Institut National des Sciences Appliquées de Toulouse
135 Avenue de Rangueil, 31400 Toulouse, France

DEPARTMENT OF
COMPUTER AND ELECTRICAL ENGINEERING

REPORT

Automation of the INSA's rooms management

Service Oriented Architecture

SUBMITTED BY

Thomas ZENNARO
Victor LE ROCH
(Valentin LICINI)

UNDER THE GUIDANCE OF

Prof. Nawal GUERMOUCHE

(Academic Year: 2020-2021)

January 3, 2021

Contents

1 Introduction

The objective is to develop a Web application (Proof-of-Concept) for managing INSA's rooms. We decided to focus on GEI's rooms. This application must allow some actions inside a room without the intervention of a human. For instance, our application must be able to open and close of doors, windows, turn on/off room lighting... For that, we need to implement services for the different sensors and actuators. The goal is to retrieve data from sensors and analyze them to enable setting actuators.

2 Software architecture and design

2.1 Global Architecture

Here, we introduce the elements that composed our architecture. Indeed, through web services, we periodically collect data from light, temperature and presence sensors. Then, an analysis of the data according to the selected scenarios is carried out. Depending on the results of the sensors, the condition of the actuators concerned (windows, door, alarm, switch) is changed. We can view in real time the current parameters of the selected part, through a dashboard (web page). As it is a PoC, we decided to create our different sensors and actuators on the OM2M platform (see Figure 2). A summarize of the architecture is given in Figure 1.

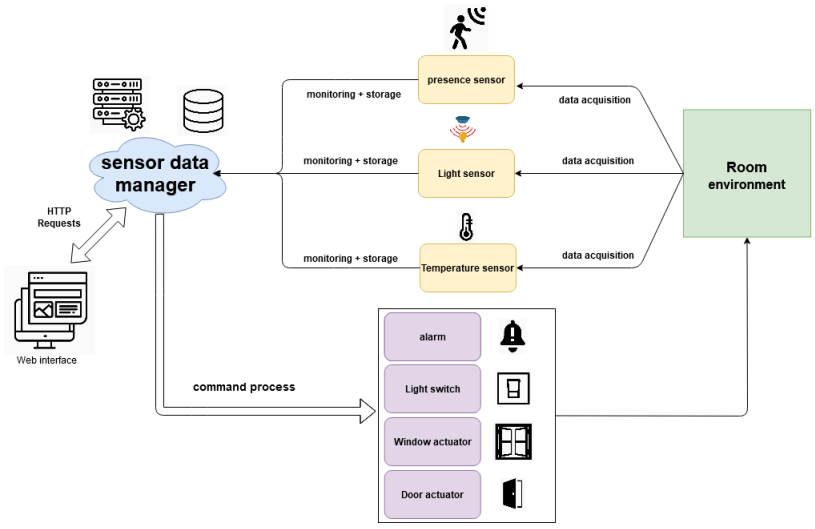

Figure 1: Global architecture

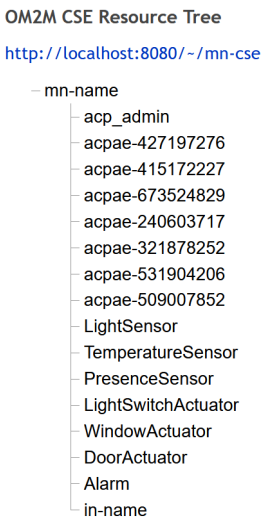

Figure 2: Sensors and actuators of the room in OM2M

2.2 JIRA Organisation

As suggested by our professor, we used JIRA to monitor our project management, using the agile methodology.

Firstly, we defined user stories for our first scenario. These stories are then decomposed in small tasks than can be achieved by one of our group members. When all tasks of the same

story are completed, we gather each part to have a working story. When all stories are completed, a scenario is created with multiple behaviours depending of the environment state.

Here is an example of our scenario1, with its stories and tasks :

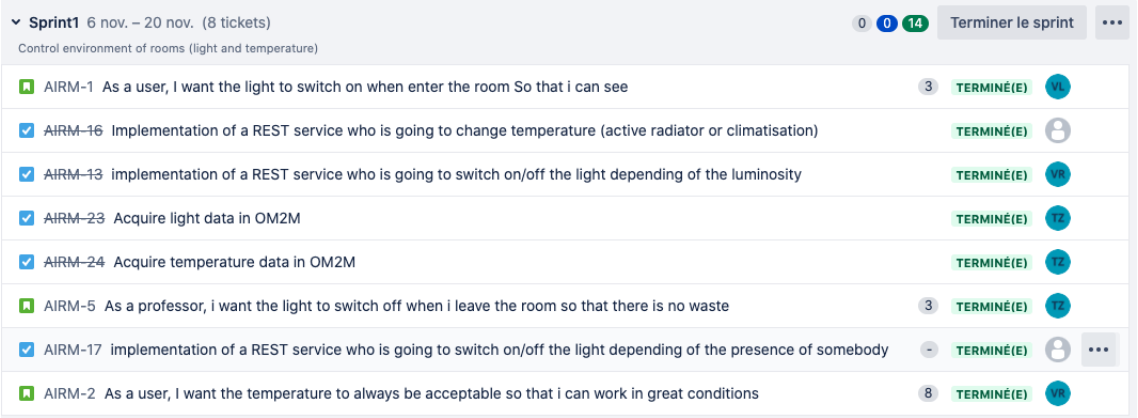

Figure 3: Sprint 1 organisation on JIRA

2.3 Implementation example: POST request

we performed functions to be able to interact with the OM2M platform. For this, we used *HttpUrlConnection*. For example, if we take the POST request scheme dedicated to OM2M, we had to take into account the following parameters concerning the header:

- define the originator: X-M2M-Origin with admin:admin identifiers
- define content format (JSON)
- define the type of entity (in our case, we created CIN)

But also at the level of the object of the request, the structure of the oneM2M standard had to be taken into account. Indeed, the *<m2m:cin>*, *<cnf>* and *<con>* tags had to be correctly specified.
You can found on figure 4, the code of the java method to send sensors data to the platform.

```java
//For HTTP POST Methods
public void sendDataToOM2M(String myUrl, String dataType, String dataValue) throws IOException, JSONException {

    URL url = new URL (myUrl);

    HttpURLConnection con = (HttpURLConnection)url.openConnection();
    con.setRequestMethod("POST");
    con.setRequestProperty("Content-Type", "application/json; ty=4");
    con.setRequestProperty("X-M2M-Origin", "admin:admin");
    con.setRequestProperty("Accept", "application/json");

    con.setDoOutput(true);

    JSONObject jsonInput= new JSONObject("{\"m2m:cin\": {\"cnf\": \"" + dataType +  "\", \"con\": \"" + dataValue + "\"} }");
    String jsonInputString = jsonInput.toString();

    try(OutputStream os = con.getOutputStream()) {
        byte[] input = jsonInputString.getBytes("utf-8");
        os.write(input, 0, input.length);
    }
```

Figure 4: Extract of the Java method used to send sensors/actuators data to OM2M

2.4 Web interface

A small dashboard was created on which we can visualize in real time the evolution of the state of the actuators and the values collected by the sensors of a GEI room. To achieve this, we have implemented a JavaScript code directly related to our GET/POST methods implemented in Java under Eclipse. Concerning the update of the data on the OM2M platform, we implemented a button on the web interface which allows to automatically generate the data capture of the sensors which are then stored directly on the platform. An example of the use of the interface is shown in the figure 5.

3 Tests

3.1 Scenario 1

The following scenario is implemented inside the Java GET request:
If the temperature is above 25 degrees and the window is closed, we open it, and on the contrary, if the temperature is below 18 degrees and the window is open, we close it. If somebody is inside and the luminosity is low, we switch on the light automatically, and if nobody is inside the room or the luminosity is high the light is switched off.

3.2 Scenario 2

The following scenario is implemented inside the Java GET request:
After 7 p.m. and before 6 a.m., if the motion sensor indicates the presence of a person inside the room and the windows and doors are closed, then the alarm goes off and the police are notified directly.

3.3 Overview

You can found below an overview of the dashboard for the first scenario. Indeed, we display all the current information of the room environment, that is to say the current values of the sensors and the sate of actuators.Displayed data is refreshed every 10 seconds.

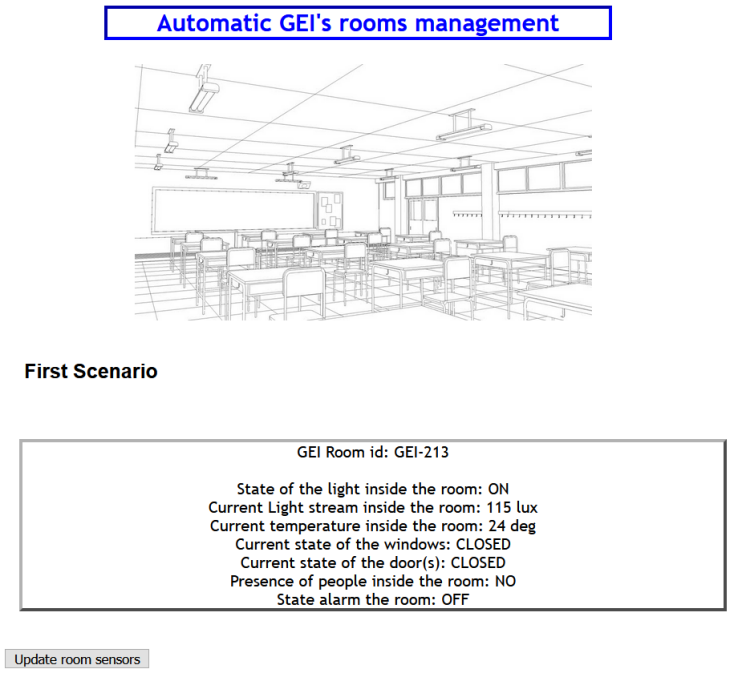

Figure 5: Dashboard overview (Scenario 1)

4 Conclusion

Through this development of a Web application, we learned how to use and implement a service oriented architecture on a concrete case: here, the management of rooms.

The user can interact with a room environment using the web interface. Here, we use simulated sensors, it is easy to change their values or states with a simple URL. It is also possible to change the state of actuators manually, but what is more interesting is let the system control the environment automatically, with scenarios.

These scenarios were implemented using an agile methodology and the project development tool JIRA to transform the project in small tasks. It was the first time for us using a project management software combined with our knowledge on the agile method. It allowed a clearer view and a better follow-up and distribution of the different tasks of the project.

Through our tests of the scenarios, we saw that our Web application works properly, and can respond to environmental changes. The changes are done manually using the web interface, but we can imagine real sensors transmitting their data to the application. Then the system retrieves these data and can act consequently to match the stories of the scenario.

List of Figures