

SOLER Lilian  
Informatique  
Rapport de stage 4A

Création de microservices pour l'architecture RAMi

Tome Principal  
ET  
Annexe

2022 - 2023  
08/05/2023 - 08/08/2023

## **Résumé**

Ce rapport est la synthèse du stage de 3 mois que j'ai réalisé à l'Université de Mons en Belgique dans le cadre de ma quatrième année à Polytech Grenoble. Au cours de mon stage, j'ai contribué au projet RAMi<sup>1</sup>. Il vise à répondre au besoin de suivi en temps réel des personnes âgées en tirant parti de l'Internet des Objets Médicaux. L'architecture comporte des couches de collecte, de traitement et d'analyse des données, intégrant des algorithmes d'IA<sup>2</sup> et la technologie blockchain<sup>3</sup> pour garantir la sécurité et la confidentialité des données. La force de RAMi réside dans son approche de logiciels open source<sup>4</sup>, offrant flexibilité et adaptabilité. Le projet répond à la demande croissante de solutions de soins de santé à distance pour les personnes âgées.

Ce stage a été très enrichissant pour moi, j'ai pu gagner en compétences et en expérience informatique et cela m'a permis de confirmer mes choix pour mes futurs projets professionnels.

## **Abstract**

This report is the summary of the 3-month internship I did at the University of Mons in Belgium as part of my fourth year at Polytech Grenoble. During my internship, I contributed to the RAMi project. It addresses the need for real-time monitoring of elderly individuals, leveraging the Internet of Medical Things. The architecture involves layers for data collection, processing, and analysis, integrating AI algorithms and blockchain technology to ensure data security and privacy. RAMi's strength lies in its open-source software approach, offering flexibility and adaptability. The project aligns with the growing elderly population's demand for remote healthcare solutions.

This internship was very rewarding for me, I was able to gain skills and computer experience and this allowed me to confirm my choices for my future professional projects.

## Table des matières

<b>Résumé</b>	<b>2</b>
<b>Abstract</b>	<b>2</b>
<b>Table des matières</b>	<b>3</b>
<b>Introduction</b>	<b>5</b>
<b>I. Présentation de l'entreprise</b>	<b>5</b>
A. Université de Mons	5
B. Institut de recherche Infortech	6
C. Service ILIA	7
<b>II. RAMi</b>	<b>8</b>
A. Introduction	8
B. Architecture RAMi	8
<b>III. Travail réalisé</b>	<b>10</b>
A. Etat de l'art des framework de FL existant	10
1. FL : Contexte et Comparaison avec le ML	10
a) Principes du FL	11
b) Comparaison avec le ML Centralisé	11
c) Défis et Complexités	11
d) Applications du FL	11
2. État de l'Art des Frameworks de FL	11
a) Frameworks existants	12
(1) Frameworks tout-en-un	12
(2) Frameworks horizontaux uniquement	12
(3) Frameworks spécialisés	12
b) Conclusion	13
3. Flower : Le Choix du service	13
4. Travaux Entrepris dans le Cadre de Flower	13
B. Implémentation de l'API pour les mesures	13
1. Conception et Choix Technologiques	13
2. Travail Effectué	14
a) Base de données	14
b) CRUDs	14
c) Documentation et test	14
d) Optimisation des performances pour l'envoi des mesures en batch	16
C. Implémentation d'un tableau de bord	17
<b>IV. Gestion de projet</b>	<b>20</b>
A. Organisation collaborative	20
B. Gestion des tâches et communication	21
C. Planification hebdomadaire	21
D. Revue Hebdomadaire	22
E. Gestion du code	22
F. Intégration et déploiement continue	23
G. Différents environnements de développement	24

<b>V. Compétences acquises</b>	<b>24</b>
A. Gérer et piloter un projet logiciel	24
B. Concevoir, développer et intégrer des briques logicielles	24
C. Administrer des infrastructures informatiques	25
<b>VI. Perspectives</b>	<b>25</b>
A. FL	25
1. Sécurité	25
2. Reproductibilité	25
B. API et tableau de bord	26
1. Sécurité	26
2. Blockchain	30
3. Fhire	30
4. Autres	30
<b>Conclusion</b>	<b>31</b>
<b>Annexes</b>	<b>32</b>
Glossaire	32
Remerciements	39
Bibliographie	40
<b>DOS DU RAPPORT</b>	<b>41</b>

## Introduction

Pendant ma quatrième année du cycle ingénieur de Polytech Grenoble, j'ai eu l'opportunité de réaliser un stage en tant qu'ingénieur informatique à l'Université de Mons chez Infortech [1], laboratoire de recherche affilié à l'Université [2] dans le département ILIA<sup>5</sup> [3], et contribuer activement au développement du projet RAMi. Ce projet a fait l'objet d'un article de recherche scientifique, qui certifie son importance et sa pertinence dans le domaine de la santé. C'est la portée du projet qui a déterminé mon choix de stage. C'est gratifiant de travailler sur un tel projet.

Mon tuteur était Saïd MAHMOUDI. J'ai effectué mon stage dans les locaux de l'Université au côté de mon tuteur et 4 autres stagiaires dont 2 qui étaient sur le même projet que moi.

J'ai pu travailler en équipe avec les contraintes que cela peut comporter. Cela m'a beaucoup apporté tant sur le plan professionnel que sur le plan personnel.

Le projet RAMi a été motivé par des enjeux sociétaux tels que le vieillissement de la population mondiale, le désir croissant des personnes âgées de rester autonomes à leur domicile et les défis auxquels le système de santé a été confronté lors de la pandémie de COVID-19. Dans ce contexte, il est essentiel de développer des systèmes de diagnostic et de suivi des patients à domicile, afin de réduire la charge économique et organisationnelle des professionnels de la santé.

L'objectif principal de RAMi est de concevoir une architecture en temps réel dédiée au suivi des patients âgés grâce à IoMT. L'IoMT englobe l'ensemble des dispositifs médicaux et des applications qui se connectent aux systèmes d'information de santé via des réseaux informatiques en ligne. Dans notre projet, nous avons développé une architecture innovante appelée RAMi, qui se compose de plusieurs couches interconnectées.

Au départ, mon rôle était lié à l'aspect de l'apprentissage fédéré<sup>7</sup>, qui permet de former des modèles<sup>22</sup> d'IA tout en préservant la confidentialité des données des patients. Cependant, au cours de mon stage, j'ai également été impliqué dans la mise en place de microservices au sein de l'infrastructure RAMi. Cette API<sup>8</sup> est un pré-requis pour la mise en œuvre de l'apprentissage fédéré dans le projet. En collaboration avec un autre stagiaire, nous travaillons sur le développement d'une API robuste, bien documentée et facile à maintenir qui permet d'enregistrer et de rapporter les données provenant de différents types de capteurs. De plus, nous travaillons sur la création d'un tableau de bord convivial qui permettra aux utilisateurs de consulter et d'analyser facilement les données collectées.

En parallèle, j'ai effectué un état des lieux des frameworks<sup>9</sup> existants en matière d'apprentissage fédéré. Cela m'a permis de prendre connaissance des dernières avancées dans ce domaine et d'évaluer les options disponibles pour intégrer cette approche de manière efficace et sécurisée dans notre projet RAMi.

La qualité professionnelle des microservices est primordiale, tout comme leur capacité à être déployés dans divers environnements. Ainsi, nous attachons une grande importance à la containerisation<sup>10</sup> des microservices<sup>11</sup> à l'aide de Docker<sup>12</sup> [4], ce qui facilite leur déploiement<sup>13</sup> et leur scalabilité<sup>14</sup>.

## I. Présentation de l'entreprise

### A. Université de Mons

L'Université de Mons (également connue sous le nom d'UMons) se situe en Belgique, dans la province de Hainaut, au cœur de la Wallonie, près de la frontière entre la Belgique et

la France. Elle possède des campus à Mons et à Charleroi. Fondée en 2009, l'UMons résulte de la fusion entre l'Université de Mons-Hainaut et la Faculté polytechnique de Mons.

*Figure 1* : Carte de la Belgique indiquant les différents emplacements des sites de l'UMons. (source: Intercarto)



Faisant partie des six universités francophones du pays, l'UMons se positionne en tant que quatrième plus grande université en termes d'effectifs. Les trois plus grandes institutions (l'Université catholique de Louvain, l'Université libre de Bruxelles et l'Université de Liège) comptent chacune plus de 25 000 étudiants, tandis que les deux plus petites (l'Université de Namur et l'Université Saint-Louis - Bruxelles) n'atteignent pas plus de 7 000 étudiants au total.

À l'UMons, près de 11 000 étudiants sont inscrits au sein de sept facultés et trois écoles, couvrant plus de 150 domaines de formation. Les champs d'études sont variés, allant de l'architecture à la linguistique, en passant par la chimie et l'urbanisme.

Cependant, l'UMons ne se limite pas à l'enseignement. Elle s'illustre également dans le domaine de la recherche, hébergeant 1 000 chercheurs et 450 doctorants répartis au sein de dix instituts de recherche organisés en une centaine de services.

Au cours de mon stage, j'ai été affilié à l'Institut de recherche Infortech.

## **B. Institut de recherche Infortech**

L'Institut de recherche en Technologie de l'Information et Sciences de l'Informatique de l'UMons, plus connu sous le nom d'Infortech, constitue un centre d'excellence au sein de l'université. Il rassemble environ 120 chercheurs et 35 doctorants, répartis dans 14 services spécialisés. Ces services se concentrent sur divers domaines tels que la collecte et la transmission de données, le Big Data<sup>15</sup> et le Cloud computing<sup>16</sup>, l'IA, le ML<sup>17</sup> et le DL<sup>18</sup>, le génie logiciel et algorithmique, le traitement du signal ainsi que les technologies de l'éducation.

Les activités menées par cet institut sont caractérisées par leur nature interdisciplinaire et leur orientation collaborative, impliquant la coopération avec d'autres organismes à des échelles allant du niveau provincial et régional jusqu'aux niveaux fédéral, européen et international.

Pendant la période de mon stage, j'ai eu l'opportunité de travailler au sein du service ILIA au sein de cet institut.

### **C. Service ILIA**

Le département ILIA a établi ses quartiers au sein de la Faculté Polytechnique de Mons, occupant les locaux qui remontent à 1545, lorsque le collège de Houdain fut fondé, et repris en 1837 par ladite Faculté.

Figure 2 : Site de Houdain, Faculté Polytechnique de Mons



Ce secteur d'Infotech se spécialise dans des domaines tels que l'Intelligence Artificielle, la vision par ordinateur, la gestion des données liées à l'IoT<sup>19</sup>, ainsi que le Cloud<sup>20</sup> et l'Edge Computing<sup>21</sup>. À l'intérieur de cette entité, divers projets sont initiés, dont :

- WALLeSmart : une plateforme qui recueille des données météorologiques, des informations sur l'état des sols, etc., dans le but d'améliorer la rentabilité des agriculteurs de Wallonie ;
- RAMi : une plateforme destinée à rassembler des données médicales afin de faciliter la prise en charge des patients.

C'est sur ce deuxième projet que s'est concentré mon stage.

## **II. RAMi**

### **A. Introduction**

À mesure que la population mondiale vieillit, le désir croissant des personnes âgées de préserver leur indépendance et les enseignements tirés de la récente pandémie de COVID-19 ont mis en évidence l'impératif pressant de développer des systèmes de surveillance et de

diagnostic à domicile. Ces systèmes visent à alléger les charges financières et organisationnelles qui pèsent sur les établissements de santé et les professionnels, tout en répondant aux besoins croissants de suivi médical. Dans ce contexte, l'IoMT émerge comme un domaine crucial de l'IoT, exigeant des traitements en temps réel et une fiabilité inébranlable.

En septembre 2022, l'architecture RAMi est proposée à travers un article scientifique. RAMi est une Architecture Temps Réel pour le Suivi des patients âgés grâce à l'IoMT. Cette architecture novatrice se compose de différentes couches. Tout d'abord, la couche "Things" capte les données provenant de capteurs ou de smartphones. Ensuite, la couche "Fog" repose sur une passerelle intelligente appelée MEC<sup>26</sup>, offrant une infrastructure intermédiaire pour le traitement des données en temps réel. Un composant cloud, associé à la blockchain et à l'intelligence artificielle, vient s'ajouter pour résoudre les problématiques spécifiques de l'IoMT. Le traitement des données se fait soit au niveau Fog, MEC ou cloud, en fonction des exigences de la charge de travail, des ressources et du niveau de confidentialité requis.

L'architecture RAMi permet le suivi continu des personnes âgées et des patients tout au long de leur hospitalisation et au-delà. De plus, elle intègre l'apprentissage fédéré pour former des algorithmes d'IA respectueux de la vie privée et de la confidentialité des données. Ces algorithmes détectent également les schémas d'intrusion, renforçant ainsi la sécurité de l'ensemble du système. La mise en place d'une blockchain locale facilite la répartition de la charge de travail entre les couches Fog, MEC et Cloud, tandis qu'une blockchain globale sécurise les échanges et le partage de données au moyen de contrats intelligents<sup>27</sup>. En somme, RAMi ouvre la voie à un suivi médical à domicile efficace et sécurisé, répondant aux défis actuels et futurs de la santé connectée.

## B. Architecture RAMi

*Figure 3* : Diagramme conceptuel et principaux logiciels de la partie Cloud de l'Architecture RAMi

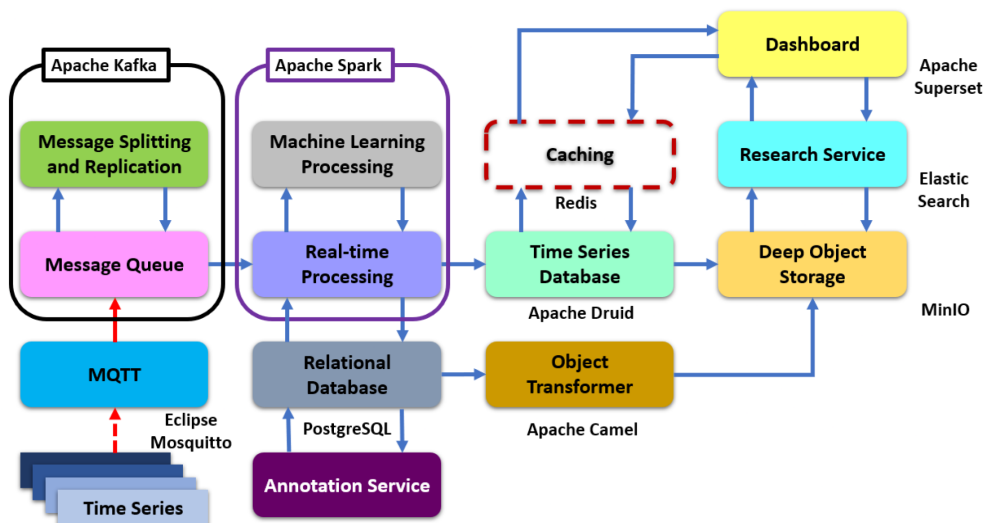
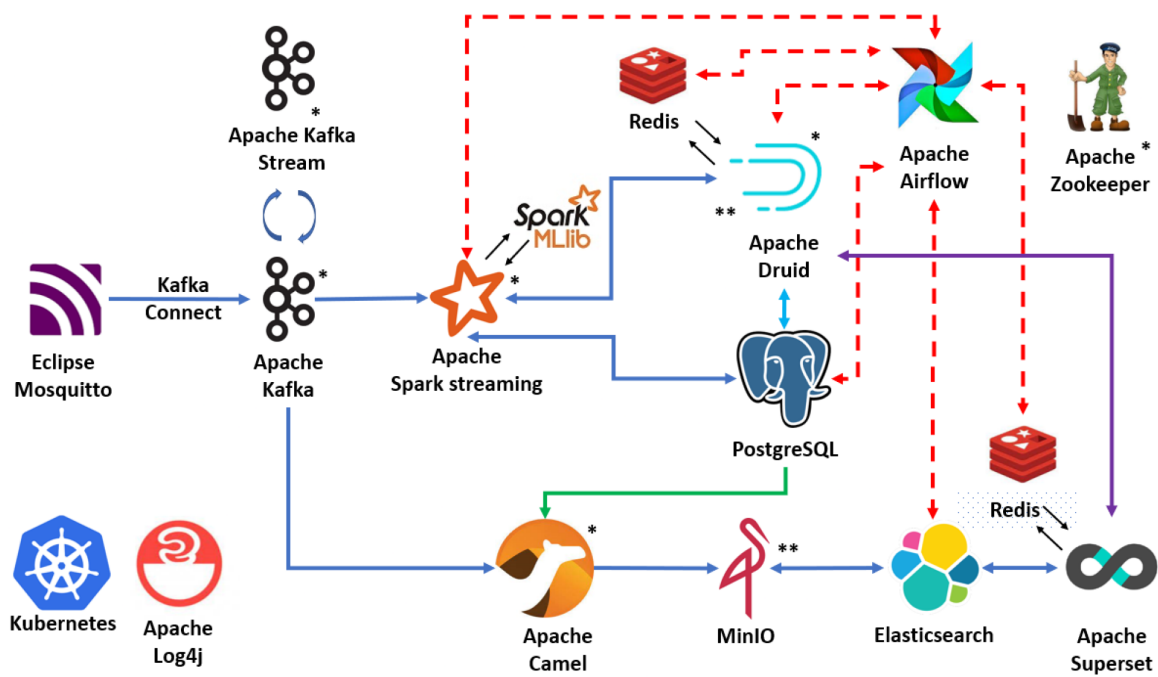




Figure 4 : Partie Cloud de l'Architecture RAMi



**Figure 5** : Diagramme conceptuel et logiciels principaux de la partie Fog de l'Architecture RAMi

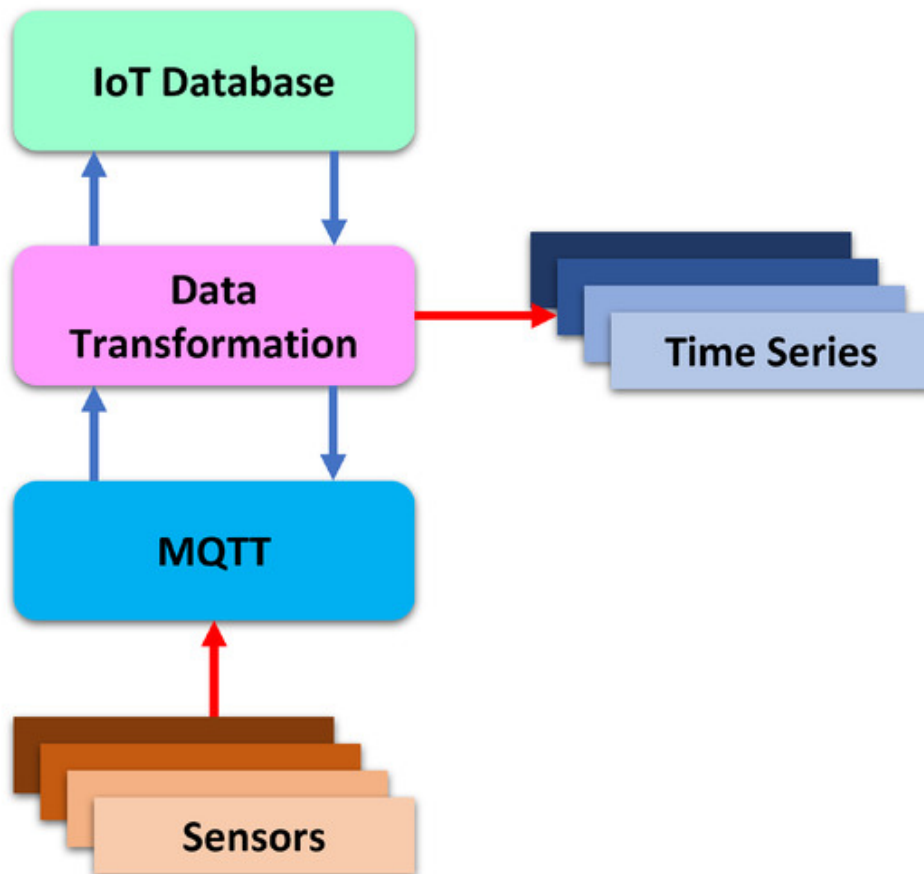
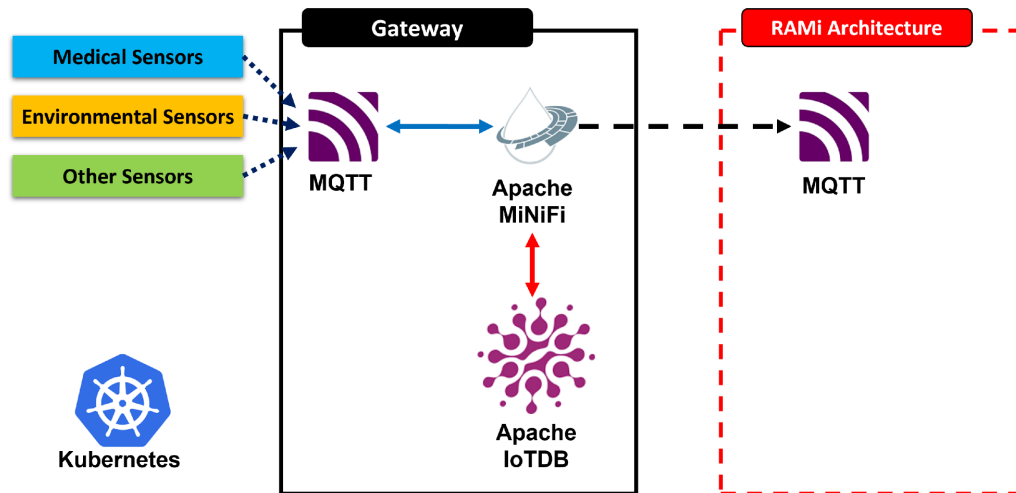


Figure 6 : Partie Fog de l'Architecture RAMi



### III. Travail réalisé

#### A. Etat de l'art des framework de FL existant

##### 1. FL : Contexte et Comparaison avec le ML

Le FL est une approche évolutive de ML qui vise à résoudre les défis liés à la centralisation des données. Contrairement aux approches de ML traditionnelles, où les données sont collectées et stockées de manière centralisée pour former un modèle global, le FL permet d'entraîner des modèles directement sur les appareils où les données résident, tout en préservant la confidentialité des données individuelles.

##### a) Principes du FL

L'idée fondamentale derrière le FL est de permettre à plusieurs appareils, tels que des smartphones, des capteurs IoT et d'autres dispositifs connectés, de collaborer à la formation d'un modèle global sans avoir à partager leurs données brutes. Au lieu de cela, seules les mises à jour de modèle (poids du modèle) sont partagées entre les appareils et agrégées de manière centralisée. Ce processus de mise à jour itérative permet au modèle global de s'améliorer au fil du temps, en tenant compte des caractéristiques uniques des données locales de chaque appareil.

##### b) Comparaison avec le ML Centralisé

Comparé au ML centralisé, le FL présente plusieurs avantages :

- Confidentialité des Données : Les données locales restent sur les appareils, ce qui élimine le besoin de partager des données sensibles ou privées avec un serveur central. Cela est particulièrement important dans les cas où la confidentialité des données est primordiale.
- Réduction des coûts de communication : Le partage des mises à jour de modèle est généralement plus efficace en termes de communication

que le partage de données brutes. Cela réduit la bande passante nécessaire et les coûts associés à la transmission de données.

- Gestion de Données Non-Équilibrées : Les données locales peuvent être non équilibrées, c'est-à-dire que différents appareils peuvent avoir des quantités différentes de données. Le FL gère cette disparité en permettant à chaque appareil de mettre à jour le modèle en fonction de sa propre distribution de données.

#### c) Défis et Complexités

Bien que le FL offre de nombreux avantages, il comporte également des défis et des complexités. Les disparités entre les appareils, la synchronisation des mises à jour de modèle, la gestion de la communication distribuée et la sécurité des données sont autant de problèmes complexes qui nécessitent des solutions novatrices.

#### d) Applications du FL

Le FL trouve des applications dans divers domaines, tels que la santé, la finance, l'IoT et plus encore. Dans le domaine de la santé, par exemple, les hôpitaux peuvent collaborer pour former des modèles de diagnostic sans partager de données patient sensibles. Dans l'IoT, les capteurs peuvent collaborer pour détecter des événements rares tout en préservant la vie privée des utilisateurs.

## 2. **État de l'Art des Frameworks de FL**

Le FL offre deux approches principales pour l'apprentissage à partir de données distribuées : le FL horizontal et le FL vertical. Chacune de ces approches cible des situations spécifiques et répond à des besoins différents en termes de collaboration et de confidentialité.

- FL Horizontal : Collaboration entre Pairs Similaires

Dans le cadre du FL horizontal, des appareils similaires collaborent pour former un modèle global. Imaginez plusieurs smartphones ou dispositifs IoT qui travaillent ensemble pour créer un modèle commun sans partager leurs données brutes. Chaque appareil effectue des mises à jour locales du modèle en fonction de ses propres données, puis partage ces mises à jour avec les autres appareils pour agrégation centralisée. Le FL horizontal est particulièrement adapté aux scénarios où des appareils similaires ont des données similaires et où la collaboration peut améliorer la performance globale du modèle. Cela garantit également que les données restent sur les appareils d'origine, préservant ainsi la confidentialité.

- FL Vertical : Complémentarité entre Données Distinctes

Le FL vertical se concentre sur la collaboration entre des appareils complémentaires qui détiennent des données différentes mais qui sont nécessaires pour former un modèle global. Prenons l'exemple d'un hôpital et d'une clinique de recherche médicale. L'hôpital peut avoir des données de patients et la clinique de recherche des données de laboratoire. Le FL vertical

permet à ces institutions de collaborer sans partager directement les données des patients. Au lieu de cela, les mises à jour de modèle contiennent des informations utiles pour améliorer la performance globale du modèle. Cette approche est pertinente dans les cas où différentes sources de données sont nécessaires pour former un modèle complet, tout en maintenant la confidentialité des données individuelles.

#### a) Frameworks existants

Dans le domaine du FL, plusieurs frameworks, bases de données et benchmarks ont été développés pour répondre aux besoins de ce paradigme de l'apprentissage distribué.

##### (1) Frameworks tout-en-un

Certains frameworks FL ont été conçus pour couvrir un large éventail de techniques dans les configurations FL horizontales et verticales. Parmi les exemples cités dans les articles, on trouve FATE, FedML, PaddleFL et Fedlearner. Ces frameworks visent à aborder différents cas d'utilisation et sont souvent maintenus par des institutions industrielles. Ils rassemblent différentes techniques et offrent une approche globale pour le FL.

##### (2) Frameworks horizontaux uniquement

D'autres frameworks, comme TFF, Flower et FLUTE, se concentrent principalement sur les algorithmes FL horizontaux. Par exemple, TFF de Google offre des API pour l'apprentissage fédéré et la conception d'algorithmes FL basés sur TensorFlow. Ces frameworks offrent des interfaces conviviales pour permettre aux utilisateurs d'adopter et de développer des algorithmes FL horizontaux.

##### (3) Frameworks spécialisés

Certains frameworks sont conçus pour résoudre des problèmes particuliers. Par exemple, CrypTen se concentre sur la sécurité des calculs multipartites, tandis que FedTree est destiné à l'entraînement fédéré d'arbres de décision. Ces frameworks ciblent des cas d'utilisation spécifiques et fournissent des solutions dédiées.

#### b) Conclusion

Malgré les benchmarks détaillés et les comparaisons minutieuses, aucun framework ne se démarque comme étant la solution dominante. Le choix d'un framework doit être judicieux et guidé par une compréhension approfondie des besoins spécifiques de l'entreprise et de l'application en question. Dans ce contexte, l'entreprise a choisi Flower en se basant sur des critères adaptés à ses objectifs.

En somme, le domaine du FL offre un éventail de choix pour répondre aux besoins d'apprentissage distribués, soulignant l'importance de

l'alignement entre les exigences de l'entreprise et les fonctionnalités offertes par les frameworks.

### **3. Flower : Le Choix du service**

Dans le cadre de l'architecture RAMi, le service a choisi Flower comme framework principal pour l'apprentissage fédéré. Flower est un framework open source conçu pour être convivial, scalable et flexible. Il vise à combler le fossé entre la recherche et la production, en offrant une variété de stratégies de fédération et une intégration transparente avec les frameworks ML existants tels que PyTorch et TensorFlow. Flower prend en charge divers scénarios de formation, allant de la formation sur périphérique aux déploiements distribués. C'est le choix idéal pour les entreprises qui cherchent à implémenter rapidement et efficacement l'apprentissage fédéré dans leurs solutions.

### **4. Travaux Entrepris dans le Cadre de Flower**

Le service a entrepris plusieurs travaux au sein du framework Flower pour le mettre en œuvre dans le contexte de l'architecture RAMi. Ces travaux comprennent une étude approfondie de la documentation de Flower, l'exploration des fonctionnalités de communication distribuée et la construction de stratégies de fédération pour s'adapter aux besoins spécifiques de l'entreprise. De plus, l'intégration d'algorithmes de sécurité, tels que la protection des données par chiffrement et la confidentialité différentielle, a été explorée pour garantir la sécurité des données et des modèles dans un environnement fédéré.

## **B. Implémentation de l'API pour les mesures**

Dans le cadre du projet RAMi, nous avons entrepris la mise en œuvre des microservices étudiés dans l'article. L'objectif ultime était de créer un écosystème capable de collecter et de gérer les données provenant de divers capteurs, pour ensuite les visualiser en temps réel et les préparer en vue d'une utilisation dans le FL. Pour atteindre ces objectifs, nous avons développé une API adaptable à différents types de capteurs en partant d'un PoC réalisé par Thomas PONS. Il a été mon binôme de développement sur cette partie et pour le tableau de bord.

### **1. Conception et Choix Technologiques**

L'architecture de notre solution s'appuie sur des choix technologiques réfléchis. Pour la création de notre API, nous avons opté pour Node.js, en utilisant TypeScript et PostgreSQL. Ce choix a été motivé par la compatibilité et la robustesse qu'offrent ces technologies. En particulier, PostgreSQL<sup>38</sup>, une base de données relationnelle<sup>37</sup> open source, a été choisie pour sa fiabilité, sa capacité à gérer les données de manière relationnelle et sa conformité avec les principes de l'open source.

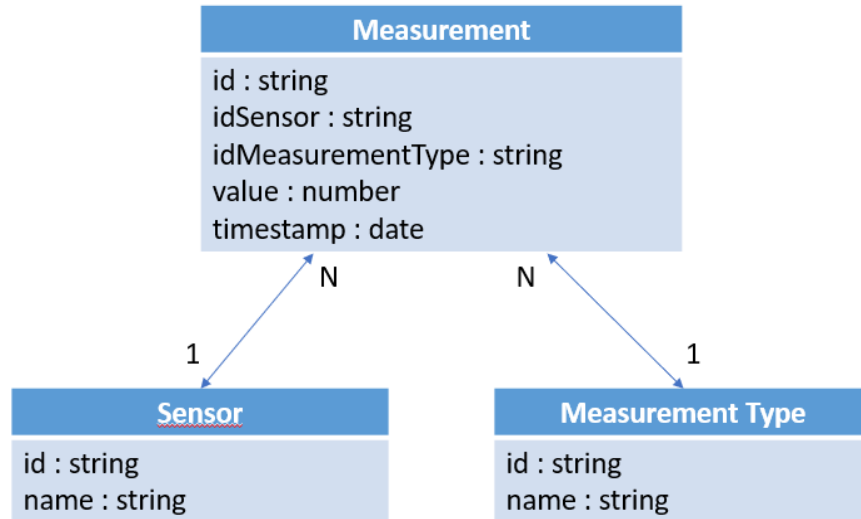
### **2. Travail Effectué**

#### **a) Base de données**

Notre première tâche a été de concevoir un modèle d'entités pour la base de données. Ce modèle reflète les relations entre les différentes entités du système. Dans la version initiale de l'API, nous avons identifié trois entités clés : "measurements" (mesures), "measurements Types" (types de

mesure) et "sensor" (capteur). Cette structure a été conçue pour garantir la souplesse nécessaire à la collecte et au stockage de diverses données provenant de capteurs.

*Figure 7* : Diagramme UML



#### b) CRUDs

Nous avons ensuite mis en œuvre des opérations CRUD pour chaque entité. Des routes ont été développées pour effectuer les opérations suivantes :


- **GET** : obtenir la liste des éléments d'une entité
- **POST** : créer un nouvel élément
- **PUT** : mettre à jour un élément
- **DELETE** : supprimer un élément

Pour l'entité "Measurement", nous avons enrichi ces opérations pour inclure des options de filtrage par date, type et/ou capteur lors de l'opération PUT.

#### c) Documentation et test

Nous avons accordé une attention particulière à la documentation tout au long du développement de l'API. Chaque nouvelle fonctionnalité et chaque route ajoutée ont été minutieusement documentées pour offrir une compréhension claire de leur utilisation et de leurs paramètres. De plus, nous avons veillé à maintenir un niveau élevé de couverture de code grâce à des tests unitaires exhaustifs réalisés avec le framework Jest.

**Figure 8 : Documentation API**

 **Swagger**  
Support by SMARTBEAR

# Umons sensor API

1.5 OAS 3.0

Umons sensor API

[Umons - Website](#)

[Send email to Umons](#)

Servers

http://localhost:3000/api/v1 - Local servers

## Sensor

Sensor related endpoints

GET

/sensors

Get all sensors or sensors by id or sensor by name

POST

/sensors

Create a new sensor

PUT

/sensors

Update a sensor name

POST

/sensors

Create a new sensor

Create a new sensor

Parameters

Try it out

No parameters

Request body

required

application/json

Example Value

Schema

```
{
  "name": "string"
}
```

Responses

Code	Description	Links
201	Sensor created	No links
<div>Media type</div> <div>application/json</div> <div>Controls Accept header.</div> <div><div>Example Value</div><div>Schema</div><div><pre>{   "id": "string",   "name": "string" }</pre></div></div>		
400	Bad request	No links
<div>Media type</div> <div>application/json</div> <div><div>Example Value</div><div>Schema</div><div><pre>{   "message": "string",   "status": 0,   "codeError": "string" }</pre></div></div>		
500	Server error	No links
<div>Media type</div> <div>application/json</div> <div><div>Example Value</div><div>Schema</div><div><pre>{   "message": "string",   "status": 0,   "codeError": "string" }</pre></div></div>		

Figure 9 : Code coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	100	100	100	100	
controllers	100	100	100	100	
home.ts	100	100	100	100	
measurement.ts	100	100	100	100	
measurementType.ts	100	100	100	100	
sensor.ts	100	100	100	100	
routes	100	100	100	100	
home.ts	100	100	100	100	
measurement.ts	100	100	100	100	
measurementType.ts	100	100	100	100	
routes.ts	100	100	100	100	
sensor.ts	100	100	100	100	
Test Suites: 5 passed, 5 total Tests: 135 passed, 135 total Snapshots: 0 total Time: 10.985 s					

d) Optimisation des performances pour l'envoi des mesures en batch

La question cruciale des performances a également été au centre de notre démarche. Nous avons particulièrement ciblé les défis posés par les données d'électrocardiogramme (ECG), qui génèrent un flux massif de données en mode batch<sup>33</sup> à haute fréquence. Pour répondre à cette exigence, nous avons minutieusement optimisé les performances de notre API.

Cette optimisation visait à assurer un traitement efficace et fluide de ces flux massifs, en prenant en compte les limites de débit imposées par les cartes embarquées des capteurs. En adaptant notre API pour gérer de manière efficiente les données envoyées en lots, nous avons réussi à garantir une collecte de données sans heurts, tout en minimisant les risques de ralentissements ou de perturbations.

Figure 10 : Tableau des différentes version pour l'envoi de mesure par groupe

	Version naïve	1ère optimisation	2nd optimisation
Temps moyen d'envoi de 100 mesures	960 ms	16 ms	22 ms
Nombre de caractères moyen d'une mesure	88	160	88
Format clair pour envoyer les données pour l'utilisateur	Oui	Non (nécessite de connaître les UUID)	Oui

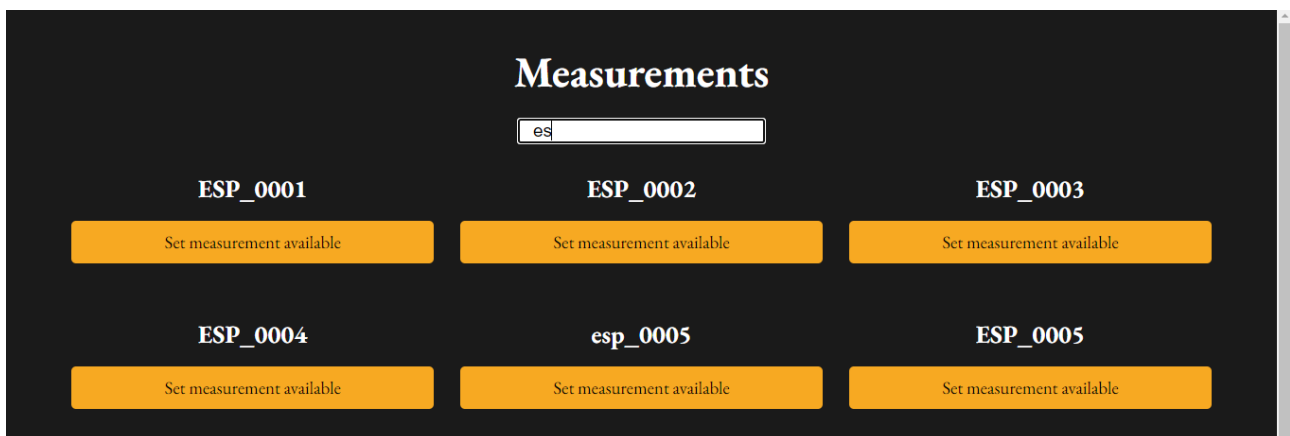


Message d'erreur explicite	Oui	Non	Oui
----------------------------	-----	-----	-----

### C. Implémentation d'un tableau de bord

Après avoir établi un solide fondement avec l'implémentation de l'API pour la collecte et le stockage des données générées par divers capteurs, une nouvelle phase critique s'est présentée : la création d'un tableau de bord pour la visualisation et l'exploitation de ces données. Alors que l'API assurait la centralisation des informations, le tableau de bord devrait permettre leur présentation conviviale, en temps réel, pour offrir une compréhension approfondie et une analyse proactive des différents types de mesures.

*Figure 11* : Vue des capteurs



*Figure 12* : Vue des mesures en cards

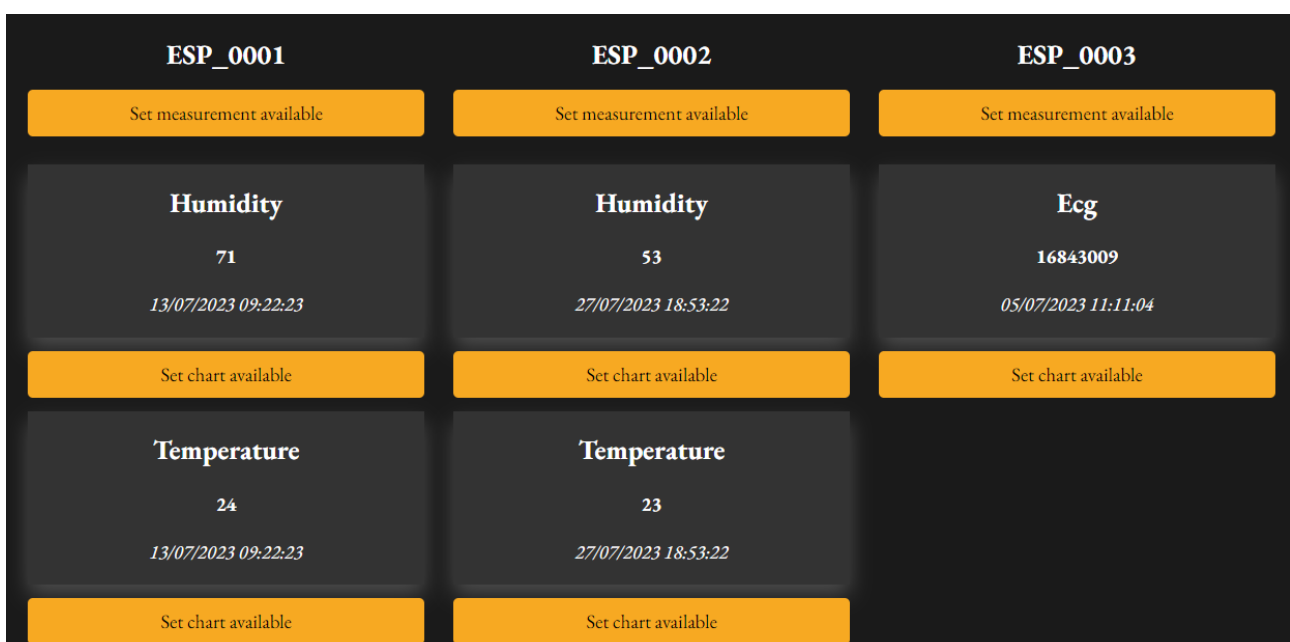


Figure 13 : Vue des mesures en tableau

ESP\_0001

Set measurement available

Humidity

71

13/07/2023 09:22:23

Value	Timestamp
71	13/07/2023 09:22:23
71	13/07/2023 09:22:22
71	13/07/2023 09:22:20
71	13/07/2023 09:22:18
71	13/07/2023 09:22:16

ESP\_0002

Set measurement available

Humidity

53

27/07/2023 18:53:22

Value	Timestamp
53	27/07/2023 18:53:22
53	27/07/2023 18:53:20
53	27/07/2023 18:53:18
53	27/07/2023 18:53:16
53	27/07/2023 18:53:14

ESP\_0003

Set measurement available

Ecg

16843009

05/07/2023 11:11:04

Value	Timestamp
16843009	05/07/2023 11:11:04
16843009	05/07/2023 11:11:03
16843009	05/07/2023 11:11:02
16843009	05/07/2023 11:11:02
16843009	05/07/2023 11:11:01

Figure 14 : Vue des mesures en graphique (ordinateur)

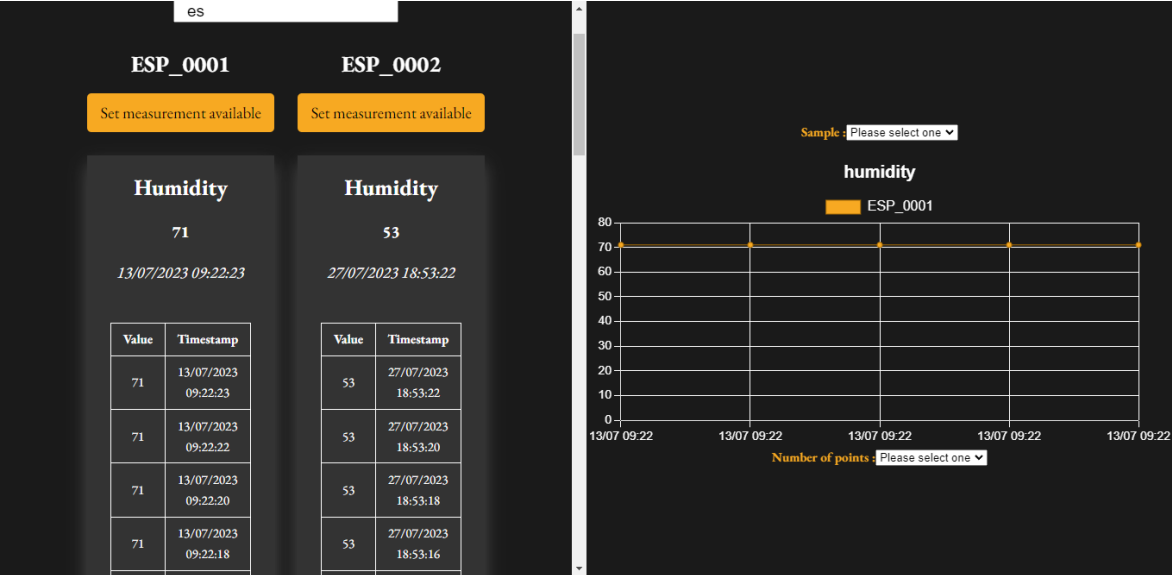


Figure 15 : Vue des mesures en graphique (téléphone)

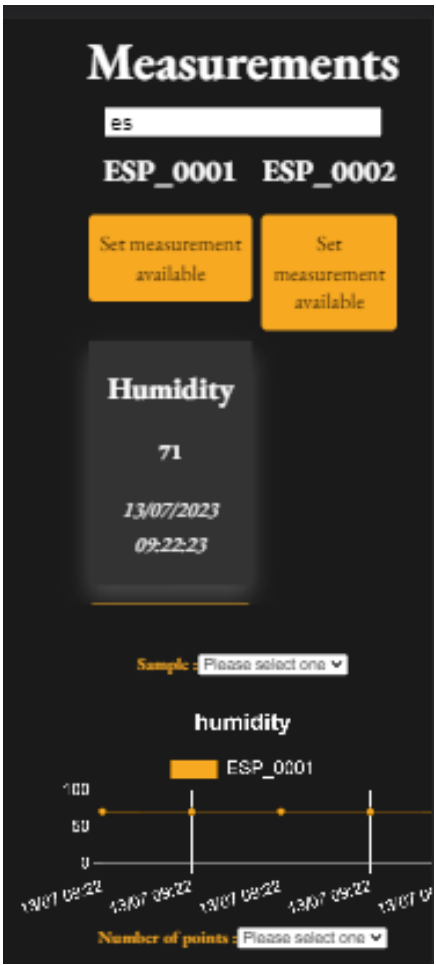
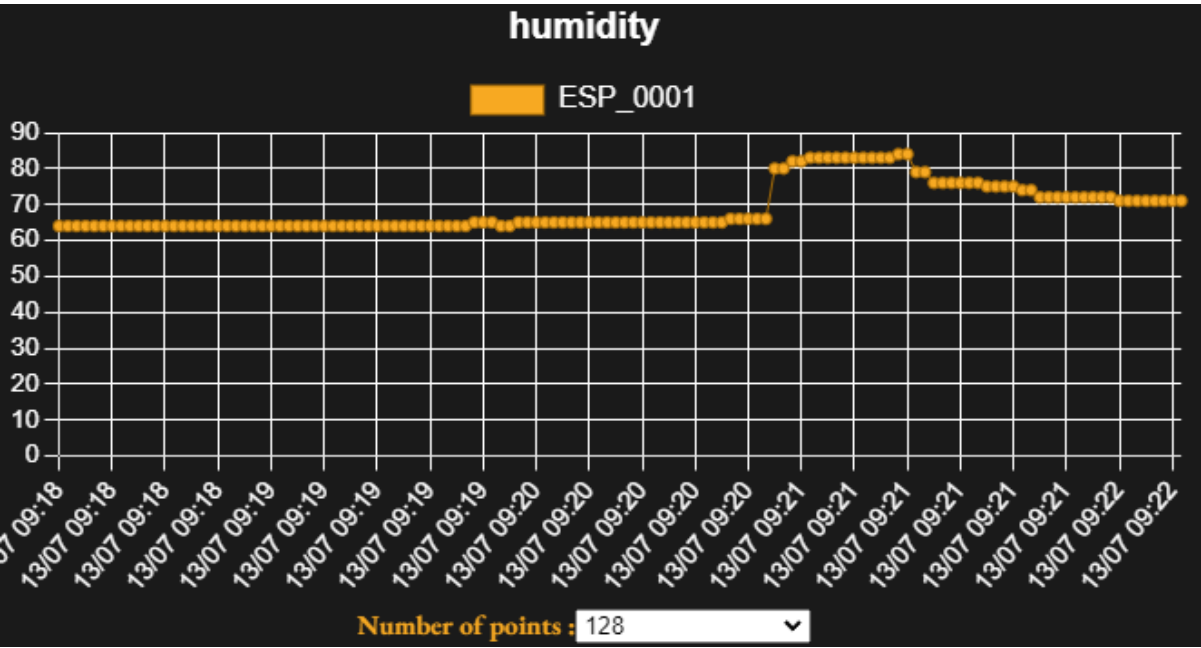
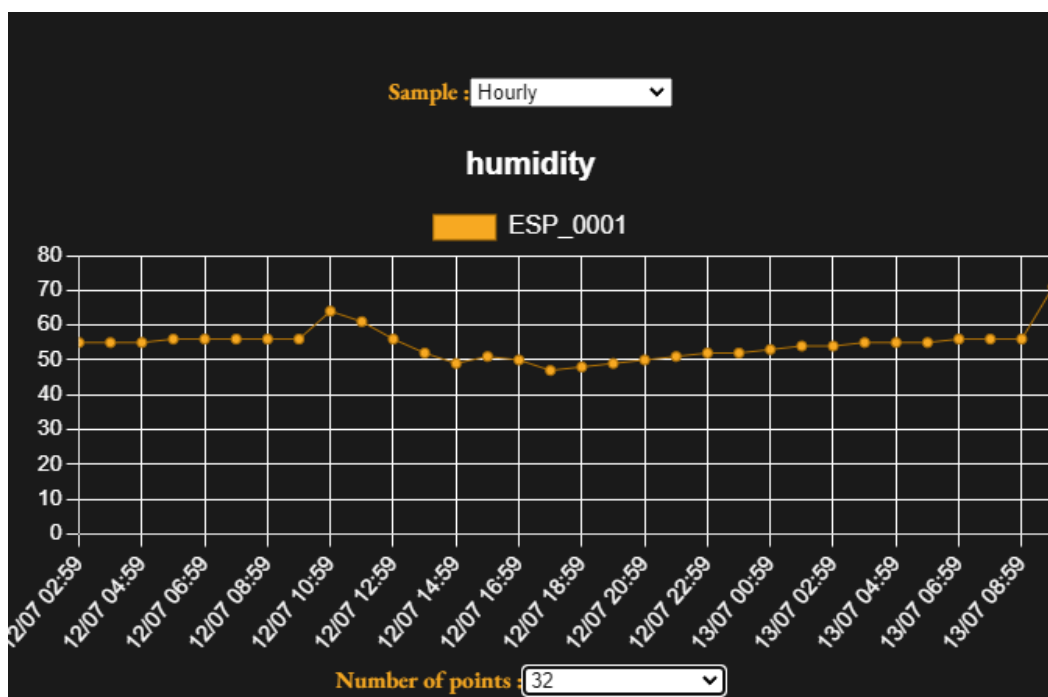


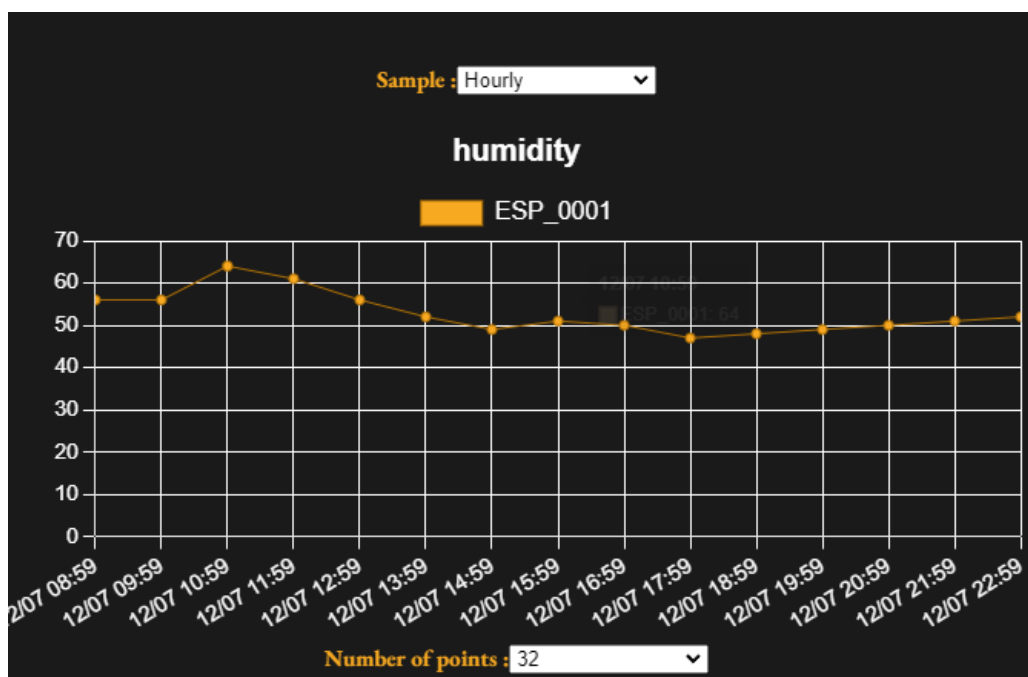
Figure 16 : Choix du nombre de mesures



*Figure 17* : Choix de l'échantillonnage



*Figure 18* : Choix du zoom horizontal



#### IV. Gestion de projet

Pour assurer une collaboration efficace et une gestion optimale du projet, nous avons adopté une approche agile<sup>23</sup>, structurée en plusieurs étapes.

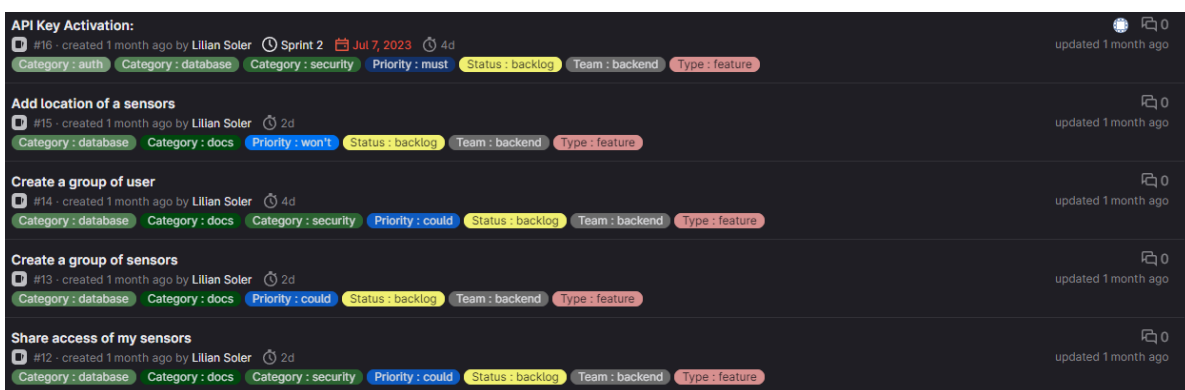
## A. Organisation collaborative

Travaillant en binôme avec un autre étudiant stagiaire, nous avons privilégié la coopération et la répartition des tâches pour optimiser notre productivité. Cette collaboration s'est appuyée sur deux dépôts GitLab distincts pour gérer les versions du front-end<sup>58</sup> et du back-end<sup>57</sup> de notre application.

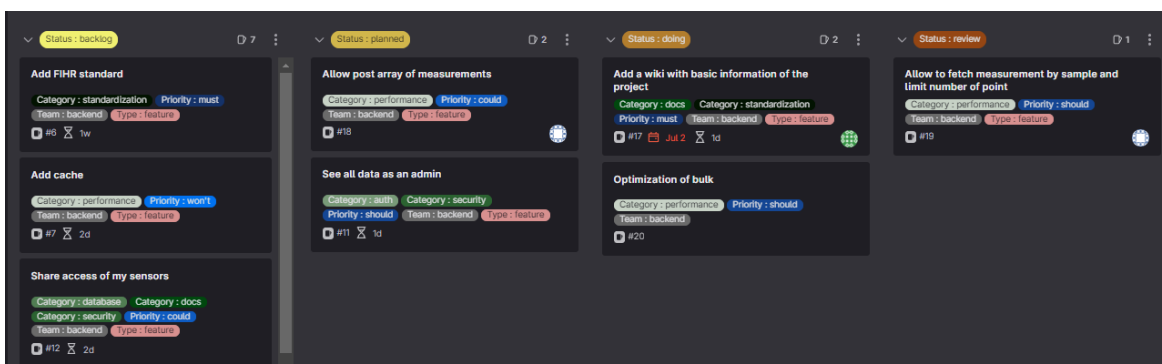
## B. Gestion des tâches et communication

Lorsque nous identifions des améliorations ou des fonctionnalités à ajouter, nous ouvrons une Issue sur GitLab, précisant la problématique et l'objectif visé. Nous utilisons des "tags" pour catégoriser les problématiques en fonction de la méthode MoSCoW<sup>24</sup>, de leur domaine (sécurité, optimisation, etc.) et de leur état (backlog<sup>59</sup>, planned<sup>60</sup>, doing<sup>61</sup>, done<sup>62</sup>, reviewed<sup>63</sup>). Ces tags permettent une vision claire des tâches à réaliser et sont organisés dans un tableau Kanban<sup>25</sup> pour le suivi visuel.

*Figure 19* : Issues



*Figure 20* : Tableau Kanban



## C. Planification hebdomadaire

Au début de chaque semaine, nous planifions les tâches prioritaires à accomplir dans la semaine. Nous effectuons une estimation du temps nécessaire pour chaque tâche et répartissons le travail en conséquence.

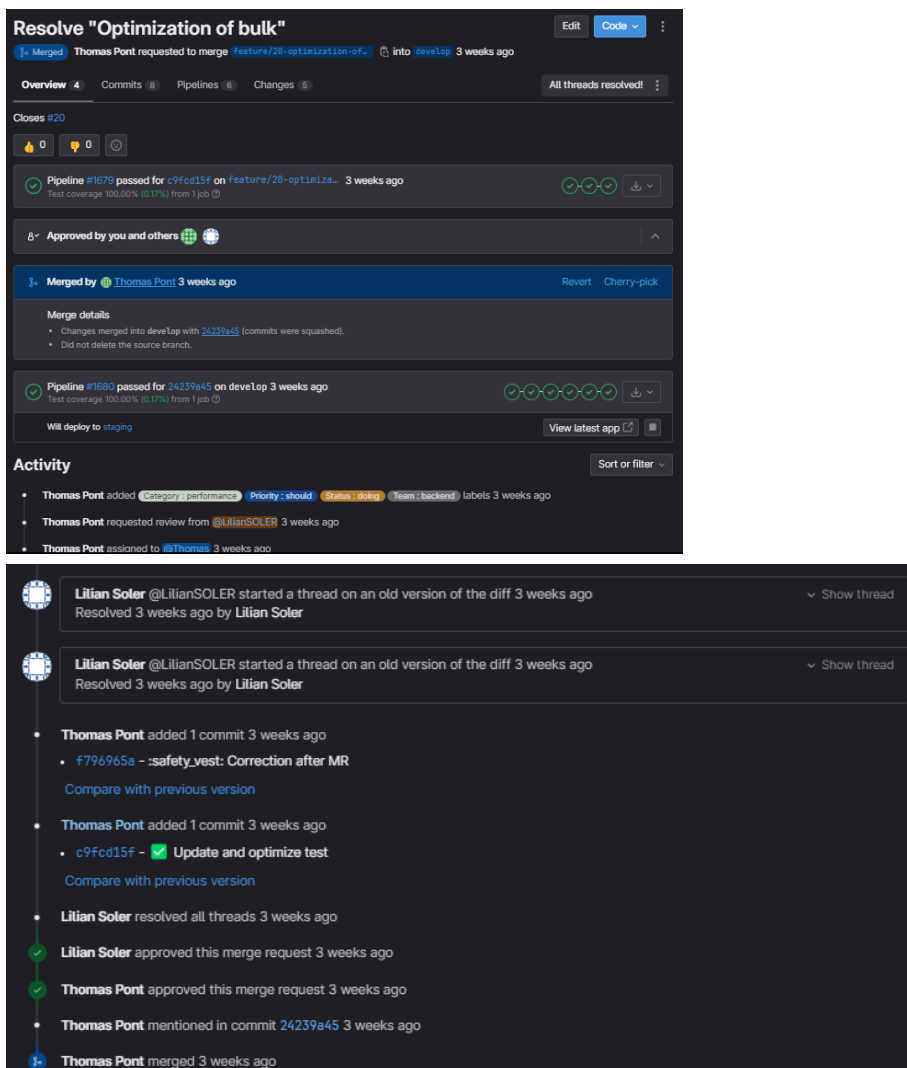
## D. Revue Hebdomadaire

En fin de semaine, nous avons une réunion avec notre tuteur pour présenter le travail effectué et discuter des orientations futures.

## E. Gestion du code

Lorsque nous commençons à travailler sur une issue, nous créons une branche spécifique correspondant à cette issue sur le dépôt GitLab. Nous travaillons sur cette branche pour réaliser les objectifs fixés. Une fois la tâche achevée, nous soumettons une MR<sup>64</sup> et demandons une validation de l'autre personne. Cette approche de code review permet de détecter les erreurs et d'optimiser le code. Après avoir corrigé les retours, la branche issue est fusionnée dans la branche develop.

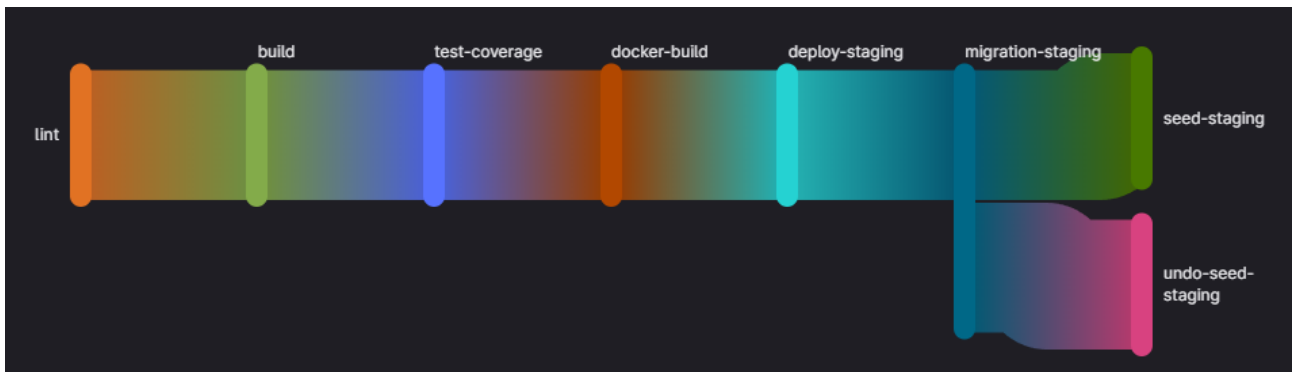
*Figure 22 : Merge request*



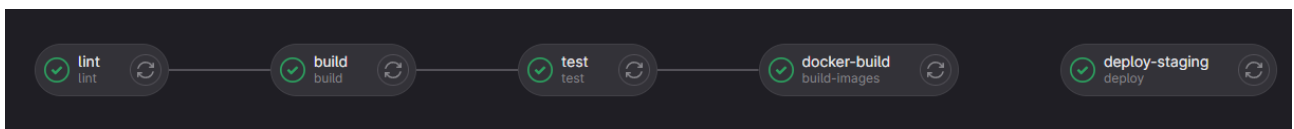
## F. Intégration et déploiement continue

Pour garantir la qualité et la fiabilité du code, nous avons mis en place des étapes d'intégration continue, y compris des tests, des builds et des vérifications de conformité. Nous utilisons également des conteneurs Docker pour assurer la portabilité et la répétabilité du déploiement.

*Figure 23* : Pipeline de l'API



*Figure 24* : Pipeline du tableau de bord



## G. Différents environnements de développement

Un script automatisé sur GitLab déclenche le déploiement du front-end et du back-end dès que des modifications sont apportées aux branches main et develop. Cela nous permet d'avoir un environnement de test pour valider les fonctionnalités avant de les déployer dans un environnement de production.

En résumé, notre gestion de projet repose sur des méthodes agiles, des outils de versionnement, des processus de validation rigoureux et une collaboration étroite avec le tuteur et les collègues. Cela nous permet de maintenir une communication fluide, de garantir la qualité du code et de répondre aux besoins changeants du projet tout en respectant les délais.

## V. Compétences acquises

### A. Gérer et piloter un projet logiciel

Lorsque j'ai participé à l'élaboration du cahier des charges, j'ai compris l'importance de définir clairement les objectifs d'un projet avant de plonger dans le développement. En

adoptant une approche agile, j'ai appris à gérer le flux de travail de manière itérative, en adaptant constamment nos priorités en fonction des besoins en évolution. En utilisant la forge GitLab du service pour gérer les versions du code et suivre les tâches, j'ai appris à organiser efficacement le travail d'une équipe. La création d'un espace de collaboration en ligne m'a montré comment la communication fluide peut améliorer la productivité et la synergie entre les membres de l'équipe.

## **B. Concevoir, développer et intégrer des briques logicielles**

En travaillant sur l'architecture du système et en choisissant des technologies comme Node.js et PostgreSQL, j'ai réalisé à quel point une décision d'architecture bien prise peut simplifier tout le processus de développement ultérieur. L'adaptation d'un système existant m'a exposé aux réalités de la maintenance et de l'évolution de logiciels existants, me montrant comment équilibrer les améliorations tout en préservant les fonctionnalités existantes. Dockeriser l'API pour le déploiement m'a ouvert les portes du monde des infrastructures modernes, ce qui m'a encouragé à en apprendre davantage sur les environnements de conteneurs.

## **C. Administrer des infrastructures informatiques**

Mon implication dans le déploiement et la gestion d'infrastructures m'a montré à quel point l'automatisation peut simplifier les tâches répétitives. En travaillant aux côtés de l'administrateur système expérimenté du service, j'ai appris énormément sur les rouages internes qui ne deviennent visibles qu'en situation de production réelle. Dockeriser l'API a été une expérience d'apprentissage marquante, qui m'a montré comment mettre en place des environnements de manière flexible. Mon rôle m'a permis d'acquérir un aperçu concret de l'impact d'une architecture basée sur les microservices sur la résilience et la scalabilité.

# **VI. Perspectives**

## **A. FL**

### **1. Sécurité**

L'utilisation du FL présente des avantages significatifs, mais nécessite une vigilance accrue en matière de sécurité pour faire face à des menaces potentielles. Parmi les défis de sécurité spécifiques auxquels le FL est exposé, on peut citer les attaques de type "model poisoning", les attaques de "data poisoning", les attaques de type "backdoors", ainsi que les attaques d'inférence de confidentialité.

L'encryption homomorphe, qui permet d'effectuer des opérations sur des données chiffrées sans avoir à les déchiffrer, offre un potentiel prometteur pour renforcer la sécurité du FL. En utilisant cette technique, les modèles d'apprentissage pourraient travailler directement avec des données cryptées, minimisant ainsi les risques d'exposition de données sensibles tout en permettant un apprentissage collaboratif efficace. Cependant, le développement et l'implémentation de l'encryption homomorphe ne sont pas sans défis, notamment en ce qui concerne



les performances et la complexité. Il est donc crucial de poursuivre les travaux dans ce domaine pour optimiser et rendre plus praticable cette technologie.

Au-delà de l'encryption homomorphique, il est également crucial de continuer à explorer d'autres techniques et approches de sécurité. La recherche de nouvelles méthodes de protection des modèles et des données, telles que la détection avancée des attaques, l'utilisation de mécanismes de vérification décentralisée et la mise en place de contrôles d'accès sophistiqués, est essentielle pour assurer une sécurité globale dans un environnement FL. L'innovation et la collaboration avec des experts en cybersécurité permettront d'élaborer des solutions adaptées aux défis spécifiques du FL.

## **2. Reproductibilité**

La reproductibilité occupe une place cruciale dans le développement et l'adoption réussie du FL. En tant qu'approche distribuée, le FL peut présenter des défis en matière de reproductibilité des résultats, ce qui peut affecter la confiance des utilisateurs et la validité des analyses menées. C'est pourquoi Flower a pris des mesures significatives pour aborder cette question en lançant un sprint dédié à la recherche sur la reproductibilité, qui comprend une offre généreuse de 100 000 euros pour soutenir ces efforts.

Au cœur de ces initiatives réside la volonté de développer des méthodologies et des pratiques qui permettent de garantir que les résultats obtenus à travers le FL sont cohérents et reproductibles. Cela implique l'identification et la documentation précise des paramètres d'entraînement, des architectures de modèles et des données utilisées. En mettant l'accent sur la transparence, Flower cherche à établir des normes qui facilitent la validation et la réplique des résultats par d'autres chercheurs et praticiens.

Le sprint de recherche sur la reproductibilité témoigne de l'engagement de Flower à jouer un rôle de premier plan dans la promotion de bonnes pratiques dans le domaine du FL. En favorisant la collaboration et le partage d'expériences, ces efforts visent à créer un écosystème où les résultats du FL peuvent être examinés, validés et utilisés en toute confiance. Par la suite, la recherche sur la reproductibilité permettra de renforcer l'adoption et l'intégration du FL dans des domaines tels que la santé, la finance et l'industrie, où des décisions cruciales reposent sur des données fiables et reproductibles.

## **B. API et tableau de bord**

### **1. Sécurité**

La section liée à la sécurité a été développée de manière exhaustive, cependant, au moment de mon départ de Mons, elle n'avait pas encore fait l'objet de tests ni de documentation (partielle). Néanmoins, ces étapes sont prévues pour être réalisées avant la fin du stage de mon binôme.

Cette partie englobe l'authentification qui intervient lors de la création ou de l'accès aux mesures, ainsi qu'aux capteurs et aux types de mesures. Toute création de capteur ou de type de mesure requiert une demande initiale de la part de l'utilisateur, suivie d'une approbation par un administrateur. Une fois approuvées, les mesures générées par les capteurs sont accessibles uniquement aux utilisateurs associés à ces derniers.

Une structure de rôles et de gestion de rôles a également été mise en place pour réguler les niveaux d'accès et les autorisations des utilisateurs.

*Figure 25* : Documentation CRUD User

User CRUD operations			API to perform CRUD operations on users	⌵
POST	/login	Login		⌵
POST	/signup	Signup		⌵
PUT	/update/role	updateRole		⌵
GET	/verify/adminPanel	verifyAdminPanel		⌵
GET	/all	getAll		⌵

*Figure 26* : Formulaire de connexion

HomeAdminUserLog out

Log in

Log in to access all features available

Email

Password

Log in

Have no account? Sign up

Figure 27 : Panneau d'administration

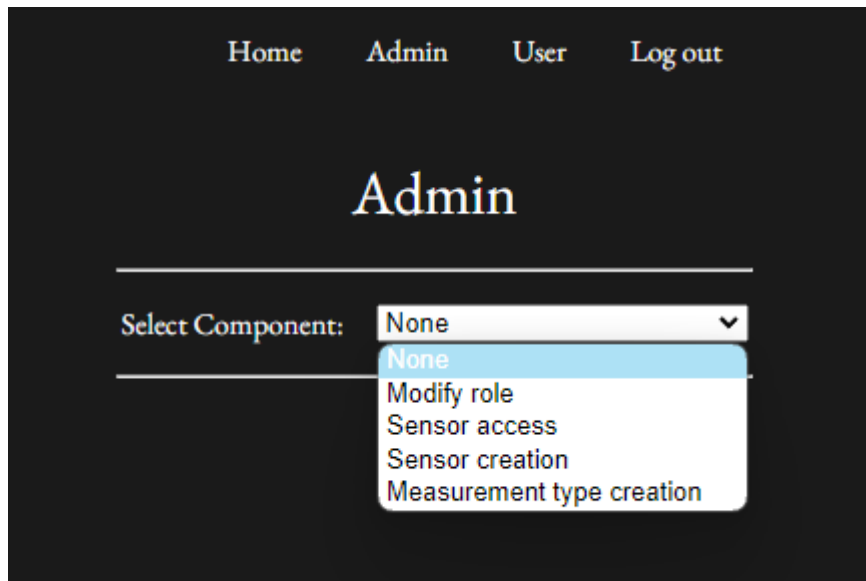


Figure 28 : Gestion des utilisateurs

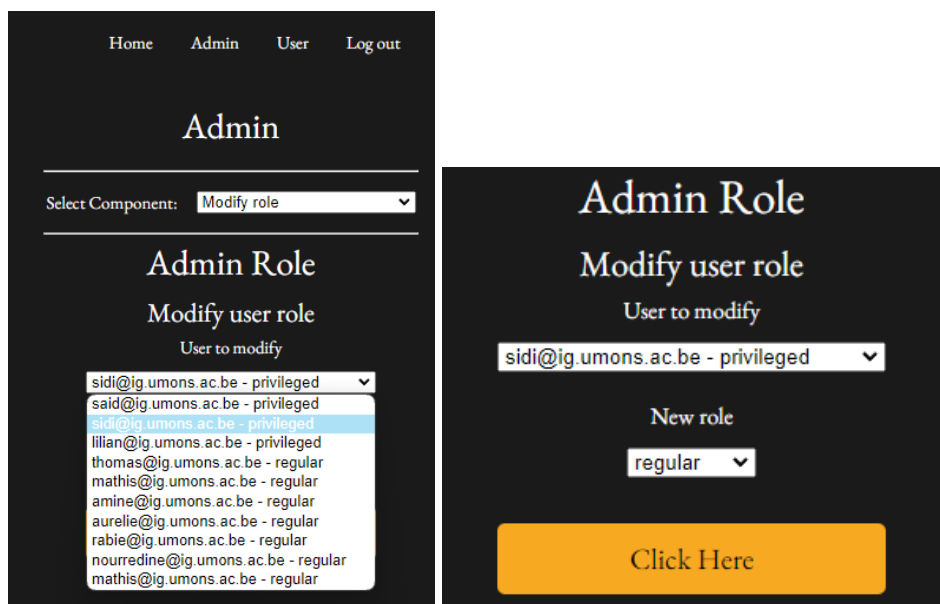


Figure 29 : Panneau utilisateur

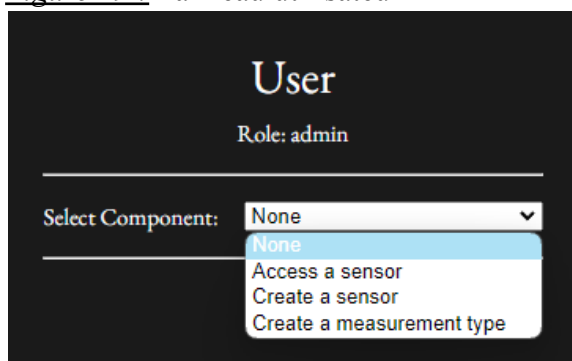


Figure 29 : Demande création capteur

Home Admin User Log out

## User

Role: admin

---

Select Component: Create a sensor ▼

---

### Ask for sensor creation

Email

soler.lilian64@gmail.com

Item

super\_capteur

Keep email

☐

Submit

Figure 30 : Demande accès capteur

### Ask for sensor access

Email

soler.lilian64@gmail.com

Item

super\_unit

Keep email

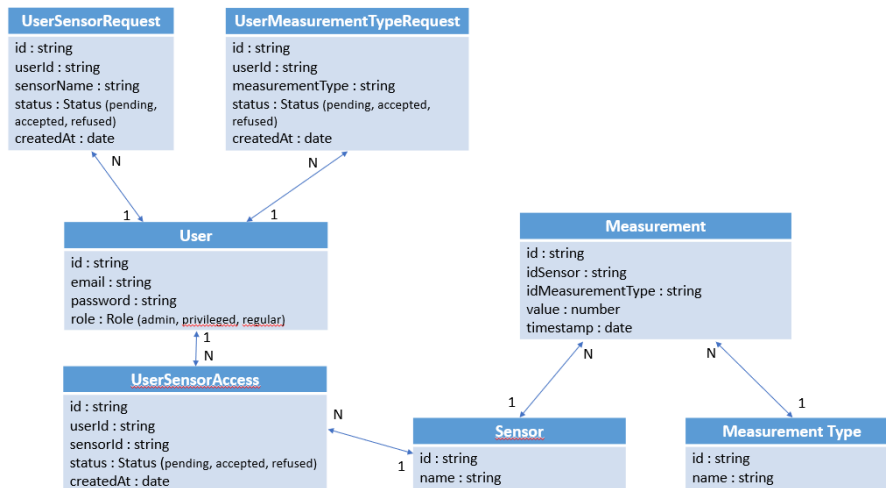
☐

Submit

**Figure 31** : Modération création capteur

Admin Request					
Grant or revoke sensor creation					
Filter by status <span>Show All</span>					
User	Sensor	Status	Created at	Accept	Reject
thomas@ig.umons.ac.be	super_unit-2	pending	15/08/2023 15:26	Accept	Rejected
thomas@ig.umons.ac.be	super_unit_5	pending	15/08/2023 15:26	Accept	Rejected
thomas@ig.umons.ac.be	esp32/004	pending	15/08/2023 15:25	Accept	Rejected
thomas@ig.umons.ac.be	OTHER6SENSOR	pending	15/08/2023 15:25	Accept	Rejected
lilian@ig.umons.ac.be	super_unit	pending	15/08/2023 15:25	Accept	Rejected

**Figure 32** : Diagramme UML



## 2. Blockchain

Une évolution significative de l'application est prévue, notamment avec l'intégration de la technologie blockchain. Cette mise en œuvre complexe sera effectuée par un doctorant qui reprendra le projet, parmi d'autres contributions. L'ajout de la blockchain constitue une avancée majeure en termes de sécurité, de transparence et de traçabilité des données.

## 3. Fhire

Une perspective importante pour l'évolution de l'API est l'adoption du formatage au format FHIR (Fast Healthcare Interoperability Resources). FHIR, un standard d'échange de données de santé publié par HL7, offre une norme bien établie pour la communication et l'intégration de données médicales entre différents systèmes. En adoptant ce format, Flower pourrait faciliter son intégration avec

d'autres pipelines et systèmes de santé, ouvrant ainsi la voie à une interopérabilité fluide et sécurisée.

#### **4. Autres**

Plusieurs autres fonctionnalités potentielles ont été envisagées pour enrichir davantage l'API, bien qu'elles n'aient pas encore été implémentées à ce stade. Parmi ces perspectives, l'idée de grouper les utilisateurs pour attribuer des accès par groupe ou sous-groupe revêt un grand intérêt. Cette fonctionnalité permettrait une gestion plus fine des autorisations, simplifiant la gestion des utilisateurs à grande échelle tout en préservant la confidentialité des données.

De même, le regroupement des capteurs constitue une possibilité intrigante. Cette fonctionnalité pourrait faciliter la gestion des capteurs similaires ou liés, en simplifiant les opérations telles que la surveillance et la maintenance. L'ajout de la localisation des capteurs représente une perspective pratique, offrant des informations géospatiales essentielles pour des analyses contextuelles plus riches.

Envisager des plages de valeurs pour les mesures, comme définir une fourchette de 0 à 220 pour les battements cardiaques, est une idée astucieuse pour contrôler la qualité et la validité des données. Cette fonctionnalité pourrait permettre d'identifier rapidement les mesures aberrantes ou incohérentes, améliorant ainsi la qualité globale des données collectées.

En ce qui concerne le tableau de bord, plusieurs fonctionnalités clés restent à explorer. L'ajout de tests, bien que partiellement initié, nécessite une attention continue pour améliorer la couverture du code et garantir la fiabilité du tableau de bord. L'optimisation de l'arrivée des mesures en temps réel est une autre perspective importante, car elle garantit une expérience utilisateur plus fluide et en phase avec les données en temps réel.

La possibilité d'ajouter manuellement des mesures constitue un atout précieux, permettant aux utilisateurs de compléter et d'enrichir leurs ensembles de données. De plus, ajuster automatiquement la date en fonction de la localisation serait une avancée pratique pour assurer une chronologie précise des mesures, en tenant compte des fuseaux horaires et des décalages.

## Conclusion

En conclusion, mon expérience de stage s'est révélée extrêmement enrichissante à bien des égards. J'ai eu l'opportunité de contribuer au développement du MVP de l'API et du tableau de bord, ce qui m'a permis de mettre en pratique mes compétences tout en découvrant de nouvelles technologies et approches. L'exploration du FL et mes premiers pas dans le domaine du ML ont été particulièrement stimulants, d'autant plus que ces domaines sont actuellement au cœur des avancées technologiques.

Travailler au sein d'une équipe dynamique m'a non seulement permis de collaborer avec des collègues talentueux, mais m'a également initié aux mécanismes de l'agilité, contribuant ainsi à une gestion plus flexible et réactive du projet. La mise en production réelle et le travail en équipe, même lors de périodes de congé, m'ont confronté à des situations réalistes et variées, ce qui a renforcé ma compréhension du développement logiciel dans un contexte professionnel.

Mon expérience a été marquée par l'apprentissage continu grâce à l'interaction avec mes pairs. Travailler sur des papiers de recherche et assister à des conférences en anglais m'a également permis d'améliorer mes compétences linguistiques et de rester en phase avec les développements internationaux dans le domaine.

Dans l'ensemble, ce stage a été une période de croissance personnelle et professionnelle. J'ai acquis des compétences techniques, mais j'ai également développé des compétences en communication, en gestion d'équipe et en adaptation aux défis variés de l'industrie. Cette expérience m'a inspiré à poursuivre ma carrière dans un environnement en constante évolution et à contribuer activement à des projets innovants et stimulants.

## Annexes

### Glossaire

1. **RAMi** : Real-Time Architecture for the Monitoring of Elderly Patients through the Internet of Medical Things - Architecture en temps réel pour le suivi des patients âgés grâce à l'Internet des Objets Médicaux.
2. **IA/AI** : Intelligence artificielle - Artificial Intelligence - est un domaine de l'informatique qui vise à créer des systèmes capables d'imiter certaines fonctions cognitives humaines, telles que l'apprentissage, la résolution de problèmes et la prise de décisions. Les technologies d'IA incluent des algorithmes, des réseaux neuronaux et des modèles statistiques qui permettent aux machines d'apprendre à partir de données et de s'améliorer avec l'expérience.
3. **Blockchain** : est une technologie de registre numérique sécurisé et décentralisé où les transactions sont enregistrées de manière transparente et immuable. Chaque bloc contient des données, et une fois ajouté à la chaîne, il ne peut plus être modifié. Cette technologie garantit l'intégrité des données, offre une vérification transparente et évite la nécessité d'une autorité centrale de confiance, ce qui la rend applicable à divers domaines tels que les transactions financières, la gestion de la chaîne d'approvisionnement et la sécurisation des données médicales.
4. **Open source** : désigne des logiciels dont le code source est rendu accessible au public, permettant à quiconque de voir, modifier et distribuer le code. Cela encourage la collaboration et la transparence dans le développement logiciel, favorisant la création de produits innovants et le partage de connaissances.
5. **ILIA** : Informatique Logiciel et Intelligence Artificielle
6. **IoMT** : Internet of Medical Thing - Internet des Objets Médicaux
7. **Apprentissage fédéré / Federated Learning / FL** : est une technique d'apprentissage automatique qui entraîne un algorithme via plusieurs sessions indépendantes, chacune utilisant son propre ensemble de données. Contrairement aux techniques traditionnelles d'apprentissage automatique centralisées où les ensembles de données locaux sont fusionnés en une seule session d'entraînement, et aux approches supposant que les échantillons de données locaux sont distribués de manière identique..
8. **API** : application programming interface - interface de programmation d'application - est un ensemble de règles et de protocoles qui permettent à différents logiciels de communiquer et d'interagir entre eux.
9. **Framework** : Cadre de travail - est une structure préétablie qui fournit des bases et des composants pour le développement d'applications logicielles. Il offre un ensemble de règles, de conventions et de bibliothèques prédéfinies pour faciliter la création de logiciels en automatisant certaines tâches récurrentes.



10. **Containerisation** : est une technique de virtualisation légère dans laquelle une application, ainsi que ses dépendances et son environnement d'exécution, sont encapsulées dans un conteneur isolé. Les conteneurs permettent de créer des environnements autonomes et cohérents, indépendants du système d'exploitation sous-jacent, ce qui facilite le déploiement et la gestion d'applications dans des environnements variés. Chaque conteneur contient tous les éléments nécessaires, tels que les bibliothèques, les configurations et le code, pour exécuter l'application de manière cohérente.
11. **Microservice** : est un style architectural de développement logiciel dans lequel une application est construite comme une suite de services autonomes et distincts. Chaque service, appelé microservice, se concentre sur une fonctionnalité spécifique et communique avec les autres services via des interfaces bien définies. Contrairement aux architectures monolithiques, les microservices favorisent la modularité et la scalabilité en permettant à chaque service d'être développé, déployé et mis à l'échelle indépendamment.
12. **Docker** : est une plateforme de virtualisation légère basée sur la technologie des conteneurs.
13. **Déployer** : fait référence au processus de mise en place et d'exécution d'une application, d'un logiciel ou d'un système sur un environnement opérationnel, prêt à être utilisé par les utilisateurs finaux. Cela implique la configuration, l'installation et la mise en marche de l'application sur les serveurs ou les dispositifs appropriés.
14. **Scalabilité** : se réfère à la capacité d'un système, d'une application ou d'une infrastructure à s'adapter efficacement à des charges de travail variables ou en croissance. Un système scalable peut maintenir des performances élevées et une réactivité satisfaisante à mesure que le nombre d'utilisateurs, de demandes ou de données augmente.
15. **Big Data** : désigne de vastes volumes de données complexes et variées qui dépassent les capacités de traitement des méthodes traditionnelles. Ces données sont caractérisées par les trois V : volume (grande quantité), variété (diversité de formats et de sources) et vélocité (vitesse à laquelle elles sont générées).
16. **Cloud Computing** : informatique en nuage - est un modèle de prestation de services informatiques via Internet. Plutôt que de gérer des ressources informatiques locales, les utilisateurs accèdent à des ressources telles que le stockage, la puissance de calcul et les applications à la demande, via des fournisseurs de services cloud.
17. **Machine Learning / ML** : Apprentissage Automatique - est une branche de l'intelligence artificielle qui se concentre sur le développement d'algorithmes et de modèles informatiques capables d'apprendre à partir de données et d'améliorer leurs performances avec l'expérience. Plutôt que de programmer explicitement des règles, le ML permet aux machines de détecter des motifs, de faire des prédictions et de prendre des décisions basées sur les données d'entraînement.

18. **Deep Learning / DL** : Apprentissage Profond - est une sous-branche du ML qui se concentre sur l'utilisation de réseaux de neurones artificiels pour modéliser et résoudre des problèmes complexes. Inspiré par la structure du cerveau humain, le DL utilise des architectures de réseaux de neurones profonds avec plusieurs couches pour apprendre des caractéristiques hiérarchiques à partir des données. Cette approche permet aux modèles d'apprendre automatiquement à partir de grandes quantités de données brutes, en découvrant des motifs et des représentations abstraites.
19. **IoT** : Internet des Objets - représente un réseau interconnecté d'objets physiques et d'appareils intégrés avec des capteurs, des logiciels et des technologies de communication, leur permettant de collecter et d'échanger des données. Ces objets, qui peuvent être n'importe quoi, des voitures aux appareils ménagers en passant par les capteurs industriels, sont équipés pour communiquer et interagir avec d'autres objets via Internet.
20. **Cloud** : Nuage - est un modèle de prestation de services informatiques via Internet.
21. **Edge Computing** : informatique périphérique - est une approche de traitement des données qui consiste à effectuer des calculs et des analyses au plus près de la source des données, plutôt que dans des centres de données distants. Cette technique vise à réduire la latence et à améliorer les performances en traitant les données localement, directement sur les dispositifs ou les serveurs situés en périphérie du réseau.
22. **Modèle** : désigne une représentation informatique d'un processus ou d'un système complexe. Il est créé à partir d'algorithmes et de données d'entraînement pour apprendre à effectuer des tâches spécifiques, comme la classification, la prédiction ou la génération. Le modèle capture des motifs et des relations dans les données afin de prendre des décisions ou d'effectuer des prédictions sur de nouvelles données. Les types de modèles couramment utilisés en IA comprennent les réseaux neuronaux, les arbres de décision, les machines à vecteurs de support (SVM), les modèles de régression, etc.
23. **Méthode Agile** : est une approche de développement logiciel centrée sur la collaboration, l'itération et la flexibilité. Plutôt que de suivre un plan rigide, les équipes Agile travaillent en cycles courts appelés itérations, adaptant constamment leur approche en fonction des retours d'utilisateurs et des changements.
24. **Méthode MoSCoW** : est une technique de gestion des priorités qui classe les éléments d'un projet en quatre catégories : "Must Have" (Doit avoir), "Should Have" (Devrait avoir), "Could Have" (Pourrait avoir) et "Won't Have" (Ne devrait pas avoir). Cela permet aux parties prenantes de clarifier et de hiérarchiser les fonctionnalités et les exigences en fonction de leur importance.
25. **Tableau Kanban** : est un outil visuel de gestion de projet qui permet de suivre le flux de travail de manière transparente. Les tâches ou les éléments sont représentés par des cartes placées sur des colonnes qui représentent les différentes étapes du processus, de la "to-do" (à faire) à la "done" (terminé).

26. **MEC** : Mobile Edge Computing - Calcul Périphérique Mobile - est une extension de la technologie Edge Computing. Il consiste à déplacer la puissance de calcul et de traitement plus près des périphériques mobiles tels que les smartphones, les tablettes et les objets connectés.
27. **Contrats intelligents** : Smart contract - est un protocole informatique auto-exécutable qui facilite, vérifie et exécute automatiquement les termes d'un accord entre parties. Basé sur la technologie blockchain, un smart contract agit comme un ensemble de règles programmées qui s'activent lorsque les conditions prédéfinies sont remplies. Cela permet d'automatiser des processus commerciaux, d'éliminer le besoin d'intermédiaires et de garantir l'exécution sécurisée des accords sans dépendre de tiers de confiance.
28. **Apache Kafka** : est une plateforme de streaming distribuée, conçue pour la gestion en temps réel des flux de données massifs. Elle permet la collecte, la transmission et le traitement des données entre différentes applications et systèmes. Kafka est utilisé pour la mise en œuvre de pipelines de données, l'intégration d'applications et la création de flux de données évolutifs et fiables.
29. **Message Splitting and Replication** : Scission et Réplication des Messages - ce processus divise les messages entrants en plusieurs parties pour une distribution et une gestion plus efficaces. La réplication garantit que les messages sont dupliqués pour une redondance et une fiabilité accrues, permettant une haute disponibilité et une récupération en cas de panne.
30. **MQTT** : Message Queuing Telemetry Transport - Transport de Télémétrie par File d'Attente de Messages - est un protocole de communication léger et efficace pour la transmission de messages entre dispositifs connectés. Il est largement utilisé dans l'IoT pour l'échange de données entre capteurs, appareils et systèmes.
31. **Time Series** : Séries Temporelles - est une séquence ordonnée de données enregistrées à des intervalles réguliers. Ces données sont associées à des horodatages, ce qui permet l'analyse et la prédiction de tendances sur une période donnée.
32. **Apache Spark** : est un framework de traitement de données rapide et distribué, conçu pour le traitement de données en temps réel et en batch. Il prend en charge diverses opérations d'analyse et de transformation sur de grands ensembles de données.
33. **Batch** : désigne l'exécution d'un groupe de tâches informatiques de manière automatisée, sans nécessiter d'intervention continue de l'utilisateur. Les tâches sont exécutées en lot, c'est-à-dire ensembles, à un moment prédéfini ou lorsqu'une condition spécifique est remplie.
34. **ML Processing** : Traitement d'Apprentissage Automatique
35. **Real-time Processing** : Traitement en Temps Réel

36. **Database** : Base de Données - est un système de stockage et de gestion structuré des données. Elle permet de stocker, organiser et récupérer des informations de manière efficace.
37. **Relational Database** : Base de Données Relationnelle - stocke les données dans des tables structurées avec des relations définies entre elles.
38. **PostgreSQL** : est un système de gestion de base de données relationnelle open source, connu pour sa robustesse, sa sécurité et ses fonctionnalités avancées.
39. **Annotation service** : Service d'Annotation - permet d'ajouter des métadonnées, des étiquettes ou des commentaires aux données, facilitant leur interprétation et leur recherche ultérieure.
40. **Caching** : Mise en Cache - consiste à stocker temporairement des données en mémoire pour accélérer leur accès ultérieur. Cela améliore les performances et réduit la charge sur les systèmes de stockage.
41. **Redis** : est une base de données en mémoire open source, utilisée pour la mise en cache, la gestion de sessions et la manipulation rapide des données.
42. **Time Series Database** : Base de Données de Séries Temporelles - est conçue pour stocker et interroger efficacement des données temporelles, optimisée pour les analyses temporelles et les prédictions.
43. **Apache Druid** : est une base de données de séries temporelles open source conçue pour l'exploration et l'analyse rapide de données en temps réel.
44. **Object Transformer** : Transformateur d'Objets - est utilisé pour convertir, mapper ou modifier des objets d'un format à un autre, facilitant l'intégration et la manipulation des données.
45. **Apache Camel** : est un framework d'intégration open source qui simplifie l'intégration entre différentes applications en fournissant des composants et des connecteurs réutilisables.
46. **Dashboard** : Tableau de Bord - est une interface visuelle qui présente des données et des informations sous forme graphique et intuitive, permettant une vue d'ensemble et une analyse des données.
47. **Apache Superset** : est une plateforme d'exploration et de visualisation de données open source, offrant des outils pour créer des tableaux de bord interactifs et informatifs.

48. **Research Service** : Service de Recherche - fournit des fonctionnalités pour l'exploration et l'extraction d'informations à partir de données, favorisant la découverte de connaissances et de tendances.
49. **Elastic Search** : est un moteur de recherche et d'analyse de données open source, utilisé pour l'indexation, la recherche et l'analyse de vastes ensembles de données non structurées.
50. **Deep object storage** : Stockage d'Objets Profond - fait référence à la capacité de stocker et de gérer des objets de données massifs et complexes, souvent utilisés dans les applications d'apprentissage automatique et d'analyse de données.
51. **MinIO** : est une plateforme de stockage d'objets open source, optimisée pour les performances et la scalabilité, utilisée pour la gestion de données massives.
52. **Kubernetes** : est un système d'orchestration de conteneurs open source qui automatise le déploiement, la mise à l'échelle et la gestion d'applications conteneurisées. Il facilite la gestion des applications dans des environnements dynamiques et complexes, en optimisant la répartition des ressources et en garantissant la disponibilité et la scalabilité.
53. **Apache Log4j** : est une bibliothèque de journalisation open source qui permet aux développeurs de gérer et de contrôler la sortie des journaux de leurs applications. Elle offre une flexibilité dans la configuration des niveaux de journalisation et des destinations de sortie, facilitant la surveillance et le dépannage des applications.
54. **Apache Spark** : est un framework de traitement de données rapide et puissant qui permet l'analyse, le traitement et la manipulation de grandes quantités de données en temps réel. Il offre une suite d'outils pour le traitement parallèle et distribué, ainsi que pour les analyses de données interactives.
55. **Apache Airflow** : est une plateforme de gestion et d'ordonnancement de flux de travail open source. Elle permet de créer, planifier et surveiller des tâches complexes dans des environnements de données et d'applications distribuées, facilitant l'automatisation et l'optimisation des processus.
56. **Apache Zookeeper** : est un service de coordination distribuée qui gère les tâches telles que la synchronisation, la configuration et le verrouillage pour les applications distribuées. Il fournit un ensemble d'outils pour garantir la cohérence et la fiabilité dans les environnements distribués.
57. **Back-end** : Partie serveur - fait référence à la partie d'une application ou d'un site web qui gère les processus et les fonctionnalités en coulisses, hors de la vue des utilisateurs. Il inclut la gestion des bases de données, le traitement des données, la gestion des utilisateurs et la logique métier.

58. **Front-end** : Partie client - désigne la partie visible et interactive d'une application ou d'un site web avec laquelle les utilisateurs interagissent
59. **Backlog** : Liste des Tâches en Attente - est une liste ordonnée de tâches, de fonctionnalités ou de demandes en attente de développement ou de traitement. Il sert de référence pour les équipes de développement afin de hiérarchiser et de planifier les travaux futurs.
60. **Planned** : Planifié - signifie qu'il a été inscrit dans le plan de développement pour être réalisé à un moment spécifique.
61. **Doing** : En Cours
62. **Done** : Terminé
63. **Reviewed** : Revu - signifie qu'il a été examiné et évalué par une personne ou une équipe. Le processus de revue garantit la qualité, la cohérence et la conformité aux normes établies.
64. **MR / Merge Request** : demande de fusion - est une demande soumise par un contributeur pour fusionner son code ou ses modifications dans la branche principale d'un projet, généralement après qu'il a été examiné et approuvé. Cela permet d'intégrer les changements de manière contrôlée et collaborative.
65. **Gateway** : Passerelle - est un dispositif ou un logiciel qui agit comme un point d'entrée entre différents réseaux ou protocoles. Dans l'IoT, une passerelle IoT facilite la communication entre les dispositifs IoT et les services cloud.
66. **Apache MiNiFi** : est une solution légère conçue pour collecter et transférer des données depuis des dispositifs IoT ou des capteurs vers des systèmes plus grands, tel que Kafka. Il permet la capture et l'acheminement des données en temps réel.
67. **Apache IoTDB** : est une base de données open source spécialement conçue pour la gestion de données générées par l'IoT. Elle offre une architecture optimisée pour le stockage et la récupération de données à haute fréquence et à grande échelle.

## Remerciements

Je tiens à exprimer ma sincère gratitude envers toutes les personnes qui ont joué un rôle, directement ou indirectement, dans le succès de mon stage.

Je voudrais tout d'abord remercier chaleureusement mon tuteur de stage, Saïd MAHMOUDI, enseignant-chercheur au sein du service ILIA. Votre encadrement, vos conseils avisés et votre disponibilité ont été essentiels pour orienter et guider mon travail tout au long de ces 3 mois. Votre contribution a profondément enrichi mon expérience et a été déterminante pour la réussite de ce projet.

J'adresse également mes remerciements les plus sincères à Thomas PONT, stagiaire et étudiant en informatique en 5ème année de l'école Centrale Marseille, ainsi qu'à Timothé BRENIER, stagiaire et étudiant en électronique en 4ème année de l'école Polytech Lille. Nos collaborations fructueuses sur les projets ont apporté un nouvel éclairage à mon travail et ont été des opportunités d'apprentissage exceptionnelles.

Un grand merci s'adresse à Mathis DELEHOUZÉE, doctorant au sein du service, pour son intégration exemplaire dans l'équipe et pour son soutien inestimable dans la partie fédérée du projet. Je tiens également à exprimer ma reconnaissance envers Adriano GUTTADAURIA, l'administrateur système, pour son assistance précieuse lors du déploiement et dans divers contextes.

Enfin, je souhaite adresser mes remerciements les plus sincères à mon professeur référent, Fabien RINGEVAL de Polytech Grenoble. Votre encadrement, vos conseils pertinents et votre soutien ont été d'une importance capitale pour la réalisation réussie de ce stage.

En conclusion, mes remerciements s'étendent à tous les membres du service ILIA. Votre accueil chaleureux, votre bienveillance et votre assistance constante ont grandement contribué à rendre mon expérience de stage extrêmement positive.

## Bibliographie

1. Site d'Infortech : <https://web.umons.ac.be/infortech/fr/>
2. Site de l'UMons: <https://web.umons.ac.be/fr/>
3. Article RAMi : <https://www.mdpi.com/2078-2489/13/9/423>
4. Site du service ILIA : <https://ig.umons.ac.be/>
5. Site de Docker : <https://www.docker.com/>
6. Site de la méthode Agile: <https://www.agilealliance.org/agile101/>
7. Site de la méthode Kanban :  
<https://kanbanize.com/kanban-resources/getting-started/what-is-kanban>
8. Site d'Apache Kafka : <https://kafka.apache.org/>
9. Site de Redis : <https://redis.io/>
10. Site de Mosquitto : <https://mosquitto.org/>
11. Site de Kubernetes : <https://kubernetes.io/>
12. Site d'Apache Spark : <https://spark.apache.org/>
13. Site d'Apache Airflow : <https://airflow.apache.org/>
14. Site d'Apache ZooKeeper : <https://zookeeper.apache.org/>
15. Site de PostgreSQL : <https://www.postgresql.org/>
16. Site d'Apache MiNiFi : <https://nifi.apache.org/minifi/>
17. Site d'Apache IoTDB : <https://iotdb.apache.org/>
18. Article "UniFed: A benchmark for Federated Learning Frameworks" :  
<https://gitlab.com/stage4a/docs/-/blob/main/data/assets/research-fl-benchmark-unifed.pdf>
19. Article "FedML: A Research Library and Benchmark for Federated Learning" :  
<https://gitlab.com/stage4a/docs/-/blob/main/data/assets/research-fl-benchmark-fedml.pdf>
20. Présentation "Secure Federated Learning" :  
<https://gitlab.com/stage4a/docs/-/blob/main/data/assets/infortech-day-secure-fl.pdf>
21. Chaîne youtube de Flower : <https://www.youtube.com/@flowerlabs/featured>
22. Site de Flower : <https://flower.dev/>
23. Rapport sur le vieillissement de la société française et européenne :  
[https://www.gouvernement.fr/sites/default/files/contenu/piece-jointe/2023/02/hcp\\_vieillissem  
ent de la societe francaise.pdf](https://www.gouvernement.fr/sites/default/files/contenu/piece-jointe/2023/02/hcp_vieillissem%20ent%20de%20la%20societe%20francaise.pdf)



## DOS DU RAPPORT

Etudiant (nom et prénom) : SOLER Lilian

Année d'étude dans la spécialité : 4A

Entreprise : Université de MONS

Adresse complète : (géographique et postale) Place du Parc, 20 7000 MONS BELGIQUE

Téléphone (standard) : +32 (0)65 37 31 11

Responsable administratif (nom et fonction) : Mme BERGER Géraldine - Coordinateur institutionnel des mobilités

Téléphone : +32 (0)65 37 31 11

Courriel: [info.mons@umons.ac.be](mailto:info.mons@umons.ac.be)

Tuteur de stage (organisme d'accueil) MAHMOUDI Saïd

- Enseignant chercheur

Téléphone : +33 6 16 76 13 20

Courriel: [said.mahmoudi@umons.ac.be](mailto:said.mahmoudi@umons.ac.be)

Enseignant-référent : RINGEVAL Fabien

Téléphone : 04 57 42 16 07

Courriel: [Fabien.Ringeval@univ-grenoble-alpes.fr](mailto:Fabien.Ringeval@univ-grenoble-alpes.fr)

Titre : (maximum 2 à 3 lignes).

Création de microservices pour l'architecture RAMi

Résumé : (minimum 15 lignes).

Pendant ma quatrième année à Polytech Grenoble, j'ai effectué un stage en tant qu'ingénieur informatique à l'Université de Mons chez Infortech, un laboratoire de recherche affilié à l'Université, au sein du département ILIA. Mon stage a été dédié au projet RAMi, un projet de grande envergure reconnu par un article de recherche scientifique, qui vise à créer une architecture en temps réel pour le suivi des patients âgés via l'Internet des Objets Médicaux (IoMT).

Sous la supervision de mon tuteur, Saïd MAHMOUDI, j'ai collaboré avec une équipe de 4 autres stagiaires, dont 2 travaillent également sur le projet RAMi. Cette expérience d'équipe m'a apporté tant sur le plan professionnel que personnel. Le projet RAMi répond à des enjeux sociétaux majeurs tels que le vieillissement de la population et les besoins en suivi à domicile, en particulier accentués par la pandémie de COVID-19.

Au départ, mon rôle se concentrait sur l'apprentissage fédéré, un moyen de former des modèles d'IA en respectant la confidentialité des données des patients. Toutefois, mon rôle s'est étendu pour inclure la création d'une API et d'un tableau de bord pour gérer les mesures de capteurs en tout genre, en collaboration avec un autre stagiaire. Cette API enregistre et rapporte les données provenant de divers capteurs. Nous travaillons également sur un tableau de bord intuitif pour la visualisation et l'analyse des données collectées.

Mon stage m'a offert une immersion concrète dans le développement d'une architecture de suivi médical avancée, tout en m'apprenant l'importance de la collaboration, de la planification et de l'adaptation dans un contexte de projet ambitieux.