Étudiants chargés du projet:
VIGNEAU Noah
SHEPHO Raad
MATTI YOUSIF Youssif
DUFAU Esteban

Lycée Saint-Cricq
4bis Av. des États Unis,
64000 Pau

RAPPORT DE PROJET

BTS CIEL Option Informatique et Réseau

Projet : Monitorack

Mise en place un moyen centralisé de surveillance d'une baie

30 Av. des Frères Lumière, 64140 Lons

SESSION 2025

SOMMAIRE

1 Présentation de projet

1.1. Présentation de l'entreprise

CI Solutions est une entreprise située à Lons (64). Elle intervient dans plusieurs domaines de l'informatique (téléphonie IP, Sécurité ,réseau, …).

Un de ses métiers est le déploiement de serveurs dans des baies. Elle héberge ou déploie plusieurs gammes de serveurs sur lesquels différents systèmes sont virtualisés.

Pour des raisons de sécurité, les baies informatiques dans lesquelles les serveurs sont rackés sont équipées d'onduleurs afin d'assurer une continuité de service.

1.2. Analyse de l'existant :

Les onduleurs montés en rack, tels que l'APC Smart UPS 3000VA, sont des modèles de type « Line Interactive ».

Ce type d'onduleur permet de corriger une baisse temporaire de tension en utilisant l'énergie stockée dans la batterie.

En cas de coupure totale du courant, il passe automatiquement sur la batterie.

Cependant, par défaut, cet onduleur n'est pas conçu pour être géré à distance. De plus, il n'a pas pour fonction d'envoyer une alerte à un administrateur réseau en cas de coupure.

1.3. OBJECTIFS DU PROJET

L'**objectif** de ce projet est de mettre en place un **moyen centralisé** de **surveillance** d'une **baie** non équipée de ce type de services par défaut. Les principales fonctionnalités seront :

- Acquisition et stockage des grandeurs caractéristiques de l'onduleur à intervalles réguliers
- Suivi de l'état de l'onduleur et alerte en cas de coupure de courant
- Surveillance des intrusions (ouverture de la baie)
- Surveillance de la température et de l'humidité dans la baie

Pour ce faire, une **carte** d'**extension** de type **AP9631** sera insérée dans l'**onduleur**. Cette carte permet d'obtenir les **valeurs** des **grandeurs** caractéristiques de l'**onduleu**r via le **réseau** en utilisant soit :

- **SSH** ou **TELNET** et des commandes en ligne
- un serveur **HTTP embarqué** consultable uniquement « en local »
- le **protocole TCP Modbus**

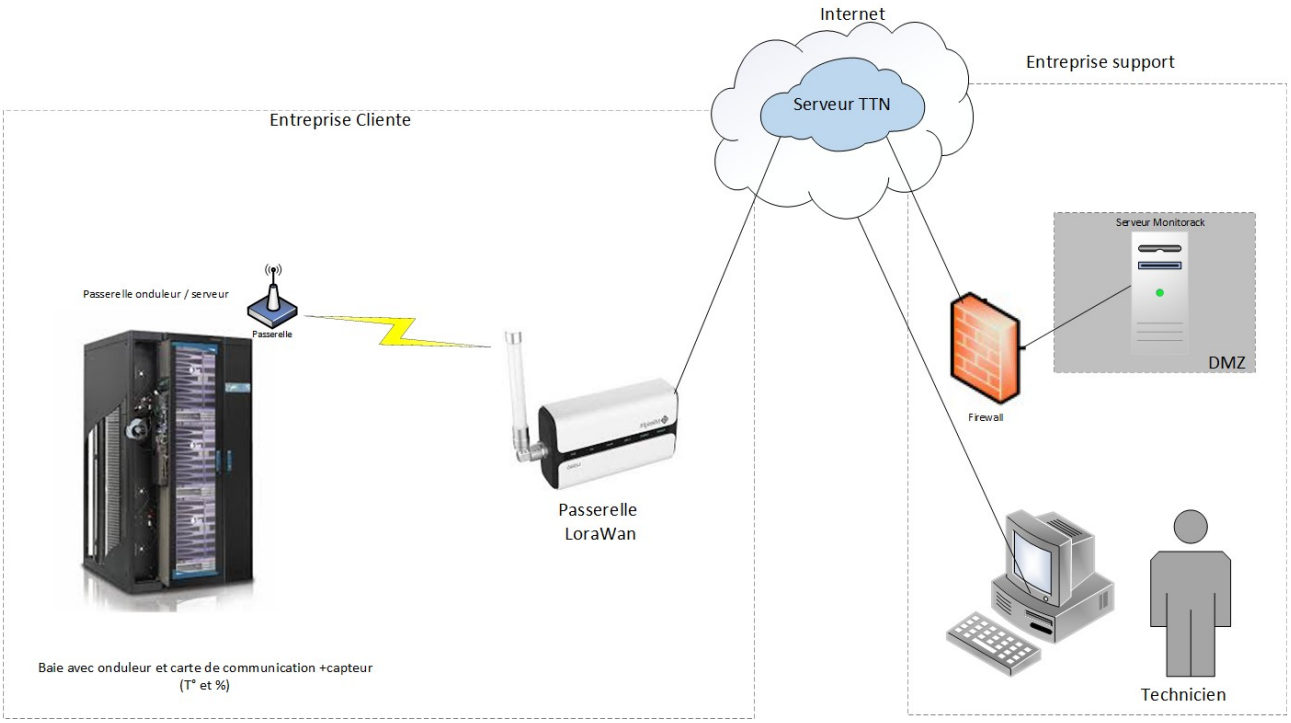
Des capteurs de **température**, **d'humidité** et de **contact** seront également **connectés** à une **« passerelle onduleur** ».
Cette « **passerelle onduleur** » permettra d'interroger à **intervalles réguliers** la **carte AP9631** pour obtenir les **valeurs** des **grandeurs** fournies par l'onduleur. Elle les enverra dans un second temps vers une **passerelle Lorawan** qui les transmettra à son tour au « **serveur Monitorack »** des **baies** situé sur la **DMZ** de l'entreprise **CI Solutions** (*une solution de secours via réseau 4G sera envisagée en cas de panne du réseau d'entreprise*).

Remarque :

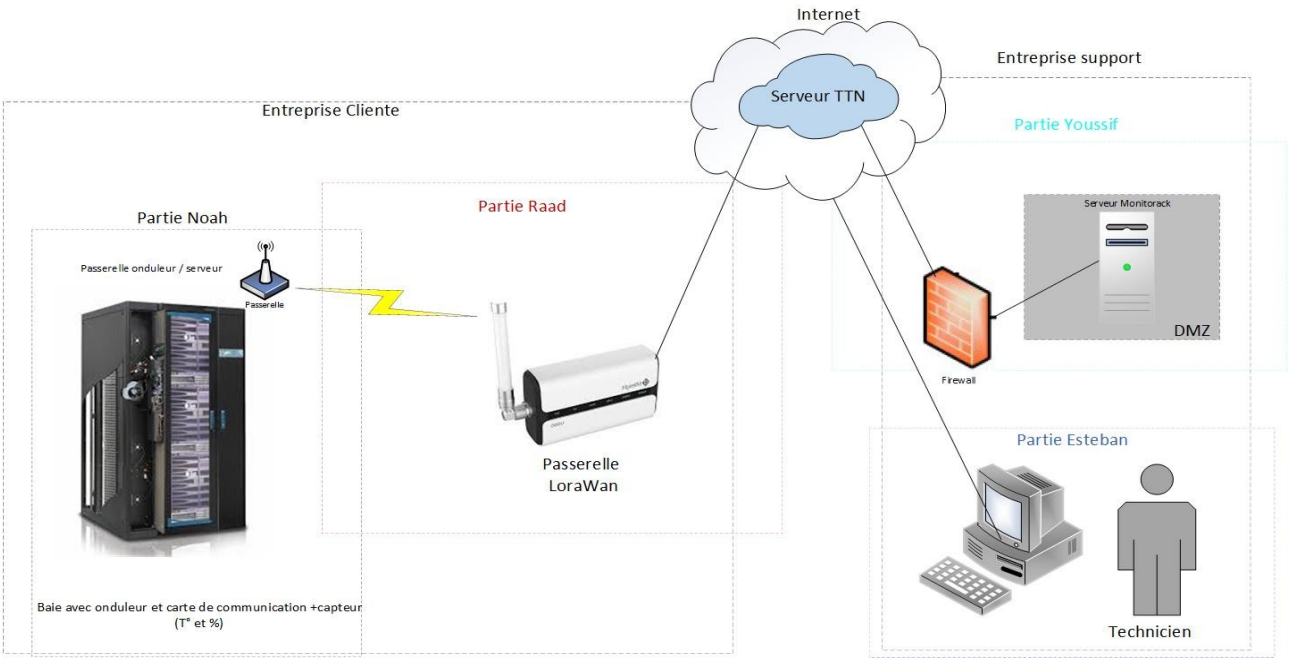
L'utilisation du réseau LoraWan est motivée par la volonté de ne pas utiliser de réseau informatique afin d'assurer une certaine continuité de service en cas de panne réseau.

La consultation des grandeurs en temps réel et des alarmes potentielles se fera via une **application de type web** depuis un poste **client** de l'entreprise **CI Solutions**. Dans la partie **visualisation** des **informations** d'une **baie**, des **critères** de **recherches** pourront être mentionnés afin d'avoir une **vision globale** des **taux** de **pannes**.

1.4. Schéma de l'infrastructure :

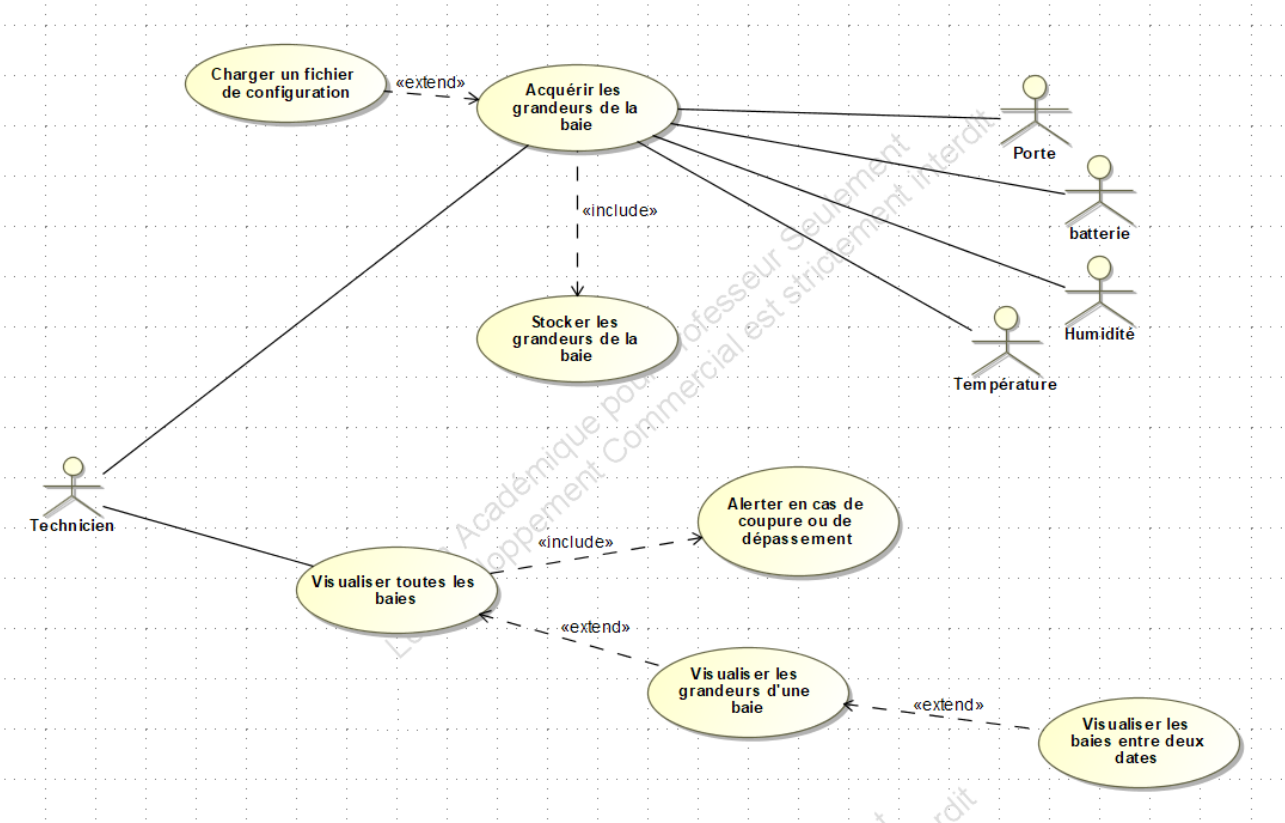


1.5. Schéma de l'infrastructure avec la répartition des tâches



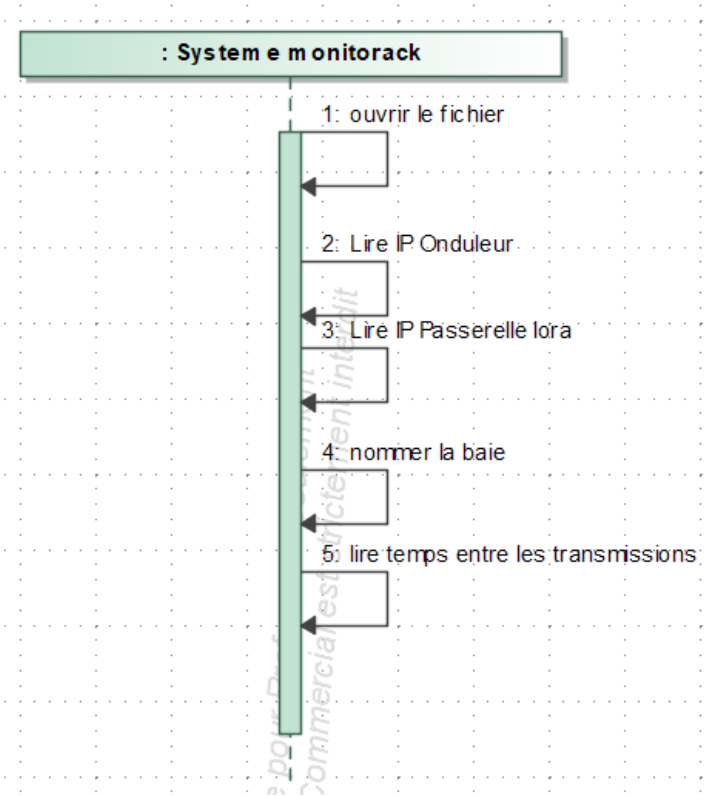
2 SPÉCIFICATION GÉNÉRALE

2.1. Diagramme use-case :

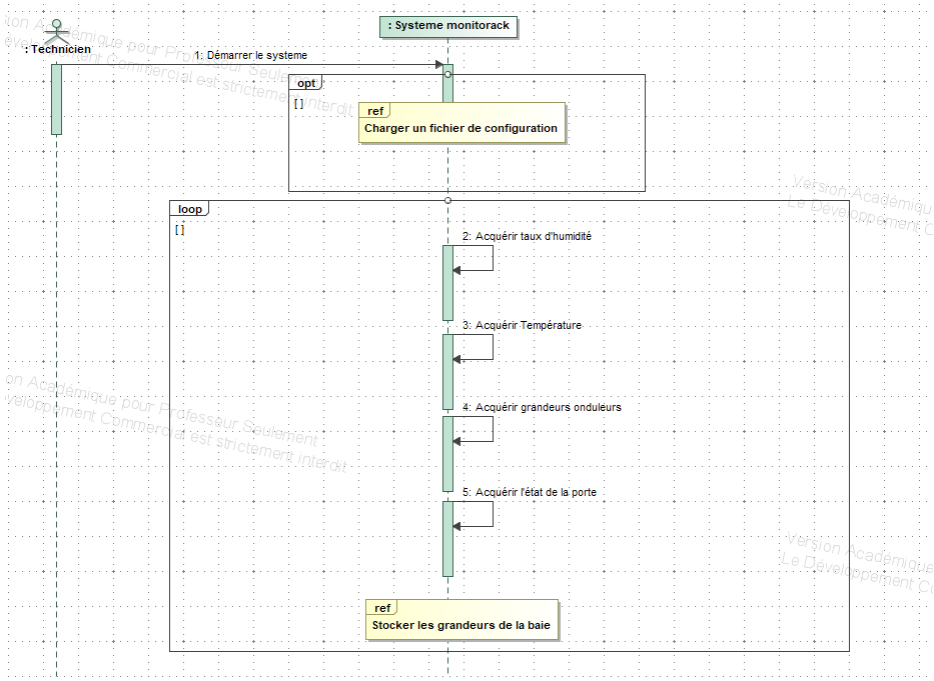


2.2. Diagrammes de séquence :

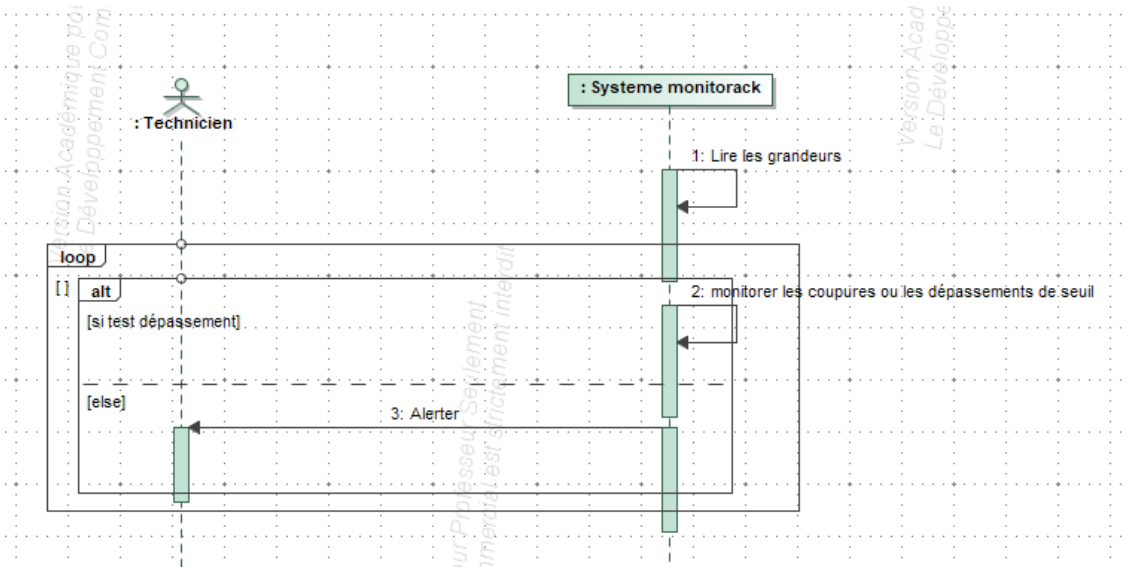
USE CASE /Charger un fichier de configuration :



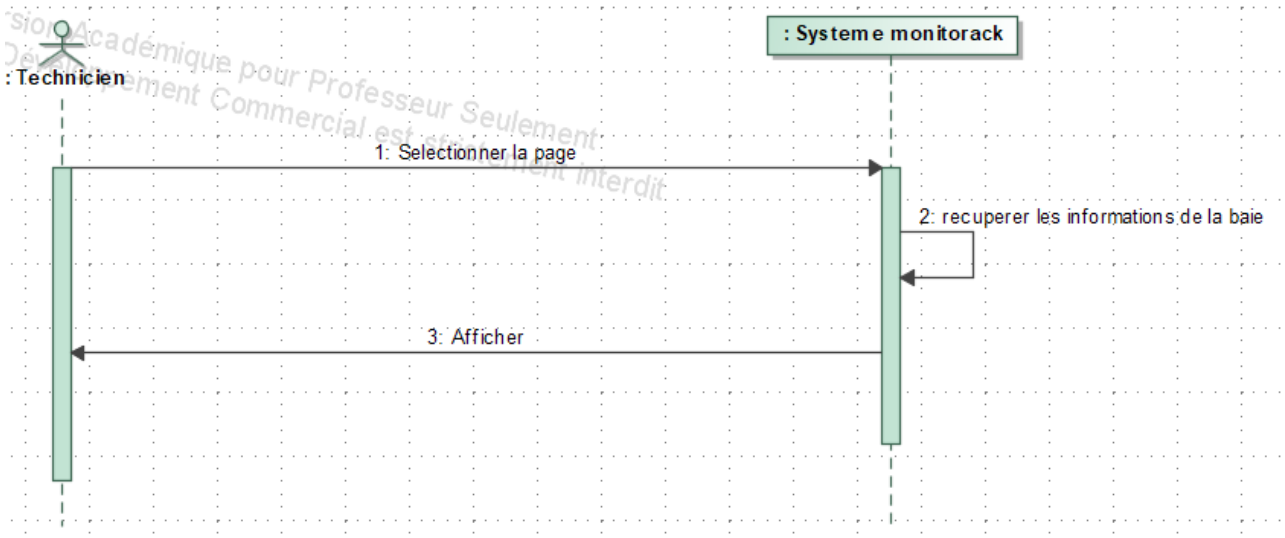
USE CASE /Acquérir les grandeurs de la baie :



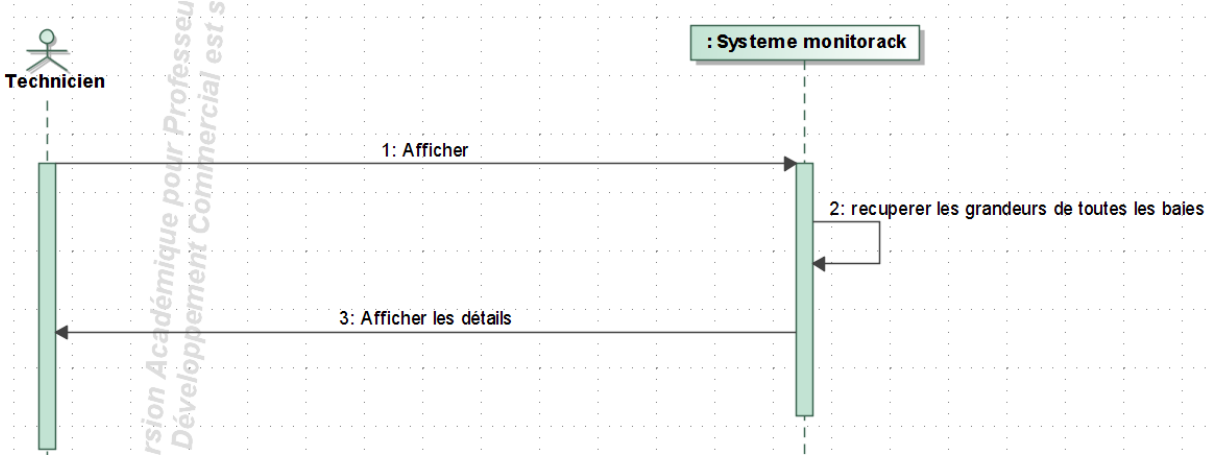
USE CASE /Alerter en cas de coupure ou de dépassement :



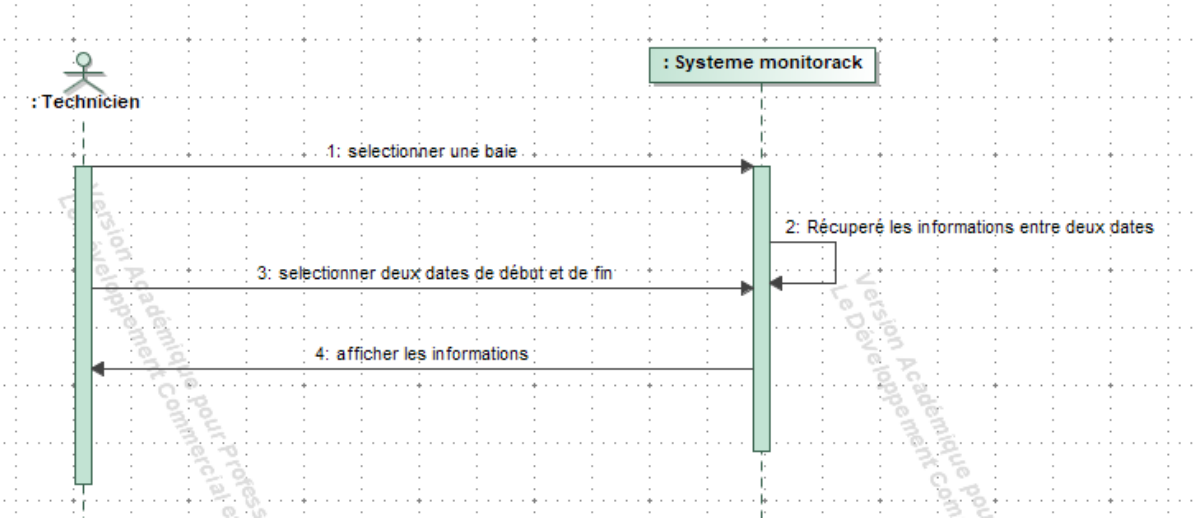
USE CASE /Visualiser les grandeurs d'une baie :



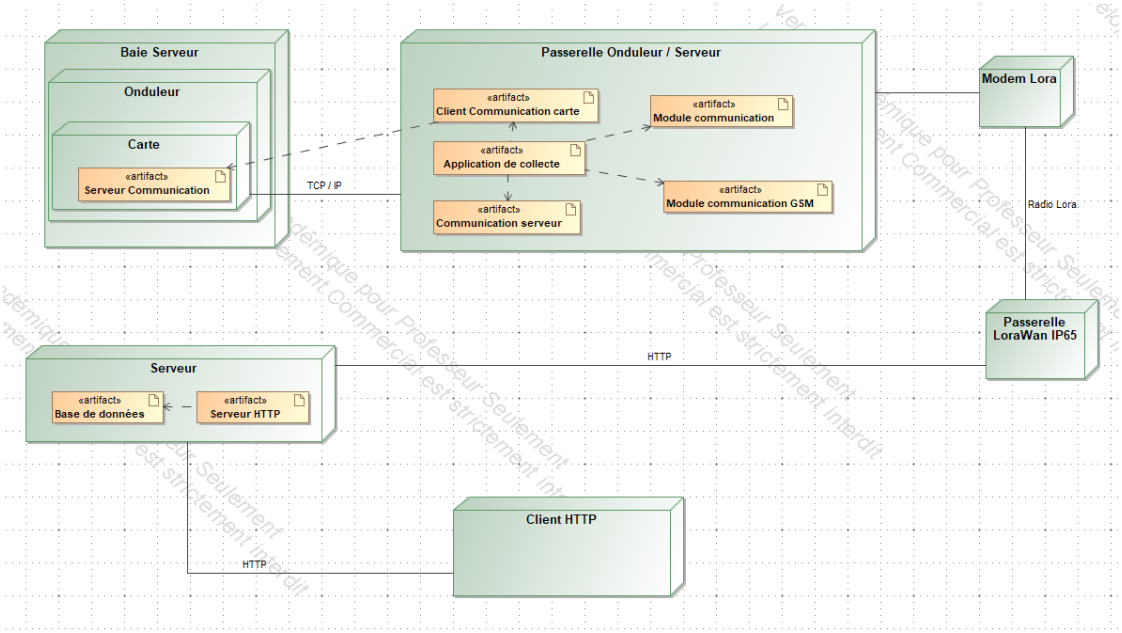
USE CASE /Visualiser toutes les baies :



Visualiser les baies entre deux dates :



2.3. diagramme de déploiement



Ce diagramme de déploiement illustre l'architecture d'un système qui intègre des composants matériels et logiciels pour la gestion et la communication des données provenant d'onduleurs, de passerelles LoRaWAN, et de serveurs.

1. **Onduleur** : Il contient une carte avec un serveur de communication qui échange des données via TCP/IP.

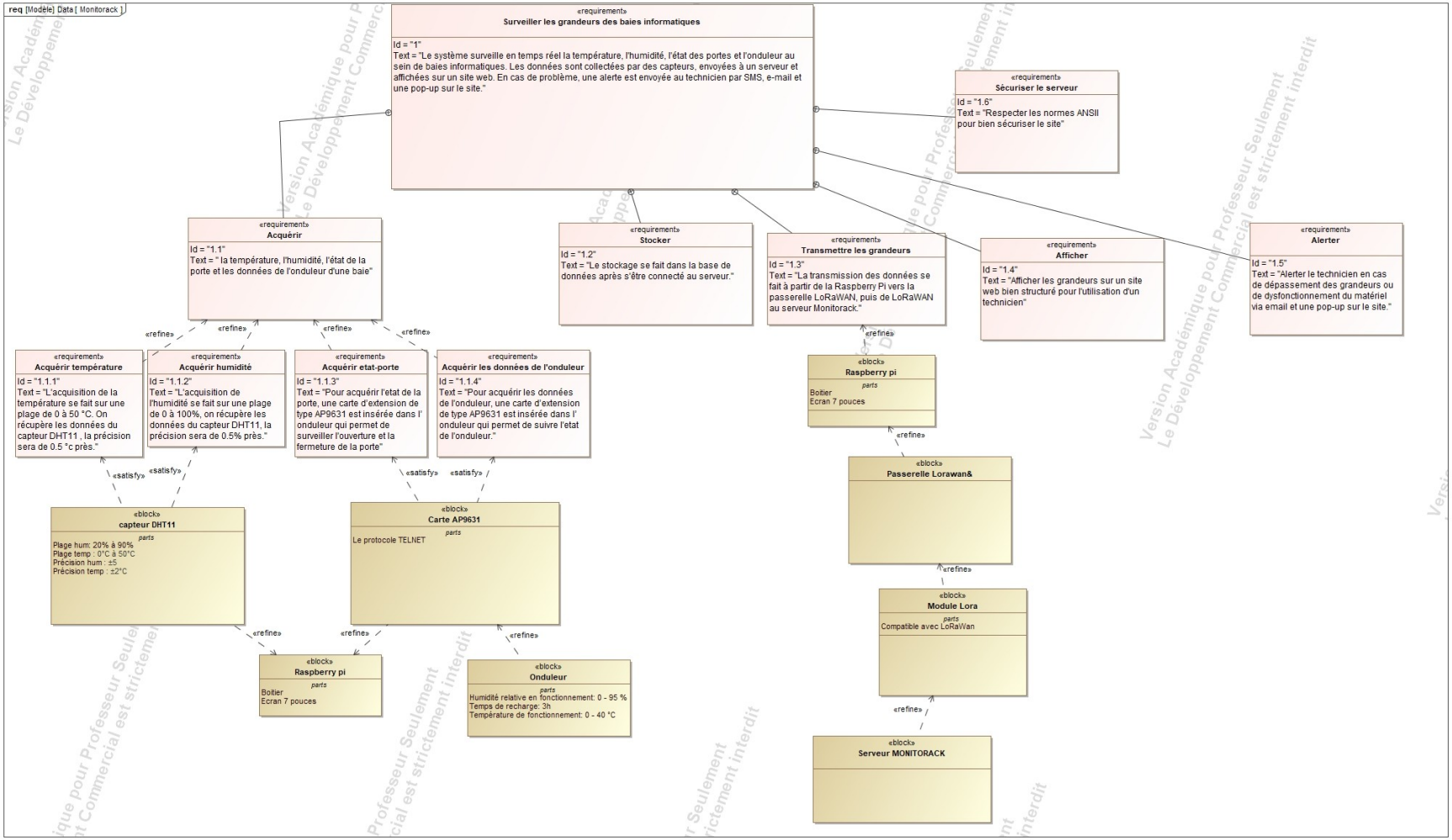
2. **Passerelle onduleur/serveur** : Elle collecte les données de l'onduleur grâce à des modules pour la communication GSM et LoRa.

3. **Modem LoRa et passerelle LoRaWAN** : Ils transmettent les données via le réseau LoRaWAN, utile pour les longues distances et la faible consommation.

4. **Serveur** : Il stocke les données dans une base de données et les rend accessibles via un serveur HTTP.

5. **Client HTTP** : L'utilisateur final accède aux données via des requêtes HTTP.

2.4. Diagramme d'exigence



3 RÉPARTITION DU TRAVAIL

Nom de l'étudiant	Objectifs principaux	Partie du système associe	interfaces / format des échanges
Vigneau Noah	Developement des classes de récupération des données ainsi qu'une IHM sur place	Passerelle onduleur/serveur	Interface avec Raad : Mise à disposition des données récupérées sur la carte de l'onduleur et stockage dans la classe DonneesOunduleur.
Shepho Raad	Développer les classes nécessaire à la communication entre le raspberry et le serveur Monitorack La passerelle et le serveur Monitorack via la communication Lora	Passerelle LoraWan Communication TTN	Interface avec Noah : Récupération des données. Interface avec Youssif : Mise à disposition des données récupérées par l'intermédiaire du module LoRaWAN via TTN.
Matti Yousif Youssif	Développement du backend et création de l'environnement sur le serveur	Partie Serveur (backend)	Interface avec Raad et Esteban : Mise à disposition de la Base de données via des routes.
Dufau Esteban	Développement de la partie frontend (IHM distant)	Partie client (frontend)	Interface avec Youssif : Récupération des données mises à disposition sur la base de données.

4 Contraintes techniques

4.1. Matériels utilisés

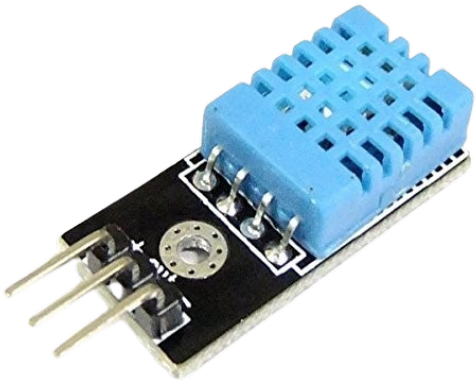
- Une baie

- Un onduleur

- Carte AP9631

- Raspberry PI5 Model B, Quad Core CPU 1.2 GHz, 1 GB RAM

- Capteur DHT11

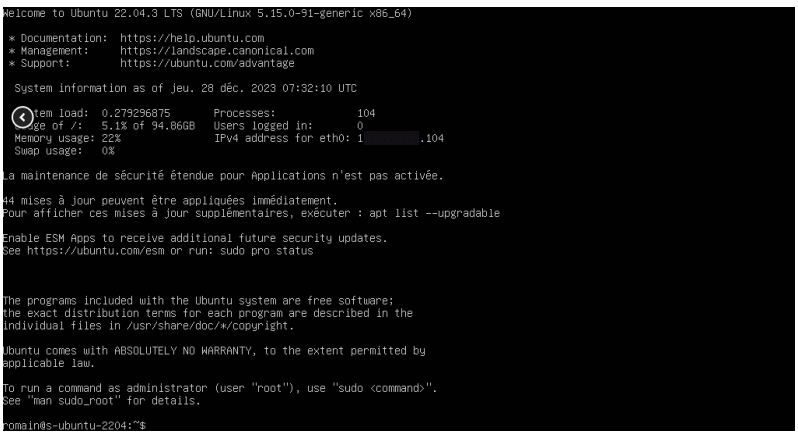


- Écran 7 pouces avec boîtier Raspberry

- Module Lora Wio-E5-LE mini

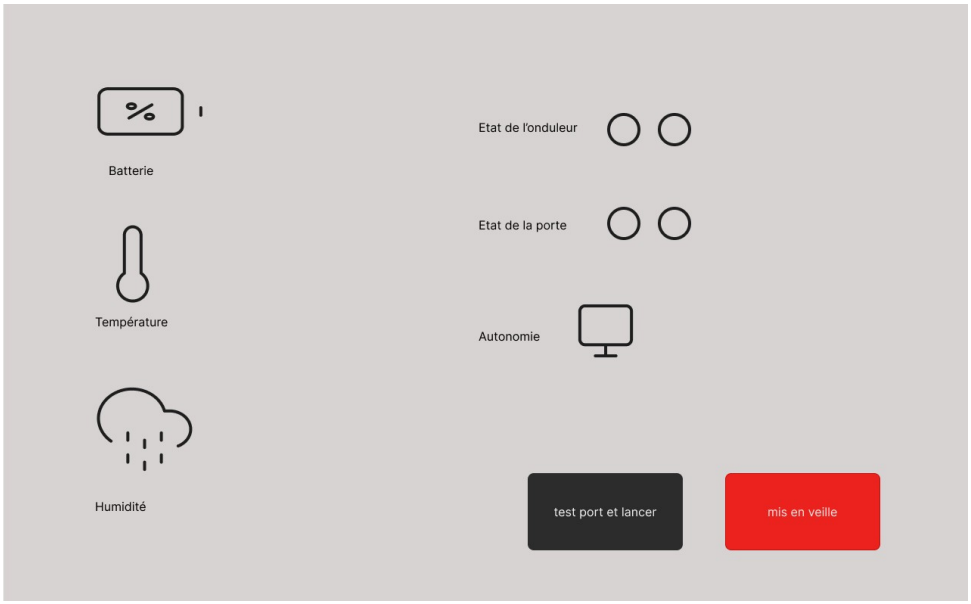
- Passerelle LoraWan, Milesight UG65

- Serveur Ubuntu Server 22.04 sur une machine virtuelle



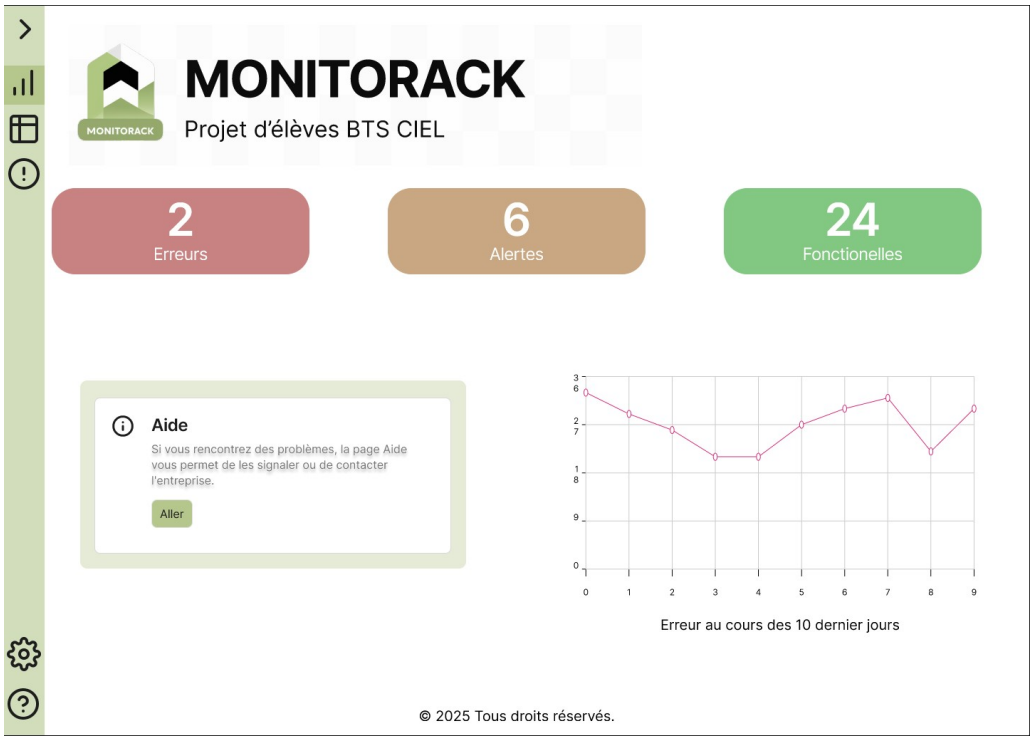
5 IHMs

IHM pour l'affichage sur un écran 7 pouces :

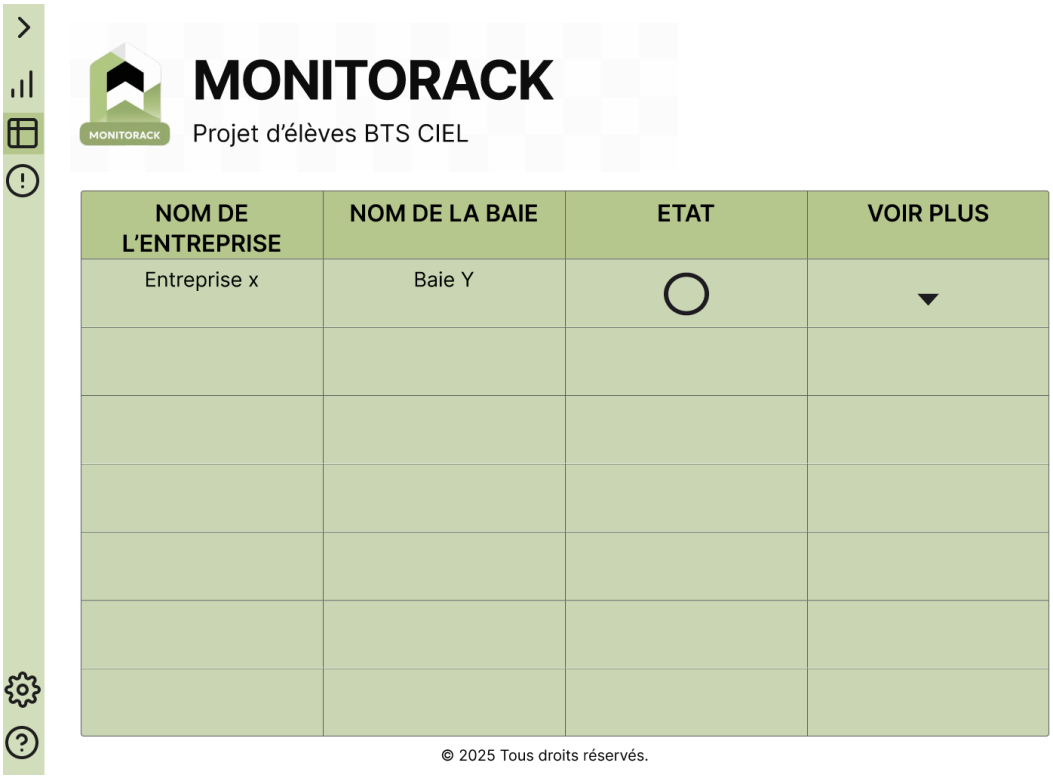


Interface utilisateur du site web :

- Page d'accueil



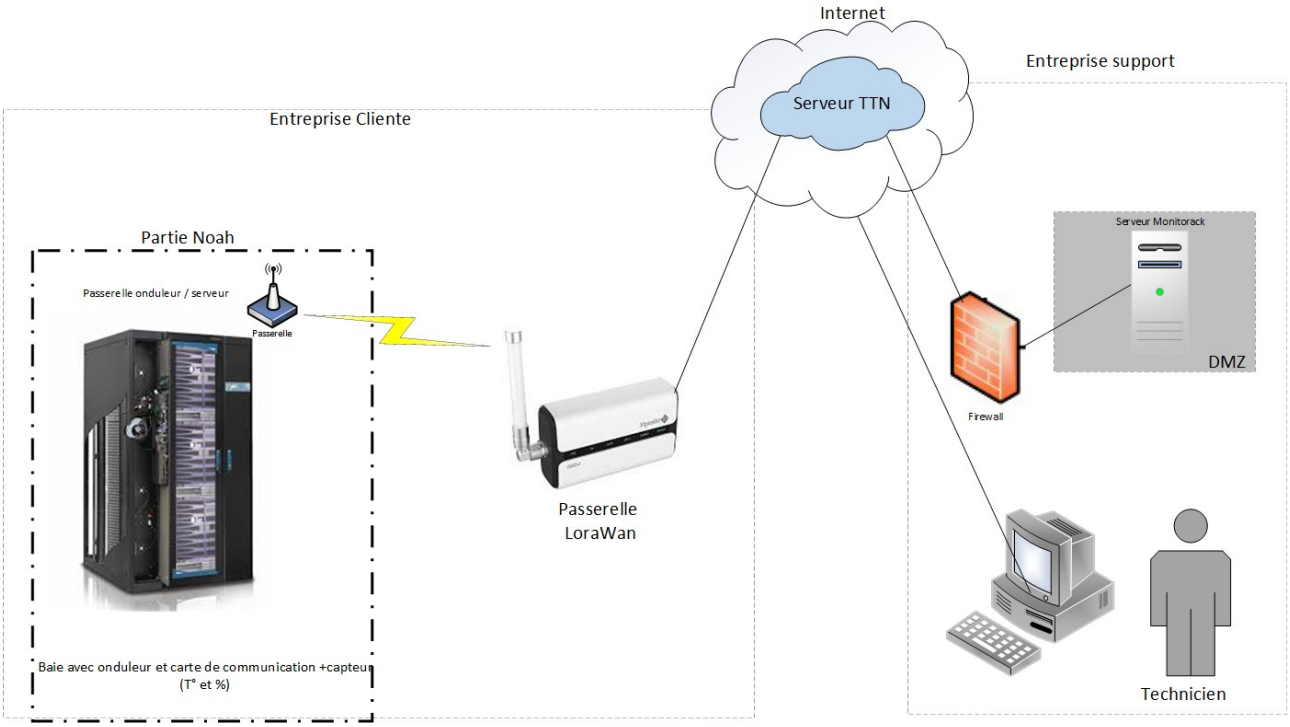
- Page tableau des baies



Deux couleurs indiquent l'état du système : vert si tout fonctionne correctement, rouge en cas de défaut. En cas d'erreur, un clic sur « VOIR PLUS » permet d'afficher des informations complémentaires

6 PARTIE ÉTUDIANT [VIGNEAU Noah]

6.1. Description de la partie personnelle



Je dois créer une application sous QT creator permettant de récupérer des données tels que la température et l'humidité etc.… Sur l'onduleur et de l'afficher sur un écran rasberry pi. L'application se met à jour toutes les minutes afin d'actualiser les données de l'onduleur . Actuellement nous possédons les valeurs de l'onduleur et ensuite récupérer celle du capteur DHT11.

6.2. Mise en œuvre des matériels et technologies employée

Pour ma partie, j'ai utilisé Qt Creator, un environnement de développement intégré (IDE) puissant et polyvalent, conçu spécifiquement pour le développement d'applications avec le framework Qt. Principalement destiné aux langages C++ et QML, Qt Creator fournit une panoplie d'outils qui facilitent grandement le processus de développement. Il intègre notamment un éditeur de code intelligent avec coloration syntaxique, auto-complétion et navigation dans le code, un concepteur graphique (Qt Designer) pour la création d'interfaces utilisateur intuitives en glisser-déposer, ainsi que des outils avancés de débogage et de gestion de projets multi-plateformes.
L'un des principaux atouts de Qt Creator est sa capacité à simplifier le développement d'interfaces graphiques modernes et réactives, tout en permettant une intégration fluide avec des bibliothèques et des fonctionnalités C++. Il prend en charge le développement pour différents systèmes d'exploitation, notamment Windows, Linux, macOS, et les plateformes mobiles.

J'ai du utiliser des classes de la bibliothèques QT :

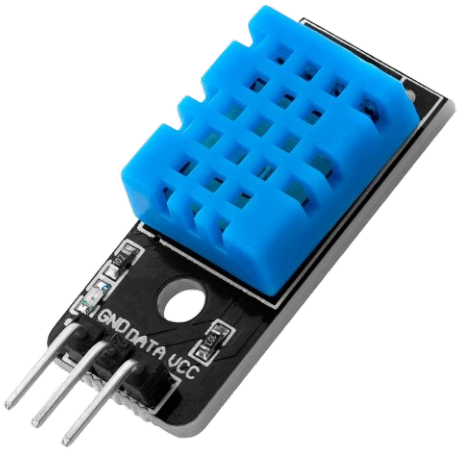
QWidget
Classe de base pour tous les éléments graphiques. Peut être une fenêtre ou un composant (QPushButton , QLabel ,).

Qdialog
Fenêtre secondaire pour interactions ponctuelles (formulaires, messages). Peut être modale ou non.

J'ai du également utiliser un capteur DHT11 :

Le capteur DHT11 sert à mesurer l'humidité et la température de l'air. Il est utilisé dans des projets électroniques pour suivre les conditions environnementales, comme dans des stations météo, des systèmes de régulation de température, ou des appareils de surveillance du taux d'humidité.

-Alimentation : 3 à 5V
-Consommation : 0.5 mA en nominal / 2.5 mA maximum
-Etendue de mesure température : 0°C à 50°C ± 2°C
-Etendue de mesure humidité : 20-90%RH ±5%RH



La carte APC AP9631 est une carte de gestion réseau pour onduleur. Elle permet de surveiller à distance l'état de l'onduleur via une interface web (HTTP/HTTPS), d'envoyer des alertes (email, SNMP), de consulter les données (tension, batterie, température), et d'intégrer l'onduleur à un système de supervision. Elle se connecte en Ethernet et supporte SNMP, Modbus , SSH.Telnet permet d'établir une connexion à distance avec la carte, à travers un terminal en ligne de commande, afin d'envoyer des instructions ou de consulter des informations sur l'état du système.

Grâce à Telnet, il est possible d'accéder à l'interface en ligne de commande de la carte APC9631, ce qui permet notamment de configurer des paramètres réseau, de consulter des logs ou encore de redémarrer certains services. Cette méthode de communication est particulièrement utile dans des environnements techniques où l'accès physique est limité ou lorsque l'on souhaite automatiser certaines tâches via des scripts.

Ensuite j'ai utiliser un rasberryPi afin de faire un passerelle entre la carte APC et le module de communication sans fil Lora permettant d'utiliser le capteur DHT 11 et effectuer des tests pour récupérer la température et le taux d'humidité.

Récupération des informations via TELNET

Avant cela j'ai du faire quelques tests préliminaires tels lire les grandeurs issues de l'onduleur implanté sur la carte AP9631. L'adresse du serveur implanté sur la carte est 10.10.203.200 j'ai pu tester la connexion avec Telnet et récupérer la trame sur WireShark.

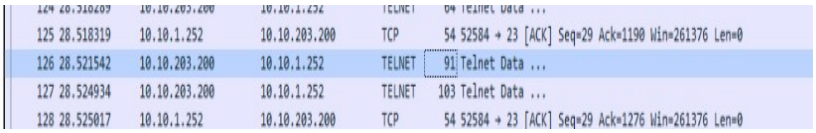
```
User Name : apc
Password  : ***

Schneider Electric                    Network Management Card AOS      v6.4.0
(c) Copyright 2015 All Rights Reserved  Smart-UPS & Matrix-UPS APP       v6.4.0
------------------------------------------------------------------------------
Name      : apcFDB7A1                         Date : 02/10/2025
Contact   : Unknown                           Time : 14:07:21
Location  : Unknown                           User : Super User
Up Time   : 0 Days 0 Hours 20 Minutes         Stat : P+ N4+ N6+ A+


Type ? for command listing
Use tcpip command for IP address(-i), subnet(-s), and gateway(-g)

apc>
History buffer empty.
apc>
History buffer empty.
apc>
```

La procédure pour se connecter a la carte via telnet est :
Se rendre sur le cmd et taper « telnet 10.10.203.200 » et rentrés les mdps apc ,apc

Une fois sur cette interface nous pouvons utiliser des commandes tels que detstatus -all ou uio -st ou encore ups -c. Les grandeurs issues de l'onduleur que l'on peut récupérer via le protocole telnet sont la température et l'humidité , l'état de la porte , online /offline . Le rôle de ces commandes est récupérer l'ensemble des données intéressantes de l'onduleur

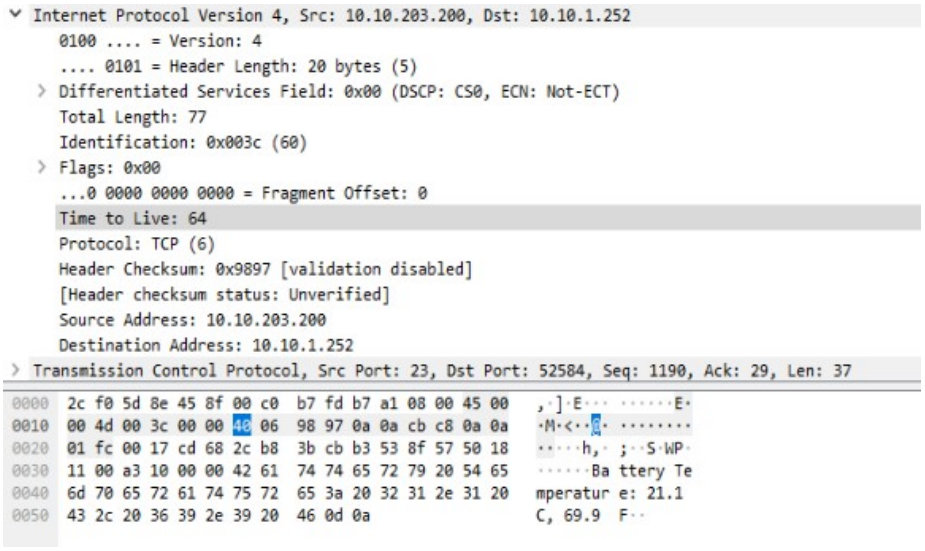
```
apc>detstatus -all
E000: Success
Status of UPS: Off
Last Transfer: None
Input Status: Acceptable
Next Battery Replacement Date:  10/10/2028
Runtime Remaining: 4 hr 27 min 0 sec
Battery State Of Charge: 99.0 %
Output Voltage: 0.0 VAC
Output Frequency: 0.0 Hz
Output Watts Percent: 0.0 %
Output VA Percent: 0.0 %
Output Current: 0.00 A
Output Efficiency: output off
Output Energy: 19556.458 kWh
Input Voltage: 233.2 VAC
Input Frequency: 50.0 Hz
Battery Voltage: 54.5 VDC
Battery Temperature: 17.1 C, 62.7 F
Self-Test Result: Refused via management device due to invalid state
Self-Test Date: 11/18/2024
Calibration Result: Unknown
Calibration Date: Unknown
```

Nous souhaitons récupérer la « température de la batterie », « l'état de l'onduleur» , « l'autonomie », « Le pourcentage de charge de la batterie »



125	28.518319	10.10.1.252	10.10.203.200	TCP	54 52584 → 23 [ACK] Seq=29 Ack=1190 Win=261376 Len=0
126	28.521542	10.10.203.200	10.10.1.252	TELNET	91 Telnet Data ...
127	28.524934	10.10.203.200	10.10.1.252	TELNET	103 Telnet Data ...
128	28.525017	10.10.1.252	10.10.203.200	TCP	54 52584 → 23 [ACK] Seq=29 Ack=1276 Win=261376 Len=0

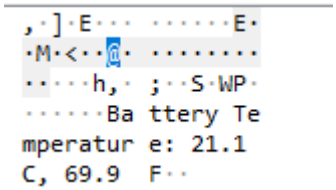
Pour la suite nous sommes sur WireShark situé sur la trame en bleu afin de retrouvé les données vu au dessus via telnet



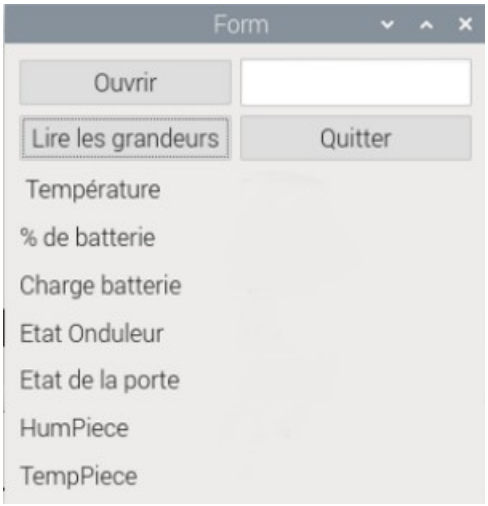
Nous apercevons l'ensemble de la trame que l'on peut décoder en cliquant sur les nombres en hexadécimal et ensuite on regarde en bas a droite le décodage.

Récupération de la trame sous wireshark
Une fois la trame affichée la totalité des informations sont présentes afin d'être traité.

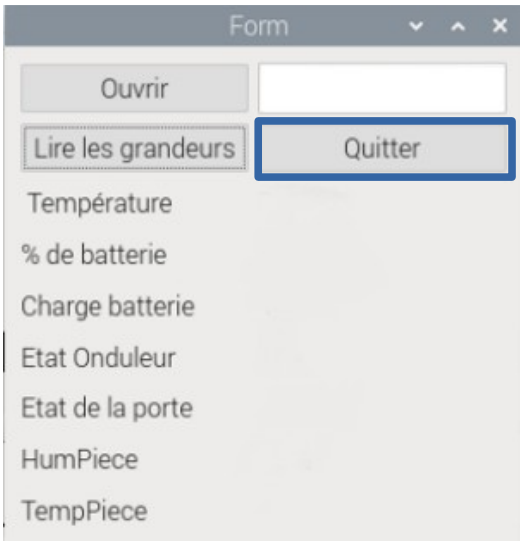
Car nous récupérons la température de la batterie cela prouve que la connexion telnet a bien marcher



Par la suite , nous allons voir sur Qt creator le fonctionnement de l'application



Le bouton Lire les grandeurs sert à déclencher une action qui permet de lire et récupérer les valeurs telles que la température, l'humidité, ou d'autres données provenant de la carte APC9631. Grace a un connect : `connect` sert à lier un signal (événement) à un slot (fonction). Quand un signal est émis (ex : clic), le slot connecté est exécuté automatiquement.



Le bouton Quitter permet a l'utilisateur de fermer immédiatement le programme

Dans l'IHMPrincipal il y'a ceci

```
connect(ord, SIGNAL(miseAJIHM()), this, SLOT(majIHM()));
```

`ord` : c'est un pointeur vers un objet de la classe `Ordonnancement`. Cet objet est responsable de récupérer ou traiter des données par un thread.

`SIGNAL(miseAJIHM())` : il s'agit d'un signal que l'objet `ord` va émettre lorsque de nouvelles données sont disponibles.

`this` : il s'agit ici de l'objet courant, donc une instance de la classe `IHMPrincipal`.

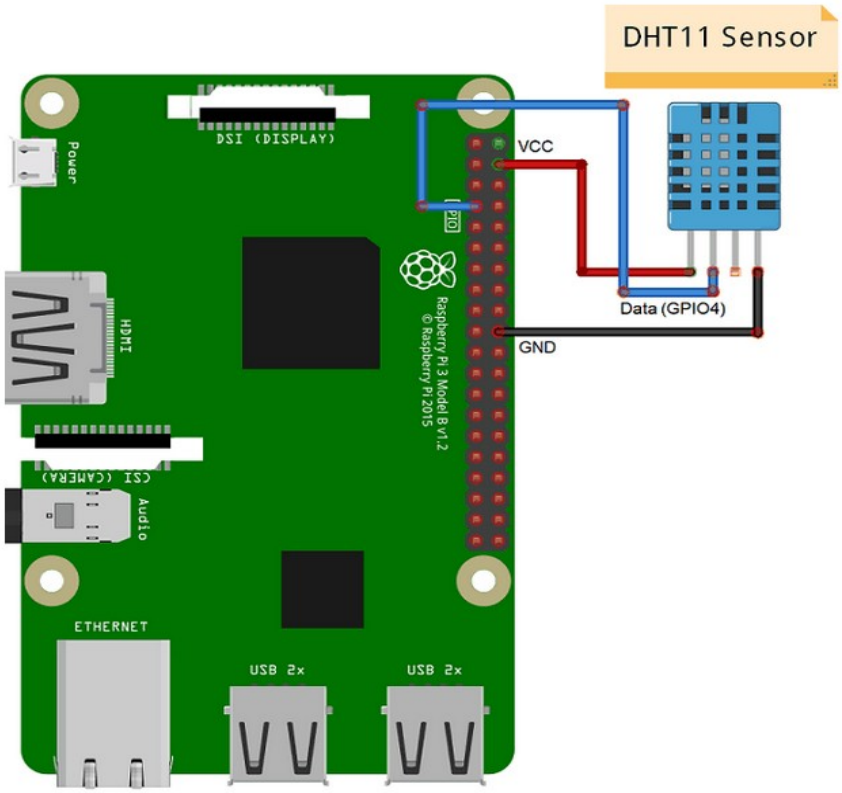
`SLOT(majIHM())` : c'est une fonction (slot) définie dans `IHMPrincipal`, qui est appelée automatiquement lorsque le signal `miseAJIHM()` est émis.

La classe `DonneeOnduleur` sert à stocker toutes les informations liées à l'onduleur. Elle contient uniquement des attributs de type `QString`, un type utilisé dans Qt pour manipuler des chaînes de caractères. Cela permet de stocker les valeurs envoyées par les capteurs ou l'onduleur, qui arrivent souvent sous forme de texte. Les attributs sont : `temp` (température de la batterie), `charge` (niveau de charge), `batVoltage` (tension de la batterie), `statusUPS` (état général de l'onduleur), `valOnduleur` (valeur mesurée), `autonomie` (autonomie restante) et `humidite` (humidité ambiante).
Cette classe est utilisée par d'autres classes qui accèdent à ses données via des pointeurs ou appellent ses méthodes `set` ou `get` :

`-GestionCapteur` utilise `DonneeOnduleur` pour stocker les données lues depuis les capteurs (via les méthodes `setTemp()`, `setHumidite()`, etc.).
`-Ordonnancement` interagit aussi avec cette classe pour lire ou mettre à jour les données depuis un thread.

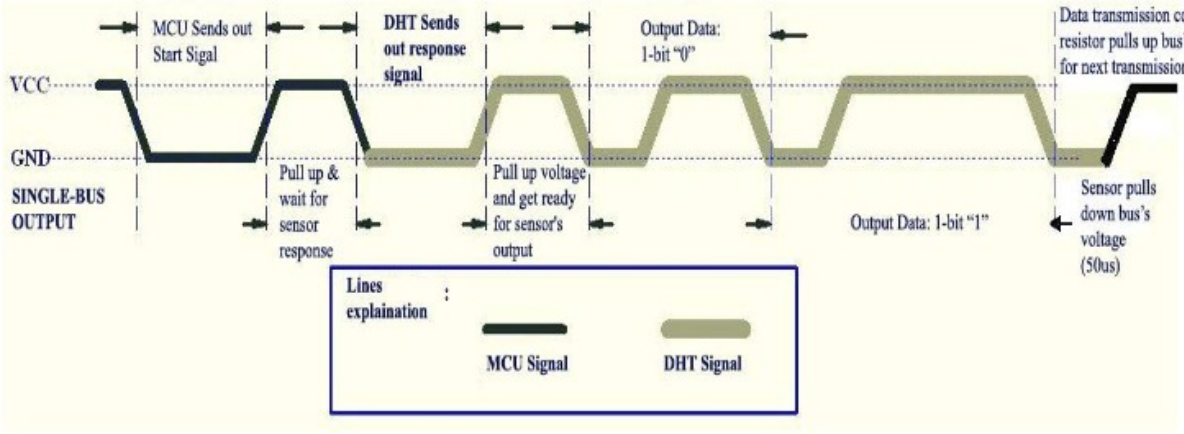
`-IHMPrincipal` possède un objet de type `DonneeOnduleur` et l'utilise pour afficher les valeurs dans l'interface graphique.
Ces classes utilisent donc `DonneeOnduleur` un peu comme un "centre de stockage", en appelant ses méthodes à travers un pointeur, grâce à l'utilisation de use.

J'ai également testé le capteur DHT11 afin de récupérer la température et l'humidité de la salle.
Le DHT11 a une sortie numérique. Les données de température et d'humidité sont envoyées sous forme de signal numérique à l'aide d'un protocole de communication 1 fil (one-wire).



Le DHT11 est en mode basse consommation par défaut et attend une requête du maître (microcontrôleur).

Requête du maître : Le microcontrôleur envoie une requête pour demander la température et l'humidité.



Le DHT11 utilise un protocole de type maître-esclave pour communiquer avec une carte comme l'Arduino. Le maître (la carte Arduino) est responsable de l'initiation de la communication, tandis que l'esclave (le capteur DHT11) ne fait que répondre à la demande.

En temps normal, la broche de signal (SIG) est au repos, maintenue à l'état haut grâce à une résistance de pull-up. Pour démarrer la communication, le maître envoie un signal de réveil au capteur. Il force la ligne SIG à l'état bas pendant au moins 18 millisecondes. Ce signal sert à prévenir le capteur qu'une demande de lecture va être faite. C'est le signal d'initiation.

Une fois ce signal terminé, le maître relâche la ligne SIG, qui repasse à l'état haut pendant un court instant. Ensuite, le capteur prend le relais et envoie un signal de réponse : il met la ligne à l'état bas pendant environ 80 microsecondes, puis à l'état haut pendant encore 80 microsecondes. Cela signifie que le capteur a bien reçu la demande et qu'il se prépare à transmettre les données.

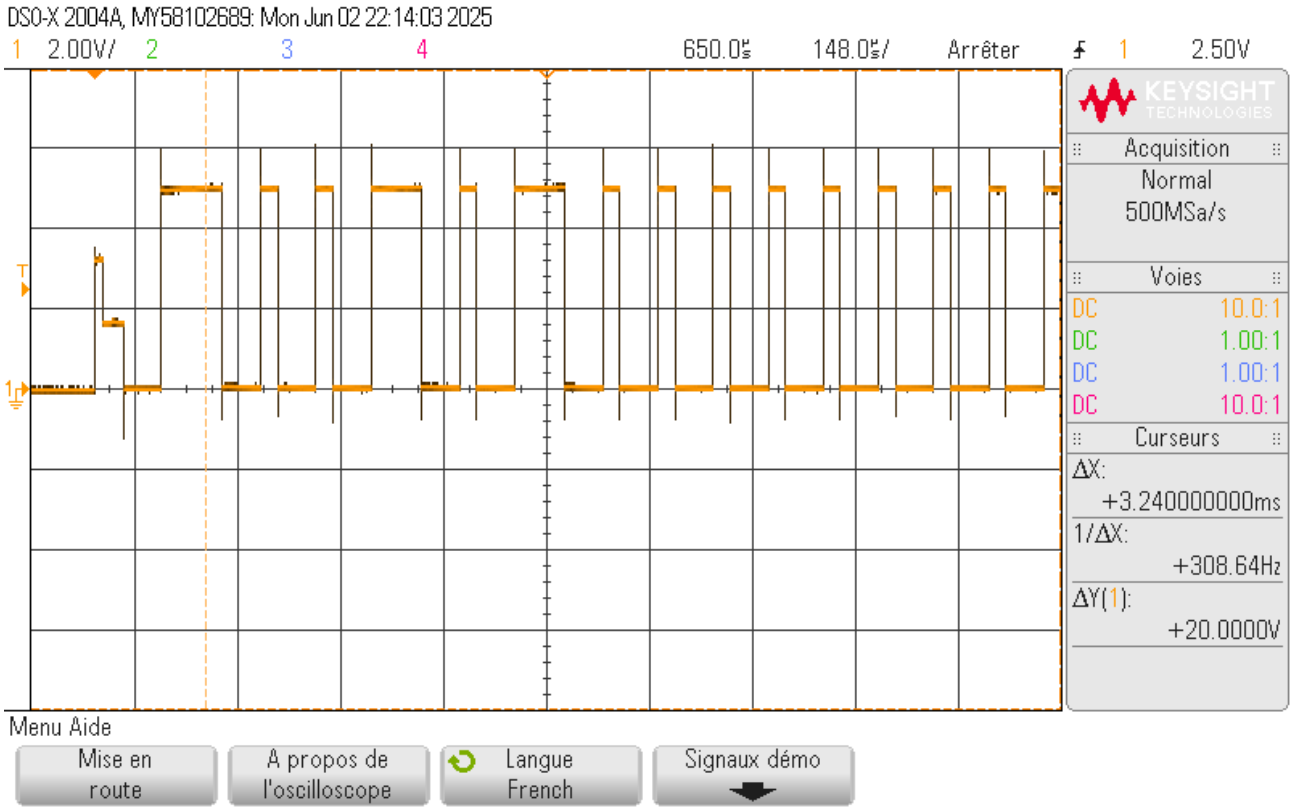
Résolution :

Température : 1°C.

Humidité : 1%.

Pour l'étendue de mesure de température vue précédemment elles sont cohérentes car dans une baie il est impossible d'avoir des températures négatives et maximum 50°c est réaliste.

Consommation : faible, environ 2 à 5 mA pendant la mesure.



Voici le déroulement de la trame du capteur DHT11 lorsqu'on souhaite apercevoir le comportement de ce capteur qui est en relation maître esclave (one wire)

Le capteur DHT11 envoie ensuite 40 bits de données au maître. Ces données sont envoyées sous forme série, bit par bit. Les 40 bits sont divisés comme suit : 8 bits pour l'humidité entière, 8 bits pour l'humidité décimale (toujours 0 avec le DHT11), 8 bits pour la température entière, 8 bits pour la température décimale (toujours 0 avec le DHT11), et 8 bits pour la somme de contrôle (checksum). Chaque bit est transmis en deux temps : d'abord un niveau bas pour indiquer le début du bit, puis un niveau haut dont la durée détermine s'il s'agit d'un 0 ou d'un 1 (court pour 0, long pour 1).

Ainsi, tout au long de la communication, c'est toujours le maître qui initie l'échange, et l'esclave ne répond que s'il y est invité. Ce protocole assure une communication simple et efficace entre le capteur et la carte.

6.3. Conception détaillée

Dans un premier temps nous allons voir le diagramme de classes afin de comprendre le système et les différentes

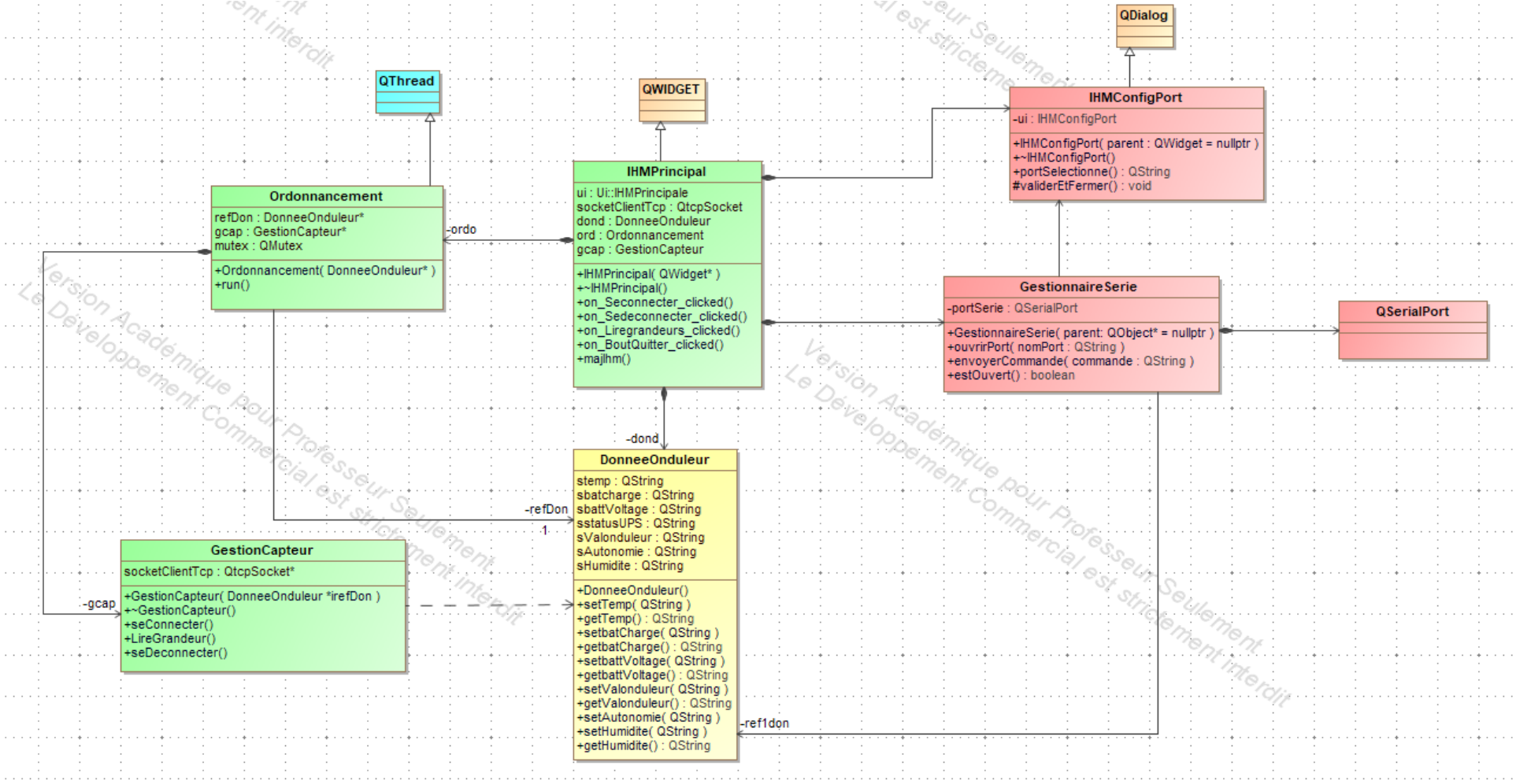
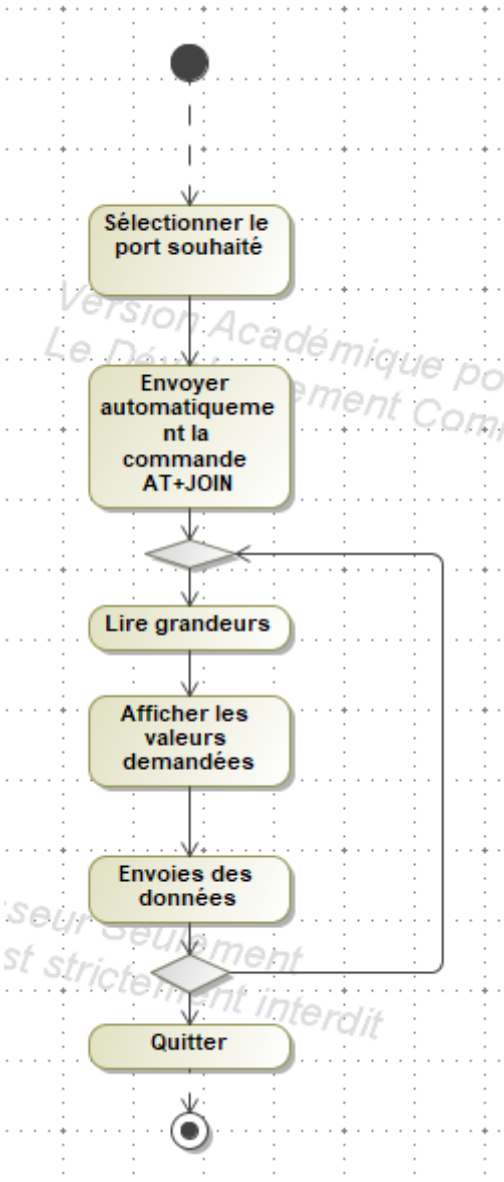


Diagramme d'activité :



6.4. Réalisation et résumé des tests fonctionnels unitaires

Unité tester	Annexes	Validité
Test récupération donnée	Annexe 1	OK
Test capteur dht11	Annexe 2	OK
Test mis a jour IHM	Annexe 3	OK
Test envoie sur le serveur	Annexe 4	OK

6.5. Planning des tâches et jalons

6.6. Bilan personnel technique

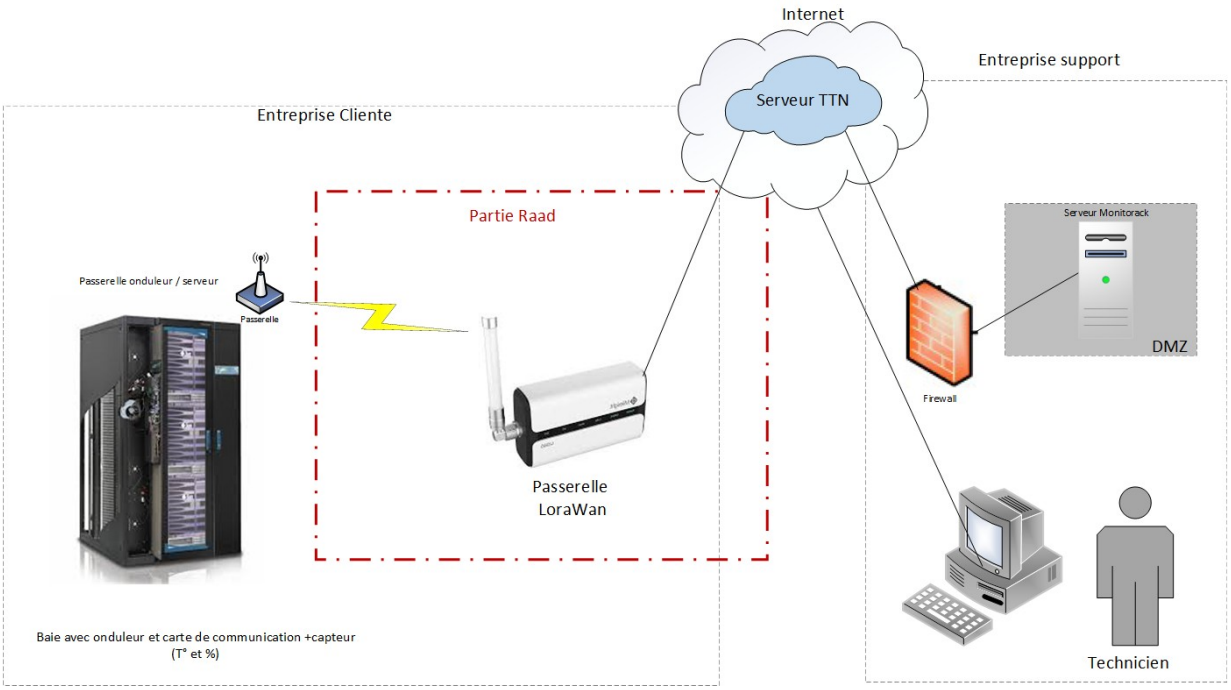
Ce projet m'a permis de développer de solides compétences sur Qt et de devenir beaucoup plus à l'aise en programmation au fil des mois. J'ai réalisé une application complète sous Qt Creator, capable de récupérer les données d'un onduleur situé dans une baie via le protocole Telnet. Les informations collectées incluent la température, l'humidité, le niveau de charge de la batterie, la tension et l'autonomie. Pour compléter ces mesures, j'ai également utilisé un capteur DHT11 afin de comparer et enrichir les données relevées.

J'ai conçu une interface IHM claire et fonctionnelle permettant d'afficher toutes ces données en temps réel. L'application a ensuite été intégrée sur un Raspberry Pi, auquel j'ai connecté un écran tactile 7 pouces pour permettre un usage pratique sur le terrain par un technicien. Cela m'a permis de travailler également sur la partie déploiement et portabilité de l'application.

Le code a été mis en commun avec mon collègue, et l'application permet désormais une connexion à l'onduleur et la récupération automatique des valeurs demandées. Il ne reste actuellement qu'à intégrer le chargement d'un fichier de configuration afin de permettre à l'utilisateur de sélectionner facilement la baie à interroger, ce qui facilitera encore davantage le travail du technicien.

7 PARTIE ÉTUDIANT [SHEPHO Raad]

7.1. Description de la partie personnelle



Dans le cadre du projet MONITORACK, ma mission consiste à assurer l'acheminement des données captées (température, humidité, tension batterie, état de la porte, etc.) depuis la passerelle (Raspberry Pi) vers le serveur Monitorack via le réseau LoRaWAN.
Pour cela, j'ai réalisé les tâches suivantes :

Configuration du module LoRa Wio-E5-LE mini pour établir la communication avec la passerelle LoRaWAN Milesight UG65.

Paramétrage de la passerelle UG65 afin qu'elle soit connectée au réseau The Things Network (TTN).

Développement d'une classe de communication série dans Qt Creator, en utilisant QSerialPort, pour envoyer les données au format hexadécimal.

Intégration des données générées par la carte APC9631 et le capteur DHT11 dans une classe commune partagée avec Noah (`DonneeOnduleur`), afin de construire les trames à transmettre.

Validation de la communication TTN, en testant l'envoi et le décodage des trames dans la

console du cloud TTN, avant leur transfert vers le serveur Monitorack.

Je travaille en collaboration directe :
Avec Noah, pour la récupération des données issues de son application (température, tension, charge batterie, etc.) connectée à la carte AP9631 d'une classe commune.
Avec Youssif, pour assurer le bon transfert et l'intégration des données dans la base de données et l'afficher à l'aide de Frontend sur le web Monitorack, hébergée sur le serveur de l'entreprise CI Solutions.

7.2. Mise en œuvre des matériels et technologies employée

La passerelle LoRaWAN UG65 sert de pont entre les dispositifs LoRa et le réseau LoRaWAN. Elle reçoit les données des modules LoRa, comme le Wio-E5-LE mini, et les transmet via Internet à un serveur LoRaWAN. Cela permet de connecter des capteurs et autres appareils à l'Internet pour le traitement des données.

Dans mon projet, la passerelle UG65 reçoit les informations du Wio-E5-LE mini et les relaie au serveur LoRaWAN pour traitement, stockage et analyse. Elle est compatible avec différentes configurations de réseau et plusieurs bandes de fréquences, ce qui la rend flexible et utilisable à l'international.

La Milesight UG65 est une passerelle LoRaWAN semi-industrielle à 8 canaux, adaptée aux environnements intérieurs et extérieurs.

Caractéristiques principales:

Fréquences supportées	EU868, US915, AU915, AS923, etc.
Modulations	LoRa (SF5 à SF12), FSK
Sensibilité	jusqu'à -140 dBm
Puissance d'émission	jusqu'à 27 dBm
Connectivités	Ethernet, Wi-Fi, 4G
Nombre de nœuds pris en charge	jusqu'à 2000
Portée	jusqu'à 15 km en champ libre, 2 km en zone urbaine
Serveurs compatibles	The Things Stack, ChirpStack, Actility, etc.

Le Wio-E5-LE mini est un module à faible consommation qui utilise LoRa pour transmettre des données sans fil. Dans mon projet, j'utilise Qt Creator pour programmer le module et envoyer des données à la passerelle via une connexion série gérée par QSerialPort, ce qui permet d'envoyer les données des capteurs comme le DHT11 à la passerelle LoRaWAN.

Le module Wio-E5-LE mini est basé sur le microcontrôleur STM32WLE5JC, intégrant une radio LoRa SX126x. Il est conçu pour des applications IoT à faible consommation et longue portée.

Pour le configurer on utilise la logiciel Hercules et pour connecter au module Lora on utilise la configuration suivante:

Paramètre	Valeur	Description
Nom du port	NOM-PORT	Nom du port série utilisé pour la communication
Vitesse (baud)	9600	Vitesse de transmission en bauds (bits/seconde)
Taille des données	8	Nombre de bits de données par caractère
Parité	Aucune (none)	Aucun bit de parité utilisé pour la vérification d'erreurs
Contrôle de flux (Handshake)	OFF	Pas de contrôle matériel (RTS/CTS) ou logiciel (XON/XOFF)
Mode	Libre (Free)	Le port est libre pour envoyer/recevoir sans contrainte particulière

Après de connecter au module Lora on a les commandes suivante pour le configurer:

AT Commandes	Definition
AT+JOIN	LoRaWAN OTAA JOIN
AT+ID	LoRaWAN DevAddr/DevEui/AppEui
AT+KEY	NWKSKEY/APPSKEY/APPKEY
AT+MODE	LWABP, LWOTAA, TEST
AT+CH	Fréquence du canal LoRaWAN
AT+DR	Débit de données LoRaWAN
AT+POWER	LoRaWAN TX power

Caractéristiques principales:

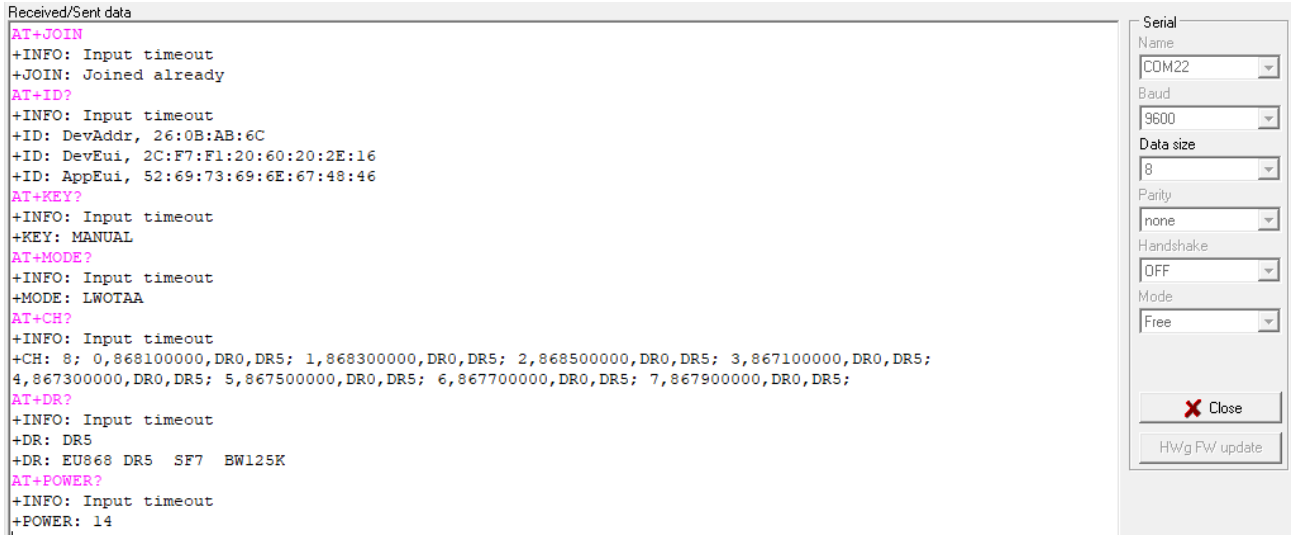
Fréquences supportées	868 MHz (Europe), 915 MHz (USA)
Modulations	LoRa, (G)FSK, BPSK, GMSK
Sensibilité	jusqu'à -136,5 dBm (SF12, BW 125 kHz)
Puissance d'émission	jusqu'à 14 dBm
Interfaces	UART, I2C, SPI, ADC, GPIO
Consommation en veille	2,1 µA
Portée	jusqu'à 10 km en champ libre

Format de la trame LoRaWAN
Les données collectées via la carte AP9631 et le capteur DHT11 sont encapsulées dans une trame hexadécimale de 9 octets:

Octet(s)	Nom du champ	Description
0-1	Température de batterie	Température ×100. Ex : 25.75°C → 2575 (2 octets, MSB-LSB)
2	Humidité de batterie	Taux d'humidité en %
3	Niveau de batterie	Niveau de charge de la batterie en %
4-5	Autonomie de batterie	Autonomie restante (ex 04:30:00)
6	Température de pièce	Température ambiante dans la baie en °C
7	Humidité de pièce	Taux d'humidité ambiant en %
8	État des capteurs	Bit 0 = porte (0: fermée / 1: ouverte) Bit 1 = onduleur (0: OFF / 1: ON)

Dans notre projet on a pas besoin d'envoyer l'humidité de batterie vers le serveur donc je envoie pas la deuxième octet, je le garde si c'est besoin.

Configuration Wio-E5-LE mini
Ce capture montre la configuration de module lora que je l'avais fais pour que il soit la même que la configuration de la passerelle LoRaWAN.



Configuration LoRaWAN

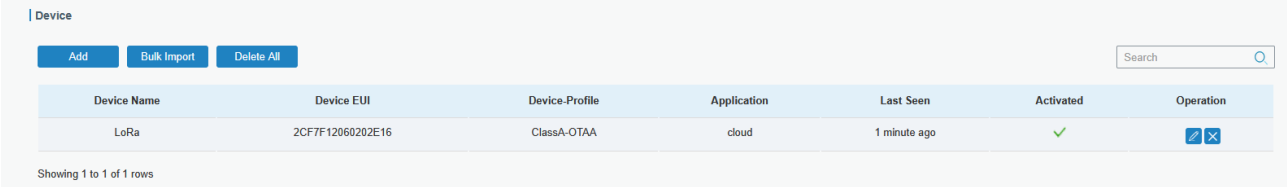
Lors de la configuration de mon module Wio-E5-LE mini sur la plateforme de gestion LoRaWAN via TTN, j'ai dû renseigner plusieurs paramètres essentiels pour permettre la communication LoRaWAN en mode OTAA (Over-The-Air Activation).

Sur la capture, on voit les informations suivantes:

Device EUI	Identifiant unique du module	AT+ID=DevEui
Application Key	Clé d'authentification utilisée pour sécuriser la jonction OTAA	AT+KEY
Device Profile	Sélectionné le profil ClassA-OTAA, adapté aux capteurs basse consommation.	AT+MODE=LWOTAA

```
Received/Sent data
AT+ID=DevEui
+INFO: Input timeout
+ID: DevEui, 2C:F7:F1:20:60:20:2E:16
AT+KEY?
+INFO: Input timeout
+KEY: MANUAL
AT+MODE?
+INFO: Input timeout
+MODE: LWOTAA
```

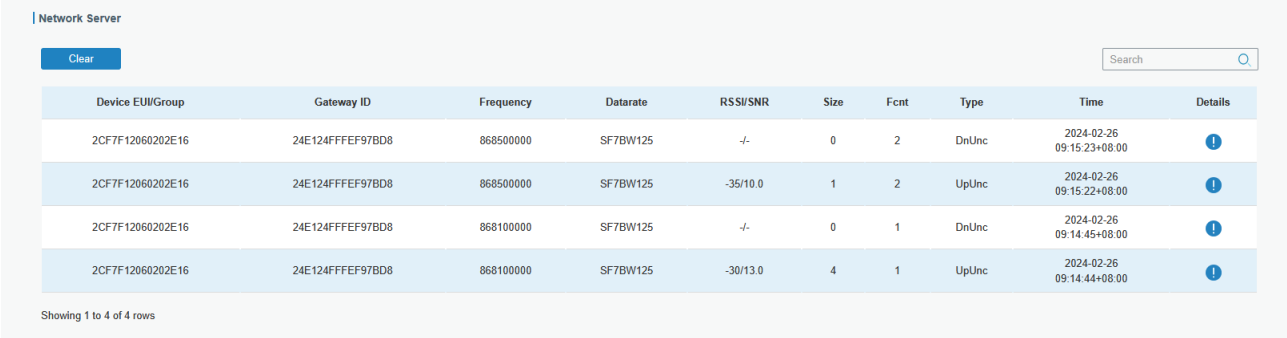
Il est évident que le module est correctement activé, comme en témoigne son état actuel, ce qui confirme que la configuration a été effectuée avec succès.



J'ai envoyé la commande AT+JOIN pour me connecter, en utilisant Hercules comme outil de communication.

```
AT+JOIN
+JOIN: Start
+JOIN: NORMAL
+JOIN: Network joined
+JOIN: NetID 010203 DevAddr 07:C8:E8:A5
+JOIN: Done
```

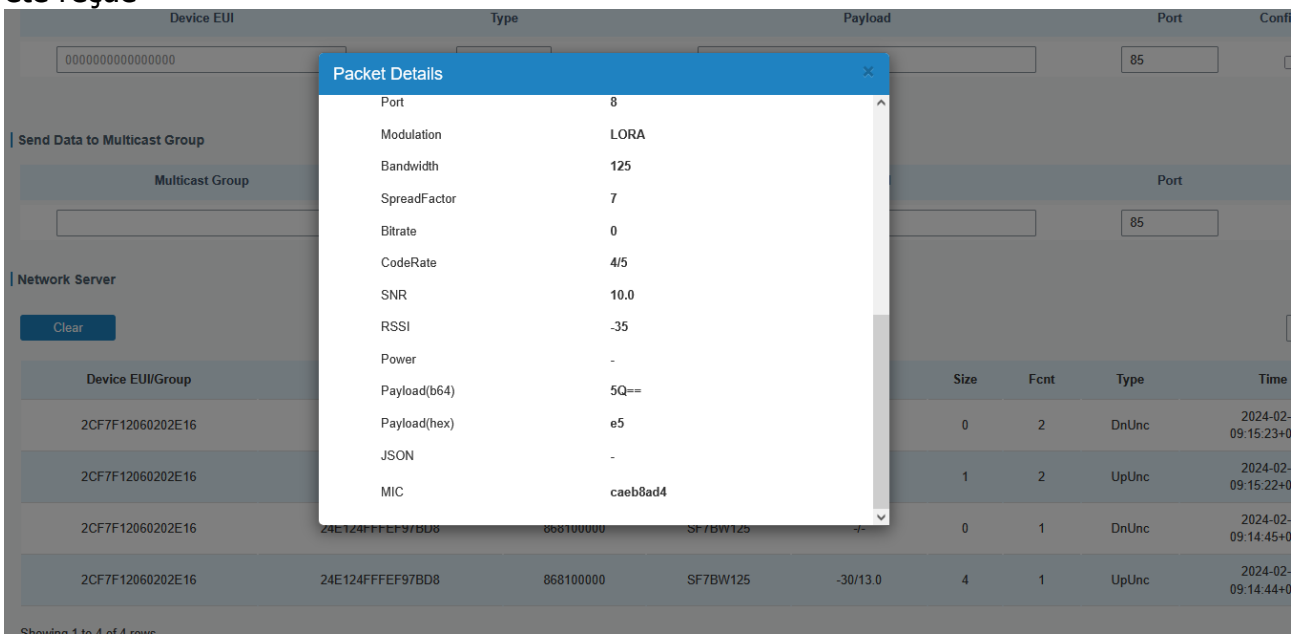
J'ai reçu les données de connexion, ce qui confirme que la connexion fonctionne correctement.



J'ai envoyé un message en format hexadécimal en utilisant la commande AT+MSGHEX=E5 via Hercules.

```
AT+MSGHEX=E5
+INFO: Input timeout
+MSGHEX: Start
+MSGHEX: FPENDING
+MSGHEX: RXWIN1, RSSI -21, SNR 9.0
+MSGHEX: Done
```

Après avoir envoyé une trame via la commande AT+MSGHEX depuis le module Wio-E5-LE mini, j'ai pu vérifier dans l'interface de la passerelle Milesight UG65 que la trame a bien été reçue



L’encadré "Packet Details" affiche les détails techniques du paquet reçu :

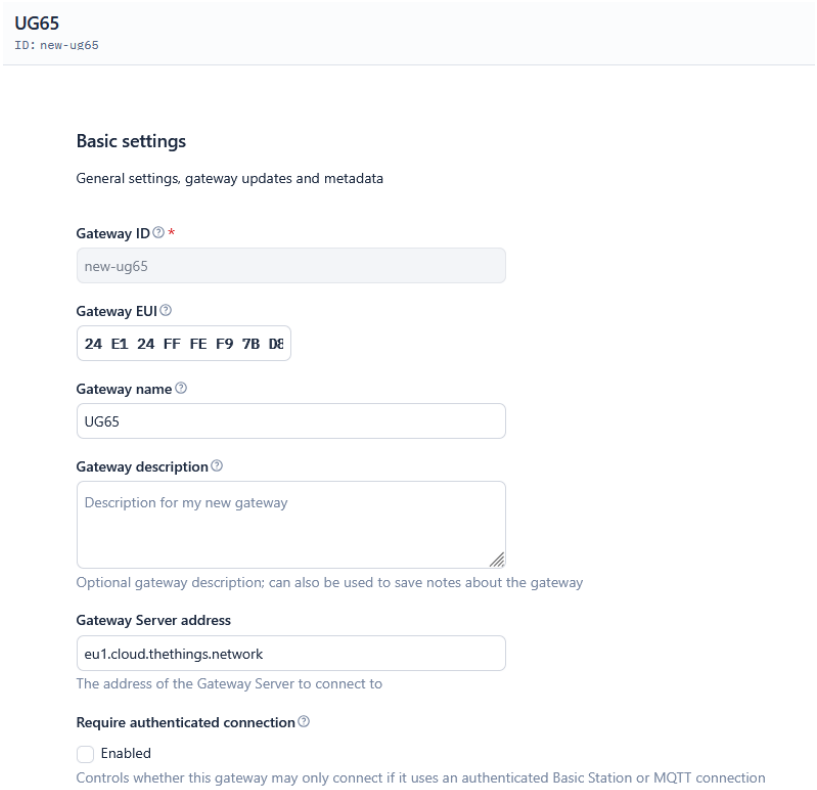
Port	8	Port applicatif défini par le module.
Modulation	LoRa	La modulation utilisée pour la transmission.
Bandwidth	125 kHz	Bande passante classique en LoRaWAN EU868.
SpreadFactor	7	Facteur d’étalement (plus il est élevé, plus la portée est grande mais le débit est faible).
CodeRate	4/5	Taux de correction d'erreur.
SNR	10.0	Rapport signal/bruit, indiquant une bonne qualité de réception.
RSSI	-35 dBm	Puissance du signal reçu, très bonne réception,
Payload (hex)	e5	Contenu de la trame en hexadécimal,
MIC	caeb8ad4	Code d'intégrité du message, utilisé pour vérifier l’authenticité du paquet.

Ces informations m’ont permis de valider que les données envoyées via la liaison série entre Hercules et le module LoRa étaient bien transmises à la passerelle, puis traitées correctement au niveau LoRaWAN.

Configuration TTN
Tout d'abord, j'ai commencé par créer un compte sur TTN, puis une application, avant d'enregistrer la passerelle Milesight.

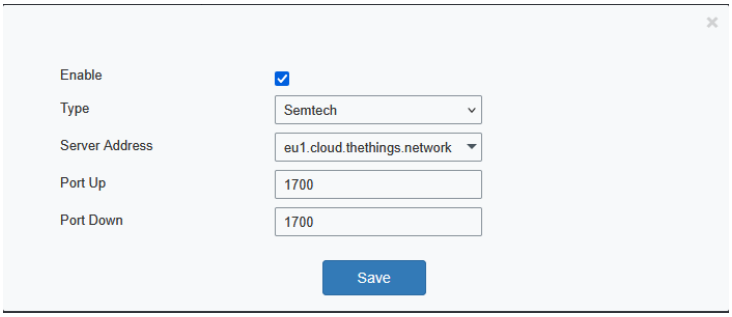
Email	prjmonitorack@gmail.com
Identifient	lyceestcricq
Mot de passe	Session2025!
Nom de application	Prj-Monitorack

J'ai ensuite procédé à l'ajout de la passerelle en renseignant tous les paramètres nécessaires à sa configuration, afin d'assurer une intégration correcte dans le réseau.



Ce capture montre la configuration de la passerelle sur TTN:
Gateway ID : Identifiant unique de la passerelle dans le réseau.
Gateway EUI : Identifiant matériel de la passerelle, généralement défini par le fabricant.
Gateway name et description : Nom convivial et description libre pour faciliter l'identification.
Gateway Server address : Adresse du serveur auquel la passerelle doit se connecter.

Ensuite, j'ai configuré la passerelle afin d'établir un lien direct avec le serveur réseau de The Things Network (TTN). Pour cela, j'ai sélectionné le type de connexion Semtech UDP, un protocole couramment utilisé pour le transfert de paquets LoRaWAN.

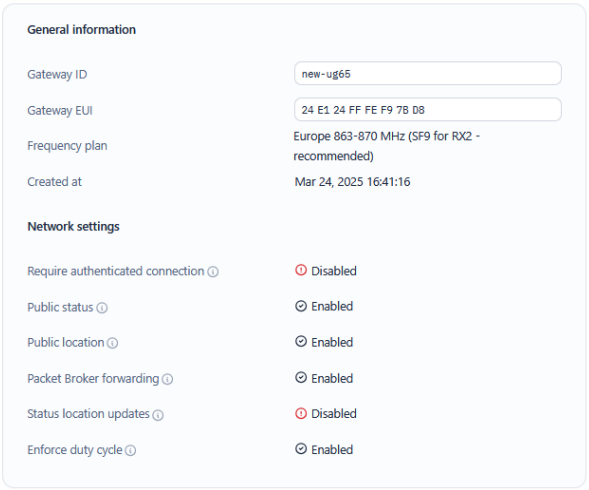


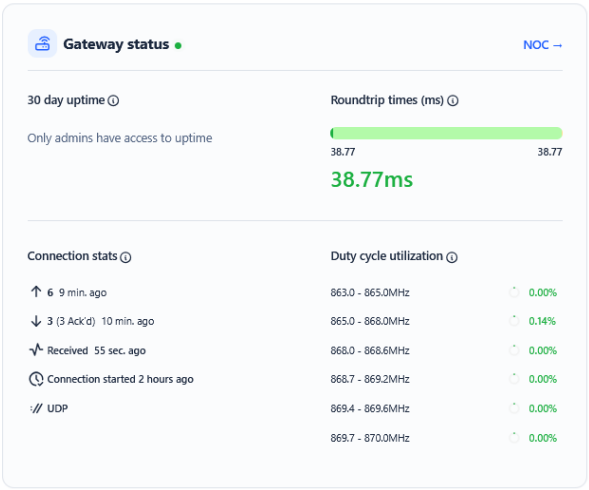
J'ai saisi l'adresse du serveur TTN pour la région Europe, à savoir `eu1.cloud.thethings.network`, ce qui permet à la passerelle de se connecter au bon cluster.

Les ports 1700 ont été renseignés pour la communication montante et descendante. Le port montant permet l'envoi des messages captés par la passerelle vers le serveur TTN, tandis que le port descendant permet de recevoir les messages envoyés depuis le serveur vers les appareils en périphérie.

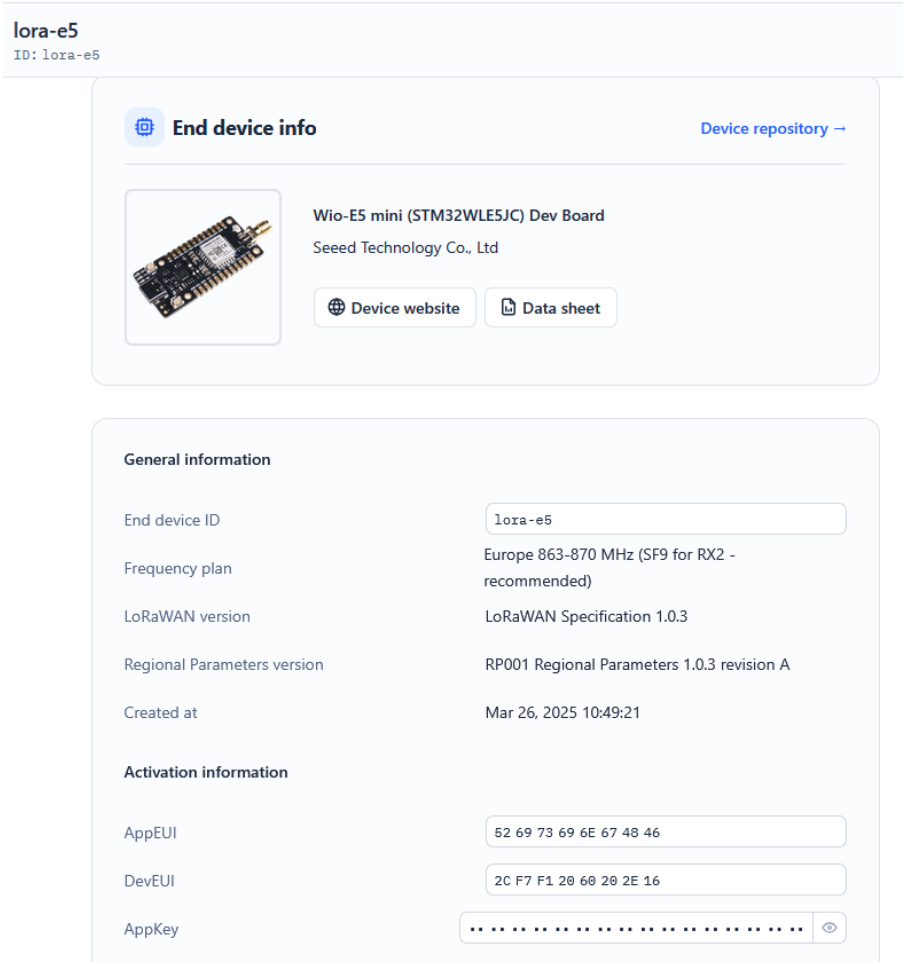
Cette configuration assure ainsi une communication bidirectionnelle correcte entre la passerelle et le réseau TTN.

On peut constater qu'il est correctement configuré et qu'il établit un lien entre la passerelle et TTN.

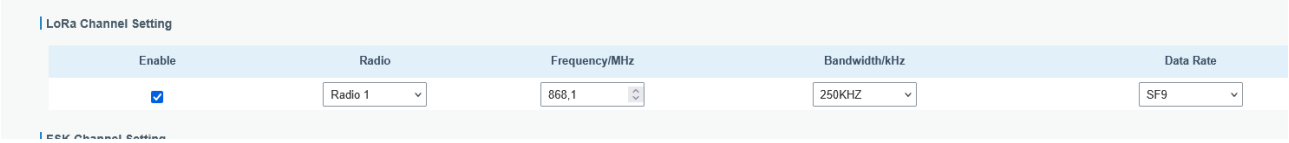




J'ai ajouté le module LoRa sur TTN pour créer une connexion directe entre le module et le réseau TTN.

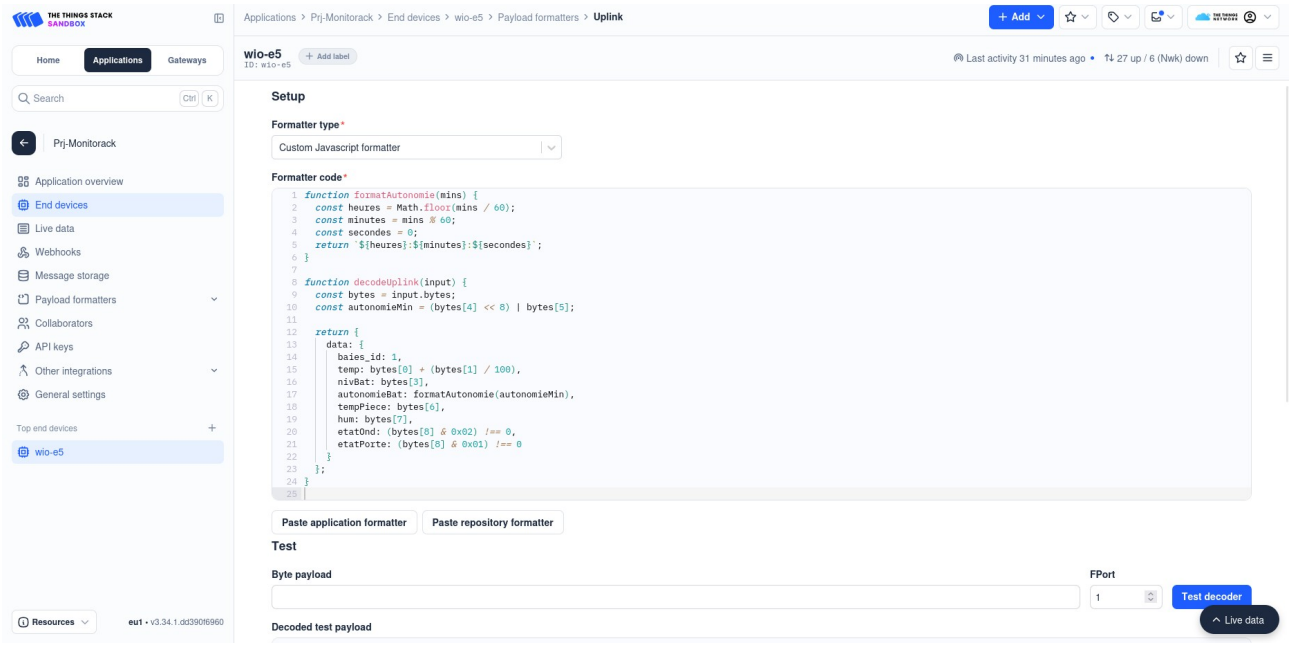


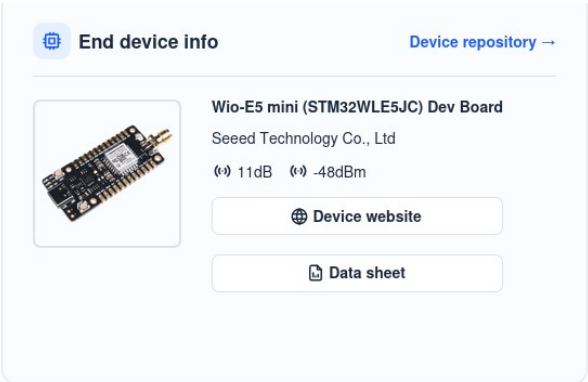
J'ai appliqué la même configuration que celle choisie sur TTN, en veillant à respecter le plan de fréquence spécifié.

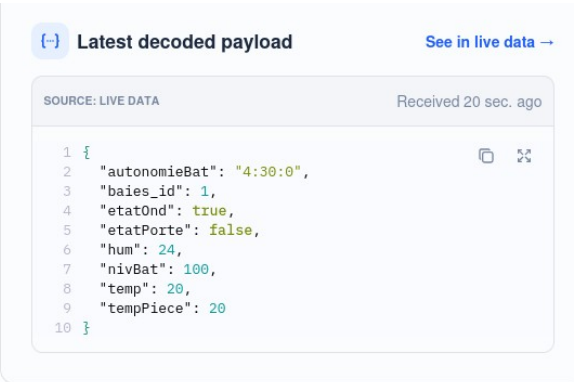


Lors de l'envoi d'une trame, les données sont initialement encodées en hexadécimal. J'ai donc développé un code de formatage permettant de les convertir en valeurs décimales, puis de les structurer au format JSON. Ce format est largement utilisé pour l'échange de données, notamment vers des plateformes IoT, car il est lisible, léger et facilement exploitable par les applications.

Le résultat après l'envoi de la trame depuis le module LoRa vers la passerelle.





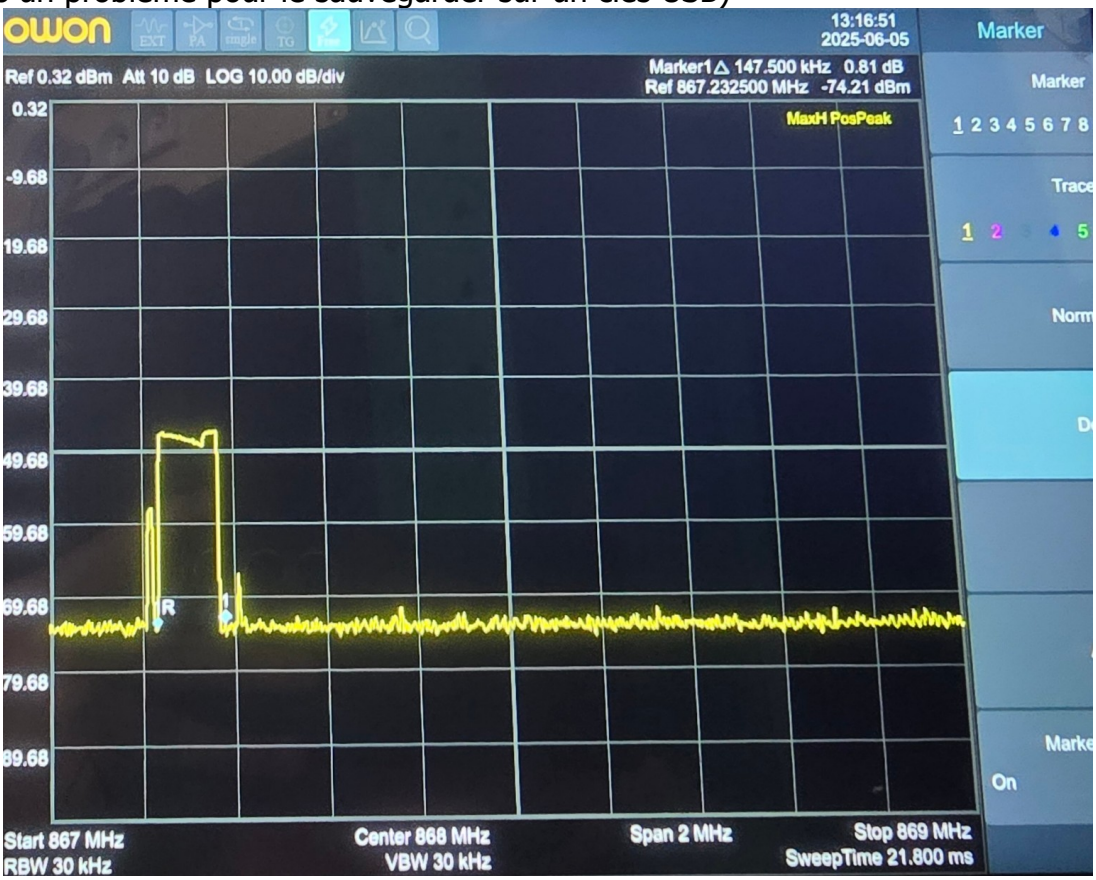


Champ	Octet(s)	Hex	Interprétation	Valeur décodée
temp	0-1	14 00	0x14 + 0x00 / 100	20
hum	7	18	0x18	24
nivBat	3	64	0x64	100
autonomieBat	4-5	01 0E	(0x01 << 8) + 0x0E = 270 min → 4:30:0	4:30:0
etatOnd	8	02	0x02 & 0x02 ≠ 0	true
etatPorte	8	02	0x02 & 0x01 = 0	false
tempPiece	6	14	0x14	20
baies_id	-	-	Statique dans le code	1

Pour l'octet 2 c'est pour l'humidité de batterie donc je l'avais pas envoie avec les autres données vers le serveur.

Afin de vérifier que les données étaient bien envoyées par le module LoRa vers le cloud The Things Network (TTN), une analyse du spectre radio a été réalisée à l'aide d'un analyseur de spectre OWON.

(J'avais un problème pour le sauvegarder sur un clés USB)

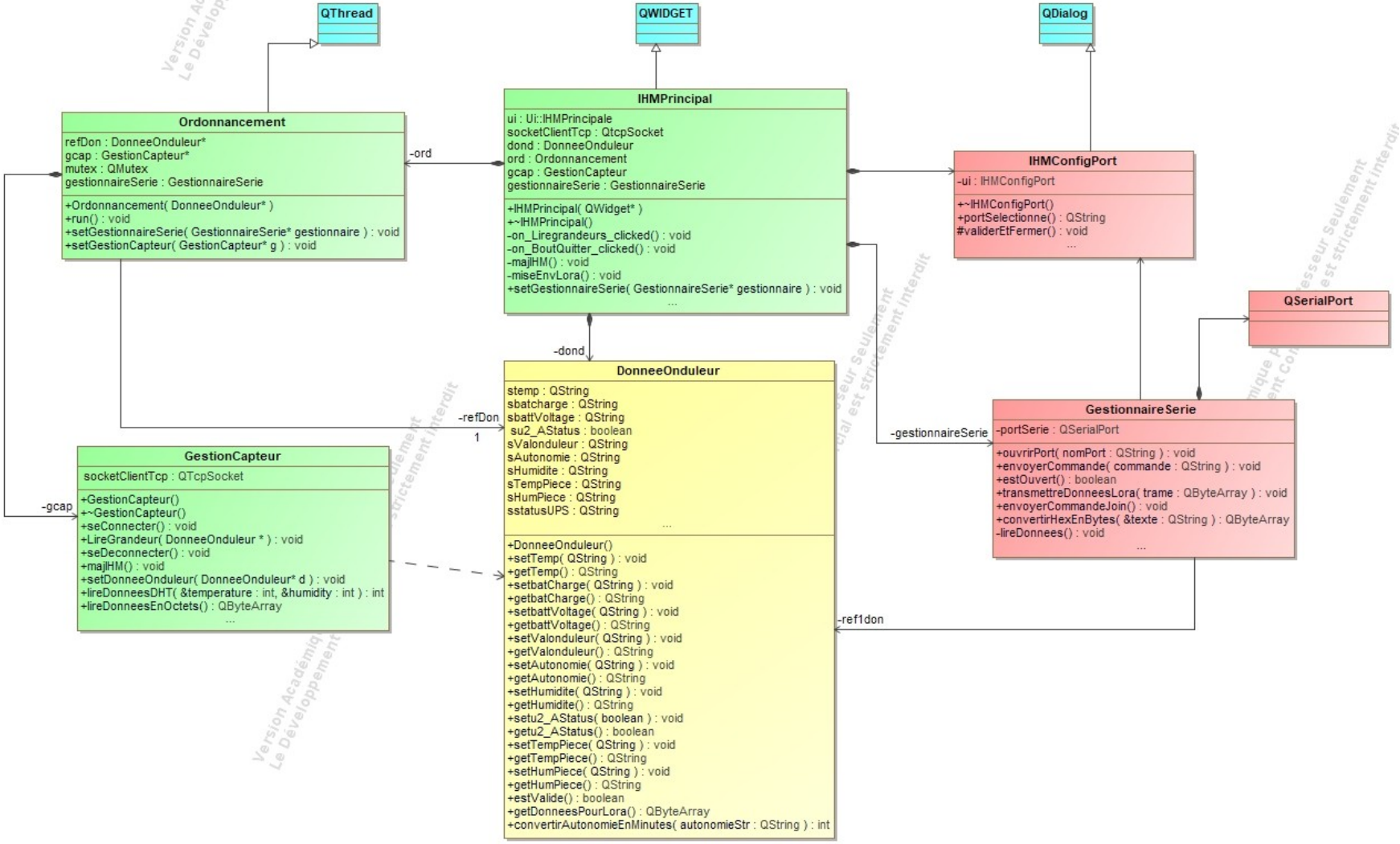


La mesure a été effectuée sur la bande ISM 868 MHz, utilisée pour les communications LoRa. L'analyseur de spectre a été configuré de 867 MHz à 869 MHz. Lors de l'envoi d'une trame, un signal net a été détecté autour de 867,5 MHz, avec une largeur d'environ 500 kHz.

La forme rectangulaire du signal confirme qu'il s'agit d'une émission LoRa structurée. Cela montre que la communication entre le module Wio-E5-LE mini et la passerelle UG65 est bien établie, et que la trame est transmise correctement vers le réseau TTN.

Cette vérification permet de confirmer le bon fonctionnement de l'envoi physique des données jusqu'au cloud.

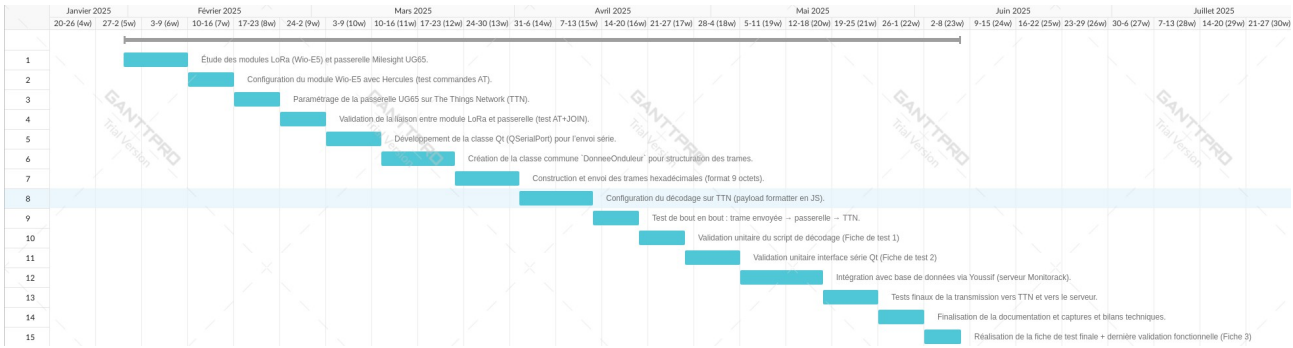
7.3. *Conception détaillée*



7.4. Réalisation et résumé des tests fonctionnels unitaires

Unité tester	Annexes	Validité
Décodage des trames LoRaWAN	Fiche de test 1	OK
Interface de communication LoRa / LoRaWAN (Qt)	Fiche de test 2	OK
Module C++ recevant les données TTN et les envoyant au serveur	Fiche de test 3	OK

7.5. Planning des tâches et jalons



7.6. Bilan personnel technique

Ce projet m'a permis de mettre en pratique les compétences acquises pendant ma formation en BTS CIEL, notamment dans les domaines de la communication série, des protocoles réseau IoT et du développement logiciel avec Qt.

Dans un premier temps, j'ai configuré et mis en service un module LoRa Wio-E5-LE mini, en le connectant à une passerelle Milesight UG65 intégrée au réseau The Things Network (TTN). Cette étape m'a permis de mieux comprendre le fonctionnement du protocole LoRaWAN, en particulier le mode d'activation OTAA et les différents paramètres nécessaires à l'appairage et à la transmission.

Côté logiciel, j'ai développé une classe en C++ avec Qt Creator permettant d'envoyer des trames de données via une liaison série. L'envoi s'effectue en utilisant les commandes AT du module, et la trame contient des données telles que la température de la batterie, son autonomie, le taux d'humidité, et l'état de l'onduleur. Ces données sont regroupées dans un format optimisé (9 octets) avant d'être envoyées au réseau.

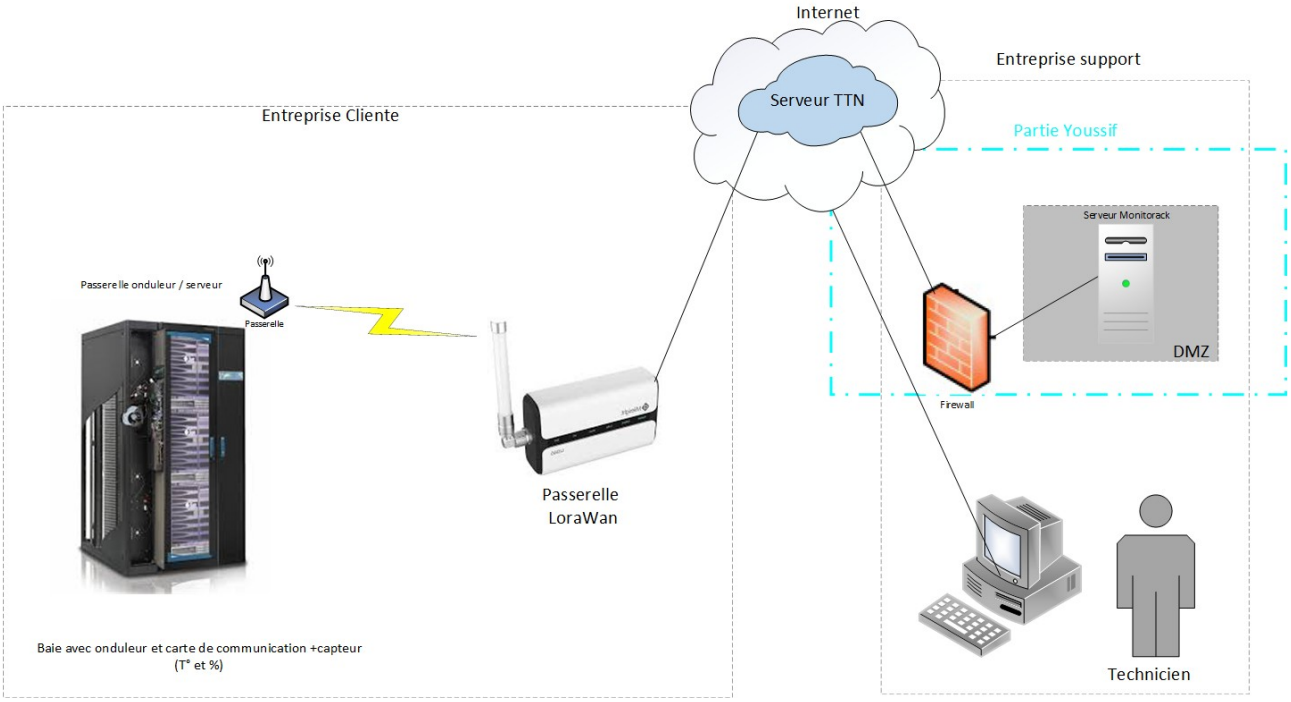
J'ai également travaillé sur le décodage des trames dans l'interface TTN, afin de vérifier que les données étaient bien reçues et interprétées correctement. Ce test m'a permis de valider l'ensemble de la chaîne de communication, depuis la récupération des grandeurs physiques jusqu'à leur affichage dans le cloud.

Le travail en équipe a été essentiel pour assurer la cohérence entre les différentes parties du projet. J'ai collaboré avec Noah pour récupérer les données issues des capteurs, et avec Youssif pour transmettre les trames décodées au serveur Monitorack.

Ces expériences m'ont aidé à renforcer mes compétences en développement embarqué, en communication réseau, et en gestion de projet technique.

8 PARTIE ÉTUDIANT [MATTI YOUSIF Youssif]

8.1. Description de la partie personnelle



Mes tâches sont :

- Prendre en charge l'intégration et la gestion du BackEnd.
- Récupération de la trame envoyée par Raad depuis le serveur TTN, contenant toutes les grandeurs nécessaires (la température, le taux humidité, le niveau de la batterie, l'autonomie de la batterie, l'état de la porte, l'état du l'onduleur).
- Stockage des grandeurs dans la base de données.
- Utilisation d'un ORM pour extraire les données nécessaires.
- Créer des variables qui permettront l'affichage de données.
- Créer le pare-feu en collaboration avec le groupe du projet DataLogger.
- Travailler spécifiquement avec Alexis Grimonpon (DataLogger) et Esteban DUFAU (Monitorack)
- Assurer la configuration et la gestion de la DMZ.

8.2. Mise en œuvre des matériels et technologies employée

Dans le cadre de la mise en œuvre du backend du serveur MONITORACK, Tout d'abord, j'ai créé une machine virtuelle *ubuntu-22.04* sur VirtualBox pour réaliser ma partie de ce projet. J'ai ensuite utilisé Symfony et Doctrine. Symfony est utilisé pour coder la partie backend, tandis que Doctrine sert à gérer la base de données.

Voici les grandeurs manipulées :

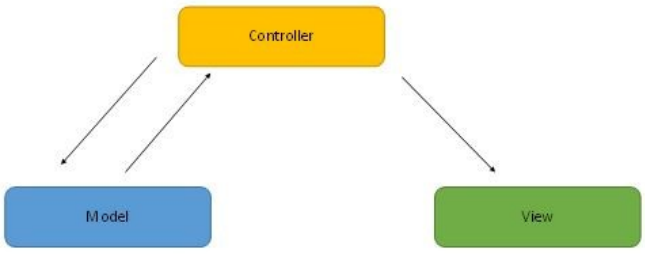
La température → capteur DHT11 (°C)
Le taux humidité → capteur DHT11 (%)
Le niveau de la batterie → la carte AP9631 (%)
L'autonomie de la batterie → la carte AP9631 (time)
L'état de la porte → capteur de contact (bool)
L'état du l'onduleur → la carte AP9631 (bool)

Que est ce que Symfony ?

Symfony est un framework PHP « *open source* » utilisé par les développeurs pour créer des sites ou applications Web complexes, robustes, fiables, évolutifs, maintenables et performants. Egalement Symfony utilise un motif d'architecture logicielle MVC.

L'architecture Modèle/Vue/Contrôleur (MVC) est une façon d'organiser une interface graphique d'un programme. Elle consiste à distinguer trois entités distinctes qui sont, le modèle, la vue et le contrôleur ayant chacun un rôle précis dans l'interface.

Voici le schéma de la MVC :



Le Modèle-Vue-Contrôleur (MVC) est une architecture logicielle qui organise les applications en trois couches distinctes :

- Modèle : Gère les données, la logique métier et l'interaction avec la base de données.
- Vue : Affiche les données et offre une interface utilisateur.
- Contrôleur : Sert d'intermédiaire, traite les actions utilisateur, interagit avec le modèle et met à jour la vue.

Cette structure améliore la clarté, facilite la maintenance et permet une meilleure réutilisation du code.

Pourquoi Symfony ?

Symfony a été utilisé dans le cadre de mon projet pour développer un backend sécurisé et structuré, avec des clean URLs, afin de recevoir, stocker et consulter les grandeurs (la température, le taux humidité, le niveau de la batterie, l'autonomie de la batterie, l'état de la porte, l'état du l'onduleur) transmises par la passerelle via le réseau LoRaWAN de TTN, et les enregistrer dans une base de données.

Exemple d'un clean URL :

On peut voir que cette clean URL permet d'accéder directement aux données de la baie n°1, de manière claire, tant pour l'utilisateur que pour le développeur, sans afficher de paramètres inutiles.

Exemple d'une route complexe sur symfony :
D'abord nous allons voir le fichier du controller pour construire la route.

MonitorackController.php :

```
/***************Une baie*****************/
    #[Route('/baie/{id}', name: 'baie', methods:['GET'])] //La route de cette page est /baie/{id} et la méthode GET permet de juste récupérer des informations sur cette page

    public function baie(int $id, GrandeursRepository $grandeursRepo ): Response
    {
        $grandeurs = $grandeursRepo→findGrandeursByBaieId($id); // Récupère les grandeurs associées à l'id de la baie via une méthode personnalisée.
        return $this->render('monitorack/uneBaie.html.twig', ['baie' => $grandeurs]);  // Rend la vue Twig avec les grandeurs passées sous le nom "baie".
    }
```

Cette route est une route complexe qui permet d'afficher les grandeurs associées à une baie spécifique, identifiée par son id. Elle utilise le fichier uneBaie.html.twig situé dans le dossier monitorack pour afficher la vue correspondante. Ce fichier doit exister pour que les données soient correctement affichées à l'écran. Nous verrons plus bas le résultat de cette route ainsi que les méthodes utilisées. .

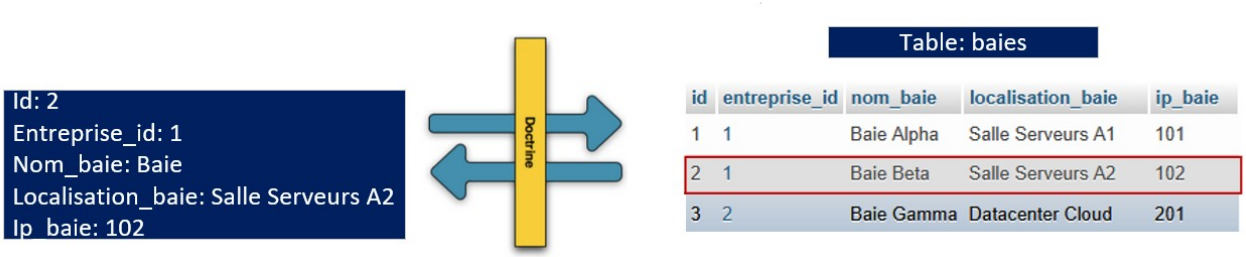
Voici une liste de toutes les routes et leurs intérêts :

Route	Description	Les données retournées
/	Cette route permet d'afficher la page d'accueil (le dashboard)	Elle retourne les informations générales du système comme les stats ou les alertes.
/entreprise/{id}	Cette route permet d'afficher une entreprise uniquement en utilisant son Id	On obtient les infos de l'entreprise : nom, adresse, et les baies qu'elle possède.
/baies	Cette route permet d'afficher toutes les baies	Elle retourne une liste avec les noms, lieux et états des différentes baies.
/baie/{id}	Cette route permet d'afficher une baie uniquement en utilisant son Id	On récupère les grandeurs (comme température, humidité…) liées à une baie spécifique.
/api/trame	Cette route permet de recevoir et stocker des données envoyées par TTN	Elle ne retourne rien de visible, elle sert juste à stocker les données envoyées par TTN dans la base de données.
/aides	Cette route permet d'afficher une page d'aide pour les techniciens	Elle affiche des explications, conseils ou procédures pour aider les techniciens.

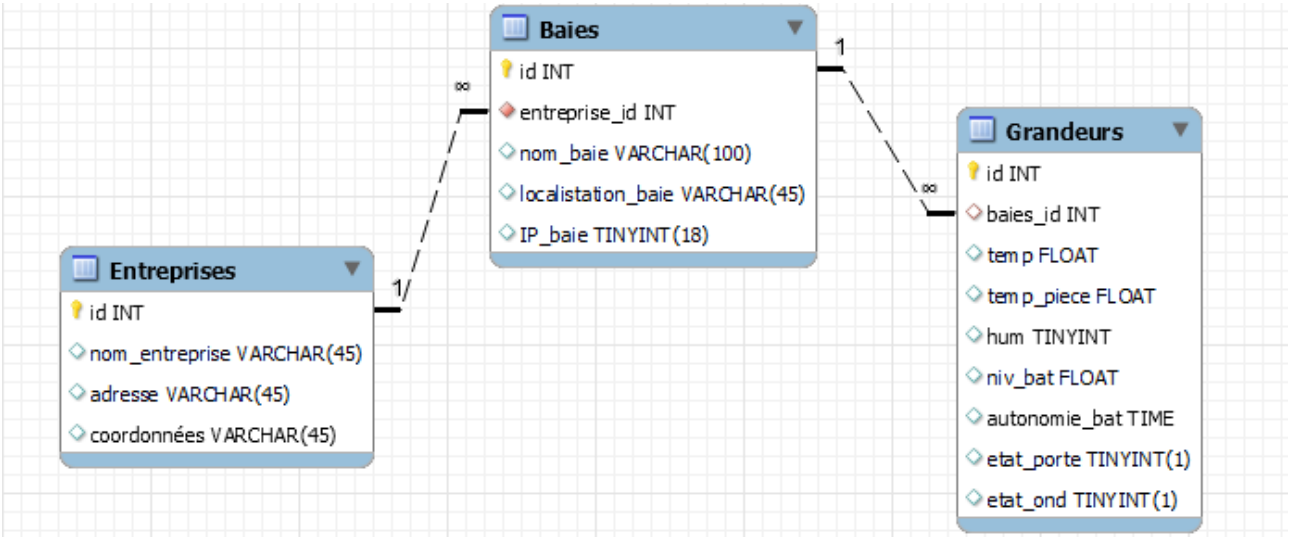
Que est ce que Doctrine?

Doctrine est une librairie PHP conçue pour faciliter la manipulation de ses bases de données et le mapping de ses objets. Il est qualifié d'ORM. On utilise ce type de librairie dans la plupart des frameworks PHP. Doctrine est utilisée par défaut par le framework Symfony. (Voir l'annexe 2 – partie Youssif – pour l'installation.

Voici un schéma expliquant le fonctionnement de Doctrine :



Voici le structure de ma base de données :



Nous avons trois tables : Entreprises, Baies et Grandeurs.
La table Entreprises contient les données des entreprises, la table Baies contient les informations sur les baies, et enfin la table Grandeurs regroupe toutes les grandeurs nécessaires pour informer les techniciens en cas de problème.

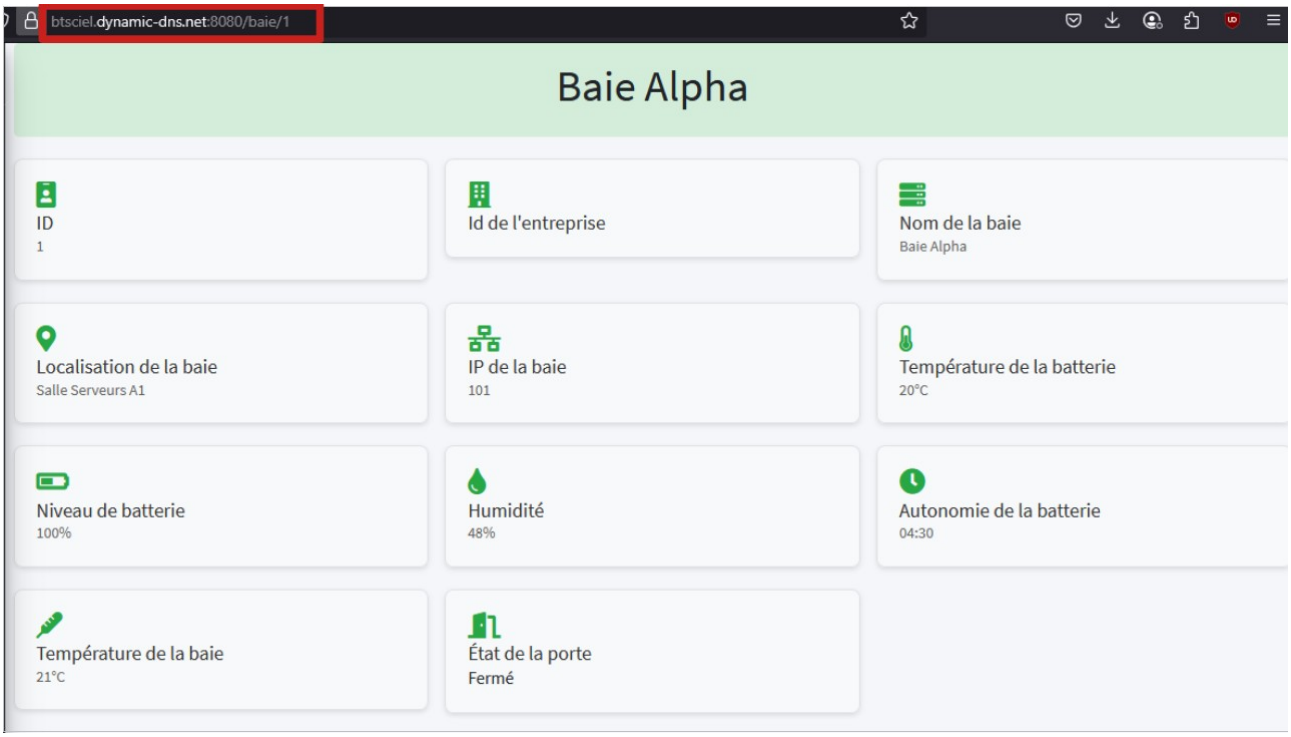
Exemple d'utilisation d'un ORM qui permet de récupérer les grandeurs d'une baie spécifique :

GrandeursRepository.php :

```
<?php
namespace App\Repository;
use App\Entity\Grandeurs; //On utilise l'entity Grandeurs
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

class GrandeursRepository extends ServiceEntityRepository
{

public function findGrandeursByBaieId(int $id): ?array
    {
        $qb = $this->createQueryBuilder('g')  // Créer un QueryBuilder pour l'entité "Grandeur", alias 'g'
            ->select('b.id AS baie_id', 'b.nomBaie', 'b.localisationBaie', 'b.ipBaie',
                'g.id AS grandeur_id', 'g.temp', 'g.tempPiece', 'g.nivBat', 'g.hum',
                'g.autonomieBat', 'g.etatPorte', 'g.etatOnd') //On sélectionne des champs
            ->leftJoin('g.baies', 'b') // La jointure entre les deux tables
            ->where('b.id = :id') // Filtration du résultat pour garder que ceux qui correspondant à l'Id de la baie
            ->setParameter('id', $id) // Filtration du résultat pour garder que ceux qui correspondant à l'Id de la baie
            ->orderBy('g.id', 'DESC') // Ordre décroissant
            →setMaxResults(1); // Limiter le résultat

        return $qb→getQuery()→getOneOrNullResult(); // Retourne le résultat sous forme d'un tableau associatif
    }

}
```

Ce bout de code se trouve dans le fichier GrandeursRepository.php, dans le dossier src/Repository. On utilise le repository car il contient des méthodes pour faire des requêtes personnalisées sur une entité.

Voici le résultat :



Le cadre rouge montre notre route complexe, et à l'intérieur on voit toutes les grandeurs récupérées pour cette baie. On peut voir qu'on a affiché toutes les grandeurs demandées.

Ensuite pendant ce projet, j'avais une machine virtuelle qui tournait seule. Pour sécuriser l'accès, on (avec le groupe DataLogger) a mis en place une DMZ et un pare-feu avec Pfsense. L'objectif était d'éviter que la VM soit directement exposée au réseau principal et de mieux contrôler les connexions entrantes et sortantes.

Qu'est-ce qu'un pare-feu sous Pfsense ?

Un pare-feu sous Pfsense est un outil de sécurité qui contrôle le trafic réseau en filtrant les communications entre différentes parties du réseau selon des règles configurées. Il protège le réseau des accès non autorisés et des menaces, tout en permettant de gérer précisément les services et connexions autorisés. Grâce à Pfsense, on dispose d'une solution flexible et puissante pour sécuriser et superviser le réseau de manière efficace.

Utilité dans notre système?

L'utilisation de PfSense dans notre système est essentielle pour assurer la sécurité du réseau. Il permet de contrôler le trafic entrant et sortant en appliquant des règles strictes,

empêchant ainsi les accès non autorisés et les menaces extérieures. Grâce à PfSense, nous bénéficions d'une solution flexible, puissante et personnalisable, idéale pour surveiller, filtrer et protéger efficacement notre infrastructure réseau. C'est un composant clé dans l'architecture de sécurité du système.

Installation du Pfsense :

Tout d'abord nous avons commencer par installer Pfsense sur un serveur, pendant l'installation du Pfsense nous avons partitionnées des répertoires en utilisant RAID 1 et RAID 5.

Explication RAID :

RAID 1 (mirroring) : Les données sont copiées à l'identique sur deux disques. Si un disque tombe en panne, l'autre a toujours les données.
RAID 5 : Combine la sécurité et la performance, avec une technique de parité pour reconstruire les données si un disque tombe en panne.

Quelle recommandation ?

Le choix d'utiliser RAID 1 et RAID 5 lors de l'installation de PfSense s'inscrit dans une démarche de sécurisation et de continuité de service, conformément aux recommandations de l'ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information).

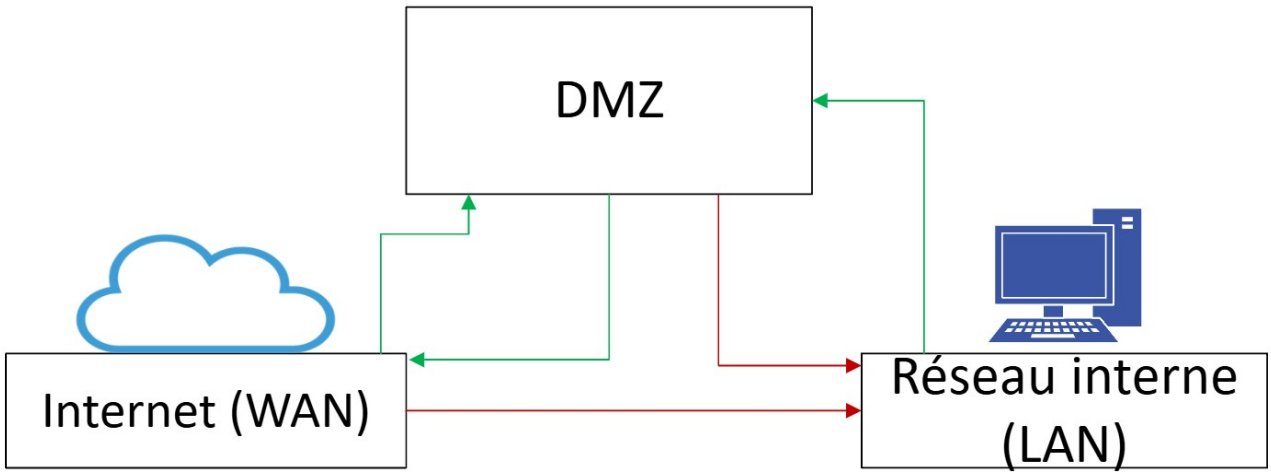
L'ANSSI recommande d'utiliser des mécanismes de redondance pour éviter la perte de données en cas de panne matérielle. RAID 1 (mirroring) permet une tolérance immédiate aux pannes, car les données sont copiées à l'identique sur deux disques. En cas de défaillance d'un disque, le second prend immédiatement le relais. RAID 5, quant à lui, permet un compromis entre performance et sécurité, en utilisant une méthode de parité pour restaurer les données si un disque tombe en panne.

Plus bas, vous verrez les règles que nous avons mises en place pour pfSense et la DMZ.

Présentation d'une DMZ

Une DMZ (Demilitarized Zone), ou zone démilitarisée, est un sous-réseau isolé utilisé en cybersécurité pour ajouter une couche de protection entre un réseau interne et Internet. Elle permet d'héberger des services accessibles depuis l'extérieur (comme un site web, un serveur mail, etc...) tout en réduisant les risques pour le réseau interne.

Voici un schéma expliquant les règles d'un pfSense et d'une DMZ :



Accès autorisés :

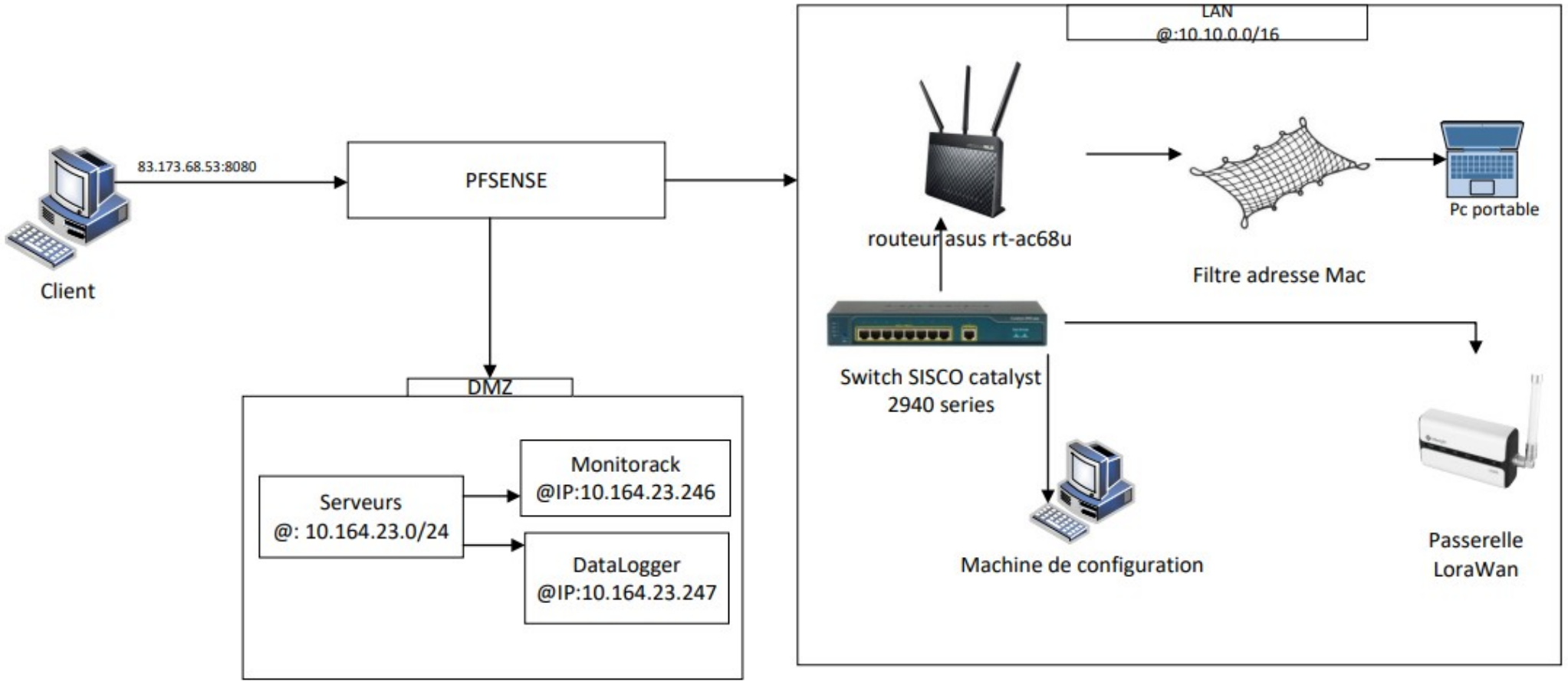
Internet → DMZ (pour accéder à un site web public)
DMZ → Internet (permet les mises à jour des serveurs ou la réception de données depuis des services comme TTN (The Things Network)).
Réseau interne → DMZ (pour administrer les serveurs de la DMZ ou consulter des logs).

Accès interdits :

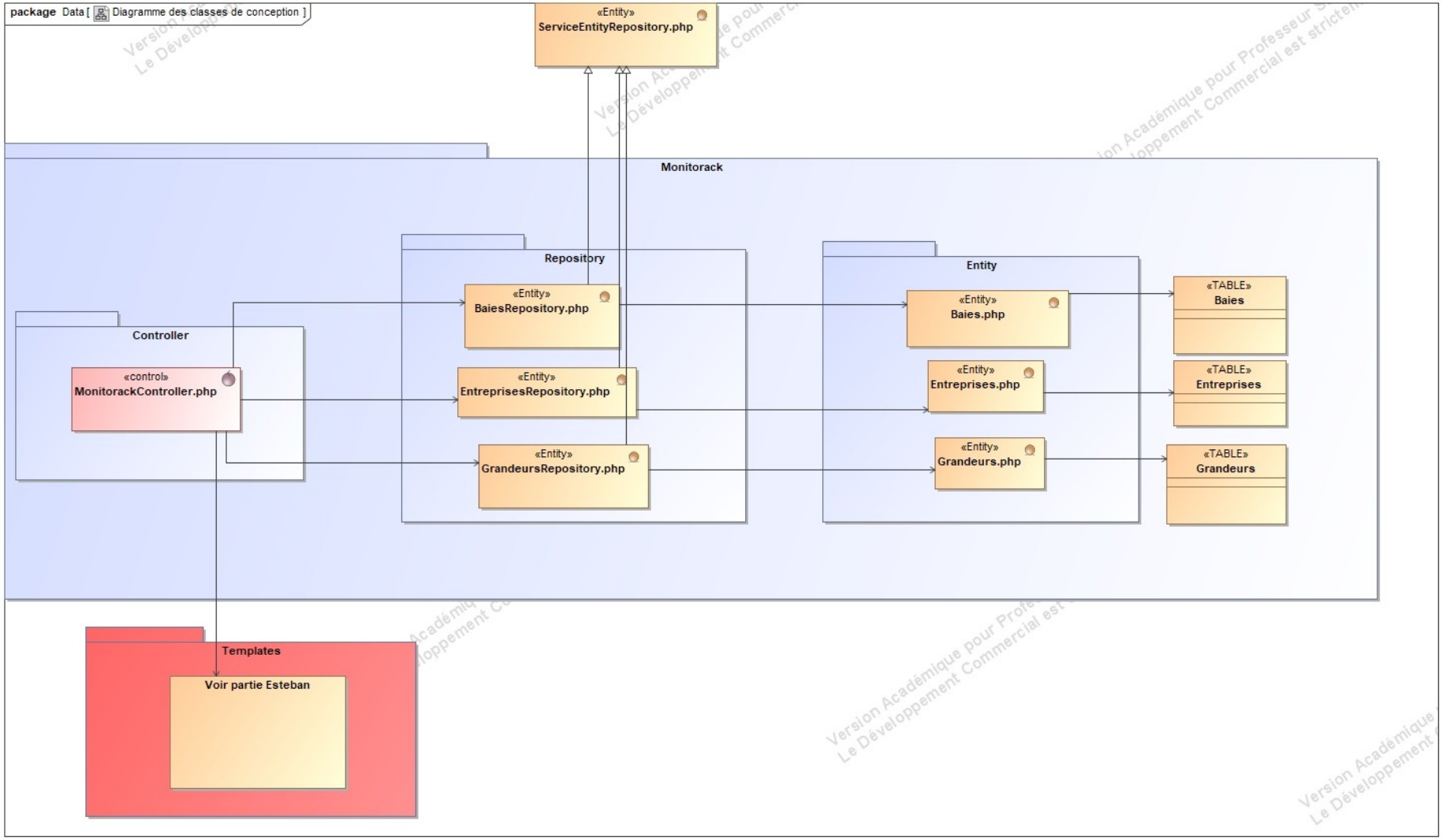
DMZ → Réseau interne (Pour éviter que les pirates atteignent le réseau interne)
Internet → Réseau interne : strictement interdit.

Ces règles permettent de séparer les zones du réseau selon leur niveau de confiance, d'autoriser les échanges nécessaires au fonctionnement (Internet, DMZ, LAN), tout en protégeant le réseau interne contre les accès non autorisés, conformément aux bonnes pratiques de sécurité.

Voici le schéma de l'infrastructure finale de notre réseau :



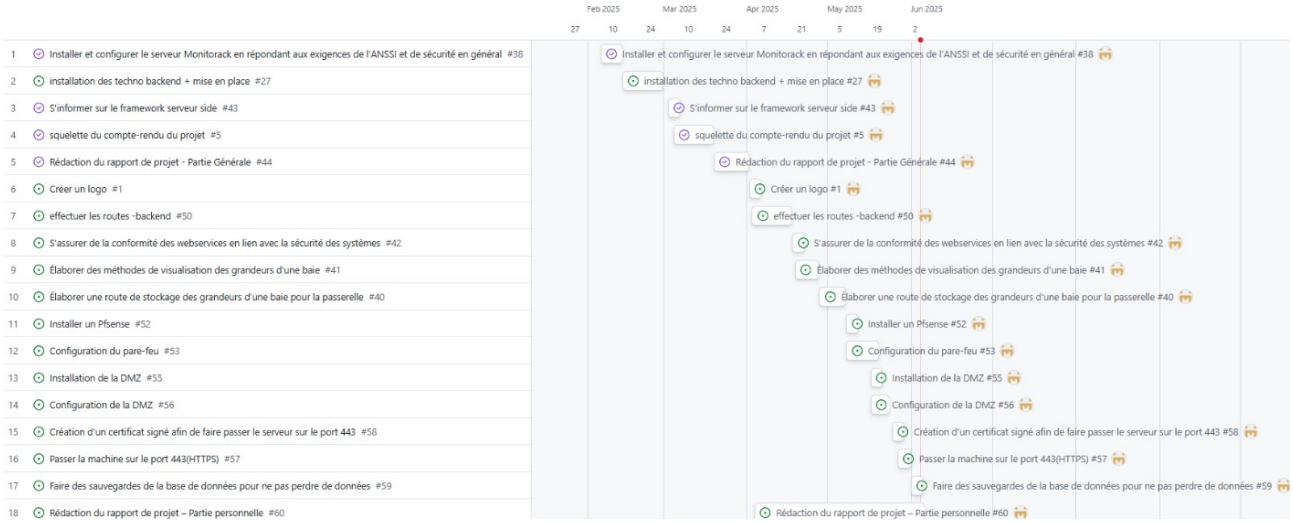
8.3. Conception détaillée



8.4. Réalisation et résumé des tests fonctionnels unitaires

Intitulé du test unitaire	Unité testée	Numéro de la fiche de validation
Test des routes	**UneBaie.html.twig**	**FICHE DE TEST FONCTIONNEL UNITAIRE N°1 (partie youssif)**
Test des routes	**MonitorackController.php**	**FICHE DE TEST FONCTIONNEL UNITAIRE N°2 (partie youssif)**

8.5. Planning des tâches et jalons



8.6. Bilan personnel technique

Pendant ce projet, j'ai appris et utilisé plusieurs compétences importantes dans le domaine de l'informatique.

J'étais responsable de toute la partie backend. J'ai utilisé le framework Symfony, un outil qui permet de mieux organiser le code. Grâce à Symfony, j'ai pu créer des pages claires, appelées routes, pour afficher les informations des baies. J'ai aussi utilisé Doctrine, qui m'a aidé à communiquer facilement avec la base de données et à récupérer les informations comme la température, l'humidité ou encore l'état des équipements.
Pour faire tout cela, j'ai installé une machine virtuelle sous Ubuntu (un système Linux) afin d'avoir un environnement de travail stable et indépendant.

En plus de la programmation, j'ai aussi travaillé sur la sécurité du réseau. Avec le groupe DataLogger, nous avons mis en place une DMZ et un pare-feu avec PfSense. Le pare-feu sert à contrôler les connexions : il bloque celles qui sont dangereuses et autorise celles qui sont utiles. Cela permet d'éviter que la machine soit attaquée depuis Internet.

J'ai aussi beaucoup travaillé en équipe, surtout avec Alexis Grimonpon et Esteban Dufau. On s'est aidé mutuellement pour que nos parties fonctionnent bien ensemble.

Ce projet m'a permis de mieux comprendre le développement web côté serveur, la gestion de base de données, et la sécurité d'un réseau. J'ai pu mettre en pratique ce que j'ai appris en cours, tout en découvrant de nouvelles choses utiles pour mon futur métier.

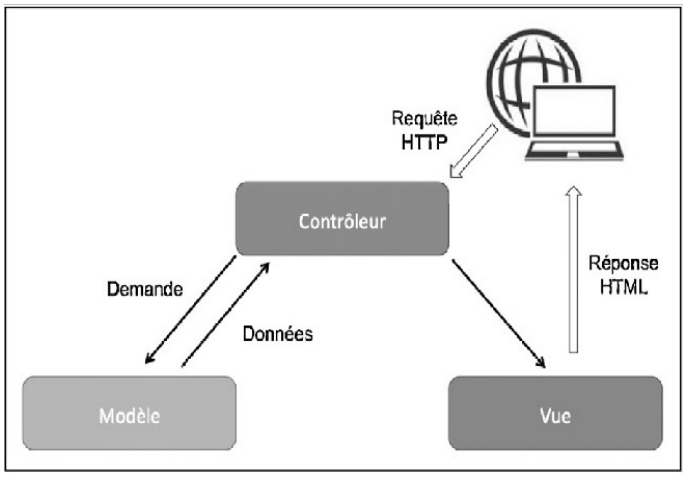
9 PARTIE ÉTUDIANT [DUFAU Esteban]

9.1. Description de la partie personnelle

Dans le cadre du projet MONITORACK, ma mission principale consiste à concevoir et développer une interface web destinée aux techniciens de CI Solutions. Cette interface permet la visualisation en temps réel des grandeurs mesurées dans les baies informatiques (telles que la température, l'humidité ou encore l'état de l'onduleur), ainsi que l'accès à l'historique des données.
Pour cela, j'utilise principalement Symfony 5 pour le backend, en collaboration avec mon binôme Youssif, ainsi que Twig et des bibliothèques JavaScript pour le frontend. Mon travail inclut également la mise en place de modules d'exportation des données et l'intégration de critères de recherche pour une consultation plus efficace.
Cette interface vise à améliorer la gestion, l'exploitation et la maintenance des baies en proposant une solution ergonomique, intuitive et sécurisée.

9.2. Mise en œuvre des matériels et technologies employée

Architecture logicielle : le modèle MVC avec Symfony



Pour structurer l'interface web du projet MONITORACK, j'ai utilisé le framework Symfony 5, qui repose sur le modèle architectural MVC (Modèle – Vue – Contrôleur). Cette architecture permet une claire séparation des responsabilités entre la gestion des données, la logique de traitement et l'affichage, ce qui facilite la maintenance, les évolutions futures et le travail collaboratif.

- Modèle (Model)

Le modèle représente les données de l'application ainsi que les règles métiers associées. Avec Symfony, cela se traduit par les entités Doctrine, qui définissent la structure des objets (par exemple : Baie, Mesure, Grandeur, etc.), et les repositories, qui permettent d'interagir avec la base de données MariaDB.

Vue (View)

La vue correspond à la partie visible de l'application pour l'utilisateur.
Symfony utilise le moteur de template Twig pour générer dynamiquement des pages HTML à partir des données transmises par le contrôleur. Les vues sont conçues pour être claires, ergonomiques et adaptées à une utilisation sur poste client.

- Contrôleur (Controller)

Le contrôleur fait l'interface entre l'utilisateur, les données et l'affichage.
Il reçoit les requêtes HTTP, utilise les services et modèles nécessaires pour effectuer les traitements, puis transmet les résultats à la vue. Dans Symfony, chaque contrôleur est une classe qui gère des routes spécifiques et orchestre le comportement de l'application.

- Exemple d'application concrète dans MONITORACK :

Lorsqu'un technicien accède à la visualisation des données d'une baie :

1. Une requête HTTP est envoyée à une route définie dans un contrôleur Symfony.
2. Le contrôleur interroge les entités via les repositories pour récupérer les mesures liées à la baie concernée.
3. Les données sont ensuite injectées dans une vue Twig, qui les affiche dans l'interface utilisateur.

Moteur de templates : Twig

Pour la gestion de l'affichage côté client dans l'interface web du projet MONITORACK, j'ai utilisé Twig, le moteur de templates intégré à Symfony. Twig joue un rôle central dans la couche Vue (V) du modèle MVC, en permettant de générer dynamiquement des pages HTML en fonction des données fournies par les contrôleurs.
Présentation de Twig
Twig est un moteur de templates moderne, rapide et sécurisé, conçu pour séparer la logique de présentation du reste de l'application. Il permet de créer des vues claires et maintenables, tout en évitant d'intégrer du code PHP brut dans les fichiers HTML.
Avantages de Twig dans le projet

- Lisibilité : syntaxe simple et expressive, plus propre que le HTML avec PHP natif.
- Réutilisabilité : possibilité d'utiliser des layouts (gabarits) et des blocs pour éviter la duplication de code.
- Sécurité : les variables sont automatiquement échappées (protection contre les injections HTML ou JavaScript).
- Modularité : possibilité de découper les vues en composants partiels (`include`, `embed`, `macro`, etc.).

Utilisation de Twig dans MONITORACK
Dans l'interface web MONITORACK, Twig est utilisé pour :

- Afficher dynamiquement les valeurs des capteurs (température, humidité, état de l'onduleur, etc.).
- Générer des listes de baies avec des critères de recherche.
- Présenter l'historique des données sous forme de tableaux ou de graphiques (en interaction avec JavaScript).
- Organiser l'interface avec une structure cohérente et réutilisable via un template principal (layout) étendu par les autres pages.

Exemple concret :

```
<h1>Données de la baie {{ baie.nom }}</h1>

<ul>
  {% for mesure in mesures %}
    <li>{{ mesure.date|date('d/m/Y H:i') }} – Température :
{{ mesure.temperature }} °C</li>
  {% else %}
    <li>Aucune donnée disponible.</li>
  {% endfor %}
</ul>
```

9.3.1. Exemple de contrôleur Symfony

```
// src/Controller/BaieController.php
#[Route('/baie/{id}', name: 'baie_show')]
public function show(Baie $baie, BaieRepository $repo): Response
{
    $mesures = $repo->findMesuresByBaie($baie->getId());
    return $this->render('baie/show.html.twig', [
        'baie' => $baie,
        'mesures' => $mesures,
    ]);
}
```

Ce contrôleur gère l'affichage des mesures pour une baie.

Utilisation de Node.js

Dans le cadre du projet MONITORACK, Node.js a été utilisé pour permettre l'actualisation automatique des données affichées sur l'interface web dès qu'une nouvelle mesure est disponible en base de données.
Présentation de Node.js
Node.js est un environnement d'exécution JavaScript côté serveur. Il est conçu pour

fonctionner en temps réel, en exploitant un modèle événementiel. Cela permet de créer un serveur WebSocket capable de pousser des données vers le client dès qu'un événement (comme l'insertion d'une nouvelle valeur) survient.

Exemple de code Node.js utilisé :

```
javascript
CopierModifier
const WebSocket = require('ws');
const wss = new WebSocket.Server({ port: 8080 });

wss.on('connection', function connection(ws) {
  console.log('Client connecté');

  // Envoi d'un message au client lorsqu'une nouvelle donnée est détectée
  setInterval(() => {
    const nouvelleDonnee = getDerniereDonnee(); // simulation d'une requête
BDD
    ws.send(JSON.stringify(nouvelleDonnee));
  }, 5000); // vérifie toutes les 5 secondes
});

function getDerniereDonnee() {
  return {
    temperature: 28.2,
    humidite: 48,
    timestamp: new Date().toLocaleString()
  };
}
```

Utilisation de JavaScript

Pour rendre l'interface plus réactive, JavaScript est utilisé côté client pour écouter les messages envoyés par le serveur Node.js (WebSocket) et mettre à jour dynamiquement les valeurs affichées sans recharger la page.

Exemple de code JavaScript client :

```
html
CopierModifier
<script>
  const socket = new WebSocket('ws://localhost:8080');

  socket.onmessage = function(event) {
    const data = JSON.parse(event.data);
    document.getElementById("temp").innerText = data.temperature + " °C";
    document.getElementById("hum").innerText = data.humidite + " %";
    document.getElementById("lastUpdate").innerText = "Mis à jour : " +
data.timestamp;
  };
</script>
```

Interface graphique : AdminLTE

L'interface web MONITORACK repose sur le thème AdminLTE, qui fournit une base solide pour construire des dashboards modernes et responsives. Il facilite l'intégration de composants visuels tels que les cartes d'information, les menus, les alertes ou les graphiques.

Exemple d'intégration d'un composant AdminLTE (boîte de données) :

```
<div class="card card-primary">
  <div class="card-header">
    <h3 class="card-title">Température en temps réel</h3>
  </div>
  <div class="card-body">
    <p>Température actuelle : <span id="temp">--</span> °C</p>
    <p>Humidité actuelle : <span id="hum">--</span> %</p>
  </div>
  <div class="card-footer">
    <small id="lastUpdate">Dernière mise à jour : --</small>
  </div>
</div>
```

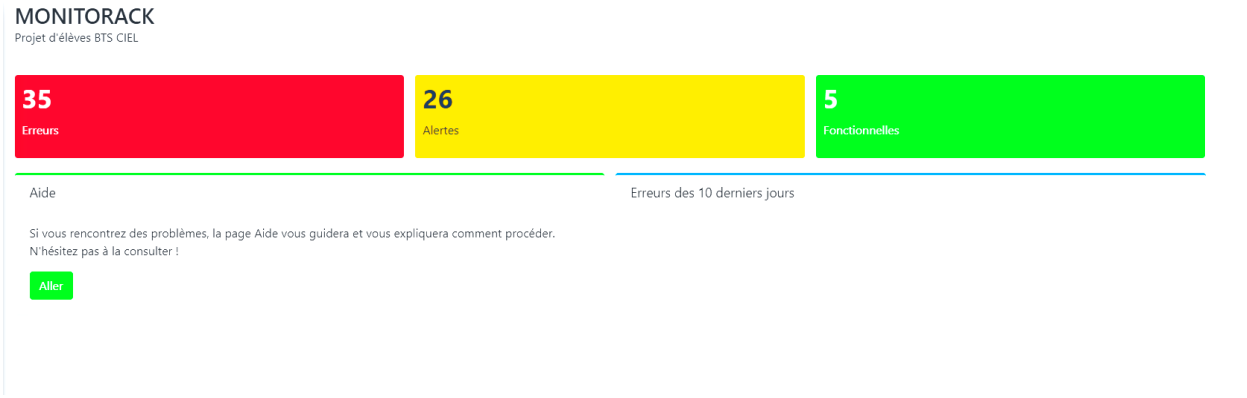
Intégration dans l'environnement Symfony

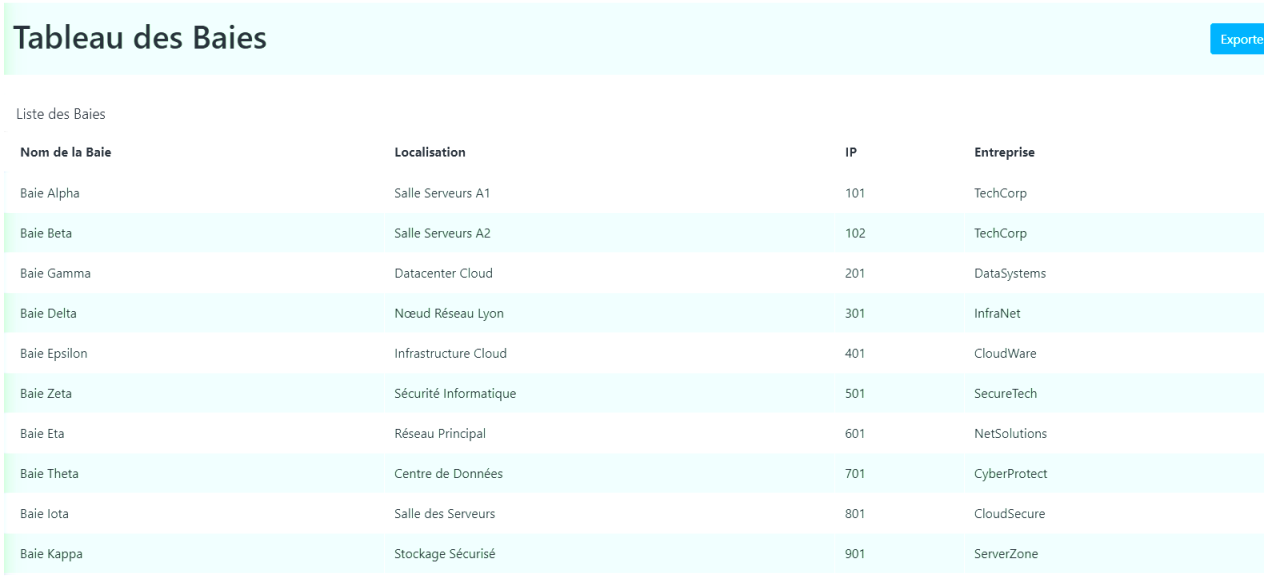
Les classes CSS fournies par AdminLTE sont directement utilisées dans les fichiers **Twig** :

```
{% extends 'base.html.twig' %}

{% block body %}
  <div class="card card-info">
    <div class="card-header">
      <h3 class="card-title">Baie {{ baie.nom }}</h3>
    </div>
    <div class="card-body">
      <p>Température : <span id="temp">--</span></p>
      <p>Humidité : <span id="hum">--</span></p>
    </div>
  </div>
{% endblock %}
```

La conception de l'interface web MONITORACK a suivi une logique modulaire et orientée utilisateur. L'application a été divisée en plusieurs composants principaux :

- **Tableau de bord principal** : affichant les baies surveillées avec un résumé de l'état (icône d'alerte, dernière mesure reçue).





- **Vue détaillée d'une baie** : accessible via un clic sur une ligne du tableau, affichant les données en temps réel et l'historique.

-
- **Filtres de recherche** : permettant de trier les baies selon différents critères (panne, température, site...) (en cour de développement)
- **Module d'export** : pour permettre l'extraction CSV ou PDF des données enregistrées. (en cours de développement)

Chaque vue a été pensée en suivant un wireframe initial validé avec les autres membres du groupe. L'expérience utilisateur (UX) a été soignée pour être intuitive : icônes visuelles, couleurs pour les alertes, disposition adaptée à un usage professionnel.

Choix techniques justifiés

Le choix de **Symfony** s'explique par sa robustesse, sa communauté active et sa parfaite intégration avec Twig pour la gestion des vues. J'ai également envisagé d'autres frameworks comme Laravel ou Flask, mais Symfony offrait une intégration plus naturelle dans notre environnement PHP et MariaDB. Pour le frontend, l'utilisation d'**AdminLTE** nous a permis d'éviter de "réinventer la roue" tout en garantissant une IHM professionnelle. Enfin, **Node.js** a été choisi pour son efficacité à gérer les WebSocket et le réel temps

Architecture réseau sécurisée avec PfSense

Dans le cadre du projet Monitorack, la sécurité et la segmentation du réseau étaient des priorités essentielles. Pour cela, j'ai mis en œuvre une architecture réseau basée sur un pare-feu PfSense, installé sur une machine dédiée, jouant un rôle central dans le contrôle et le routage des communications.

Schéma général

Le réseau est organisé autour de plusieurs sous-réseaux distincts :

- LAN principal (10.10.0.0/16) : utilisé pour la communication interne et l'accès à Internet. Il est connecté à un Switch Cisco Catalyst 2940, garantissant une gestion fiable du trafic local.

- DMZ (Zone Démilitarisée) : accessible depuis l'extérieur (adresse publique 83.173.68.53:8080), cette zone héberge uniquement les services exposés, notamment le serveur Web Monitorack. Cela permet de limiter les risques en cas de tentative d'intrusion.
- Sous-réseau serveur (10.164.23.0/24) : segment isolé contenant le cœur du système de supervision :
    - Le serveur Monitorack (10.164.23.246)
    - Le DataLogger (10.164.23.247)

Ces machines sont protégées derrière PfSense, avec des règles de pare-feu strictes n'autorisant que le trafic nécessaire.

Routeur et filtrage

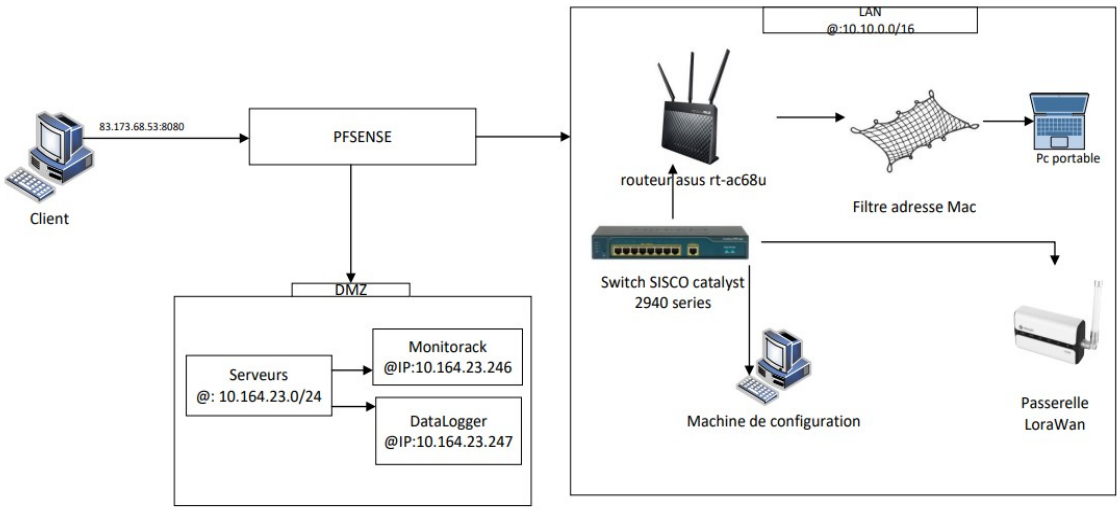
La connexion Internet est assurée par un routeur Asus RT-AC68U, tandis que le PfSense agit en passerelle de sécurité entre les zones. Il permet notamment :

- Le filtrage par adresse MAC sur certains ports critiques
- Le NAT/PAT pour exposer uniquement les services nécessaires
- Une journalisation des accès et une supervision des tentatives de connexion suspectes

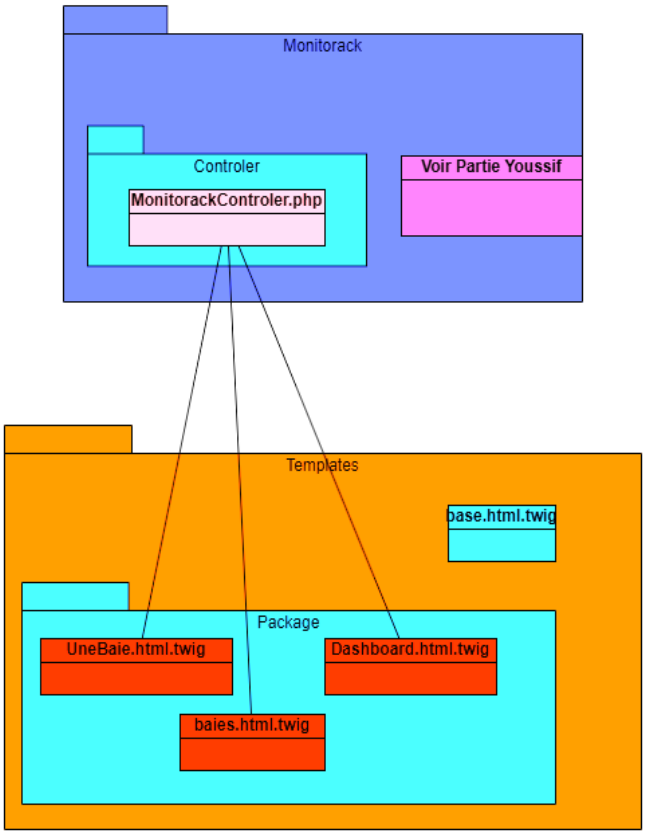
Poste d'administration

Un PC portable d'administration est connecté au LAN. Il permet :

- La configuration initiale et continue du PfSense
- L'accès sécurisé à l'interface web de Monitorack
- Le diagnostic réseau et la maintenance



9.3. Conception détaillée



9.4. Réalisation et résumé des tests fonctionnels unitaires

9.5. Planning des tâches et jalons

9.6. Bilan personnel technique

Ce projet m'a permis de faire un grand bond dans mes compétences techniques. J'ai non seulement consolidé mes bases sur le framework Symfony, mais aussi développé une maîtrise concrète du modèle MVC, du routage, de la gestion d'entités Doctrine et des formulaires complexes.
L'utilisation de Twig a été un levier essentiel pour structurer des interfaces claires, réutilisables et dynamiques. J'ai pris conscience de l'importance de bien structurer les layouts, d'utiliser des blocs pour éviter les répétitions, et de toujours penser à l'accessibilité et la lisibilité des interfaces.
Le développement d'une communication temps réel via WebSocket, en intégrant un serveur Node.js parallèle, m'a permis de découvrir une architecture asynchrone et événementielle, bien différente du modèle classique PHP. Cette partie m'a demandé de m'adapter rapidement et de comprendre des notions plus avancées comme le broadcast, la sérialisation JSON et la réception dynamique côté client en JavaScript.
Côté frontend, j'ai appris à structurer une interface ergonomique avec AdminLTE et à l'intégrer dans Symfony sans casser le MVC. Cela m'a demandé une rigueur dans le respect des classes CSS, la logique responsive, mais aussi dans le rendu multi-navigateurs.
J'ai également travaillé avec des données réelles issues de capteurs, ce qui a apporté une complexité supplémentaire : la gestion d'erreurs, les cas de non-réception, le formatage des dates et le tri temporel. Cela m'a appris à prévoir des cas d'exception, à afficher des messages clairs pour les utilisateurs, et à toujours penser au "scénario du pire".
Enfin, ce projet m'a sensibilisé à l'importance de la documentation (internes, routes, modèles de données), de la clarté du code et du travail collaboratif. Il représente une expérience riche, technique mais aussi humaine, que je valoriserai dans mes projets futurs, en particulier dans la cybersécurité ou le développement fullstack.

10 INTÉGRATION ET RECETTE UTILISATEUR

10.1. Intégration

SHEPHO Raad:

Intitulé du test d'intégration	Étudiants concernés	Modules logiciels concernés	État de l'intégration
Récupérer les valeurs de l'onduleur (carte APC9631) et du capteur DHT11 et les encapsuler dans une trame	Noah	Classe DonneeOnduleur	OK
Transmettre la trame LoRa au serveur via TTN et afficher les données dans Monitorack	Youssif	Script de décodage TTN, API serveur	OK

VIGNEAU Noah :

Intitulé du test d'intégration	Étudiants concernés	Modules logiciels concernés	État de l'intégration
Récupérer les valeurs de l'onduleur et pouvoir les envoyés sur le serveur TTN	Noah	Classe DonneOnduleur	OK
	Raad	Classe DonneeOnduleur/gestion capteur	

10.2. Recette utilisateur

Étape	Action attendue	Résultat attendu	Satisfait
Connexion à l'application	L'utilisateur accède à l'interface Monitorack via http://btsciel.dynamic-dns.net	Affichage du tableau de bord avec les cartes erreurs, alertes, fonctionnelles	✔
Navigation vers la page Baies	L'utilisateur clique sur le lien "Baies" dans le menu	Affichage du tableau des baies avec colonnes Nom, IP, Localisation, Entreprise	✔
Consultation d'une baie	L'utilisateur clique sur une ligne du tableau des baies	Affichage de la fiche complète de la baie avec toutes les données disponibles	✔
Affichage des données capteurs	L'utilisateur visualise température, humidité, niveau de batterie, etc.	Les données sont visibles, à jour ou affichées en "N/A" si indisponibles	✔
Export des données	L'utilisateur effectue un clic droit sur le tableau des baies	Menu contextuel avec options Export CSV, Excel, PDF	X
Accès à l'aide	L'utilisateur clique sur "Aide" ou le bouton présent dans le dashboard	Redirection vers la page d'aide	X

11 CONCLUSION

Le projet MONITORACK a été une expérience très enrichissante pour nous tous. Il nous a permis de mettre en application une grande partie des compétences que nous avons acquises pendant notre formation en BTS CIEL, que ce soit en développement, en réseau, en électronique ou même en cybersécurité.

Au début, le projet paraissait complexe, notamment à cause du nombre d'équipements à intégrer (onduleur, capteurs, passerelle LoRaWAN, serveur web, etc.) et des différents protocoles utilisés comme Telnet, LoRa, TCP/IP, ou encore HTTP. Chacun de nous avait une partie bien précise à réaliser, et il a fallu rapidement apprendre à collaborer efficacement. Cela nous a aussi appris à bien documenter notre travail pour que chaque membre du groupe puisse comprendre et s'intégrer dans le projet global.

Par exemple, nous avons utilisé Qt Creator pour créer une interface locale capable de lire les données de l'onduleur en temps réel, grâce à une connexion Telnet. Ensuite, les données sont envoyées vers une passerelle LoRaWAN configurée manuellement, ce qui nous a permis de découvrir la plateforme TTN (The Things Network). Ces données ont été ensuite réceptionnées par notre serveur hébergé dans une machine virtuelle sous Ubuntu, et stockées dans une base de données via Symfony et Doctrine.

Nous avons également mis en place une interface web ergonomique avec Twig et AdminLTE, pour afficher les informations de manière claire. L'utilisation de Node.js et de WebSockets a aussi été une vraie découverte, surtout pour l'affichage des données en temps réel.

Un autre aspect très formateur a été la sécurité réseau. Grâce à la mise en place d'une DMZ et d'un pare-feu PfSense, nous avons mieux compris comment protéger une infrastructure informatique, notamment en appliquant les recommandations de l'ANSSI (comme l'utilisation de RAID 1 et RAID 5 pour sécuriser les données).

C'est un projet concret, complet et motivant, qui nous a donné un vrai aperçu du métier d'informaticien dans un environnement professionnel. Il nous servira clairement de base pour nos futurs projets et stages.

Nous tenons également à remercier nos professeurs pour leur accompagnement, leurs conseils tout au long du projet, et leur disponibilité à chaque étape. Leur aide nous a permis de mieux comprendre les enjeux techniques et de progresser tout au long de cette aventure.

12 ANNEXES

FICHE D'INTEGRATION		
Projet **Monitorack**	*Nom du module logiciel testé* **Insertion dans la base**	*Date* **01 juin 2025**
Nom du technicien impliqué dans le test : ***Youssif MATTI YOUSIF, SHEPHO RAAD***		
Description du module logiciel : Une fiche qui représente l'intégration de notre système.		

Environnement du test :

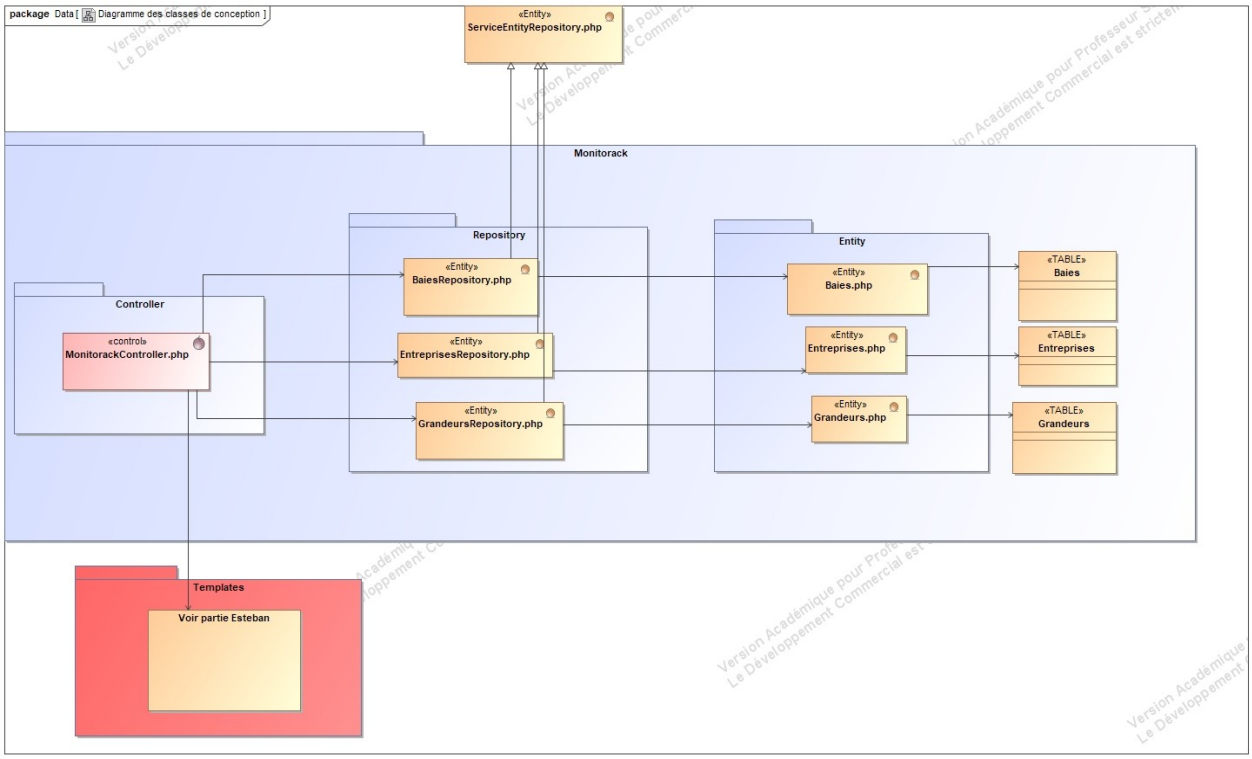
Type de test : □ Unitaire Nominal ■ **Unitaire aux limites**

Objectifs du test :

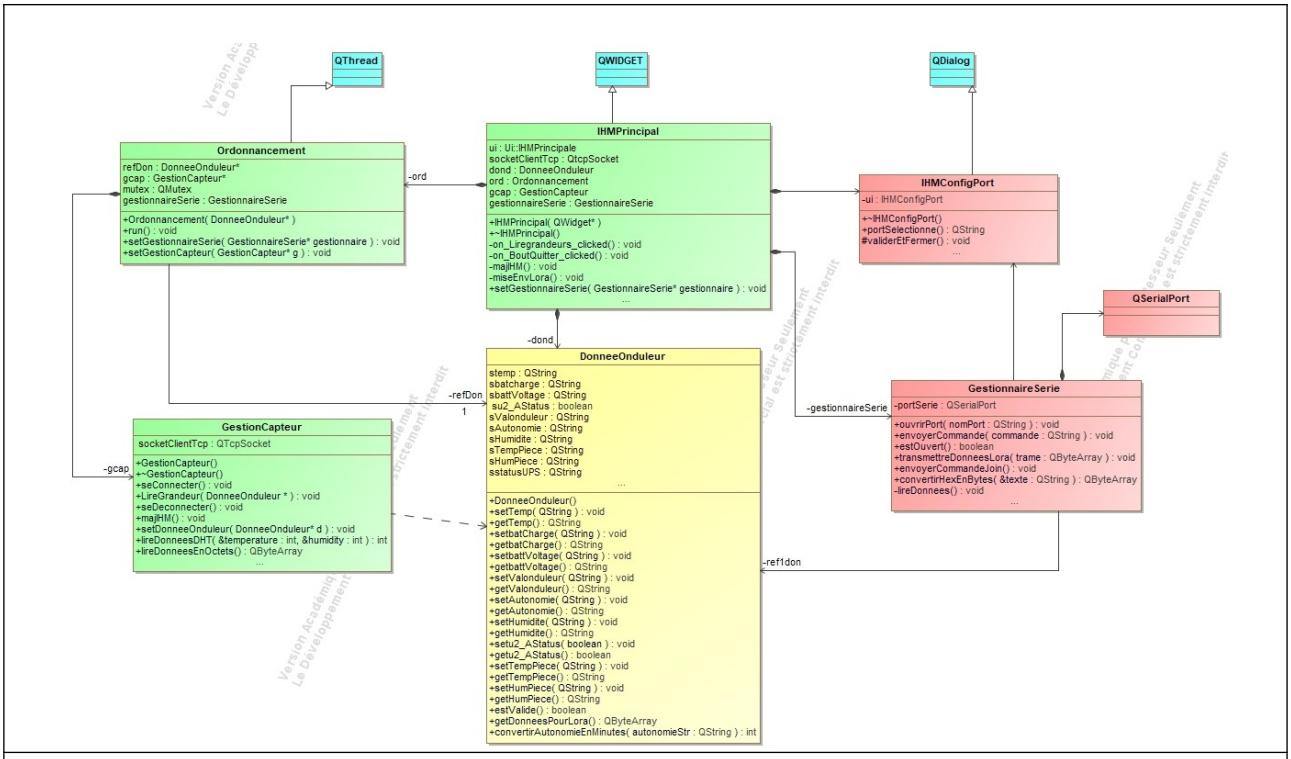
- **Valider la fonctionnement de la route**
- **Valider l'obtention** de **l'ensemble** des **relevés.**
- **Valider le fonctionnement de l'intégration**

Environnement matériel du test (diagrammes de paquets et / ou de classes) :

Backend :



Transmission de données:



donneeonduleur.cpp:

```
int DonneeOnduleur::convertirAutonomieEnMinutes(const QString& autonomieStr) {
 int heures = 0;
 int minutes = 0;

 QRegExp regexHeures("(\\d+)\\s*hr");
 QRegExp regexMinutes("(\\d+)\\s*min");

 if (regexHeures.indexIn(autonomieStr) != -1) {
 heures = regexHeures.cap(1).toInt();
 }
 if (regexMinutes.indexIn(autonomieStr) != -1) {
 minutes = regexMinutes.cap(1).toInt();
 }

 int totalMinutes = heures * 60 + minutes;
 if (totalMinutes > 65535) totalMinutes = 65535; // borne max 16 bits

 return totalMinutes;
}



QByteArray DonneeOnduleur::getDonneesPourLora() {
    QByteArray trame;
```

```
bool ok = false;

float tempF = stemp.trimmed().toFloat(&ok);
if (!ok) tempF = 0;
int tempEntier = static_cast<int>(tempF);
int tempCentieme = static_cast<int>((tempF - tempEntier) * 100);

int hum = sHumidite.trimmed().toFloat(&ok);
if (!ok) hum = 0;

int charge = sbatcharge.trimmed().toFloat(&ok);
if (!ok) charge = 0;

int autoUPS = convertirAutonomieEnMinutes(sAutonomie);

int tempPiece = sTempPiece.trimmed().toFloat(&ok);
if (!ok) tempPiece = 0;

int humPiece = sHumPiece.trimmed().toFloat(&ok);
if (!ok) humPiece = 0;

bool etatOnd = sValonduleur.trimmed().toFloat(&ok) > 0;
char flags = 0;
if (etatOnd) flags |= 0x02;
if (su2_AStatus) flags |= 0x01;

// DEBUG
qDebug() << "[Trame LoRa]"
        << "Temp:" << tempF
        << "Hum:" << hum
        << "Charge:" << charge
        << "Auto (min):" << autoUPS
        << "TempPièce:" << tempPiece
        << "HumPièce:" << humPiece
        << "EtatOnd:" << etatOnd
        << "EtatPorte:" << su2_AStatus;

trame.append(static_cast<unsigned char>(tempEntier));
trame.append(static_cast<unsigned char>(tempCentieme));
trame.append(static_cast<unsigned char>(hum));
trame.append(static_cast<unsigned char>(charge));

trame.append(static_cast<unsigned char>((autoUPS >> 8) & 0xFF));
trame.append(static_cast<unsigned char>(autoUPS & 0xFF));
```

```
    trame.append(static_cast<unsigned char>(tempPiece));
    trame.append(static_cast<unsigned char>(humPiece));
    trame.append(flags);

    return trame;
}

bool DonneeOnduleur::estValide() {
    return !stemp.isEmpty() && !sbattVoltage.isEmpty(); // ou toute autre logique de
validation
}
```

gestioncapteur.cpp:

```
bool GestionnaireSerie::ouvrirPort(const QString &nomPort) {
    if (portSerie.isOpen())
        portSerie.close();

    portSerie.setPortName(nomPort);
    portSerie.setBaudRate(9600);
    portSerie.setDataBits(QSerialPort::Data8);
    portSerie.setParity(QSerialPort::NoParity);
    portSerie.setStopBits(QSerialPort::OneStop);
    portSerie.setFlowControl(QSerialPort::NoFlowControl);
    return portSerie.open(QIODevice::ReadWrite);
}

void GestionnaireSerie::envoyerCommande(const QString &commande) {
    if (portSerie.isOpen()) {
        portSerie.write(commande.toUtf8());
    }
}

void GestionnaireSerie::lireDonnees() {
    QByteArray donnees = portSerie.readAll();
    emit donneesRecues(donnees);
}

bool GestionnaireSerie::estOuvert() const {
    return portSerie.isOpen();
}

void GestionnaireSerie::envoyerCommandeJoin() {
    envoyerCommande("AT+JOIN\r\n");
}
```

```
QByteArray GestionnaireSerie::convertirHexEnBytes(const QString &texte) {
    QByteArray resultat;
    QStringList octets = texte.split(' ', Qt::SkipEmptyParts);
    for (const QString &octet : octets) {
        bool ok;
        char byte = octet.toUInt(&ok, 16);
        if (ok) {
            resultat.append(byte);
        }
    }
    return resultat;
}

void GestionnaireSerie::transmettreDonneesLora(QByteArray trame) {
    if (!estOuvert()) {
        qDebug() << "Port série non ouvert !";
        return;
    }

    if (trame.isEmpty() || trame == QByteArray(trame.size(), 0x00)) {
        qDebug() << "Trame vide détectée, envoi annulé :" << trame.toHex();
        return;
    }

    QString hex = trame.toHex().toUpper();
    QString commande = "AT+MSGHEX=" + hex + "\r\n";

    envoyerCommande(commande);  // Envoi immédiat
    qDebug() << "Commande envoyée :" << commande;
}
```

ihmprincipal.cpp:

```
void IHMPrincipal::miseEnvLora()
{
    if (gestionnaireSerie && dond) {
        if (dond->estValide()) {
            QByteArray trame = dond->getDonneesPourLora();
            qDebug() << "[Envoi LoRa] Trame : " << trame.toHex();
            gestionnaireSerie->transmettreDonneesLora(trame);
        } else {
            qDebug() << "Données non valides, trame non envoyée.";
        }
    }
}
```

```
void IHMPrincipal::setGestionnaireSerie(GestionnaireSerie* gestionnaire) {
    this->gestionnaireSerie = gestionnaire;
    if (ordonnancement)
        ordonnancement->setGestionnaireSerie(gestionnaire);
}

void IHMPrincipal::on_confPort_clicked() {
    qDebug() << "on_confPort_clicked appelé";
    IHMConfigPort configDialog(this);
    if (configDialog.exec() == QDialog::Accepted) {
        QString port = configDialog.portSelectionne();
        qDebug() << "Port sélectionné :" << port;

        if (!gestionnaireSerie) {
            gestionnaireSerie = new GestionnaireSerie(this);
            connect(gestionnaireSerie, &GestionnaireSerie::donneesRecues, this, []
(QByteArray donnees){
                qDebug() << "Données LoRa reçues :" << donnees;
            });
        }
        if (gestionnaireSerie->ouvrirPort(port)) {
            gestionnaireSerie->envoyerCommandeJoin();
        }

    }
}
```

ordonnancement.cpp:

```
void Ordonnancement::run()
{
    if (!gcap || !refDon) {
        qDebug() << "Erreur : gcap ou refDon non initialisé.";
        return;
    }

    gcap->seConnecter();

    while (true) {
        qDebug() << Q_FUNC_INFO;

        gcap->lireGrandeur(refDon);

        emit miseAJIHM();
        emit miseEnvLora();
```

```
        QThread::sleep(10);  // Pause avant la prochaine lecture
    }

    gcap->seDeconnecter();
}


void Ordonnancement::setGestionnaireSerie(GestionnaireSerie* gestionnaire) {
    this->gestionnaireSerie = gestionnaire;
}

void Ordonnancement::setGestionCapteur(GestionCapteur* g) {
    gcap = g;
    gcap->setDonneeOnduleur(refDon);
}
```

Formatter Code:

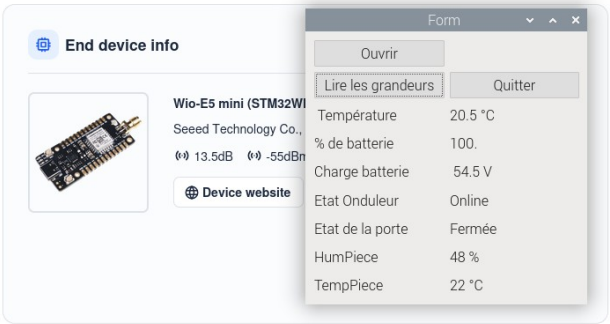
```
function formatAutonomie(mins) {
  const heures = Math.floor(mins / 60);
  const minutes = mins % 60;
  const secondes = 0;
  return `${heures}:${minutes}:${secondes}`;
}

function decodeUplink(input) {
  const bytes = input.bytes;
  const autonomieMin = (bytes[4] << 8) | bytes[5];

  return {
    data: {
      baies_id: 1,
      temp: bytes[0] + (bytes[1] / 100),
      nivBat: bytes[3],
      autonomieBat: formatAutonomie(autonomieMin),
      tempPiece: bytes[6],
      hum: bytes[7],
      etatOnd: (bytes[8] & 0x02) !== 0,
      etatPorte: (bytes[8] & 0x01) !== 0
    }
  };
}
```

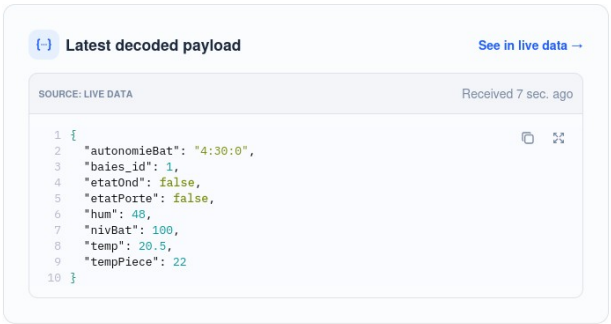
Programme de test (MonitorackController.php):

```
    #[Route("/api/trame", name: "app_trame", methods: ["POST"])] //La route de cette page est /api/trame et la route n'accepte que les requêtes HTTP POST.

   public function saveTrame(Request $request): JsonResponse //La fonction renvoie toujours une réponse JSON.
    {
    $data = json_decode($request->getContent(), true);//transformer un JSON en tableau associatif PHP.

    //Vérification que les données sont présentes
    if (!$data || empty($data['uplink_message']['decoded_payload'])) {
        return new JsonResponse(['error' => 'Invalid TTN payload'], JsonResponse::HTTP_BAD_REQUEST);
    }

    $payload = $data['uplink_message']['decoded_payload'];

    // Validation des données
    if (
        empty($payload['baies_id']) ||
        empty($payload['temp']) ||
        empty($payload['tempPiece']) ||
        empty($payload['nivBat']) ||
        empty($payload['hum']) ||
        empty($payload['autonomieBat']) ||
        !isset($payload['etatPorte']) ||
        !isset($payload['etatOnd'])
    ) {
        return new JsonResponse(['error' => 'Missing required fields'], JsonResponse::HTTP_BAD_REQUEST);
    }

    // Recherche de la baie
    $baie = $this->baiesRepository->find($payload['baies_id']);
    if (!$baie) {
        return new JsonResponse(['error' => 'Baie not found'], JsonResponse::HTTP_NOT_FOUND);
    }
    // Création de l'objet Grandeurs
    $grandeur = new Grandeurs();
    $grandeur->setBaiesId($baie);
    $grandeur->setTemp($payload['temp']);
```

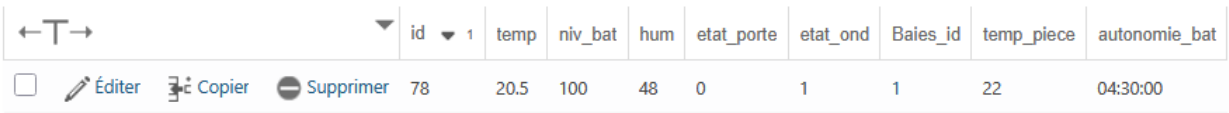
```
    $grandeur->setTempPiece($payload['tempPiece']);
    $grandeur->setNivBat($payload['nivBat']);
    $grandeur->setHum($payload['hum']);
    $grandeur->setAutonomieBat(new \DateTime($payload['autonomieBat']));
    $grandeur->setEtatPorte($payload['etatPorte']);
    $grandeur→setEtatOnd($payload['etatOnd']);

    // Sauvegarde
    $this→entityManager→persist($grandeur); //persist() indique à Doctrine qu'on veut ajouter cette entité.
    $this->entityManager->flush();flush() applique les changements en base de données (INSERT).

    return new JsonResponse([
        'message' => 'Trame saved successfully',
        'grandeur' => [
            'id' => $grandeur->getId(),
            'temp' => $grandeur->getTemp(),
            'tempPiece' => $grandeur->getTempPiece(),
            'nivBat' => $grandeur->getNivBat(),
            'hum' => $grandeur->getHum(),
            'autonomieBat' => $grandeur->getAutonomieBat()->format('Y-m-d H:i:s'),
            'etatPorte' => $grandeur->isEtatPorte(),
            'etatOnd' => $grandeur->isEtatOnd(),
        ]
    ], JsonResponse::HTTP_OK);
}
```

Résultats du test (validation de l'envoie de données):





Résultats du test (dans Phpmyadmin) :



	id	temp	niv_bat	hum	etat_porte	etat_ond	Baies_id	temp_piece	autonomie_bat
Éditer Copier Supprimer	78	20.5	100	48	0	1	1	22	04:30:00

Résultats du test (sur le site web)

12.1. Annexes Noah VIGNEAU

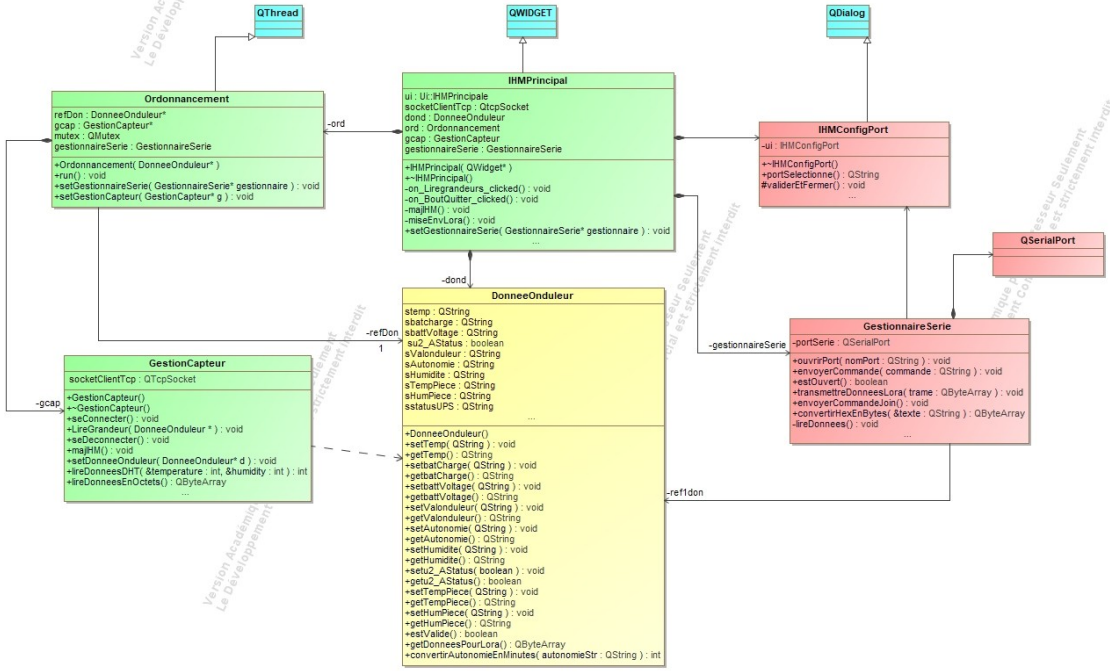
FICHE DE TEST FONCTIONNEL UNITAIRE N°1		
Projet **PRJ-Monitorack**	*Nom du module logiciel testé* **Classe GestionCapteur**	*Date* **7 Janvier 2025**
Nom(s) du (des) technicien(s) impliqué(s) dans le test : **Vigneau Noah**		
Description du module logiciel : **Code pour récupérer les valeurs de températures et humidité depuis le capteur DHT11**		

Environnement du test :

Type de test : □ Unitaire Nominal

Objectifs du test :

- **Valider l'obtention** de la mise a jour des valeurs demandés
- **Valider** affichage du taux d'humidité et de la température
- **Valider test DHT11**



Environnement matériel du test (diagrammes de paquets et / ou de classes) :

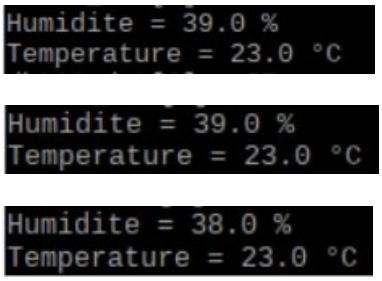
Programme de test (dht11):

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#define MAXTIMINGS 85
#define DHTPIN 7
int dht11_dat[5];
void comDht11()
{
```

```
//État bas pendant 18 millisecondes sur la sortie
pinMode(DHTPIN, 1);
digitalWrite(DHTPIN, 0);
delay(18);
//État haut pendant 30 microsecondes sur la sortie
digitalWrite(DHTPIN, 1);
delayMicroseconds(30);
//configurer la broche en entrée
pinMode(DHTPIN, 0);
}
void getDht11Donnees()
{
int laststate = 1;
int counter = 0;
int j = 0, i;
dht11_dat[0] = dht11_dat[1] = dht11_dat[2] = dht11_dat[3] = dht11_dat[4] = 0;
// détecter les changements et lire la donnee
for (i = 0; i < MAXTIMINGS; i++)
{
counter = 0;
while (digitalRead(DHTPIN) == laststate)
{
counter++;
delayMicroseconds(1);
if (counter == 255) break;
}
laststate = digitalRead(DHTPIN);
if (counter == 255) break;
// Ignorer les trois premiers fronts
if ((i >= 4) && (i%2 == 0))
{
// Insérer chaque bit dans les octets de stockage
dht11_dat[j/8] <<= 1;
if (counter > 26) dht11_dat[j/8] |= 1;
j++;
}
}
printf("dht11_dat[0] = %x\n", dht11_dat[0]); // dans la version finale, pour un
printf("dht11_dat[1] = %x\n", dht11_dat[1]); // confort évident de l'affichage
printf("dht11_dat[2] = %x\n", dht11_dat[2]); // des résultats, ces lignes
printf("dht11_dat[3] = %x\n", dht11_dat[3]); // n'apparaissent plus
// Vérifier que nous lisons les 40 bits (8 bits x 5 + vérifier la somme logique des octets précédents dans le
dernier octet
// Afficher en console les bonnes données
if ((j >= 40) && (dht11_dat[4] == ((dht11_dat[3] + dht11_dat[2] + dht11_dat[1] +
dht11_dat[0]) & 0xFF)) )
{
printf("Humidite = %d.%d %% \nTemperature = %d.%d °C \n", dht11_dat[0],
dht11_dat[1], dht11_dat[2], dht11_dat[3]);
}
else
{
printf("Donnees fausses\n");
}
```

```
}
int main (void)
{
printf ("Raspberry Pi wiringPi DHT11: Programme de Test Temperature et Humidite\n");
if (wiringPiSetup () == -1) //Initialisation de la connexion des broches du connecteur GPIO
exit (1) ;
while (1)
{
comDht11();
getDht11Donnees();
delay(10000); // Attente de rafraîchissement toutes les 10 secondes
}
return 0 ;
}
```

Résultats du test (validation du code) :



FICHE DE TEST FONCTIONNEL UNITAIRE N°2

Projet	*Nom du module logiciel testé*	*Date*
PRJ-Monitorack	**Classe IHMPrincipale**	**7 Janvier 2025**

Nom(s) du (des) technicien(s) impliqué(s) dans le test : **Vigneau Noah**

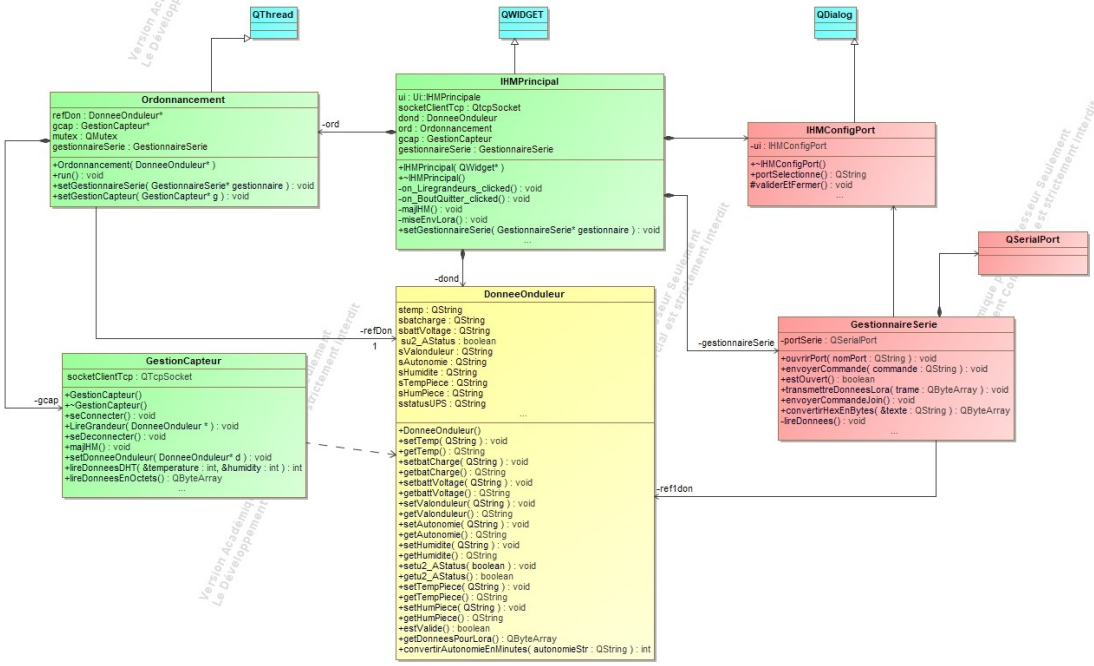
Description du module logiciel : **Classe développée en C++ IHMPrincipal (voir code page xx des annexes)**

Environnement du test :

Type de test : □ Unitaire Nominal ■ **Unitaire aux limites**

Objectifs du test :

- **Valider** la récupération des données de l'onduleur
- **Valider l'obtention des données dans l'application**



Environnement matériel du test (diagrammes de paquets et / ou de classes)

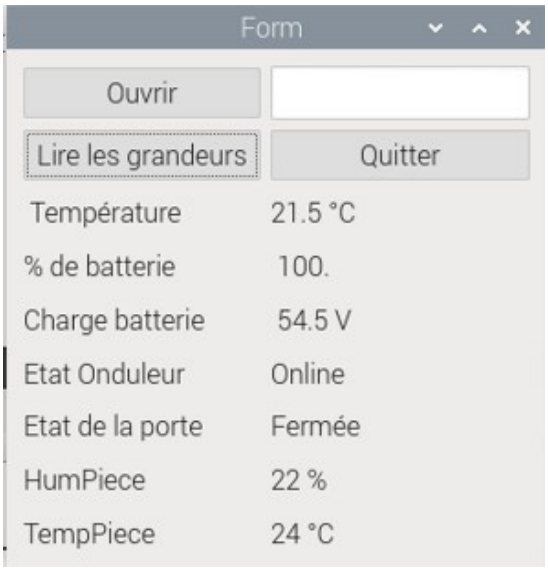
Programme de test (IHMPRINCIPAL:

```
#include "ihmprincipal.h"
#include "ui_ihmprincipal.h"
#include "ihmconfigport.h"
#include <QPushButton>
#include <QTcpSocket>
#include <iostream>
#include <QString>
#include <QDebug>
#include <QThread>
#include "ordonnancement.h"
#include "donneeonduleur.h"
#include "gestioncapteur.h"

IHMPrincipal::IHMPrincipal(QWidget *parent)
```

```
    : QWidget(parent)
    , ui(new Ui::IHMPrincipal)
{
ui->setupUi(this);
//socketClientTcp = new QTcpSocket();
dond = new DonneeOnduleur;
ord = new Ordonnancement(dond);
connect(ord,SIGNAL(miseAJIHM()),this,SLOT(majIHM()));

}
IHMPrincipal::~IHMPrincipal()
{
 // Libère la mémoire
           delete ord; // Libérer la mémoire du thread
           delete dond;

    delete ui;
}

void IHMPrincipal::on_Seconnecter_clicked()
{
        socketClientTcp->connectToHost("10.10.203.200",23,QIODevice::ReadWrite);//Appel de la méthode de l'instance socketClientTcp pour se connecter à l'hote
    if(socketClientTcp->waitForConnected()!=true)  // Test si la connexion de l''instance socketClientTcp en appelant la méthode adéquate est établie
    {
            qDebug()<<"La connexion avec l'hote "<<"10.10.203.200 est impossible"<<endl; // Affichage en console Connexion impossible avec l'hote: 10.10.203.100 ou 105
            qDebug()<<"L'etat de connexion est : "<<socketClientTcp->state()<<endl; // Affichage en console l'état de la connexion

    }
    else
    {
            qDebug()<<"La connexion avec l'hote 10.10.203.200 est effective"<<endl; // Affichage en console Connecte à l'hote: 10.10.203.200 ou 105
            qDebug()<<"L'etat de connexion est : "<<socketClientTcp->state()<<endl; // Affichage en console l'état de la connexion
            QThread::msleep(1000);
            socketClientTcp->waitForBytesWritten(3000);
            socketClientTcp->write("apc");
            socketClientTcp->write("\r\n");
            socketClientTcp->waitForBytesWritten(3000);
            socketClientTcp->write("apc");
            socketClientTcp->write("\r\n");

            QThread::msleep(1000);
            socketClientTcp->waitForBytesWritten(3000);
            socketClientTcp->write("detstatus -all");
            socketClientTcp->write("\r\n");
            socketClientTcp->write("uio -st");
            socketClientTcp→write("\r\n");
```

```
}

}

void IHMPrincipal::on_Sedeconnecter_clicked()
{
    socketClientTcp->disconnectFromHost(); // Déconnexion de l'hote
     qDebug()<<"La déconnexion avec l'hote 10.10.203.200 est effective"<<endl;
// Affichage en console Connecte à l'hote: 10.10.203.200 ou 105

}

void IHMPrincipal::on_Liregrandeurs_clicked()
{
             ord->start();
}

void IHMPrincipal::on_BoutQuitter_clicked()
{
    close();
}

void IHMPrincipal::majIHM()
{
    qDebug() << Q_FUNC_INFO;
    ui->Valtemp1->setText(dond->getTemp());
    qDebug() << "Valeur de la température"<< dond->getTemp();
    ui->Valtempbat->setText(dond->getbatCharge());
    ui->Valchargebat->setText(dond->getbattVoltage());
    ui->Valonduleur->setText(dond->getstatusUPS());

}
```

Résultats du test (validation de la classe) :



FICHE DE TEST FONCTIONNEL UNITAIRE N°3		
Projet **PRJ-Monitorack**	*Nom du module logiciel testé* **Classe IHMPrincipale**	*Date* **7 Janvier 2025**

Nom(s) du (des) technicien(s) impliqué(s) dans le test : **Vigneau Noah**

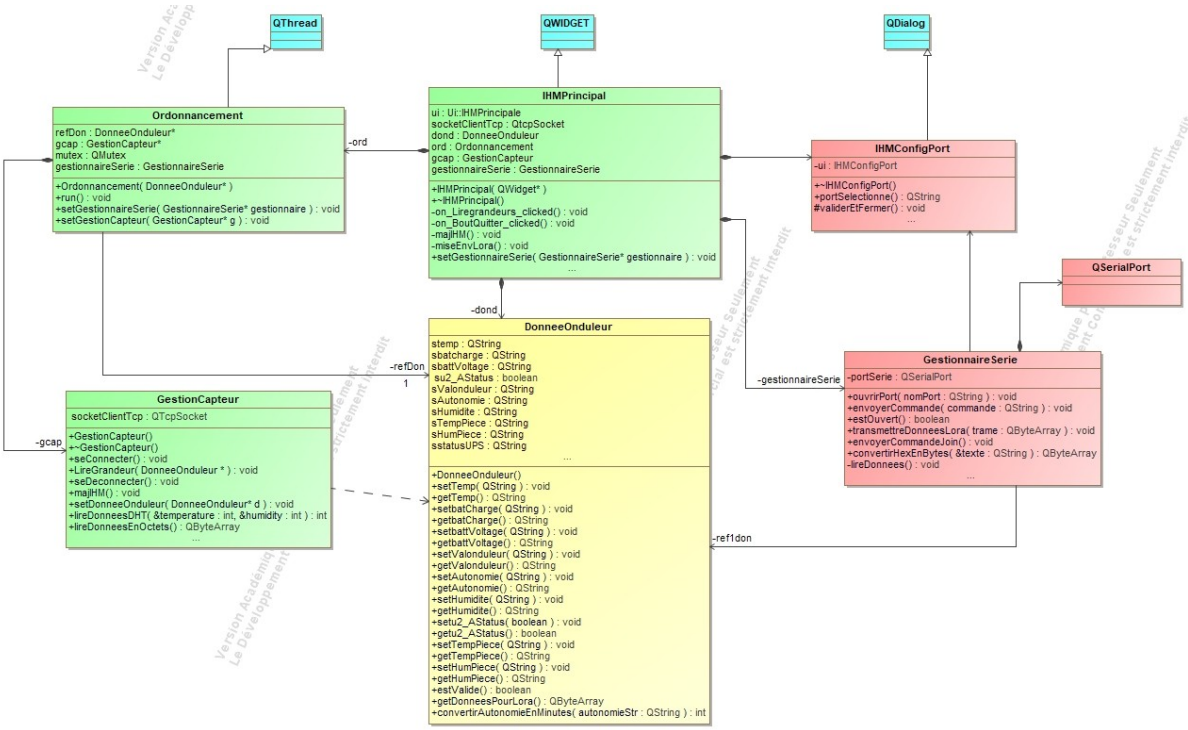
Description du module logiciel : **Classe développée en C++ IHMPrincipal (voir code page xx des annexes)**

Environnement du test :

Type de test : □ Unitaire Nominal ■ **Unitaire aux limites**

Objectifs du test :

- Test mise a jour IHM
- **Valider le fonctionnement du thread**



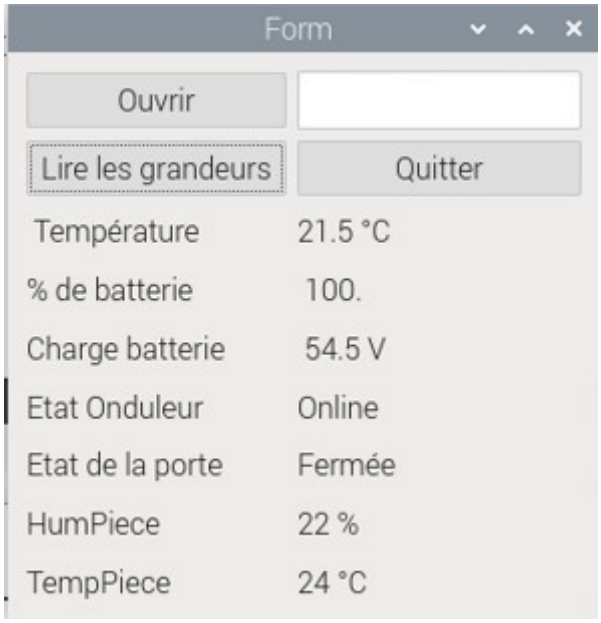
Environnement matériel du test (diagrammes de paquets et / ou de classes)

```
#include "ordonnancement.h"
#include <QThread>
#include <QDebug>
#include "donneeonduleur.h"
#include "ihmprincipal.h"
#include "QString"
#include "gestioncapteur.h"


Ordonnancement::Ordonnancement(DonneeOnduleur *irefDon)
{
```

```
    refDon = irefDon;
    gcap = nullptr;  // ne pas initialiser ici directement
}


void Ordonnancement::run()
{
        gcap->lireGrandeur(refDon);
        gcap->seConnecter();

    while (true) {

       // mutex.lock();
 qDebug() << Q_FUNC_INFO;
        // Connexion à la carte de gestion des capteurs

gcap->lireGrandeur(refDon);
        // Récupérer les données des capteurs
        //DonneeOnduleur ref1Don(gcap->temp,gcap->batCharge);
      //  ref1Don.setTemp(gcap->temp);
       // qDebug() << "Valeur de la température"<<gcap->temp;
       // qDebug() << "Valeur de la température"<< ref1Don.getTemp();
     //   ref1Don.setbatCharge(gcap->batCharge);
   //     ref1Don.setbattVoltage(gcap->battVoltage);
   //     ref1Don.setstatusUPS(gcap->statusUPS);
   //     ref1Don.setAutonomie(gcap->autonomie);
  //      ref1Don.setHumidite(gcap->humidite);


        emit miseAJIHM();  // Déclenche la mise à jour de l'IHM
        emit miseEnvLora();

         // Déconnexion de l'hôte après la récupération des données

       // mutex.unlock();
        if (!gcap || !refDon) {
         qDebug() << "Erreur : gcap ou refDon non initialisé.";
           return;
        }
        QThread::sleep(10);  // Pause de 60 secondes avant la prochaine mise à jour
    }

  gcap->seDeconnecter();
}

void Ordonnancement::setGestionnaireSerie(GestionnaireSerie* gestionnaire) {
    this->gestionnaireSerie = gestionnaire;
}



void Ordonnancement::miseEnvLora()
{
    if (gestionnaireSerie && refDon) {
        if (refDon->estValide()) {
            QByteArray trame = refDon->getDonneesPourLora();
            qDebug() << "[Envoi LoRa] Trame : " << trame.toHex();
```

```
            gestionnaireSerie->transmettreDonneesLora(trame);
        } else {
            qDebug() << "❌ Données non valides, trame non envoyée.";
        }
    }
}


void Ordonnancement::setGestionCapteur(GestionCapteur* g) {
    gcap = g;
    gcap->setDonneeOnduleur(refDon);
}
```

Résultats du test (validation de la classe) :




```
batterie de charge: " 100."
batterie voltage: " 54.5"
 État U2_A (brut) : "C"
fermée
Température pièce (DHT11) :  24
Humidité pièce (DHT11) :  22
void IHMPrincipal::majIHM()
Valeur de la température "21.5"
virtual void Ordonnancement::run()
Test1
void GestionCapteur::lireGrandeur(DonneeOnduleur*) appelé avec DonneeOnduleur* 0x55561d3f23b0
Réponse reçue :  "d"
Température pièce (DHT11) :  24
Humidité pièce (DHT11) :  22
void IHMPrincipal::majIHM()
Valeur de la température "21.5"
15:11:53: /home/etudiant/Desktop/build-ProJetOnd-Raspberry-Debug/ProJetOnd exited with code 0
```

FICHE DE TEST FONCTIONNEL UNITAIRE N°4		
Projet **PRJ-Monitorack**	*Nom du module logiciel testé* **Classe IHMPrincipale**	*Date* **7 Janvier 2025**

Nom(s) du (des) technicien(s) impliqué(s) dans le test : **Vigneau Noah**

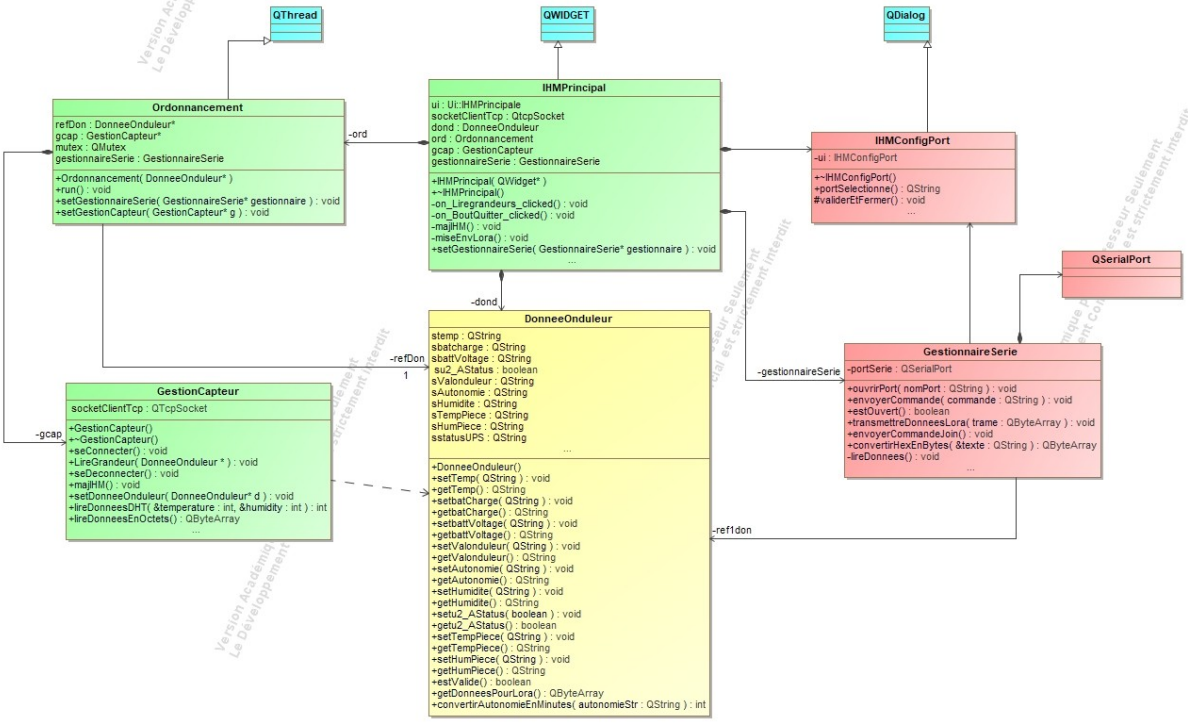
Description du module logiciel : **Classe développée en C++ IHMPrincipal (voir code page xx des annexes)**

Environnement du test :

Type de test : □ Unitaire Nominal ■ **Unitaire aux limites**

Objectifs du test :

- Envoyer les valeurs de l'onduleur sur le serveur
- **Travail en collaboration**



Environnement matériel du test (diagrammes de paquets et / ou de classes)

Programme de test (Donnee Onduleur:

```
#include "donneeonduleur.h"
#include <QDebug>


DonneeOnduleur::DonneeOnduleur()
{

}

DonneeOnduleur::DonneeOnduleur(QString irefTemp, QString irefbatCharge)
```

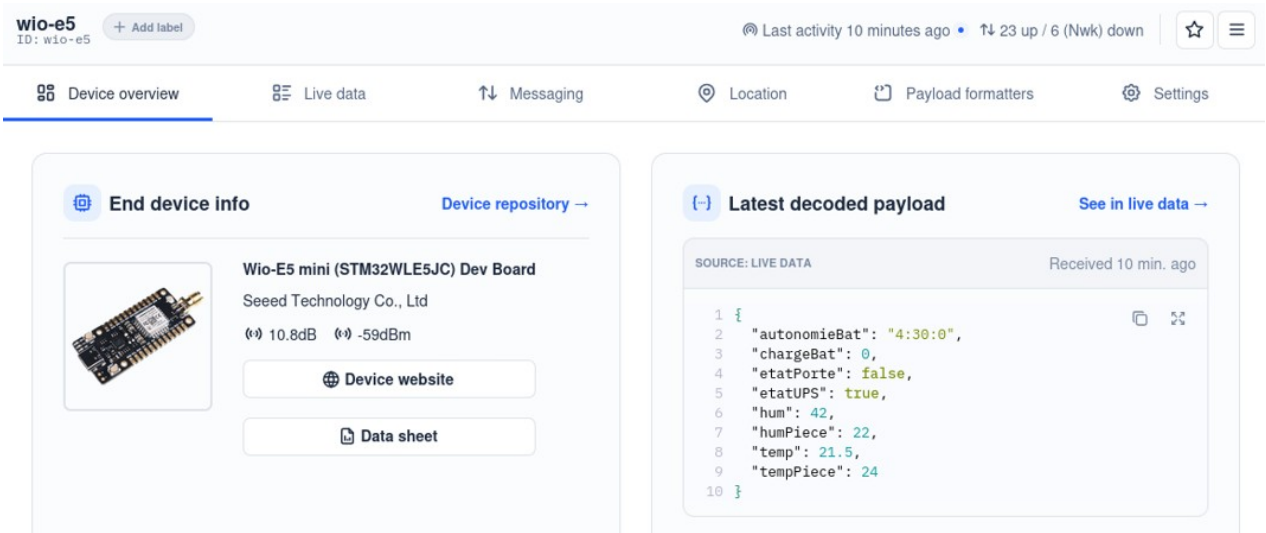
```
{
   stemp = irefTemp;
   sbatcharge = irefbatCharge;
}

void DonneeOnduleur::setTemp(QString itemp)
{
   stemp = itemp;

}

QString DonneeOnduleur::getTemp()
{
   return stemp;

}

void DonneeOnduleur::setbatCharge(QString ibatcharge)
{
   sbatcharge = ibatcharge;
}

QString DonneeOnduleur::getbatCharge()
{
   return sbatcharge;
}

void DonneeOnduleur::setbattVoltage(QString ibattVoltage)
{
   sbattVoltage = ibattVoltage;
}

QString DonneeOnduleur::getbattVoltage()
{
   return sbattVoltage;
}

void DonneeOnduleur::setstatusUPS(QString istatusUPS)
{
   sstatusUPS = istatusUPS;
}

QString DonneeOnduleur::getstatusUPS()
{
   return sstatusUPS;
}

void DonneeOnduleur::setValonduleur(QString iValonduleur)
{
   sValonduleur = iValonduleur;
}

QString DonneeOnduleur::getValonduleur()
{
```

```
    return sValonduleur;
}

void DonneeOnduleur::setAutonomie(QString iAutonomie)
{
    sAutonomie = iAutonomie;
}

QString DonneeOnduleur::getAutonomie()
{
    return sAutonomie;
}

void DonneeOnduleur::setHumidite(QString iHumidite)
{
    sHumidite = iHumidite;
}

QString DonneeOnduleur::getHumidite()
{
    return sHumidite;
}

void DonneeOnduleur::setu2_AStatus(bool iu2_AStatus)
{
    su2_AStatus = iu2_AStatus;
}

bool DonneeOnduleur::getu2_AStatus()
{
    return su2_AStatus;
}

void DonneeOnduleur::setTempPiece(QString iTempPiece)
{
    sTempPiece = iTempPiece;
}

QString DonneeOnduleur::getTempPiece()
{
    return sTempPiece;
}

void DonneeOnduleur::setHumPiece(QString iHumPiece)
{
    sHumPiece = iHumPiece;
}

QString DonneeOnduleur::getHumPiece()
{
    return sHumPiece;
}

int DonneeOnduleur::convertirAutonomieEnMinutes(const QString& autonomieStr) {
```

```
    int heures = 0;
    int minutes = 0;

    QRegExp regexHeures("(\\d+)\\s*hr");
    QRegExp regexMinutes("(\\d+)\\s*min");

    if (regexHeures.indexIn(autonomieStr) != -1) {
        heures = regexHeures.cap(1).toInt();
    }
    if (regexMinutes.indexIn(autonomieStr) != -1) {
        minutes = regexMinutes.cap(1).toInt();
    }

    int totalMinutes = heures * 60 + minutes;
    if (totalMinutes > 65535) totalMinutes = 65535; // borne max 16 bits

    return totalMinutes;
}


QByteArray DonneeOnduleur::getDonneesPourLora() {
    QByteArray trame;
    bool ok = false;

    // Température onduleur
    float tempF = stemp.trimmed().toFloat(&ok);
    if (!ok) tempF = 0;
    int tempEntier = static_cast<int>(tempF);
    int tempCentieme = static_cast<int>((tempF - tempEntier) * 100);

    // Humidité onduleur
    int hum = sHumidite.trimmed().toFloat(&ok);
    if (!ok) hum = 0;

    // Charge batterie
    int charge = sbatcharge.trimmed().toFloat(&ok);
    if (!ok) charge = 0;

    // Autonomie en minutes (2 octets)
    int autoUPS = convertirAutonomieEnMinutes(sAutonomie);

    // Température pièce
    int tempPiece = sTempPiece.trimmed().toFloat(&ok);
    if (!ok) tempPiece = 0;

    // Humidité pièce
    int humPiece = sHumPiece.trimmed().toFloat(&ok);
    if (!ok) humPiece = 0;

    // États combinés dans un octet (bit 1 = onduleur, bit 0 = porte)
    bool etatOnd = sValonduleur.trimmed().toFloat(&ok) > 0;
    char flags = 0;
    if (etatOnd) flags |= 0x02;
    if (su2_AStatus) flags |= 0x01;
```

```
    // DEBUG
    qDebug() << "[Trame LoRa]"
             << "Temp:" << tempF
             << "Hum:" << hum
             << "Charge:" << charge
             << "Auto (min):" << autoUPS
             << "TempPièce:" << tempPiece
             << "HumPièce:" << humPiece
             << "EtatOnd:" << etatOnd
             << "EtatPorte:" << su2_AStatus;

    // Construction trame
    trame.append(static_cast<unsigned char>(tempEntier));    // Byte 0
    trame.append(static_cast<unsigned char>(tempCentieme));  // Byte 1
    trame.append(static_cast<unsigned char>(hum));           // Byte 2
    trame.append(static_cast<unsigned char>(charge));        // Byte 3

    // AUTONOMIE 2 octets (MSB puis LSB)
    trame.append(static_cast<unsigned char>((autoUPS >> 8) & 0xFF));  // Byte 4
    trame.append(static_cast<unsigned char>(autoUPS & 0xFF));         // Byte 5

    trame.append(static_cast<unsigned char>(tempPiece));     // Byte 6
    trame.append(static_cast<unsigned char>(humPiece));      // Byte 7
    trame.append(flags);                                     // Byte 8

    return trame;
}

bool DonneeOnduleur::estValide() {
    return !stemp.isEmpty() && !sbattVoltage.isEmpty(); // ou toute autre logique de validation
}
```

Résultats du test (validation de la classe) :



12.2. Annexes Raad SHEPHO

Fiche de test 1 :

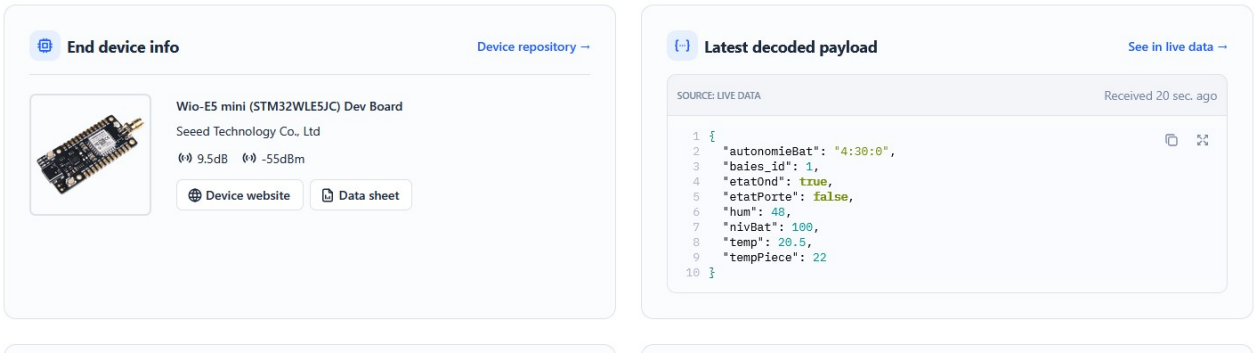
FICHE DE TEST FONCTIONNEL UNITAIRE N°1		
Projet **PRJ-MONITORACK**	*Nom du module logiciel testé* **Décodage des trames LoRaWAN**	*Date* **24 Avril 2025**
Nom(s) du (des) technicien(s) impliqué(s) dans le test : **SHEPHO Raad**		
Description du module logiciel : **Script développé en JavaScript utilisé dans The Things Network (TTN) pour le décodage des trames LoRaWAN qui envoyer par le module LoRa depuis QT Creator. Il permet d'extraire les données brutes du payload et de les convertir en informations compréhensibles.**		
Environnement du test : *Type de test* : □ Unitaire Nominal ■ **Unitaire aux limites** *Objectifs du test :* • **Vérifier** le bon fonctionnement du **décodage du payload** LoRaWAN. • **Valider** l'extraction correcte des **données (température, humidité, batterie, etc.).** • **Contrôler la gestion des erreurs de format ou de taille de trame.** • **Assurer** l'affichage **des données décodées** dans la console **TTN.**		

Programme de test (Javascript formatter):

```
function formatAutonomie(mins) {
  const heures = Math.floor(mins / 60);
  const minutes = mins % 60;
  const secondes = 0;
  return `${heures}:${minutes}:${secondes}`;
}

function decodeUplink(input) {
  const bytes = input.bytes;
  const autonomieMin = (bytes[4] << 8) | bytes[5];

  return {
    data: {
      baies_id: 1,
      temp: bytes[0] + (bytes[1] / 100),
      nivBat: bytes[3],
      autonomieBat: formatAutonomie(autonomieMin),
      tempPiece: bytes[6],
      hum: bytes[7],
```

```
      etatOnd: (bytes[8] & 0x02) !== 0,
      etatPorte: (bytes[8] & 0x01) !== 0
    }
  };
}
```

Résultats du test (validation de script) :



Fiche de test 2

FICHE DE TEST FONCTIONNEL UNITAIRE N°2

Projet	*Nom du module logiciel testé*	*Date*
PRJ-MONITORACK	**Interface de communication LoRa / LoRaWAN (Qt)**	**02 Mai 2025**

Nom(s) du (des) technicien(s) impliqué(s) dans le test : **SHEPHO Raad**

Description du module logiciel : **Application développée en C++ avec Qt Creator permettant la communication série entre un ordinateur et un module LoRa (Wio-E5-LE mini) via des commandes AT.**

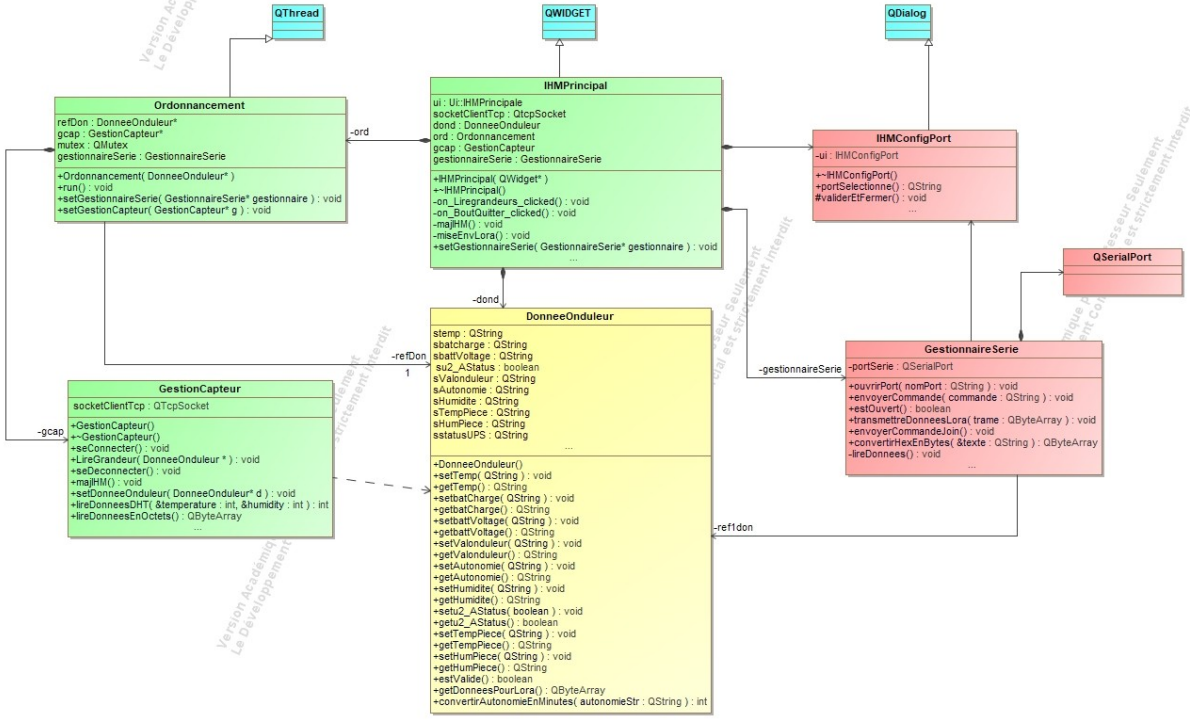
Environnement du test :

Type de test : □ Unitaire Nominal ■ **Unitaire aux limites**

Objectifs du test :

- **Vérifier** l'ouverture du port série et la communication avec le module **LoRa**
- **Tester** l'**envoie des données** pour **vérifier** l'enregistrement du module sur **TTN**
- **Confirmer** la réception des données dans **TTN**
- **Vérifier** la gestion des réponses du module en cas de jonction déjà effectuée

Environnement matériel du test (Ma partie est en rouge):



donneeonduleur.cpp:

```
int DonneeOnduleur::convertirAutonomieEnMinutes(const QString& autonomieStr) {
 int heures = 0;
 int minutes = 0;

 QRegExp regexHeures("(\\d+)\\s*hr");
 QRegExp regexMinutes("(\\d+)\\s*min");
```

```
 if (regexHeures.indexIn(autonomieStr) != -1) {
 heures = regexHeures.cap(1).toInt();
 }
 if (regexMinutes.indexIn(autonomieStr) != -1) {
 minutes = regexMinutes.cap(1).toInt();
 }

 int totalMinutes = heures * 60 + minutes;
 if (totalMinutes > 65535) totalMinutes = 65535; // borne max 16 bits

 return totalMinutes;
}

QByteArray DonneeOnduleur::getDonneesPourLora() {
    QByteArray trame;
    bool ok = false;

    float tempF = stemp.trimmed().toFloat(&ok);
    if (!ok) tempF = 0;
    int tempEntier = static_cast<int>(tempF);
    int tempCentieme = static_cast<int>((tempF - tempEntier) * 100);

    int hum = sHumidite.trimmed().toFloat(&ok);
    if (!ok) hum = 0;

    int charge = sbatcharge.trimmed().toFloat(&ok);
    if (!ok) charge = 0;

    int autoUPS = convertirAutonomieEnMinutes(sAutonomie);

    int tempPiece = sTempPiece.trimmed().toFloat(&ok);
    if (!ok) tempPiece = 0;

    int humPiece = sHumPiece.trimmed().toFloat(&ok);
    if (!ok) humPiece = 0;

    bool etatOnd = sValonduleur.trimmed().toFloat(&ok) > 0;
    char flags = 0;
    if (etatOnd) flags |= 0x02;
    if (su2_AStatus) flags |= 0x01;

    // DEBUG
    qDebug() << "[Trame LoRa]"
             << "Temp:" << tempF
             << "Hum:" << hum
             << "Charge:" << charge
             << "Auto (min):" << autoUPS
             << "TempPièce:" << tempPiece
             << "HumPièce:" << humPiece
             << "EtatOnd:" << etatOnd
             << "EtatPorte:" << su2_AStatus;

    trame.append(static_cast<unsigned char>(tempEntier));
    trame.append(static_cast<unsigned char>(tempCentieme));
    trame.append(static_cast<unsigned char>(hum));
    trame.append(static_cast<unsigned char>(charge));
```

```
    trame.append(static_cast<unsigned char>((autoUPS >> 8) & 0xFF));
    trame.append(static_cast<unsigned char>(autoUPS & 0xFF));

    trame.append(static_cast<unsigned char>(tempPiece));
    trame.append(static_cast<unsigned char>(humPiece));
    trame.append(flags);

    return trame;
}

bool DonneeOnduleur::estValide() {
    return !stemp.isEmpty() && !sbattVoltage.isEmpty(); // ou toute autre
logique de validation
}
```

gestioncapteur.cpp:

```
bool GestionnaireSerie::ouvrirPort(const QString &nomPort) {
    if (portSerie.isOpen())
        portSerie.close();

    portSerie.setPortName(nomPort);
    portSerie.setBaudRate(9600);
    portSerie.setDataBits(QSerialPort::Data8);
    portSerie.setParity(QSerialPort::NoParity);
    portSerie.setStopBits(QSerialPort::OneStop);
    portSerie.setFlowControl(QSerialPort::NoFlowControl);
    return portSerie.open(QIODevice::ReadWrite);
}

void GestionnaireSerie::envoyerCommande(const QString &commande) {
    if (portSerie.isOpen()) {
        portSerie.write(commande.toUtf8());
    }
}

void GestionnaireSerie::lireDonnees() {
    QByteArray donnees = portSerie.readAll();
    emit donneesRecues(donnees);
}

bool GestionnaireSerie::estOuvert() const {
    return portSerie.isOpen();
}

void GestionnaireSerie::envoyerCommandeJoin() {
    envoyerCommande("AT+JOIN\r\n");
}

QByteArray GestionnaireSerie::convertirHexEnBytes(const QString &texte) {
    QByteArray resultat;
    QStringList octets = texte.split(' ', Qt::SkipEmptyParts);
    for (const QString &octet : octets) {
        bool ok;
        char byte = octet.toUInt(&ok, 16);
        if (ok) {
            resultat.append(byte);
        }
```

```
    }
    return resultat;
}

void GestionnaireSerie::transmettreDonneesLora(QByteArray trame) {
    if (!estOuvert()) {
        qDebug() << "Port série non ouvert !";
        return;
    }

    if (trame.isEmpty() || trame == QByteArray(trame.size(), 0x00)) {
        qDebug() << "Trame vide détectée, envoi annulé :" << trame.toHex();
        return;
    }

    QString hex = trame.toHex().toUpper();
    QString commande = "AT+MSGHEX=" + hex + "\r\n";

    envoyerCommande(commande);  // Envoi immédiat
    qDebug() << "Commande envoyée :" << commande;
}
```

ihmprincipal.cpp:

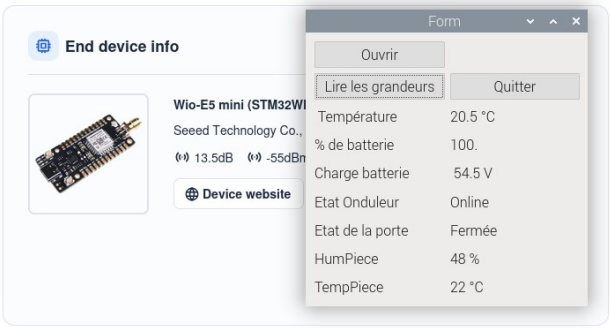
```
void IHMPrincipal::miseEnvLora()
{
    if (gestionnaireSerie && dond) {
        if (dond->estValide()) {
            QByteArray trame = dond->getDonneesPourLora();
            qDebug() << "[Envoi LoRa] Trame : " << trame.toHex();
            gestionnaireSerie->transmettreDonneesLora(trame);
        } else {
            qDebug() << "Données non valides, trame non envoyée.";
        }
    }
}

void IHMPrincipal::setGestionnaireSerie(GestionnaireSerie* gestionnaire) {
    this->gestionnaireSerie = gestionnaire;
    if (ordonnancement)
        ordonnancement->setGestionnaireSerie(gestionnaire);
}

void IHMPrincipal::on_confPort_clicked() {
    qDebug() << "on_confPort_clicked appelé";
    IHMConfigPort configDialog(this);
    if (configDialog.exec() == QDialog::Accepted) {
        QString port = configDialog.portSelectionne();
        qDebug() << "Port sélectionné :" << port;

        if (!gestionnaireSerie) {
            gestionnaireSerie = new GestionnaireSerie(this);
            connect(gestionnaireSerie, &GestionnaireSerie::donneesRecues,
this, [](QByteArray donnees){
                qDebug() << "Données LoRa reçues :" << donnees;
            });
        }
        if (gestionnaireSerie->ouvrirPort(port)) {
```

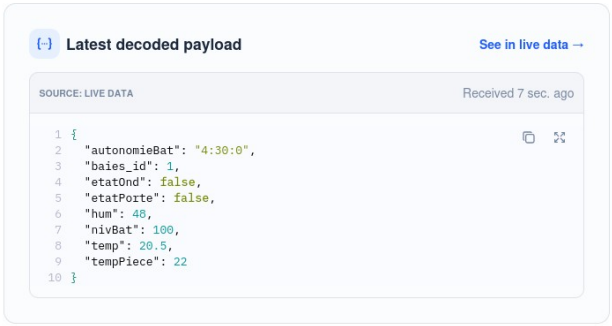
```
            gestionnaireSerie->envoyerCommandeJoin();
        }

    }
}
```

ordonnancement.cpp:

```
void Ordonnancement::run()
{
    if (!gcap || !refDon) {
        qDebug() << "Erreur : gcap ou refDon non initialisé.";
        return;
    }

    gcap->seConnecter();

    while (true) {
        qDebug() << Q_FUNC_INFO;

        gcap->lireGrandeur(refDon);

        emit miseAJIHM();
        emit miseEnvLora();

        QThread::sleep(10);  // Pause avant la prochaine lecture
    }

    gcap->seDeconnecter();
}

void Ordonnancement::setGestionnaireSerie(GestionnaireSerie* gestionnaire) {
    this->gestionnaireSerie = gestionnaire;
}

void Ordonnancement::setGestionCapteur(GestionCapteur* g) {
    gcap = g;
    gcap->setDonneeOnduleur(refDon);
}
```

Résultats du test (validation de l'envoie de données :





On peut voir que c'est bien les données qui on l'a reçu sur l'ihm c'est la même de qui avais afficher sur TTN.

Fiche de test 3

FICHE DE TEST FONCTIONNEL UNITAIRE N°3

Projet	*Nom du module logiciel testé*	*Date*
PRJ-MONITORACK	**Module C++ recevant les données TTN et les envoyant au serveur**	**05 Juin 2025**

Nom(s) du (des) technicien(s) impliqué(s) dans le test : **SHEPHO Raad**

Description du module logiciel : **Application développée en C++ avec Qt Creator permettant la communication série entre un ordinateur et un module LoRa (Wio-E5-LE mini) via des commandes AT. Elle récupère les données issues de l'acquisition des grandeurs, les intègre dans le protocole de communication LoRa, puis les transmet au réseau The Things Network (TTN). Les données reçues sur TTN sont ensuite redirigées automatiquement, au format JSON, vers un serveur distant.**

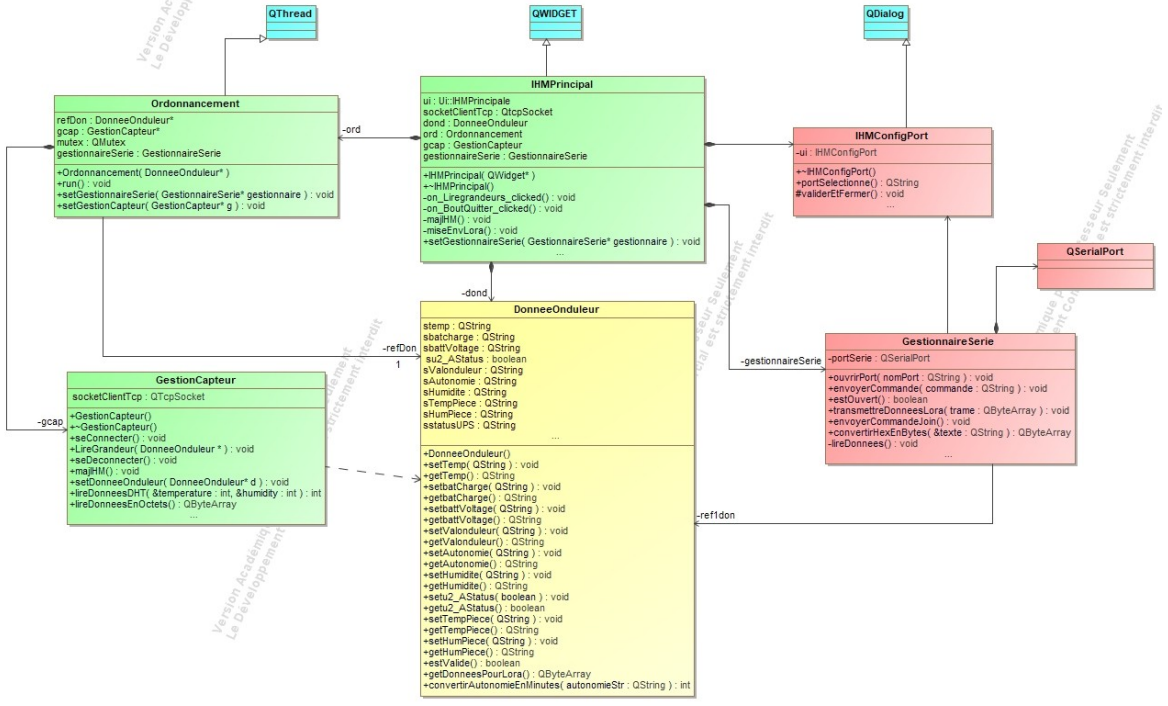
Environnement du test :

Type de test : □ Unitaire Nominal ■ **Unitaire aux limites**

Objectifs du test :

- **Vérifier** l'ouverture du port série et la communication avec le module **LoRa**
- **Tester** l'**envoie des données** pour **vérifier** l'enregistrement du module sur **TTN**
- **Confirmer** la réception des **données** dans **TTN**
- **Vérifier** la gestion des réponses du module en cas de jonction déjà effectuée
- **Vérifier** la réception des **données sur la base de données**

Environnement matériel du test (Ma partie est en rouge):



donneeonduleur.cpp:

```
int DonneeOnduleur::convertirAutonomieEnMinutes(const QString& autonomieStr) {
 int heures = 0;
 int minutes = 0;
```

```
 QRegExp regexHeures("(\\d+)\\s*hr");
 QRegExp regexMinutes("(\\d+)\\s*min");

 if (regexHeures.indexIn(autonomieStr) != -1) {
 heures = regexHeures.cap(1).toInt();
 }
 if (regexMinutes.indexIn(autonomieStr) != -1) {
 minutes = regexMinutes.cap(1).toInt();
 }

 int totalMinutes = heures * 60 + minutes;
 if (totalMinutes > 65535) totalMinutes = 65535; // borne max 16 bits

 return totalMinutes;
}



QByteArray DonneeOnduleur::getDonneesPourLora() {
    QByteArray trame;
    bool ok = false;

    float tempF = stemp.trimmed().toFloat(&ok);
    if (!ok) tempF = 0;
    int tempEntier = static_cast<int>(tempF);
    int tempCentieme = static_cast<int>((tempF - tempEntier) * 100);

    int hum = sHumidite.trimmed().toFloat(&ok);
    if (!ok) hum = 0;

    int charge = sbatcharge.trimmed().toFloat(&ok);
    if (!ok) charge = 0;

    int autoUPS = convertirAutonomieEnMinutes(sAutonomie);

    int tempPiece = sTempPiece.trimmed().toFloat(&ok);
    if (!ok) tempPiece = 0;

    int humPiece = sHumPiece.trimmed().toFloat(&ok);
    if (!ok) humPiece = 0;

    bool etatOnd = sValonduleur.trimmed().toFloat(&ok) > 0;
    char flags = 0;
    if (etatOnd) flags |= 0x02;
    if (su2_AStatus) flags |= 0x01;

    // DEBUG
    qDebug() << "[Trame LoRa]"
             << "Temp:" << tempF
             << "Hum:" << hum
             << "Charge:" << charge
             << "Auto (min):" << autoUPS
             << "TempPièce:" << tempPiece
             << "HumPièce:" << humPiece
             << "EtatOnd:" << etatOnd
             << "EtatPorte:" << su2_AStatus;
```

```
    trame.append(static_cast<unsigned char>(tempEntier));
    trame.append(static_cast<unsigned char>(tempCentieme));
    trame.append(static_cast<unsigned char>(hum));
    trame.append(static_cast<unsigned char>(charge));

    trame.append(static_cast<unsigned char>((autoUPS >> 8) & 0xFF));
    trame.append(static_cast<unsigned char>(autoUPS & 0xFF));

    trame.append(static_cast<unsigned char>(tempPiece));
    trame.append(static_cast<unsigned char>(humPiece));
    trame.append(flags);

    return trame;
}

bool DonneeOnduleur::estValide() {
    return !stemp.isEmpty() && !sbattVoltage.isEmpty(); // ou toute autre
logique de validation
}
```

gestioncapteur.cpp:

```
bool GestionnaireSerie::ouvrirPort(const QString &nomPort) {
    if (portSerie.isOpen())
        portSerie.close();

    portSerie.setPortName(nomPort);
    portSerie.setBaudRate(9600);
    portSerie.setDataBits(QSerialPort::Data8);
    portSerie.setParity(QSerialPort::NoParity);
    portSerie.setStopBits(QSerialPort::OneStop);
    portSerie.setFlowControl(QSerialPort::NoFlowControl);
    return portSerie.open(QIODevice::ReadWrite);
}

void GestionnaireSerie::envoyerCommande(const QString &commande) {
    if (portSerie.isOpen()) {
        portSerie.write(commande.toUtf8());
    }
}

void GestionnaireSerie::lireDonnees() {
    QByteArray donnees = portSerie.readAll();
    emit donneesRecues(donnees);
}

bool GestionnaireSerie::estOuvert() const {
    return portSerie.isOpen();
}

void GestionnaireSerie::envoyerCommandeJoin() {
    envoyerCommande("AT+JOIN\r\n");
}

QByteArray GestionnaireSerie::convertirHexEnBytes(const QString &texte) {
    QByteArray resultat;
    QStringList octets = texte.split(' ', Qt::SkipEmptyParts);
```

```
    for (const QString &octet : octets) {
        bool ok;
        char byte = octet.toUInt(&ok, 16);
        if (ok) {
            resultat.append(byte);
        }
    }
    return resultat;
}

void GestionnaireSerie::transmettreDonneesLora(QByteArray trame) {
    if (!estOuvert()) {
        qDebug() << "Port série non ouvert !";
        return;
    }

    if (trame.isEmpty() || trame == QByteArray(trame.size(), 0x00)) {
        qDebug() << "Trame vide détectée, envoi annulé :" << trame.toHex();
        return;
    }

    QString hex = trame.toHex().toUpper();
    QString commande = "AT+MSGHEX=" + hex + "\r\n";

    envoyerCommande(commande);  // Envoi immédiat
    qDebug() << "Commande envoyée :" << commande;
}
```

ihmprincipal.cpp:

```
void IHMPrincipal::miseEnvLora()
{
    if (gestionnaireSerie && dond) {
        if (dond->estValide()) {
            QByteArray trame = dond->getDonneesPourLora();
            qDebug() << "[Envoi LoRa] Trame : " << trame.toHex();
            gestionnaireSerie->transmettreDonneesLora(trame);
        } else {
            qDebug() << "Données non valides, trame non envoyée.";
        }
    }
}


void IHMPrincipal::setGestionnaireSerie(GestionnaireSerie* gestionnaire) {
    this->gestionnaireSerie = gestionnaire;
    if (ordonnancement)
        ordonnancement->setGestionnaireSerie(gestionnaire);
}

void IHMPrincipal::on_confPort_clicked() {
    qDebug() << "on_confPort_clicked appelé";
    IHMConfigPort configDialog(this);
    if (configDialog.exec() == QDialog::Accepted) {
        QString port = configDialog.portSelectionne();
        qDebug() << "Port sélectionné :" << port;

        if (!gestionnaireSerie) {
```

```
            gestionnaireSerie = new GestionnaireSerie(this);
            connect(gestionnaireSerie, &GestionnaireSerie::donneesRecues,
this, [](QByteArray donnees){
                qDebug() << "Données LoRa reçues :" << donnees;
            });
        }
        if (gestionnaireSerie->ouvrirPort(port)) {
            gestionnaireSerie->envoyerCommandeJoin();
        }

    }
}
```

ordonnancement.cpp:

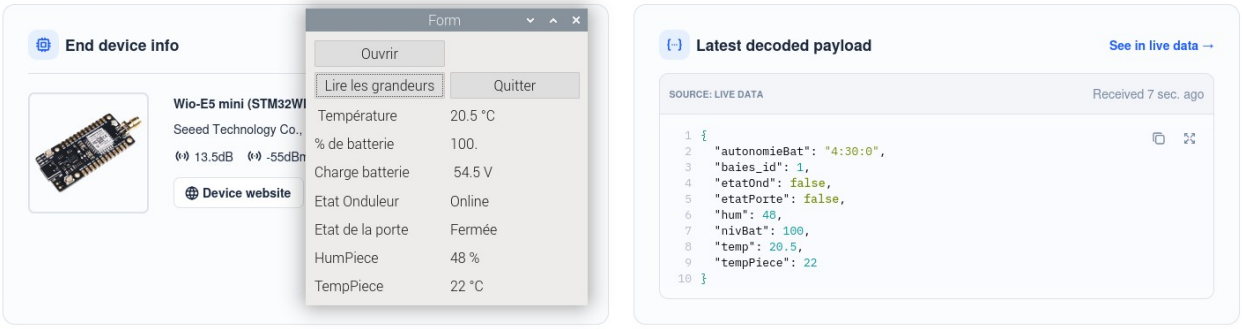
```
void Ordonnancement::run()
{
    if (!gcap || !refDon) {
        qDebug() << "Erreur : gcap ou refDon non initialisé.";
        return;
    }

    gcap->seConnecter();

    while (true) {
        qDebug() << Q_FUNC_INFO;

        gcap->lireGrandeur(refDon);

        emit miseAJIHM();
        emit miseEnvLora();

        QThread::sleep(10);  // Pause avant la prochaine lecture
    }

    gcap->seDeconnecter();
}


void Ordonnancement::setGestionnaireSerie(GestionnaireSerie* gestionnaire) {
    this->gestionnaireSerie = gestionnaire;
}

void Ordonnancement::setGestionCapteur(GestionCapteur* g) {
    gcap = g;
    gcap->setDonneeOnduleur(refDon);
}
```

Formatter Code:

```
function formatAutonomie(mins) {
  const heures = Math.floor(mins / 60);
  const minutes = mins % 60;
  const secondes = 0;
  return `${heures}:${minutes}:${secondes}`;
}

function decodeUplink(input) {
  const bytes = input.bytes;
```

```
  const autonomieMin = (bytes[4] << 8) | bytes[5];

  return {
    data: {
      baies_id: 1,
      temp: bytes[0] + (bytes[1] / 100),
      nivBat: bytes[3],
      autonomieBat: formatAutonomie(autonomieMin),
      tempPiece: bytes[6],
      hum: bytes[7],
      etatOnd: (bytes[8] & 0x02) !== 0,
      etatPorte: (bytes[8] & 0x01) !== 0
    }
  };
}
```

Résultats du test (validation de l'envoie de données):



Résultats du test (validation de l'envoie de données vers la base de données):

	id	temp	niv_bat	hum	etat_porte	etat_ond	Baies_id	temp_piece	autonomie_bat
Éditer Copier Supprimer	78	20.5	100	48	0	1	1	22	04:30:00

ANNEXE : Code de programme

gestionnaireserie.h

```
#ifndef GESTIONNAIRESERIE_H
#define GESTIONNAIRESERIE_H

#include <QObject>
#include <QSerialPort>
#include <QSerialPortInfo>

class GestionnaireSerie : public QObject {
    Q_OBJECT
public:
    explicit GestionnaireSerie(QObject *parent = nullptr);
    bool ouvrirPort(const QString &nomPort);
    void envoyerCommande(const QString &commande);
    bool estOuvert() const;
    void transmettreDonneesLora(QByteArray trame);
    void envoyerCommandeJoin();
    QByteArray convertirHexEnBytes(const QString &texte);

signals:
    void donneesRecues(const QByteArray &donnees);

private slots:
    void lireDonnees();

private:
    QSerialPort portSerie;
};

#endif // GESTIONNAIRESERIE_H
```

ihmconfigport.h

```
#ifndef IHMCONFIGPORT_H
#define IHMCONFIGPORT_H

#include <QDialog>

namespace Ui {
class IHMConfigPort;
}

class IHMConfigPort : public QDialog {
    Q_OBJECT

public:
    explicit IHMConfigPort(QWidget *parent = nullptr);
    ~IHMConfigPort();
    QString portSelectionne() const;

private slots:
```

```
    void validerEtFermer();

private:
    Ui::IHMConfigPort *ui;
};

#endif // IHMCONFIGPORT_H
```

ihmprincipal.h

```
#ifndef IHMPRINCIPAL_H
#define IHMPRINCIPAL_H

#include <QWidget>
#include <QTcpSocket>
#include "donneeonduleur.h"
#include "ordonnancement.h"
#include "gestioncapteur.h"

QT_BEGIN_NAMESPACE
namespace Ui { class IHMPrincipal; }
QT_END_NAMESPACE

class IHMPrincipal : public QWidget
{
    Q_OBJECT

public:
    IHMPrincipal(QWidget *parent = nullptr);
    ~IHMPrincipal();
    void setGestionnaireSerie(GestionnaireSerie* gestionnaire);


private slots:




    void on_Liregrandeurs_clicked();

    void on_BoutQuitter_clicked();
    void majIHM();
    void miseEnvLora();
    void on_confPort_clicked();


private:
    Ui::IHMPrincipal *ui;
    QTcpSocket *socketClientTcp;
    DonneeOnduleur *dond;
    Ordonnancement *ord;
    GestionCapteur *gcap;
    Ordonnancement* ordonnancement;
    GestionnaireSerie* gestionnaireSerie;
```

```
};
#endif // IHMPRINCIPAL_H
```

gestioncapteur.h

```
#ifndef GESTIONCAPTEUR_H
#define GESTIONCAPTEUR_H
#include "ordonnancement.h"
#include "donneeonduleur.h"
#include <QTcpSocket>
#include <QWidget>
#include <wiringPi.h>
class Ordonnancement;
class GestionCapteur
{
private:
    QTcpSocket *socketClientTcp;
    DonneeOnduleur *ref1Don;

public:
    GestionCapteur();
    ~GestionCapteur();

    void seConnecter();
    void lireGrandeur(DonneeOnduleur *);
    void seDeconnecter();
    void majIHM();
    void setDonneeOnduleur(DonneeOnduleur* d);
    int parseAutonomie(const QString& autonomieStr);

    // Méthode interne de lecture du capteur DHT11
    int lireDonneesDHT(int &temperature, int &humidity);
    QString autonomie, humidite, temp, batCharge, battVoltage, statusUPS;
    //DonneeOnduleur *refDon;  // Ajout du pointeur vers DonneeOnduleur
    QByteArray lireDonneesEnOctets();
    bool debugSansCapteur = false;

};

#endif // GESTIONCAPTEUR_H
```

ordonnancement.h

```
#ifndef ORDONNANCEMENT_H
#define ORDONNANCEMENT_H

#include <QThread>
#include <QObject>
#include <QMutex>
#include "donneeonduleur.h"
#include "gestioncapteur.h"
#include "gestionnaireserie.h"  // Ajout utile ici
```

```
class GestionCapteur;

class Ordonnancement : public QThread
{
    Q_OBJECT

private:
    DonneeOnduleur *refDon;
    GestionCapteur *gcap;
    QMutex mutex;
    GestionnaireSerie* gestionnaireSerie = nullptr;

signals:
    void miseAJIHM();
    void miseEnvLora();

public:
    Ordonnancement(DonneeOnduleur*);
    void setGestionnaireSerie(GestionnaireSerie* gestionnaire);
    void run();
    void setGestionCapteur(GestionCapteur* g);
};

#endif // ORDONNANCEMENT_H
```

gestionnaireserie.cpp

```
#include "gestionnaireserie.h"
#include <QDebug>
#include <QTimer>

GestionnaireSerie::GestionnaireSerie(QObject *parent) : QObject(parent) {
    connect(&portSerie, &QSerialPort::readyRead, this, &GestionnaireSerie::lireDonnees);
}

bool GestionnaireSerie::ouvrirPort(const QString &nomPort) {
    if (portSerie.isOpen())
        portSerie.close();

    portSerie.setPortName(nomPort);
    portSerie.setBaudRate(9600);
    portSerie.setDataBits(QSerialPort::Data8);
    portSerie.setParity(QSerialPort::NoParity);
    portSerie.setStopBits(QSerialPort::OneStop);
    portSerie.setFlowControl(QSerialPort::NoFlowControl);
    return portSerie.open(QIODevice::ReadWrite);
}

void GestionnaireSerie::envoyerCommande(const QString &commande) {
    if (portSerie.isOpen()) {
        portSerie.write(commande.toUtf8());
```

```
    }
}

void GestionnaireSerie::lireDonnees() {
    QByteArray donnees = portSerie.readAll();
    emit donneesRecues(donnees);
}

bool GestionnaireSerie::estOuvert() const {
    return portSerie.isOpen();
}

void GestionnaireSerie::envoyerCommandeJoin() {
    envoyerCommande("AT+JOIN\r\n");
}

QByteArray GestionnaireSerie::convertirHexEnBytes(const QString &texte) {
    QByteArray resultat;
    QStringList octets = texte.split(' ', Qt::SkipEmptyParts);
    for (const QString &octet : octets) {
        bool ok;
        char byte = octet.toUInt(&ok, 16);
        if (ok) {
            resultat.append(byte);
        }
    }
    return resultat;
}

void GestionnaireSerie::transmettreDonneesLora(QByteArray trame) {
    if (!estOuvert()) {
        qDebug() << "Port série non ouvert !";
        return;
    }

    if (trame.isEmpty() || trame == QByteArray(trame.size(), 0x00)) {
        qDebug() << "Trame vide détectée, envoi annulé :" << trame.toHex();
        return;
    }

    QString hex = trame.toHex().toUpper();
    QString commande = "AT+MSGHEX=" + hex + "\r\n";

    envoyerCommande(commande);  // Envoi immédiat
    qDebug() << "Commande envoyée :" << commande;
}
```

ihmconfigport.cpp

```
#include "ihmconfigport.h"
#include "ui_ihmconfigport.h"
#include <QSerialPortInfo>
```

```
IHMConfigPort::IHMConfigPort(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::IHMConfigPort) {
    ui->setupUi(this);

    foreach (const QSerialPortInfo &info, QSerialPortInfo::availablePorts()) {
        ui->listePorts->addItem(info.portName());
    }

    connect(ui->boutonOk, &QPushButton::clicked, this, &IHMConfigPort::validerEtFermer);
    connect(ui->boutonAnnuler, &QPushButton::clicked, this, &IHMConfigPort::reject);
}

IHMConfigPort::~IHMConfigPort() {
    delete ui;
}

QString IHMConfigPort::portSelectionne() const {
    return ui->listePorts->currentText();
}

void IHMConfigPort::validerEtFermer() {
    accept();
}
```

Ihmprincipal,cpp

```
#include "ihmprincipal.h"
#include "ui_ihmprincipal.h"
#include "ihmconfigport.h"
#include <QPushButton>
#include <QTcpSocket>
#include <iostream>
#include <QString>
#include <QDebug>
#include <QThread>
#include <QTimer>
#include "ordonnancement.h"
#include "donneeonduleur.h"
#include "gestioncapteur.h"
#include <wiringPi.h>
#include "gestionnaireserie.h"


IHMPrincipal::IHMPrincipal(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::IHMPrincipal)
{

    ui->setupUi(this);


    dond = new DonneeOnduleur;
```

```
    gcap = new GestionCapteur;

    ordonnancement = new Ordonnancement(dond);
    ordonnancement->setGestionCapteur(gcap);

    connect(ordonnancement, SIGNAL(miseAJIHM()), this, SLOT(majIHM()));
    connect(ordonnancement, &Ordonnancement::miseEnvLora, this, &IHMPrincipal::miseEnvLora);
    socketClientTcp = new QTcpSocket(this);

    if (wiringPiSetup() == -1) {
        qDebug() << "Erreur lors de l'initialisation de wiringPi.";
    }
}

IHMPrincipal::~IHMPrincipal()
{
 // Libère la mémoire
        delete ordonnancement; // Libérer la mémoire du thread
        delete dond;
        delete ui;
        delete gcap;
}


void IHMPrincipal::on_Liregrandeurs_clicked()
{
    qDebug() << "ordonnancement =" << ordonnancement;
    qDebug() << "gcap =" << gcap;

    if (!ordonnancement->isRunning())
        ordonnancement->start();

    if (!gestionnaireSerie || !gestionnaireSerie->estOuvert()) {
        qDebug() << "Port série non initialisé ou fermé.";
        return;
       }
    QTimer::singleShot(20000, this, [=]() {
        QByteArray donnees = gcap->lireDonneesEnOctets();
        qDebug() << "Après 2s, donnees capteurs = " << donnees.toHex().toUpper();

        if (!donnees.isEmpty()) {
            gestionnaireSerie->transmettreDonneesLora(donnees);
        } else {
            qDebug() << "Aucune donnée capteur disponible.";
        }
      });
}



void IHMPrincipal::on_BoutQuitter_clicked()
{
    close();
```

```
}

void IHMPrincipal::majIHM()
{
    qDebug() << Q_FUNC_INFO;
    ui->Valtemp1->setText(dond->getTemp() + " °C");
    qDebug() << "Valeur de la température"<< dond->getTemp();
    ui->Valtempbat->setText(dond->getbatCharge());
    ui->Valchargebat->setText(dond->getbattVoltage() + " V");
    ui->Valonduleur->setText(dond->getstatusUPS());
    if(dond->getu2_AStatus()==false)ui->labelPorte->setText("Fermée");
    if(dond->getu2_AStatus()==true)ui->labelPorte->setText("Ouverte");

// 🏠 Température et humidité pièce (capteur DHT11)
    ui->TempPiece->setText(dond->getTempPiece() + " °C");
    ui->HumPiece->setText(dond->getHumPiece() + " %");
}

void IHMPrincipal::miseEnvLora()
{
    if (gestionnaireSerie && dond) {
        if (dond->estValide()) {
            QByteArray trame = dond->getDonneesPourLora();
            qDebug() << "[Envoi LoRa] Trame : " << trame.toHex();
            gestionnaireSerie->transmettreDonneesLora(trame);
        } else {
            qDebug() << "Données non valides, trame non envoyée.";
        }
    }
}


void IHMPrincipal::setGestionnaireSerie(GestionnaireSerie* gestionnaire) {
    this->gestionnaireSerie = gestionnaire;
    if (ordonnancement)
        ordonnancement->setGestionnaireSerie(gestionnaire);
}

void IHMPrincipal::on_confPort_clicked() {
    qDebug() << "on_confPort_clicked appelé";
    IHMConfigPort configDialog(this);
    if (configDialog.exec() == QDialog::Accepted) {
        QString port = configDialog.portSelectionne();
        qDebug() << "Port sélectionné :" << port;

        if (!gestionnaireSerie) {
            gestionnaireSerie = new GestionnaireSerie(this);
            connect(gestionnaireSerie, &GestionnaireSerie::donneesRecues, this, [](QByteArray donnees){
                qDebug() << "Données LoRa reçues :" << donnees;
            });
        }
        if (gestionnaireSerie->ouvrirPort(port)) {
            gestionnaireSerie->envoyerCommandeJoin();
```

```
    }

  }
}
```

donneeonduleur.cpp

```
#include "donneeonduleur.h"
#include <QDebug>


DonneeOnduleur::DonneeOnduleur()
{

}

DonneeOnduleur::DonneeOnduleur(QString irefTemp, QString irefbatCharge)
{
   stemp = irefTemp;
   sbatcharge = irefbatCharge;
}

void DonneeOnduleur::setTemp(QString itemp)
{
   stemp = itemp;

}

QString DonneeOnduleur::getTemp()
{
   return stemp;

}

void DonneeOnduleur::setbatCharge(QString ibatcharge)
{
   sbatcharge = ibatcharge;
}

QString DonneeOnduleur::getbatCharge()
{
   return sbatcharge;
}

void DonneeOnduleur::setbattVoltage(QString ibattVoltage)
{
   sbattVoltage = ibattVoltage;
}

QString DonneeOnduleur::getbattVoltage()
{
   return sbattVoltage;
```

```
}

void DonneeOnduleur::setstatusUPS(QString istatusUPS)
{
   sstatusUPS = istatusUPS;
}

QString DonneeOnduleur::getstatusUPS()
{
   return sstatusUPS;
}

void DonneeOnduleur::setValonduleur(QString iValonduleur)
{
   sValonduleur = iValonduleur;
}

QString DonneeOnduleur::getValonduleur()
{
   return sValonduleur;
}

void DonneeOnduleur::setAutonomie(QString iAutonomie)
{
   sAutonomie = iAutonomie;
}

QString DonneeOnduleur::getAutonomie()
{
   return sAutonomie;
}

void DonneeOnduleur::setHumidite(QString iHumidite)
{
   sHumidite = iHumidite;
}

QString DonneeOnduleur::getHumidite()
{
   return sHumidite;
}

void DonneeOnduleur::setu2_AStatus(bool iu2_AStatus)
{
   su2_AStatus = iu2_AStatus;
}

bool DonneeOnduleur::getu2_AStatus()
{
   return su2_AStatus;
}

void DonneeOnduleur::setTempPiece(QString iTempPiece)
```

```
{
    sTempPiece = iTempPiece;
}

QString DonneeOnduleur::getTempPiece()
{
    return sTempPiece;
}

void DonneeOnduleur::setHumPiece(QString iHumPiece)
{
    sHumPiece = iHumPiece;
}

QString DonneeOnduleur::getHumPiece()
{
    return sHumPiece;
}

int DonneeOnduleur::convertirAutonomieEnMinutes(const QString& autonomieStr) {
    int heures = 0;
    int minutes = 0;

    QRegExp regexHeures("(\\d+)\\s*hr");
    QRegExp regexMinutes("(\\d+)\\s*min");

    if (regexHeures.indexIn(autonomieStr) != -1) {
        heures = regexHeures.cap(1).toInt();
    }
    if (regexMinutes.indexIn(autonomieStr) != -1) {
        minutes = regexMinutes.cap(1).toInt();
    }

    int totalMinutes = heures * 60 + minutes;
    if (totalMinutes > 65535) totalMinutes = 65535; // borne max 16 bits

    return totalMinutes;
}



QByteArray DonneeOnduleur::getDonneesPourLora() {
    QByteArray trame;
    bool ok = false;

    float tempF = stemp.trimmed().toFloat(&ok);
    if (!ok) tempF = 0;
    int tempEntier = static_cast<int>(tempF);
    int tempCentieme = static_cast<int>((tempF - tempEntier) * 100);

    int hum = sHumidite.trimmed().toFloat(&ok);
    if (!ok) hum = 0;
```

```
    int charge = sbatcharge.trimmed().toFloat(&ok);
    if (!ok) charge = 0;

    int autoUPS = convertirAutonomieEnMinutes(sAutonomie);

    int tempPiece = sTempPiece.trimmed().toFloat(&ok);
    if (!ok) tempPiece = 0;

    int humPiece = sHumPiece.trimmed().toFloat(&ok);
    if (!ok) humPiece = 0;

    bool etatOnd = sValonduleur.trimmed().toFloat(&ok) > 0;
    char flags = 0;
    if (etatOnd) flags |= 0x02;
    if (su2_AStatus) flags |= 0x01;

    // DEBUG
    qDebug() << "[Trame LoRa]"
             << "Temp:" << tempF
             << "Hum:" << hum
             << "Charge:" << charge
             << "Auto (min):" << autoUPS
             << "TempPièce:" << tempPiece
             << "HumPièce:" << humPiece
             << "EtatOnd:" << etatOnd
             << "EtatPorte:" << su2_AStatus;

    trame.append(static_cast<unsigned char>(tempEntier));
    trame.append(static_cast<unsigned char>(tempCentieme));
    trame.append(static_cast<unsigned char>(hum));
    trame.append(static_cast<unsigned char>(charge));

    trame.append(static_cast<unsigned char>((autoUPS >> 8) & 0xFF));
    trame.append(static_cast<unsigned char>(autoUPS & 0xFF));

    trame.append(static_cast<unsigned char>(tempPiece));
    trame.append(static_cast<unsigned char>(humPiece));
    trame.append(flags);

    return trame;
}




bool DonneeOnduleur::estValide() {
    return !stemp.isEmpty() && !sbattVoltage.isEmpty(); // ou toute autre logique de validation
}
```

gestioncapteur.cpp

```
#include "gestioncapteur.h"
#include <QTcpSocket>
#include "donneeonduleur.h"
#include <QDebug>
#include "ordonnancement.h"
#include <QThread>
#include <wiringPi.h>

GestionCapteur::GestionCapteur()
{

   socketClientTcp = new QTcpSocket;

}


void GestionCapteur::seConnecter()
{
   if (socketClientTcp->waitForConnected() != true)
   {
      socketClientTcp->connectToHost("10.10.203.200", 23, QIODevice::ReadWrite);
      {
         qDebug() << "La connexion avec l'hote " << "10.10.203.200 est impossible";
         qDebug() << "L'etat de connexion est : " << socketClientTcp->state();
         QThread::msleep(1000);
         socketClientTcp->waitForBytesWritten(3000);
         socketClientTcp->write("apc\r\n");
         socketClientTcp->waitForBytesWritten(3000);
         socketClientTcp->write("apc\r\n");

         QThread::msleep(1000);
      }
   }
   else
   {
      qDebug() << "La connexion avec l'hote 10.10.203.200 est effective";
      qDebug() << "L'etat de connexion est : " << socketClientTcp->state();

     //  socketClientTcp->waitForBytesWritten(3000);
     //  socketClientTcp->write("detstatus -all\r\n");
     //  socketClientTcp->write("uio -st\r\n");

   }
}

void GestionCapteur::lireGrandeur(DonneeOnduleur *iref1Dond)
{
   qDebug() << "Test1";
  qDebug() << Q_FUNC_INFO << "appelé avec DonneeOnduleur*" << iref1Dond;

      if (iref1Dond == nullptr) {
         qDebug() << "✗ DonneeOnduleur (iref1Dond) est NULL, arrêt.";
         return;
```

```
    }
ref1Don = iref1Dond;

QThread::msleep(1000);
socketClientTcp->waitForBytesWritten(3000);
socketClientTcp->write("detstatus -all\r\n");
socketClientTcp->write("uio -st\r\n");


if (socketClientTcp->waitForReadyRead(3000))
{
    QByteArray recept = socketClientTcp->readAll();
    qDebug() << "Réponse reçue : " << recept;


    if (recept.contains("Runtime Remaining:"))
    {
        int startPos = recept.indexOf("Runtime Remaining:") + 18;
        int endPos = recept.indexOf(" sec", startPos);
        autonomie = recept.mid(startPos, endPos - startPos);
        qDebug() << "Autonomie:" <<     autonomie;
        ref1Don->setAutonomie(autonomie);
    }

    if (recept.contains("%RH"))
    {
        int startPos = recept.indexOf(":OK:") + 4;
        int endPos = recept.indexOf("%RH", startPos);
        humidite = recept.mid(startPos, endPos - startPos);
       ref1Don->setHumidite(humidite);
    }

    if (recept.contains("U1:"))
    {
        temp = recept.mid(recept.indexOf("U1:") + 3, 4);
       ref1Don->setTemp(temp);
    }

    if (recept.contains("Battery State Of Charge:"))
    {
        batCharge = recept.mid(recept.indexOf("Battery State Of Charge:") + 24, 5);
                    qDebug() << "batterie de charge:" <<     batCharge;
      ref1Don->setbatCharge(batCharge.trimmed());
    }

    if (recept.contains("Battery Voltage:"))
    {
        battVoltage = recept.mid(recept.indexOf("Battery Voltage:") + 16, 5);
                qDebug() << "batterie voltage:" <<     battVoltage;
       ref1Don->setbattVoltage(battVoltage);
    }

    if (recept.contains("Status of UPS:"))
```

```
    {
        statusUPS = recept.mid(recept.indexOf("Status of UPS:") + 15, 6);
        ref1Don->setstatusUPS(statusUPS);
    }

    if (recept.contains("U2_A:"))
    {
        int startIndex = recept.indexOf("U2_A:") + 6;
        QString u2_AStatus = recept.mid(startIndex - 1, 1);  // "O" ou "C"
        qDebug() << "▨ État U2_A (brut) :" << u2_AStatus;

        if (u2_AStatus == "O") {
            ref1Don->setu2_AStatus(true);
            qDebug() << "ouverte";
        } else if (u2_AStatus == "C") {
            ref1Don->setu2_AStatus(false);
            qDebug() << "fermée";
        } else {
            qDebug() << "état inconnu";
        }
    }
        // ▼ AJOUT ICI avant de fermer le bloc
    int tempPiece = 0, humPiece = 0;
    if (lireDonneesDHT(tempPiece, humPiece) == 0) {
        qDebug() << "Température pièce (DHT11) : " << tempPiece;
        qDebug() << "Humidité pièce (DHT11) : " << humPiece;
        ref1Don->setTempPiece(QString::number(tempPiece));
        ref1Don->setHumPiece(QString::number(humPiece));
    } else {
        qDebug() << "Erreur de lecture du capteur DHT11.";
    }
        } // ✔ fin du `if (waitForReadyRead)`
        else
        {
    qDebug() << "Erreur : Aucun retour de l'hôte.";
  }
}

void GestionCapteur::seDeconnecter()
{
  socketClientTcp->disconnectFromHost();
  qDebug() << "Déconnecté de l'hôte.";
}

int GestionCapteur::lireDonneesDHT(int &temperature, int &humidity)
{
  if (wiringPiSetup() == -1) {
      qDebug() << "Erreur : wiringPiSetup() a échoué.";
      return -1;
  }

  unsigned char data[5] = {0}; // Tableau pour stocker les 5 octets du capteur
    int bitIndex = 0;
```

```
const int DHT_PIN = 7; // Broche GPIO selon wiringPi (GPIO 4 physique 7)

// Étape 1 : Demande de communication
pinMode(DHT_PIN, OUTPUT);
digitalWrite(DHT_PIN, LOW);
delay(18); // signal bas pendant 18 ms
digitalWrite(DHT_PIN, HIGH);
delayMicroseconds(40);
pinMode(DHT_PIN, INPUT); // bascule en mode lecture

// Étape 2 : Attente réponse capteur
unsigned int loopCnt = 10000;
while (digitalRead(DHT_PIN) == LOW) {
    if (loopCnt-- == 0) return -1; // Timeout
}

loopCnt = 10000;
while (digitalRead(DHT_PIN) == HIGH) {
    if (loopCnt-- == 0) return -1; // Timeout
}

// Étape 3 : Lecture des 40 bits (5 octets)
for (int i = 0; i < 40; i++) {
    // Attente que le signal devienne haut
    loopCnt = 10000;
    while (digitalRead(DHT_PIN) == LOW) {
        if (loopCnt-- == 0) return -1;
    }

    // Mesure de la durée du signal haut
    delayMicroseconds(30);
    if (digitalRead(DHT_PIN) == HIGH) {
        data[i / 8] |= (1 << (7 - (i % 8))); // Si 1, on place le bit
    }

    // Attente que le signal redescende
    loopCnt = 10000;
    while (digitalRead(DHT_PIN) == HIGH) {
        if (loopCnt-- == 0) return -1;
    }
}

// Étape 4 : Vérification du checksum
int checksum = data[0] + data[1] + data[2] + data[3];
if (data[4] != (checksum & 0xFF)) {
    qDebug() << "Checksum invalide.";
    return -1;
}

// Étape 5 : Extraction des valeurs
humidity = data[0];
temperature = data[2];
```

```
    return 0; // OK
}
GestionCapteur::~GestionCapteur()
{
   delete socketClientTcp;

}

// Nouvelle fonction pour parser autonomie en minutes, renvoie int
int parseAutonomieEnMinutes(const QString& autonomieStr) {
   int heures = 0;
   int minutes = 0;

   QRegExp regexHeures("(\\d+)\\s*hr");
   QRegExp regexMinutes("(\\d+)\\s*min");

   if (regexHeures.indexIn(autonomieStr) != -1) {
      heures = regexHeures.cap(1).toInt();
   }
   if (regexMinutes.indexIn(autonomieStr) != -1) {
      minutes = regexMinutes.cap(1).toInt();
   }

   return heures * 60 + minutes;  // Total en minutes
}



QByteArray GestionCapteur::lireDonneesEnOctets()
{
   QByteArray donnees;
   if (!ref1Don) {
      qDebug() << "✗ ref1Don est null !";
      return donnees;
   }
   bool ok;

   float temp = ref1Don->getTemp().toFloat(&ok);
   if (!ok) temp = 0;
   int tempEntier = static_cast<int>(temp);
   int tempCentieme = static_cast<int>((temp - tempEntier) * 100);
   donnees.append(static_cast<unsigned char>(tempEntier));
   donnees.append(static_cast<unsigned char>(tempCentieme));

   int hum = ref1Don->getHumidite().toInt(&ok);
   donnees.append(ok ? static_cast<unsigned char>(hum) : 0);

   float chargeFloat = ref1Don->getbatCharge().toFloat(&ok);
   int charge = ok ? static_cast<int>(chargeFloat) : 0;
   donnees.append(static_cast<unsigned char>(charge));

   QString autonomieStr = ref1Don->getAutonomie();
   qDebug() << "Chaîne autonomie reçue :" << autonomieStr;
```

```
    int autonomieMin = parseAutonomieEnMinutes(autonomieStr);

    donnees.append(static_cast<unsigned char>((autonomieMin >> 8) & 0xFF));
    donnees.append(static_cast<unsigned char>(autonomieMin & 0xFF));

    int tempPiece = ref1Don->getTempPiece().toInt(&ok);
    donnees.append(ok ? static_cast<unsigned char>(tempPiece) : 0);

    int humPiece = ref1Don->getHumPiece().toInt(&ok);
    donnees.append(ok ? static_cast<unsigned char>(humPiece) : 0);

    bool upsStatus = ref1Don->getstatusUPS().contains("ON", Qt::CaseInsensitive);
    bool porte = ref1Don->getu2_AStatus();
    char flags = 0;
    if (upsStatus) flags |= 0x02;
    if (porte) flags |= 0x01;
    donnees.append(flags);

    qDebug() << "Données LoRa (hex): " << donnees.toHex(' ').toUpper();

    return donnees;
}



void GestionCapteur::setDonneeOnduleur(DonneeOnduleur* d) {
    ref1Don = d;
}
```

ordonnancement.cpp

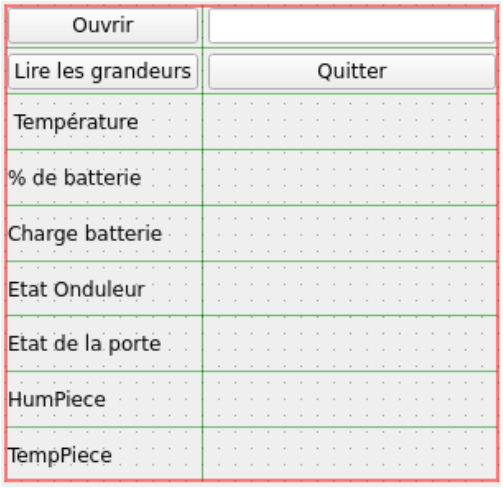
```
#include "ordonnancement.h"
#include <QThread>
#include <QDebug>
#include "donneeonduleur.h"
#include "ihmprincipal.h"
#include "QString"
#include "gestioncapteur.h"


Ordonnancement::Ordonnancement(DonneeOnduleur *irefDon)
{
    refDon = irefDon;
    gcap = nullptr;  // ne pas initialiser ici directement
}


void Ordonnancement::run()
{
    if (!gcap || !refDon) {
        qDebug() << "Erreur : gcap ou refDon non initialisé.";
        return;
    }
```

```
    gcap->seConnecter();

    while (true) {
        qDebug() << Q_FUNC_INFO;

        // mutex.lock();  // Décommenter si gcap est partagé entre threads
        gcap->lireGrandeur(refDon);

        emit miseAJIHM();
        emit miseEnvLora();
        // mutex.unlock();

        QThread::sleep(10);  // Pause avant la prochaine lecture
    }

    gcap->seDeconnecter();
}


void Ordonnancement::setGestionnaireSerie(GestionnaireSerie* gestionnaire) {
    this->gestionnaireSerie = gestionnaire;
}

void Ordonnancement::setGestionCapteur(GestionCapteur* g) {
    gcap = g;
    gcap->setDonneeOnduleur(refDon);
}
```

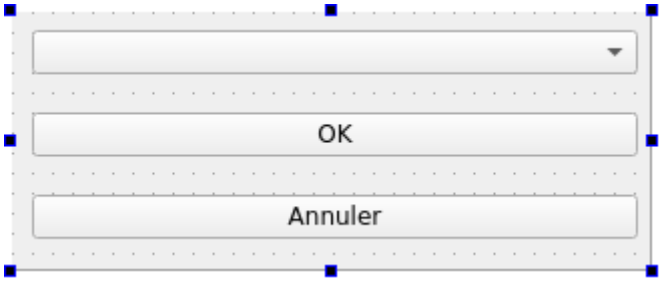
main.cpp

```
#include <QApplication>
#include <QDebug>
#include "ihmprincipal.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    IHMPrincipal* w = new IHMPrincipal();

    w->show();
    return a.exec();
}
```

ihmprincipal.ui



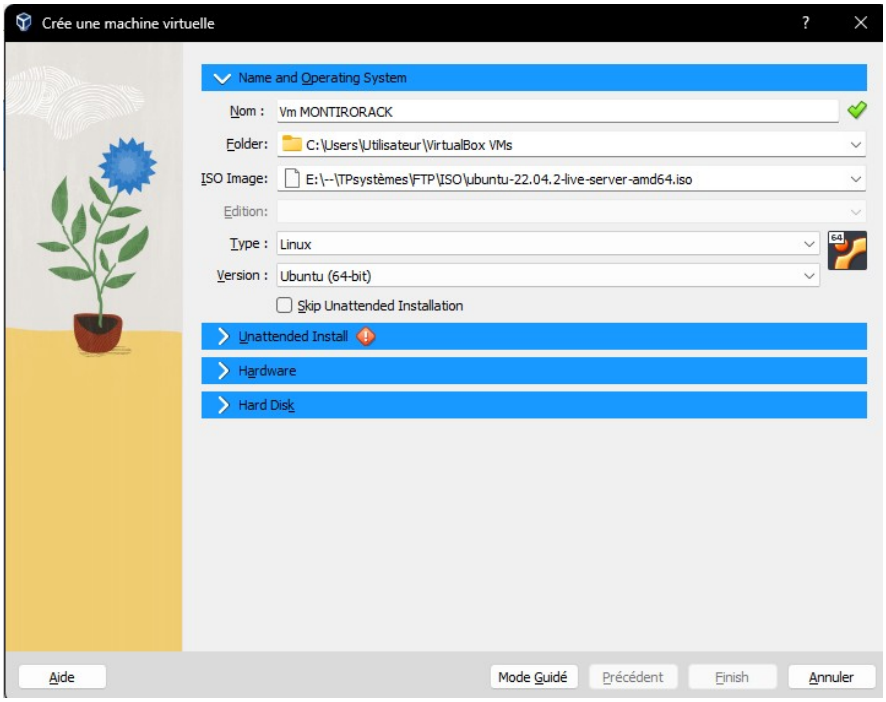
Ihmconfigport.ui



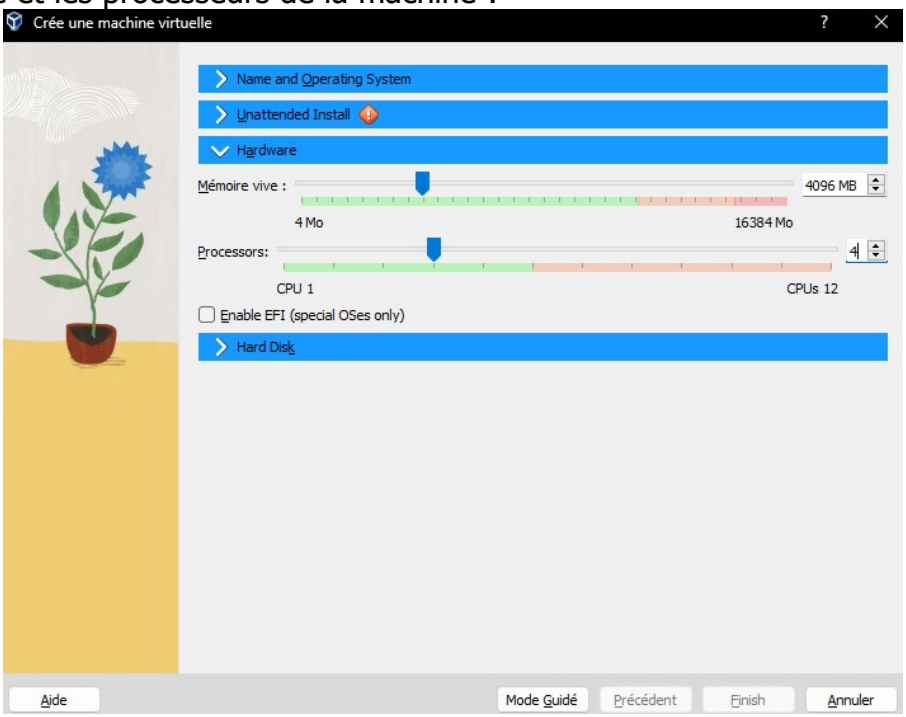
12.3. Annexes Youssif MATTI YOUSIF

Création d'une machine virtuelle sous VirtualBox :

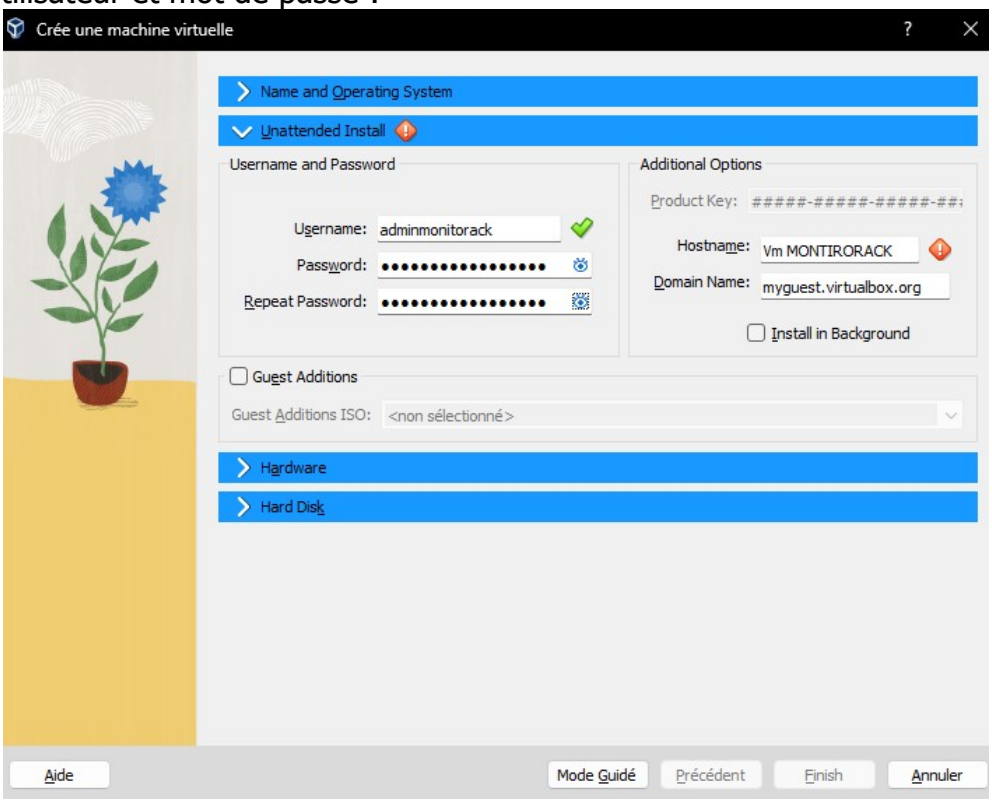
Informations de la machine :



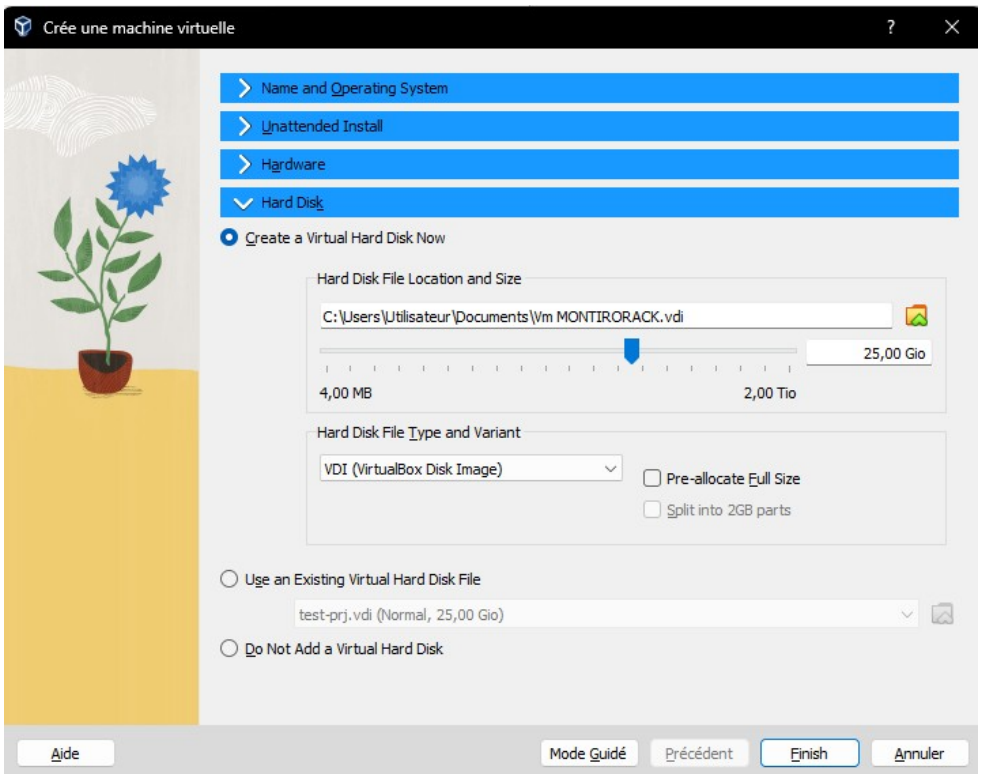
La mémoire et les processeurs de la machine :



Nom d'utilisateur et mot de passe :

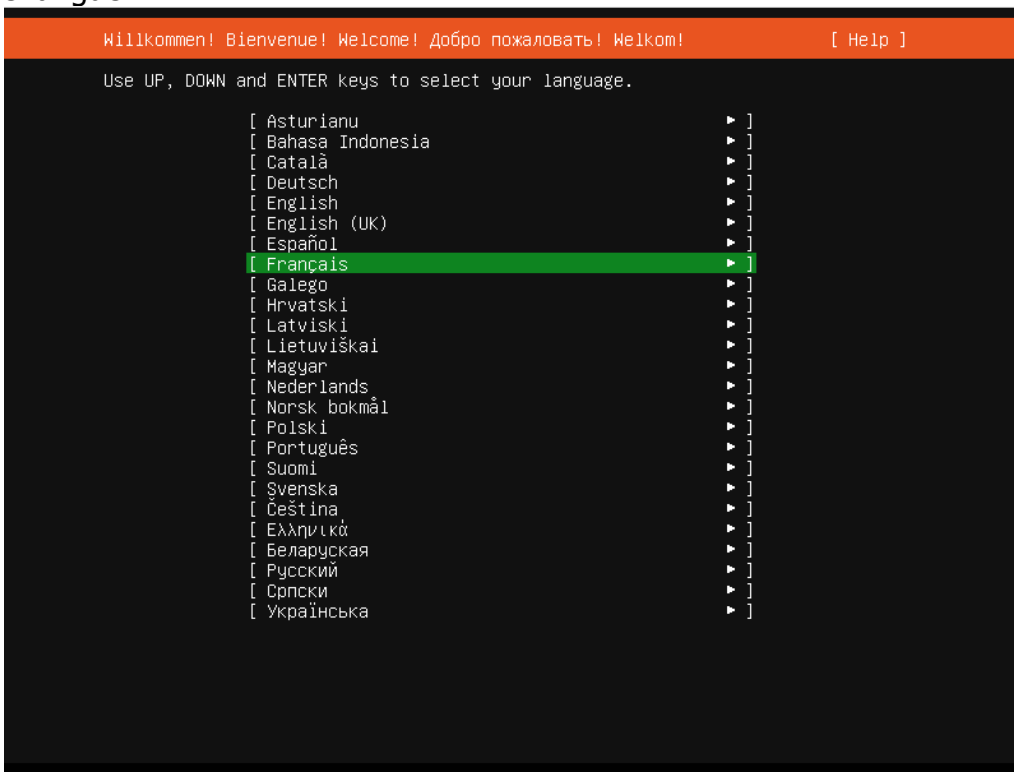


La taille de la machine :

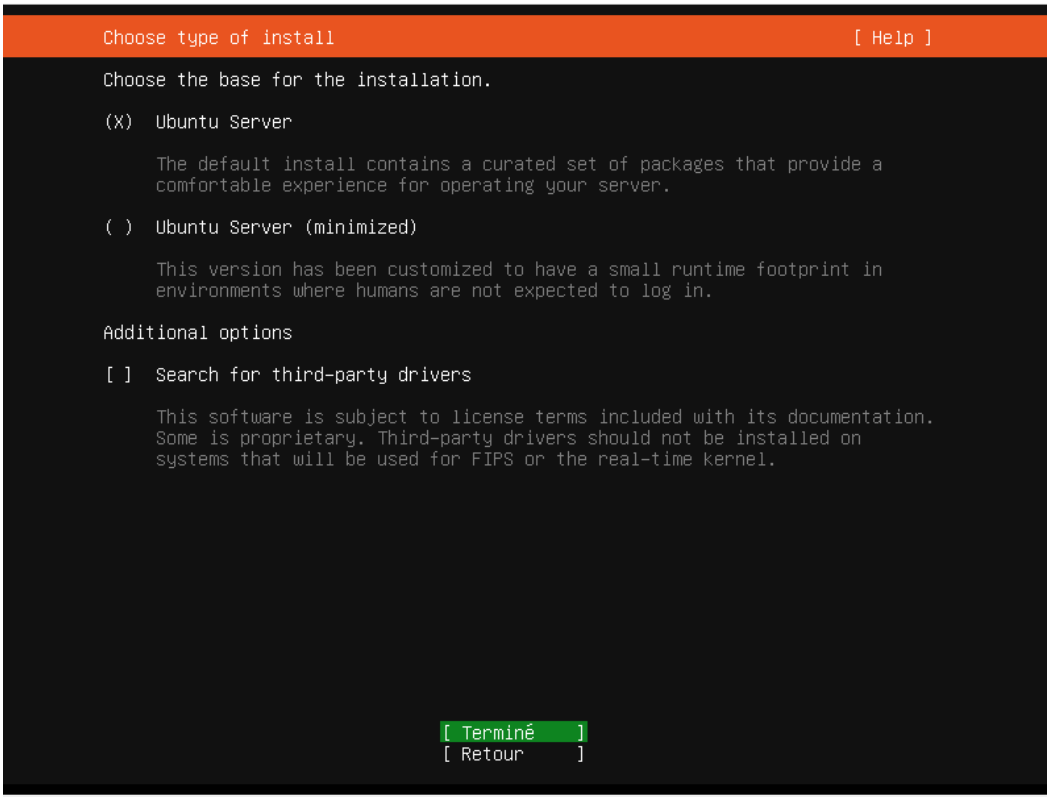


Installation de la machine virtuelle :

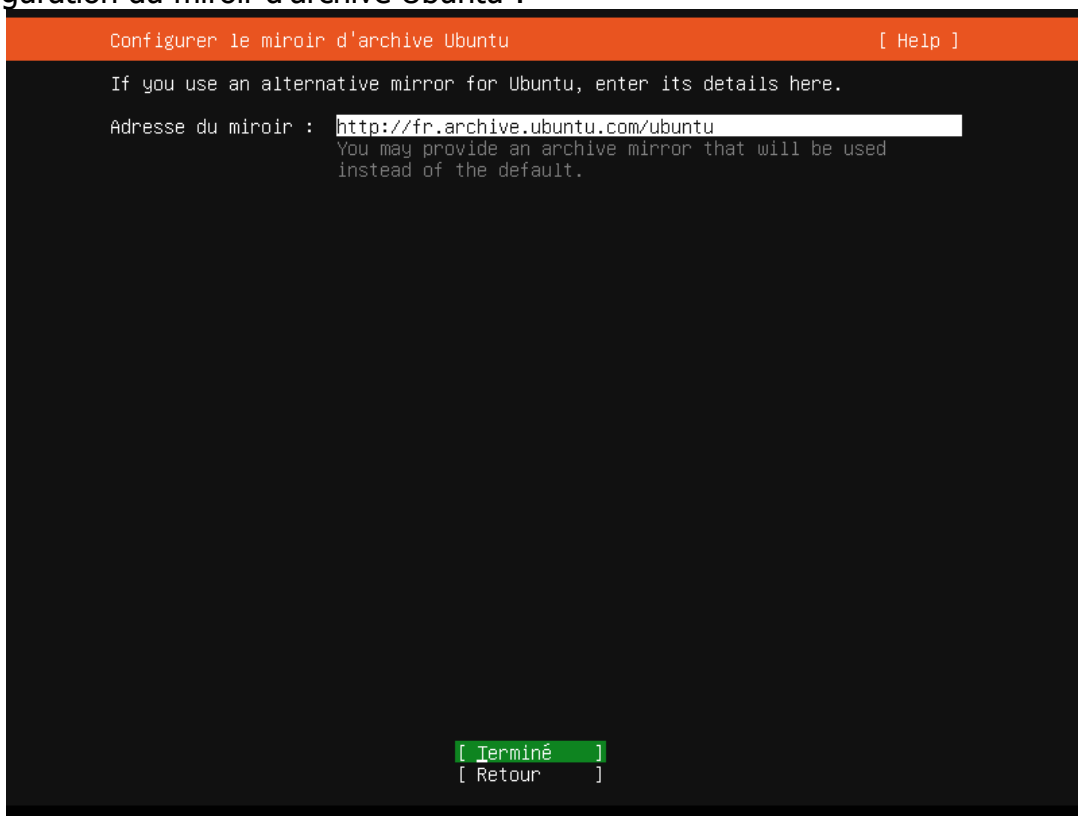
Choix de langue :



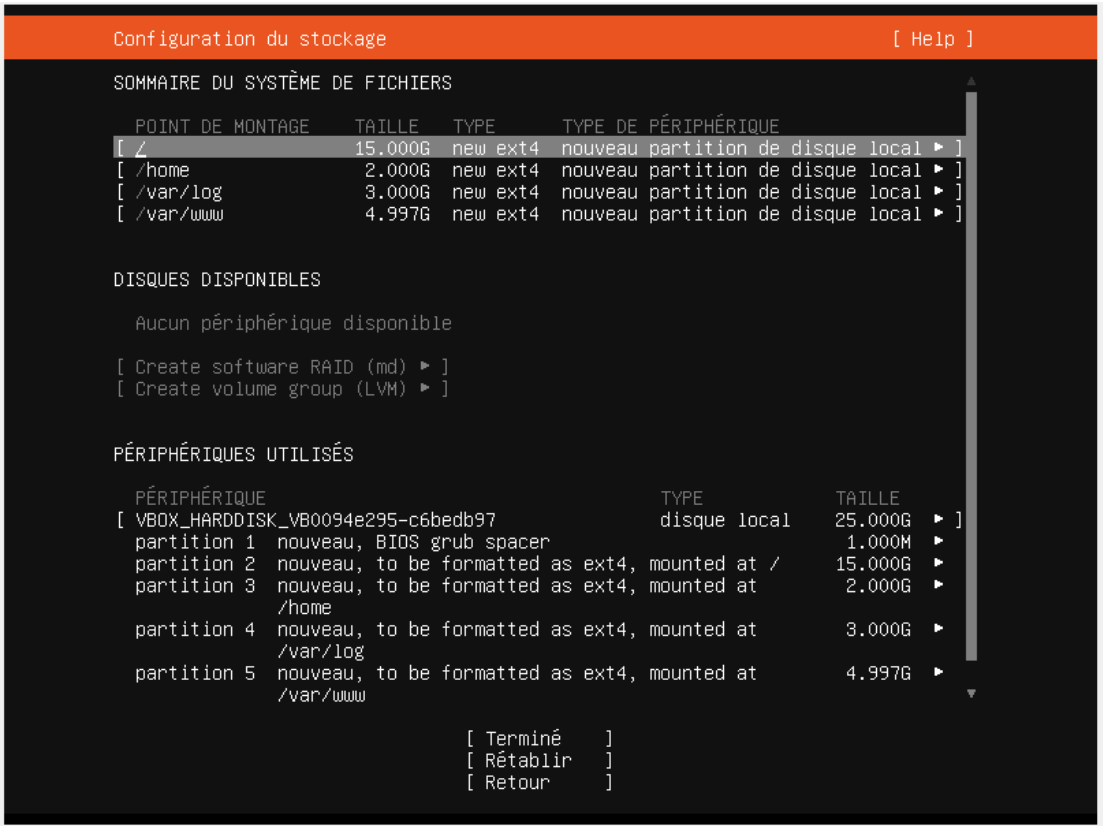
Type d'installation :



Configuration du miroir d'archive Ubuntu :



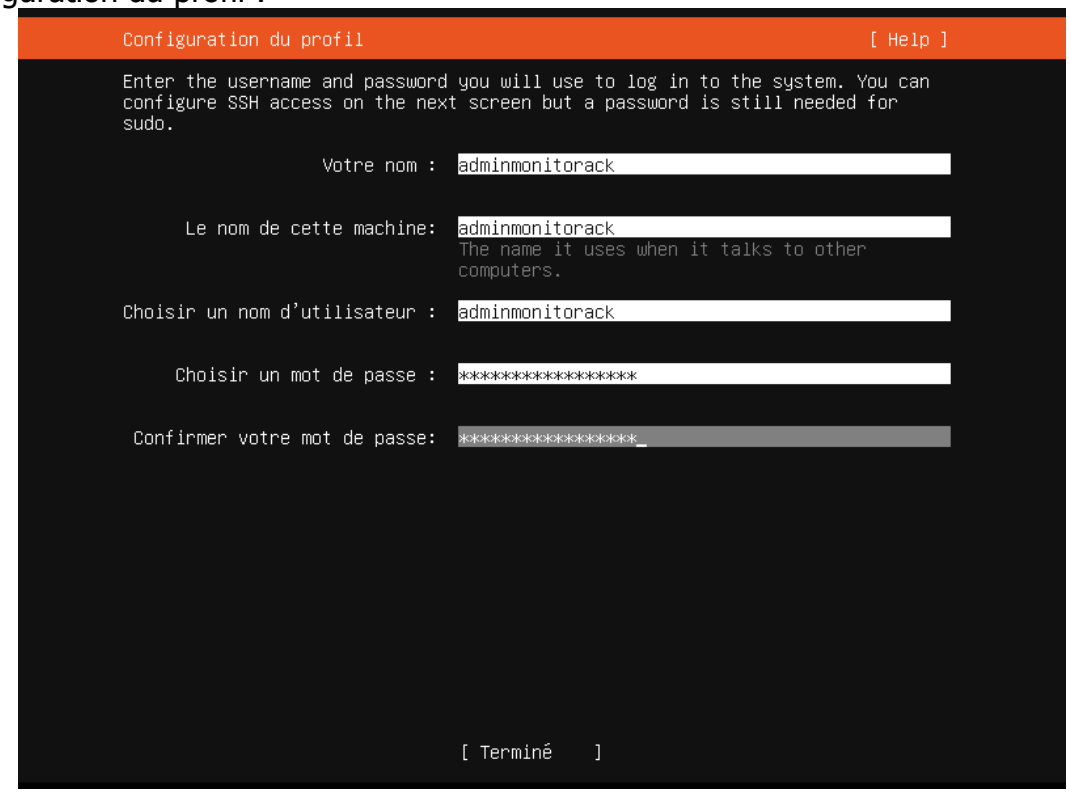
Partitionnement de la machine :



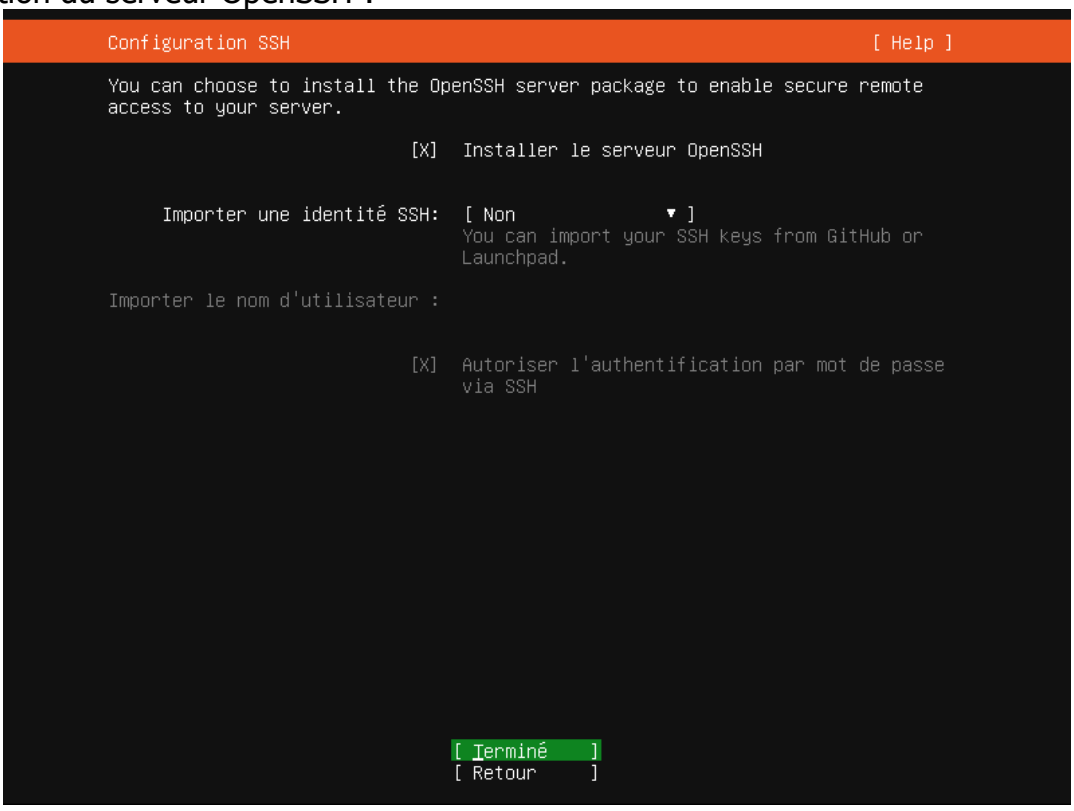
Recommandations de l'ANSSI pour le partitionnement :

Point de montage	Options	Description
/	<sans option>	Partition racine, contient le reste de l'arborescence
/boot	nosuid,nodev,noexec (noauto optionnel)	Contient le noyau et le chargeur de démarrage. Pas d'accès nécessaire une fois le boot terminé (sauf mise à jour)
/opt	nosuid,nodev (ro optionnel)	Packages additionnels au système. Montage en lecture seule si non utilisé
/tmp	nosuid,nodev,noexec	Fichiers temporaires. Ne doit contenir que des éléments non exécutables. Nettoyé après redémarrage ou préférablement de type tmpfs
/srv	nosuid,nodev (noexec,ro optionnels)	Contient des fichiers servis par un service type Web, FTP…
/home	nosuid,nodev,noexec	Contient les HOME utilisateurs.
/proc	hidepid=2 [14]	Contient des informations sur les processus et le système
/usr	nodev	Contient la majorité des utilitaires et fichiers système
/var	nosuid,nodev,noexec	Partition contenant des fichiers variables pendant la vie du système (mails, fichiers PID, bases de données d'un service)
/var/log [15]	nosuid,nodev,noexec	Contient les logs du système
/var/tmp	nosuid,nodev,noexec	Fichiers temporaires conservés après extinction

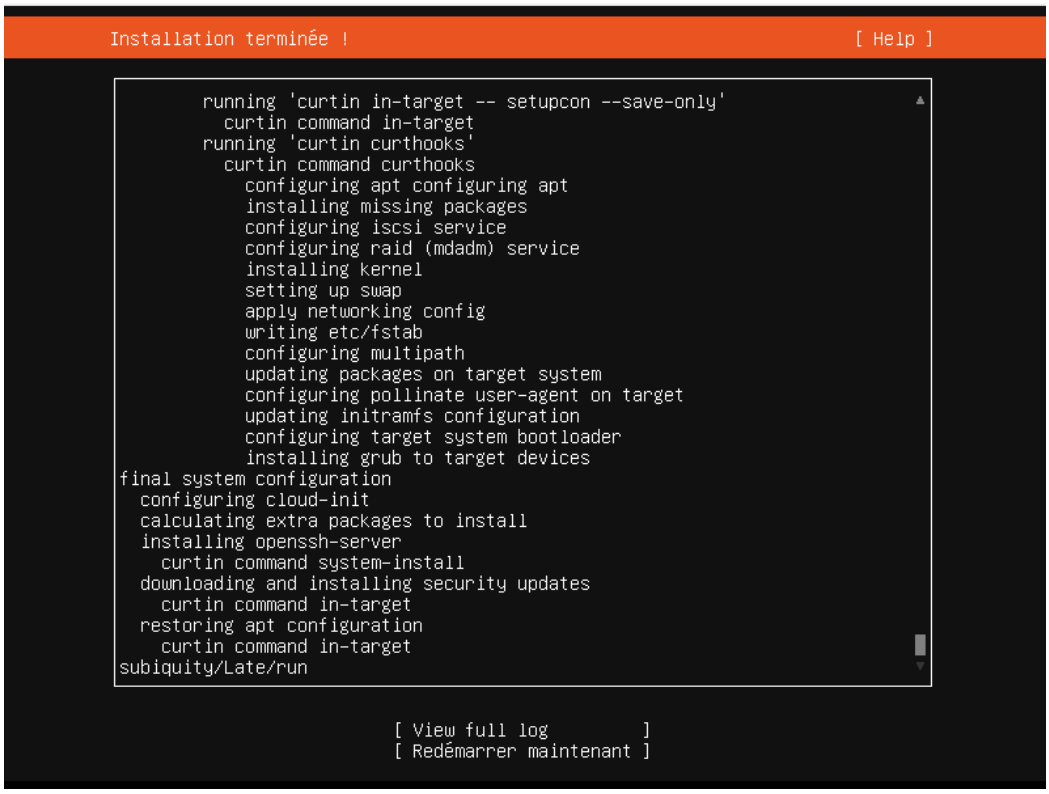
Configuration du profil :



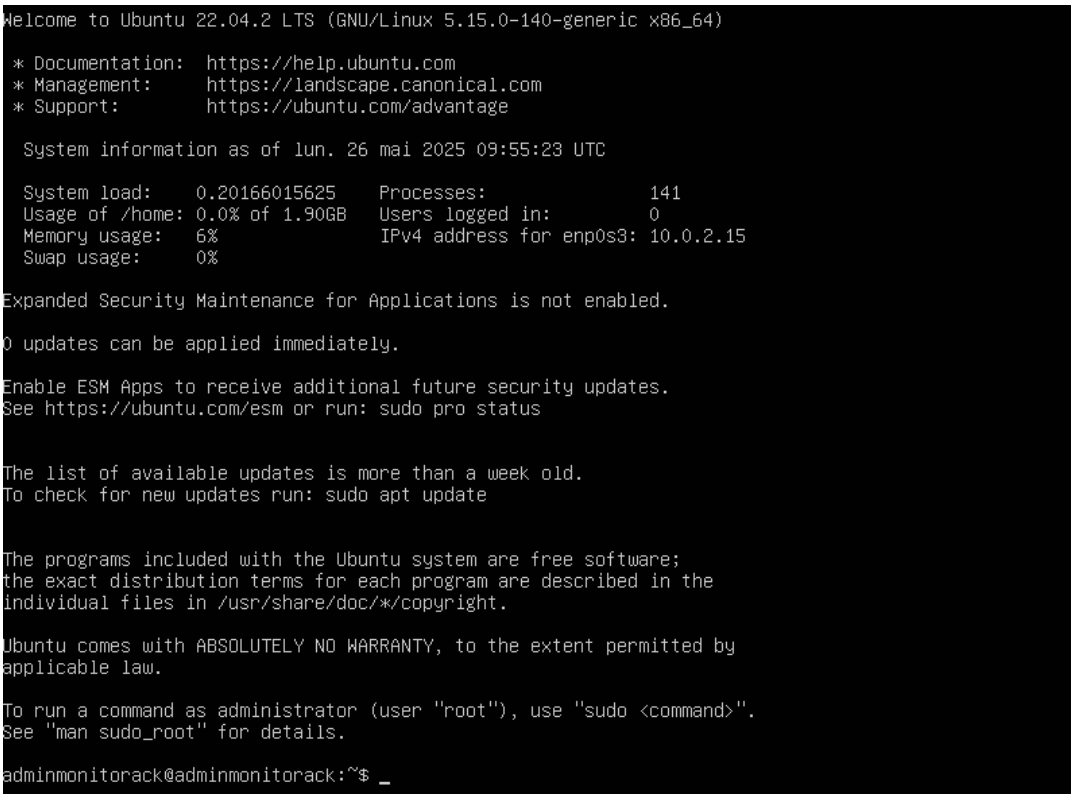
Installation du serveur OpenSSH :



Finaliser l'installation de la machine :



Test de fonctionnement :



J'ai commencer par faire les mises a jours :

```
adminmonitorack@adminmonitorack:~$ sudo apt upgrade
[sudo] password for adminmonitorack:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
Les NOUVEAUX paquets suivants seront installés :
  python3-packaging ubuntu-pro-client ubuntu-pro-client-l10n
Les paquets suivants seront mis à jour :
  apport apt apt-utils base-files cloud-init coreutils cryptsetup cryptsetup-bin
  cryptsetup-initramfs distro-info distro-info-data dmeventd dmidecode dmsetup dpkg e2fsprogs
  ethtool fwupd fwupd-signed gir1.2-packagekitglib-1.0 initramfs-tools initramfs-tools-bin
  initramfs-tools-core iptables irqbalance isc-dhcp-client isc-dhcp-common kpartx landscape-common
  libapt-pkg6.0 libcom-err2 libcryptsetup12 libdevmapper-event1.02.1 libdevmapper1.02.1 libext2fs2
  libfwupd2 libfwupdplugin5 libgpgme11 libip4tc2 libip6tc2 libldap-2.5-0 libldap-common
  liblvm2cmd2.03 libmbim-glib4 libmbim-proxy libmm-glib0 libnss-systemd libopeniscsiusr
  libpackagekit-glib2-18 libpam-modules libpam-modules-bin libpam-runtime libpam-systemd libpam0g
  libpcap0.8 libpython3-stdlib libqmi-glib5 libqmi-proxy libsasl2-2 libsasl2-modules
  libsasl2-modules-db libseccomp2 libsgutils2-2 libss2 libsystemd0 libudev1 libunwind8
  libxtables12 linux-base logsave lvm2 mdadm modemmanager motd-news-config multipath-tools
  open-iscsi packagekit packagekit-tools pci.ids pollinate python-apt-common python3
  python3-apport python3-apt python3-debian python3-distro-info python3-distupgrade
  python3-minimal python3-problem-report python3-software-properties python3-tz
  python3-update-manager sg3-utils sg3-utils-udev snapd software-properties-common sosreport
  systemd systemd-hwe-hwdb systemd-sysv systemd-timesyncd tcpdump ubuntu-advantage-tools
  ubuntu-minimal ubuntu-release-upgrader-core ubuntu-server ubuntu-server-minimal ubuntu-standard
  udev ufw update-manager-core update-notifier-common xfsprogs
113 mis à jour, 3 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 58,1 Mo dans les archives.
Après cette opération, 7 439 ko d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n] _
```

```
adminmonitorack@adminmonitorack:~$ sudo apt upgrade
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Calcul de la mise à jour... Fait
```

Ensuite faire l'installation d'apache :

```
adminmonitorack@adminmonitorack:~$ sudo apt install apache2 apache2-utils
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  apache2-bin apache2-data bzip2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
  liblua5.3-0 mailcap mime-support ssl-cert
Paquets suggérés :
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser bzip2-doc
Les NOUVEAUX paquets suivants seront installés :
  apache2 apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap liblua5.3-0 mailcap mime-support ssl-cert
0 mis à jour, 13 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 2 142 ko dans les archives.
Après cette opération, 8 528 ko d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n] _
```

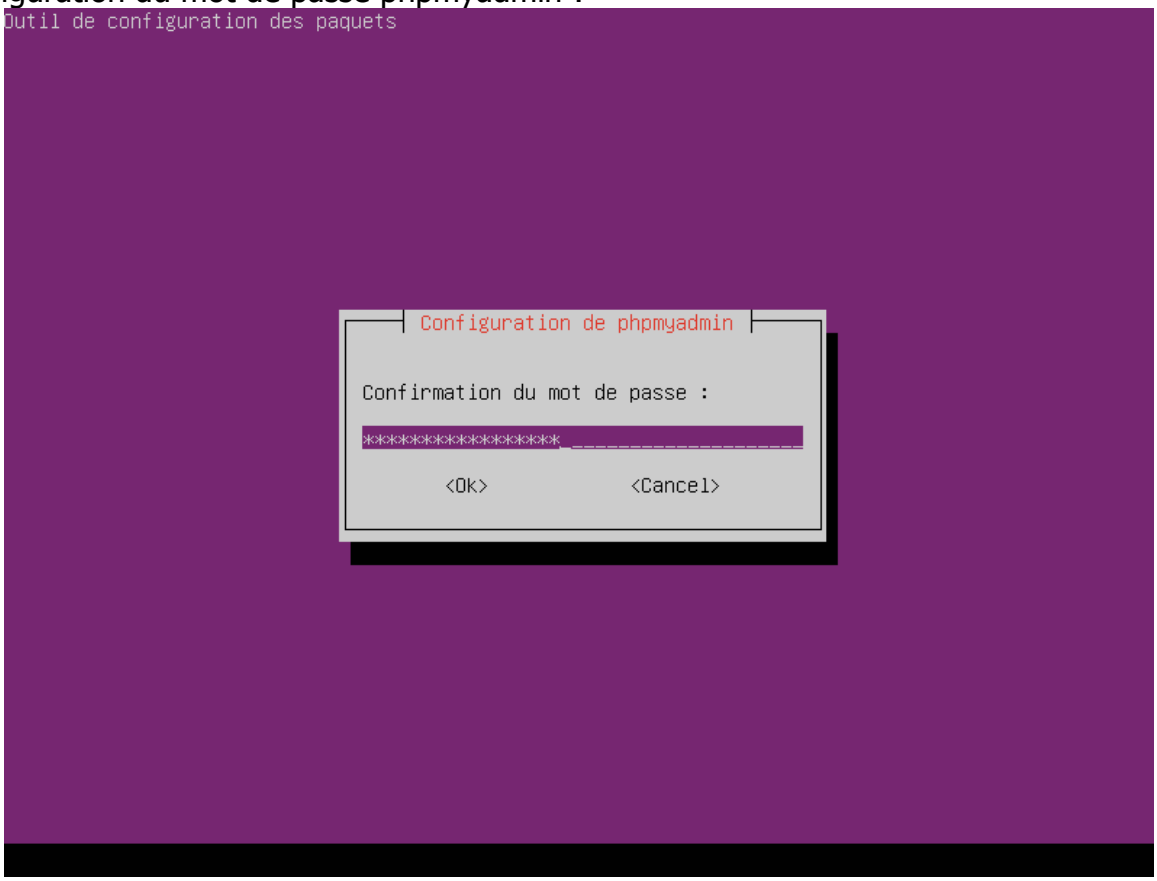
Installation de MariaDB :

```
adminmonitorack@adminmonitorack:~$ sudo apt install mariadb-server
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  galera-4 libcgi-fast-perl libcgi-pm-perl libclone-perl libconfig-inifiles-perl libdbd-mysql-perl
  libdbi-perl libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl libhtml-parser-perl
  libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl
  liblwp-mediatypes-perl libmariadb3 libmysqlclient21 libsnappy1v5 libtimedate-perl liburi-perl
  liburing2 mariadb-client-10.6 mariadb-client-core-10.6 mariadb-common mariadb-server-10.6
  mariadb-server-core-10.6 mysql-common socat
Paquets suggérés :
  libmldbm-perl libnet-daemon-perl libsql-statement-perl libdata-dump-perl libipc-sharedcache-perl
  libbusiness-isbn-perl libwww-perl mailx mariadb-test
Les NOUVEAUX paquets suivants seront installés :
  galera-4 libcgi-fast-perl libcgi-pm-perl libclone-perl libconfig-inifiles-perl libdbd-mysql-perl
  libdbi-perl libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl libhtml-parser-perl
  libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl
  liblwp-mediatypes-perl libmariadb3 libmysqlclient21 libsnappy1v5 libtimedate-perl liburi-perl
  liburing2 mariadb-client-10.6 mariadb-client-core-10.6 mariadb-common mariadb-server
  mariadb-server-10.6 mariadb-server-core-10.6 mysql-common socat
0 mis à jour, 32 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 19,0 Mo dans les archives.
Après cette opération, 168 Mo d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n]
```

Installation de l'ensemble du bundle PHP et phpMyAdmin :

```
adminmonitorack@adminmonitorack:~$ sudo add-apt-repository ppa:ondrej/php -y && sudo apt install php
8.3 php8.3-cli php8.3-common php8.3-ctype php8.3-curl php8.3-mbstring php8.3-xml php8.3-zip php8.3-
okenizer php8.3-mysql phpmyadmin -y
Co-installable PHP versions: PHP 5.6, PHP 7.x, PHP 8.x and most requested extensions are included.
ackages are provided for *Current* Ubuntu *LTS* releases (https://wiki.ubuntu.com/Releases).  Expan
ed Security Maintenance releases ARE NOT supported.

Debian stable, oldstable and Debian LTS packages are provided from a separate repository: https://d
b.sury.org/#debian-dpa

You can get more information about the packages at https://deb.sury.org

BUGS&FEATURES: This PPA has a issue tracker:
https://deb.sury.org/#bug-reporting

Issues reported in a private email don't scale and most likely will be ignored.  I simply don't hav
 capacity to answer questions privately.

CAVEATS:
1. If you are using apache2, you are advised to add ppa:ondrej/apache2
2. If you are using nginx, you are advised to add ppa:ondrej/nginx

DONATION: If you like my work and you want to show appreciation, please consider donating regularly
 https://donate.sury.org/

COMMERCIAL SUPPORT: Support for PHP packages for older Debian and Ubuntu release can be bought from
https://www.freexian.com/lts/php/

WARNING: add-apt-repository is broken with non-UTF-8 locales, see
https://github.com/oerdnj/deb.sury.org/issues/56 for workaround:

# LC_ALL=C.UTF-8 add-apt-repository ppa:ondrej/php
More info: https://launchpad.net/~ondrej/+archive/ubuntu/php
Adding repository.
Found existing deb entry in /etc/apt/sources.list.d/ondrej-ubuntu-php-jammy.list
Adding deb entry to /etc/apt/sources.list.d/ondrej-ubuntu-php-jammy.list
Found existing deb-src entry in /etc/apt/sources.list.d/ondrej-ubuntu-php-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/ondrej-ubuntu-php-jammy.list
```

Configuration du mot de passe phpmyadmin :



S'assurer que la même version de php est utilisée dans tout le système :

```
adminmonitorack@srv-monitorack:~$ sudo update-alternatives --set php /usr/bin/php8.3
adminmonitorack@srv-monitorack:~$
```

Ensuite, j'ai fais l'installation de Symfony :

```
adminmonitorack@srv-monitorack:~$ curl -1sLf 'https://dl.cloudsmith.io/public/symfony/stable/setup.d
eb.sh' | sudo -E bash
Executing the  setup script for the 'symfony/stable' repository ...

   OK: Checking for required executable 'curl' ...
   OK: Checking for required executable 'apt-get' ...
   OK: Detecting your OS distribution and release using system methods ...
 ^^^^: ... Detected/provided for your OS/distribution, version and architecture:
 >>>>:
 >>>>: ... distro=ubuntu  version=22.04  codename=jammy  arch=x86_64
 >>>>:
 NOPE: Checking for apt dependency 'apt-transport-https' ...
   OK: Updating apt repository metadata cache ...
   OK: Attempting to install 'apt-transport-https' ...
   OK: Checking for apt dependency 'ca-certificates' ...
   OK: Checking for apt dependency 'gnupg' ...
   OK: Checking for apt signed-by key support ...
   OK: Importing 'symfony/stable' repository GPG keys ...
   OK: Checking if upstream install config is OK ...
   OK: Installing 'symfony/stable' repository via apt ...
   OK: Updating apt repository metadata cache ...
   OK: The repository has been installed successfully - You're ready to rock!

adminmonitorack@srv-monitorack:~$ _
```

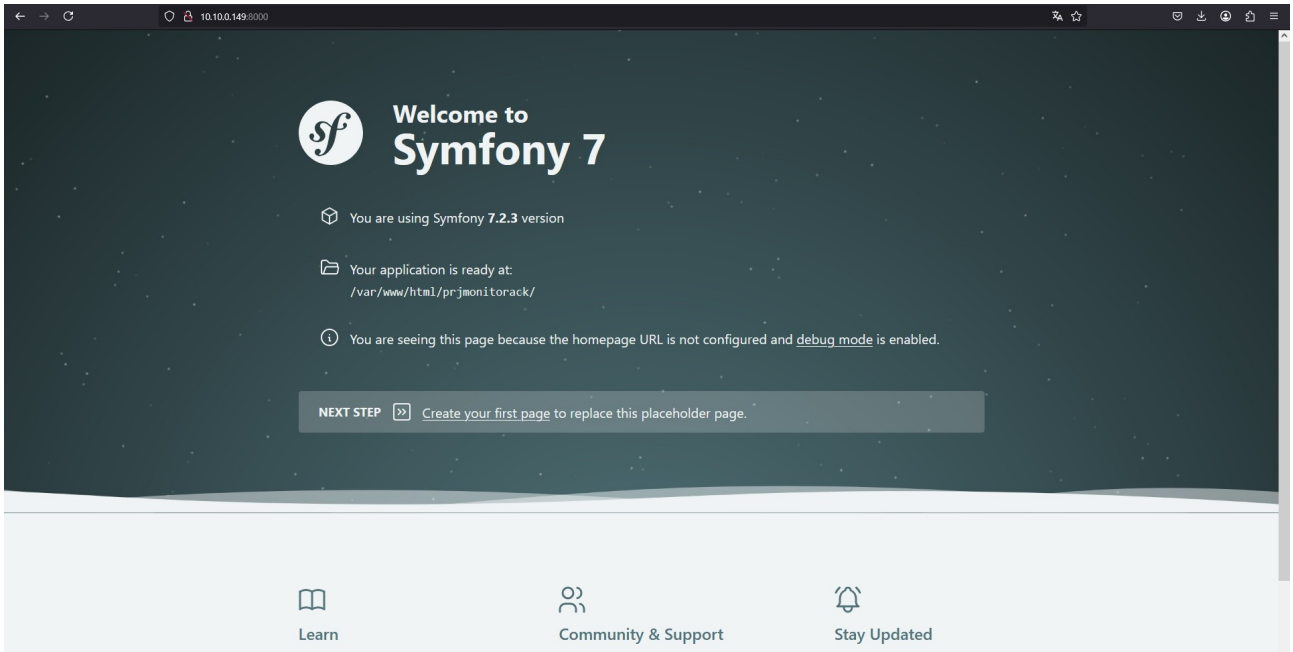
Et l'interpréteur de commande Symfony (le cli) :

```
adminmonitorack@srv-monitorack:~$ sudo apt install symfony-cli
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les NOUVEAUX paquets suivants seront installés :
  symfony-cli
0 mis à jour, 1 nouvellement installés, 0 à enlever et 3 non mis à jour.
Il est nécessaire de prendre 5 819 ko dans les archives.
Après cette opération, 14,8 Mo d'espace disque supplémentaires seront utilisés.
Réception de :1 https://dl.cloudsmith.io/public/symfony/stable/deb/ubuntu jammy/main amd64 symfony-c
li amd64 5.10.7 [5 819 kB]
5 819 ko réceptionnés en 1s (9 665 ko/s)
Sélection du paquet symfony-cli précédemment désélectionné.
(Lecture de la base de données... 83626 fichiers et répertoires déjà installés.)
Préparation du dépaquetage de .../symfony-cli_5.10.7_amd64.deb ...
Dépaquetage de symfony-cli (5.10.7) ...
Paramétrage de symfony-cli (5.10.7) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
adminmonitorack@srv-monitorack:~$
```

Test du fonctionnement de Symfony sur le port 8000 :

```
^Cadminmonitorack@srv-monitorack:/var/www/html/prjmonitorack$ sudo symfony server:start --allow-allp

 [WARNING] run "symfony server:ca:install" first if you want to run the web server with TLS suppor
t, or use "--p12" or "--no-tls" to avoid this warning


Following Web Server log file (/root/.symfony5/log/289c74897d2629ad52e2145ea5b4e864c7c3f35c.log)
Following PHP log file (/root/.symfony5/log/289c74897d2629ad52e2145ea5b4e864c7c3f35c/7daf403c7589f49
27632ed3b6af762a992f09b78.log)

 [WARNING] The local web server is optimized for local development and MUST never be used in a pro
duction setup.


 [OK] Web server listening
      The Web server is using PHP CLI 8.3.16
      http://127.0.0.1:8000

[Web Server ] Feb 10 15:32:44 |DEBUG  | PHP    Reloading PHP versions
[Web Server ] Feb 10 15:32:44 |DEBUG  | PHP    Using PHP version 8.3.16 (from default version in $PA
TH)
[PHP        ] Feb 10 15:32:44 |DEBUG  | RUNNER Waiting for channels (first boot) cmd="PHP"
[Web Server ] Feb 10 15:32:44 |INFO   | PHP    listening path="/usr/bin/php8.3" php="8.3.16" port=36
259
[PHP        ] Feb 10 15:32:46 |DEBUG  | RUNNER Received timer message (first boot) cmd="PHP"
^[S
```

Résultat :



On peut voir qu'on a bien Symfony

Pour installer les fichiers nécessaires de Symfony nous avons installer composer :

```
adminmonitorack@srv-monitorack:~$ sudo apt install composer -y
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
```

Après avoir créer un répertoire pour le projet dans /var/www/html qui est /prjmonitorack. J'ai installer en utilisant compser le module twig et le module maker :

Twig-bundle :

```
adminmonitorack@srv-monitorack:/var/www/html/prjmonitorack$ sudo composer require symfony/twig-bundl
e
[sudo] password for adminmonitorack:
Deprecation Notice: Using ${var} in strings is deprecated, use {$var} instead in /usr/share/php/Symf
ony/Component/Console/Command/DumpCompletionCommand.php:48
Deprecation Notice: Using ${var} in strings is deprecated, use {$var} instead in /usr/share/php/Symf
ony/Component/Console/Command/DumpCompletionCommand.php:56
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Continue as root/super user [yes]? _
```

Maker-bundle :

```
adminmonitorack@srv-monitorack:/var/www/html/prjmonitorack$ sudo composer require symfony/maker-bund
le
[sudo] password for adminmonitorack:
Deprecation Notice: Using ${var} in strings is deprecated, use {$var} instead in /usr/share/php/Symf
ony/Component/Console/Command/DumpCompletionCommand.php:48
Deprecation Notice: Using ${var} in strings is deprecated, use {$var} instead in /usr/share/php/Symf
ony/Component/Console/Command/DumpCompletionCommand.php:56
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Continue as root/super user [yes]?
```

Enfin, j'ai installé Doctrine en utilisant Composer également :

```
adminmonitorack@srv-monitorack:/var/www/html/prjmonitorack$ sudo composer require symfony/orm-pack
[sudo] password for adminmonitorack:
Deprecation Notice: Using ${var} in strings is deprecated, use {$var} instead in /usr/share/php/Symf
ony/Component/Console/Command/DumpCompletionCommand.php:48
Deprecation Notice: Using ${var} in strings is deprecated, use {$var} instead in /usr/share/php/Symf
ony/Component/Console/Command/DumpCompletionCommand.php:56
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Continue as root/super user [yes]? _
```

J'ai également configuré le fichier .env pour qu'il soit adapté à ma base de données. Mais avant cela, j'ai créé un utilisateur MariaDB :

Création de l'utilisateur :

```
adminmonitorack@srv-monitorack:~$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 132
Server version: 10.6.18-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,000 sec)

MariaDB [(none)]> CREATE USER 'adminmonitorack'@'localhost' IDENTIFIED BY 'Administrateur123';
Query OK, 0 rows affected (0,003 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'Adminmonitorack'@'localhost' WITH GRANT OPTION;
```

Configuration du fichier .env :

```
#
# DATABASE_URL="sqlite:///%kernel.project_dir%/var/data.db"
 DATABASE_URL="mysql://Symfony:etudiant@127.0.0.1:3306/monitorack?serverVersion=8.0.32"
# DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=10.11.2-MariaDB&charset=utf
DATABASE_URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=16&charset=utf8"
###< doctrine/doctrine-bundle ###



adminmonitorack@srv-monitorack:/var/www/html/prjmonitorack$
```

FICHE DE TEST FONCTIONNEL UNITAIRE N°1		
Projet **Monitorack**	*Nom du module logiciel testé* **Route d'une baie**	*Date* **20 mars 2025**
Nom du technicien impliqué dans le test : ***Youssif MATTI YOUSIF***		
*Description du module logiciel : Un*e route qui permet de voir une seule baie codé avec Symfony		

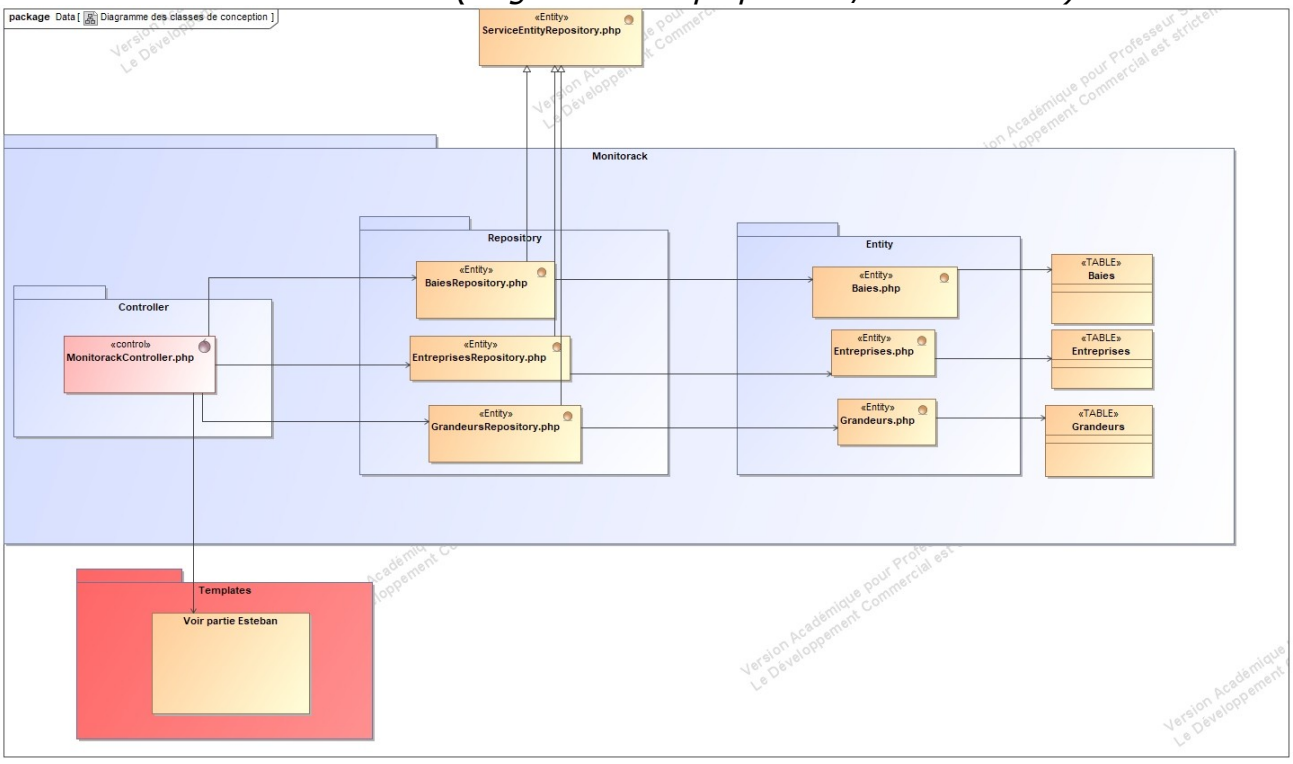
Environnement du test :

Type de test : □ Unitaire Nominal ■ **Unitaire aux limites**

Objectifs du test :

- **Valider la fonctionnement de la route**
- **Valider l'obtention** de **l'ensemble** des **relevés.**

Environnement matériel du test (diagrammes de paquets et / ou de classes) :



Programme de test (MonitorackController.php):

```
    #[Route('/baie/{id}', name: 'baie', methods:['GET'])] //La route de cette page est /baie/{id} et la méthode GET permet de juste récupérer des informations sur cette page

    public function baie(int $id, BaiesRepository $unebaie ): Response //Méthode appelée quand un utilisateur visite /baie/{id}.
    {
        $grandeurs = $unebaie->findOneBy(['id' => $id]); //On utilise findOneBy pour récupérer une seule valeur
        return $this->render('monitorack/uneBaie.html.twig', ['baie' => $grandeurs]); //Affiche la page Twig uneBaie.html.twig.
    }
```

Programme de test (uneBaie.html.twig):

```
{% extends 'base.html.twig' %}

{% block title %}Serveur Serre{% endblock %}

{% block body %}

<div>

<h1 align=center> La baie {{baie.id}} </h1>

<table align=center>
    <tr>
        <th>| ID</th>
        <th>| id de l'entreprises</th>
        <th>| Nom de la baie</th>
        <th>| localisation de la baie</th>
        <th>| ip de la baie |</th>
    </tr>
    <tr>
        <td>| baie.id </td>
        <td>| baie.entreprise_id </td>
        <td>| baie.nom_baie </td>
        <td>| baie.localisation_baie </td>
        <td> | baie.ip_baie |</td>
    </tr>
</table>

</div>
{% endblock %}
```

Résultats du test (validation de la classe) :

ANNEXE : Code de la classe Baies – Fichier Baies.php

```
<?php

namespace App\Entity;
use App\Repository\EntreprisesRepository;
use App\Repository\BaiesRepository;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: BaiesRepository::class)]
class Baies
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

   /* #[ORM\Column]
    private ?int $entreprise_id = null;
*/
    #[ORM\ManyToOne(targetEntity: Entreprises::class, inversedBy: "baies")]
        #[ORM\JoinColumn(name: "entreprise_id", referencedColumnName: "id", nullable:
false)]
    private ?Entreprises $entreprise_id = null;

    #[ORM\Column(length: 100)]
    private ?string $nomBaie = null;

    #[ORM\Column(length: 45)]
    private ?string $localisationBaie = null;
```

```
#[ORM\Column(type: Types::SMALLINT)]

private ?int $ipBaie = null;

public function getId(): ?int
{
    return $this->id;
}

public function getEntrepriseId(): ?Entreprises
{
    return $this->entreprise_id;
}


public function setEntrepriseId(?Entreprises $entreprise_id): static
{
    $this->entreprise_id = $entreprise_id;

    return $this;
}

public function getNomBaie(): ?string
{
    return $this->nomBaie;
}

public function setNomBaie(string $nomBaie): static
{
    $this->nomBaie = $nomBaie;

    return $this;
}

public function getLocalisationBaie(): ?string
{
    return $this->localisationBaie;
}

public function setLocalisationBaie(string $localisationBaie): static
{
    $this->localisationBaie = $localisationBaie;

    return $this;
}
```

```
    public function getIpBaie(): ?int
    {
        return $this→ipBaie;
    }

    public function setIpBaie(int $ipBaie): static
    {
        $this->ipBaie = $ipBaie;

        return $this;
    }
}
```

FICHE DE TEST FONCTIONNEL UNITAIRE N°2		
Projet **Monitorack**	*Nom du module logiciel testé* **Insertion dans la base**	*Date* **27 mai 2025**

Nom du technicien impliqué dans le test : ***Youssif MATTI YOUSIF***

*Description du module logiciel : Un*e route qui permet de tester l'insertion dans la base de données

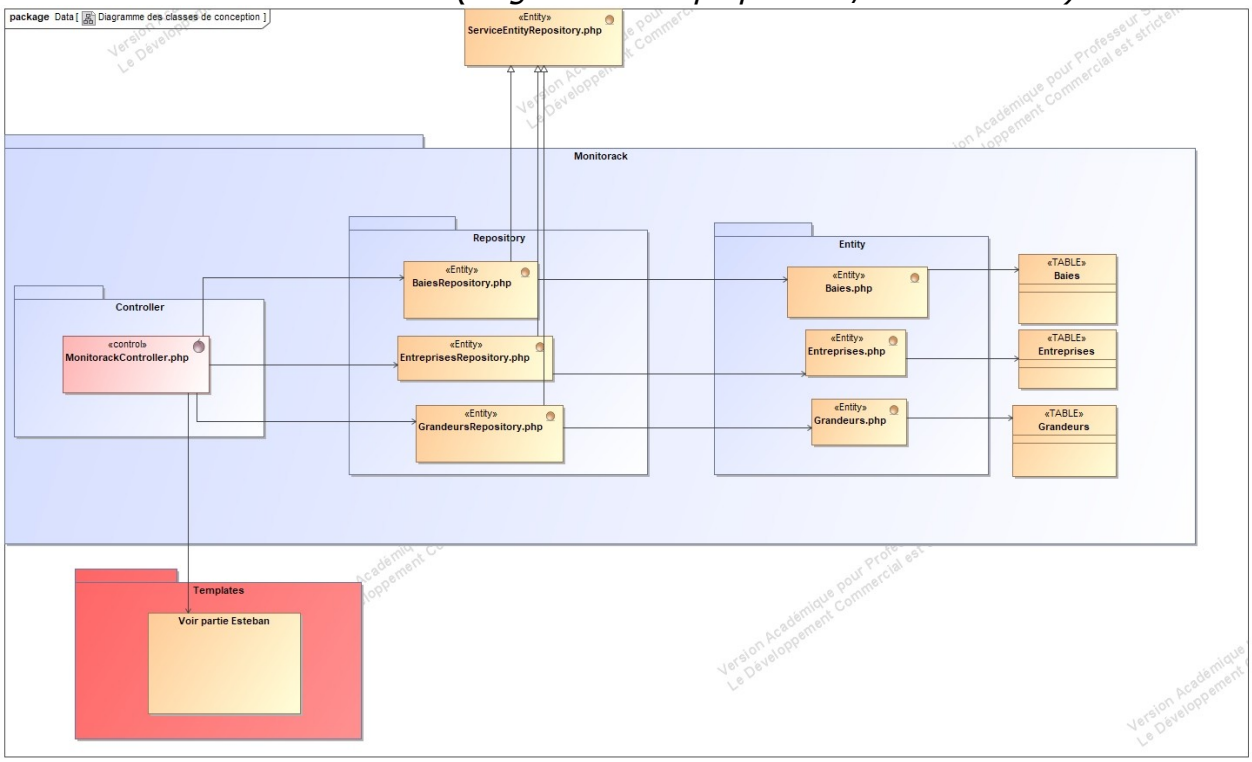
Environnement du test :

Type de test : □ Unitaire Nominal ■ **Unitaire aux limites**

Objectifs du test :

- **Valider la fonctionnement de la route**
- **Valider l'obtention** de **l'ensemble** des **relevés.**

Environnement matériel du test (diagrammes de paquets et / ou de classes) :



Programme de test (MonitorackController.php):

```
#[Route("/api/trame", name: "app_trame", methods: ["POST"])] //La route de cette page est /api/trame et la route n'accepte que les requêtes HTTP POST.

public function saveTrame(Request $request): JsonResponse //La fonction renvoie toujours une réponse JSON.
{
$data = json_decode($request->getContent(), true);//transformer un JSON en tableau
```

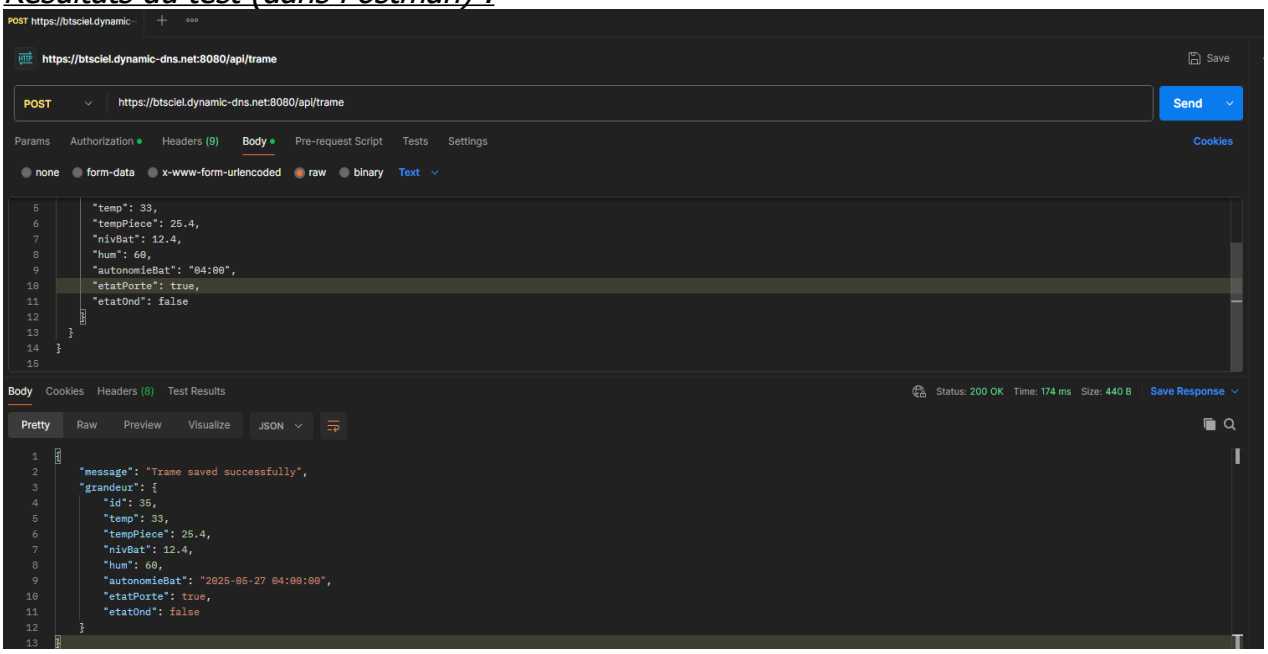
```
associatif PHP.

    //Vérification que les données sont présentes
    if (!$data || empty($data['uplink_message']['decoded_payload'])) {
        return new JsonResponse(['error' => 'Invalid TTN payload'],
JsonResponse::HTTP_BAD_REQUEST);
    }

    $payload = $data['uplink_message']['decoded_payload'];

    // Validation des données
    if (
        empty($payload['baies_id']) ||
        empty($payload['temp']) ||
        empty($payload['tempPiece']) ||
        empty($payload['nivBat']) ||
        empty($payload['hum']) ||
        empty($payload['autonomieBat']) ||
        !isset($payload['etatPorte']) ||
        !isset($payload['etatOnd'])
    ) {
        return new JsonResponse(['error' => 'Missing required fields'],
JsonResponse::HTTP_BAD_REQUEST);
    }

    // Recherche de la baie
    $baie = $this->baiesRepository->find($payload['baies_id']);
    if (!$baie) {
        return new JsonResponse(['error' => 'Baie not found'],
JsonResponse::HTTP_NOT_FOUND);
    }
    // Création de l'objet Grandeurs
    $grandeur = new Grandeurs();
    $grandeur->setBaiesId($baie);
    $grandeur->setTemp($payload['temp']);
    $grandeur->setTempPiece($payload['tempPiece']);
    $grandeur->setNivBat($payload['nivBat']);
    $grandeur->setHum($payload['hum']);
    $grandeur->setAutonomieBat(new \DateTime($payload['autonomieBat']));
    $grandeur->setEtatPorte($payload['etatPorte']);
    $grandeur→setEtatOnd($payload['etatOnd']);

    // Sauvegarde
    $this→entityManager→persist($grandeur); //persist() indique à Doctrine qu'on veut ajouter
cette entité.
```

```
    $this->entityManager->flush();flush() applique les changements en base de données (INSERT).

    return new JsonResponse([
        'message' => 'Trame saved successfully',
        'grandeur' => [
            'id' => $grandeur->getId(),
            'temp' => $grandeur->getTemp(),
            'tempPiece' => $grandeur->getTempPiece(),
            'nivBat' => $grandeur->getNivBat(),
            'hum' => $grandeur->getHum(),
            'autonomieBat' => $grandeur->getAutonomieBat()->format('Y-m-d H:i:s'),
            'etatPorte' => $grandeur->isEtatPorte(),
            'etatOnd' => $grandeur->isEtatOnd(),
        ]
    ], JsonResponse::HTTP_OK);



}
```

Programme de test (Postman):

```
{
  "uplink_message": {
    "decoded_payload": {
      "baies_id": 1,
      "temp": 33,
      "tempPiece": 25.4,
      "nivBat": 12.4,
      "hum": 60,
      "autonomieBat": "04:00:00",
      "etatPorte": true,
      "etatOnd": false
    }
  }
}
```

Résultats du test (dans Postman) :



Résultats du test (dans Phpmyadmin) :

id 1	temp	niv_bat	hum	etat_porte	etat_ond	Baies_id	temp_piece	autonomie_bat
35	33	12.4	60	1	0	1	25.4	04:00:00

Résultats du test (sur le site web)

Contrôleur

MonitorackController.php :

```
<?php

namespace App\Controller;
use App\Repository\BaiesRepository;
use App\Repository\EntreprisesRepository;
use App\Repository\GrandeursRepository;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;
use Doctrine\ORM\Mapping as ORM;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use App\Entity\Grandeurs;
use App\Entity\Entreprises;
use App\Entity\Baies;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Request;



final class MonitorackController extends AbstractController
{

    private EntityManagerInterface $entityManager;
    private BaiesRepository $baiesRepository;
    private GrandeursRepository $grandeursRepository;

    public function __construct(
        EntityManagerInterface $entityManager,
        BaiesRepository $baiesRepository,
        GrandeursRepository $grandeursRepository
    ) {
        $this->entityManager = $entityManager;
        $this->baiesRepository = $baiesRepository;
        $this->grandeursRepository = $grandeursRepository;
    }




   /**********************DashBoard**************************/
/*************************************************************/

    #[Route('/', name: 'Dashboard', methods:['GET'])]
    public function dashboard(GrandeursRepository $grandeursRepo): Response
    {

        $timeWarn = $grandeursRepo->countTempsup();

        $alertes= $grandeursRepo->countAlertes();
```

```
        $fonctionnelles= $grandeursRepo->countFonctionnelles();
        return $this->render('monitorack/dashboard.html.twig', ['Dashboard' =>
$timeWarn, 'alertes' => $alertes, 'func' => $fonctionnelles,]);
    }

/*****************************************************************/



   /********************  entreprise  ************************/
/*****************************************************************/

    #[Route('/entreprises/{id}', name: 'entreprise', methods:['GET'])]
    public function entreprise(int $id, EntreprisesRepository
$une_entreprise ): Response
    {
        $grandeurs = $une_entreprise->findOneBy(['id' => $id]);
        return $this->render('monitorack/uneEntreprise.html.twig',
['entreprise' => $grandeurs]);
    }

/*****************************************************************/


   /***********************   baie   ****************************/
/*****************************************************************/

                        //Toutes les baies//

   #[Route('/baies', name: 'baies', methods:['GET'])]
    public function baies(BaiesRepository $baies): Response
    {

        $baies = $baies->findBaiesAndEntreprises();

        return $this->render('monitorack/baies.html.twig', ['baies' =>
$baies]);
    }




                            //Une baie//

    #[Route('/baie/{id}', name: 'baie', methods:['GET'])]
    public function baie(int $id, GrandeursRepository $grandeursRepo ):
Response
    {
        // $grandeurs = $unebaie->findOneBy(['id' => $id]);
        $grandeurs = $grandeursRepo->findGrandeursByBaieId($id);
        //var_dump($grandeurs);
        return $this->render('monitorack/uneBaie.html.twig', ['baie' =>
$grandeurs]);
    }

/*****************************************************************/
```

```
/**********************payload LoRaWan*****************************/


    #[Route("/api/trame", name: "app_trame", methods: ["POST"])]
public function saveTrame(Request $request): JsonResponse
{
    $data = json_decode($request->getContent(), true);

    if (!$data || empty($data['uplink_message']['decoded_payload'])) {
        return new JsonResponse(['error' => 'Invalid TTN payload'],
JsonResponse::HTTP_BAD_REQUEST);
    }

    $payload = $data['uplink_message']['decoded_payload'];

    // Validation des données
    if (
        empty($payload['baies_id']) ||
        empty($payload['temp']) ||
        empty($payload['tempPiece']) ||
        empty($payload['nivBat']) ||
        empty($payload['hum']) ||
        empty($payload['autonomieBat']) ||
        !isset($payload['etatPorte']) ||
        !isset($payload['etatOnd'])
    ) {
        return new JsonResponse(['error' => 'Missing required fields'],
JsonResponse::HTTP_BAD_REQUEST);
    }

    // Recherche de la baie
    $baie = $this->baiesRepository->find($payload['baies_id']);
    if (!$baie) {
        return new JsonResponse(['error' => 'Baie not found'],
JsonResponse::HTTP_NOT_FOUND);
    }

    // Création de l'objet Grandeurs
    $grandeur = new Grandeurs();
    $grandeur->setBaiesId($baie);
    $grandeur->setTemp($payload['temp']);
    $grandeur->setTempPiece($payload['tempPiece']);
    $grandeur->setNivBat($payload['nivBat']);
    $grandeur->setHum($payload['hum']);
    $grandeur->setAutonomieBat(new \DateTime($payload['autonomieBat']));
    $grandeur->setEtatPorte($payload['etatPorte']);
    $grandeur->setEtatOnd($payload['etatOnd']);

    // Sauvegarde
    $this->entityManager->persist($grandeur);
    $this->entityManager->flush();

    return new JsonResponse([
        'message' => 'Trame saved successfully',
        'grandeur' => [
            'id' => $grandeur->getId(),
```

```
            'temp' => $grandeur->getTemp(),
            'tempPiece' => $grandeur->getTempPiece(),
            'nivBat' => $grandeur->getNivBat(),
            'hum' => $grandeur->getHum(),
            'autonomieBat' => $grandeur->getAutonomieBat()->format('Y-m-d
H:i:s'),
            'etatPorte' => $grandeur->isEtatPorte(),
            'etatOnd' => $grandeur->isEtatOnd(),
        ]
    ], JsonResponse::HTTP_OK);
}

/*****************************************************************/


/**********************aide*****************************/
/*****************************************************************/

   #[Route('/aides', name: 'aide', methods:['GET'])]
    public function aide(): Response
    {
        return $this->render('monitorack/aide.html.twig', ['aide' =>
'MonitorackController']);
    }

/*****************************************************************/
}
```

Entité

Baies.php :

```
<?php

namespace App\Entity;
use App\Repository\EntreprisesRepository;
use App\Repository\BaiesRepository;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;
use Doctrine\Common\Collections\Collection;
use Doctrine\Common\Collections\ArrayCollection;

#[ORM\Entity(repositoryClass: BaiesRepository::class)]
class Baies
{
   #[ORM\Id]
   #[ORM\GeneratedValue]
   #[ORM\Column]
```

```
    private ?int $id = null;


    #[ORM\ManyToOne(targetEntity: Entreprises::class, inversedBy: "baies")]
    #[ORM\JoinColumn(name: "entreprise_id", referencedColumnName: "entreprise_id",
nullable: false)]
    private ?Entreprises $entreprise = null;

    #[ORM\Column(length: 100)]
    private ?string $nomBaie = null;

    #[ORM\Column(length: 45)]
    private ?string $localisationBaie = null;

    #[ORM\Column(length: 15)]
    private ?string $ipBaie = null;

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getEntrepriseId(): ?Entreprises
    {
        return $this->entreprise ;
    }


    public function setEntrepriseId(?Entreprises $entreprise): static
    {
        $this->entreprise  = $entreprise ;

        return $this;
    }

    public function getNomBaie(): ?string
    {
        return $this->nomBaie;
    }

    public function setNomBaie(string $nomBaie): static
    {
        $this->nomBaie = $nomBaie;
```

```
        return $this;
    }

    public function getLocalisationBaie(): ?string
    {
        return $this->localisationBaie;
    }

    public function setLocalisationBaie(string $localisationBaie): static
    {
        $this->localisationBaie = $localisationBaie;

        return $this;
    }

    public function getIpBaie(): ?string
    {
        return $this->ipBaie;
    }

    public function setIpBaie(int $ipBaie): static
    {
        $this->ipBaie = $ipBaie;

        return $this;
    }


#[ORM\OneToMany(mappedBy: "baies", targetEntity: Grandeurs::class)]
    private Collection $grandeurs;

    public function __construct()
    {
        $this->grandeurs = new ArrayCollection();
    }



   public function getGrandeurs(): Collection
    {
        return $this->grandeurs;
    }
```

```
}
```

Entreprises.php :

```
<?php

namespace App\Entity;
use App\Repository\BaiesRepository;
use App\Repository\EntreprisesRepository;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: EntreprisesRepository::class)]
class Entreprises
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $entreprise_id = null;

    #[ORM\Column(length: 45)]
    private ?string $nomEntreprise = null;

    #[ORM\Column(length: 45)]
    private ?string $adresse = null;

    #[ORM\Column(length: 45)]
    private ?string $coordonnees = null;

    public function getId(): ?int
    {
        return $this->entreprise_id;
    }

    public function getNomEntreprise(): ?string
    {
        return $this->nomEntreprise;
    }

    public function setNomEntreprise(string $nomEntreprise): static
    {
        $this->nomEntreprise = $nomEntreprise;

        return $this;
    }
```

```
    public function getAdresse(): ?string
    {
        return $this->adresse;
    }

    public function setAdresse(string $adresse): static
    {
        $this->adresse = $adresse;

        return $this;
    }

    public function getCoordonnees(): ?string
    {
        return $this->coordonnees;
    }

    public function setCoordonnees(string $coordonnees): static
    {
        $this->coordonnees = $coordonnees;

        return $this;
    }


    #[ORM\OneToMany(mappedBy: "entreprise", targetEntity: Baies::class)]
    private Collection $baies;

    public function __construct()
    {
        $this->baies = new ArrayCollection();
    }



   public function getBaies(): Collection
    {
        return $this->baies;
    }

}
```

Grandeurs.php :

```
<?php

namespace App\Entity;

use App\Repository\GrandeursRepository;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: GrandeursRepository::class)]
class Grandeurs
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    //#[ORM\Column]
    #[ORM\ManyToOne(targetEntity: Baies::class)]
    #[ORM\JoinColumn(name: "Baies_id", referencedColumnName: "id", nullable: false)]
    private ?Baies $baies = null;

    #[ORM\Column]
    private ?float $temp = null;

    #[ORM\Column]
    private ?float $tempPiece = null;



    #[ORM\Column]
    private ?float $nivBat = null;

    #[ORM\Column(type: Types::SMALLINT)]
    private ?int $hum = null;

    #[ORM\Column]
    private ?bool $etatPorte = null;

    #[ORM\Column]
    private ?bool $etatOnd = null;

    #[ORM\Column(type: Types::TIME_MUTABLE, nullable: true)]
```

```
    private ?\DateTimeInterface $autonomieBat = null;

    /*#[ORM\Column(type: Types::DATETIME_MUTABLE)]
    private ?\DateTimeInterface $autonomieBat = null;
*/



    public function getId(): ?int
    {
        return $this->id;
    }

    public function getBaiesId(): ?Baies
    {
        return $this->baies;
    }

    public function setBaiesId(?Baies $baies): static
    {
        $this->baies = $baies;

        return $this;
    }

    public function getTemp(): ?float
    {
        return $this->temp;
    }

    public function setTemp(float $temp): static
    {
        $this->temp = $temp;

        return $this;
    }


    public function getTempPiece(): ?float
    {
        return $this->tempPiece;
    }

    public function setTempPiece(float $tempPiece): static
```

```
    {
        $this->tempPiece = $tempPiece;
        return $this;
    }


    public function getNivBat(): ?float
    {
        return $this->nivBat;
    }

    public function setNivBat(float $nivBat): static
    {
        $this->nivBat = $nivBat;

        return $this;
    }

    public function getHum(): ?int
    {
        return $this->hum;
    }

    public function setHum(int $hum): static
    {
        $this->hum = $hum;

        return $this;
    }

    public function isEtatPorte(): ?bool
    {
        return $this->etatPorte;
    }

    public function setEtatPorte(bool $etatPorte): static
    {
        $this->etatPorte = $etatPorte;

        return $this;
    }

    public function isEtatOnd(): ?bool
    {
```

```
        return $this->etatOnd;
    }

    public function setEtatOnd(bool $etatOnd): static
    {
        $this->etatOnd = $etatOnd;

        return $this;
    }

    public function getAutonomieBat(): ?\DateTimeInterface
    {
        return $this->autonomieBat;
    }

    public function setAutonomieBat(?\DateTimeInterface $autonomieBat): static
    {
        $this->autonomieBat = $autonomieBat;

        return $this;
    }
}
```

Repository

BaiesRepository.php

```
<?php

namespace App\Repository;

use App\Entity\Baies;
use App\Entity\Entreprises;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @extends ServiceEntityRepository<Baies>
 */
class BaiesRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
```

```
        parent::__construct($registry, Baies::class);
    }



/
**********************************ORM*********************************/
/**********Afficher le nom de l'entreprise pour chaque baie*************/
    public function findBaiesAndEntreprises(): array
{
    return $this->createQueryBuilder('b')
        ->select('b.id', 'b.nomBaie', 'b.localisationBaie', 'b.ipBaie', 'e.nomEntreprise')
        ->leftJoin('b.entreprise', 'e')
        ->getQuery()
        ->getResult();
}




/
**********************************DQL*********************************/
/**********Afficher le nom de l'entreprise pour chaque baie*************/
    /* public function findBaiesAndEntreprises(): array
    {
    $entityManager = $this->getEntityManager();

    $query = $entityManager->createQuery(
        'SELECT baies.id, baies.nomBaie, baies.localisationBaie, baies.ipBaie,
entreprises.nomEntreprise
         FROM App\Entity\Baies baies
         LEFT JOIN baies.entreprise entreprises
         '
    );

    return $query->getResult();
    }*/



}
```

EntreprisesRepository.php

```php
<?php

namespace App\Repository;

use App\Entity\Entreprises;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @extends ServiceEntityRepository<Entreprises>
 */
class EntreprisesRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Entreprises::class);
    }

    //    /**
    //     * @return Entreprises[] Returns an array of Entreprises objects
    //     */
    //    public function findByExampleField($value): array
    //    {
    //        return $this->createQueryBuilder('e')
    //            ->andWhere('e.exampleField = :val')
    //            ->setParameter('val', $value)
    //            ->orderBy('e.id', 'ASC')
    //            ->setMaxResults(10)
    //            ->getQuery()
    //            ->getResult()
    //        ;
    //    }

    //    public function findOneBySomeField($value): ?Entreprises
    //    {
    //        return $this->createQueryBuilder('e')
    //            ->andWhere('e.exampleField = :val')
    //            ->setParameter('val', $value)
    //            ->getQuery()
    //            ->getOneOrNullResult()
    //        ;
    //    }
```

```
}
```

GrandeursRepository.php

```
<?php

namespace App\Repository;

use App\Entity\Grandeurs;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @extends ServiceEntityRepository<Grandeurs>
 */
class GrandeursRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Grandeurs::class);
    }




public function findLastByBaie(string $baie)
{
    return $this->createQueryBuilder('g')
        ->where('g.baie = :baie')
        ->setParameter('baie', $baie)
        ->orderBy('g.id', 'DESC') // ou 'g.date', si vous avez un champ date
        ->setMaxResults(1)
        ->getQuery()
        ->getOneOrNullResult();
}

/
**********Temperature>50°**********
**********/

public function countTempsup(): int
{
return $this->createQueryBuilder('g')
```

```
->select('COUNT(g.id)')
->where('g.temp > 40 OR g.etatOnd = 0 OR g.hum < 35 OR g.hum > 70 OR g.nivBat < 20 OR g.tempPiece > 40')
->getQuery()
->getSingleScalarResult();
}

public function countAlertes(): int
{
return $this->createQueryBuilder('g')
->select('COUNT(g.id)')
->where('(g.temp BETWEEN 33 AND 39) OR g.etatPorte = 1')
->getQuery()
->getSingleScalarResult();
}


public function countFonctionnelles(): int
{
return $this->createQueryBuilder('g')
->select('COUNT(g.id)')
->where('g.temp <= 32 AND g.etatPorte = 0 AND g.etatOnd = 1 AND (g.hum BETWEEN 36 AND 65) AND g.tempPiece <= 32')
->getQuery()
->getSingleScalarResult();
}
/
**********************************************************************
**********/


/
*****************************************ORM*************************
*****************/
/*******************************Affichage d'une baie*************************************/

/***public function findGrandeursByBaieId(int $id): array
{
   $qb = $this->createQueryBuilder('g')
       ->select('b.id', 'b.nomBaie', 'b.localisationBaie', 'b.ipBaie', 'g.id', 'g.temp', 'g.nivBat', 'g.hum', 'g.autonomieBat', 'g.etatPorte', 'g.etatOnd')
       ->leftJoin('g.baies', 'b')
       ->where('b.id = :id')
```

```
        ->setParameter('id', $id);

    return $qb->getQuery()->getResult();
     ****/

public function findGrandeursByBaieId(int $id): ?array
    {
        $qb = $this->createQueryBuilder('g')
            ->select('b.id AS baie_id', 'b.nomBaie', 'b.localisationBaie', 'b.ipBaie',
                    'g.id AS grandeur_id', 'g.temp', 'g.tempPiece', 'g.nivBat', 'g.hum',
                    'g.autonomieBat', 'g.etatPorte', 'g.etatOnd')
            ->leftJoin('g.baies', 'b')
            ->where('b.id = :id')
            ->setParameter('id', $id)
            ->orderBy('g.id', 'DESC')
            ->setMaxResults(1);

        return $qb->getQuery()->getOneOrNullResult();
    }









/
**********************************************DQL*****************************
****************/
/*******************************Affichage d'une
baie*************************************/
/*public function findGrandeursByBaieId(int $id): array
{
    $entityManager = $this->getEntityManager();

    $query = $entityManager->createQuery(
        'SELECT baies.id, baies.nomBaie, baies.localisationBaie, baies.ipBaie, grandeurs.id,
grandeurs.temp, grandeurs.nivBat, grandeurs.hum, grandeurs.autonomieBat,
grandeurs.etatPorte, grandeurs.etatOnd
         FROM App\Entity\Grandeurs grandeurs
         LEFT JOIN grandeurs.baies baies
         WHERE baies.id = :id'
    )->setParameter('id', $id);
```

```
    return $query->getResult();
}*/

/
**********************************************************************
*************/

}
```

12.4. Annexes Esteban DUFAU

Base.html.twig :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>{% block title %}Dashboard{% endblock %}</title>
  <!-- Ajouter ici les fichiers CSS d'AdminLTE -->
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/admin-lte@3.2.0/dist/css/adminlte.min.css">
  <!-- Optionnel : ajouter d'autres fichiers CSS si besoin -->
  {% block stylesheets %}{% endblock %}
</head>
<body class="hold-transition  layout-fixed no-padding no-sidebar no-margin">
  <div class="wrapper">
    <!-- Ajouter ici le menu sur la gauche de l'écran -->
    <aside class="main-sidebar sidebar-dark-primary elevation-4">
      <a href="#" class="brand-link">
        <span class="brand-text font-weight-light">Menu</span>
      </a>
      <div class="sidebar">
        <nav class="mt-2">
          <ul class="nav nav-pills nav-sidebar flex-column" data-widget="treeview"
role="menu" data-accordion="false">
            <li class="nav-item">
              <a href="/" class="nav-link">
                <i class="nav-icon fas fa-tachometer-alt"></i>
                <p>Dashboard</p>
              </a>
            </li>
            <li class="nav-item">
              <a href="/baies" class="nav-link">
                <i class="nav-icon fas fa-server"></i>
                <p>Baies</p>
              </a>
            </li>
            <!-- Ajouter d'autres éléments de menu ici -->
          </ul>
        </nav>
      </div>
    </aside>
    <div class="content-wrapper no-padding no-sidebar"><!-- Removed white space
here -->
      {% block body %}{% endblock %}
    </div>
  </div>
  <!-- Ajouter ici les fichiers JS nécessaires à AdminLTE -->
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

```
  <script
src="https://cdn.jsdelivr.net/npm/admin-lte@3.2.0/dist/js/adminlte.min.js"></
script>
  <!-- Bootstrap JS -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></
script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></
script>
  {% block javascripts %}{% endblock %}
</body>
</html>
```

dashboard.html.twig :

```
{% extends 'base.html.twig' %}

{% block title %}Dashboard - Monitorack{% endblock %}

{% block body %}
<meta http-equiv="refresh" content="10">
  <!-- En-tête du tableau de bord -->
  <section class="content-header">
    <div class="container-fluid">
      <div class="row mb-2">
        <div class="col-sm-6">
          <h1>MONITORACK</h1>
          <p>Projet d'élèves BTS CIEL</p>
        </div>
      </div>
    </div>
  </section>

  <!-- Contenu principal -->
  <section class="content">
    <div class="container-fluid">
      <div class="row">
        <!-- Carte Erreurs -->
        <div class="col-lg-4 col-12">
          <div class="small-box bg-danger">
            <div class="inner">
              <h3>{{Dashboard}}</h3> <!-- Nombre dynamique d'erreurs -->
```

```
              <p>Erreurs</p>
            </div>
            <div class="icon">
              <i class="fas fa-exclamation-triangle"></i>
            </div>
          </div>
        </div>

        <!-- Carte Alertes -->
        <div class="col-lg-4 col-12">
          <div class="small-box bg-warning">
            <div class="inner">
              <h3>{{alertes}}</h3> <!-- Nombre dynamique d'alertes -->
              <p>Alertes</p>
            </div>
            <div class="icon">
              <i class="fas fa-bell"></i>
            </div>
          </div>
        </div>

        <!-- Carte Fonctionnelles -->
        <div class="col-lg-4 col-12">
          <div class="small-box bg-success">
            <div class="inner">
              <h3>{{func}}</h3> <!-- Nombre dynamique de fonctionnalités -->
              <p>Fonctionnelles</p>
            </div>
            <div class="icon">
              <i class="fas fa-check-circle"></i>
            </div>
          </div>
        </div>
      </div>

      <!-- Section Aide -->
      <div class="row">
        <div class="col-md-6">
          <div class="card card-outline card-success">
            <div class="card-header">
              <h3 class="card-title"><i class="fas fa-info-circle"></i> Aide</h3>
            </div>
            <div class="card-body">
              <p>
                Si vous rencontrez des problèmes, la page Aide vous guidera et
vous expliquera comment procéder.
                N'hésitez pas à la consulter !
              </p>
              <a href="aide" class="btn btn-success"><i class="fas fa-arrow-
right"></i> Aller</a>
            </div>
          </div>
```

```
        </div>

        <!-- Graphique -->
        <div class="col-md-6">
          <div class="card card-outline card-primary">
            <div class="card-header">
              <h3 class="card-title"><i class="fas fa-chart-line"></i> Erreurs des
10 derniers jours</h3>
            </div>
            <div class="card-body">
              <canvas id="errorChart"></canvas>
            </div>
          </div>
        </div>
      </div>
    </div>
  </section>
{% endblock %}
```

unebaie.html.twig

```
{% extends 'base.html.twig' %}

{% block title %}Monitorack-{{baie.nomBaie}}{% endblock %}

{% block body %}
<html lang="fr">
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Monitorack-grille baies</title>
  <!-- Bootstrap CSS -->
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <!-- AdminLTE CSS -->
  <link rel="stylesheet"
href="https://adminlte.io/themes/v3/dist/css/adminlte.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
  <style>
    .header-green {
      background-color: #d4edda;
      padding: 15px;
      border-radius: 5px;
      margin-bottom: 20px;
    }
    .data-card {
```

```
      background-color: #f8f9fa;
      border: 1px solid #dee2e6;
      border-radius: 8px;
      padding: 20px;
      margin-bottom: 20px;
      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    }
    .data-card i {
      font-size: 24px;
      color: #28a745;
      margin-right: 10px;
    }
    .data-card h5 {
      margin: 0;
      font-size: 18px;
      color: #333;
    }
    .data-card p {
      margin: 0;
      font-size: 14px;
      color: #666;
    }
  </style>
</head>

<body class="hold-transition sidebar-mini">

<div class="wrapper">
  <!-- Content Wrapper. Contains page content -->

    <!-- Main content -->
    <section class="content">
      <div class="container-fluid">
        <div class="header-green text-center">
          <h1>{{ baie.nomBaie }}</h1>
        </div>
        <div class="row">
          <div class="col-md-4">
            <div class="data-card">
              <i class="fas fa-id-badge"></i>
              <h5>ID</h5>
              <p>{{ baie.baie_id }} <!-- Valeur fictive pour ID, numéro 18 --></p>
            </div>
          </div>
          <div class="col-md-4">
            <div class="data-card">
              <i class="fas fa-building"></i>
              <h5>Id de l'entreprise</h5>
               <p></p>
            </div>
          </div>
          <div class="col-md-4">
```

```
            <div class="data-card">
              <i class="fas fa-server"></i>
              <h5>Nom de la baie</h5>
              <p>{{ baie.nomBaie }} <!-- Valeur fictive pour Nom de la baie,
numéro 18 --></p>
            </div>
          </div>
          <div class="col-md-4">
            <div class="data-card">
              <i class="fas fa-map-marker-alt"></i>
              <h5>Localisation de la baie</h5>
              <p>{{ baie.localisationBaie }} <!-- Valeur fictive pour Localisation
de la baie, numéro 18 --></p>
            </div>
          </div>
          <div class="col-md-4">
            <div class="data-card">
              <i class="fas fa-network-wired"></i>
              <h5>IP de la baie</h5>
              <p>{{ baie.ipBaie }} <!-- Valeur fictive pour IP de la baie, numéro
18 --></p>
            </div>
          </div>
          <div class="col-md-4">
            <div class="data-card">
              <i class="fas fa-thermometer-half"></i>
              <h5>Température de la batterie</h5>
              <p>{{ baie.temp }}°C <!-- Valeur fictive pour Température, numéro 18
--></p>
            </div>
          </div>
          <div class="col-md-4">
            <div class="data-card">
              <i class="fas fa-battery-half" style="color: {{ baie.nivBat < 20 ?
'red' : '#28a745' }};"></i>
              <h5>Niveau de batterie</h5>
              <p>{{ baie.nivBat }}% <!-- Valeur fictive pour Niveau de batterie,
numéro 18 --></p>
            </div>
          </div>

          <div class="col-md-4">
            <div class="data-card">
              <i class="fas fa-tint"></i>
              <h5>Humidité</h5>
              <p>{{ baie.hum }}% <!-- Valeur fictive pour Humidité, numéro 18 --
></p>
            </div>
          </div>
          <div class="col-md-4">
            <div class="data-card">
              <i class="fas fa-clock"></i>
```

```
              <h5>Autonomie de la batterie</h5>
              <p>{{ baie.autonomieBat|date('H:i') }} <!-- Valeur fictive pour
Autonomie de la batterie, numéro 18 --></p>
            </div>
          </div>
          <div class="col-md-4">
            <div class="data-card">
              <i class="fas fa-thermometer"></i>
              <h5>Température de la baie</h5>
              <p>
                {% if baie.tempPiece is defined and baie.tempPiece is not null %}
                  {{ baie.tempPiece }}°C
                {% else %}
                  N/A
                {% endif %}
              </p>
            </div>
          </div>
          <div class="col-md-4">
            <div class="data-card">
              <i class="fas fa-door-open"></i>
              <h5>État de la porte</h5>
                {% if baie.etatPorte == 1 %}
                  Ouvert
                {% else %}
                  Fermé
                {% endif %} </p>
            </div>
          </div>
        </div>
      </div><!-- /.container-fluid -->
    </section>
    <!-- /.content -->

  <!-- /.content-wrapper -->
</div>
<!-- ./wrapper -->

<!-- Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
<!-- AdminLTE App -->
<script src="https://adminlte.io/themes/v3/dist/js/adminlte.min.js"></script>

</body>

</html>
{% endblock %}
```

baies.html.twig :

```
{# templates/baie/index.html.twig #}
{% extends 'base.html.twig' %}

{% block body %}
<html lang="fr">
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tableau des Baies</title>
  <!-- Bootstrap CSS -->
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <!-- AdminLTE CSS -->
  <link rel="stylesheet"
href="https://adminlte.io/themes/v3/dist/css/adminlte.min.css">
  <style>
    .header-green {
      background-color: #d4edda;
    }
    .table-striped-green tbody tr:nth-of-type(odd) {
      background-color: #e9f7ef;
    }
    .table-striped-green tbody tr:nth-of-type(even) {
      background-color: #d4edda;
    }
    .no-margin-left {
      margin-left: 0 !important;
    }
    .no-padding {
      padding: 0 !important;
    }
    .no-horizontal-scroll {
      overflow-x: hidden;
    }
    #contextMenu a {
      display: block;
      padding: 8px 20px;
      text-decoration: none;
      color: black;
    }
    .table-striped-green tbody tr:hover {
    background-color: #c3e6cb;
    }
    .table-striped-green tbody tr,
    .table-striped-green thead th,
    h1 {
      user-select: none;
    }
```

```
#contextMenu a:hover {
  background-color: #f0f0f0;
}

  </style>
</head>

<body class="hold-transition sidebar-mini no-horizontal-scroll">

<div class="wrapper">
  <!-- Content Wrapper. Contains page content -->
  <div class="content-wrapper no-padding no-margin-left">
    <!-- Main content -->
    <section class="content no-padding no-margin">
      <div class="container-fluid no-padding no-margin">
        <div class="d-flex justify-content-between align-items-center my-4 header-
green p-3">
          <h1>Tableau des Baies</h1>
          <button class="btn btn-primary">Exporter</button>
          <div id="contextMenu" style="display:none; position:absolute;
background-color:white; border:1px solid #ccc; box-shadow: 0px 0px 5px
rgba(0,0,0,0.2);">
 <ul style="list-style:none; padding: 5px 0; margin: 0;">
  <li><a href="#" id="exportCSV">Exporter en CSV</a></li>
  <li><a href="#" id="exportExcel">Exporter en Excel</a></li>
  <li><a href="#" id="exportPDF">Exporter en PDF</a></li>
 </ul>
</div>
        </div>
        <div class="card no-margin-left">
          <div class="card-header">
            <h3 class="card-title">Liste des Baies</h3>
          </div>
          <!-- /.card-header -->
          <div class="card-body table-responsive p-0">
            <table class="table table-bordered table-striped table-striped-green
no-margin">
              <thead>
                <tr>
                  <th>Nom de la Baie</th>
                  <th>Localisation</th>
                  <th>IP</th>
                  <th>Entreprise</th>
                </tr>
              </thead>
              <tbody>
                {% for baie in baies %}
                <tr data-id="{{ baie.id }}" class="clickable-row">
                  <td>{{ baie.nomBaie }}</td>
                  <td>{{ baie.localisationBaie }}</td>
                  <td>{{ baie.ipBaie }}</td>
                  <td>{{ baie.nomEntreprise }}</td>
```

```
                </tr>
                {% endfor %}
              </tbody>
            </table>
          </div>
          <!-- /.card-body -->
{# Script to make table rows clickable and redirect to the corresponding baie page #}
<script>
  document.addEventListener('DOMContentLoaded', function() {
    const rows = document.querySelectorAll('.clickable-row');
    rows.forEach(row => {
      row.addEventListener('click', function() {
        const id = this.getAttribute('data-id');
        window.location.href = `/baie/${id}`;
      });
    });
  });
</script>
        </div>
        <!-- /.card -->
      </div><!-- /.container-fluid -->
    </section>
    <!-- /.content -->
  </div>
  <!-- /.content-wrapper -->
</div>
<!-- ./wrapper -->

<!-- Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
<!-- AdminLTE App -->
<script src="https://adminlte.io/themes/v3/dist/js/adminlte.min.js"></script>

</body>

</html>
{% endblock %}
```