 }, 20 "sourceSystemId": "ems-1", 21 "alarmRaisedTime": "2023-05-08T16:40:56Z", 22 "alarmReportingTime": "2023-05-08T16:40:56Z", 23 "alarmChangedTime": "2023-05-08T16:40:57Z", 24 "ackUserId": "null", 25 "ackState": "Unacknowledged", 26 "correlatedAlarm": {}, 27 "@baseType": "Alarm", 28 "@type": "Alarm", 29 "@schemaLocation": "/alarmmgr/repository/docs/tmf642/alarm-schema.json" 30 } </pre>	<p>Response</p> <p>201</p> <pre> { "id": "8675309", "href": "/alarmManagement/v4/alarm/8675309", "externalAlarmId": "5551212", "state": "raised", "alarmType": "environmentalAlarm", "perceivedSeverity": "minor", "probableCause": "rectifierLowVoltage", "specificProblem": "ps=3,sl=1,in=8", "alarmedObjectType": "Rectifier", "alarmedObject": { "id": "93051825", "href": "/resourceInventoryManagement/v4/resource/93051825" }, "sourceSystemId": "ems-1", "alarmDetails": "voltage=204", "alarmRaisedTime": "2019-07-03T03:32:17.235Z", "alarmReportingTime": "2019-07-03T03:32:17.552Z" } </pre>

Figura 27: Comparação entre as respostas do pedido *POST* – Criação de alarme.

Para efetuar o teste do pedido de criação de alarme através da *CTK*, foi necessário proceder à edição do ficheiro de configurações (*config.json*), nomeadamente o *body/payload* do pedido, para que o mesmo incluísse os atributos obrigatórios de criação (mencionados na Tabela 6). A Figura 28 ilustra essa edição.

```

{
  "url": " http://[REDACTED]/alarmmgr/rest/tmf/",
  "headers": {
    "Accept": "application/json",
    "Content-Type": "application/json",
    "Authorization": "Basic [REDACTED]"
  },
  "payloads": {
    "Alarm": {
      "POST": {
        "payload": {
          "state": "Open",
          "alarmType": "Environmental",
          "perceivedSeverity": "Minor",
          "probableCause": "RLFV",
          "specificProblem": "ps=3,s1=1,in=19",
          "alarmedObjectType": "NePowerSupply",
          "alarmedObject": {
            "id": "tmf_moi_05"
          },
          "alarmRaisedTime": "2023-02-20T16:00:00Z"
        },
        "sourceSystemId": "ems-1",
        "subsystem": "tmf_sub",
        "domain": "AMDEV",
        "localCode": "tmf_35",
        "alarmDetails": "summary_example",
        "probeDirectory": "tmf_dir",
        "probeHost": "tmf_host",
        "technology": "SYS"
      }
    }
  }
}

```

Figura 28: Configuração do ficheiro *config.json* para teste automático do pedido *POST* – Criação de alarme.

O resultado final dos testes automáticos do pedido de criação pode ser observado no Apêndice V, em [Resultados CTK do pedido POST – Criação de alarme](#), onde se verifica que além do alarme ter sido criado com sucesso, a resposta do pedido contém todos os atributos obrigatórios especificados pela *TM Forum*.

4.8 Desenvolvimento do pedido *PATCH* – Alteração de um alarme

Inicialmente foi feita uma análise dos resultados esperados na *CTK*, para se verificar os atributos obrigatórios de implementar com a funcionalidade de alteração. Os mesmos são apresentados na Figura 29.

Patch Alarm <u>probableCause</u>	
Method	PATCH
URL	http://[IP]:[PORT]/alarmmgr/rest/tmf/alarm/{{IDAL01}}
Patch Alarm <u>perceivedSeverity</u>	
Method	PATCH
URL	http://[IP]:[PORT]/alarmmgr/rest/tmf/alarm/{{IDAL01}}
Patch Alarm <u>alarmType</u>	
Method	PATCH
URL	http://[IP]:[PORT]/alarmmgr/rest/tmf/alarm/{{IDAL01}}
Patch Alarm <u>state</u>	
Method	PATCH
URL	http://[IP]:[PORT]/alarmmgr/rest/tmf/alarm/{{IDAL01}}

Figura 29: Identificação de atributos obrigatórios do pedido *PATCH* – Alteração de alarme.

O desenvolvimento do pedido *PATCH* – Alteração de alarme foi baseado no pedido *POST* – Criação de alarme, uma vez que o sistema utiliza o mesmo método interno para efetuar as operações de criação e de alteração de alarme. A distinção entre estas duas operações é feita através das chaves de alarme introduzidas no pedido de criação.

De acordo com as regras do sistema, para fazer a alteração de um alarme, é necessário que as chaves (*managed_entity_key* e *alarm_detail_key*) se mantenham iguais, ou seja, os atributos pertencentes às chaves nunca podem ser alterados, caso contrário é criado um alarme em vez de proceder à alteração do mesmo. Na Figura 30 é apresentado um diagrama que explica de que forma os alarmes são criados ou alterados no sistema.

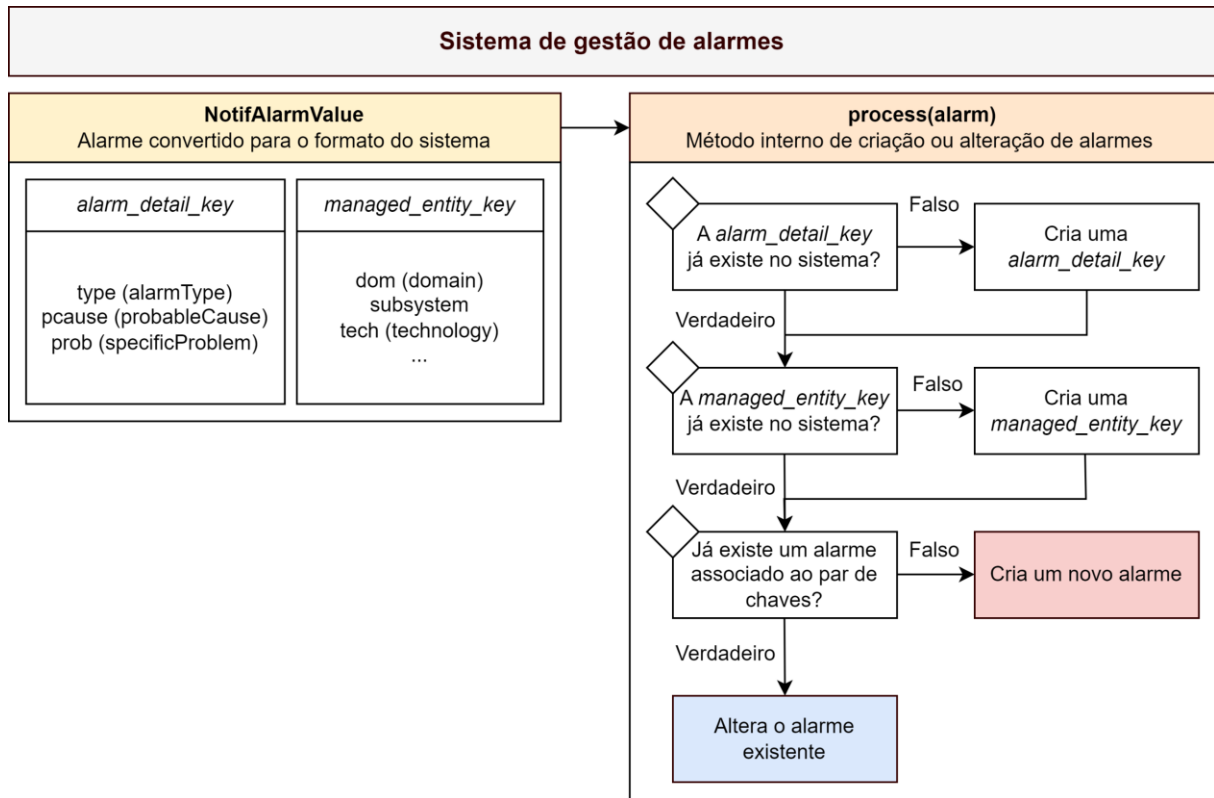


Figura 30: Distinção entre a criação e alteração de alarmes no sistema

Devido à especificação da *TM Forum* indicar os atributos *alarmType* e *probableCause* como alteráveis (correspondentes aos atributos *type* e *pcause* incluídos na *alarm_detail_key* do sistema), foi necessário alterar esta chave do alarme para que a mesma não incluisse estes atributos, passando somente a incluir o atributo *prob*, correspondente ao *specificProblem* da *TMF642*.

De acordo com a especificação e com a análise dos testes de conformidade, foi implementada a possibilidade de alterar os atributos:

- *probableCause*;
- *perceivedSeverity*;
- *alarmType*;
- *state*.

As classes relacionadas com o pedido de alteração de alarme e a forma como foi ligado ao código do *sistema* podem ser observadas através do diagrama da Figura 31, cuja numeração é explicada na Tabela 8. Na figura é utilizado um exemplo de alteração do atributo *perceivedSeverity*.

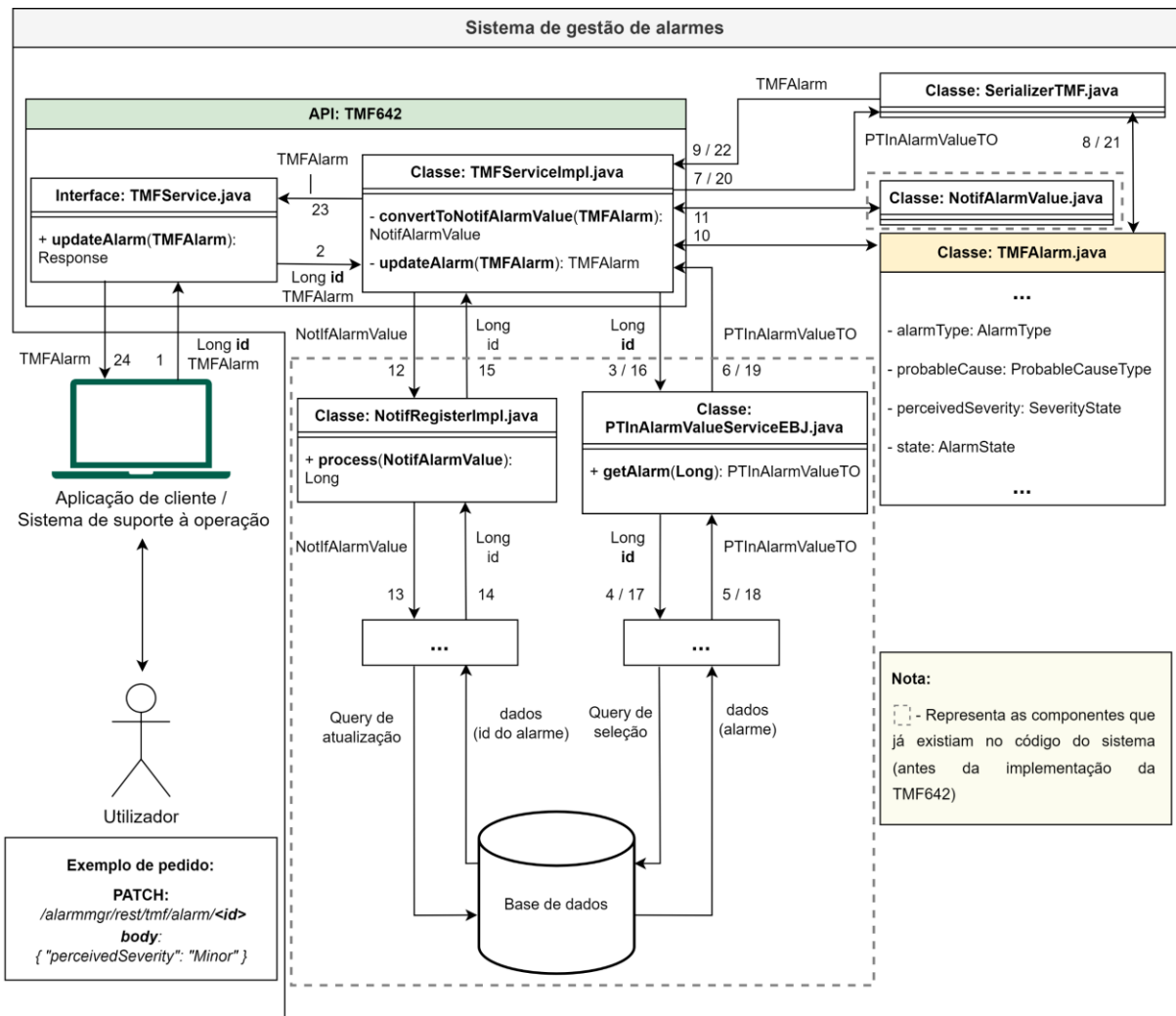


Figura 31: Diagrama de classes e de fluxo de informação do pedido *PATCH* – Alteração de alarme.

Tabela 8: Descrição do diagrama do pedido *PATCH* – Alteração de alarme.

Nº	Descrição
1	O utilizador envia um pedido do tipo <i>PATCH</i> para o <i>endpoint</i> – <i>https://<server_ip>/alarmmgr/rest/tmf/alarm/<id></i> . No body do pedido inclui um objeto <i>json</i> com a estrutura da classe <i>TMFAlarm.java</i> que tem de conter o(s) atributo(s) que pretende alterar (estes têm de constar na lista de atributos possíveis de alterar, representada pela Figura 29). Um exemplo de preenchimento do body é ilustrado na Figura 32.
2	O <i>TMFAlarm</i> recebido na interface é encaminhado para o método <i>updateAlarm()</i> , responsável por fazer todas as invocações para a alteração do alarme e eventual devolução do mesmo na resposta do pedido.

3, 4	Inicialmente é feita uma procura do alarme à base de dados. Para tal, o <i>updateAlarm()</i> invoca o método <i>getAlarm()</i> com o <i>id</i> do alarme como parâmetro que por sua vez invoca uma série de métodos até enviar uma <i>query</i> de procura à base de dados.
5, 6	A base de dados devolve o alarme procurado ao <i>updateAlarm()</i> .
7, 8, 9	O <i>updateAlarm()</i> altera o(s) valor(es) do(s) atributo(s) do <i>PTInAlarmValue</i> que o utilizador enviou no <i>body</i> no pedido, através do <i>setter</i> do atributo. De seguida faz a conversão do alarme do tipo <i>PTInAlarmValue</i> no tipo <i>TMFAlarm</i> recorrendo aos métodos da classe <i>Serializer.java</i> (representadas na Figura 15, Tabela 2).
10, 11	De seguida é invocado o método <i>convertToNotifAlarmValue()</i> para converter o alarme do formato <i>TMFAlarm.java</i> no formato <i>NotifAlarmValue.java</i> (formato compatível com método interno de processamento de alarmes).
12, 13	O <i>updateAlarm()</i> invoca a <i>process()</i> com o <i>NotifAlarmValue</i> como parâmetro. Neste método é feita a verificação da existência de um alarme associado às chaves (<i>alarm_detail_key</i> e <i>managed_entity_key</i>), caso se verifique, procede à substituição do alarme existente por um novo alarme com os valores enviados no pedido (operação de <i>UPDATE</i>).
14, 15	O <i>id</i> do alarme criado que é devolvido ao <i>updateAlarmValue()</i> .
16, 17	O <i>id</i> recebido é utilizado para fazer um último acesso à base de dados, este é enviado como parâmetro para o método <i>getAlarm()</i> que invoca outros métodos até que seja enviada uma <i>query</i> de busca pelo alarme alterado à base de dados.
18, 19	O alarme alterado é devolvido ao <i>updateAlarm()</i> no formato <i>PTInAlarmValue</i> .
20, 21, 22	O <i>updateAlarm()</i> faz a conversão do <i>PTInAlarmValue</i> num alarme do tipo <i>TMFAlarm</i> através das funções da classe <i>Serializer.java</i> . E de seguida converte o alarme para o formato <i>JSON</i> .
23, 24	Por fim, o objeto é devolvido à interface <i>TMFService.java</i> que por sua vez devolve-o na <i>Response</i> à aplicação de cliente ou a outro sistema de suporte à operação.

Após o desenvolvimento do pedido foram executados testes manuais para verificar a sua conformidade com as especificações da *TM Forum*. Através da Figura 32 é possível observar a comparação entre um pedido enviado a partir do *Postman* e o pedido exemplo fornecido pela documentação.

Pedido da <i>API</i> em desenvolvimento	Exemplo de pedido do <i>user guide</i> da <i>TMF642</i>
<p>PATCH <code>http://10.10.10.10:8080/alarmmgr/rest/tmf/alarm/43337635</code></p> <p>Params Authorization Headers (10) Body Pre-request Script Tests Settings</p> <p>none form-data x-www-form-urlencoded raw binary GraphQL JSON</p> <pre> 1 { 2 "probableCause": "BaF", 3 "perceivedSeverity": "Minor", 4 "state": "Open" 5 } </pre>	<p>PATCH /tmf-api/alarmManagement/v4/alarm/8675309 Content-Type: application/merge-patch+json</p> <pre> { "state": "updated", "perceivedSeverity": "major", "alarmDetails": "voltage=95", "alarmChangedTime": "2019-07-03T03:32:24.715Z" } </pre>
<p>Body Cookies (1) Headers (10) Test Results</p> <p>Pretty Raw Preview Visualize JSON</p> <pre> 1 { 2 "id": "43337681", 3 "href": "/alarmmgr/rest/tmf/alarm/43337681", 4 "state": "Open", 5 "alarmType": "Communications", 6 "perceivedSeverity": "Minor", 7 "probableCause": "BaF", 8 "specificProblem": "tmf specific description 66", 9 "subsystem": "tmf_sub", 10 "domain": "AMDEV", 11 "localCode": "tmf_196", 12 "probeDirectory": "tmf_dir", 13 "probeHost": "tmf_host", 14 "technology": "SVS", 15 "alarmedObjectType": "NePowerSupply", 16 "alarmedObject": { 17 "id": "1234", 18 "href": "/alarmmgr/rest/tmf/alarm/1234" 19 }, 20 "sourceSystemId": "ems-1", 21 "alarmRaisedTime": "2023-05-08T17:46:02Z", 22 "alarmReportingTime": "2023-05-08T17:47:20Z", 23 "alarmChangedTime": "2023-05-08T17:47:20Z", 24 "ackUserId": "null", 25 "ackState": "Unacknowledged", 26 "correlatedAlarm": {}, 27 "@baseType": "Alarm", 28 "@type": "Alarm", 29 "@schemaLocation": "/alarmmgr/repository/docs/tmf642/alarm-schema.json" 30 } </pre>	<p>Response</p> <p>200</p> <pre> { "id": "8675309", "href": "/alarmManagement/v4/alarm/8675309", "externalAlarmId": "5551212", "state": "updated", "alarmType": "environmentalAlarm", "perceivedSeverity": "major", "probableCause": "rectifierLowVoltage", "specificProblem": "ps=3,sl=1,in=8", "alarmedObjectType": "Rectifier", "alarmedObject": { "id": "93051825", "href": "/resourceInventoryManagement/v4/resource/93051825" }, "sourceSystemId": "ems-1", "alarmDetails": "voltage=95", "alarmRaisedTime": "2019-07-03T03:32:17.235Z", "alarmReportingTime": "2019-07-03T03:32:17.552Z", "alarmChangedTime": "2019-07-03T03:32:24.715Z" } </pre>

Figura 32: Comparação entre as respostas do pedido *PATCH* – Alteração de alarme.

Os resultados finais dos testes *CTK* do pedido de alteração de alarme podem ser observados no Apêndice V, em [Resultados *CTK* do pedido *PATCH* – Alteração de alarme](#), onde é possível verificar que todos os atributos obrigatórios mencionados pela *TM Forum* (apresentados na Figura 29) são possíveis de alterar.

4.9 Implementação de valores enumerados

A última fase de desenvolvimento da *API TMF642* foi a de implementação de valores enumerados. Os atributos que possuem valores enumerados são:

- *perceivedSeverity*;
- *state*;
- *alarmType*;
- *probableCause*.

Para cada um dos atributos foi feita uma comparação entre:

- Os valores enumerados apresentados na documentação da *TM Forum (TMF642 user guide)*;
- Os valores enumerados existentes no *sistema*.

A comparação entre os valores enumerados permitiu identificar alguns problemas de correspondência entre o sistema e a especificação. Através da tabela do [Apêndice III](#) é possível verificar os problemas de compatibilidade encontrados nos diferentes atributos e na Tabela 9 é possível verificar uma descrição detalhada dos mesmos.

Para contornar este problema foi feita uma consulta manual às principais normas de gestão com que a *TM Forum* afirma estar alinhada, nomeadamente a *ITU-T X.733* e a *3GPP TS 32.111-2*. Através desta consulta foi possível verificar que apesar do sistema e da especificação possuírem diferenças todos os valores eram válidos. Para se contornar este obstáculo, consultou-se a opinião da equipa de desenvolvimento e adotou-se as soluções apresentadas na Tabela 9.

Tabela 9: Identificação de problemas e respetivas soluções para o mapeamento de valores enumerados.

Atributo	Problema	Solução
state	A <i>TMF642</i> possui um <i>state</i> com valor " <i>updated</i> " que não tem correspondência no sistema. Por outro lado, o sistema possui um <i>state</i> " <i>manuallyClosed</i> " que não tem correspondência do lado <i>TMF642</i> .	O <i>state</i> " <i>updated</i> " será ignorado e só será possível os alarmes mudarem do <i>state</i> " <i>raised</i> " para " <i>cleared</i> " e vice-versa. O <i>state</i> " <i>manuallyClosed</i> " será mapeado, tal como o " <i>closed</i> " para o valor " <i>cleared</i> " da <i>TMF642</i> .

alarmType	<p>A <i>TM Forum</i> separou os <i>alarmTypes</i> “mechanismViolation” e “securityService” quando no sistema só existe um valor com ambos juntos e na norma <i>3GPP TS 32.111-2 Anex A</i> estes são igualmente classificados dentro do mesmo <i>Event/Alarm Type</i>: “Mechanism violation or security service”.</p>	<p>Com base na informação retirada da norma <i>3GPP TS 32.111-2 Anex A</i>, os <i>alarmTypes</i> foram unidos no mesmo tipo de alarme e a nomenclatura atribuída foi: “mechanismViolation OrSecurityService”.</p>
	<p>A <i>TM Forum</i> não especificou um <i>alarmType</i> de tipo indefinido/desconhecido, para ocasiões em que o mesmo não é identificado, no entanto, no sistema existia o <i>alarmType</i> “Indeterminate” e na norma <i>3GPP TS 32.111-2 Anex B</i> é feita uma referência a um tipo de alarme “unknown”.</p>	<p>Com base na informação retirada da norma <i>3GPP TS 32.111-2 Anex B</i> é adicionado o valor “unknown” ao enumerado de <i>alarmTypes</i> da <i>TMF642</i> que mapeia para o “Indeterminate” do sistema.</p>
probableCause	<p>A <i>TM Forum</i> possuía vários atributos que não possuíam correspondência no enumerado do sistema, no entanto existiam na norma <i>3GPP TS 32.111-2 Anex B</i>.</p>	<p>As <i>probableCauses</i> inexistentes no sistema foram adicionadas, seguindo a regra de nomenclatura estabelecida pela equipa de desenvolvimento.</p>
	<p>Por outro lado, o sistema possuía alguns atributos que não tinham correspondência no enumerado da <i>TMF642</i>, no entanto existiam na norma <i>3GPP TS 32.111-2 Anex B</i>.</p>	<p>As <i>probableCauses</i> inexistentes na <i>TMF642</i> foram adicionadas ao enumerado de conversão, seguindo uma nomenclatura idêntica à apresentada na especificação.</p>

As tabelas dos atributos com valores enumerados que possuíam problemas (*state*, *alarmType* e *probableCause*) foram atualizadas de acordo com as soluções apresentadas na Tabela 9 e podem ser visualizadas no [Apêndice IV](#).

Com base nas tabelas de mapeamento foi criada uma nova classe de conversão de valores enumerados de acordo com as especificações da *TM Forum*, a classe ***TMFAlarmEnums.java***. No interior desta classe foram criados os enumerados com métodos de conversão para transformar os valores do sistema no formato da *TMF642*, nomeadamente:

- *TMFSate*;
- *TMFPerceivedSeverity*;
- *TMFAlarmType*;
- *TMFProbableCause*.

Além disso foram atualizados os tipos de dados recebidos nos atributos enumerados da classe ***TMFAlarm.java***, apresentados no interior desta classe na Figura 33.

É possível observar de que forma foi criada a conversão de valores e a sua ligação com o restante código do sistema através da Figura 33 e da Tabela 10 que descreve a numeração da figura. Neste caso foi acrescentado código à classe de serialização (***SerializerTMF.java***) para que os atributos com valores enumerados fossem mapeados para os valores correspondentes da *TMF642*.

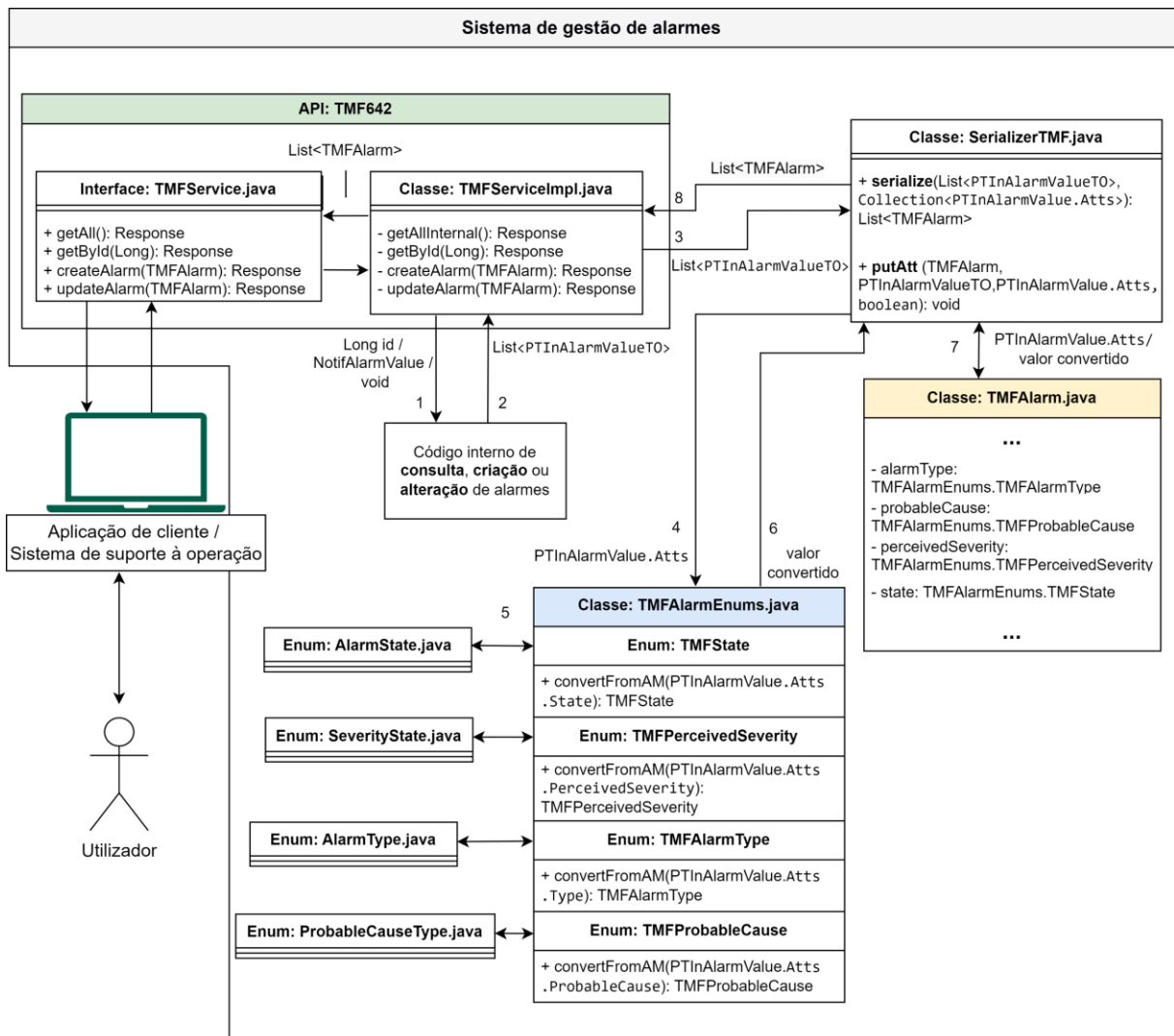


Figura 33: Diagrama de classes e de fluxo de informação do mapeamento de valores enumerados.

Tabela 10: Descrição do diagrama de mapeamento de valores enumerados.

Nº	Descrição
1	<p>Inicialmente o utilizador envia um dos pedidos disponíveis na <i>API TMF642</i>:</p> <p><i>GET</i> – Consulta de alarmes;</p> <p><i>GET</i> – Consulta de alarme pelo <i>id</i>;</p> <p><i>POST</i> – Criação de alarme;</p> <p><i>PATCH</i> – Alteração de alarme.</p> <p>Qualquer um dos pedidos envolve um acesso inicial ao código interno de consulta, criação ou alteração de alarmes.</p>
2	<p>Após este acesso ocorre uma devolução de um ou vários alarmes no formato <i>PTInAlarmValueTO</i>.</p>

3	Os alarmes recebidos na classe <i>TMFServiceImpl.java</i> no formato <i>PTInAlarmValueTO</i> têm de ser convertidos no formato <i>TMFAlarm</i> para serem devolvidos na <i>Response</i> do pedido, no formato da <i>TMF642</i> . Para tal são invocados os métodos da classe <i>SerializerTMF.java</i> (explicados com mais detalhe na Tabela 2).
4	O método <i>putAtt()</i> que preenche os atributos de cada alarme, sempre que tem de popular um atributo com valor enumerado passa a invocar os métodos de conversão existentes da classe <i>TMFAlarmEnums.java</i> . Por exemplo: para popular o atributo <i>alarmType</i> de um alarme recorre à função de conversão <i>convertFromAM()</i> existente no enumerado <i>TMFAlarmType</i> .
5	O método de conversão existente em cada um dos enumerados da classe <i>TMFAlarmEnums.java</i> procura a correspondência do valor recebido no formato do sistema e substitui esse valor pelo valor do enumerado. Por exemplo no caso do <i>alarmType</i> recebido ter valor “ <i>Environmental</i> ”, este valor é substituído por “ <i>environmentalAlarm</i> ” (formato compatível da <i>TMF642</i>).
6, 7	O valor substituído é devolvido à função <i>putAtt()</i> que atribui o valor ao respetivo atributo.
8	A lista de alarmes no formato <i>TMFAlarm</i> é devolvida ao método da classe <i>TMFServiceImpl.java</i> que por sua vez converte-a para o formato JSON devolve a mesma na <i>Response</i> do pedido.

Após a implementação dos valores enumerados foram executados testes manuais para verificar a sua conformidade com a especificação. Através da Figura 34 é possível observar a comparação entre um pedido enviado a partir do *Postman* e o pedido exemplo fornecido pela documentação. Para exemplificar a implementação dos valores enumerados foi utilizado o pedido de criação de alarme.

Pedido da API em desenvolvimento	Exemplo de pedido do <i>user guide</i> da TMF642
POST <code>http://10.10.10.10/alarmmgr/rest/tmf/alarm</code>	POST /tmf-api/alarmManagement/v4/alarm Content-Type: application/json
Params ● Auth ● Headers (10) Body ● Pre-req. Tests Settings raw JSON <pre> 1 { 2 "state": "raised", 3 "alarmType": "environmentalAlarm", 4 "perceivedSeverity": "minor", 5 "probableCause": "rectifierLowVoltage", 6 "specificProblem": "ps=3,sl=5,in=8", 7 "alarmedObjectType": "NePowerSupply", 8 "alarmedObject": { 9 "id": "93051825" 10 }, 11 "sourceSystemId": "ems-1", 12 "alarmDetails": "voltage=204", 13 "alarmRaisedTime": "2019-07-03T03:32:17.235Z", 14 "domain": "AMDEV", 15 "subsystem": "tmf_sub", 16 "localCode": "tmf_197", 17 "probeDirectory": "tmf_dir", 18 "probeHost": "tmf_host", 19 "technology": "SYS" 20 } </pre>	<pre> { "externalAlarmId": "5551212", "state": "raised", "alarmType": "environmentalAlarm", "perceivedSeverity": "minor", "probableCause": "rectifierLowVoltage", "specificProblem": "ps=3,sl=1,in=8", "alarmedObjectType": "NePowerSupply", "alarmedObject": { "id": "93051825", "href": "/resourceInventoryManagement/v4/resource/93051825" }, "sourceSystemId": "ems-1", "alarmDetails": "voltage=204", "alarmRaisedTime": "2019-07-03T03:32:17.235Z", "alarmReportingTime": "2019-07-03T03:32:17.552Z" } </pre>
Body 201 Created 1987 ms 1.26 KB Pretty Raw Preview Visualize JSON <pre> 1 { 2 "id": "43338892", 3 "href": "/alarmmgr/rest/tmf/alarm/43338892", 4 "state": "raised", 5 "alarmType": "environmentalAlarm", 6 "perceivedSeverity": "minor", 7 "probableCause": "rectifierLowVoltage", 8 "specificProblem": "ps=3,sl=5,in=8", 9 "subsystem": "tmf_sub", 10 "domain": "AMDEV", 11 "localCode": "tmf_197", 12 "probeDirectory": "tmf_dir", 13 "probeHost": "tmf_host", 14 "technology": "SYS", 15 "alarmedObjectType": "NePowerSupply", 16 "alarmedObject": { 17 "id": "93051825", 18 "href": "alarmmgr/rest/tmf/alarm/93051825" 19 }, 20 "sourceSystemId": "ems-1", 21 "alarmDetails": "voltage=204", 22 "alarmRaisedTime": "2019-07-03T03:32:17Z", 23 "alarmReportingTime": "2019-07-03T03:32:17Z", 24 "alarmChangedTime": "2023-05-08T19:43:19Z", 25 "ackUserId": "null", 26 "ackState": "Unacknowledged", 27 "correlatedAlarm": {}, 28 "@baseType": "Alarm", 29 "@type": "Alarm", 30 "@schemaLocation": "alarmmgr/repository/docs/tmf642/alarm-schema.json" 31 } </pre>	Response 201 <pre> { "id": "8675309", "href": "/alarmManagement/v4/alarm/8675309", "externalAlarmId": "5551212", "state": "raised", "alarmType": "environmentalAlarm", "perceivedSeverity": "minor", "probableCause": "rectifierLowVoltage", "specificProblem": "ps=3,sl=1,in=8", "alarmedObjectType": "Rectifier", "alarmedObject": { "id": "93051825", "href": "/resourceInventoryManagement/v4/resource/93051825" }, "sourceSystemId": "ems-1", "alarmDetails": "voltage=204", "alarmRaisedTime": "2019-07-03T03:32:17.235Z", "alarmReportingTime": "2019-07-03T03:32:17.552Z" } </pre>

Figura 34: Comparação entre as respostas do pedido *POST* – Criação de alarme com os valores enumerados alinhados com a *TMF642*.

Após a verificação de conformidade através dos testes manuais procedeu-se à execução dos testes *CTK*. Para tal foi necessário fazer uma última configuração ao ficheiro de configurações (*config.json*), no sentido de atualizar os valores enumerados enviados no *body/payload* do pedido de acordo com as alterações implementadas. A edição do ficheiro pode ser observada na Figura 35, onde do lado direito consta o ficheiro com os valores enumerados atualizados.

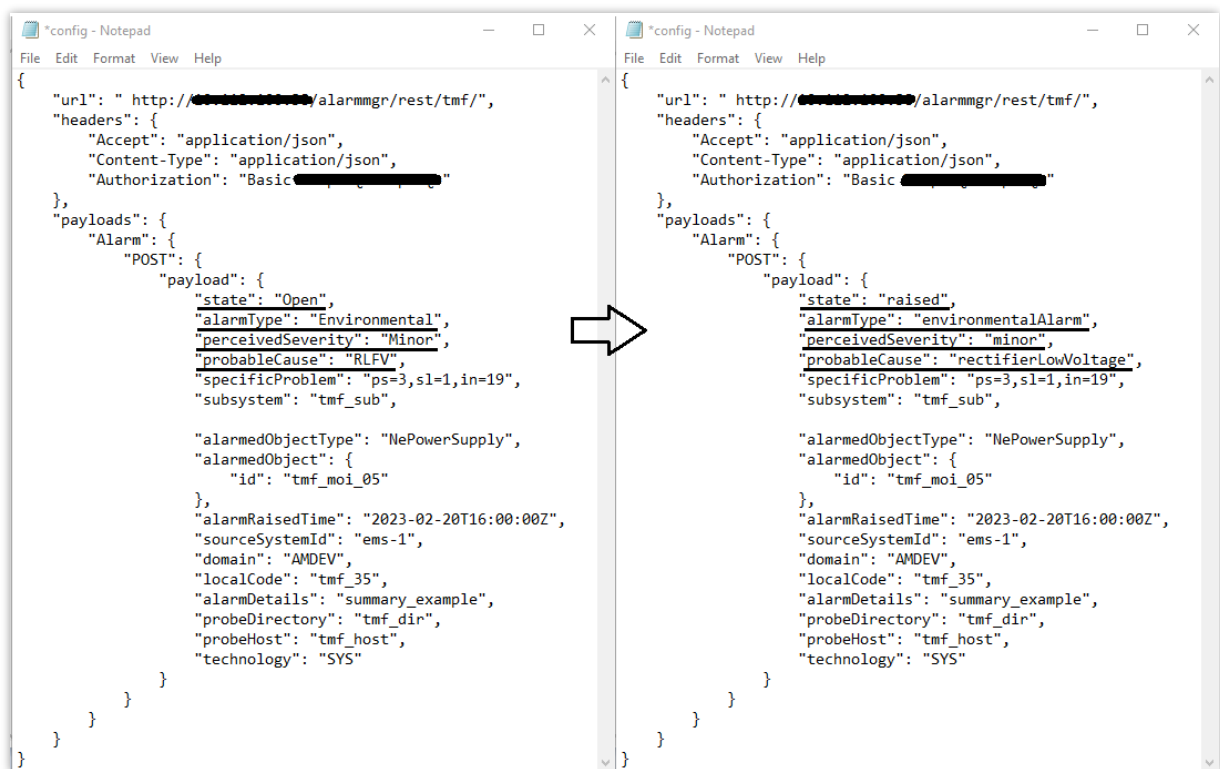


Figura 35: Atualização dos valores enumerados no ficheiro de configurações (config.json) da *CTK*.

A execução dos testes *CTK* serviu unicamente para testar a existência de zero falhas após a conclusão da implementação dos valores enumerados, uma vez que não existia qualquer teste específico para verificar o alinhamento destes valores. No [Apêndice V](#) é possível verificar a utilização dos valores enumerados de acordo com as especificações da *TM Forum*, nomeadamente na Figura 54, Figura 55, Figura 63 e na Figura 65.

4.10 Validação final da *API* através dos testes de conformidade automáticos (*CTK*)

Após a implementação de todas as pedidas e funcionalidades da *API* foi feito um teste final para verificar que a *API* estava completamente alinhada com a especificação da *TM Forum*. O resultado global dos testes *CTK* é apresentado na Figura 36 onde é possível observar que a execução dos mesmos ocorreu sem qualquer falha.

Os resultados individuais relativos a este resultado foram apresentados ao longo de cada fase e podem ser consultados no [Apêndice V](#).

Newman Report

Collection CTK-Alarm-4.0.0
Time Fri May 05 2023 00:02:59 GMT+0000 (Coordinated Universal Time)
Exported with Newman v4.6.1

	Total	Failed
Iterations	1	0
Requests	22	0
Prerequisite Scripts	0	0
Test Scripts	22	0
Assertions	1513	0

Total run duration 50.1s
Total data received 94.36KB (approx)
Average response time 2.2s

Total Failures 0

Figura 36: Resultado global das *CTK*

5. ADOÇÃO DA *OPEN API*

Este capítulo refere-se à solução obtida após a implementação da especificação *TMF642*. Neste capítulo são apresentadas:

- [Apresentação da API desenvolvida](#) – onde são apresentados os pedidos disponibilizados pela *API*, a forma de interagir com a mesma e exemplos de pedidos e respostas;
- [Testes de desempenho](#) – a submissão da solução desenvolvida a testes de desempenho, para verificar que a mesma possui os requisitos mínimos do sistema;
- [Obtenção do certificado de implementação](#) – depois de se concluir a implementação e de se verificar que a *API* corresponde aos requisitos mínimos do sistema foi requisitado e obtido o certificado de conformidade emitido pela *TM Forum*.

5.1 Apresentação da *API* desenvolvida

De acordo com Robillard M., um dos maiores obstáculos que os utilizadores encontram ao aprender a utilizar *APIs* está relacionado com a análise da documentação da mesma e com a forma como é apresentada, referindo que é essencial a presença de exemplos de aplicação e documentação orientada às tarefas que serão efetuadas. Desta forma, permite que os utilizadores se mantenham mais focados e reduz o esforço de aprendizagem (Robillard & Deline, 2011).

No sentido de clarificar a solução final e facilitar a perceção do funcionamento da mesma, foi desenvolvida uma documentação através da especificação *OpenAPI 3*, pertencente às ferramentas *Swagger* (Swagger, 2021). A especificação *OpenAPI* tem-se tornado numa das formas mais populares de descrever como aceder a um serviço através de uma *REST API*, incluindo os pedidos que é possível executar, os tipos de respostas que podem ser recebidas e os seus formatos (Atlidakis et al., 2019).

A representação desenvolvida foi utilizada tanto para documentar este relatório (através de imagens da mesma) como para ser adicionada ao código do sistema e permitir à restante equipa de desenvolvimento do sistema ter um acesso mais direto à mesma. Com recurso à ferramenta *Docker* (Docker, 2022) foi possível transformar a documentação da *API* do formato *OpenApi 3* para *html* tornando-a numa página facilmente acessível através de um *browser*.

A representação final da *API* encontra-se organizada nos seguintes tópicos:

- [Pedidos disponibilizados pela *API*](#);
- [Autenticação necessária para a execução de pedidos](#);
- [Pedidos *GET* – Consulta de alarmes e consulta de alarme pelo seu *id*](#);
- [Pedido *POST* – Criação de alarme](#);
- [Pedido *PATCH* – Alteração de alarme](#).

5.1.1 Pedidos disponibilizados pela *API*

Na Figura 37 é representada a lista de pedidos disponibilizados pela *API*.

TMF642 applied to Alarm Manager 1.0.0 OAS3

General description: This API is aligned with TM Forum Alarm Management Open API (TMF642) specifications and with 3GPP TS 32.111-2 and ITU-T X733 standards

[Contact the developer](#)
Alarm Manager

Servers
http://00.000.000.000/alarmmgr/rest/tmf - SwaggerHub API Auto Mocking

admins Secured Admin-only calls

- POST** /alarm create an alarm
- PATCH** /alarm/{id} update an alarm

developers Operations available to regular developers

- GET** /alarm consult all alarms
- GET** /alarm/{id} consult alarm by id

Figura 37: Lista de pedidos disponibilizados na *TMF642*

5.1.2 Autenticação necessária para a execução de pedidos

Na Figura 38 são apresentadas as autenticações necessárias para efetuar os pedidos, sendo necessário somente uma das opções entre a *basic auth* ou a *bearer auth*.

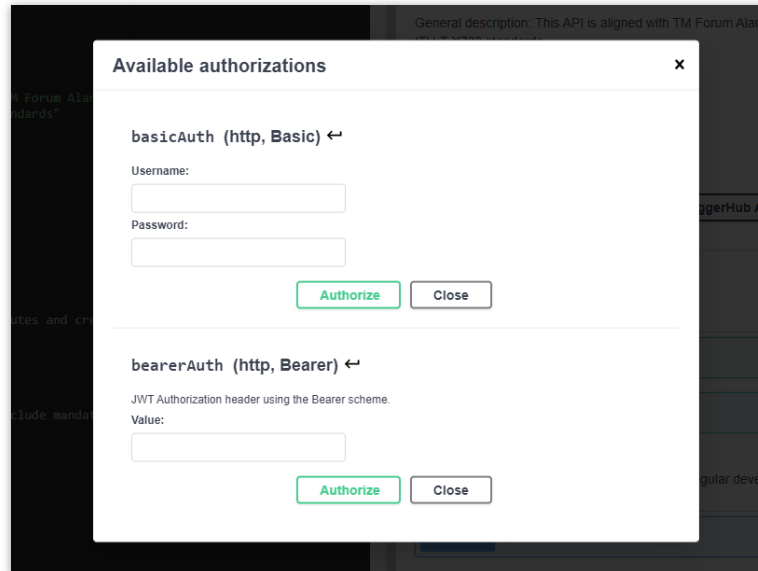


Figura 38: Autenticações disponíveis para efetuar pedidos na *TMF642*

5.1.3 Pedidos *GET* – Consulta de alarmes e consulta de alarme pelo *id*

A Figura 39 representa o pedido *GET* – consulta de alarmes, nomeadamente o *path* e os parâmetros opcionais que se pode acrescentar ao pedido, incluem a seleção de atributos (*fields*) e a aplicação de filtros para valores de atributos (*id*, *alarmRaisedTime*, *probableCause*, *sourceSystemId*, *state*). Na Figura 41 é apresentada um exemplo de resposta do pedido, no caso de sucesso.

A Figura 40 representa o pedido *GET* – consulta de alarme pelo seu *id*, nomeadamente o *path*, o atributo obrigatório do *path* (*id*) e os parâmetros opcionais (*fields*). Para este pedido, a Figura 41 também representa uma resposta de sucesso, sendo a única diferença do pedido de consulta de alarmes é que a resposta só pode conter um alarme.

GET /alarm consult all alarms ^ 🔒

Retrieves the list of active alarms

Parameters Try it out

Name	Description
fields string (header)	optional string with fields to be displayed for each alarm (fields names must be separated by coma) <i>Example</i> : alarmRaisedTime, specificProblem
	<input type="text" value="alarmRaisedTime, specificProblem"/>
id string (header)	id optional filter <i>Example</i> : 43311574
	<input type="text" value="43311574"/>
alarmRaisedTime string(\$date-time) (header)	alarmRaisedTime optional filter <i>Example</i> : 2023-04-21T09:31:23Z
	<input type="text" value="2023-04-21T09:31:23Z"/>
probableCause string (header)	probableCause optional filter <i>Example</i> : demodulationFailure
	<input type="text" value="demodulationFailure"/>
sourceSystemId string (header)	sourceSystemId optional filter <i>Example</i> : ems-1
	<input type="text" value="ems-1"/>
state string (header)	state optional filter <i>Example</i> : raised
	<input type="text" value="raised"/>

Figura 39: GET – Consulta de alarmes – path e parâmetros disponíveis

GET /alarm/{id} consult alarm by id

This operation retrieves a single alarm entity.

Parameters Try it out

Name	Description
id * required string(\$int32) (path)	id of the alarm that will be consulted
fields string(\$int32) (header)	optional string with fields to be displayed for each alarm (fields names must be separated by coma) Example : alarmRaisedTime, specificProblem

id:

fields:

Figura 40: GET – Consulta de alarme pelo *id* – *path* e parâmetros disponíveis

Code Description

200 Alarm obtained with success

Media type
application/json

Controls Accept header.

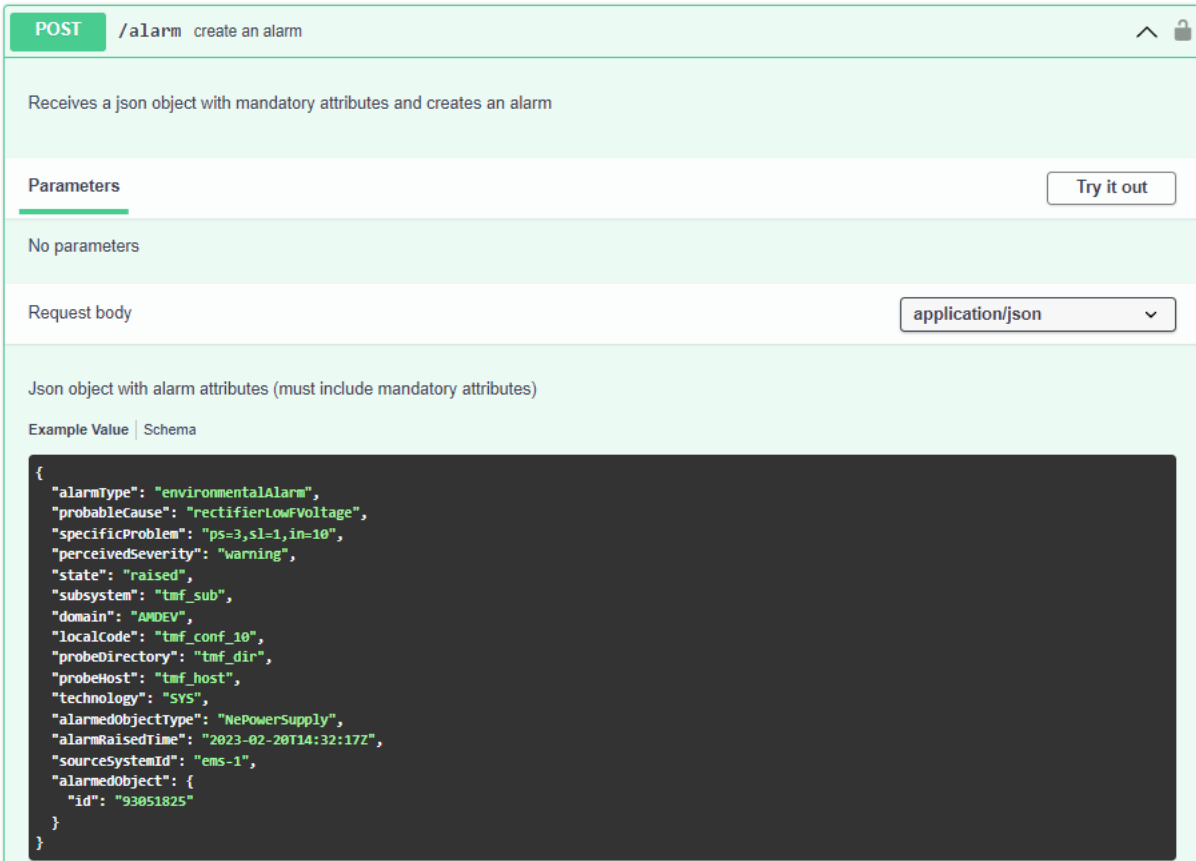
Example Value | Schema

```
[
  {
    "id": "41967207",
    "href": "/alarmmgr/rest/tmf/alarm/41967207",
    "alarmType": "environmentalAlarm",
    "probableCause": "rectifierLowFvoltage",
    "specificProblem": "ps=3,sl=1,in=10",
    "state": "raised",
    "perceivedSeverity": "warning",
    "subsystem": "tmf_sub",
    "domain": "AMDEV",
    "localCode": "tmf_conf_10",
    "probeDirectory": "tmf_dir",
    "probeHost": "tmf_host",
    "technology": "SYS",
    "alarmedObjectType": "NePowerSupply",
    "ackState": "Unacknowledged",
    "ackUserId": "null",
    "alarmRaisedTime": "2023-02-20T14:32:17Z",
    "alarmReportingTime": "2023-02-20T14:32:68Z",
    "alarmChangedTime": "2023-02-20T14:32:68Z",
    "sourceSystemId": "ems-1",
    "correlatedAlarm": {
      "id": "43309517",
      "href": "/alarmmgr/rest/tmf/alarm/43309517"
    },
    "alarmedObject": {
      "id": "93051825",
      "href": "/alarmmgr/rest/tmf/alarm/93051825"
    },
    "@baseType": "Alarm",
    "@type": "Alarm",
    "@schemaLocation": "/alarmmgr/repository/docs/tmf642/Alarm.schema.json"
  }
]
```

Figura 41: GET – Consulta de alarme(s) – exemplo de resposta de sucesso

5.1.4 Pedido *POST* – Criação de alarme

A Figura 42 representa o pedido *POST* – criação de alarme, através da mesma é possível observar o *path* do pedido e um exemplo de *body* válido para a execução do mesmo. No *body* do pedido é necessário constarem todos os atributos obrigatórios para a criação de alarmes (representados na Tabela 6). A Figura 43 apresenta um exemplo de resposta de sucesso para o pedido.



POST /alarm create an alarm

Receives a json object with mandatory attributes and creates an alarm

Parameters Try it out

No parameters

Request body application/json

Json object with alarm attributes (must include mandatory attributes)

Example Value | Schema

```
{
  "alarmType": "environmentalAlarm",
  "probableCause": "rectifierLowVoltage",
  "specificProblem": "ps=3,sl=1,in=10",
  "perceivedSeverity": "warning",
  "state": "raised",
  "subsystem": "tmf_sub",
  "domain": "AMDEV",
  "localCode": "tmf_conf_10",
  "probeDirectory": "tmf_dir",
  "probeHost": "tmf_host",
  "technology": "SYS",
  "alarmedObjectType": "NePowerSupply",
  "alarmRaisedTime": "2023-02-20T14:32:17Z",
  "sourceSystemId": "ems-1",
  "alarmedObject": {
    "id": "93051825"
  }
}
```

Figura 42 : *POST* – Criação de alarme – exemplo de pedido.

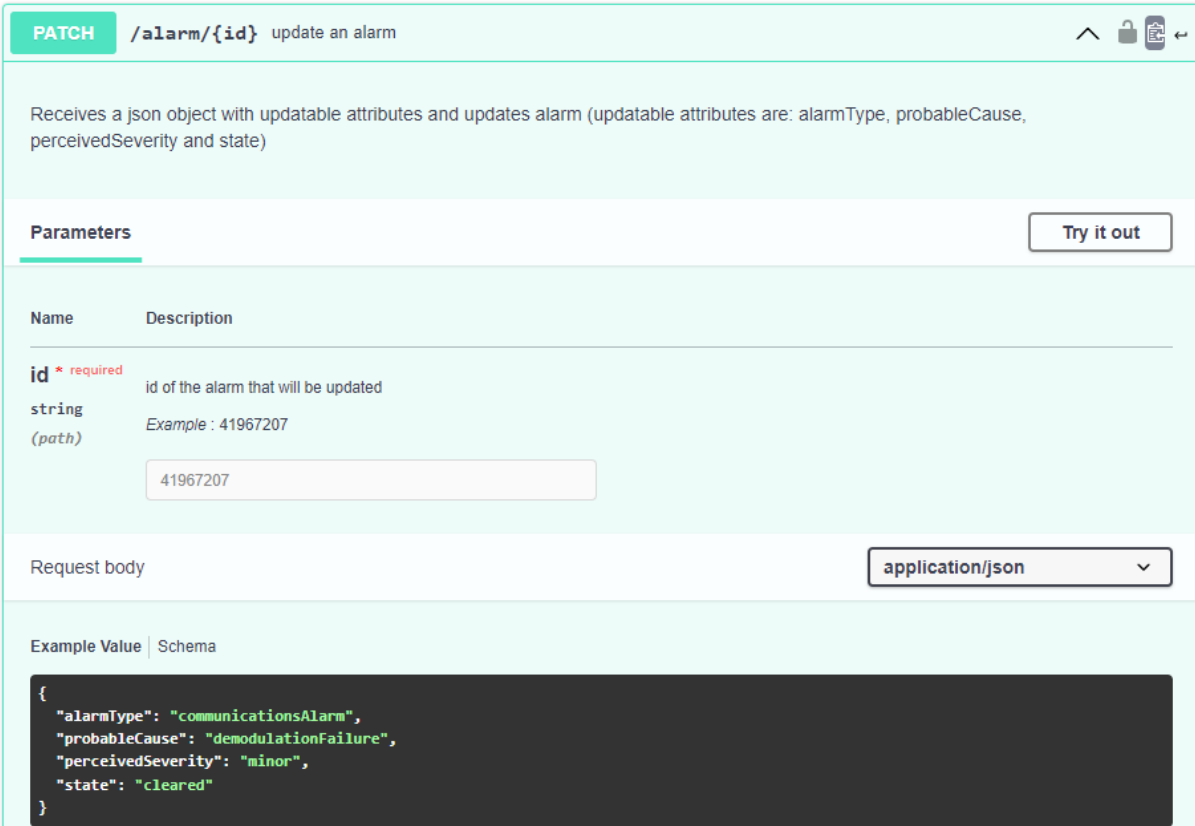
Code	Description
201	Alarm created successfully
	<p>Media type</p> <p><input type="text" value="application/json"/></p> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>[{ "id": "41967207", "href": "/alarmmgr/rest/tmf/alarm/41967207", "alarmType": "environmentalAlarm", "probableCause": "rectifierLowVoltage", "specificProblem": "ps=3,sl=1,in=10", "state": "raised", "perceivedSeverity": "warning", "subsystem": "tmf_sub", "domain": "AMDEV", "localCode": "tmf_conf_10", "probeDirectory": "tmf_dir", "probeHost": "tmf_host", "technology": "SYS", "alarmedObjectType": "NePowerSupply", "ackState": "Unacknowledged", "ackUserId": "null", "alarmRaisedTime": "2023-02-20T14:32:17Z", "alarmReportingTime": "2023-02-20T14:32:68Z", "alarmChangedTime": "2023-02-20T14:32:68Z", "sourceSystemId": "ems-1", "correlatedAlarm": { "id": "43309517", "href": "/alarmmgr/rest/tmf/alarm/43309517" }, "alarmedObject": { "id": "93051825", "href": "/alarmmgr/rest/tmf/alarm/93051825" }, "@baseType": "Alarm", "@type": "Alarm", "@schemaLocation": "/alarmmgr/repository/docs/tmf642/Alarm.schema.json" }]</pre>

Figura 43: *POST* – Criação de alarme – exemplo de resposta de sucesso.

5.1.5 Pedido *PATCH* – Alteração de alarme

A Figura 44 representa o pedido *PATCH* – alteração de alarme, através da mesma é possível observar o *path* do pedido (que inclui o atributo obrigatório *id*) e um exemplo de *body* válido para a execução do mesmo. No *body* do pedido é necessário constar pelo menos um atributo alterável (representado na Figura 29) para que o pedido seja executado com sucesso.

A Figura 45 apresenta um exemplo de resposta de sucesso para o pedido, onde é possível observar que o *id* do alarme devolvido na resposta tem de corresponder ao *id* enviado no *path* e que os respetivos campos enviados no *body* do pedido foram alterados.



PATCH /alarm/{id} update an alarm

Receives a json object with updatable attributes and updates alarm (updatable attributes are: alarmType, probableCause, perceivedSeverity and state)

Parameters Try it out

Name	Description
id * required	id of the alarm that will be updated
string (path)	Example : 41967207

Request body application/json

Example Value | Schema

```
{
  "alarmType": "communicationsAlarm",
  "probableCause": "demodulationFailure",
  "perceivedSeverity": "minor",
  "state": "cleared"
}
```

Figura 44: *PATCH* – Alteração de alarme – exemplo de pedido

Code	Description
201	<p>Alarm updated sucessfully</p> <p>Media type <input type="text" value="application/json"/></p> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>[{ "id": "41967207", "href": "/alarmmgr/rest/tmf/alarm/41967207", "alarmType": "communicationsAlarm", "probableCause": "demodulationFailure", "specificProblem": "ps=3,s1=1,in=10", "state": "cleared", "perceivedSeverity": "minor", "subsystem": "tmf_sub", "domain": "AMDEV", "localCode": "tmf_conf_10", "probeDirectory": "tmf_dir", "probeHost": "tmf_host", "technology": "SYS", "alarmedObjectType": "NePowerSupply", "ackState": "Unacknowledged", "ackUserId": "null", "alarmRaisedTime": "2023-02-20T14:32:17Z", "alarmReportingTime": "2023-02-20T14:32:68Z", "alarmChangedTime": "2023-02-20T16:12:20Z", "sourceSystemId": "ems-1", "correlatedAlarm": {</pre>

Figura 45: *PATCH* – Alteração de alarme – exemplo de resposta de sucesso

5.2 Testes de desempenho da API

O *Apache JMeter* é um projeto *open source* da *Apache* desenvolvido para a medição de desempenho e a execução de testes de *stress*. Esta ferramenta foi inicialmente desenvolvida para testar aplicações web, mas acabou por se estender para outras áreas, incluindo o teste de pedidos do tipo *HTTP*.

A aplicação *JMeter* permite simular cenários em que ocorrem grandes quantidades de pedidos, verificar a capacidade de resposta do serviço que está a ser testado e os aspetos em que varia. Além disso, permite verificar se o funcionamento de um serviço supera os requisitos mínimos estipulados pela organização (Wang & Wu, 2019).

Para verificar se a API implementada possui os requisitos mínimos, a equipa de desenvolvimento do sistema acordou que a solução desenvolvida fosse comparada com a API já utilizada, pois esta possuía as características necessárias exigidas pelos seus clientes.

Apesar da API existente não conter todos os pedidos da *TMF642*, os tempos e comportamentos (variação do tempo de resposta ao longo do período de teste) foram utilizados como referência para validar o desempenho da nova solução (Oliveira, 2023).

Os testes de desempenho foram divididos nas seguintes fases:

- [Teste de desempenho da API desenvolvida \(TMF642\);](#)
- [Teste comparativo entre a API desenvolvida e a API já utilizada pelo sistema;](#)
- [Conclusões acerca do desempenho.](#)

Através da Figura 46 é possível observar os pedidos criados em cada cenário de teste e os tipos de *listeners* adicionados para registo e monitorização dos resultados.

Para a configuração dos testes foi consultada a opinião da equipa de desenvolvimento e foram estipulados os valores mais adequados conforme a quantidade de clientes que utilizam atualmente a API do sistema. Em cada um dos testes executados foram utilizadas as configurações ilustradas na figura 47, nomeadamente:

- um grupo de 6 utilizadores, onde cada um envia 100 pedidos de cada tipo à API;
- o número inicial de utilizadores é 1 e é incrementado um utilizador a cada 40 segundos até atingir 6 utilizadores concorrentes;
- No total são executados 2400 pedidos durante um período de aproximadamente 25 minutos.

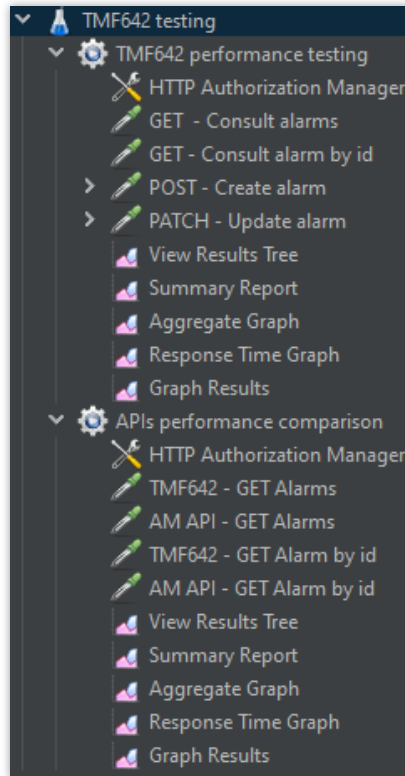


Figura 46: Grupos de teste criados no *JMeter*

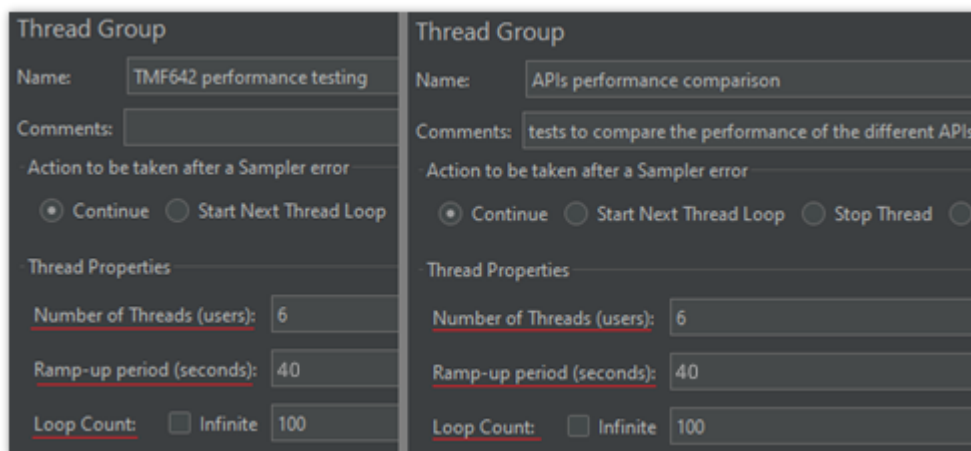


Figura 47: Configurações dos testes do *JMeter*.

5.2.1 Teste de desempenho da API desenvolvida (TMF642)

Através da Tabela 11 e da Figura 48 é possível observar que a resposta dos pedidos teve um tempo médio de 3,8 segundos que foi praticamente igual em todos os pedidos. Além disso, é possível observar através do *throughput* que a quantidade de pedidos processados por minuto também foi igual, com uma média de 23,2 pedidos por minuto.

A principal diferença identificada é a quantidade de dados recebidos (*KB/seg* recebidos), representada na Tabela 11, onde se verifica que o *GET* Consulta de alarmes envolve uma quantidade bastante superior de dados recebidos, uma vez que recebe a lista completa de alarmes existentes na base de dados. Relativamente aos percentis, estes fornecem uma visão mais abrangente do tempo de resposta, pois fornecem mais detalhes sobre a sua distribuição. Como se pode observar através dos valores apresentados o tempo de resposta dos percentis é superior à média e tende a aumentar ligeiramente à medida que estes aumentam a percentagem, o que indica que só uma pequena fração dos utilizadores irá sofrer ligeiros atrasos na realização dos pedidos

Através da Figura 49 é possível observar a evolução dos tempos de resposta ao longo das iterações. As *threads* de execução (utilizadores) foram sendo aumentados de 40 em 40 segundos ao longo do primeiro minuto e meio de execução. Como é possível verificar através da figura ocorreu um aumento do tempo de resposta, apesar disso, a *API* manteve um comportamento estável ao longo do período de execução do teste, registando um tempo de resposta predominantemente entre os 3 e os 4,5 segundos.

Tabela 11: Resultados do teste de desempenho da *TMF642*

Pedido	# Iterações	Média (ms)	Mediana (ms)	Percentis			% de erro	Throughput	KB/seg recebidos	KB/seg enviados
				90%	95%	99%				
<i>GET</i> – Consultar alarmes	600	3814	3812	4564	4853	5576	0,00	23,2/min	10,12	0,07
<i>GET</i> – Consultar alarme por id	600	3748	3745	4464	4783	5436	0,00	23,2/min	0,50	0,07
<i>POST</i> – Criar alarme	600	3799	3794	4527	4752	5262	0,00	23,2/min	0,51	0,31
<i>PATCH</i> – Alterar alarme	600	3833	3825	4621	4940	5704	0,00	23,2/min	0,50	0,11
TOTAL	2400	3799	3796	4554	4813	5513	0,00	1,5/seg	11,60	0,55

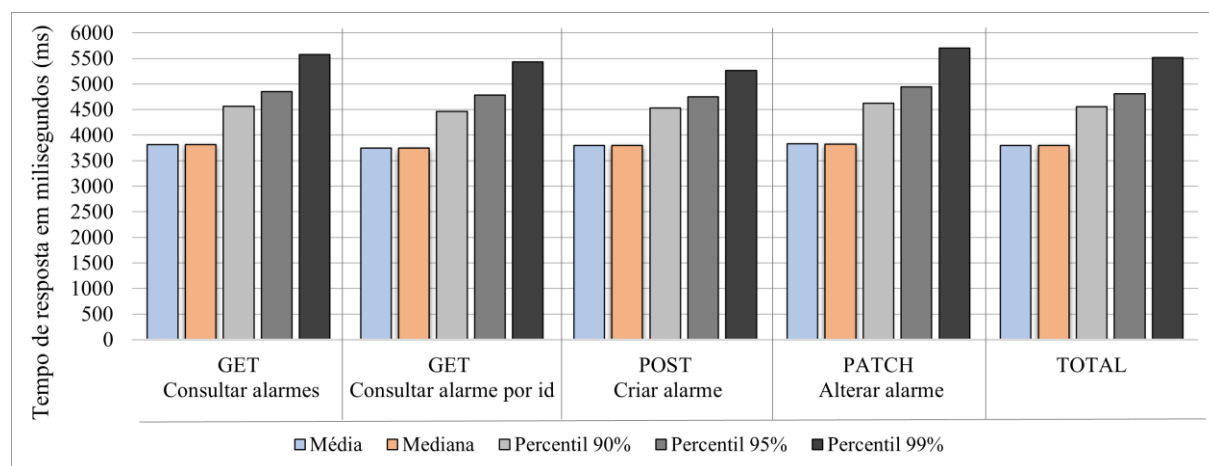


Figura 48: Gráfico de tempos de resposta da *TMF642*.

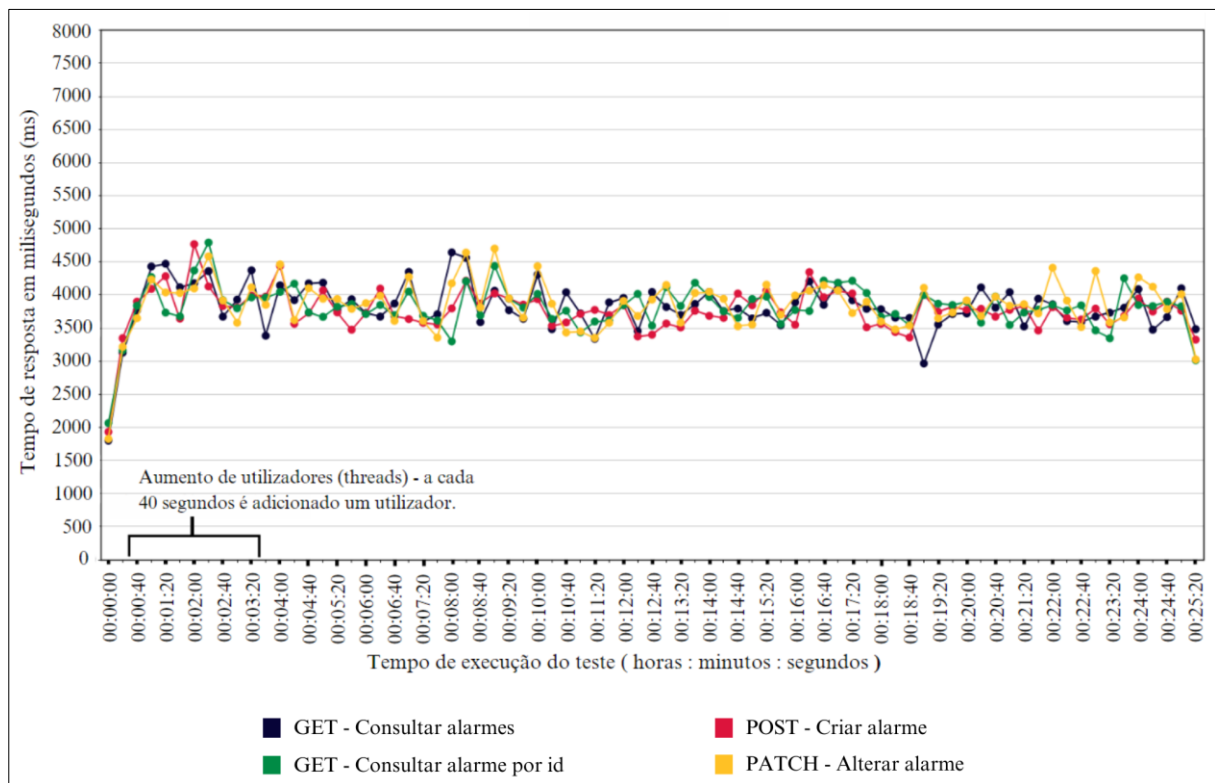


Figura 49: Gráfico da evolução dos tempos de resposta ao longo da execução do teste da *TMF642*.

5.2.2 Teste comparativo entre a *API* desenvolvida e a *API* existente no sistema

Por último foram novamente executados os testes, mas desta vez somente dos pedidos que existiam em ambas as *APIs*.

As *APIs* testadas são diferenciadas pelo nome inicial que designa os pedidos:

- *TMF642* – *API* desenvolvida no âmbito deste trabalho;
- *API* do sistema – *API* que já existia no sistema.

Através da Tabela 12 e do gráfico da Figura 50 é possível verificar que a *API* que já existia registou uma média de tempos de resposta praticamente igual à nova solução. Além disso a quantidade média de pedidos efetuados por minuto (*throughput*) foi igual em todos os pedidos. Relativamente aos percentis verificou-se que tiveram uma variação muito idêntica em ambas as soluções, indicando que em ambos os casos uma pequena fração dos pedidos sofreram um atraso e que o atraso foi muito semelhante em ambas as *APIs*.

A principal diferença identificada entre as duas *APIs* está na quantidade de dados recebidos (*KB/seg* recebidos), onde a *TMF642* regista valores significativamente inferiores, uma vez que possui uma quantidade bastante menor de atributos em cada alarme.

Relativamente à evolução dos tempos de resposta, representada na Figura 51, inicialmente ocorreu um ligeiro aumento do tempo de resposta causado pelo aumento das *threads* de execução. Apesar disso ambas as soluções mantiveram um comportamento constante, com tempos de resposta entre os 3 e os 4,5 segundos.

Tabela 12: Resultado do teste de desempenho comparativo

Pedido	# Iterações	Média (ms)	Mediana (ms)	Percentis (ms)			% de erro	Throughput	KB/seg recebidos	KB/seg enviados
				90%	95%	99%				
(TMF642) GET - Consultar alarmes	600	3717	3711	4362	4562	5044	0,00	23,7/min	10,34	0,07
(API do sistema) GET - Consultar alarmes	600	3738	3748	4365	4581	5026	0,00	23,7/min	16,54	0,07
(TMF642) GET - Consultar alarme por id	600	3682	3661	4301	4534	5250	0,00	23,7/min	0,51	0,07
(API do sistema) GET - Consultar alarme por id	600	3672	3675	4325	4532	4983	0,00	23,7/min	0,69	0,07
TOTAL	2400	3702	3698	4343	4551	5079	0,00	1,6/seg	27,99	0,28

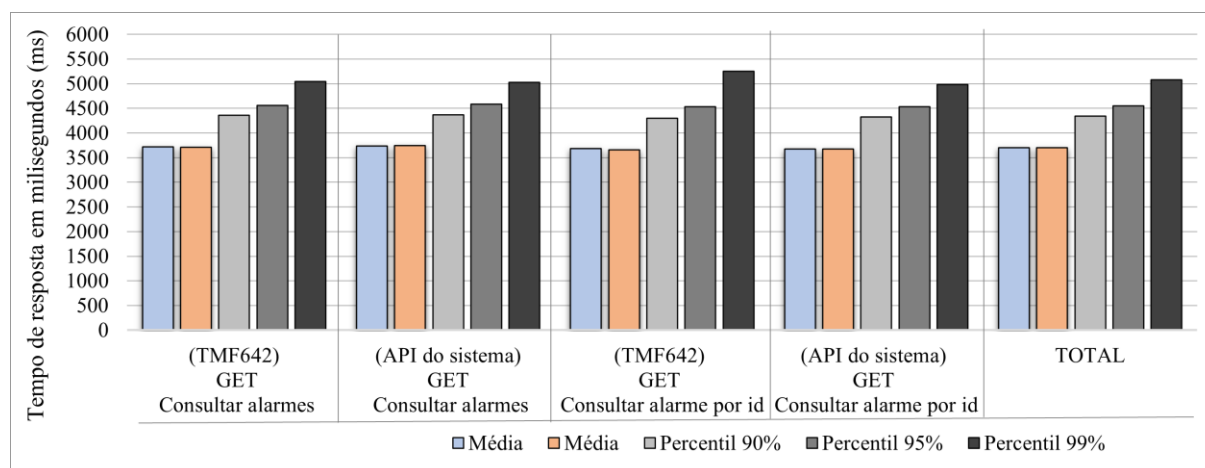


Figura 50: Gráfico comparativo dos tempos de resposta.

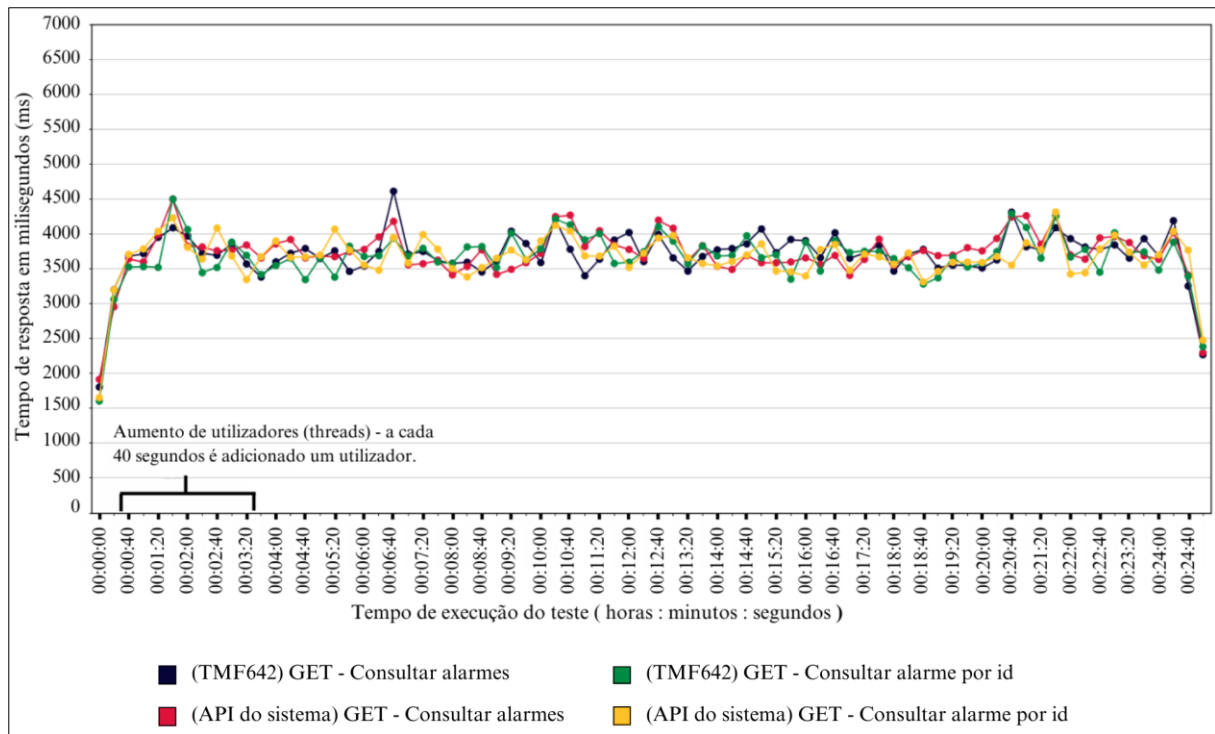


Figura 51: Gráfico comparativo da evolução dos tempos de resposta ao longo do tempo de execução do teste.

5.2.3 Conclusões acerca do desempenho

Através da comparação dos resultados obtidos em ambos os testes, verifica-se que a *TMF642* aplicada ao sistema possui um comportamento praticamente igual à *API* existente. Apesar da *TMF642* incluir mais pedidos, estes também registaram tempos de resposta e variações praticamente iguais.

Com base nesta análise pode-se concluir que a solução desenvolvida possui os requisitos mínimos estipulados pela equipa de desenvolvimento do sistema. A fornecedora de serviços de telecomunicações verificou os resultados e aprovou a *API* desenvolvida para ser utilizada pelos seus clientes.

5.3 Requerimento e obtenção do certificado de implementação

Após a conclusão da implementação da *Open API* e depois de se obter:

- os resultados dos testes automáticos (*CTK*) sem falhas;
- confirmação de que a *API* possui as características necessárias para ser utilizada pelos clientes da fornecedora de serviços de telecomunicações (demonstrado através dos testes de desempenho);

a empresa fornecedora de serviços de telecomunicações procedeu à requisição do certificado. Para tal foi necessário submeter um [formulário do site da TM Forum](#) destinado a esse propósito e juntamente os ficheiros de resultados dos testes automáticos (*results.html* e *results.json*).

Na Figura 52 é apresentado o certificado obtido pela fornecedora que consta na [página de certificação do site da TM Forum](#). Além disso, informação adicional sobre a certificação também pode ser consultada no [relatório de certificação](#) publicado no mesmo site.

Company	Certification	Certification Scope	Release	Product/Solution	Certification Date
	TMF642 - Alarm Management	Open API	4.0.0	NOSSIS* - Network Operations Support Systems Integrated Solutions	June 2023

Figura 52: Obtenção do certificado de implementação

6. MÉTODO DE IMPLEMENTAÇÃO

Com base na implementação da *TMF642* no sistema de gestão de alarmes, foi desenvolvido um método de adoção global. Esta tem como objetivo resumir as fases envolvidas no processo de implementação de *Open APIs* da *TM Forum* e facilitar a adoção das mesmas por parte de outras fornecedoras de serviços de telecomunicações, no sentido de promover a interoperabilidade entre os sistemas de gestão de redes.

O método pode ser observada na Figura 53. Esta serve para ser aplicada a partir do momento em que uma empresa de telecomunicações decide adotar uma *Open API*.

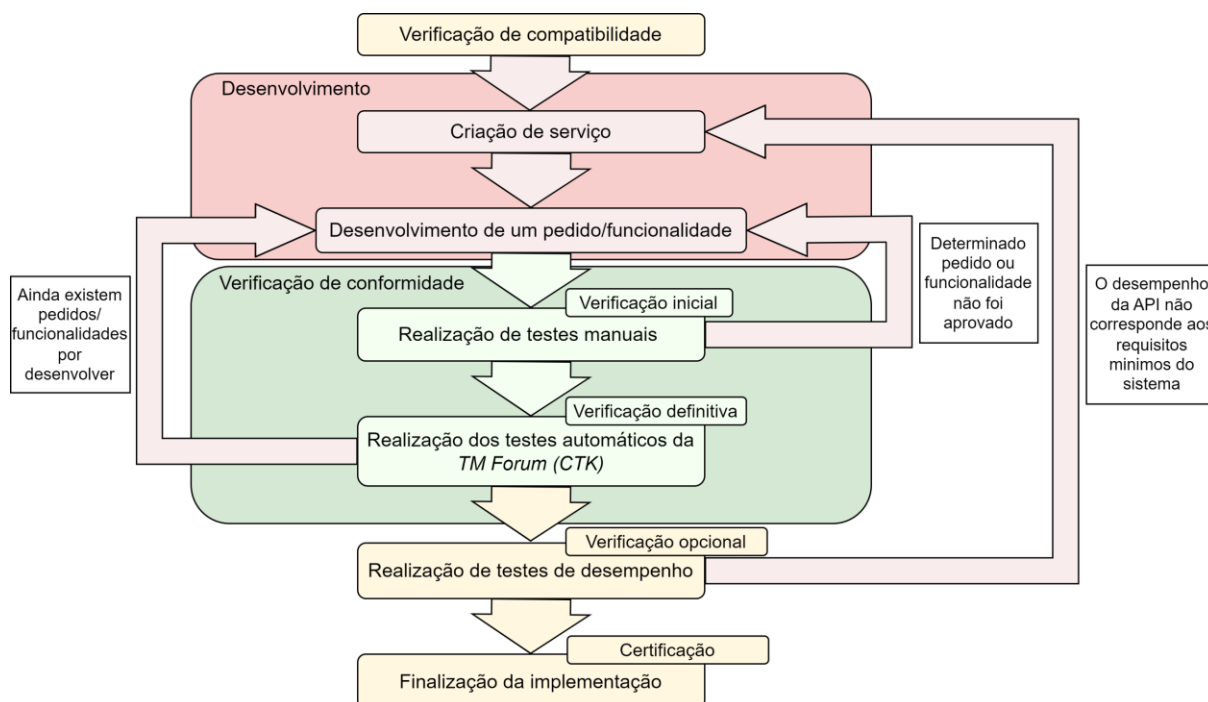


Figura 53: Método de implementação de *Open APIs* da *TM Forum*.

Tal como demonstrado na Figura 53, as fases que a constituem a método são as seguintes:

1. **Verificação da compatibilidade** – Nesta fase é feita uma análise inicial da especificação e é feita uma comparação com o sistema onde a mesma vai ser implementada. Para tal começa-se por analisar a documentação disponibilizada na *Open API table* da *TM Forum* (TM Forum, 2023). Nesta fase é aconselhado uma execução inicial dos testes de conformidade automáticos (*CTK*) e a posterior análise dos resultados esperados, presente nos ficheiros de resultado dos testes (*results.html* e *results.json*). Isto permitirá facilitar a identificação de pedidos, funcionalidades e atributos obrigatórios de implementar. Com base nestas informações deve-se:

- Proceder ao levantamento dos pedidos e funcionalidades exigidas pela especificação e verificar a existência de operações e funcionalidades correspondentes no sistema.
- Efetuar uma análise dos recursos (*resources*) envolvidos na implementação e realizar o mapeamento entre os atributos obrigatórios da especificação e os do sistema.

Caso se verifique a possibilidade de implementar as funcionalidades no sistema e a existência de todos os atributos obrigatórios, passa-se para a fase de criação de um novo serviço *REST*.

2. **Criação de um serviço** – Esta fase tem como objetivo criar o serviço que disponibiliza a interface para os clientes da *Open API* interagirem com o sistema. Nesta fase são realizadas as seguintes tarefas:

- É criado um novo serviço no código do sistema. Este serviço é composto por uma interface que irá conter os pedidos identificados na fase anterior e uma classe de implementação com métodos necessários para processar os diferentes pedidos.
- São criadas as classes relativas aos recursos (*resources*), com os atributos mapeados na fase anterior.

Após a criação do serviço e das classes de recursos, é iniciado o desenvolvimento de um pedido ou funcionalidade.

3. **Desenvolvimento de pedido ou funcionalidade** – Nesta fase procede-se ao desenvolvimento de um dos pedidos ou funcionalidades que constam na especificação. Normalmente as funcionalidades são desenvolvidas posteriormente aos pedidos, uma vez que são acrescentadas aos mesmos. Por exemplo inicialmente desenvolve-se um pedido de consulta e numa segunda iteração desta fase desenvolve-se a funcionalidade de filtragem de alarmes associada a esse pedido. O desenvolvimento de pedidos e funcionalidades deve reaproveitar o máximo de recursos existentes no código do sistema e ser ligado às funções internas de processamento dos recursos. Por exemplo a consulta de alarmes deve ser ligada à função interna responsável por fazer as invocações necessárias até que ocorra uma busca dos alarmes à base de dados.

Esta fase envolve tarefas como:

- O desenvolvimento de um método de implementação que invoca métodos internos de consulta e processamento de recursos;
- O desenvolvimento de métodos de conversão baseados na correspondência de atributos entre os objetos do sistema e os recursos da especificação;

- A invocação dos métodos de conversão, sempre que necessário, para que os utilizadores da *Open API* consigam submeter e receber os recursos de acordo com a especificação.

Após o desenvolvimento do pedido ou funcionalidade passa-se para a fase de realização de testes manuais.

4. **Realização de testes manuais** (verificação inicial) – A realização de testes manuais permite que o implementador verifique o alinhamento do pedido ou funcionalidade desenvolvida com a documentação fornecida pela especificação. Caso o pedido/funcionalidade não seja aprovado, deve-se regressar à fase de desenvolvimento de pedido/funcionalidade para corrigir eventuais desalinhamentos.

Caso seja aprovado, deve proceder-se à execução dos testes automáticos para se verificar os pedidos que ainda não são aprovados pela *TM Forum*.

5. **Realização de testes automáticos da *TM Forum* (CTK)** (verificação definitiva) – A realização dos testes automáticos fornecidos pela *TM Forum* consiste no principal mecanismo de verificação fornecido pela organização para certificar a *API*. Apesar deste teste ser bastante eficiente em verificar o alinhamento da *API* em desenvolvimento com a especificação da *TM Forum*, ele segue uma ordem específica de execução dos pedidos. O teste possui uma sequência de operações, este inicia com a criação de um recurso (*resource*) e de seguida efetua pedidos como a consulta do recurso criado e a sua alteração. Se o implementador por algum motivo não seguir essa ordem exata de desenvolvimento (*POST-> GET-> PATCH*) deve ter algum espírito crítico para interpretar os resultados dos testes automáticos *CTK*.

Caso a *CTK* detete que ainda existem pedidos ou funcionalidades por desenvolver e o implementador verifique que realmente ainda estão em falta, deve regressar para a fase de desenvolvimento e proceder ao desenvolvimento do próximo pedido/funcionalidade.

Caso todos os pedidos e funcionalidades tenham sido desenvolvidos e os testes automáticos possuam um resultado sem falhas passa-se para uma fase opcional de realização de testes de desempenho.

6. **Realização de testes de desempenho** (verificação opcional) – A realização de testes de desempenho consiste numa fase opcional em que o implementador deve verificar se existem requisitos mínimos de desempenho estabelecidos pela fornecedora de serviços de telecomunicações. O implementador deve tentar reunir informações importantes para o cenário de teste, como o tempo de resposta médio esperado, a quantidade de utilizadores concorrentes

esperados, entre outros, de forma a configurar os testes e verificar se a solução desenvolvida preenche os requisitos.

Caso a *API* não possua os requisitos mínimos estipulados pela fornecedora deve ser feita uma revisão completa da mesma, regressando à fase de criação do serviço.

Caso a *API* seja aprovada, passa-se para a última fase, a finalização da implementação.

7. **Finalização da implementação** (certificação) – Nesta fase é preenchido e emitido um formulário de certificação disponibilizado no site da *TM Forum*, este é submetido para requerimento do certificado da implementação da *Open API*. Além do formulário é necessário serem enviados os ficheiros de resultados obtidos nos testes automáticos, para confirmar o alinhamento da *API* com a especificação.

O certificado é a principal forma de comprovar que a implementação da *API* foi bem-sucedida e aprovada pela *TM Forum*, após a sua obtenção o mesmo pode ser utilizado para atrair novos clientes e demonstrar que o sistema da fornecedora está alinhado com as *Open APIs* da *TM Forum*. Além disso a obtenção do certificado também permite que a fornecedora seja divulgada nos canais de marketing da *TM Forum*, reforçando a sua probabilidade de ser reconhecida por novos clientes.

Nesta fase também é possível que o implementador negocie o nível de implementação com a empresa de telecomunicações, mediante a necessidade de implementação de funcionalidades adicionais.

7. CONCLUSÕES

Informação sobre o cumprimento dos objetivos:

Através do desenvolvimento da presente dissertação foi possível alcançar todos os objetivos definidos. O primeiro objetivo é atingido no [capítulo 2](#), onde é feito um estudo do programa das Open APIs da TM Forum e de que forma o mesmo pode ser aplicado para melhorar a interoperabilidade do sistema. O segundo objetivo é demonstrado no [capítulo 4](#), onde é demonstrado o desenvolvimento da solução e a aprovação da mesma através da realização dos testes de conformidade. O terceiro objetivo é alcançado no [capítulo 5](#), onde são apresentados os resultados dos testes de desempenho. O quarto objetivo é apresentado igualmente no [capítulo 5](#), onde é apresentada a obtenção do certificado e a divulgação da implementação nos canais de marketing da *TM Forum*. O último objetivo é apresentado no [capítulo 6](#), onde consta o método de implementação.

Conclusões sobre a implementação:

A *TMF642* permitiu acrescentar uma solução alternativa ao sistema, sem para tal implicar uma alteração do código de processamento de alarmes e sem que os serviços disponibilizados até ao momento fossem afetados. Este aspeto ficou comprovado pela execução dos testes de desempenho que demonstram a utilização das duas *APIs* em simultâneo.

Além disso, através dos testes de desempenho foi possível concluir que a solução implementada não introduz qualquer tipo de atraso no tempo de resposta dos pedidos, permitindo disponibilizar a novos clientes e outros sistemas de suporte à operação uma nova forma de aceder às funcionalidades do sistema que possua um desempenho muito idêntica à já existente.

A empresa fornecedora de serviços de telecomunicações fica assim com uma nova alternativa para os seus clientes acederem à informação e funcionalidades de gestão de alarmes, apesar desta não ser desenvolvida de uma forma tão conveniente como a *API* existente e envolver atributos mais extensos é bastante mais descritiva. Isto quer dizer que os clientes terão bastante mais facilidade em reconhecer os seus termos uma vez que está alinhada com os termos utilizados nas principais normas de gestão de alarmes, permitindo-lhes entender a sua informação sem necessitar de apoio técnico ou de consultar documentação adicional. Estes fatores contribuem para que os clientes integrem o sistema de uma forma mais simples e rápida, melhorando a interoperabilidade entre o sistema de gestão de alarmes e os restantes sistemas de gestão de redes.

Do trabalho realizado resultou uma publicação científica (Oliveira, 2023) que apresenta de forma resumida o trabalho realizado e as conclusões obtidas.

Trabalho futuro:

A implementação da *TMF642* permitiu identificar algumas diferenças entre a gestão de alarmes do sistema e a especificação da *TM Forum*. O sistema demonstrou ser bastante mais complexo e conter uma quantidade significativamente maior de atributos para cada alarme, inclusive atributos relativos às entidades de gestão que do lado da *TM Forum* é uma área menos explorada. Isto deve-se à *TM Forum* desenvolver as especificações para serem adotadas por uma grande diversidade de sistemas, inclusive os mais simples, e os sistemas mais complexos acabam por evoluir conforme necessidades específicas dos seus clientes ou do negócio.

Apesar disto a especificação da *TM Forum* tem a possibilidade de ser extensível (TM Forum, 2022c), o que significa que a empresa fornecedora tem a oportunidade de acrescentar os atributos e funcionalidades extra de que necessita à solução desenvolvida, permitindo satisfazer as necessidades específicas dos seus clientes e mantendo o alinhamento com a *Open API* da *TM Forum*.

Relativamente à funcionalidade de filtrar alarmes, considero que seja vantajoso adicionar um parâmetro opcional no pedido de consulta de alarmes que permita filtrar os alarmes no formato *ODATA*. Isto permitiria agilizar a funcionalidade de filtragem a facilitar a procura de alarmes, pois permite utilizar uma grande diversidade de combinações de pesquisa (Thoma et al., 2014) em vez de disponibilizar somente a igualdade de um valor, como na especificação atual.

Por fim, nesta dissertação também foi incluído um método para a implementação de *Open APIs* da *TM Forum*, concebida com o objetivo de ajudar as fornecedoras de serviços de telecomunicações a implementar as especificações de uma forma mais eficiente. Este método destina-se a ser alvo de crítica em futuras implementações para que a mesma possa ser gradualmente aperfeiçoada e desta forma tornar o processo cada vez mais prático mediante as experiências de novos implementadores.

REFERÊNCIAS

- 3GPP. (2022a). *3GPP TS 32.111-2 – Telecommunication management; Fault Management; Part 2: Alarm Integration Reference Point (IRP): Information Service (IS) – TechSpec*. <https://itecspec.com/archive/3gpp-specification-ts-32-111-2/>
- 3GPP. (2022b). *Introducing 3GPP*. <https://www.3gpp.org/about-us/introducing-3gpp> (visitado em maio de 2023)
- Altice Labs. (2020). *Alarm Manager About Altice Labs Fault Management System Gain control on your network Reduce the impact in QoE*. Altice Labs. www.alticelabs.com (visitado em maio de 2023)
- Asres, M. W., Mengistu, M. A., Castrogiovanni, P., Bottaccioli, L., Macii, E., Patti, E., & Acquaviva, A. (2021). Supporting Telecommunication Alarm Management System with Trouble Ticket Prediction. *IEEE Transactions on Industrial Informatics*, *17*(2), 1459–1469. <https://doi.org/10.1109/TII.2020.2996942>
- Atlidakis, V., Godefroid, P., & Polishchuk, M. (2019). RESTler: Stateful REST API Fuzzing. *Proceedings - International Conference on Software Engineering, 2019-May*, 748–758. <https://doi.org/10.1109/ICSE.2019.00083>
- Baras, J. S., Ball, M., Gupta, S., Viswanathan, P., & Shah, P. (1997). Automated network fault management. *Proceedings - IEEE Military Communications Conference MILCOM, 3*, 1244–1250. <https://doi.org/10.1109/MILCOM.1997.644967>
- Bhat, J. R., & Alqahtani, S. A. (2021). 6G Ecosystem: Current Status and Future Perspective. *IEEE Access*, *9*, 43134–43167. <https://doi.org/10.1109/ACCESS.2021.3054833>
- Docker. (2022). *Docker: Accelerated, Containerized Application Development*. <https://www.docker.com/> (visitado em maio de 2023)
- Douvinet, J. (2018). Smartphone Applications: a Means to Promote Emergency Management in France? *How Information Systems Can Help in Alarm/Alert Detection*, 55–71. <https://doi.org/10.1016/B978-1-78548-302-8.50003-4>
- Flood, J. (1997). *Telecommunication Networks - Google Livros*. https://books.google.pt/books?hl=pt-PT&lr=&id=GR-1UXIYlwWC&oi=fnd&pg=PR10&dq=telecommunications+networks&ots=TuCITwZZrx&sig=dORH_hLUrHwVQ8xxv2e6qfLGwug&redir_esc=y#v=onepage&q=telecommunications%20networks&f=false

- Gorod, A., Gove, R., Sauser, B., Boardman, J., & Schaefer, C. V. (2007). System of Systems Management: A Network Management Approach bsouser(sevens edu boardmangstevens edu. *IEEE International Conference on System of Systems Engineering*.
- Gutierrez-Estevez, D. M., Gramaglia, M., Domenico, A. De, Dandachi, G., Khatibi, S., Tsolkas, D., Balan, I., Garcia-Saavedra, A., Elzur, U., & Wang, Y. (2019). Artificial intelligence for elastic management and orchestration of 5G networks. *IEEE Wireless Communications*, 26(5), 134–141. <https://doi.org/10.1109/MWC.2019.1800498>
- Hasan, M., Sugla, B., & Viswanathan, R. (1999). A conceptual framework for network management event correlation and filtering systems. *Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management: Distributed Management for the Networked Millennium*, 233–246. <https://doi.org/10.1109/INM.1999.770686>
- IntelliJ. (2021). *IntelliJ IDEA – the Leading Java and Kotlin IDE*. <https://www.jetbrains.com/idea/> (visitado em maio de 2023)
- ITU. (1992). *X.733 : Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function*. <https://www.itu.int/rec/T-REC-X.733-199202-I/en>
- ITU. (2005). *M.3100 : Generic network information model*. <https://www.itu.int/rec/T-REC-M.3100-200504-I/en>
- Jager, B., Doucette, J., & Tipper, D. (2008). Network Survivability. *Information Assurance*, 81–112. <https://doi.org/10.1016/B978-012373566-9.50006-9>
- Jenkins. (2023). *Jenkins*. <https://www.jenkins.io/> (visitado em maio de 2023)
- Liang, R., Zhang, Z., Liu, F., & Qu, J. (2019). *A Bayesian-based Self-Diagnosis Approach for Alarm Prognosis in Communication Networks; A Bayesian-based Self-Diagnosis Approach for Alarm Prognosis in Communication Networks*. <https://doi.org/10.1016/j.ins.2010.04.013>
- Mall, R. (2018). *FUNDAMENTALS OF SOFTWARE ENGINEERING, FIFTH EDITION - MALL, RAJIB - Google Livros*. https://books.google.pt/books?hl=pt-PT&lr=&id=JNuDwAAQBAJ&oi=fnd&pg=PP1&dq=software+development+methodologies&ots=PAMk_PU8pd&sig=Fno7bP7eSYvuUYW17OgK_whk8AI&redir_esc=y#v=onepage&q=software%20development%20methodologies&f=false
- Misra, Kundan. (2004). *OSS for telecom networks: an introduction to network management*. 302. https://books.google.com/books/about/OSS_for_Telecom_Networks.html?hl=pt-PT&id=s7kfMfB2o6kC

- Nally M. (2020). *gRPC vs REST: Understanding gRPC, OpenAPI and REST and when to use them in API design* / Google Cloud Blog. <https://cloud.google.com/blog/products/api-management/understanding-grpc-openapi-and-rest-and-when-to-use-them>
- Oliveira, A. (2023). Evolução de um sistema de gestão de alarmes. *Atas Do Décimo Quarto Simpósio de Informática (INForum 2023)*.
- Peffer, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2014). A Design Science Research Methodology for Information Systems Research. *https://doi.org/10.2753/MIS0742-1222240302*, 24(3), 45–77. <https://doi.org/10.2753/MIS0742-1222240302>
- Postman. (2023). *Postman API Platform / Tools*. <https://www.postman.com/product/tools/> (visitado em maio de 2023)
- Quiña-Mera, A., Fernandez, P., García, J. M., & Ruiz-Cortés, A. (2023). GraphQL: A Systematic Mapping Study. *ACM Computing Surveys*, 55(10). <https://doi.org/10.1145/3561818>
- Robillard, M. P., & Deline, R. (2011). A field study of API learning obstacles. *Empirical Software Engineering*, 16(6), 703–732. <https://doi.org/10.1007/S10664-010-9150-8/METRICS>
- Salau, A., Yinka-Banjo, C., Misra, S., Adewumi, A., Ahuja, R., & Maskeliunas, R. (2019). Design and implementation of a fault management system. *Advances in Intelligent Systems and Computing*, 939, 495–505. https://doi.org/10.1007/978-3-030-16681-6_49/COVER
- Schwaber, K., & Sutherland, J. (2020). *Manifesto for Agile Software Development*.
- Scrum.org. (2020). *Scrum.org Scrum Framework*. <https://www.scrum.org/resources/what-scrum-module> (visitado em maio de 2023)
- Sturm, R., Pollard, C., & Craig, J. (2017). Management of Traditional Applications. *Application Performance Management (APM) in the Digital Enterprise*, 25–39. <https://doi.org/10.1016/B978-0-12-804018-8.00003-6>
- Swagger. (2021). *OpenAPI Specification - Version 3.0.3* / Swagger. <https://swagger.io/specification/> (visitado em maio de 2023)
- Swagger. (2023). *API Documentation & Design Tools for Teams* / Swagger. <https://swagger.io/> (visitado em maio de 2023)
- Tawalbeh, L. (2020). Network Management. *The NICE Cyber Security Framework*, 99–115. https://doi.org/10.1007/978-3-030-41987-5_5
- Tawalbeh, L., Hashish, S., & Tawalbeh, H. (2017). Quality of Service requirements and Challenges in Generic WSN Infrastructures. *Procedia Computer Science*, 109, 1116–1121. <https://doi.org/10.1016/J.PROCS.2017.05.441>

- Thoma, M., Kakantousis, T., & Braun, T. (2014). Rest-based sensor networks with OData. *11th Annual Conference on Wireless On-Demand Network Systems and Services, IEEE/IFIP WONS 2014 - Proceedings*, 33–40. <https://doi.org/10.1109/WONS.2014.6814719>
- TM Forum. (2021). *TM Forum - How to manage Digital Transformation, Agile Business Operations & Connected Digital Ecosystems*. <https://www.tmforum.org/> (visitado em maio de 2023)
- TM Forum. (2022a). *Open APIs - TM Forum*. <https://www.tmforum.org/oda/implementation/open-apis/> (visitado em maio de 2023)
- TM Forum. (2022b). *TM Forum Open API Conformance Overview | TM Forum*. <https://www.tmforum.org/conformance-certification/open-api-conformance/> (visitado em maio de 2023)
- TM Forum. (2022c). *TMF642 Alarm Management API User Guide v4.0.1 | TM Forum*. <https://www.tmforum.org/resources/specification/tmf642-alarm-management-api-user-guide-v4-0/>
- TM Forum. (2023). *Open API Table - TM Forum Ecosystem API Portal - TM Forum Confluence*. <https://projects.tmforum.org/wiki/display/API/Open+API+Table> (visitado em maio de 2023)
- vom Brocke, J., Hevner, A., & Maedche, A. (2020). *Introduction to Design Science Research*. 1–13. https://doi.org/10.1007/978-3-030-46781-4_1
- Wang, J., & Wu, J. (2019). Research on Performance Automation Testing Technology Based on JMeter. *2019 International Conference on Robots & Intelligent System (ICRIS)*, 55–58. <https://doi.org/10.1109/ICRIS.2019.00023>
- Xie, M., Zhang, Q., Gonzalez, A. J., Grønsund, P., Palacharla, P., & Ikeuchi, T. (2019). *Service Assurance in 5G Networks: A Study of Joint Monitoring and Analytics; Service Assurance in 5G Networks: A Study of Joint Monitoring and Analytics*.
- Yamakami, T. (2010). OSS as a digital ecosystem: A reference model for digital ecosystem of OSS. *Proceedings of the International Conference on Management of Emergent Digital EcoSystems, MEDES'10*, 207–208. <https://doi.org/10.1145/1936254.1936291>
- Yang, X., Lee, J., & Jung, H. (2019). Regular paper 128 Fault Diagnosis Management Model using Machine Learning. *J. Inf. Commun. Converg. Eng.*, 17(2), 128–134. <https://doi.org/10.6109/jicce.2019.17.2.128>

APÊNDICE I: COMPARAÇÃO DE RESPOSTAS DO PEDIDO DE CONSULTA DE ALARME

Tabela 13: Comparação de respostas dos pedidos de consulta de alarme

API do sistema	Open API da TM Forum (TMF642)
Exemplo de respostas do pedido <i>GET</i> de consulta de alarmes:	
<pre>{ "id": "34918209", "ackDur": 0, "ack": "Unacknowledged", "acksystem": "SSO.SS03.SS003", "acktime": "2019-02-27T10:18:00Z", "ackuser": "amint", "addtext": "<ATARGET=_blankHREF=http://sga.telecom.pt/portal/AM/ACs/CadRede_AC_00AA00.html> CadRede-Detalheda&Aacute;readeCentral

", "type": "Communications", "extref": "2039501", "changed": "2021-01-21T14:35:26Z", "cleared": "2019-02-27T10:18:00Z", "creation": "2019-02-27T10:17:00Z", "alarmDur": 3, "corrid": "129324", "dom": "ALMMON", "ev": 3, "extkey": "GEO_CLI00AA00CEM1291911108,2912220631", "s.sFlap": false, "s.lFlap": false, "s.mFlap": false, "s.cFlap": false, "s.eer": false,</pre>	<pre>{ "id": "8675309", "href": "/alarmManagement/v4/alarm/8675309", "@baseType": "Alarm", "@type": "Alarm", "@schemaLocation": "/alarmManagement/v4/schema/Alarm.schema.json", "externalAlarmId": "5551212", "state": "updated", "alarmType": "environmentalAlarm", "perceivedSeverity": "major", "probableCause": "rectifierLowVoltage", "specificProblem": "ps=3,sl=1,in=8", "alarmedObjectType": "Rectifier", "alarmedObject": { "id": "93051825", "href": "/resourceInventoryManagement/v4/resource/93051825" }, "sourceSystemId": "ems-1", "alarmDetails": "voltage=95", "alarmRaisedTime": "2019-07-03T03:32:17.235Z", "alarmReportingTime": "2019-07-03T03:32:17.552Z", "alarmChangedTime": "2019-07-03T03:32:52.744Z", "ackSystemId": "ems-1", "ackUserId": "bob@example.net", "ackTime": "2019-07-03T03:33:12.623Z",</pre>

<pre> "start": "2019-02-27T10:22:00Z", "lcode": "00AA00", "moc": "LOCAL", "moi": "291911108", "moiextid": "839402", "moifn": "SEMDADOSDECADASTRO", "p.inh": false, "sev": "Critical", "pcause": "IND", "phost": "pkpsga21", "pdir": "probe-dir", "raised": "2019-02-27T10:20:00Z", "prob": "shutdown", "sprob": "shutdown", "probwdet": "2912220631", "st": "Open", "subsystem": "GEO_CLI", "tags": [], "tech": "CEM", "urg": "Critical", "node": "10.112.100.38", "nodeAlias": "MACHINE_1", "summary": "SHUTDOWN CAUSED BY BLACKOUT", "shelf": "SHELF_1", "inMaintenance": false, "inInhibition": false, "extraAttributes": { "sol_lisbon": "extra_lisbon_1" } } </pre>	<pre> "ackState": "acknowledged", "isRoot": false, "parentAlarm": { "id": "8675300" }, "correlatedAlarm": [{ "id": "8675399", "href": "/alarmManagement/v4/alarm/868675399" }], "comment": [{ "userId": "bob@example.net", "systemId": "ems-1", "time": "2019-07-03T03:37:33.827Z", "comment": "Dispatched" }] } </pre>
--	---

APÊNDICE II: MAPEAMENTO DE ATRIBUTOS ENTRE O SISTEMA E A *API*

Tabela 14: Mapeamento de atributos entre o sistema e a *TMF642*.

API do sistema		Open API da TM Forum (TMF642)	
id	Sem descrição (long)	id	Identificador do alarme, determinado pelo Sistema que possui o alarme. <i>(integer)</i>
	"39359854"		"8675309"
ack	Indica se um alarme está reconhecido (acknowledged) ou não	ackState	Fornece o estado de reconhecimento de um alarme <i>(unacknowledged / acknowledged)</i> . <i>(string)</i>
	"Acknowledged"		"acknowledged"
acksystem	Sistema do utilizador que reconheceu o alarme.	ackSystemId	Fornece o nome do sistema que alterou o estado (ackState) do alarme. <i>(string)</i>
	"SSO.SS03.SS003"		"ems-1"
acktime	Data e hora de reconhecimento do alarme.	ackTime	Data e hora do reconhecimento de um alarme. <i>(DateTime)</i>
	"2022-10-26T12:28:38Z"		"2019-07-03T03:33:12.623Z"
ackuser	Identificação do utilizador que reconheceu o alarme.	ackUserId	Nome do utilizador que reconheceu o alarme. <i>(String)</i>
	"amint"		"bob@example.net"
type	Tipo de alarme.	alarmType	Categoriza um alarme. (Os valores possíveis são definidos nas normas X.733 8.1.1, 3GPP TS 32.111-2 Annex A). <i>(AlarmType)</i>
	"ProcessingError"		"environmentalAlarm"
changed	Data do Sistema da ultima alteração do alarme.	alarmChangedTime	Indica a data e hora da última alteração do alarme. Pode incluir uma alteração proveniente de um recurso ou despoletada por um cliente. <i>(DateTime)</i>

	"2022-11-29T11:04:09Z"		"2019-07-03T03:32:52.744Z"
cleared	Data e hora da alteração do estado do alarme para fechado.	alarmClearedTime	Indica a data e hora em que o alarme é fechado. (<i>DateTime</i>)
	"2019-02-27T10:18:00Z"		"2019-07-03T03:55:45.937Z"
sev	Gravidade (<i>severity</i>). (<i>Text/Enumerate: Critical, Major, Minor, Warning, Indeterminate, Cleared</i>)	perceivedSeverity	Lista os valores de gravidade possíveis de atribuir a um alarme. Estes valores são alinhados com a recomendação ITU_T X.733 (De acordo com a especificação, o nível de gravidade pode ser descrito como o nível de degradação que a falha causa no serviço. Os valores possíveis são: <i>critical, major, minor</i> e <i>warning</i>) Assim que um alarme tenha sido fechado, o nível de gravidade deve ser alterado para <i>cleared</i> . (<i>PerceivedSeverity</i>)
	"Critical"		"major"
pcause	Causa Provável (<i>Probable cause</i>)	probableCause	Fornece a causa provável de um alarme. Os valores são baseados com as normas ITU-T X.733 e 3GPP TS 32.111-2 (<i>ProbableCause</i>)
	"IND"		"rectifierLowVoltage"
raised	Data e hora da deteção do alarme.	alarmRaisedTime	Indica a data e hora em que o alarme foi detetado na fonte (<i>source system</i>) (<i>DateTime</i>)
	"2019-02-27T10:20:00Z"		"2019-07-03T03:32:17.235Z"
start	Data e hora do começo do registo do alarme no sistema. (<i>DateTime</i>)	alarmReportingTime	Indica a data e hora em que o alarme foi reportado pelo sistema de deteção de alarmes. Pode ser diferente do <i>alarmRaisedTime</i> , uma vez que o alarme ainda demora algum tempo a chegar ao sistema depois de ser disparado. (<i>DateTime</i>)
	"2019-02-27T10:22:00Z"		"2019-07-03T03:32:17.552Z"
prob	Problema específico (<i>Specific Problem</i>)	specificProblem	Fornece informação específica acerca de um alarme e da causa da sua ocorrência.

			(String)
	"Test SNMP fail"		"ps=3,sl=1,in=8"
st	Estado (State) (Text/Enumerate: Undefined, Open, Closed, ManuallyClosed)	state	Define o estado do alarme durante o seu ciclo de vida. (String: raised updated cleared).
	"Open"		"updated"
corrid	Identificador do alarme correlacionado (String/Integer)	correlatedAlarm	A list of alarm references. (AlarmRef [*])
	"129324"		[{ "id": "8675399", "href": "/alarmManagement/v4/alarm/868675399" }]
summary	Descrição do problema. (String)	alarmDetails	Contém informação complementar acerca da ocorrência de um alarme. (String)
	"Test SNMP fail 5/5 errors"		"voltage=95"
moi	Entidade (Entity) (String)	alarmedObject	Um objeto alarmado (AlarmedObject). Identifica a instância de um objeto alarmado associado ao alarme.
	"blade-web trap (10.112.100.38:161, 1.3.6.1.4.1.31126.2)"		{ "id": "93051825", "href": "/resourceInventoryManagement/v4/resource/93051825" }
moc	Tipo de entidade (String)	alarmedObjectType	Tipo (classe) de objeto manuseado associado ao alarme. (String)
	"Monitor"		"Rectifier"
origin	Sistema de origem (String)	sourceSystem / sourceSystemId	Identificador do Sistema de origem (String)
	"GEO_CLI"		"ems-1"

extkey	Chave externa (External Key), proveniente do Sistema externo. (<i>String</i>)	externalAlarmId	Um identificador do alarme no Sistema de origem. (<i>String</i>)
	"GEO_CLI00AA00CEM1291911108,2912220631"		"5551212"
Atributos que não são mencionados na TMF642		Atributos que não são mencionados na API do sistema	
ackDur	Tempo em milisegundos desde o reconhecimento do alarme (<i>Integer</i>)	alarmEscalation (Existe no sistema mas não é usado na API)	Indica se o alarme foi escalado (<i>escalated</i>) ou não. (<i>Boolean</i>)
	1107857		true
alarmDur	Duração em segundos do alarme. (<i>Integer</i>)	isRootCause or isRoot (Existe no sistema mas não é usado na API)	Indica se o alarme é uma origem do problema (<i>root cause alarm</i>). (<i>Boolean</i>)
	4040388		false
creation	Data e hora da criação de um alarme. (<i>DateTime</i>)	parentAlarm (Existe no sistema mas não é usado na API)	Lista de referências de alarmes. (<i>AlarmRef[*]</i>)
	"2022-10-13T16:44:21Z"		{ "id": "8675300" }
subsystem	Subsistema (fornecedor ou modelo) (<i>String</i>)		
	"GEO_CLI"		
dom	Domínio (<i>Domain</i>) (<i>String</i>)		
	"ALMMON"		
ev	Número de eventos (<i>events</i>). (<i>Integer</i>)		
	44311		

lcode	Código local (<i>local code</i>) (<i>String</i>)
	"undef"
moiextid	Id externo da entidade/moi. (<i>String</i>)
	"839402"
moifn	Nome "amigável" para a entidade. (<i>String</i>)
	"blade-web trap (10.112.100.38:161, 1.3.6.1.4.1.31126.2)"
phost	Máquina de coleta (<i>Collection machine</i>) (<i>String</i>)
	"almgr-dev-evl-ampp2"
pdir	Diretoria da coleção (<i>Collection Directory</i>) (<i>String</i>)
	"system-mgr-adapter"
sprob	Problema específico variável (<i>Variable Specific Problem</i>) (<i>String</i>)
	"Test SNMP fail"
probwdet	Problema detalhado (<i>String</i>)
	"Test SNMP fail 5/5 errors"
tech	Tecnologia (<i>technology - ex. 2G, 3G, GPON, ..</i>) (<i>String</i>)
	"SYS"
	Urgência de atuação (<i>Urgency of action</i>)

urg	(Text/Enumerate: <i>Critical, Major, Minor, Warning, Indeterminate</i>)
	"Critical"
node	IP of the equipment (<i>String</i>)
	"10.112.100.38"
nodeAlias	Nome de rede do equipamento (<i>hostname</i>) (<i>String</i>)
	"blade-web trap"
inMaintenance	Indicação de manutenção (<i>maintenance</i>) por parte da entidade. (<i>Boolean</i>)
	false
inInhibition	Indicação de inibição (<i>inhibition</i>) por parte da entidade. (<i>Boolean</i>)
	false
extraAttributes	Um conjunto de atributos extra (são atributos específicos exigidos pelos clientes)
	...

APÊNDICE III: PROBLEMAS ENCONTRADOS NO MAPEAMENTO DE VALORES DOS ATRIBUTOS ENUMERADOS

Tabela 15: Problemas encontrados no mapeamento de valores dos atributos enumerados

API do sistema	Open API da TM Forum (TMF642)
Nome de atributo: st	Nome de atributo: state
ManuallyClosed	-
-	updated
Nome de atributo: type	Nome de atributo: alarmType
Indeterminate	-
SecurityViolation	-
Nome de atributo: pcause	Nome de atributo: probableCause
Valores que não possuem correspondência no sistema:	
-	authenticationFailure
-	breachOfConfidentiality
-	cableTamper
-	connectionEstablishmentError
-	delayedInformation
-	demodulationFailure
-	denialOfService
-	duplicateInformation
-	externalPointFailure
-	informationMissing
-	informationModificationDetected
-	informationOutOfSequence
-	intrusionDetection
-	keyExpired
-	lossOfRealTime
-	modulationFailure
-	nonRepudiationFailure
-	outOfHoursActivity

	outOfService
	proceduralError
	protectingResourceFailure protectionMechanismFailure
	realTimeClockFailure
	reinitialized
	transmitterFailure
	unauthorizedAccessAttempt
	unexpectedInformation
Valores sem correspondência do lado da TF642:	
DSP("Data set problem")	-
DDIE("Dte dce interface error")	-
IMR("Invalid msu received")	-
TrF("Transmission failure")	-
AIS("Alarm indication signal")	-
Valores duplicados:	
-	ioDeviceError
-	responseTimeExcessive
-	retransmissionRateExcessive
-	authenticationFailure

APÊNDICE IV: MAPEAMENTO DE VALORES DOS ATRIBUTOS ENUMERADOS

Tabela 16: Mapeamento de valores dos atributos enumerados.

API do sistema	Open API da TM Forum (TMF642)	
Nome do atributo: sev	Nome do atributo: perceivedSeverity	
Cleared	cleared	
Critical	critical	
Major	major	
Minor	minor	
Warning	warning	
Indeterminate	indeterminate	
Nome de atributo: st	Nome de atributo: state	
Open	raised	
Closed	cleared	
ManuallyClosed		
-	Updated (não implementado)	
Nome de atributo: type	Nome de atributo: alarmType	
Indeterminate	unknown	<i>Unknown alarm type.</i>
Communications	communicationsAlarm	<i>A communications fault.</i>
Environmental	environmentalAlarm	<i>A condition related to an enclosure in which the equipment resides.</i>
Equipment	equipmentAlarm	<i>An equipment fault.</i>
TimeDomainViolation	timeDomainViolation	<i>An event has occurred at an unexpected or prohibited time.</i>
IntegrityViolation	integrityViolation	<i>Information may have been illegally modified, inserted or deleted.</i>
PhysicalViolation	physicalViolation	<i>A physical resource has been violated in a way that suggests a security attack.</i>

ProcessingError	processingErrorAlarm	<i>A software or processing fault.</i>
QualityOfService	qualityOfServiceAlarm	<i>A degradation in the quality of a service.</i>
SecurityViolation	mechanismViolationOrSecurityService	<i>A security attack has been detected by a security mechanism or by a security service.</i>
OperationalViolation	operationalViolation	<i>The unavailability, malfunction or incorrect invocation of a service.</i>
pcause	probableCause (X.721 Value)	
Novas <i>probableCauses</i> adicionadas ao código interno do <i>Alarm Manager</i> (identificadas em falta através da norma 3GPP TS 32.111-2 Annex-B)		
AuF ("Authentication Failure")	authenticationFailure	
BOC ("Breach of Confidentiality")	breachOfConfidentiality	
CaT ("Cable Tamper")	cableTamper	
CoEE ("Connection establishment error")	connectionEstablishmentError	
Del ("Delayed Information")	delayedInformation	
DeF ("Demodulation Failure")	demodulationFailure	
DOS ("Denial of Service")	denialOfService	
Dul ("Duplicate Information")	duplicateInformation	
EPF ("External Point Failure")	externalPointFailure	
InM ("Information Missing")	informationMissing	
IMD ("Information Modification detected")	informationModificationDetected	
IOOS ("Information out of Sequence")	informationOutOfSequence	
InD ("Intrusion Detection")	intrusionDetection	
KeE ("Key Expired")	keyExpired	
LORT ("Loss of Real Time")	lossOfRealTime	
MoF ("Modulation Failure")	modulationFailure	
NRF ("Non-Repudiation Failure")	nonRepudiationFailure	
OOHA ("Out of Hours Activity")	outOfHoursActivity	
OOS ("Out of Service")	outOfService	
PrE ("Procedural Error")	proceduralError	

PRF ("Protecting Resource Failure")	protectingResourceFailure
PMF ("Protection Mechanism Failure")	protectionMechanismFailure
RTCF ("Real Time Clock Failure")	realTimeClockFailure
Rei ("Reinitialized")	reinitialized
UAA ("Unauthorised Access Attempt")	unauthorizedAccessAttempt
UnI ("Unexpected Information")	unexpectedInformation
AuF("Authentication Failure")	authenticationFailure (valor duplicado)
Novas probableCauses adicionadas à TMF642 (identificadas em falta através da norma 3GPP TS 32-111.2 Annex-B)	
DSP("Data set problem")	dataSetProblem
DDIE("Dte dce interface error")	dteDceInterfaceError
ODE("Output device error")	outputDeviceError
IMR("Invalid msu received")	invalidMsuReceived
TrF("Transmission failure")	transmissionFailure
AIS("Alarm indication signal")	alarmIndicationSignal
Valores duplicados:	
RTE ("Response time excessive")	excessiveResponseTime
	responseTimeExcessive
RRE ("Retransmission rate excessive")	excessiveRetransmissionRate
	retransmissionRateExcessive
IODE ("Input output device error")	inputOutputDeviceError
	ioDeviceError
EqM ("Equipment malfunction")	equipmentMalfunction
	equipmentFailure
Restantes valores (valores com correspondência direta onde não foram identificados problemas):	
IND("Indeterminate")	other
CaSF("Call setup failure")	callSetUpFailure
DSM("Degraded signal M.3100")	degradedSignal
FERF("Far end receiver failure")	farEndReceiverFailure
FEM("Framing error M.3100")	framingError
LOF("Loss of frame")	lossOfFrame
LOP("Loss of pointer")	lossOfPointer
LoOS("Loss of signal")	lossOfSignal
PaTM("Payload type mismatch")	payloadTypeMismatch
TrE("Transmission error")	transmissionError

ReAI("Remote alarm interface")	remoteAlarmInterface
EBER("Excessive bit error rate")	excessiveBitErrorRate
PTM("Path trace mismatch")	pathTraceMismatch
Una("Unavailable")	unavailable
SILM("Signal label mismatch")	signalLabelMismatch
LOMF("Loss of multi frame")	lossOfMultiFrame
BPF("Back plane failure")	backplaneFailure
EID("Equipment identifier duplication")	equipmentIdentifierDuplication
EDP("External device problem")	externalDeviceProblem
LiCP("Line card problem")	lineCardProblem
MPM("Multiplexer problem M.3100")	multiplexerProblem
NEID("NE identifier duplication")	neIdentifierDuplication
PoPM("Power problem M.3100")	powerProblems
PPM("Processor problem M.3100")	processorProblem
PPF("Protection path failure")	protectionPathFailure
RFM("Receiver failure M.3100")	receiverFailure
RUM("Replaceable unit missing")	replaceableUnitMissing
RUTM("Replaceable unit type mismatch")	replaceableUnitTypeMismatch
SSM("Synchronisation source mismatch")	synchronizationSourceMismatch
TeP("Terminal problem")	terminalProblem
TPM("Timing problem M.3100")	timingProblem
TFM("Transmitter failure M.3100")	transmitterFailure
TCP("Trunk card problem")	trunkCardProblem
RUP("Replaceable unit problem")	replaceableUnitProblem
AiCF("Air compressor failure")	airCompressorFailure
ACF("Air conditioning failure")	airConditioningFailure
ADF("Air dryer failure")	airDryerFailure
BaD("Battery discharging")	batteryDischarging
BaF("Battery failure")	batteryFailure
CPF("Commercial power failure")	commercialPowerFailure
CFF("Cooling fan failure")	coolingFanFailure
EF("Engine failure")	engineFailure
FDF("Fire detector failure")	fireDetectorFailure
FuF("Fuse failure")	fuseFailure

GeF("Generator failure")	generatorFailure
LBT("Low battery threshold")	lowBatteryThreshold
PFM("Pump failure M.3100")	pumpFailure
RecF("Rectifier failure")	rectifierFailure
RHV("Rectifier high voltage")	rectifierHighVoltage
RLFV("Rectifier low f voltage")	rectifierLowVoltage
VSF("Ventilation system failure")	ventilationsSystemFailure
EDOM("Enclosure door open M.3100")	enclosureDoorOpen
ExG("Explosive gas")	explosiveGas
Fir ("Fire")	fire
Flo ("Flood")	flood
HiH ("High humidity")	highHumidity
HiT ("High temperature")	highTemperature
HiW ("High wind")	highWind
IBU ("Ice build up")	iceBuildUp
LoF ("Low fuel")	lowFuel
LoH ("Low humidity")	lowHumidity
LCP ("Low cable pressure")	lowCablePressure
LoT ("Low temperature")	lowTemperatue (erro sintatico)
LoW ("Low water")	lowWater
Smo ("Smoke")	smoke
ToG ("Toxic gas")	toxicGas
SCPM ("Storage capacity problem M.3100")	storageCapacityProblem
MeM ("Memory mismatch")	memoryMismatch
CDM ("Corrupt data M.3100")	corruptData
OOCC ("Out of cpu cycles")	outOfCpuCycles
SEP ("Software environment problem")	softwareEnvironmentProblem
SDF ("Software download failure")	softwareDownloadFailure
AdE ("Adapter error")	adapterError
ASF ("Application subsystem failure")	applicationSubsystemFailure
BaR ("Bandwidth reduction")	bandwidthReduced
CPE ("Communication protocol error")	communicationsProtocolError
CoSF ("Communication subsystem failure")	communicationsSubsystemFailure
COCE ("Configuration or customizing error")	configurationOrCustomizationError

Con ("Congestion")	congestion
CACLE ("Cpu cycles limit exceeded")	cpuCyclesLimitExceeded
DSOME ("Data set or modem error")	dataSetOrModemError
ExV ("Excessive vibration")	excessiveVibration
FiE ("File error")	fileError
HOVOCSP ("Heating or ventilation or cooling system problem")	heatingVentCoolingSystemProblem
HuU ("Humidity unacceptable")	humidityUnacceptable
IDE ("Input device error")	inputDeviceError
LaE ("Lan error")	lanError
LeD ("Leak detection")	leakDetected
LNTE ("Local node transmission error")	localNodeTransmissionError
MSE ("Material supply exhausted")	materialSupplyExhausted
OOM ("Out of memory")	outOfMemory
PeD ("Performance degraded")	performanceDegraded
PrU ("Pressure unacceptable")	pressureUnacceptable
QSE ("Queue size exceeded")	queueSizeExceeded
ReF ("Receive failure")	receiveFailure
RNTE ("Remote node transmission error")	remoteNodeTransmissionError
RAONC ("Resource at or nearing capacity")	resourceAtOrNearingCapacity
SoE ("Software error")	softwareError
SPAT ("Software program abnormally terminated")	softwareProgramAbnormallyTerminated
SPE ("Software program error")	softwareProgramError
TeU ("Temperature unacceptable")	temperatureUnacceptable
ThC ("Threshold crossed")	thresholdCrossed
TLD ("Toxic leak detected")	toxicLeakDetected
TraF ("Transmit failure")	transmitFailure
URU ("Underlying resource unavailable")	underlyingResourceUnavailable
VeM ("Version mismatch")	versionMismatch
ABTBIF ("A bis to bts interface failure")	abisBtsInterfaceFailure
ABTTIF ("A bis to trx interface failure")	abisTrxInterfaceFailure
AnP ("Antenna problem")	antennaFailure
BaB ("Battery breakdown")	batteryBreakdown
BaCF ("Battery charging fault")	batteryChargingFailure

CSP ("Clock synchronisation problem")	clockSynchronizationProblem
CoP ("Combiner problem")	combinerProblem
DiP ("Disk problem")	diskFailure
ERT ("Excessive receiver temperature")	excessiveReceiverTemperature
ETOP ("Excessive transmitter output power")	excessiveTransmitterOutputPower
ETT ("Excessive transmitter temperature")	excessiveTransmitterTemperature
FHD ("Frequency hopping degraded")	frequencyHoppingDegraded
FHF ("Frequency hopping failure")	frequencyHoppingFailure
FRF ("Frequency redefinition failed")	frequencyRedefinitionFailed
LIF ("Line interface failure")	lineInterfaceFailure
LiF ("Link failure")	linkFailure
LOS ("Loss of synchronisation")	lossOfSynchronisation
LoR ("Lost redundancy")	lossOfRedundancy
MBWBB ("Mains breakdown with battery backup")	mainsBreakdownWithBatteryBackUp
MBWoBB ("Mains breakdown without battery backup")	mainsBreakdownWithoutBatteryBackUp
PSF ("Power supply failure")	powerSupplyFailure
RAF ("Receiver antenna fault")	receiverAntennaFault
RMF ("Receiver multicoupler failure")	receiverMulticouplerFailure
RTOP ("Reduced transmitter output power")	reducedTransmitterOutputPower
SQEF ("Signal quality evaluation fault")	signalQualityEvaluationFailure
THF ("Timeslot hardware failure")	timeslotHardwareFailure
TraP ("Transceiver problem")	transceiverFailure
TrP ("Transcoder problem")	transcoderProblem
TORAP ("Transcoder or rate adapter problem")	transcoderOrRateAdapterProblem
TAF ("Transmitter antenna failure")	transmitterAntennaFailure
TANA ("Transmitter antenna not adjusted")	transmitterAntennaNotAdjusted
TLVOC ("Transmitter low voltage or current")	transmitterLowVoltageOrCurrent
TOF ("Transmitter off frequency")	transmitterOffFrequency
Dal ("Database inconsistency")	databaseInconsistency
FSCU ("File system call unsuccessful")	fileSystemCallUnsuccessful
IPOOR ("Input parameter out of range")	inputParameterOutOfRange
InvP ("Invalid parameter")	invalidParameter
InP ("Invalid pointer")	invalidPointer
MNE ("Message not expected")	messageNotExpected

MNI ("Message not initialised")	messageNotInitialized
MOOS ("Message out of sequence")	messageOutOfSequence
SCU ("System call unsuccessful")	systemCallUnsuccessful
TiE ("Timeout expired")	timeoutExpired
VOOR ("Variable out of range")	variableOutOfRange
WDTE ("Watch dog timer expired")	watchdogTimerExpired
CSF ("Cooling system failure")	coolingSystemFailure
EEF ("External equipment failure")	externalEquipmentFailure
EPSF ("External power supply failure")	externalPowerSupplyFailure
ETDF ("External transmission device failure")	externalTransmissionDeviceFailure
RAR ("Reduced alarm reporting")	reducedAlarmReporting
RER ("Reduced event reporting")	reducedEventReporting
RLC ("Reduced logging capability")	reducedLoggingCapability
SRO ("System resources overload")	systemResourcesOverload
BCF ("Broadcast channel failure")	broadcastChannelFailure
CEE ("Call establishment error")	callEstablishmentError
InMR ("Invalid message received")	invalidMessageReceived
LLPF ("LAPD link protocol failure")	lapdLinkProtocolFailure
LAI ("Local alarm indication")	localAlarmIndication
RAI ("Remote alarm indication")	remoteAlarmIndication
RoF ("Routing failure")	routingFailure
SPF ("SS7 protocol failure")	ss7ProtocolFailure

APÊNDICE V: RESULTADOS FINAIS DOS TESTES AUTOMÁTICOS (CTK)

Resultados CTK do pedido GET – Consulta de alarmes

/Alarm			
Description	This operation search for the created Alarm		
Method	GET		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm		
Mean time per request	1557ms		
Mean size per request	33.13KB		
Total passed tests	501		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200 or 206	1	0
	Instance has all mandatory attributes	38	0
	Response has alarmRaisedTime attribute	38	0
	Response has href attribute	38	0
	Response has id attribute	38	0
	Response has probableCause attribute	38	0
	Response has sourceSystemId attribute	38	0
	Response has state attribute	38	0
	Body includes value held on alarmRaisedTime	38	0
	Body includes value held on href	38	0
	Body includes value held on id	38	0
	Body includes value held on probableCause	38	0
	Body includes value held on sourceSystemId	38	0
	Body includes value held on state	38	0
	alarmRaisedTime is 2023-02-20T14:32:17Z	1	0
	href is /alarmmgr/rest/tmf/alarm/41967207	1	0
	id is 41967207	1	0
	probableCause is demodulationFailure	1	0
	sourceSystemId is ems-1	1	0
	state is raised	1	0

Figura 54: Resultado CTK – consulta de alarmes

Resultados *CTK* do pedido *GET* – Consulta de alarme pelo *id*

/Alarm/{{IDAL01}}			
Description	This operation search for one of the created Alarm		
Method	GET		
URL	http://10.102.100.100/alarmmgr/rest/tmf/alarm/41967207		
Mean time per request	1404ms		
Mean size per request	790B		
Total passed tests	20		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200	1	0
	Instance has all mandatory attributes	1	0
	Response has alarmRaisedTime attribute	1	0
	Response has href attribute	1	0
	Response has id attribute	1	0
	Response has probableCause attribute	1	0
	Response has sourceSystemId attribute	1	0
	Response has state attribute	1	0
	Body includes value held on alarmRaisedTime	1	0
	Body includes value held on href	1	0
	Body includes value held on id	1	0
	Body includes value held on probableCause	1	0
	Body includes value held on sourceSystemId	1	0
	Body includes value held on state	1	0
	alarmRaisedTime is 2023-02-20T14:32:17Z	1	0
	href is /alarmmgr/rest/tmf/alarm/41967207	1	0
	id is 41967207	1	0
	probableCause is demodulationFailure	1	0
	sourceSystemId is ems-1	1	0
	state is raised	1	0

Figura 55: Resultado *CTK* – consulta de alarme pelo *id*

Resultados *CTK* da funcionalidade de seleção

/Alarm?fields=alarmRaisedTime	
Description	This operation filter a Alarm
Method	GET
URL	http://10.10.10.100/alarmmgr/rest/tmf/alarm?fields=alarmRaisedTime
Mean time per request	1481ms
Mean size per request	3.79KB
Total passed tests	0
Total failed tests	0
Status code	200

Figura 56: Resultado *CTK* – seleção do atributo *alarmRaisedTime*

/Alarm?fields=id			
Description	This operation filter a Alarm		
Method	GET		
URL	http://10.10.10.100/alarmmgr/rest/tmf/alarm?fields=id		
Mean time per request	1316ms		
Mean size per request	2.26KB		
Total passed tests	77		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200 or 206	1	0
	Instance must have id, href and filtered attribute	38	0
	Instance has only id, href and filtered attribute	38	0

Figura 57: Resultado *CTK* – seleção do atributo *id*

/Alarm?fields=probableCause			
Description	This operation filter a Alarm		
Method	GET		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?fields=probableCause		
Mean time per request	1422ms		
Mean size per request	3.33KB		
Total passed tests	77		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200 or 206	1	0
	Instance must have id, href and filtered attribute	38	0
	Instance has only id, href and filtered attribute	38	0

Figura 58: Resultado *CTK* – seleção do atributo *probableCause*

/Alarm?fields=sourceSystemId			
Description	This operation filter a Alarm		
Method	GET		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?fields=sourceSystemId		
Mean time per request	1354ms		
Mean size per request	3.27KB		
Total passed tests	77		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200 or 206	1	0
	Instance must have id, href and filtered attribute	38	0
	Instance has only id, href and filtered attribute	38	0

Figura 59: Resultado *CTK* – seleção do atributo *sourceSystemId*

/Alarm?fields=state			
Description	This operation filter a Alarm		
Method	GET		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?fields=state		
Mean time per request	1425ms		
Mean size per request	2.9KB		
Total passed tests	77		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200 or 206	1	0
	Instance must have id, href and filtered attribute	38	0
	Instance has only id, href and filtered attribute	38	0

Figura 60: Resultado *CTK* – seleção do atributo *state*

Resultados *CTK* da funcionalidade de filtragem

/Alarm?alarmRaisedTime='{{ALARMRAISEDTIMEAL01}}'			
Description	This operation filter a Alarm		
Method	GET		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?alarmRaisedTime='2023-02-20T14:32:17Z'		
Mean time per request	1653ms		
Mean size per request	792B		
Total passed tests	20		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200 or 206	1	0
	Instance has all mandatory attributes	1	0
	Response has alarmRaisedTime attribute	1	0
	Response has href attribute	1	0
	Response has id attribute	1	0
	Response has probableCause attribute	1	0
	Response has sourceSystemId attribute	1	0
	Response has state attribute	1	0
	Body includes value held on alarmRaisedTime	1	0
	Body includes value held on href	1	0
	Body includes value held on id	1	0
	Body includes value held on probableCause	1	0
	Body includes value held on sourceSystemId	1	0
	Body includes value held on state	1	0
	alarmRaisedTime is 2023-02-20T14:32:17Z	1	0
	href is /alarmmgr/rest/tmf/alarm/41967207	1	0
	id is 41967207	1	0
	probableCause is demodulationFailure	1	0
	sourceSystemId is ems-1	1	0
	state is raised	1	0

Figura 61: Resultado *CTK* – filtragem do atributo *alarmRaisedTime*

/Alarm?id={{IDAL01}}			
Description	This operation filter a Alarm		
Method	GET		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?id=41967207		
Mean time per request	1377ms		
Mean size per request	792B		
Total passed tests	20		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200 or 206	1	0
	Instance has all mandatory attributes	1	0
	Response has alarmRaisedTime attribute	1	0
	Response has href attribute	1	0
	Response has id attribute	1	0
	Response has probableCause attribute	1	0
	Response has sourceSystemId attribute	1	0
	Response has state attribute	1	0
	Body includes value held on alarmRaisedTime	1	0
	Body includes value held on href	1	0
	Body includes value held on id	1	0
	Body includes value held on probableCause	1	0
	Body includes value held on sourceSystemId	1	0
	Body includes value held on state	1	0
	alarmRaisedTime is 2023-02-20T14:32:17Z	1	0
	href is /alarmmgr/rest/tmf/alarm/41967207	1	0
	id is 41967207	1	0
	probableCause is demodulationFailure	1	0
	sourceSystemId is ems-1	1	0
	state is raised	1	0

Figura 62: Resultado *CTK* – filtragem do atributo *id*

/Alarm?probableCause={{PROBABLECAUSEAL01}}			
Description	This operation filter a Alarm		
Method	GET		
URL	http://10.102.100.10/alarmmgr/rest/tmf/alarm?probableCause=demodulationFailure		
Mean time per request	1811ms		
Mean size per request	1.55KB		
Total passed tests	33		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200 or 206	1	0
	Instance has all mandatory attributes	2	0
	Response has alarmRaisedTime attribute	2	0
	Response has href attribute	2	0
	Response has id attribute	2	0
	Response has probableCause attribute	2	0
	Response has sourceSystemId attribute	2	0
	Response has state attribute	2	0
	Body includes value held on alarmRaisedTime	2	0
	Body includes value held on href	2	0
	Body includes value held on id	2	0
	Body includes value held on probableCause	2	0
	Body includes value held on sourceSystemId	2	0
	Body includes value held on state	2	0
	alarmRaisedTime is 2023-02-20T14:32:17Z	1	0
	href is /alarmmgr/rest/tmf/alarm/41967207	1	0
	id is 41967207	1	0
	probableCause is demodulationFailure	1	0
	sourceSystemId is ems-1	1	0
	state is raised	1	0

Figura 63: Resultado CTK – filtragem do atributo *probableCause*

/Alarm?sourceSystemId={{SOURCESYSTEMIDAL01}}			
Description	This operation filter a Alarm		
Method	GET		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?sourceSystemId=ems-1		
Mean time per request	1421ms		
Mean size per request	2.32KB		
Total passed tests	46		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200 or 206	1	0
	Instance has all mandatory attributes	3	0
	Response has alarmRaisedTime attribute	3	0
	Response has href attribute	3	0
	Response has id attribute	3	0
	Response has probableCause attribute	3	0
	Response has sourceSystemId attribute	3	0
	Response has state attribute	3	0
	Body includes value held on alarmRaisedTime	3	0
	Body includes value held on href	3	0
	Body includes value held on id	3	0
	Body includes value held on probableCause	3	0
	Body includes value held on sourceSystemId	3	0
	Body includes value held on state	3	0
	alarmRaisedTime is 2023-02-20T14:32:17Z	1	0
	href is /alarmmgr/rest/tmf/alarm/41967207	1	0
	id is 41967207	1	0
	probableCause is demodulationFailure	1	0
	sourceSystemId is ems-1	1	0
	state is raised	1	0

Figura 64: Resultado *CTK* – filtragem do atributo *sourceSystemId*

/Alarm?state={{STATEAL01}}			
Description	This operation filter a Alarm		
Method	GET		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm?state=raised		
Mean time per request	1692ms		
Mean size per request	32.29KB		
Total passed tests	488		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200 or 206	1	0
	Instance has all mandatory attributes	37	0
	Response has alarmRaisedTime attribute	37	0
	Response has href attribute	37	0
	Response has id attribute	37	0
	Response has probableCause attribute	37	0
	Response has sourceSystemId attribute	37	0
	Response has state attribute	37	0
	Body includes value held on alarmRaisedTime	37	0
	Body includes value held on href	37	0
	Body includes value held on id	37	0
	Body includes value held on probableCause	37	0
	Body includes value held on sourceSystemId	37	0
	Body includes value held on state	37	0
	alarmRaisedTime is 2023-02-20T14:32:17Z	1	0
	href is /alarmmgr/rest/tmf/alarm/41967207	1	0
	id is 41967207	1	0
	probableCause is demodulationFailure	1	0
	sourceSystemId is ems-1	1	0
	state is raised	1	0

Figura 65: Resultado *CTK* – filtragem do atributo *state*

Resultados *CTK* do pedido *POST* – Criação de alarme

/Alarm			
Description	This operation creates a Alarm		
Method	POST		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm		
Mean time per request	4.6s		
Mean size per request	790B		
Total passed tests	14		
Total failed tests	0		
Status code	201		
Tests	Name	Pass count	Fail count
	Status code is 201	1	0
	Instance has all mandatory attributes	1	0
	Response has alarmRaisedTime attribute	1	0
	Response has href attribute	1	0
	Response has id attribute	1	0
	Response has probableCause attribute	1	0
	Response has sourceSystemId attribute	1	0
	Response has state attribute	1	0
	Body includes value held on alarmRaisedTime	1	0
	Body includes value held on href	1	0
	Body includes value held on id	1	0
	Body includes value held on probableCause	1	0
	Body includes value held on sourceSystemId	1	0
	Body includes value held on state	1	0

Figura 66: Resultado *CTK* – Criação de alarme

Resultados *CTK* do pedido *PATCH* – Alteração de alarme

Patch Alarm probableCause			
Method	PATCH		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm/41967207		
Mean time per request	14.4s		
Mean size per request	791B		
Total passed tests	1		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200	1	0

Figura 67: Resultado *CTK* – Alteração da *probableCause* de um alarme

Patch Alarm perceivedSeverity			
Method	PATCH		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm/41967207		
Mean time per request	1754ms		
Mean size per request	790B		
Total passed tests	1		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200	1	0

Figura 68: Resultado *CTK* – Alteração da *perceivedSeverity* de um alarme

Patch Alarm alarmType			
Method	PATCH		
URL	http://10.10.10.10/alarmmgr/rest/tmf/alarm/41967207		
Mean time per request	1432ms		
Mean size per request	790B		
Total passed tests	1		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200	1	0

Figura 69: Resultado *CTK* – Alteração do *alarmType* de um alarme

Patch Alarm state			
Method	PATCH		
URL	http://.../alarmmgr/rest/tmf/alarm/41967207		
Mean time per request	1592ms		
Mean size per request	790B		
Total passed tests	1		
Total failed tests	0		
Status code	200		
Tests	Name	Pass count	Fail count
	Status code is 200	1	0

Figura 70: Resultado *CTK* – Alteração do *state* de um alarme