[5ISS] & [MSIoT]
2021/2022

Service Based Architecture and Software Engineering

Project:
"Automatic management of INSA's rooms"

by
Rami KARAOUD
Abir BENAZZOUZ
Grégoire HEBRAS

https://github.com/RamiKaraoud2019/SOA

Introduction

This semester we have been introduced to the notion of Service oriented architecture, which is based on building elementary microservices and composing web applications by combining them. This allows the reuse of functionalities for different applications while also allowing the opportunity to evolve micro-services and easily integrate them without damaging the larger applications.

Our mission here is to develop a web application responsible for automating the management of INSA's rooms in a smart building. This includes features like turning on and off heaters or air conditioning, closing windows and doors, shutting lights off, and so on. The application is based on microservices to retrieve data from sensors distributed throughout the building, analyze it and trigger actions on actuators.

For that purpose we imagine different scenarios describing what the application must do depending on the data that is retrieved. To implement those scenarios we use a Restful architecture, the different services are written in Java and the project uses the SpringBoot framework.

This report presents the specifications of our project, and the chosen scenarios, to be more precise, it describes the functionalities implemented and the aim of those specific actions. Secondly, it explains the implementation process, and the project management.

Project definition

First, we need to define what our application aims to manage, and what it will be capable of doing. To supervise the project we use the Scrum method through the project management platform JIRA.

We decided as a group to focus the following scenarios:

1. Managing the heating in a large building can be a lot of work, and is often a matter of complaints. Therefore, we want the heating to be managed automatically.
2. Lights are sometimes left off when someone enters a room.. We want to automate the light so that it turns on by itself when someone is in the room.
3. At the end of the day no one thinks of shutting the shutters down. Having them down during the night is important because it ensures a better isolation at the coldest time of the day. We want to make sure they are down during the majority of the night.

User stories:

Following those scenarios we define user stories, these are implemented in JIRA and help us supervise the advancement of the project. User stories help us answer the question Who? What? and Why?

- **US1**: As an administrator, I want to turn on the heating when the room temperature gets below a certain value to keep the room comfortable.
- **US2**: As an administrator, I want to turn off the heating once the temperature exceeds a certain value to not waste energy on unneeded heating.
- **US3**: As an administrator, I want to control the light to ensure it's on when needed.
- **US4**: As an administrator, I want the shutters to close after a certain hour to avoid heat loss during the night.

Tasks:

For each user story we decline the tasks needed to reach our specifications. Each task describes what needs to be coded.

- **US1 & US2:**
  - Get room temperature: we need to periodically check the value sent by the temperature sensor in the room.
  - Heater state: we need to get the state of the heater at any point.
  - Turn on heater: we want to activate the heating when the room temperature gets below a certain point.
  - Turn off heater: we want to turn off the heater when the room temperature exceeds the "comfortable range".
- **US3**:
  - Movement detection: a signal must be sent whenever a movement is detected in the room.
  - Turn on light: we want to turn on the light when a movement is detected.

- **US4**:
  - Get luminosity: a signal must be sent when the night has fallen

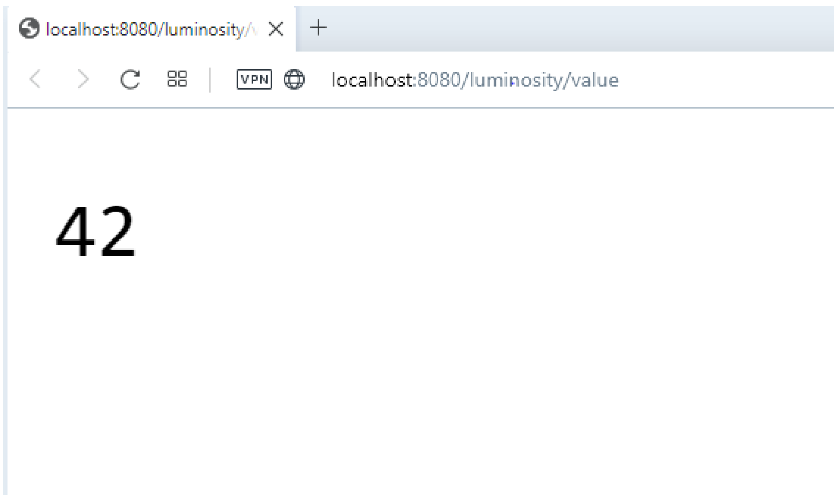
- ➢ Shut down shutters: when the night has fallen, we want to close the shutters.

Implementation

To deploy our application, we develop various web services that simulate sensors and actuators, and exchange data following our architecture.

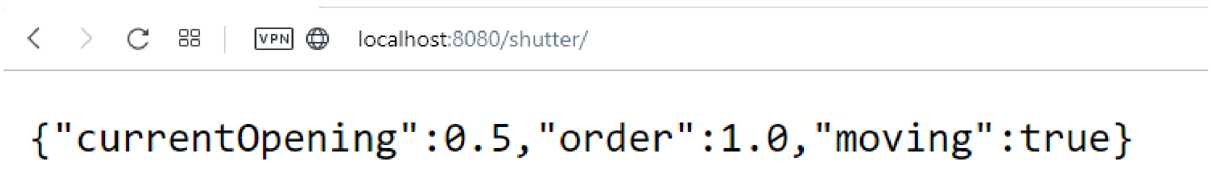
For the first scenario, we started by simulating the temperature sensor that would be located inside every room. It can return the temperature.
Then, we created the service for the heating system, it return its stat.

We can access a generated luminosity value:



```
localhost:8080/luminosity/value

42
```

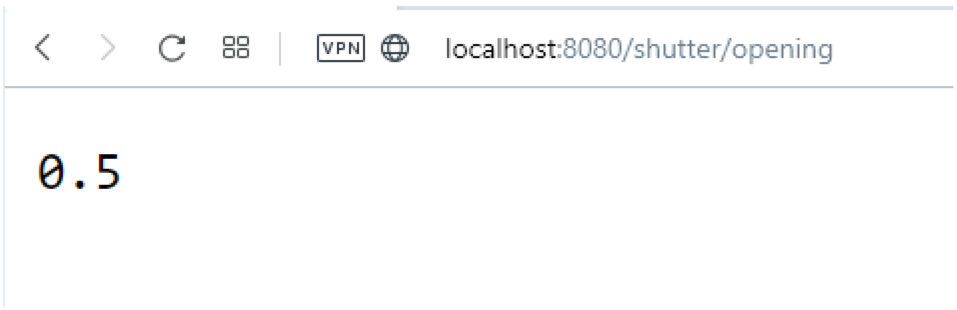
We can access the shutter’s state :



localhost:8080/shutter/

```
{"currentOpening":0.5,"order":1.0,"moving":true}
```

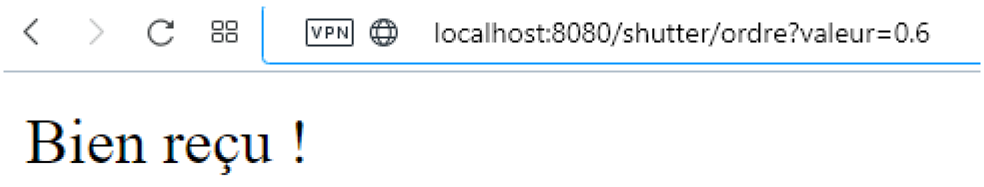
We can also access the current Opening value solely :



localhost:8080/shutter/opening

```
0.5
```

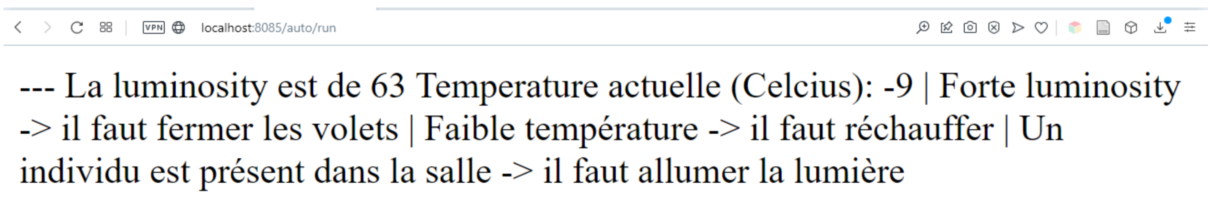
We can give an order to the shutter :



localhost:8080/shutter/ordre?valeur=0.6

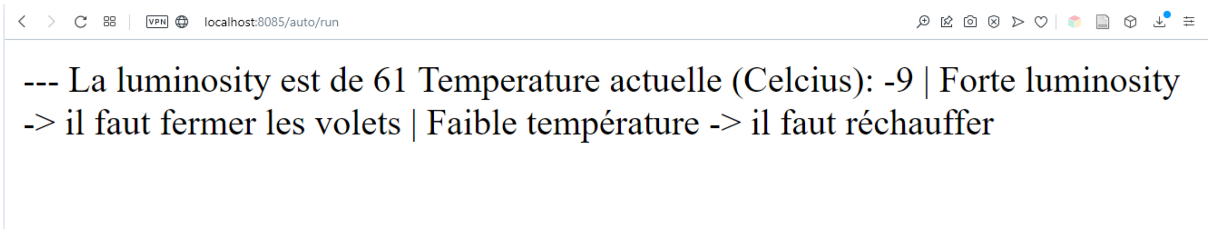
Bien reçu !

We can easily observe the behavior of out system through the webpage provided thanks the orchestrator service :



localhost:8085/auto/run

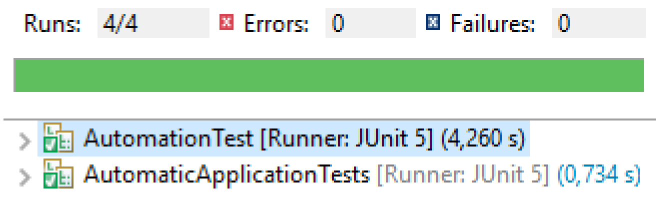
--- La luminosity est de 63 Temperature actuelle (Celcius): -9 | Forte luminosity -> il faut fermer les volets | Faible température -> il faut réchauffer | Un individu est présent dans la salle -> il faut allumer la lumière



--- La luminosity est de 61 Temperature actuelle (Celcius): -9 | Forte luminosity -> il faut fermer les volets | Faible température -> il faut réchauffer

Testing

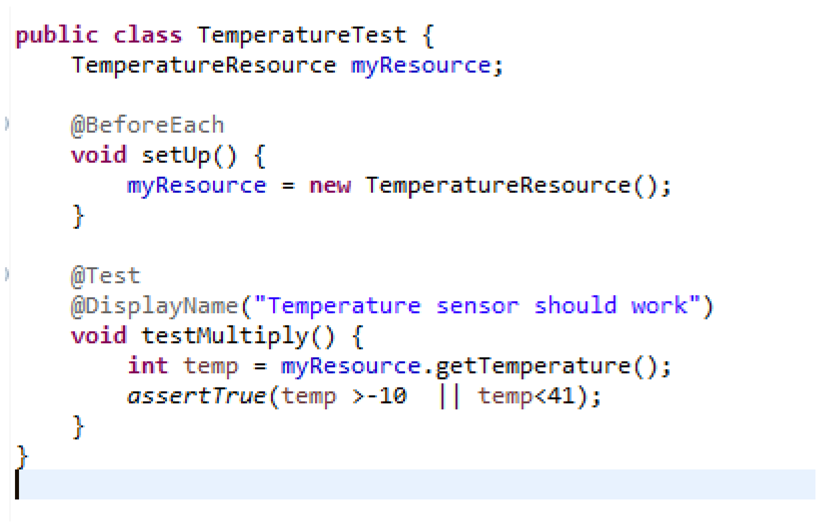
Testing can also be done automatically thanks to JUNIT.
We created some main tests for the orchestrator :



```
Runs: 4/4    Errors: 0    Failures: 0

> AutomationTest [Runner: JUnit 5] (4,260 s)
> AutomaticApplicationTests [Runner: JUnit 5] (0,734 s)
```

```
@Test
@DisplayName("Automation of lights should work")
void testLights() {
    String msg = " | Un individu est présent dans la salle -> il faut allumer la lumière ";
    assertEquals(msg, myResource.lights(40, 1),
            "Problème de lumière");
}

@Test
@DisplayName("Automation of windows should work")
void testWindows() {
    String msg = " | Forte luminosity -> il faut fermer les volets ";
    assertEquals(msg, myResource.windows(60),
            "Problème de volets");
}

@Test
@DisplayName("Automation of heater should work")
void testHeater() {
    String msg = " | Forte température -> il faut refroidir ";
    assertEquals(msg, myResource.chauffage(40),
            "Problème de chauffage");
}
```

Each service also has its own test :



```java
public class TemperatureTest {
    TemperatureResource myResource;

    @BeforeEach
    void setUp() {
        myResource = new TemperatureResource();
    }

    @Test
    @DisplayName("Temperature sensor should work")
    void testMultiply() {
        int temp = myResource.getTemperature();
        assertTrue(temp >-10  || temp<41);
    }
}
```

Project management

For this project we distributed the tasks according to each member's strengths to ensure maximum efficiency while still sharing the knowledge to all. That way, the work was well done and every team member learned from the project. Also, we decided that the person that implements a class does not implement its tests.

- All:
  - Scenario definition
  - Architecture building
  - Report redaction
  - Project management

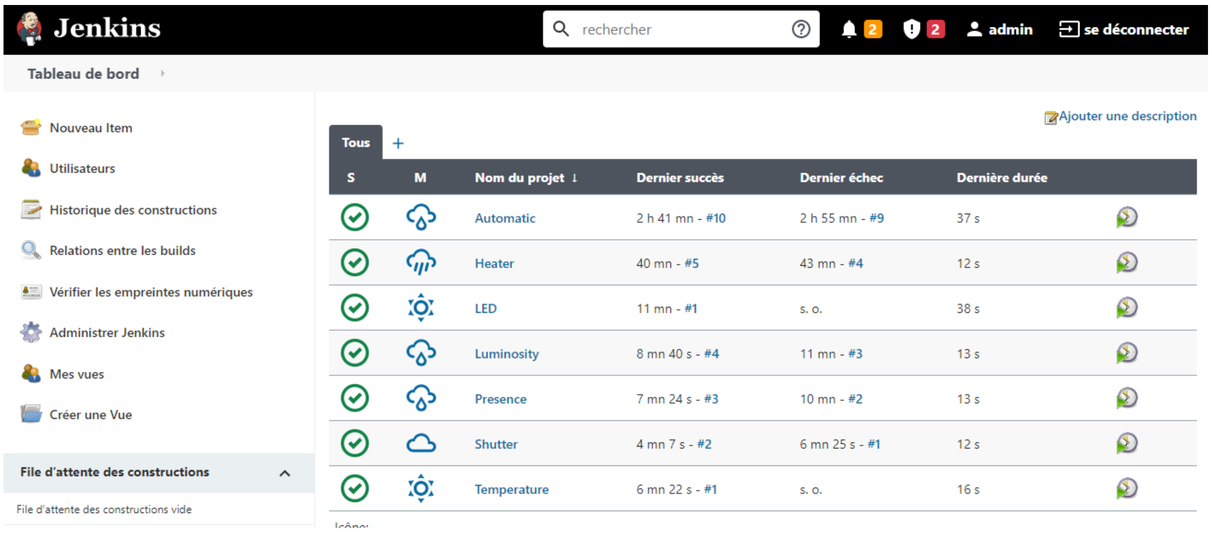
- Grégoire HEBRAS:
  - Test of Heater, Luminosity
  - LED service, shutter service
  - Updating of the JIRA platform

- Abir BENAZZOUZ
  - Luminosity service, Heater service
  - Test of orchestrator, temperature
  - Test of LED, shutter

- Rami KARAOUD
  - Orchestrator service, Temperature service
  - Test of luminosity, heater
  - Jenkins integration

Integration

Jenkins makes it possible to automatically build and test each service. To do so, we created a job per service with the “clean compile test” command. Then we added each job to the pipeline. We did not deploy our pipeline.



Jenkins rechercher 2 2 admin se déconnecter

Tableau de bord

- Nouveau Item
- Utilisateurs
- Historique des constructions
- Relations entre les builds
- Vérifier les empreintes numériques
- Administrer Jenkins
- Mes vues
- Créer une Vue

File d'attente des constructions

File d'attente des constructions vide

Ajouter une description

Tous +

S	M	Nom du projet ↓	Dernier succès	Dernier échec	Dernière durée	
✓	☁	Automatic	2 h 41 mn - #10	2 h 55 mn - #9	37 s	
✓	☁	Heater	40 mn - #5	43 mn - #4	12 s	
✓	☀	LED	11 mn - #1	s. o.	38 s	
✓	☁	Luminosity	8 mn 40 s - #4	11 mn - #3	13 s	
✓	☁	Presence	7 mn 24 s - #3	10 mn - #2	13 s	
✓	☁	Shutter	4 mn 7 s - #2	6 mn 25 s - #1	12 s	
✓	☀	Temperature	6 mn 22 s - #1	s. o.	16 s	

Conclusion

This project made it possible for us to understand a full SOA project from designing to implementing, testing and automation. Managerial skills were also acquired thanks to JIRA and the agile method.