

INSA TOULOUSE



SERVICES ORIENTED ARCHITECTURE

5ISS

---

# Rapport des Travaux Pratiques

---

*Auteurs:*

Aymen Boukezzata

Thomas Berton

Arselan Meslough

January 25, 2022

# Contents

<b>1</b>	<b>Scenarios</b>	<b>4</b>
<b>2</b>	<b>Architecture</b>	<b>4</b>
<b>3</b>	<b>Implémentation</b>	<b>4</b>
3.1	Micro-service Actionneurs: Lampe, Volet et Fenêtre . . . . .	4
3.2	Micro-service Capteurs: Luminosité, Gaz et Présence . . . . .	5
3.3	Micro-service Contrôleur . . . . .	6
<b>4</b>	<b>Exploitation des données / GUI</b>	<b>6</b>
<b>5</b>	<b>Intégration du projet</b>	<b>7</b>
<b>6</b>	<b>Organisation du projet</b>	<b>8</b>

## List of Figures

1	Architecture du projet . . . . .	5
2	Affichage des données brutes des différents scénarios . . . . .	6
3	IHM sous HTML . . . . .	7
4	Les règles Cross Origin . . . . .	7
5	Github action . . . . .	8
6	Sprint 1 . . . . .	8
7	Création des scénarios . . . . .	8
8	Création des projets SpringBoot pour les microservices . . . . .	9
9	Création des classes . . . . .	9
10	Création des ressources . . . . .	10
11	Sprint 2 . . . . .	10
12	Développement des micro-services liés au contrôleur . . . . .	10
13	Développement des micro-services liés au actionneurs . . . . .	10
14	Développement des micro-services liés au capteurs . . . . .	11
15	Sprint 3 . . . . .	11

# Introduction

Le but de ce projet est de créer une application de contrôle dans un contexte IoT, en se basant sur Architecture Orientée Services (SOA). Le style architectural adopté suit l'approche des micro-services ; Qui représentent une suite logicielle ayant une seule responsabilité et construite autour de la capacité d'être légère en terme de communication (REST/HTTP) et d'indépendance. Nous réalisons durant ce TP, une application en JAVA capable d'interagir avec des valeurs de capteurs stockées dans une base de données (ou récupérées à travers une structure intergicielle) et en mesure de d'envoyer des commandes à des actionneurs suivant la logique d'implémentation considérée.

En terme de ressources, nous utilisons SpringBoot qui est un framework de développement applicatif Java open source pour l'injection des dépendances. Il est particulièrement recommandé pour le développement d'API. Nous utilisons aussi Maven qui est un outil de gestion des dépendances utilisé pour automatiser l'intégration continue lors du développement d'un logiciel.

Vous trouver le projet sur Github

<https://github.com/Aymen-bkz/Microservices.git>

Pour ce qui est de la gestion du projet nous utilisons la méthode scrum agile avec JIRA ou nous organisons notre projet en Sprints.

## 1 Scenarios

Pour notre projet, nous avons opté pour trois (03) scénarios qui interagissent avec un total de trois (03) capteurs et trois (03) actionneurs.

Nos scénarios choisis sont :

1. Si la luminosité est inférieure à un certain seuil et qu'il y a une présence dans la salle => Ouvrir les volets ;
2. Si le taux de CO<sup>2</sup> est supérieur à un certain seuil => Ouvrir les fenêtres ;
3. Si la luminosité est inférieure à un certain seuil et qu'il y a une présence dans la salle => Allumer les lumières.

## 2 Architecture

Notre architecture est découpé en plusieurs morceaux. Nous avons associé un micro-service pour chaque capteur et actionneur qui sont représentés par des SpringBoot Java projet dans notre code. Puis nous avons crée un controlleur qui va gérer et faire interagir ses micro-services à travers des requêtes GET et POST. Nous n'avons pas eu le temps de réaliser une véritable base de données, alors nous avons crée une liste de classe dans le code pour représenter les différents capteurs/actionneurs.

## 3 Implémentation

### 3.1 Micro-service Actionneurs: Lampe, Volet et Fenêtre

Dans cette famille de microservice nous pouvons récupérer les différents états des actionneurs à l'aide de requête GET et pouvons les commander pour leur faire changer d'état à l'aide de requête POST. Dans la suite, le mot "xxxx" fait référence à Lampe ou Volet ou Fenêtre.

- Avec le GetMapping `"/xxxx/status"` nous affichons l'état de tous les actionneurs de type xxxx.
- Le GetMapping `"/xxxx/all"` permet de récupérer une liste de tous les actionneurs xxxx.
- Le GetMapping `"/xxxx/id"` nous pouvons récupérer l'actionneur de type xxxx à partir de son id.

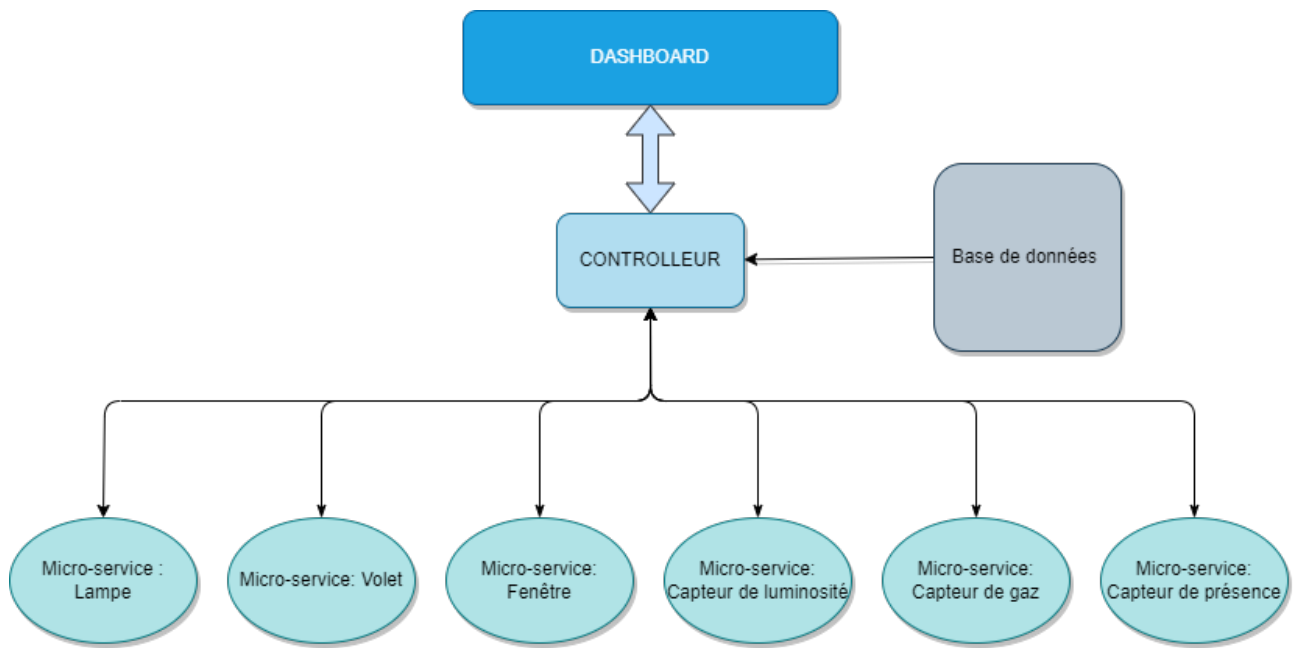


Figure 1: Architecture du projet

- Avec GetMapping "xxxx/etages/etages" nous pouvons récupérer une liste des actionneurs xxxx de l'étage en question.
- Nous pouvons la liste de tous les actionneurs xxxx dans la salle en question avec le GetMapping "xxxx/salles/salles"
- Le PostMapping "/xxxx/id/etat" permet de changer l'état de l'actionneur de type xxxx

les ports des actionneurs:

- Fenêtre 8082
- Volet 8087
- Lampe 8084

### 3.2 Micro-service Capteurs: Luminosité, Gaz et Présence

Dans cette famille de micro-service, nous pouvons récupérer les valeurs de chaque capteur. Les valeurs des capteurs sont générés de manière aléatoire à chaque appel du mapping en question. Dans la suite, le mot "xxxx" fait référence à Luminosité ou Gaz ou Présence.

- Avec le GetMapping "/xxxx/status" nous affichons l'état de tous les capteurs de type xxxx.
- Le GetMapping "/xxxx/all" permet de récupérer une liste de tous les capteurs xxxx.
- Le GetMapping "/xxxx/id" nous pouvons récupérer le capteur de type xxxx à partir de son id.
- Avec GetMapping "xxxx/etages/etages" nous pouvons récupérer une liste des capteurs xxxx de l'étage en question.
- Nous pouvons la liste de tous les capteurs xxxx dans la salle en question avec le GetMapping "xxxx/salles/salles"

les ports des capteurs:

- Luminosité 8085
- Gaz 8083
- Présence 8086

### 3.3 Micro-service Contrôleur

Le Contrôleur va exécuter les différents scénarios à l'aide de la requête `"/run/status"` qui va lancer tous les scénarios et les afficher de cette façon. En fonction des valeurs des capteurs qui sont gérés de manière aléatoire, les actionneurs vont changer leurs états.

le capteur de présence 11 de l'étage 1, salle 1 est true	le capteur de gas 11 de l'étage 1, salle 1 : 3689.0
le capteur de présence 12 de l'étage 1, salle 2 est true	le capteur de gas 12 de l'étage 1, salle 2 : 538.0
le capteur de présence 21 de l'étage 2, salle 1 est false	le capteur de gas 21 de l'étage 2, salle 1 : 1288.0
le capteur de présence 22 de l'étage 2, salle 2 est false	le capteur de gas 22 de l'étage 2, salle 2 : 3906.0
le capteur de Luminosité 11 de l'étage 1, salle 1 est 596.0	les fenetres 11 de l'étage 1, salle 1 sont ouvert
le capteur de Luminosité 12 de l'étage 1, salle 2 est 284.0	les fenetres 12 de l'étage 1, salle 2 sont fermé
le capteur de Luminosité 21 de l'étage 2, salle 1 est 364.0	les fenetres 21 de l'étage 2, salle 1 sont fermé
le capteur de Luminosité 22 de l'étage 2, salle 2 est 272.0	les fenetres 22 de l'étage 2, salle 2 sont ouvert
le Voilet 11 de l'étage 1, salle 1 est fermé	les Lampes 11 de l'étage 1, salle 1 sont éteinte
le Voilet 12 de l'étage 1, salle 2 est ouvert à 100.0	les Lampes 12 de l'étage 1, salle 2 sont allumé
le Voilet 21 de l'étage 2, salle 1 est fermé	les Lampes 21 de l'étage 2, salle 1 sont éteinte
le Voilet 22 de l'étage 2, salle 2 est fermé	les Lampes 22 de l'étage 2, salle 2 sont éteinte

Figure 2: Affichage des données brutes des différents scénarios

## 4 Exploitation des données / GUI

Pour avoir une vue globale et plus visuelle de nos différentes données des capteurs et des actionneurs, nous avons réalisé un Dashboard Web sous HTML/JS sous la page principal du contrôleur sur le chemin `"Microservice_controlleur/src/main/resources/static"` (8081) qui retourne les données des capteurs et commander les actionneurs manuellement.



Figure 3: IHM sous HTML

Remarque: pour pouvoir appliquer des requêtes GET/POSTE depuis la page HTML du Dashboard il faut rajouter "@CrossOrigin" dans chaque microservice.

```
@CrossOrigin(origins = "**")
```

Figure 4: Les règles Cross Origin

## 5 Intégration du projet

Tout au long du développement du projet, nous avons utilisé Github comme environnement de travail. Nous avons donc décidé de remplacer Jenkins par Github action pour la partie d'intégration continue du projet et pour chaque événement Push Pull.

<https://github.com/Aymen-bkz/Microservices/actions>

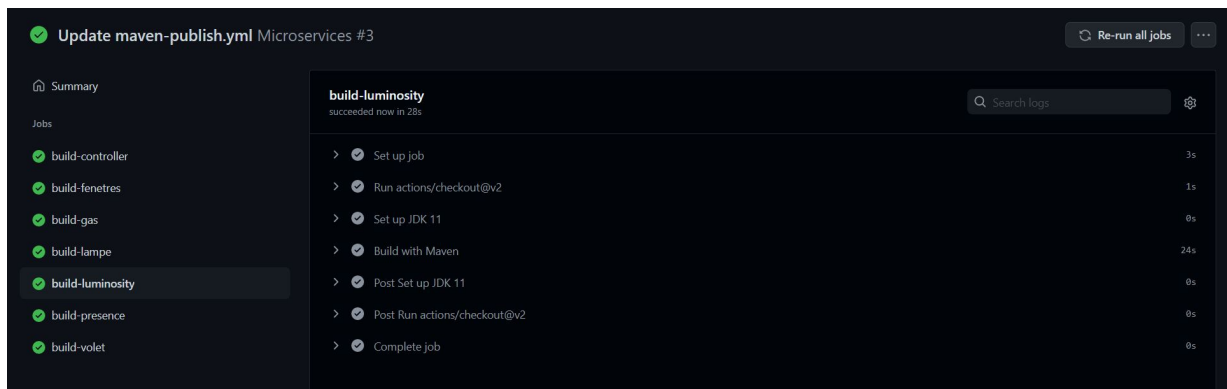


Figure 5: Github action

## 6 Organisation du projet

Durant le développement de notre application, nous avons opté pour la méthode agile scrum pour l'organisation de notre projets. Nous répartissons notre travail en sprints et chaque sprint comprend une (01) ou plusieurs stories et chacune des stories contient une ou plusieurs tâches. Pour ce faire, nous avons utilisé le software JIRA.

Pour ce qui est de l'architecture et des scénarios, nous en avons décidé tous ensemble lors d'un premier brainstorming. Pour ce qui est du développement de l'application, la répartition fut quelques peu complexe pour qu'elle soit détaillée ici, néanmoins, elle s'apparente à quelque chose comme ce qui suit :

- Berton Thomas : Développement des micro services liés aux capteurs ;
- Meslouh Mohamed-Arselan : Développement des micro services liés aux capteurs ;
- Boukezzata Aymen : Développement des micro services liés au contrôleur + Conception de la Dashboard.

Nous commençons par créer notre premier sprint qui contient les tâches principales :

AAT2-15	Création des scénarios	Thomas Berton	Thomas Berton	TERMINÉ(E)	Terminé	10/déc./21	17/déc./21	
AAT2-16	Conception de l'architecture	Thomas Berton	Mohamed Arselan MESLOUH	TERMINÉ(E)	Terminé	10/déc./21	05/janv./22	
AAT2-17	Création des différents projets pour les microservices	Mohamed Arselan MESLOUH	Thomas Berton	TERMINÉ(E)	Terminé	10/déc./21	05/janv./22	
AAT2-18	Création des classes	Aymen Boukezzata	Meslouh Mohamed Arselan	TERMINÉ(E)	Terminé	10/déc./21	05/janv./22	
AAT2-19	Simulation des premières applications	Aymen Boukezzata	Thomas Berton	TERMINÉ(E)	Terminé	10/déc./21	17/déc./21	...
AAT2-20	Création des Ressources	Mohamed Arselan MESLOUH	Thomas Berton	TERMINÉ(E)	Terminé	10/déc./21	05/janv./22	
AAT2-21	Mise en place des microservices dans le controleur	Thomas Berton	Thomas Berton	TERMINÉ(E)	Terminé	10/déc./21	17/déc./21	...

Figure 6: Sprint 1

La première story "création des scénarios" contient les tâches suivantes :





T	Clé	Résumé	Responsable	Rapporteur	Pr	État	Résolution	Création	Mise à jour	Date d'échéance
	AAT2-69	AAT2-15 / Scenario 3 : We would like to switch on the lights when there is presence in the room and the luminosity is under a certain threshold.	Non assigné	Thomas Berton	==	TERMINÉ(E)	Terminé	17/déc./21	05/janv./22	...
	AAT2-68	AAT2-15 / Scenario 2 : We would like to open the windows when there are too much CO2 in the room.	Non assigné	Thomas Berton	==	TERMINÉ(E)	Terminé	17/déc./21	05/janv./22	
	AAT2-67	AAT2-15 / Scenario 1: We would like to have the windows shutters opened when the luminosity is under a certain threshold and when there is presence in the room.	Non assigné	Thomas Berton	==	TERMINÉ(E)	Terminé	17/déc./21	05/janv./22	
	AAT2-15	Création des scénarios	Thomas Berton	Thomas Berton	==	TERMINÉ(E)	Terminé	10/déc./21	17/déc./21	

Figure 7: Création des scénarios



La troisième story "création des différents projets pour les micro-services" contient les tâches suivantes :



 <del>AAT2-46</del>	Microservice_volet	-		TERMINÉ(E) ▼
 <del>AAT2-47</del>	Microservice_controlleur	-		TERMINÉ(E) ▼
 <del>AAT2-48</del>	Microservice_fenetre	-		TERMINÉ(E) ▼
 <del>AAT2-49</del>	Microservice_gas	-		TERMINÉ(E) ▼
 <del>AAT2-50</del>	Microservice_lampe	-		TERMINÉ(E) ▼
 <del>AAT2-51</del>	Microservice_luminosity	-		TERMINÉ(E) ▼
 <del>AAT2-52</del>	Microservice presence	-		TERMINÉ(E) ▼

Figure 8: Création des projets SpringBoot pour les microservices

La quatrième story "création des classes" contient les tâches suivantes :















 <del>AAT2-60</del>	Lampe	-		TERMINÉ(E) ▼
 <del>AAT2-61</del>	Luminosity	-		TERMINÉ(E) ▼
 <del>AAT2-62</del>	Presence	-		TERMINÉ(E) ▼
 <del>AAT2-63</del>	Controlleur	-		TERMINÉ(E) ▼
 <del>AAT2-64</del>	Fenetre	-		TERMINÉ(E) ▼
 <del>AAT2-65</del>	Gas	-		TERMINÉ(E) ▼
 <del>AAT2-66</del>	Volet	-		TERMINÉ(E) ▼

Figure 9: Création des classes

La sixième story "Création des ressources" contient les tâches suivantes :















	AAT2-53	controlleurRessource	-		TERMINÉ(E) ▼
	AAT2-54	fenetreRessource	-		TERMINÉ(E) ▼
	AAT2-55	gasRessource	-		TERMINÉ(E) ▼
	AAT2-56	lampeRessource	-		TERMINÉ(E) ▼
	AAT2-57	luminosityRessource	-		TERMINÉ(E) ▼
	AAT2-58	presenceRessource	-		TERMINÉ(E) ▼
	AAT2-59	voletRessource	-		TERMINÉ(E) ▼

Figure 10: Création des ressources

Nous créons ensuite notre deuxième sprint qui contient les stories suivantes :




T	Clé	Résumé	Responsable	Rapporteur	Pr	État	Résolution	Création	Mise à jour	Date d'échéance	
	AAT2-72	Développement des microservices lié au contrôleur	Aymen Boukezzata	Thomas Berton	EN COURS	Non résolu	05/janv./22	05/janv./22			...
	AAT2-71	Développement des microservices liés aux actionneurs	Mohamed Arselan MESLOUH	Thomas Berton	TERMINÉ(E)	Terminé	05/janv./22	07/janv./22			
	AAT2-70	Développement des microservices liés aux capteurs	Thomas Berton	Thomas Berton	TERMINÉ(E)	Terminé	05/janv./22	07/janv./22			

Figure 11: Sprint 2

La première story "Développement des micro-services liés au contrôleur" contient les tâches suivantes :





	AAT2-79	Mise en relation des capteurs et des actionneurs	-		À FAIRE ▼
	AAT2-81	Mise en commun de l'ensemble des API	-		À FAIRE ▼

Figure 12: Développement des micro-services liés au contrôleur

La deuxième story "Développement des micro-services liés au actionneurs" contient les tâches suivantes :







	AAT2-76	Développement de l'API des fenêtres	-		À FAIRE ▼
	AAT2-77	Développement de l'API des volets	-		À FAIRE ▼
	AAT2-78	Développement de l'API des lampes	-		À FAIRE ▼

Figure 13: Développement des micro-services liés au actionneurs

La troisième story "Développement des micro-services liés au capteurs" contient les tâches suivantes :




	AAT2-73	Développement de l'API du capteur de présence	-		À FAIRE ▾
	AAT2-74	Développement de l'API du capteur de luminosité	-		À FAIRE ▾
	AAT2-75	Développement de l'API du détecteur de gaz	-		À FAIRE ▾
	AAT2-80	Générer des valeurs aléatoires pour les données des capteurs	-		À FAIRE ▾

Figure 14: Développement des micro-services liés au capteurs

Nous avons terminé avec le troisième sprint :



T	Clé	Résumé	Responsable	Rapporteur	Pr	État	Résolution	Création ▾	Mise à jour	Date d'échéanc
	AAT2-84	Rédaction du Rapport SOA	Non assigné	Thomas Berton	=	TERMINÉ(E)	Terminé	11/janv./22	14/janv./22	
	AAT2-83	Développement du DashBoard en HTML	Non assigné	Thomas Berton	=	TERMINÉ(E)	Terminé	11/janv./22	14/janv./22	

Figure 15: Sprint 3

## Conclusion

Durant l'élaboration de ce projet, nous avons eu l'occasion de nous familiariser avec des outils d'intégration continue dans le domaine du génie logiciel (Maven, SpringBoot). Nous avons aussi pu développer une application basée sur l'approche micro-service et le REST.

Nous avons travaillé en équipe ce qui nous a permis de mettre en pratique la méthode agile pour le management de projet en utilisant un outil software dédié (JIRA).

Dans l'ensemble, nous pouvons dire que le projet s'est bien déroulé et que l'objectif de se familiariser avec ces outils pour des étudiants comme nous (venants d'AE et d'électronique) nous a permis d'avoir une idée mieux construite sur les couches d'abstraction supérieures dans les architectures IoT.