

# Service Architecture and Middleware for IoT

## NRJSaver project

**Emilie Estival, Fanny Michel, Maxime Pages**

January 13, 2021

---

# 1 Introduction

We developed a Web application (Proof-of-Concept) for managing INSA's rooms. This application allows automatic closing windows, heating management, and lights management. This application relies on software services, sensors, and actuators. The goal is to retrieve data from sensors and analyze them to enable taking decisions.

Through software services (Restful Web services), the application retrieves data from sensors (temperature, presence, ...), and according to the values of the retrieved data, actions on actuators can be triggered.

We designed our application based on a Rest architecture, we implemented the different services and services calls as well as a web interface for viewing the history of actions. The language used is Java on the SpringBoot for Rest services framework. We used OM2M for sensors and actuators based architecture.

## 2 Project Management

### 2.1 Definitions

#### 2.1.1 Agile Method

The Agile Method is a project management methodology, with a new way of approaching problems, of thinking and of working. This innovative method stands out from traditional predictive and sequential methods through an iterative and incremental process that allows adaptability and inclusion of the client in the realization of the project.

#### 2.1.2 SCRUM Method

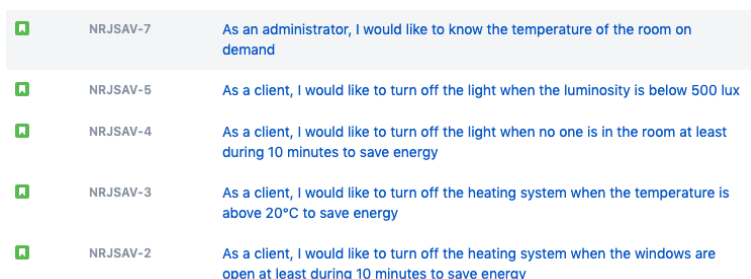
The SCRUM method is the most widely used Agile method. It allows you to break down the project into simple tasks and plan them in small groups over short periods called sprints. Each sprint ends with a functional rendering, then adjustments are made and the next sprint is planned, and so on.

### 2.2 Application

In order to plan and organize our project, we use JIRA following the scrum method. JIRA is an excellent tool used for issue tracking and project management.

#### Features and Backlogs

Following the scrum method, we need first to define the backlog that gathers the user stories. For this, we create several tickets representing our stories. Each story corresponds to a feature of our product.



NRJS-7	As an administrator, I would like to know the temperature of the room on demand
NRJS-5	As a client, I would like to turn off the light when the luminosity is below 500 lux
NRJS-4	As a client, I would like to turn off the light when no one is in the room at least during 10 minutes to save energy
NRJS-3	As a client, I would like to turn off the heating system when the temperature is above 20°C to save energy
NRJS-2	As a client, I would like to turn off the heating system when the windows are open at least during 10 minutes to save energy

Figure 1: Definition of stories on JIRA

Each time, as you can see on the figure above, user stories are based on three questions : "who, what and why ?". For example, one of the stories, we call Story (1) here, is : "As a **client**, I would like to **turn off the heating system** when the temperature is above 20°C to **save energy**".

Then, we need to set the effort of the user story and to define its degree of priority. Some stories need to be completed before others and we prefer to prioritize some of them. We estimate the priority of the user stories with the Planning Poker method. For example, we define a *High* priority and 5 Story points for the Story (1) because we consider it one of the most important stories.

## Sprints

Once all our features and backlogs are defined, it is time to plan the sprints. At the beginning of a sprint, we define the tasks we are going to deal with in the next (or next two) week(s). Each sprint generally begins with a meeting to plan the different tasks and who is in charge of each. The aim is to deliver a functional render at the end of the sprint. At the beginning of the sprint, all tasks are planned, and can be associated to a responsible. Next during the sprint, tasks are placed in 3 categories (Todo, In progress, Done) depending on their advancement. At the end, a final meeting is planned to review the work done and the tasks for the next sprint. And the next sprint starts.

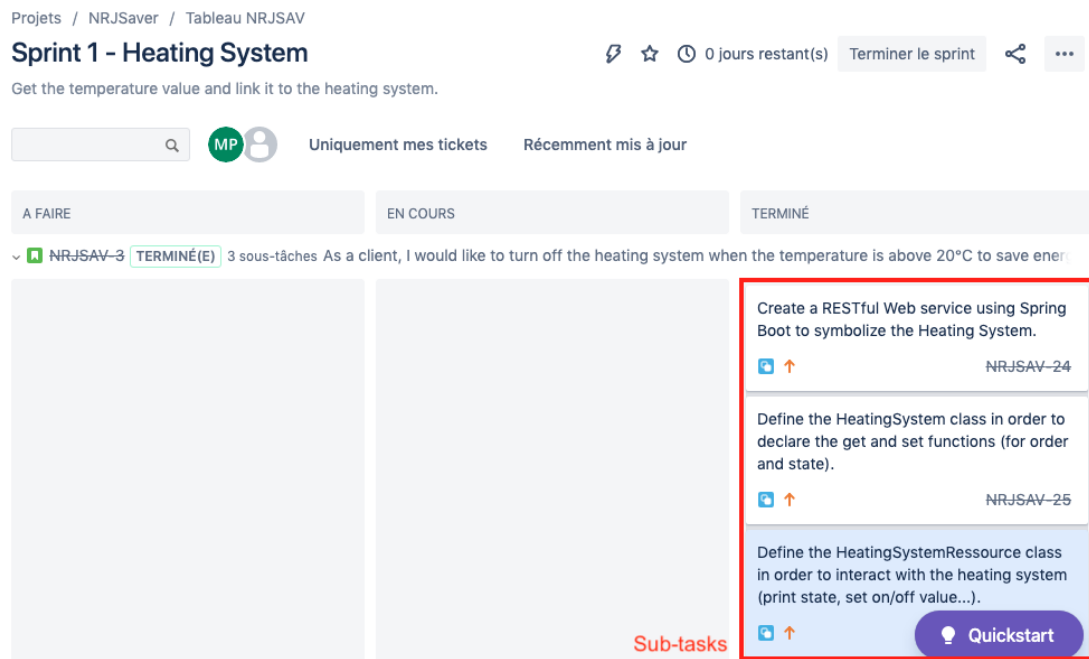


Figure 2: Sprints example on JIRA

The figure above is an example of our first sprint entitled "Sprint 1 - Heating System". Linked to the NRJSAV3 story, previously detailed, we have defined sub-tasks to organize our work. Sub-tasks are shown in the red box.

## 3 Project Development

### 3.1 Controller

"AutomationService" is our controller. This Java Application supervises the lower level microservices by POST and GET requests.

First, it allows the user to know the actual state of the heating system and the windows. Then, according to the temperature (which value comes from OM2M), the AutomationService manages the order of the heating system and sends a new message to inform the user of the changes.

### 3.2 Microservices

The microservices are java scripts allocated to specific entities (like heating system, windows). They are used to get or set the states of the entities they are linked to. These entities are TemperatureService and LightService, that are directly related to the OM2M platform.

**Note :** in the following, we will skip the details for the luminosity management since it would be very similar to the heating system management.

### 3.3 Tests

In order to test the project without the use of OM2M, we write a code in TemperatureService that will return a random temperature.

```
1 package fr.insa.mas.TemperatureService;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RestController;
6
7 @RestController
8 @RequestMapping("/temperature")
9 public class TemperatureRessource {
10
11     @GetMapping("/value")
12     public int getTemperature() {
13         return 15 + (int) Math.ceil(Math.random()*10);
14     }
15 }
```

Figure 3: TemperatureService

Then, this temperature is used in the controller AutomationService. Below, we can see the displays once the AutomationService is launched.

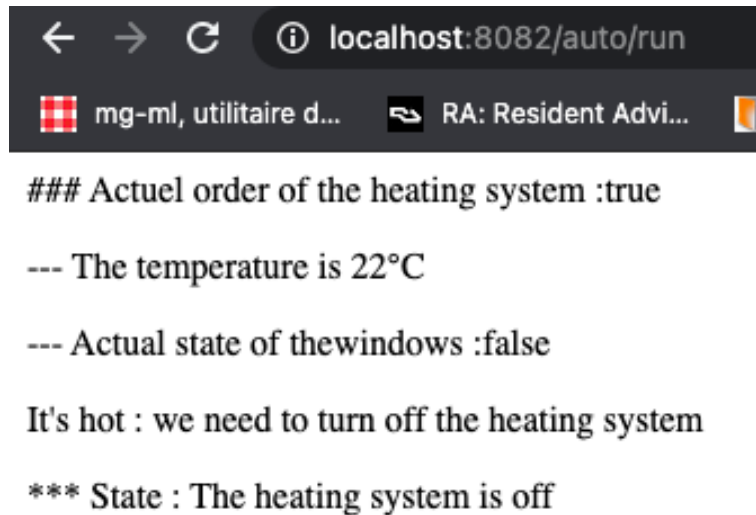


Figure 4: AutomationService test 1

For this test, the randomly generated temperature is 22°C so we want to turn off the heating system.

Now, let's see how the system reacts if the temperature is below 20°C.

```
### Actuel order of the heating system :false
--- The temperature is 17°C
--- Actual state of the windows :false
Brrrr it's really cold : we need to turn on the heating system
*** State : The heating system is on
```

Figure 5: AutomationService test 2

For these two tests, we set the windows in a false state (windows are closed), but if we set them in a true state (windows are opened) then the heating system will be turned off.

### 3.4 OM2M

OM2M contains the data, like states of the GEI entities. It simulates the physical captors that are supposed to send us information about the environment. We can see below the OM2M resource tree for our project. We created (via Postman) two instances in TemperatureSensor/DATA, each one associated with a temperature.

## OM2M CSE Resource Tree

<http://127.0.0.1:8080/~mn-cse/cin-787770573>

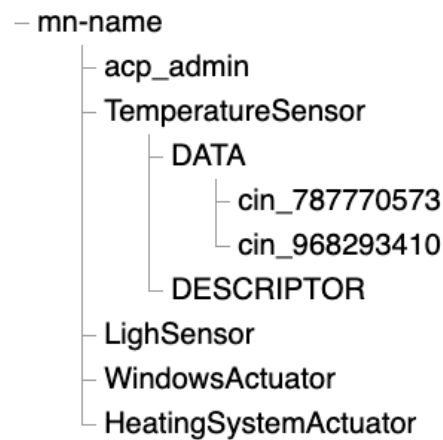


Figure 6: OM2M Resource Tree

Then, we have to link this resource tree with our Java code. This is done in `TemperatureService` (we modify the initial code shown in Figure 2). We did not manage to get the instances into our Service, maybe due to a code error. At the time this report was made, we were in contact with a teacher to help us finalize our application.

### 3.5 Conclusion

With this project, we learned how to develop a web application and how to interface OM2M with a Service Oriented Architecture. We developed a lot of skills in web and network programming. We could also progress in Agile mode through the definition of sprints and user stories. Unfortunately, we lacked some time to add some functionalities like the luminosity management. But it was foreseeable that we wouldn't have the time to tackle all the tasks, which is why we prioritised the setup of the so-called "high importance" tasks on Jira.

This kind of application, with some improvements, could be useful for the future of the GEI to automate heating and light systems and thus save energy.