Rapport final Mycelium 3.0

CHEREL Valentin - DERRIEN Thomas
MAUGER Arthur - OUVRARD Pierre - POURCHER Pierrick

Encadrants : PARLAVANTZAS Nikos - PAROL-GUARINO Volodia

Avec la participation de :
MOUREAU Julien

2023-2024

FIGURE 1 – Mycélium : projet de suivi environnemental

1 Remerciements

Nous tenons tout d'abord à remercier M. Volodia PAROL-GUARINO et M. Nikolaos PARLAVANTZAS, nos encadrants à l'INSA, pour leur suivi, leur forte implication et leurs nombreux conseils réguliers tout au long du projet.

Nous souhaitons également remercier notre encadrant extérieur, M. Julien MOUREAU, travaillant à l'OSUR pour sa présence régulière aux réunions et pour son expertise sur les différentes technologies.

Enfin, nous remercions tous les membres de l'équipe de Mycélium pour l'intérêt qu'ils ont porté au bon déroulement du projet. Merci aux membres de Mycélium 3.0 - Leo LESSIRARD et Maïwenn LE GOASTELLER - partis au second semestre étudier à l'étranger, pour leur investissement et leur présence. Merci enfin aux membres de Mycélium 2.0 qui ont accepté une réunion afin de nous éclairer un peu plus sur le projet.

Table des matières

2 Glossaire

— **API** : *Application Programming Interface*, ensemble de règles permettant à des logiciels de communiquer entre eux de manière standardisée, par exemple pour interagir avec les fonctionnalités ou les données d'une application, service ou système externe.

— **Broker MQTT** : Serveur intermédiaire qui facilite la communication asynchrone entre les dispositifs IoT en gérant les opérations de *publish* (publication) et *subscribe* (abonnement) pour l'échange de messages.

— **ChirpStack [1]** : Logiciel open-source pour la gestion des appareils utilisant LoRaWAN.

— **FaaS** : *Function as a service*, modèle de déploiement de cloud computing en microservices serverless.

— **Flux RSS** : Flux *Really Simple Syndication*, format de données XML permettant de distribuer des mises à jour de contenu de manière structurée.

— **Fog computing** : Modèle informatique décentralisé qui rapproche le traitement des données vers la périphérie du réseau, où elles sont générées, plutôt que de les centraliser dans des centres de données distants, afin de réduire leur quantité et d'augmenter la réactivité du système suivant les informations traitées.

— **Grafana [2]** : Plateforme open source de visualisation et d'analyse de données, utilisée pour créer des tableaux de bord interactifs et informatifs.

— **InfluxDB [3]** : Base de données de séries temporelles open source conçue pour stocker, interroger et visualiser des données chronologiques.

— **IoT** : Internet des objets, décrit le réseau d'objets physiques constitués de capteurs, de logiciels et d'autres technologies dans le but de connecter et d'échanger des données avec d'autres appareils et systèmes sur Internet.

— **Kubernetes (K8S) [4]** : Plateforme open source destinée à automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Kubernetes facilite la gestion d'applications distribuées et microservices sur un cluster de machines.

— **K3S [5]** : Distribution légère de Kubernetes conçue pour les environnements avec des ressources limitées, tels que les micro-ordinateurs, les dispositifs IoT ou les clusters de développement.

— **LoRaWAN [6]** : Protocole de communication basse consommation, longue portée et faible bande passante très utilisé pour l'IoT. Il se base sur le réseau LoRa (*Long range*).

— **Microservice** : Architecture logicielle basée sur le développement et le déploiement de petits services autonomes, indépendants et spécialisés, qui interagissent entre eux pour former une application complète.

— **MQTT [7]** : *Message Queuing Telemetry Transport*, protocole de messagerie léger *publish-subscribe* basé sur le protocole TCP/IP.

— **OpenFaaS [8]** : Plateforme open-source qui facilite le déploiement et la gestion de fonctions serverless, permettant ainsi l'exécution de code de manière événementielle sans se soucier de l'infrastructure sous-jacente.

— **Serverless** : Modèle d'exécution pour les applications sans gestion de serveur par l'utilisateur, avec des coûts payés selon l'utilisation.

— **VPS** : *Virtual Private Server*, machine virtuelle, créée sur un serveur physique et employant ses ressources pour offrir à ses utilisateurs les mêmes fonctionnalités qu'un serveur dédié.

3 Introduction

Le projet Mycélium, initié en 2021 en partenariat avec le laboratoire des Géosciences de Rennes, et sous la supervision de Laurent LONGUEVERGNE, a pour but de suivre et d'évaluer la renaturation du parc de la Croix Verte sur le campus de Rennes 1. Cette initiative intervient en réponse aux impacts environnementaux engendrés par les différents chantiers initiés par la ville.

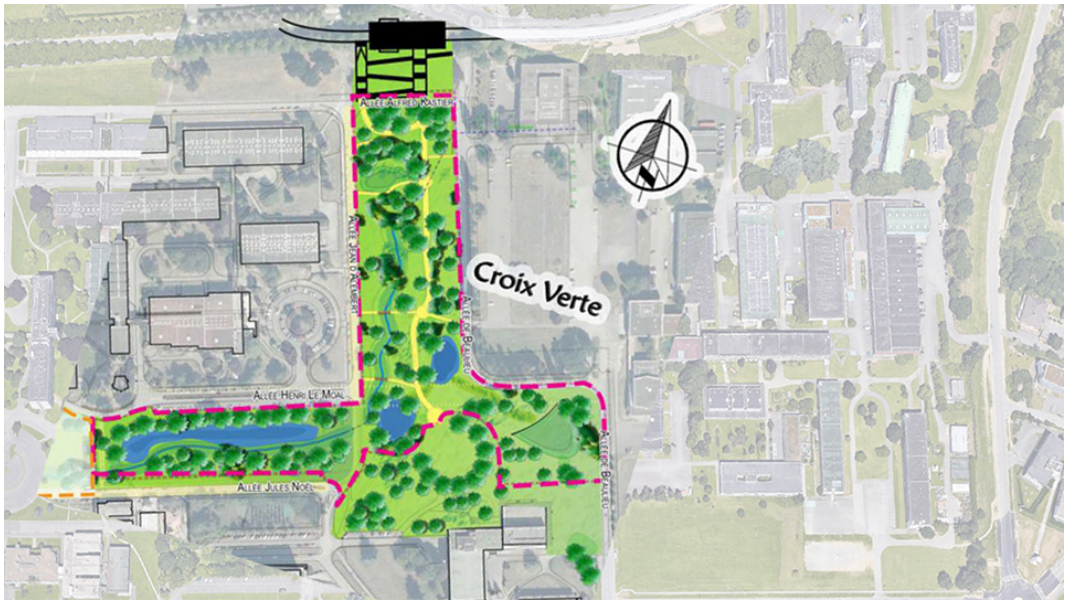

Figure 2 – Parc de La Croix Verte.

Concrètement, des capteurs économes en énergie sont disposés dans différents parcs, afin de mesurer différents indicateurs météorologiques, tels que l'humidité et la température. Il y a aussi des capteurs utilisés pour analyser l'état des rivières comme le niveau de l'eau notamment. Au cours de son évolution entre 2021 et 2023, le projet Mycélium a connu deux versions, mettant en place une architecture basée sur le fog computing pour la collecte et le traitement des données. La version actuelle, Mycélium 3.0, vise à simplifier l'architecture pour améliorer les performances et réduire la consommation énergétique, tout en facilitant son utilisation. Ce rapport a pour but de dresser un bilan sur tout ce qui a été réalisé au cours de l'année, d'aborder les choix que nous avons fait ainsi que de faire un retour sur notre expérience. Nous allons d'abord lister les rectificatifs réalisés par rapport aux objectifs des rapports de conception et de spécification.

4 Rectificatifs

Tout d'abord, nous avons décidé de changer l'architecture des guides que nous souhaitions réaliser. Ceux-ci ont pour but de faciliter la transmission et l'assimilation du projet en donnant des indications pas-à-pas concrètes sur l'installation et l'utilisation des différents éléments composant l'architecture globale du projet. Nous voulions faire en sorte que chaque composant du projet ait le droit à un guide d'installation et un guide d'utilisation, les guides devaient être séparés dans des dossiers différents en fonction du composant auxquels ils correspondant, recréant ainsi de manière virtuelle l'architecture du projet. Cependant, nous nous sommes rendu compte que cela créait de nombreux documents avec peu de contenus voir aucun, le tout réparti dans de nombreux dossiers, ce qui complexifie la recherche d'information. Nous avons donc opté pour une solution plus simple : tous les différents guides sont réunis dans un même document, avec un grand sommaire cliquable au début afin de pouvoir accéder en un clic à n'importe quelle partie du projet qui nous intéresse. Afin de garder l'idée de recréer un plan virtuel de l'architecture du projet nous avons également inséré au début de ce guide un plan afin de pouvoir repérer d'un coup d'œil les différents composants, leur rôle et articulation entre eux.

Nous souhaitions aussi arrêter d'utiliser Python pour nos fonctions OpenFaas, et privilégier le langage Go afin de gagner en espace, en fiabilité et en utilisation des resources. Cependant, étant donné qu'en cas de besoin nous pouvons directement transférer ces données à l'OSUR et que nos contacts ont des compétences en Python, nous avons décidé de continuer à utiliser Python pour les fonctions susceptibles d'être reprises par notre partenaire. C'est le cas pour une fonction permettant de changer la fréquence d'échantillonnage des capteur par exemple.

Dans le rapport de spécification, nous avions prévu de pouvoir identifier et distinguer les différents capteurs à l'aide de leur position GPS. Cependant, nous nous sommes rendu compte que les capteurs n'avaient pas de balise GPS. Nous avons donc décidé d'abandonner cet ajout qui n'était pas primordial.

Au niveau des fonctions, nous n'avons pas réalisé tous les scénarios que nous avions prévus à l'origine. L'infrastructure globale n'étant pas terminée, nous avons décidé de nous concentrer sur les tâches les plus importantes. Cependant, les fonctions que nous avons créées possèdent toutes les fonctionnalités de base nécessaires au fonctionnement de tous les scénarios, tels que le décodage des données des capteurs, l'analyse des données, l'envoi de notifications et l'affichage de celles-ci, ainsi que le stockage de ces données dans InfluxDB. Nous avons créé un nouveau scénario, prenant en compte tous ces éléments, qui a pour but de connaître la température ressentie en fonction de la température réelle et de l'humidité. Ce scénario sera explicité plus en détail dans la partie portant sur l'état final du projet.

Dans les rapports précédents, nous avons émis l'idée de faire des statistiques sur les données en utilisant Jupyter. Or, nous ne disposons pas d'un grand jeu de données, nécessaire à la réalisation de ces statistiques, car le système n'a fonctionné que très tard dans le projet. L'intérêt était avant tout de pouvoir faire une démonstration plus visuelle du projet mais cela n'était pas essentiel par rapport aux besoins de l'OSUR.

Nous avions aussi pensé à mettre en place la configuration automatique des capteurs, mais nous avons finalement dû abandonner cette idée, par manque de temps pour une tâche aussi complexe et peu précise.

Nous avons globalement eu un manque de temps au cours de ce projet, qui a compromis le développement de toutes les fonctionnalités que l'on voulait mettre en place. Cela s'explique par de multiples facteurs, comme nous allons le voir dans la partie suivante, qui dresse un bilan sur la planification du projet.

5 Bilan de planification

Il y a de nombreuses difficultés au niveau de ce projet, qui ont pour effet que toute avancée est alors complexe et prend beaucoup de temps. La recherche de documentation sur les technologies que nous utilisons est compliquée, car certaines sont des technologies à la pointe de ce qui se fait dans le domaine, et que ce n'est donc pas encore stabilisé, ni extrêmement utilisé par le grand public. Ainsi, en cas de problème, il est difficile de trouver de l'aide ou un bon guide pour résoudre ce qui ne fonctionne pas. Un problème pouvant être corrigé en une minute si on dispose des bonnes connaissances peut alors faire perdre plusieurs heures.

Par exemple, pour faire le lien entre les différentes fonctions, nous utilisons MQTT connector, qui permet de mettre en place un système Publish/Subscribe. C'est un mécanisme de publication et d'abonnement de message dans lequel les diffuseurs envoient des messages qui n'ont à priori pas de destinataire, mais qui sont associés à des topics (sujets). Les destinataires s'abonnent aux topics et reçoivent les messages les concernant indépendamment des diffuseurs. On peut voir un exemple de ce mécanisme dans le cadre de mesure de température à la figure 3.

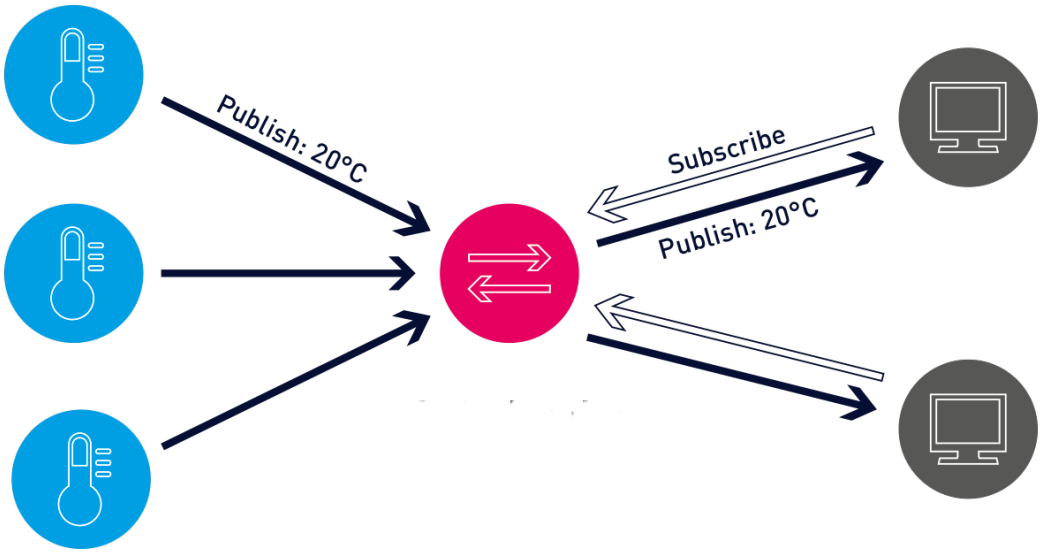

FIGURE 3 – Schéma du principe du système Publish/Subscribe

MQTT connector permet de gérer ce mécanisme entre les différentes fonctions OpenFaas simplement. Nous avons eu du mal à le mettre en place, car nous disposions seulement du dépôt Git des développeurs d'OpenFaas, avec une documentation succincte. Nous avons alors dû chercher par nous-mêmes comment le faire fonctionner sur notre projet, avec parfois des subtilités difficiles à repérer.

Le problème qui nous a fait perdre le plus de temps concerne la communication avec les DSIs de l'INSA et de l'OSUR. Nous voulons recevoir les données provenant de l'OSUR directement via le réseau Wi-Fi de l'INSA. Pour cela, l'OSUR doit nous autoriser à les recevoir, et l'INSA doit nous autoriser à y accéder. Comme nous n'y arrivions pas, nous avons dû contacter la DSI de l'INSA afin qu'ils nous donne la permission. Notre contact à l'OSUR, M. Julien MOREAU, a aussi dû contacter la DSI de l'OSUR pour voir si le problème venait de leur côté. Ces échanges ont

pris beaucoup de temps, car il y avait à chaque fois un délai avant de recevoir une réponse. Nous pouvons retrouver un résumé du déroulé de ces échanges avec la frise chronologique en figure 4. À l'heure où nous écrivons ce rapport, il y a toujours des conversations en cours afin de régler certains problèmes.

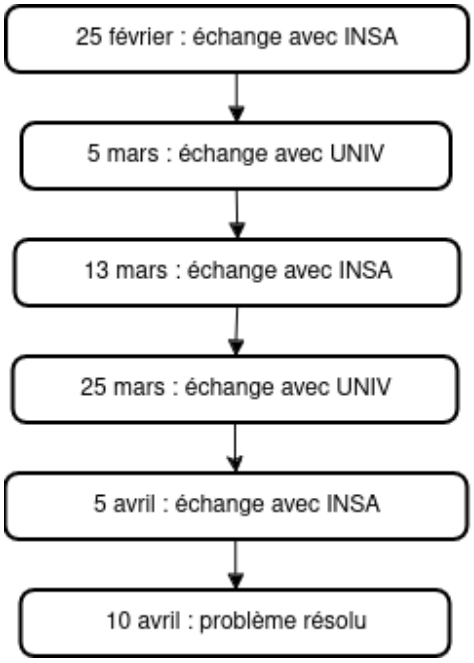

FIGURE 4 – Frise chronologique des échanges avec les DSI concernant les problèmes de communications avec l'OSUR

Une autre difficulté est liée à la mise en place du VPS. Celui-ci est contrôlé par la DSI de l'INSA, et pour y accéder, il faut utiliser un outil qui est parfois compliqué à installer. Il y a également de nombreuses règles de sécurité sur l'utilisation du VPS, imposant donc beaucoup de limitations. Parfois, certaines applications ou fonctionnalités ne fonctionnent pas à cause de ces règles, ou bien le VPS plante, et il faut alors demander l'aide de la DSI. Un problème qui est survenu était que les ports n'étaient pas ouverts sur le pare-feu du VPS. Comme pour le problème précédent, les demandes nécessaires auprès de la DSI ont ainsi engendré des délais supplémentaires. Nous avons pu rajouter dans le guide les solutions qui ont été trouvées face à ces problèmes afin d'éviter de perdre ce temps précieux à l'avenir.

De légers problèmes annexes ont aussi engendré du retard, comme par exemple la VM qui est parfois capricieuse avec OpenFaas. Nous ne savons pas gérer certains problèmes qui peuvent arriver. Par exemple, il est déjà arrivé que tous les pods OpenFaas se coupent, et qu'ils ne redémarrent plus. Dans ce genre de situations, nous devons alors recréer une VM en partant de zéro.

Tous ces différents retards dû à des problèmes imprévus ont ainsi rendu l'organisation du groupe plus complexe. En effet, si une personne ne peut pas travailler sur ce qu'elle a prévu de faire, il faut alors redistribuer les tâches.

Malgré toutes ces difficultés, nous avons quand même réussi à avancer dans le projet. Nous allons maintenant décrire tout ce que nous avons réalisé dans la partie suivante.

6 Etat de finalisation du projet

Au niveau de l'architecture globale du projet, elle a peu changé. La seule différence est que les données que l'OSUR nous envoie ne passent pas par notre broker MQTT, mais bien par le leur. Nous pensions que nous allions recevoir les données sur un broker MQTT, et que nous devrions les republier sur un topic, mais finalement, nous les récupérons directement en utilisant MQTT connector, qui est abonné à l'adresse du serveur de l'OSUR. Le schéma de l'architecture globale peut être observé à la figure 5.

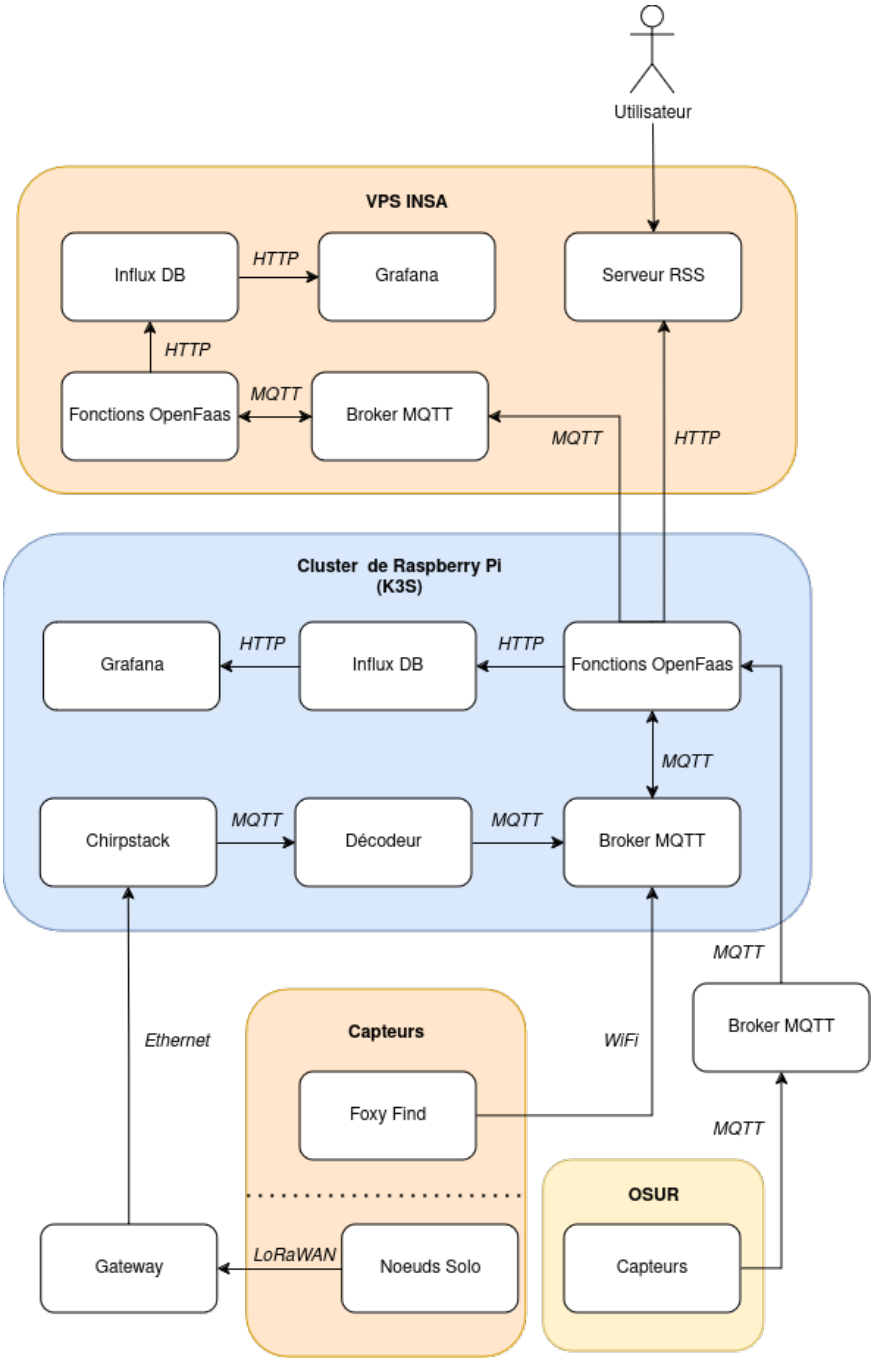

FIGURE 5 – Schéma de l'architecture finale de Mycelium3.0

Nous allons maintenant expliquer ce qui fonctionne concrètement au sein du projet. Nous disposons d'un capteur Milesight EM300-TH (voir photo à la figure 6) à l'INSA. Ce capteur permet de mesurer la température et l'humidité. Nous disposons aussi des données d'un capteur Milesight EM500 (voir photo à la figure 7) appartenant à l'OSUR. Ce capteur nous envoie de nombreuses données à intervalles réguliers, concernant notamment la température, l'humidité, la pression ainsi que le taux de CO2 dans l'air. Nous disposons également des informations concernant de nombreux capteurs de l'OSUR dans un fichier Excel. Il sera donc possible de récupérer les données de ceux-ci de la même manière qu'avec le capteur avec lequel nous avons déjà mis en place une connexion.

FIGURE 6 – Photo du capteur Milesight EM300-TH

FIGURE 7 – Photo du capteur Milesight EM500

À partir de ces données, nous effectuons plusieurs traitements. Il y a tout d'abord une fonction qui permet de calculer un indice portant sur la sensation ressentie à partir de la température et de l'humidité. Nous utilisons la formule de l'indice Humidex [9], développée en 1979 au Canada. Cet indice permet de savoir lorsque l'on arrive dans différentes zones :

— Inconfort
— Beaucoup d'inconfort
— Danger
— Coup de chaleur imminent

Cet indice est calculé à l'aide de la formule suivante où l'humidex H est donné par une fonction de la température mesurée de l'air Tair et de la témparature du point de rosée Tdew :

$$H = T_{air} + \frac{5}{9}\left[6.11 \times e^{5417.7530}\left(\frac{1}{273.16} - \frac{1}{273.15 + T_{dew}}\right) - 10\right]$$

Les différentes zones peuvent se retrouver dans le tableau de valeurs à la figure 8.

Humidité relative (%)	Température (°C)															
	21	25	30	31	32	33	34	35	36	37	38	39	40	41	42	43
20											40	41	43	44	46	47
30			31	33	34	36	37	38	40	42	43	45	47	48	50	51
40		26	34	35	37	39	40	42	44	45	47	49	51	53	54	56
50	22	28	36	38	40	41	43	45	47	49	51	53	55	57		
60	24	30	38	40	42	44	46	48	50	52	54	57				
70	25	32	41	43	45	47	49	51	53	56	58					
80	26	33	43	45	47	50	52	54	57	59						
90	28	35	45	48	50	52	55	57	60							
100	29	37	48	50	53	55	58									

FIGURE 8 – Valeurs de l'humidex selon l'humidité relative et la température réelle

Une notification est envoyée sur un topic dédié aux notifications pour indiquer l'état dans lequel on se trouve actuellement. Lorsque l'on est dans la zone de danger, un message est envoyé à une fonction permettant d'augmenter la fréquence d'échantillonnage du capteur, afin d'obtenir plus d'informations. Nous avons choisi d'effectuer ce traitement, car il prend en compte seulement la température et l'humidité. Or, au moment où nous avons mis en place ce scénario, nous disposions seulement du capteur à l'INSA, qui permet de mesurer uniquement ces deux paramètres.

Une autre fonction qui a été développée a pour but de savoir si nous recevons des valeurs erratiques depuis nos capteurs. Les données mesurées sont comparées avec les données de Météo France obtenues via une API [10]. Nous utilisons les données récoltées en temps réel sur une station météo de Rennes. Lorsque les valeurs sont trop différentes des valeurs que l'on a obtenues avec nos capteurs, l'envoi d'une notification est déclenché. Cela permet ainsi de détecter si un capteur est complètement défectueux.

Les différentes notifications envoyées par les fonctions sont traitées par une fonction spéciale. Les différentes alertes sont retransmises sur un flux RSS sur un serveur codé en Python. Nous pouvons alors accéder aux notifications depuis un navigateur web.

Parfois, les capteurs enregistrent des informations intéressantes, et nous aimerions pouvoir effectuer plus de mesures sur le capteur afin de bien étudier ce qu'il se passe. C'est par exemple le cas si des températures extrêmes sont atteintes. Dans ce cas, une fonction permet d'augmenter la fréquence d'échantillonnage du capteur. Malheureusement, cette fonctionnalité n'est compatible qu'avec le capteur dont nous disposons à l'INSA, les capteurs de l'OSUR ne permettant pas de faire cela. En effet, pour pouvoir modifier cela, il faudrait accéder à l'application Chirpstack du côté de l'OSUR, ce qui n'est pas possible. De plus, ces capteurs ne nous appartiennent pas donc on ne peut pas modifier leur comportement.

Enfin, une dernière fonction permet de stocker les données collectées dans une base de données InfluxDB. Ces données peuvent être observées en utilisant l'outil

Grafana. Pour s'assurer que tout ce qu'on a implémenté fonctionne, nous avons fait des tests, comme nous allons le présenter dans la partie qui suit.

7 Compte-rendu des phases de test

À chaque fois que nous avons ajouté une nouvelle fonction sur la VM, nous l'avons testée localement. Nous pouvions regarder dans les logs si tout se déroulait comme prévu. Nous pouvons ainsi voir si les notifications sont bien envoyées, et si les données sont bien traitées. Pour tester, nous utilisions le capteur de l'INSA, car le capteur de l'OSUR n'était disponible que très tard. Une fois que tout fonctionnait, nous avons pu réaliser des tests de fonctionnement concret sur l'ensemble des éléments du projet.

Au niveau du guide réalisé au fur et à mesure du projet, nous nous sommes assurés que tout ce qui est écrit permet bien de faire fonctionner les composants du projet. Pour cela, les membres du groupe n'ayant pas participé à l'élaboration d'une partie du guide utilisent le guide afin de voir si cela donne le résultat attendu, et si celui-ci est logique et facile à comprendre. Le guide a ainsi subi une amélioration constante tout au long du projet. La documentation actuelle a évolué positivement par rapport à l'année dernière. En termes d'organisation, toutes les explications qui étaient disséminées à plusieurs endroits ont été réunies de façon claire et précise. Les guides précédents étaient aussi beaucoup moins précis, avec seulement les commandes strictement nécessaires. Nous avons essayé de fournir tous les outils et tout ce que nous avons appris de façon claire et précise. Enfin, en termes de quantités d'informations, il y a beaucoup plus de contenus que dans les indications présentes dans le projet de l'année précédente, sans compter le fait que l'évolution du projet les a en grande partie rendu obsolète.

Malgré tous les avancements sur le projet que nous avons pu tester, il reste encore beaucoup de travail à effectuer pour aller plus loin.

8 Tâches restantes

De nombreuses fonctionnalités restent à ajouter au projet. Tout d'abord, l'interface utilisateur peut être améliorée, notamment au niveau de la visualisation des données ou de l'affichage des alertes sur le flux RSS. Nous ne nous sommes pas focalisés sur ces aspects, car nous estimons que cela n'était pas le plus urgent. Grafana pourrait aussi être utilisé pour afficher des mesures sur l'état actuel du cluster, au niveau de l'usage des ressources.

Enfin, de nombreuses fonctions peuvent être implémentées, reprenant tous les scénarios évoqués lors des rapports précédents. De nouveaux scénarios pourront aussi être créés, avec les nouvelles données fournies par l'OSUR. En effet, l'OSUR dispose de nombreux capteurs dont nous n'avons pas encore l'accès et de nouveaux capteurs disposant de mesures complètement différentes seront rajoutés au fil du temps. Un exemple de scénario qui pourra être rajouté est d'analyser la corrélation entre la quantité de pluie tombée avec la hauteur d'un cours d'eau et la hauteur d'eau dans les nappes phréatiques. Les scénarios des années précédentes peuvent aussi être réimplémentés. Il faudra peut-être cependant revoir certains scénarios qui sont loins de décrire la réalité, comme le scénario chargé de détecter la neige qui est actuellement assez peu réaliste (détection de la neige lorsqu'il fait sombre et moins de 10°C).

9 Conclusion

Pour conclure, nous pouvons affirmer que ce projet est ambitieux, car il se positionne à la pointe de l'innovation, en utilisant des technologies telles que OpenFaas et Kubernetes. Notre implication active dans le projet nous a permis de mettre en place des solutions innovantes, malgré les nombreux défis rencontrés. Bien que nous n'ayons pas pu réaliser tous nos objectifs en raison des nombreuses difficultés rencontrées, nous avons trouvé le processus très enrichissant. Ce projet a été l'occasion de nous former sur des technologies qui seront surement de plus en plus utilisées, avec une architecture de Fog Computing qui est de plus en plus utilisée. En laissant derrière nous nos connaissances et notre expérience, nous espérons que ceux qui reprendront peut-être le projet après, nous pourrons aller plus vite et plus loin dans la réalisation de ce projet, en exploitant pleinement les opportunités offertes par ces technologies innovantes.

Table des figures

10 Bibliographie

Références

[1] Site de Chirpstack. https://www.chirpstack.io/

[2] Site de Grafana. https://grafana.com/

[3] Site d'InfluxDB. https://www.influxdata.com/

[4] Site de Kubernetes. https://kubernetes.io/fr/

[5] Site de K3S. https://k3s.io/

[6] L. Vangelista, "Frequency Shift Chirp Modulation : The LoRa Modulation," in IEEE Signal Processing Letters, vol. 24, no. 12, pp. 1818-1821, Dec. 2017, doi : 10.1109/LSP.2017.2762960.

[7] Site de MQTT. https://mqtt.org/

[8] Site d'OpenFaaS. https://www.openfaas.com/

[9] Masterson, J. et Richardson, F. A., Humidex, A Method of Quantifying Human Discomfort Due to Excessive Heat and Humidity, Downsview, Ontario, Environnement Canada, 1979, 45 p.

[10] Site de l'API de météo France. https://portail-api.meteofrance.fr/web/fr/api/DonneesPubliquesObservation

11 Annexes

Lien du dépôt GitLab contenant le code du projet : `https://gitlab.insa-rennes.fr/mycelium-3.0/mycelium-3.0`

Lien vers le guide : `https://docs.google.com/document/d/1zeqyLGFVsIeQFenrWVIXsBdJN7fD8fSSPS8Tly6i5KA/edit?usp=sharing`