Service Architecture project:
Automatic management of GEI's rooms

****

Students: Raphael Bernard - Maryam Boujdaa - Vatosoa Razafiniary / 5 ISS 2022
Github repository access: https://github.com/Vatosoa285/Projet_room_services.git

Introduction

The GEI is a building used by many people everyday for various purposes, and therefore sees a lot of coming and going in its rooms. Students and teachers, opening windows to change the air, heating up the room or leaving lights on when they leave.
The objective of our project is to offer a solution to automate the GEI's classroom management, using various sensors and actuators. We will use microservices to collect and analyze data, and to launch actions based on the retrieved data. The idea is to ultimately have a web application managing everything.
To do this, the project went through 3 main development phases:

I. a brainstorm of the various services that could be implemented as part of this automatic management of GEI's rooms
II. the project planning
III. the realization: using Spring Boot and the java jee eclipse IDE

I - Brainstorm of the various services that could be implemented as part of this automatic management of GEI's rooms

In the optimal case and if we had enough time available our hands, here are the brainstormed specifications we would have liked to implement for our room management system by priority order:

- **Light management system:** whose mission will be to properly turn on or off the light in a room based the presence detection and defined times,
- **Heat management:** whose mission will be to 1) open or close the blinds/windows regarding the outside and inside temperature or 2) turning on or off a heater based on the presence of people and temperature level in the room,
- **Security system:** whose mission is to identify and notify any abnormal functioning of the previously identified services

Each of these features is translatable into a set of microservices that each have a specific role and interact with one-another. This is actually detailed in part II.

II - Project planning

We have chosen the project management software "Jira" to manage and monitor the project. The idea is to achieve a good organization of the realization stage based on the identified "user stories" and following the "scrum" method.
To set up the project planning:

- we first defined the different categories of microservices fundamental to the implementation of the functionalities presented in section I
- then, we defined properly the user stories and sprints on our Jira backlog

II.1 - Microservices and their interactions

Since we want to work with a REST architecture, we need to define the microservices we need.

What is a microservice and how do we define it?

→ A microservice can be seen as a main responsibility related to an homogenous application perimeter. Example a Microservice for presence detection = {methods related to presence detection: processing presence detector data}

→ The microservices are defined in such a way that they are **as atomic** and **re-usable** as possible. This eases the implementation process and the different possible uses.

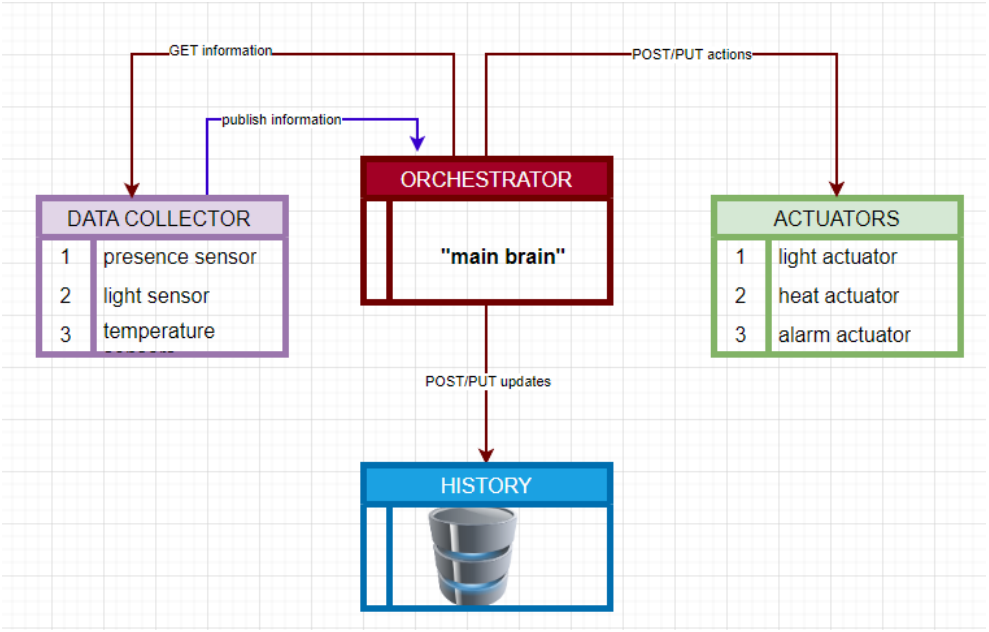

Figure 1. Microservices categories

We identified four categories of microservices:

- **the data collectors:** refer to the entities that are in charge of the reception and acquisition of data from the various sensors(at the beginning of the chain)
- **the actuators:** These are the microservices of "end of chain", whose mission is to process the final actions to be carried out
- **the orchestrator:** it is the main brain of the whole system. It analyzes the information gathered from the "collectors" and activates the actions to be undertaken by the “actuators”. It also plays a role in feeding history.
- **the history:** it is the history of the various orchestrated actions.

II.2 - Jira backlog

We have then defined the specific functionalities we want to implement. We begin by defining “user stories” : real life situations needing an intervention by our system, from an user point of view. Each of these stories is then divided into a set of elementary tasks,

making reference to the part of the corresponding microservice to . These stories define the backlog of our project : our “to-do list”.

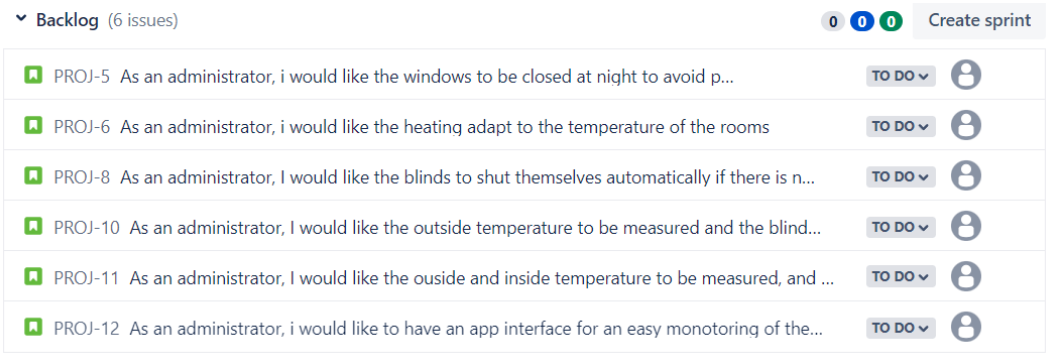

Figure 2. User stories inventory

NB: Some academic adaptation points:

- If a sprint is usually programmed for a minimum period of 2 weeks in company, we have adapted it to a duration of 1 week : we had **4 sprints** which correspond to the number of school-weeks we had from project’s kick-off to the final week.
- since the team takes on the role of scrum master, product owner and developers at the same time → our strategy was above all to base ourselves on the backlog in order to divide the tasks between us as “developers” and move forward effectively in the realization

We have divided our tasks as follow :

- Vatosoa Razafiniary: specialized on the “data collectors” and the cloud migration
- Raphael Bernard: specialized on the “orchestrator”
- Maryam Boujdaa: specialized on the end-services ie. the “actuators”

III - Realization

As presented in the introduction, we used Spring Boot and the java jee eclipse IDE for the implementation of microservices.
Our development strategy can be divided into three phases:

- A local development phase: where each member take the time to model and code his specific microservices and ensure to validate them locally
- A "gathering" phase: where interoperability between the different microservices is tested in local;
- A cloud migration phase: to make microservices discoverable and usable by clients regardless of their deployment address.

We have provided in the archive all the codes produced from the project. Below is a summary table of all the methods tested.

PHASE 1					PHASE 2	PHASE 3
CATEGORY	**Microservice name**	**Running port**	**Methods**	**tested locally ?**	**registered on cloud ?**	**tested on cloud ?**
data collector	LightInfoMS	8080	1)method that generates an object "light sensor" from an id 2)method that displays the list of all light sensors	☑	☑	
	PresenceInfoMS	8083	1)method that generates an object "presence sensor" from an id 2)method that displays the list of all presence sensors	☑	☑	
orchestrator	ControlMS	8081	1)method to view the list of all presence sensors 2)method to view the list of all light sensors 3)method that activates an alarm if if a presence is detected after 22h 4)method that switch on/off light in a room according to presence sensor status and light status	☑	☑	☑
actuators	lightEndMS	8086	1) method to switch on the light 2) method to switch off the light	☑	☑	
	alarmeEndMS	8085	1) method to activate the alarm 2) method to deactivate the alarm	☑	☑	
Spring Cloud	msDiscovery	8161	this is the Eureka server, where the above microservices are registered on the cloud	☑		

Conclusion

We can say that this project has allowed us to better appropriate the main concepts of the Service Oriented Architecture course: the concept of microservices, the use of Spring Boot as well as the agile method. It also taught us about teamwork and time management in a development project. Although the final features implemented do not cover the initial specifications, we find it more valuable to have focused on the basics to ensure a good understanding and mastery of what we have developed.

****