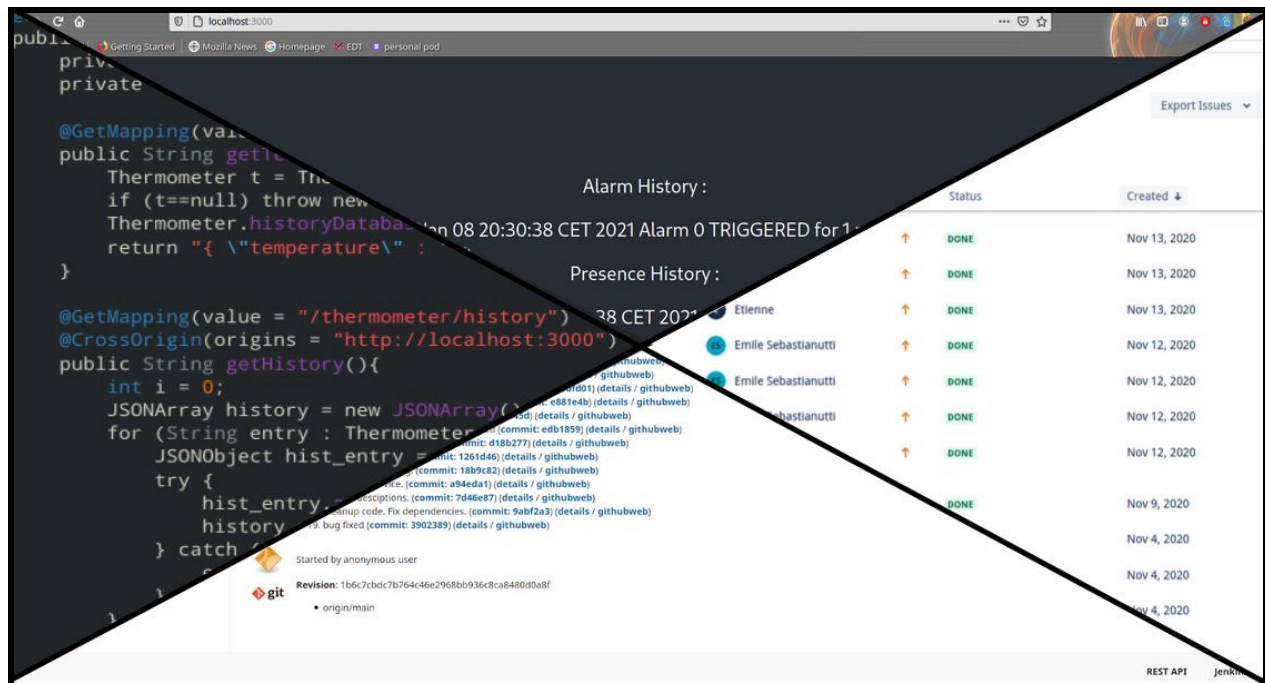


# Automatic management of INSA's rooms

## INSA Toulouse - Innovative Smart Systems



*Etienne De Prémare*  
*Marc Kasse*  
*Emile Sebastianutti*

<https://github.com/emileseb/AutomaticRoomManagment>

## Projet requirements

"You are requested to develop a Web application (Proof-of-Concept) for managing INSA's rooms . This application must allow automatic closing windows, doors, turning on heating, turning off lights ... etc. 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 of sensors (temperature, presence, ...), and according to the values of the retrieved data, actions on actuators can be triggered. Your application must be based on a Restful architecture."

## Conception choices

In this project, we chose to focus on the **robustness** of each part of the architecture. The different services of our architecture communicate using **JSON**, most of the functions have **unit tests**. Their **HTTP endpoints** are documented on the readme, and the usage of all the services is very simple, using a **makefile** with all the **maven instructions**.

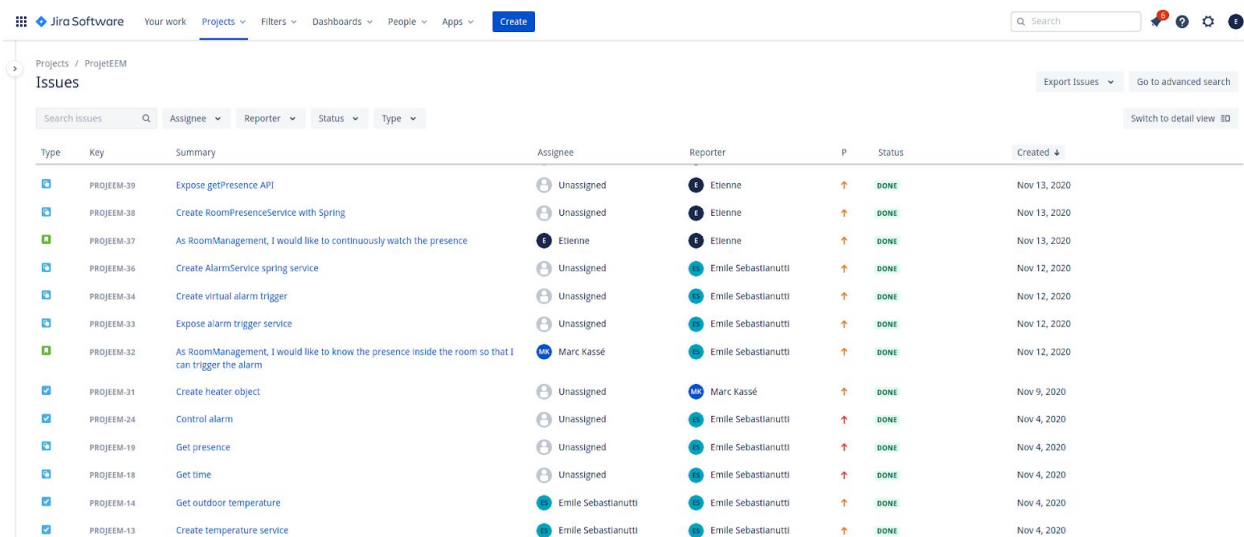
## Agile Methods

### Jira

We used JIRA as a tool to manage the project. The project required us to create five different Sprints : Temperature, Presence, Heater, Alarm and Web Application. Each sprint corresponds to one service. In each sprint we defined user stories and different tickets with a level of priority.

We got familiar with the main functions in JIRA but there are many which were not needed for our project.

Below is a sample of the list of tickets we generated in JIRA:



Type	Key	Summary	Assignee	Reporter	P	Status	Created
Task	PROJEEEM-39	Expose getPresence API	Unassigned	Etienne	↑	DONE	Nov 13, 2020
Task	PROJEEEM-38	Create RoomPresenceService with Spring	Unassigned	Etienne	↑	DONE	Nov 13, 2020
Task	PROJEEEM-37	As RoomManagement, I would like to continuously watch the presence	Etienne	Etienne	↑	DONE	Nov 13, 2020
Task	PROJEEEM-36	Create AlarmService spring service	Unassigned	Emile Sebastianutti	↑	DONE	Nov 12, 2020
Task	PROJEEEM-34	Create virtual alarm trigger	Unassigned	Emile Sebastianutti	↑	DONE	Nov 12, 2020
Task	PROJEEEM-33	Expose alarm trigger service	Unassigned	Emile Sebastianutti	↑	DONE	Nov 12, 2020
Task	PROJEEEM-32	As RoomManagement, I would like to know the presence inside the room so that I can trigger the alarm	Marc Kassé	Emile Sebastianutti	↑	DONE	Nov 12, 2020
Task	PROJEEEM-31	Create heater object	Unassigned	Marc Kassé	↑	DONE	Nov 9, 2020
Task	PROJEEEM-24	Control alarm	Unassigned	Emile Sebastianutti	↑	DONE	Nov 4, 2020
Task	PROJEEEM-19	Get presence	Unassigned	Emile Sebastianutti	↑	DONE	Nov 4, 2020
Task	PROJEEEM-18	Get time	Unassigned	Emile Sebastianutti	↑	DONE	Nov 4, 2020
Task	PROJEEEM-14	Get outdoor temperature	Emile Sebastianutti	Emile Sebastianutti	↑	DONE	Nov 4, 2020
Task	PROJEEEM-13	Create temperature service	Emile Sebastianutti	Emile Sebastianutti	↑	DONE	Nov 4, 2020

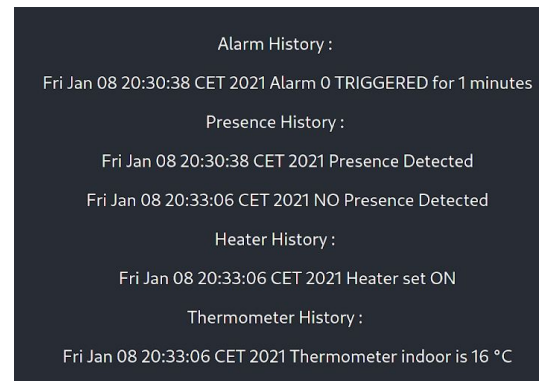
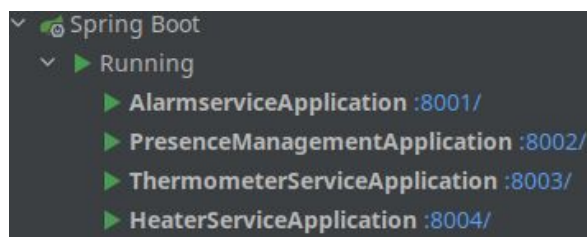
---

# Functionalities

## REST Architecture

We developed a **RESTful** architecture composed of four **service** applications each one providing two services, one **management** application and one **web** application. The details of the services can be found in the README file of the project, this part will present the architecture of them.

## The provided services



### Service app:

Each application provides the **service** to manage it's simulated domotic type of equipment. This service logs each request into a history database and provides the service to access this database. For example, the indoor temperature is accessible through the thermometer service by doing **HTTP request GET** <http://localhost:8003/thermometer/indoor> and return a JSON { "temperature" : 22 }.

### Management app:

The management app goal is to **automatically** GET on the sensor dedicated services and, regarding the result, POST an activation request to use the actuators, it runs continuously. Note that the services have weird behaviour for testing purposes in order to simulate a real time sensing.

### Web app:

The web app is a REACT application that uses a GET Hook on the four history services and **displays** them to allow the user to follow the **history** of the global system.

### OneM2M:

We started to implement a OneM2M architecture to model the sensors as it can be seen on the alarm resource file but with the lack of time we chose to focus on the **REST aspect** of the project by developing a RESTful web app instead.

# Continuous Integration

## Maven

We used the **maven** configuration provided by **spring-boot**, and added a root **pom.xml** that includes all the different spring-boot apps. We were then able to easily compile, test and run our services with the **spring-boot maven plugin**. (See README #Usage)

```
16     <modules>
17         <module>AutoManagement</module>
18         <module>AlarmService</module>
19         <module>PresenceManagement</module>
20         <module>ThermometerService</module>
21         <module>HeaterService</module>
22     </modules>
23 </project>
```

## Jenkins

We used **Jenkins** for continuous integration. We configured it to use our root project's pom.xml with **maven**. On each build, the goals **compile** and **test** are executed. Here is a screenshot of one of our latest successful build:

The screenshot displays the Jenkins web interface for a project named 'RoomManagement'. The main heading is 'Build #12 (Jan 8, 2021 6:32:10 PM)'. On the left sidebar, there are links for 'Back to Project', 'Status', 'Changes', 'Console Output', 'Edit Build Information', 'Delete build #12', 'Git Build Data', 'No Tags', and 'Previous Build'. The 'Status' section is active, showing a list of 19 changes with commit hashes and links to details. Below the changes, it indicates the build was 'Started by anonymous user' and shows the 'Revision' as '1b6c7cbd7b764c46e2968bb936c8ca8480d3a8f' from the 'origin/main' branch. At the bottom right, there are links for 'REST API' and 'Jenkins 2.271'.