Integração de Sistemas

Service Oriented Middleware for Interoperability and Open Data

Miguel Crespo   
Engenharia Informática  
Politécnico de Leiria - ESTGLeiria, Portugal  
2222046@my.ipleiria.pt

José Delgado  
Engenharia Informática  
Politécnico de Leiria - ESTGLeiria, Portugal  
2222049@my.ipleiria.pt

João Tendeiro  
Engenharia Informática  
Politécnico de Leiria - ESTGLeiria, Portugal  
2222047@my.ipleiria.pt

Marisa Maximiano  
Engenharia Informática  
Politécnico de Leiria - ESTGLeiria, Portugal  
marisa.maximiano@ipleiria.pt

Bernardo Lopes  
Engenharia Informática  
Politécnico de Leiria - ESTGLeiria, Portugal  
2222048@my.ipleiria.pt

Nuno Costa  
Engenharia Informática  
Politécnico de Leiria - ESTGLeiria, Portugal  
nuno.costa@ipleiria.pt

*Abstract*

A fragmentação de soluções IoT em estruturas isoladas limita a interoperabilidade e dificulta a partilha de dados. O projeto SOMIOD propõe um middleware inovador que uniformiza o acesso e a escrita de dados na IoT, promovendo interoperabilidade e simplificando operações CRUD para recursos como aplicações, containers, registos e notificações. Além disso, o SOMIOD facilita a descoberta de hierarquias de recursos e oferece mensagens de erro detalhadas para suporte ao desenvolvedor, optando por um design que reduz a complexidade do processo. A arquitetura e as funcionalidades do SOMIOD foram validadas num cenário de teste, confirmando a sua eficiência como solução prática no contexto IoT.

# Introduction

Na perspetiva da Internet das Coisas (IoT), a fragmentação de soluções em “estruturas isoladas” compromete a interoperabilidade e a partilha de dados.

O projeto SOMIOD visa superar essas limitações, ao introduzir um middleware que estipula de forma padrão o acesso a escrita de dados na IoT.

* Este projeto tem como o objetivo definir uma arquitetura de middleware para uniformizar operações IoT.
* Implementar operações CRUD para recursos como *applications*, *containers*, *records* e *notifications*.
* Facilitar a descoberta de recursos através de operações específicas na API RESTful.
* Demonstrar a aplicabilidade prática do SOMIOD num cenário de teste no contexto IoT.

# System Architecture

A seguinte arquitetura foi uma peça fundamental para entender o funcionamento do projeto SOMIOD.

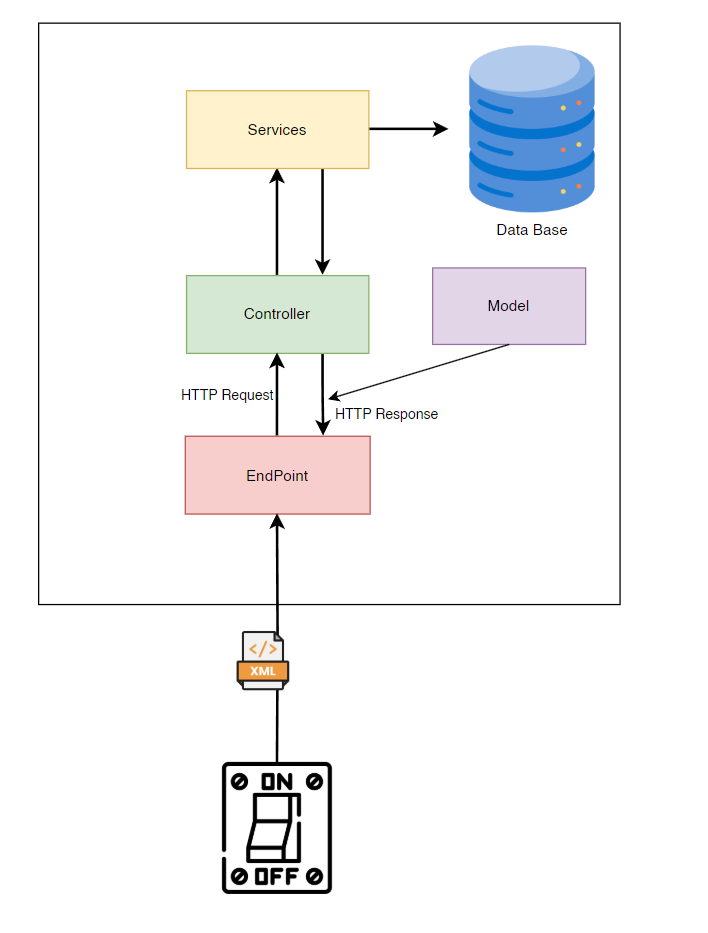


Figure 1 - Arquitetura do SOMID

## Middleware SOMIOD

O nosso middleware SOMIOD é a peça central deste cenário, pois atua como o elo de integração entre dispositivos IoT, bem como qualquer aplicação que um desenvolvedor possa criar que cumpra as suas normas, promovendo interoperabilidade. O SOMIOD dá suporte a operações CRUD para os múltiplos recursos (application, container, record e notification). É nele que é feita a serialização dos dados em XML, a persistência dos dados na base de dados (relacional) e a criação de notificações por HTTP e MQTT.

O SOMIOD é fácil de utilizar porque garantimos que o desenvolvedor recebe mensagens de erro detalhadas, para dar o máximo suporte durante o processo de desenvolvimento.

Além disso, optámos por utilizar o nó raiz do XML para identificar o recurso a que o desenvolvedor se está a referir, em vez de exigir o atributo **res\_type** no header, como mencionado no enunciado. Dessa forma, eliminamos a necessidade de passar esse atributo adicional dentro do corpo do nosso recurso e termos de fazer a separação no XML do **res\_type** da restante informação do recurso. Com esta abordagem conseguimos reduzir alguma da complexidade e tornámos o processo o mais intuitivo e eficiente para o desenvolvedor.

A validação dos dados na comunicação é fundamental para o bom funcionamento do SOMIOD, para isso, desenvolvemos um XSD para estipular a estrutura os tipos de dados e as restrições dos elementos XML de cada recurso e criámos um método no HandlerXML para validar os dados que vêm no body do pedido de acordo com o que foi definido no XSD.

O processo de resposta aos pedidos feitos ao nosso SOMIOD, passam quase sempre pelo nosso HandlerXML, onde é criada uma mensagem detalhada de acordo com o recurso que está a ser solicitado pelo desenvolvedor, este estilo de respostas detalhadas e personalizadas está também presente nas mensagens de erro o que permite uma melhor compreensão por parte do desenvolvedor durante o processo de desenvolvimento.

Para além dos requisitos do enunciado, para facilitarmos o processo de desenvolvimento criámos um endpoint (https://localhost:44322/api/somiod/{name}/parent) que ajuda o desenvolvedor a descobrir as hierarquias existentes entre os recursos do middleware, no somiod-locate é indicado o recurso para o qual queremos obter a hierarquia, caso seja um container, é retornado o nome da aplicação cujo o filho foi o container passado no url, caso seja um record ou uma notification, é retornado o nome do container a que o recurso(record ou notification) pertence, bem como, o nome da aplicação a que o container encontrado pertence.

Não foi requerido que desenvolvêssemos os endpoints de modificação (update) para a notification, mas para complementar toda a implementação, decidimos criar um endpoint para que os desenvolvedores consigam gerir quais as notificações ativas num respetivo container. Este endpoint aceita um elemento xml com o atributo “enabled”, que poderá ter o valor 1 para ativada ou o valor zero para desativada. Acreditamos que esse endpoint será sem dúvida, uma grande vantagem para o desenvolvedor.

# Evaluation

## Plataforma de Testes

De forma a testar a api, considerámos viável no nosso ambiente de testes a utilização da aplicação/ferramenta para desenvolvedores *POSTMAN*.

## Análise de Dados

Através da ferramenta supracitada, e de consultas simples à base de dados pudemos confirmar a persistência dos dados, bem como a sua análise, com o objetivo de validar e confirmar o bom funcionamento do software

# Integration/App Development

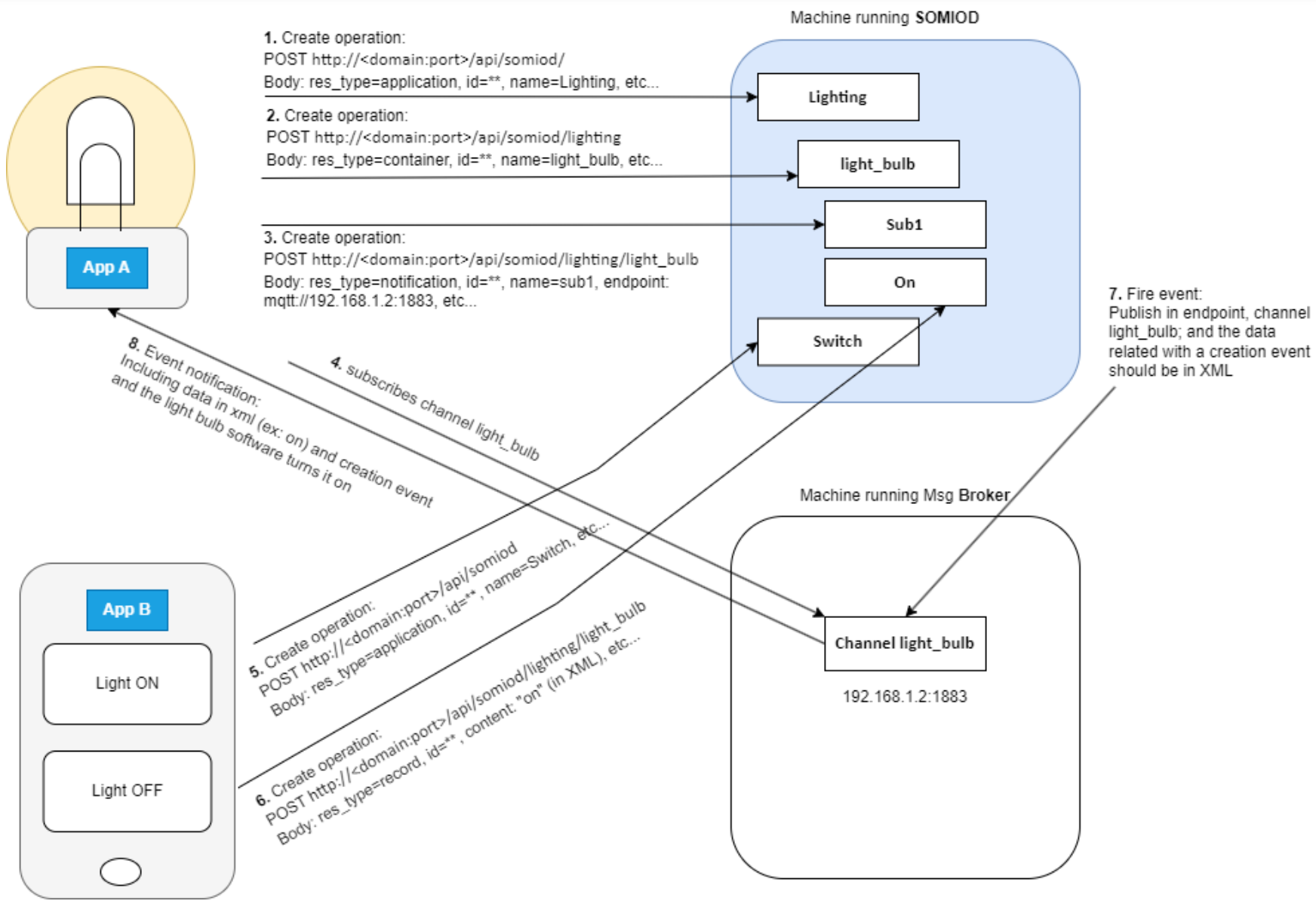


Figure 2 – Fluxo da aplicação

## App A

A aplicação Subscriber é uma solução que comunica com o middleware SOMIOD para monitorizar eventos e visualizar notificações relacionadas com o estado de um dispositivo de iluminação denominado de light\_bulb. Através de uma interface web, permite visualizar e acompanhar, em tempo real, as mensagens enviadas pelo sistema.

Ao ser iniciada, a aplicação verifica automaticamente se a aplicação Lighting já existe no middleware SOMIOD. Caso não exista, é criada. Em seguida, é verificado se o container light\_bulb e as notificações associadas (sub\_mqtt e sub\_http) estão configurados. Caso algum destes recursos não exista, a aplicação trata da sua criação automaticamente.

A interface principal exibe o estado atual do dispositivo light\_bulb, indicando se a lâmpada está "Ligada" ou "Desligada". Este estado é atualizado dinamicamente com base nos eventos recebidos do servidor.

Adicionalmente, a aplicação apresenta dois retângulos principais que exibem mensagens relacionadas com o dispositivo:

* Mensagens MQTT: Apresenta os eventos recebidos via protocolo MQTT, publicados no canal associado ao container light\_bulb (Post record).
* Mensagens HTTP: Exibe os eventos recebidos via notificações HTTP, enviados diretamente ao endpoint configurado na aplicação (Delete record).

Estas mensagens apresentam informações detalhadas, como o identificador do record, o conteúdo e a data de criação, permitindo o acompanhamento completo dos eventos gerados pelo dispositivo.

**Nota:** Para a inicialização da app é apenas necessário ter o python na versão 3.11.4, as dependências são instaladas automaticamente (127.0.0.1:5000).

Uma imagem com texto, captura de ecrã, logótipo, design gráfico

Descrição gerada automaticamente

Figure 3 - Interface da Aplicação A

## App B

A aplicação é um Publisher que faz pedidos ao servidor para gerir records e controlar o estado de um dispositivo de iluminação denominado por light\_bulb. Permite criar, visualizar e eliminar records relacionados com este dispositivo.

Ao ser iniciada, é verificado automaticamente se a aplicação “Switch” existe no servidor. Caso não exista, é criada através de uma requisição POST para o endpoint correspondente.

Após esta verificação, a aplicação carrega todos os records existentes relacionados com o dispositivo light\_bulb através de uma requisição GET ao servidor. Os records obtidos são exibidos na interface gráfica e o utilizador pode eliminar records específicos selecionando-os na lista apresentada e, de seguida, clicando no botão "Delete".

Esta funcionalidade foi criada com o objetivo de remover dados antigos, evitando a sobrecarga de informação desnecessária no servidor.

A interface possui dois botões principais que permitem interagir com o dispositivo de iluminação. O botão "Light On" cria um record que indica que o dispositivo foi ligado, enviando uma requisição POST ao servidor com o estado on. Da mesma forma, o botão "Light Off" cria um record que indica que o dispositivo foi desligado, enviando uma requisição POST com o estado off. Após a criação de cada record, a lista é atualizada automaticamente.

A aplicação comunica com o servidor através da serialização dos dados em formato XML tanto para envio como para receção de dados. Adicionalmente, nas requisições GET para carregar os records, é utilizado o cabeçalho somiod-locate: record.

Uma imagem com texto, captura de ecrã, Retângulo, Tipo de letra

Descrição gerada automaticamente

Figure 4 - Interface da Aplicação B

## Swagger

O Swagger é uma ferramenta amplamente utilizada para documentar, projetar, testar e consumir APIs REST. Tem como principal objetivo facilitar o desenvolvimento de software ao permitir que o desenvolvedor compreenda, teste e interaja com a sua API de forma mais eficiente.

Em questões de funcionamento, é disponibilizada uma interface web com uma lista de todos os endpoints da API, basta escolher um e clicar no botão “Try it out”, que os campos requeridos já se encontram pré preenchidos.

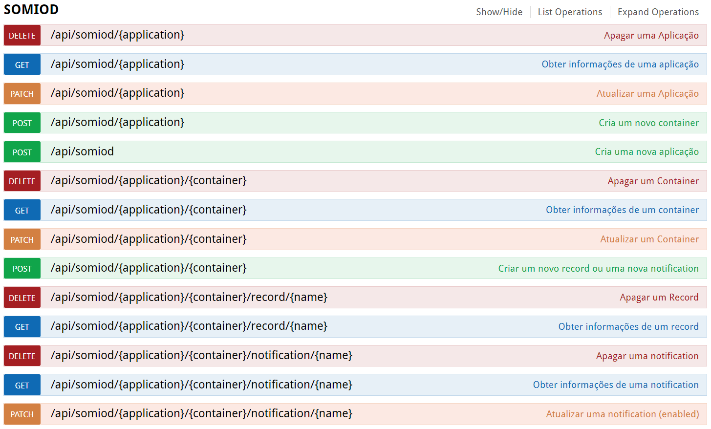


Figure 5 - Swagger UI

Para facilitar o teste dos locates, criamos uma solução alternativa à interface padrão do Swagger. Essa funcionalidade permitiu-nos testar o locate de todos os recursos de forma prática e intuitiva, sem depender da UI do Swagger.

Tomámos esta iniciativa porque, no Swagger, a inserção de um cabeçalho necessário para testar os diferentes locate’s não era muito amigável, muitas vezes exigia o uso de comandos curl. Com o objetivo de simplificar e otimizar o desenvolvimento, implementamos esta nova funcionalidade, que torna o processo de teste dos locates mais acessível e eficiente.

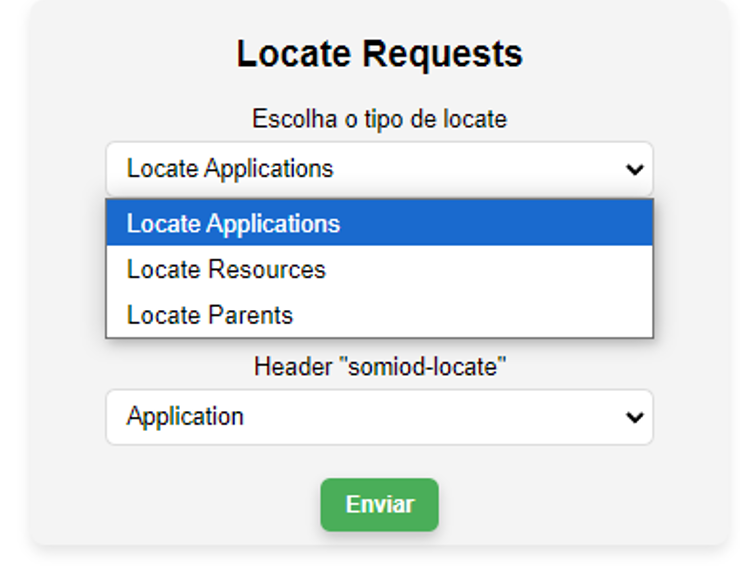


Figure 6 - Locate UI

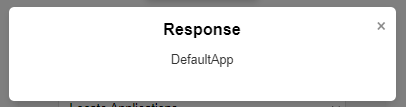


Figure 7- Resposta Locate-Application

# Conclusions and Future Work

Finalizado o Projeto de Integração de Sistemas, podemos afirmar que com ele solidificámos a nossa compreensão sobre a arquitetura de um middleware e componentes inerentes que moldaram a solução.

Este projeto não apenas fortaleceu a nossa compreensão prática na área de integração de sistemas, como também nos permitiu desenvolver *softskills* (trabalho em equipas, comunicação, gestão de versões, organização...).

Em resumo, o SOMIOD não é apenas uma solução em prol da unidade curricular Integração de Sistemas, mas também um exemplo prático de como esta contribui para um futuro mais inteligente e automatizado. Este projeto é uma base sólida para enfrentar desafios no futuro, tendo contribuído para a nossa experiência académica e profissional.

# Bibliography

ASPNET Core. *Web API help pages using Swagger*. Disponível em: <https://aspnetcore.readthedocs.io/en/stable/tutorials/web-api-help-pages-using-swagger.html>. Acesso em: 22 dez. 2024.

# Appendix

*Appendix A*

## CRUD Application Resource

1. Read

curl -X GET "https://localhost:44322/api/somiod/App1"

2. Create

curl -X POST "https://localhost:44322/api/somiod"

-H "Content-Type: application/xml"

-d "<Application>

<Name>App1</Name>

</Application>"

3. Update

curl -X PATCH "https://localhost:44322/api/somiod/App1"

-H "Content-Type: application/xml"

-d "<Application>

<Name>App2</Name>

</Application>"

4. Delete

curl -X DELETE "https://localhost:44322/api/somiod/App2"

## CRUD Container Resource

1. Read

curl -X GET "https://localhost:44322/api/somiod/App1/Cont1"

2. Create

curl -X POST "https://localhost:44322/api/somiod/App1"

-H "Content-Type: application/xml"

-d "<Container>

<Name>Cont1</Name>

</Container>"

3. Update

curl -X PATCH "https://localhost:44322/api/somiod/App1/Cont1"

-H "Content-Type: application/xml"

-d "<Container>

<Name>Cont2</Name>

<Parent>38</Parent>

</Container>"

4. Delete

curl -X DELETE "https://localhost:44322/api/somiod/App1/Cont1"

## CRUD Record Resource

1. Read

curl -X GET

"https://localhost:44322/api/somiod/App1/Cont1/record/Record1"

2. Create

curl -X POST "https://localhost:44322/api/somiod/App1/Cont1"

-H "Content-Type: application/xml"

-d "<Record>

<Name>Record1</Name>

<Content>On</Content>

</Record>"

3. Delete

curl -X DELETE

"https://localhost:44322/api/somiod/App1/Cont1/record/Record1"

## CRUD Notification Resource

1. Read

curl -X GET "https://localhost:44322/api/somiod/App1/Cont1/notification/Not1"

2. Create

curl -X POST "https://localhost:44322/api/somiod/App1/Cont1"

-H "Content-Type: application/xml"

-d "<Notification>

<Name>Not1</Name>

<Event>1</Event>

<Endpoint>mqtt://example.com</Endpoint>

<Enabled>true</Enabled>

</Notification>"

3. Update

curl -X PATCH

"https://localhost:44322/api/somiod/App1/Cont1/notification/Not1"

-H "Content-Type: application/xml"

-d "<Notification>

<Enabled>0</Enabled>

</Notification>"

4. Delete

curl -X DELETE

"https://localhost:44322/api/somiod/App1/Cont1/notification/Not1"

## Locate

1. Locate Applications

curl -X GET "https://localhost:44322/api/somiod"

-H "somiod-locate: Application"

2. Locate containers, record ou notifications de uma aplicação

curl -X GET "https://localhost:44322/api/somiod/App1"

-H "somiod-locate: container"

3. Locate pais de um containers, record ou notifications

curl -X GET "https://localhost:44322/api/somiod/Not1/parent"

-H "somiod-locate: notification"

*Appendix B*

Miguel Crespo - 2222046

Middleware: CRUD+Locate operations for Application resource

Testing applications: Application Scenario 1 (App A)

João Tendeiro - 2222047

Middleware: CRUD+Locate operations for Container resource

Testing applications: Application Scenario 1 (App B)

Bernardo Lopes – 2222048

Middleware: CRUD+Locate operations for Record resource

Testing applications: Application Scenario 2

José Delgado – 2222049

Middleware: CRUD+Locate operations for Notification resource

Testing applications: Application Scenario 2

Report:

O report foi desenvolvido por todos os membros do grupo tendo cada um escrito a respetiva implementação.