SOMIOD: Service Oriented Middleware for Interoperability and Open Data

Project Report

Alberto Pingo  
2202145@my.ipleiria.pt

Francisco Marques

2211059@my.ipleiria.pt

Diogo Moreira

2211051@my.ipleiria.pt

Rodrigo Sousa  
*2191243@my.ipleiria.pt*

*Abstract*—This document consists of a report about the SOMIOD: Service Oriented Middleware for Interoperability and Open Data project, developed by a group of 4 students enrolled in the “Integração de Sistemas” course of the Computer Engineering Bachelor of the Polytechnic Institute of Leiria.

# Introduction

The Internet of Things (IoT) concept emerges as a potential and distinctive contribution to innumerable societal issues. These include the contemporary challenges surrounding climate changes, energy prodution and consumption, and broad sustainability concerns, among others. From our smart homes and workspaces to the landscape of smart cities, the effortlessly gathered data holds promise for societal benefits, impacting not only the general population but also the business sector and local as well as state governments. This surge in data availability fosters the creation of new applications and pioneering services designed to propel social harmony, environmental betterment, and economic prosperity, thereby contributing to the realization of a more developed society.

As nowadays, a significant part of the proposed and already developed IoT solutions leans heavily on proprietary protocols and secluded cloud services, the focal point of this project is to create a service-oriented middleware proficient in standardizing the approach to data access, writing, and notification, irrespective of the application domain. The overarching aim is to champion interoperability and the practice of open data, ensuring uniformity in how data is accessed and written, and how applications are crafted. This standardized paradigm empowers individuals to tap into data, broaden capabilities, and create new applications and services, always following the same approach. To materialize this vision, we adopted Web Services and a Web-based resource structure, firmly grounded in established web standards.

To achieve our goals, the following resources were created: applications, containers, data, and subscriptions, all organized in a hierarchy. Each resource type has his unique properties.In the following sections of this document the System Architecture will be presented, with all of included components being explained in detail.

This document will also include an Evaluation section where the operation of the testing application will be described, a Conclusion section and an Appendix including the cRUD commands for crud operations for all supported resources and a section including the work done by each group member.

# System Architecture

In this section we present a diagram of the system architecture, followed by the description of each of his components, all represented by a different letter.

(This section must introduce the system architecture diagram, present the diagram and then detail every component of the diagram. Delete this.)

Uma imagem com texto, captura de ecrã, diagrama, Paralelo

Descrição gerada automaticamente

1. Example of a figure caption. (*figure caption*)

# Evaluation

# Integration/App Development

So we integrated with this API two apps:

## SmartShuttersApp

In this application we can check the status of a certain shutter decided by us. We can choose which shutter we are accessing or if we have a new one you can also create it. It receives the data from the switch and appears to the user..

## SwitchApp

In this application we can control our shutter to go up or down depending on what we need and want. It can choose any shutter that we have available(you can add more shutters via the SmartShuttersApp).

# Conclusions and Future Work

In summary, the primary goal of this project was to create SOMIOD, a middleware for standardized data access, writing, and notification in the Internet of Things (IoT). We successfully structured resources like applications, containers, data, and subscriptions, each with unique properties. The system supports CRUD operations through a RESTful API, ensuring a consistent approach to data management. By minimizing reliance on private protocols, SOMIOD addresses interoperability challenges, fostering open data practices.

For users and the community, SOMIOD brings standardization, ensuring a consistent approach to IoT data access. It enhances interoperability, promoting collaboration in diverse IoT environments. Additionally, it simplifies application development through a unified structure and RESTful API.

Looking ahead, potential improvements include enhanced security measures to protect sensitive IoT data, extended support for additional communication protocols, optimization for scalability, a user-friendly interface for easy resource management, and exploration of advanced notification mechanisms. Addressing these aspects will contribute to SOMIOD's evolution into a more robust and versatile solution, further advancing standardized practices in the IoT landscape.

# Appendix

*Appendix A*

Create Application:

POST http://localhost:65252/api/somiod

Content-Type: application/xml

<EntityRequest xmlns="Middleware-d26">

<res\_type>application</res\_type>

<name>NewApplication3</name>

</EntityRequest>

Get Application by name:

GET http://localhost:65252/api/somiod/NewApplication

Accept: application/xml

Modify Application name:

PUT http://localhost:65252/api/somiod/NewApplication

Content-Type: application/xml

<EntityRequest xlmns=”Middleware-d26”>

<res\_type>application</res\_type>

<name>ModifiedApplication</name>

</EntityRequest>

Delete Application:

DELETE <http://localhost:65252/api/somiod/NewApplication>

Create new Container:

POST http://localhost:65252/api/somiod/App1

Content-Type: application/xml

<EntityRequest xlmns=”Middleware-d26”>

<res\_type>container</res\_type>

<name>NewContainer</name>

</EntityRequest>

Get container by name:

GET http://localhost:65252/api/somiod/App1/NewContainer

Accept: application/xml

Modify Container name:

PUT http://localhost:65252/api/somiod/App1/NewContainer

Content-Type: application/xml

<EntityRequest xlmns=”Middleware-d26”>

<res\_type>container</res\_type>

<name>ModifiedContainer</name>

</EntityRequest>

Delete Container:

DELETE <http://localhost:65252/api/somiod/App1/ModifiedContainer>

Create Subscription:

OST http://localhost:65252/api/somiod/App1/Container1

Content-Type: application/xml

<EntityRequest xlmns=”Middleware-d26”>

<res\_type>subscription</res\_type>

<subscription>

<name>NewSubscription</name>

<event>YourEvent</event>

<endpoint>YourEndpoint</endpoint>

</subscription>

</EntityRequest>

Get Subscription by name:

GET http://localhost:65252/api/somiod/App1/Container1/sub/NewSubscription

Accept: application/xml

Delete Subscription:

DELETE

http://localhost:65252/api/somiod/App1/Container1/sub/NewSubscription

Create Data:

POST http://localhost:65252/api/somiod/App1/Container1

Content-Type: application/xml

<EntityRequest xlmns=”Middleware-d26”>

<res\_type>data</res\_type>

<data>

<name>NewData</name>

<content>YourContent</content>

</data>

</EntityRequest>

Get Data:

GET http://localhost:65252/api/somiod/App1/Container1/data/NewData

Accept: application/xml

Delete Data:

DELETE <http://localhost:65252/api/somiod/App1/Container1/data/NewData>

Discover Applications:

GET http://localhost:65252/api/somiod

Accept: application/xml

somiod-discover: application

Discover Subscriptions:

GET <http://localhost:65252/api/somiod/App1> Accept: application/xml

somiod-discover: container

Discover Datas

GET http://localhost:65252/api/somiod/App1/Container1

Accept: application/xml   
somiod-discover: data

*Appendix B*

Alberto Pingo –

- Discover

- Container CRUD

- Data CRUD

- Database, Migration and Seeds

- Models

- Routing

- XML Validation and Schema

- MessageHandler

Diogo Moreira –

- Subscription CRUD

- I, II and V sections of the report

Francisco Marques –

- Application CRUD

- SmartShutterApp

- CRUD commands in Report

Rodrigo Sousa –

- Data CRUD

- Switch APP