Evolução de um sistema de gestão de alarmes

Adriano Oliveira[1][0009-0001-8405-2005], Filipe Meneses[1,2][0000-0003-0575-981X]

[1] Centro ALGORITMI, Universidade do Minho, Guimarães, Portugal
[2] Instituto CCG/ZGDV, Campus Azurém, Guimarães, Portugal

Resumo. A evolução das redes de telecomunicações tem pressionado as empresas a procurar novas soluções que permitam melhorar a eficiência da gestão das redes e assegurar a satisfação dos seus clientes. Devido ao processo de gestão estar normalmente distribuído por diferentes sistemas que executam funções complementares, a interoperabilidade ainda representa um grande desafio para a sua evolução.
Neste artigo é apresentada uma prova de conceito sobre a implementação de uma solução apresentada pelo programa de *Open APIs* da *TM Forum*. Um programa que disponibiliza especificações que promovem a interoperabilidade entre sistemas através da adoção de métodos de comunicação normalizado e do alinhamento com normas de carater global. A prova de conceito é aplicada ao sistema de gestão de alarmes de uma fornecedora de serviços de telecomunicações, um sistema integrante ao processo de gestão de redes.
Com base na solução obtida são executados testes de desempenho e são tiradas algumas conclusões relativamente ao processo de implementação e à utilidade da *API* para o sistema.

Palavras-chave: Gestão de alarmes, gestão de redes, telecomunicações, *Open APIs*, interoperabilidade.

1 Introdução

As redes de telecomunicações têm crescido muito ao longo dos últimos anos. À medida que se tornam mais complexas, aumenta a probabilidade de ocorrência de falhas nas comunicações. Neste sentido são utilizados sistemas de gestão de redes, sistemas utilizados para monitorizar e gerir os recursos de forma a assegurar a operabilidade da rede [1].

A gestão de redes tornou-se um processo indispensável às empresas de telecomunicações, pois permite assegurar o funcionamento contínuo e seguro das mesmas. De forma geral, esta gestão é dividida nas áreas de desempenho, configurações e falhas [2].

A gestão de redes a nível operacional é feita através de sistemas de suporte à operação. Estes sistemas correspondem a mecanismos e ferramentas de controlo e monitorização integradas em atividades de gestão. São normalmente constituídos por uma interface e um conjunto de funcionalidades que facilitam a realização dessas atividades. Este tipo de sistemas permite que toda a informação relevante seja reunida e apresentada de forma centralizada [3].

O crescimento das redes e o aumento da sua complexidade tem influenciado a evolução dos sistemas de gestão, estes foram evoluindo no sentido de automatizar os processos e de melhorar a comunicação entre sistemas [4].

Apesar da evolução notória destes sistemas, a comunicação entre os mesmos ainda representa um grande desafio para melhorar a eficiência da gestão de redes. Estes necessitam de comunicar entre si devido a cumprirem funções complementares [2]. Neste sentido existe uma procura por melhorar a interoperabilidade dos mesmos e permitir que toda a gestão de redes seja otimizada [4].

No sentido de promover a interoperabilidade entre os sistemas foi identificado um programa de *Open APIs* lançado pela *TM Forum,* uma organização reconhecida globalmente no setor das comunicações que se baseia na cooperação entre organizações para acompanharem a transformação digital [5].

O programa de *Open APIs* tem como finalidade promover a interoperabilidade dos sistemas através da especificação de estruturas de comunicação normalizadas e da adoção de normas de carater global [6]. Este consiste num conjunto de especificações para o desenvolvimento de *APIs* do tipo *REST* que se destinam a ser implementadas nos sistemas de gestão e que prometem fornecer um novo acesso às funcionalidades dos mesmos, sem implicar que serviços já existentes sejam afetados.

Neste artigo é apresentada uma prova de conceito sobre a implementação de uma das especificações, nomeadamente a de gestão de alarmes (*TMF642*). A solução é implementada no sistema de gestão de alarmes de uma fornecedora de serviços de telecomunicações. Com base na *API* obtida é feita uma comparação com a *API* utilizada anteriormente pelo sistema, e são apresentadas algumas conclusões sobre a implementação e a sua utilidade.

2 Gestão de falhas

A gestão de falhas é um processo crucial para a deteção de anomalias no funcionamento de uma rede [7]. Este processo inclui a deteção, o isolamento e a resolução de problemas através do rastreamento de falhas [8]. As falhas podem ser definidas como eventos ou mensagens de erro enviadas pelos dispositivos ligados à rede [9]. A gestão de falhas tem como principal objetivo assegurar a operabilidade da rede, através da mesma é possível localizar, analisar, comunicar e resolver problemas de comunicação [10].

Uma das grandes complexidades desta gestão encontra-se em distinguir a importância dos eventos e encontrar a causa raiz de um determinado problema [8]. Para endereçar esta complexidade, tem-se intensificado o uso de tecnologias de informação, estas permitem melhorar a identificação e a comunicação de erros e falhas. Neste sentido têm sido adotados sistemas de gestão de alarmes que centralizam toda a informação acerca das falhas e erros ocorridos na rede e facilitam a sua análise às empresas de telecomunicações [11].

Os sistemas de gestão de alarmes têm evoluído no sentido de facilitar todo o processo de gestão. Estes começam a implementar funcionalidades que permitem automatizar algumas tarefas como a procura de alarmes através da aplicação de filtros e da correlação automática de alarmes para chegar mais facilmente às causas dos problemas [12].

Além destas funcionalidades, tem-se verificado a investigação da aplicação de modelos de *machine learning* que permitam igualmente facilitar a gestão de alarmes, através da identificação de probabilidades de correlação de alarmes e da predição de causas [13], [9].

Apesar da evolução notória destes sistemas, a interoperabilidade continua a representar uma grande limitação. Os sistemas de alarmes, tal como os restantes sistemas de gestão de rede, devido a possuírem funções complementares são geralmente integrados em sistemas com funções complementares e vice-versa [2]. A interoperabilidade permite que as organizações e os seus sistemas evoluam cooperativamente, facilitando o processo de comunicação e compreensão da sua informação [14].

2.1 Sistema de gestão de alarmes proprietário

O sistema de gestão de alarmes utilizado pertence a uma fornecedora de serviços de telecomunicações e é responsável por captar, armazenar e gerir eventos de falha nas redes dos seus clientes.

Relativamente à arquitetura do sistema, representada na figura 1, esta é composta por uma componente de captação, constituída por um *gateway* que recebe vários tipos de eventos provenientes de diferentes aparelhos ligados à rede. Os eventos são processados num *protocol adapter* e são transformados no formato consumível pelo sistema (eventos normalizados). De seguida são redirecionados para dois locais, diretamente para a *API* interna (*JDBC*) e para um filtro que seleciona eventos com caraterísticas 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 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.

API de acesso externo existente no sistema

O sistema onde foi realizada a implementação já possui uma *API* de acesso externo. Esta é utilizada pelos clientes da fornecedora de serviços de telecomunicações para consultar informação sobre os alarmes. A mesma é acedida através de pedidos *REST* e inclui os pedidos:

- *GET* - Consulta de alarmes;
- *GET* - Consulta de alarme pelo id.

Além disso, inclui as funcionalidades:

- Filtragem de alarmes;
- Seleção de atributos.

Apesar disto, esta *API* foi desenvolvida de forma "livre", não estando alinhada com nenhuma especificação de carater global. Isto implica um maior esforço e tempo despendido para que os clientes aprendam a utilizar a mesma e para a sua integração noutros sistemas de suporte à operação ou aplicações.

Visão geral sobre implementação da *TMF642* no sistema

A implementação da *API* especificada pela *TM Forum* representa uma alternativa de acesso externo que permite a outros sistemas de suporte à operação ou aplicações de cliente interagir com o sistema de gestão de alarmes de acordo com uma especificação de carater global. Esta iniciativa promove o desenvolvimento de um ecossistema digital [15], um ecossistema constituído por componentes digitais que funcionam em conjunto e de forma sustentável [16].

A figura 1 apresenta a integração da especificação da *TM Forum* na arquitetura do sistema e demonstra a comunicação de alarmes no formato especificado.

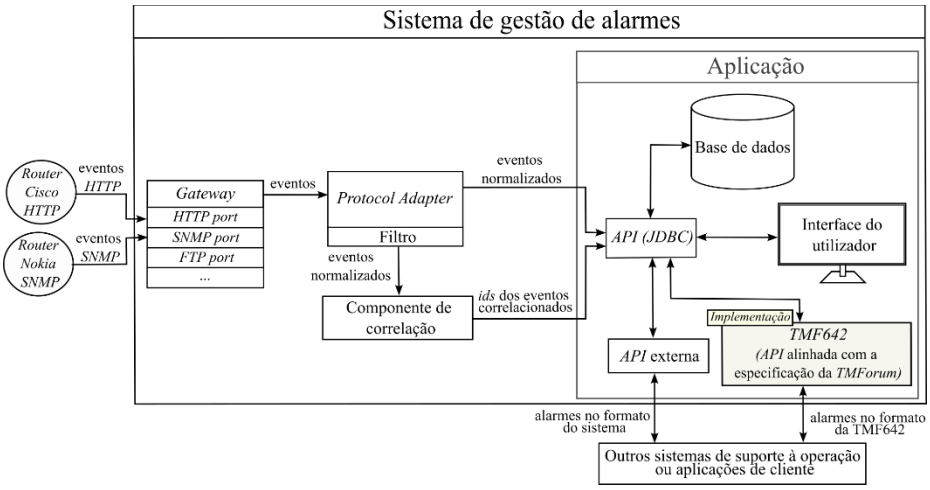


Figura 1. Arquitetura do sistema de gestão de alarmes e representação da implementação da *TMF642*.

3 Implementação da *TMF642*

O processo de implementação envolveu uma fase inicial do estudo da solução apresentada pela *TM Forum*, através da análise do guia de utilizador da *TMF642* [17].

Neste estudo foi necessário entender a compatibilidade do sistema com a solução apresentada. Para tal procedeu-se à comparação da estrutura dos alarmes, nomeadamente dos atributos que as constituem. Com base nesta comparação e com auxílio das descrições dos atributos e de conhecimentos da equipa de desenvolvimento do sistema foi estabelecido um mapeamento de atributos entre as duas soluções. Alguns exemplos do mapeamento efetuado podem ser observados na tabela 1.

Com base na correspondência entre atributos, foi verificada a existência de todos os atributos obrigatórios da *TMF642* no sistema de gestão de alarmes. Procedeu-se à criação de uma classe para instanciar alarmes nas interações entre os utilizadores da *TMF642* e o sistema, a *TMFAlarm*. Nesta classe foram adicionados todos os atributos conversíveis, resultantes do mapeamento representado na tabela 1.

Tabela 1. Exemplos de mapeamentos de atributos entre o sistema de gestão de alarmes e a *TMF642*.

Sistema (classe de alarme)			TMF642 (classe de alarme)		
Nome do atributo	Tipo de dados	Exemplo de valor	Nome do atributo	Tipo de dados	Exemplo de valor
type	Enumerado	“ProcessingError”	alarmType	Enumerado	“environmentalAlarm”
pcause	Enumerado	“IND”	probableCause	Enumerado	“rectifierLowVoltage”
prob	*String*	“Test SNMP fail”	specificProblem	*String*	“ ps=3,sl=1,in=8”
summary	*String*	“Test SNMP fail 5/5 errors”	alarmDetails	*String*	“voltage=95”
sev	Enumerado	“Critical”	perceivedSeverity	Enumerado	“major”
st	Enumerado	“Open”	state	Enumerado	“raised”
…	…	…	…	…	…

De seguida, foi efetuado o levantamento dos pedidos exigidos pela especificação e das suas funcionalidades.

Os pedidos identificados foram:

- *GET* - Consulta de alarmes – permite obter a lista de todos os alarmes existentes na base de dados.
- *GET* - Consulta de alarme pelo id – permite obter um alarme em específico através da procura pelo seu id.
- *POST* - Criação de alarme – permite proceder ao registo de um novo alarme na base de dados do sistema;
- *PATCH* - Alteração de um alarme – permite que determinados atributos de um alarme sejam alterados.

As funcionalidades identificadas foram:

- Seleção de atributos – permite selecionar os atributos que se quer visualizar em cada alarme;
- Filtragem de alarmes – permite que sejam obtidos somente alarmes com valores de atributos selecionados.

Com base nos pedidos e funcionalidades identificadas criou-se um novo módulo no código do sistema onde foi inserida a interface e a respetiva classe de implementação com os métodos necessários para processar os pedidos especificados. Em cada método foram acrescentadas as conversões necessárias para tornar a informação consumível pelo sistema e foi reutilizado o máximo de código possível, nomeadamente os métodos de processamento de alarmes já existentes.

De uma forma geral foi necessário criar métodos de conversão para:

- Atributos das classes de alarme – como o sistema usa diferentes classes de alarme para diferentes propósitos (armazenamento, criação e atualização), foi necessário criar uma forma de converter o alarme do tipo *TMFAlarm* nos diferentes tipos de alarme, de forma a assegurar que os alarmes são visualizados pelos utilizadores da *TMF642* de acordo com a especificação. Para tal foi utilizado o mapeamento de atributos representado na tabela 1;

- Valores enumerados – determinados atributos de um alarme possuem valores enumerados que têm de estar alinhados com normas de gestão de alarmes. No caso da *TMF642*, as normas utilizadas foram a *ITU-T X.733* [18] e a *3GPP TS 32.111-2* [19]. Semelhantemente ao processo de conversão dos atributos de um alarme, procedeu-se ao mapeamento de valores enumerados através da comparação entre valores do sistema e valores apresentados na documentação da *TM Forum.* Devido à existência de dificuldades no mapeamento de atributos, foi realizada uma consulta em simultâneo às normas, onde foi possível identificar mais facilmente a correspondência de valores. Além disso, também foram identificados alguns problemas como valores em falta tanto no sistema como na especificação e valores duplicados no lado da *TMF642*. Apesar disso os problemas foram contornados através de soluções acordadas com a equipa de desenvolvimento.

- Funcionalidade de filtragem – tanto a *TMF642* como o sistema possuem a funcionalidade de filtragem, no entanto na *TMF642* os filtros são recebidos a partir de parâmetros opcionais, enquanto no sistema são processados a partir de um único parâmetro que recebe uma *query* de busca. Foi necessário fazer uma conversão dos parâmetros recebidos para a *query* consumida pelo sistema.

O resultado da implementação da *TMF642* pode ser observado na figura 2 onde é apresentada uma comparação do pedido de consulta de um alarme entre a *API* existente e a nova solução. Como se pode observar, apesar dos atributos e valores serem bastante idênticos verifica-se que a *TMF642* possui atributos e valores enumerados, como o *probableCause* (correspondente ao *pcause* do lado do sistema), mais intuitivos. Estes podem ser mais facilmente identificados nas normas globais de gestão de alarmes (*ITU-T X.733, 3GPP TS 32.111-2*), sem que seja necessário verificar a sua correspondência em documentações adicionais.

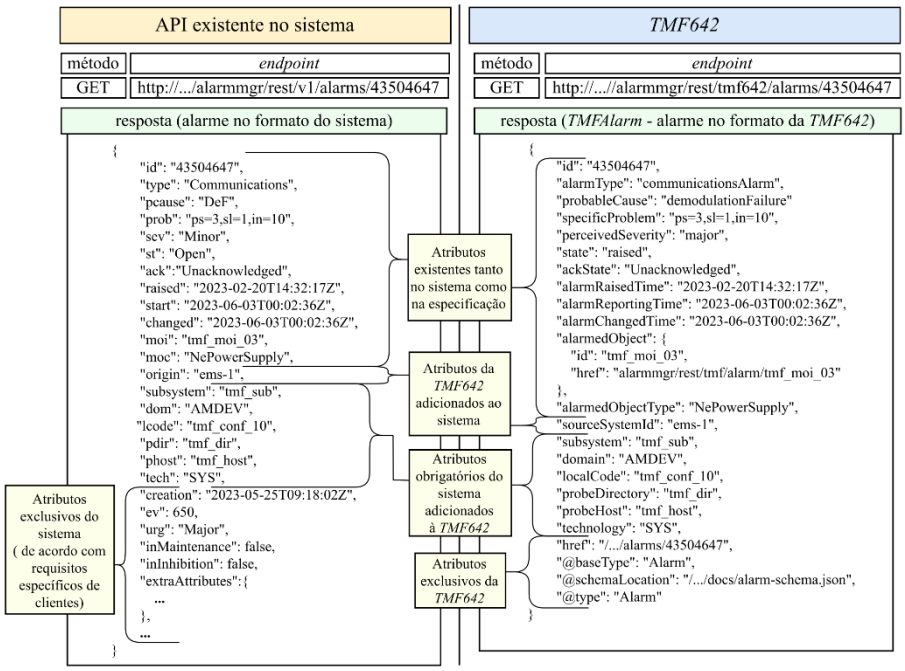


Figura 2 – Comparação de pedidos de consulta de alarme entre a *API* existente no sistema e a *TMF642*.

3.1 Testes e verificação do alinhamento

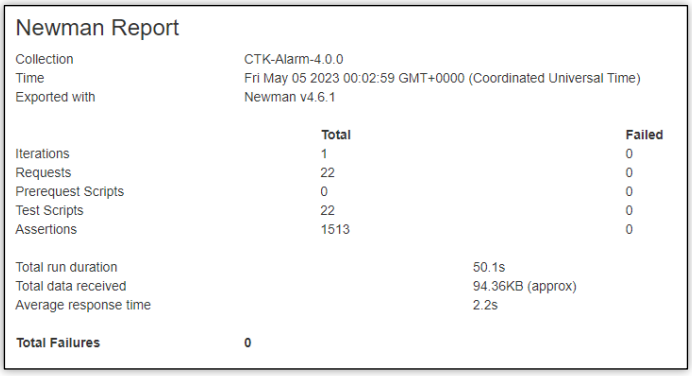
Ao longo do processo de implementação foram realizados testes para verificar o alinhamento entre a *API* desenvolvida e as especificações da *TM Forum*. Para tal recorreu-se à ferramenta *Postman* para efetuar os pedidos e procedeu-se à comparação entre os pedidos desenvolvidos e os pedidos fornecidos na documentação, nomeadamente o conteúdo do *path*, *headers*, *body* e respostas. Além disso, no caso das funcionalidades, estas foram testadas com diferentes valores e foi verificado o funcionamento correto das mesmas.

Além da verificação manual, os testes foram complementados pela utilização de um *kit* de testes automáticos (*CTK – conformance toolkit*) fornecido na tabela de *Open APIs* da *TM Forum* [20]. Este *kit* executa uma série de pedidos e funcionalidades obrigatórias de serem implementadas e emite um resultado com as falhas existentes. Este é o principal mecanismo de verificação utilizado pela *TM Forum* para assegurar a conformidade da solução com a sua especificação.

A obtenção de um resultado do *kit* sem falhas permite a emissão de um certificado de conformidade por parte da *TM Forum* que pode ser utilizado em campanhas de marketing e comunicação para atrair novos clientes. Além disso, a *TM Forum* divulga a

empresa que obtém a certificação através dos seus canais de divulgação, a fim de anunciar um novo participante alinhado com as suas especificações [21].

O resultado obtido no fim da implementação, antes da requisição do certificado é apresentado na figura 3, onde se verifica a execução dos testes automáticos sem falhas.



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
Prerequest 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 3 – Resultado dos testes automáticos (*CTK*) após a implementação da *TMF642*.

4 Testes de desempenho

Relativamente aos testes de desempenho, estes foram executados para verificar que a nova solução satisfaz os requisitos mínimos do sistema. Para esta verificação foi recomendado pela equipa de desenvolvimento ser efetuada uma comparação com a *API* existente. Devido a esta não disponibilizar pedidos de criação e de alteração, foram somente testados os pedidos de consulta.

Para a execução de testes foi utilizada a ferramenta *JMeter* que permite verificar a capacidade de resposta de um serviço, ao simular cenários em que ocorrem grandes quantidades de pedidos e concorrência entre utilizadores [22].

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 são utilizadas as seguintes configurações:

- 6 *threads* de execução (número de utilizadores concorrentes) – este valor é estipulado com base na quantidade máxima de utilizadores que a fornecedora de serviços de telecomunicações prevê que utilize a *API* em simultâneo;
- O número inicial de *threads* é 1 e é acrescentada uma a cada 40 segundos – este intervalo de entrada de *threads* é determinado para facilitar a analise das variações do tempo de resposta à medida que os utilizadores são aumentados;
- Cada *thread* executa 60 iterações a cada pedido – este valor foi determinado para que sejam executados vários registos de cada pedido, permitindo verificar variações no comportamento da(s) *API(s)* e facilitar a identificação de valores atípicos *(outliers)*.

No total foram executadas 2400 iterações, 600 iterações por pedido, num período de aproximadamente 25 minutos.

4.1 Teste de desempenho da *TMF642*

Com base nos resultados apresentados na tabela 2 e na figura 4 podemos verificar que a API possui um tempo médio de resposta de 3,8 segundos que é praticamente igual para todos os pedidos. A principal diferença entre os pedidos foi identificada na quantidade de dados recebidos na consulta de alarmes, uma vez que o pedido devolve a lista completa de alarmes armazenados no sistema.

Através do gráfico da figura 5 podemos verificar que o aumento de utilizadores teve algum impacto no tempo de resposta. Apesar disso, este situou-se predominantemente entre os 3 e os 4,5 segundos e manteve este comportamento ao longo de todo o período de teste.

Tabela 2. Tabela de resultados do desempenho da *TMF642*.

Pedido	# Iterações	Média (ms)	Mediana (ms)	% de erro	*Throughput*	KB/seg recebidos	KB/seg enviados
GET - Consultar alarmes	600	3814	3812	0,00	23,2/min	10,12	0,07
GET – Consultar alarme por id	600	3748	3745	0,00	23.2/min	0,50	0,07
POST – Criar alarme	600	3799	3794	0,00	23,2/min	0,51	0,31
PATCH – Alterar alarme	600	3833	3825	0,00	23,2/min	0,50	0,11
TOTAL	2400	3799	3796	0,00	1.5/seg	11,60	0,55

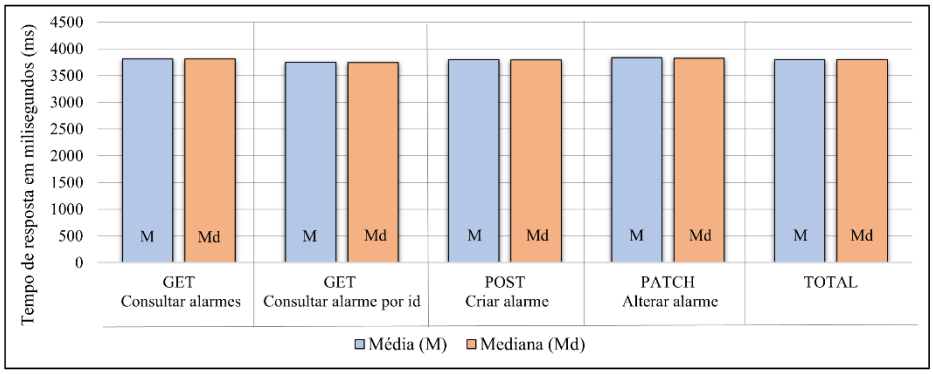


Figura 4. Gráfico de tempos de resposta da *TMF642*.

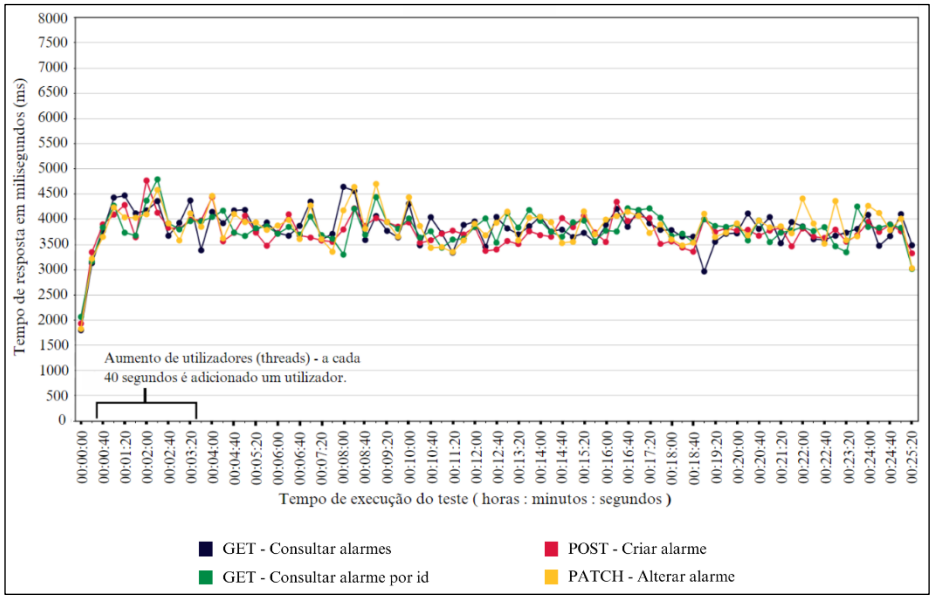


Figura 5. Gráfico de evolução do tempo de resposta ao longo do teste de desempenho da *TMF642*.

4.2 Teste comparativo entre a *API* existente no sistema e a *TMF642*

Através da tabela 3 e da figura 6 é possível observar que os tempos médios de resposta de ambas as soluções foram muito idênticos. A principal diferença entre as duas soluções é a quantidade de dados recebidos (*Received KB/sec*) no pedido de consulta de alarmes, onde a *TMF642* apresenta valores bastante menores uma vez que possui alarmes com muito menos atributos comparativamente à *API* já utilizada.

Relativamente à evolução dos tempos de resposta, representada na figura 7, inicialmente ocorreu um ligeiro aumento do tempo de resposta causado pelo aumento *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 3. Tabela de resultados do desempenho da TMF642

Pedido	# Iterações	Média (ms)	Mediana (ms)	% de erro	*Throughput*	KB/seg recebidos	KB/seg enviados
(TMF642) *GET* - Consultar alarmes	600	3717	3711	0,00	23,7/min	10,34	0,07
(API do sistema) *GET* – Consultar alarmes	600	3738	3748	0,00	23.7/min	16,54	0,07
(TMF642) *GET* – Consultar alarme por id	600	3682	3661	0,00	23,7/min	0,51	0,07
(API do sistema) *GET* – Consultar alarme por id	600	3672	3675	0,00	23,7/min	0,69	0,07
TOTAL	2400	3702	3698	0,00	1.6/seg	27,99	0,28

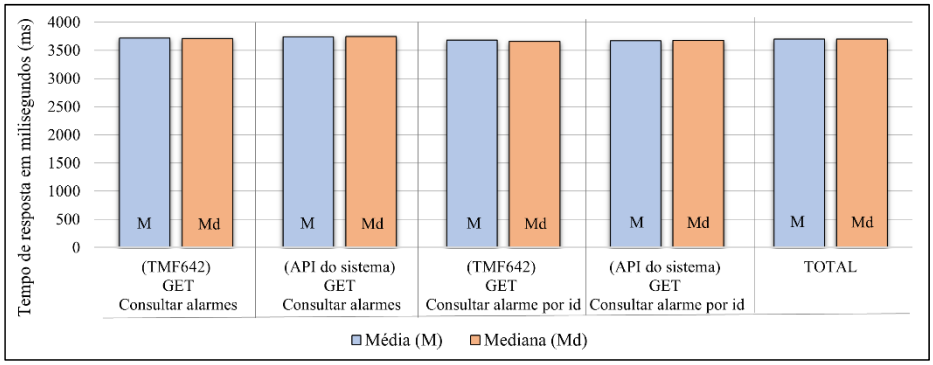


Figura 6. Gráfico comparativo dos tempos de resposta entre a *API* utilizada pelo sistema e a *TMF642*.

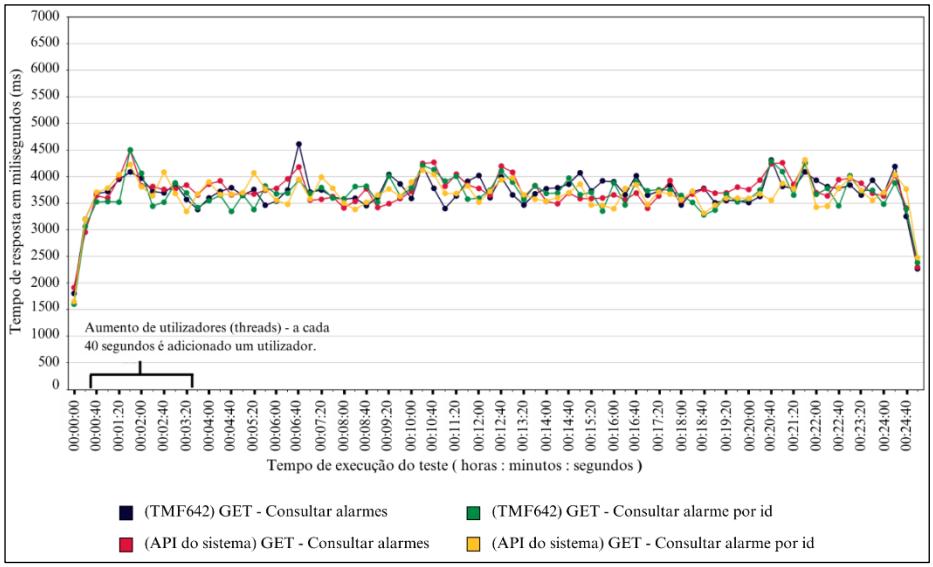


Figura 7. Gráfico comparativo da evolução dos tempos de resposta entre a *API* utilizada pelo sistema e a *TMF642* ao longo do período de execução de teste.

Conclusão sobre o desempenho da *TMF642*

Ambas as *APIs* foram submetidas às mesmas configurações de teste e a quantidade de pedidos executados por segundo foi praticamente igual em ambos os testes (representado na coluna *throughput* da tabela 2 e tabela 3). Os resultados obtidos permitiram observar que a *TMF642* possui um comportamento e tempos de resposta muito idênticos à *API* já utilizada no sistema, como tal, pode-se concluir que corresponde aos requisitos mínimos do sistema

5 Conclusões

A implementação da *TMF642* permitiu identificar algumas diferenças entre a gestão de alarmes do sistema e a especificação da *TM Forum*. O sistema pré-existente 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 *TMF642* parece ser 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 as necessidades dos seus clientes ou do negócio.

Apesar disto a especificação da *TM Forum* tem a possibilidade de ser extensível [16], o que significa que a empresa fornecedora pode acrescentar os atributos e funcionalidades extra de que necessita à solução desenvolvida, contribuindo igualmente para a evolução da mesma.

A especificação da *TM Forum* 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 performance que demostram a utilização das duas *APIs* em simultâneo.

Além disso, através dos testes de performance foi possível concluir que a *TMF642* 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 uma performance muito idêntica à já existente.

A empresa fornecedora de serviços de telecomunicações fica assim com a oportunidade de atrair clientes através de uma solução normalizada que promove a cooperação entre organizações e uma gestão mais eficiente das redes de telecomunicações. Com base nesta solução a empresa tem a oportunidade de evoluir a mesma, adequando-a à complexidade do sistema ou disponibilizar a mesma como uma alternativa mais simples e conforme necessidade dos clientes, proceder à transição para a solução mais complexa. Esta transição será facilitada pelo facto de possuir os conhecimentos acerca do mapeamento entre as duas soluções.

Bibliografia

[1] Kundan. Misra, “OSS for telecom networks : an introduction to network management,” p. 302, 2004, Accessed: Feb. 06, 2023. [Online]. Available: https://books.google.com/books/about/OSS_for_Telecom_Networks.html?hl=pt-PT&id=s7kfMfB2o6kC

[2] B. Jager, J. Doucette, and D. Tipper, “Network Survivability,” *Information Assurance*, pp. 81–112, 2008, doi: 10.1016/B978-012373566-9.50006-9.

[3] L. Tawalbeh, S. Hashish, and H. Tawalbeh, “Quality of Service requirements and Challenges in Generic WSN Infrastructures,” *Procedia Comput Sci*, vol. 109, pp. 1116–1121, Jan. 2017, doi: 10.1016/J.PROCS.2017.05.441.

[4] L. Tawalbeh, "Network Management," *The NICE Cyber Security Framework*, pp. 99–115, 2020, doi: 10.1007/978-3-030-41987-5_5.

[5] TM Forum, "About TM Forum | TM Forum," 2014. https://www.tmforum.org/about-tm-forum/ (accessed Jun. 06, 2023).

[6] "Open API Introduction - TM Forum." https://www.tmforum.org/oda/about-open-apis/ (accessed Nov. 28, 2022).

[7] R. Sturm, C. Pollard, and J. Craig, "Management of Traditional Applications," *Application Performance Management (APM) in the Digital Enterprise*, pp. 25–39, 2017, doi: 10.1016/B978-0-12-804018-8.00003-6.

[8] J. S. Baras, M. Ball, S. Gupta, P. Viswanathan, and P. Shah, "Automated network fault management," *Proceedings - IEEE Military Communications Conference MILCOM*, vol. 3, pp. 1244–1250, 1997, doi: 10.1109/MILCOM.1997.644967.

[9] M. W. Asres *et al.*, "Supporting Telecommunication Alarm Management System with Trouble Ticket Prediction," *IEEE Trans Industr Inform*, vol. 17, no. 2, pp. 1459–1469, Feb. 2021, doi: 10.1109/TII.2020.2996942.

[10] S. Hajela, " HP OEMF: Alarm management in telecommunications networks.," *Hewlett Packard Journal*, vol. 47(5), pp. 22–30, 1996.

[11] A. Salau, C. Yinka-Banjo, S. Misra, A. Adewumi, R. Ahuja, and R. Maskeliunas, "Design and implementation of a fault management system," *Advances in Intelligent Systems and Computing*, vol. 939, pp. 495–505, 2019, doi: 10.1007/978-3-030-16681-6_49/COVER.

[12] M. Hasan, B. Sugla, and R. Viswanathan, "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*, pp. 233–246, 1999, doi: 10.1109/INM.1999.770686.

[13] X. Yang, J. Lee, and H. Jung, "Regular paper 128 Fault Diagnosis Management Model using Machine Learning," *J. Inf. Commun. Converg. Eng*, vol. 17, no. 2, pp. 128–134, 2019, doi: 10.6109/jicce.2019.17.2.128.

[14] F. S. Kades, M., & Morton, "Interoperability as a competition remedy for digital networks.," *Washington Center for Equitable Growth Working Paper Series*, 2020.

[15] TM Forum, "Open APIs - TM Forum," 2022. https://www.tmforum.org/oda/implementation/open-apis/ (accessed Nov. 28, 2022).

[16] T. Yamakami, "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*, pp. 207–208, 2010, doi: 10.1145/1936254.1936291.

[17] TM Forum, "TMF642 Alarm Management API User Guide v4.0.1 | TM Forum," 2022. https://www.tmforum.org/resources/specification/tmf642-alarm-management-api-user-guide-v4-0/ (accessed Dec. 12, 2022).

[18] ITU, "X.733 : Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function," 1992. https://www.itu.int/rec/T-REC-X.733-199202-I/en (accessed Feb. 13, 2023).

[19] 3GPP, "3GPP TS 32.111-2 – Telecommunication management; Fault Management; Part 2: Alarm Integration Reference Point (IRP): Information Service (IS) – TechSpec," 2022. https://itecspec.com/archive/3gpp-specification-ts-32-111-2/ (accessed Feb. 13, 2023).

[20] TM Forum, "Open API Table - TM Forum Ecosystem API Portal - TM Forum Confluence," 2023. https://projects.tmforum.org/wiki/display/API/Open+API+Table (accessed May 04, 2023).

[21] TM Forum, "TM Forum Open API Conformance Overview | TM Forum," 2022. https://www.tmforum.org/conformance-certification/open-api-conformance/ (accessed Dec. 22, 2022).

[22] J. Wang and J. Wu, "Research on Performance Automation Testing Technology Based on JMeter," *2019 International Conference on Robots & Intelligent System (ICRIS)*, pp. 55–58, Jun. 2019, doi: 10.1109/ICRIS.2019.00023.