POLYTECH NICE SOPHIA

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

UNIVERSITÉ CÔTE D'AZUR

Rapport d'Architecture Logicielle

ElderSpy

Carla WAGSCHAL
Benjamin NEUVILLE-BURGUIERE
Ivan VAN DER TUIJN
Dorian BOUCHARD
Julien DIDIER

Sommaire

1. Contextualisation

1.1. Préambule

Pour aider les personnes âgées à rester à la maison, ElderSpy propose une solution d'e-santé qui permet de partager des informations avec leur entourage et le personnel médical.

1.2. Contexte Global

Le projet ElderSpy est pensé pour offrir une solution complète de surveillance de métriques sur la qualité de vie et la santé des patients. Ce système comprend une structure dans laquelle un ensemble d'appareils et de services est mis en place pour l'installation et la maintenance d'ElderSpy chez la personne et le suivi des patients. La solution comprend différentes composantes. Une première partie, qui s'applique au patient, lui permet d'informer les autres acteurs (médecin, infirmier, proches) directement (formulaire) ou indirectement (capteurs). Une seconde composante, la partie visualisation des données, s'articule en différentes interfaces dédiées, qui sont orientées selon le type d'utilisateur. Enfin, pour assurer une maintenance en continue, une composante permettra une administration de tous les systèmes à distance mais aussi monitorer en direct depuis la maison de la personne, afin d'ajouter des capteurs par exemple.

1.3. Hypothèses de travail

Dans cette partie, il sera question de lister les différentes hypothèses de travail qui ont été mises en place lors de l'étude et la conception de ce projet.

Hypothèse 1 :

Les dispositifs sont uniformisés entre chaque domicile et se trouvent dans une liste préétablie. Cela permet de définir les moyens de communication et de connexion de l'ensemble des appareils connectés. De plus, cela assure une maintenance simplifiée.

Hypothèse 2 :

L'installation de nouveaux dispositifs se fait par des techniciens agréés.

Hypothèse 3 :

La maintenance d'alimentation se fait par différents acteurs (comme un proche, un infirmier) autre que le patient et le médecin.

Hypothèse 4 :

Les appareils fournissent des données précises et ne font pas d'erreurs.

Hypothèse 5 :

Un partenariat avec Pro Santé Connect pour nous permettre d'utiliser leur système d'authentification pour les médecins et les infirmiers (assure qu'il s'agit de vraies personnes de la santé).

Hypothèse 6 :

Les patients sont familiers avec les technologies leur permettant l'utilisation d'un assistant vocal et de formulaire.

1.4. Contraintes

Dans cette partie, il s'agit de contraintes imposées par des hypothèses.

Contrainte 1 :

La solution n'est utilisable que pour les médecins et les infirmiers provenant de France seulement (causé par l'Hypothèse 5).

2. Analyse des besoins

2.1. Users Stories

Cette partie regroupe l'ensemble des Users Stories qui permettent de définir les différents besoins envers la solution.

2.1.1. Patient

En tant que patient …

> J'aimerais partager mon ressenti à tout moment de la journée
> Afin de que les personnes suivant mon état de santé soient tenus au courant

> Je veux pouvoir communiquer facilement avec le docteur/infirmière ?
> Afin de prévenir au mieux les personnes ayant accès au système en cas de soucis

> Je veux que mon état de santé soit partagé
> Afin de bénéficier d'un suivi régulier par les personnes habilités

2.1.2. Infirmier

En tant qu'infirmier…

> Je veux pouvoir être alerté automatiquement de l'état de santé de mon patient en cas de changement
> Afin de venir le traiter sur place et éventuellement contacter le médecin

> Je veux pouvoir être alerté directement en cas d'urgence si le patient me déclenche
> Afin de venir le traiter sur place et éventuellement contacter le médecin

> Je veux avoir un espace dédié
> Afin de connaître l'état de santé de mes patients et préparer mes interventions

2.1.3. Médecin

En tant que médecin…

> Je veux avoir un planning
> Afin de gérer mes rendez-vous avec mes patients

> Je veux pouvoir gérer les passages d'infirmier pour mes patients
> Afin de s'assurer que les patients soient suivi en présence

> Je veux être alerté en cas d'aggravation du patient sur le court terme
> Afin d'intervenir auprès du patient

> Je veux avoir le contact de l'infirmier
> Afin de pouvoir le contacter par téléphone lors d'une urgence du patient

> Je veux pouvoir personnaliser les intervalles de temps entre les différentes prises de données du patient sur des capteurs
> Afin d'avoir des données à jour, sans gêner la vie du patient

> Je veux pouvoir accéder à toutes les données de santé, ainsi qu'à des analyses de leur évolution
> Afin de diagnostiquer l'état de mon patient

> Je veux pouvoir prendre un rendez-vous d'installation pour mon patient
> Afin d'ajouter des nouveaux dispositifs pour suivre l'état de mon patient

> Je veux pouvoir gérer les profils de mes patients
> Afin de mettre à jour mon espace dédié

2.1.4. Personnel Médical

Cette section des User Stories regroupent les cas applicables à la fois au médecin et à l'infirmier

En tant que personnel médical (médecin/infirmier)...

> Je veux pouvoir noter les observations et retours recueillies lors ma visite chez le patient
> Afin que l'information soit facilement partagée entre les différents acteurs, cela permet d'informer de manière passive les proches ou le médecin de certains éléments, et permet de les analyser lors d'une future visite et/ou diagnostic du médecin

> Je veux être alerté en cas de changement soudain et important de l'état du patient
> Afin d'ajuster le traitement du patient

2.1.5. Proche du patient

En tant que proche du patient…

> Je veux connaître les prochains rendez-vous du patient (médecin, infirmière, technicien)
> Afin de suivre les visites du patient

> Je veux être averti si l'un des capteurs manque de batterie
> Afin d'assurer un suivi constant du matériel présent

> Je veux connaître les métriques environnementales
> Afin de s'assurer que les pièces de vie sont saines (thermomètre, qualité d'air)

> Je veux avoir un espace dédié

> Afin de connaître l'état de santé actuel du patient (non détaillé, ex : température normale, tension légèrement élevé etc)

> Je veux avoir des retours sur les visites du patient
> Afin d'être constamment informé de son état

2.1.6. Utilisateur

Cette partie des User Stories concerne les médecins, infirmiers et proches du patient.

En tant qu'utilisateur…

> Je veux pouvoir consulter le journal d'alerte (environnement, légère, grave et critique)
> Afin de rester informé des événements passés dont je n'ai pas été notifié

2.1.7. Administrateur

En tant qu'administrateur…

> Je veux avoir accès à tous les dispositifs déployés chez les clients
> Afin de les maintenir (mettre à jour, changer les piles des appareils,…)

> Je veux connaître l'état des services avec une interface dédiée
> Afin de les maintenir facilement

2.1.8. Technicien

En tant que technicien…

> Je veux une interface de configuration
> Afin de gérer les dispositifs présents dans le système (ajouter, supprimer, modifier,...)

2.2. Analyse globale

2.2.1. Appareils connectés

Cette partie se consacre principalement à présenter les différents capteurs et appareils mis à disposition des patients. Comme énoncé dans l'Hypothèse 1 (cf. 1.3), seuls des capteurs compatibles avec l'écosystème Elderspy peuvent être utilisés. Cela a comme avantage de contrôler toutes les entités composant l'écosystème Elderspy.

2.2.1.1. Métriques de santé

Capteur	Format de données	Fréquence d'envoi	Utilisation
Fréquence cardiaque	entier	~ 1s	Automatique, équipé sur le patient
Température du patient	flottant	déclencher par la prise	Prise faite par le patient
Détecteur de chute (accéléromètre)	booléen	déclencher par le capteur, à la détection d'une chute	Automatique, équipé sur le patient
Glucomètre	flottant	déclencher par la prise	Prise faite par le patient

2.2.1.2. Métriques d'environnement

Capteur	Format de données	Fréquence d'envoi	Utilisation
Qualité de l'air intérieur	flottant	(variable) 2 fois par jour (matin/soir)	Automatique, équipé dans les pièces
Température ambiante de la/des pièces	flottant	(variable) 2 fois par jour (matin/soir)	Automatique, équipé dans les pièces

2.2.1.3. Métriques mentales

Pour aller au-delà de la simple collecte de données de télémétrie, la solution e-santé propose de recueillir des informations sur le ressenti du patient, dans un processus fluide et accessible, ne nécessitant ni compétences informatiques ni dextérité particulière.

En cas de réticence de la part du patient, ce système peut être compensé par un système de formulaire simple, mais qui va demander au patient une familiarité et des compétences minimales avec l'informatique.

Les métriques psychologiques sont collectées via un formulaire interactif, guidé par un assistant vocal. Celui-ci pose les questions et gère la progression du formulaire. Le

patient répond à l'aide de sa voix, ce qui rend l'interaction plus intuitive et naturelle, tout en permettant de capturer les ressentis du patient avec une précision supérieure qu'un questionnaire écrit.

Si la personne âgée ne peut pas ou ne veut pas parler à l'assistant vocal, il lui est possible d'utiliser une alternative. Il s'agit d'un formulaire qui permet au patient de remonter ses ressentis lorsqu'il en a envie. Cela va contenir les informations suivantes :

Type de données	Format de données
Etat général	une saisie de texte pour donner son avis
Faim	input bouton radio (pas du tout, peu, moyen, normal)
Fatigue	input bouton radio (pas du tout, peu, moyen, normal)
Douleur	input bouton radio (oui / non) / une saisie de texte pour localiser
Ressenti chaleur	input bouton radio (très froid, froid, normal, chaud, très chaud)
Ressenti énergie	input bouton radio (très faible, faible, fort)

2.2.2. Passerelle - Edge Gateway

2.2.2.1. Objectif

Ce dispositif se concentre sur la centralisation des données et le pré-traitement des données envoyées par les différents capteurs. Câblé sur internet, il permet de faire la liaison entre le réseau du foyer (LAN) et le reste du système dans le cloud.

Ce dispositif permet le partage d'information et de flux de données entre les Things et le cloud. La edge gateway intègre également des fonctionnalités d'agrégation des données et de data processing des données reçues des différents capteurs (Things), ainsi que des fonctionnalités d'alerte simple, basées sur des seuils.

2.2.2.2. Centralisation

Il répond donc au besoin technique de raccord entre les dispositifs et le reste du système car les appareils distants n'ont pas la capacité (et dont ce n'est pas la portée) d'envoyer des données directement vers les serveurs. C'est pour cela que cette passerelle est essentielle à notre système.

2.2.2.3. Data processing

De plus, ce edge traite une première fois les données en définissant des points d'alerte (sur une fréquence cardiaque trop basse ou trop élevée par exemple). Ce pré-traitement s'occupe aussi de regrouper les données avant de les envoyer vers les serveurs, pour réduire la consommation de bande passante et de charge serveur, il y a moins de messages et donc moins de overhead.

2.2.2.4. Connexion

Pour réduire les pertes de données, cette passerelle est conçue pour persister ses données si une perte de connexion avec les serveurs est occasionnée. Pour s'assurer de la réquisition des données et surtout de l'envoi d'alerte, en cas de coupure de courant, ce dispositif comprend une batterie de secours.

2.2.3. Visualisation

Cette partie se dédie principalement à une analyse des besoins et à une définition des acteurs du système. Elle vient surtout détailler certaines Users Stories.

2.2.3.1. Médecins

Le médecin est l'un des utilisateurs du système, principalement du côté de la visualisation (mais aussi un acteur sur d'autres composantes du système). Le système lui permet d'être alerté lors de problèmes provenant de ses patients et de visualiser l'ensemble de leurs données. Le médecin n'a aucune restriction sur les données de ces patients pour qu'il puisse analyser comme il veut les évolutions des métriques.

2.2.3.2. Infirmiers

Les infirmiers sont un personnel essentiel pour le patient, qui s'en occupe de manière régulière. C'est pourquoi ce système doit aussi comprendre un espace dédié pour leur analyse, et leurs interventions afin d'intervenir correctement auprès de leur patient. Ils ont à leur disposition un maximum de données sur court et moyen terme, ainsi que le système d'alertes pour intervenir au plus vite auprès du patient.

2.2.3.3. Proches du patient

Le système est aussi utile aux proches du patient pour connaître, sans besoin de présence, son état. Il permet aussi d'être averti en cas de problème. Pour ne pas inquiéter les proches, le système doit "épurer" les informations mises à leur disposition.

2.2.3.4. Données partagées

Voici les différentes données partagées par ElderSpy pour les différents acteurs :

Acteurs	Données visibles
Médecins	- Les données brutes sous forme de graphes - Les rapports d'observations - Les rapports d'analyses poussés - Les rapport de feedback patient - Récapitulatif des alertes - Le planning du corps médical en charge du patient
Infirmiers	- Les récentes données brutes sous forme de graphes - Les rapports d'observations - Les rapport de feedback patient - Récapitulatif des alertes - Le planning du corps médical en charge du patient - Statut des installations
Proches	- Données simplifiées pour chaque métrique - Les rapports d'observations - Les rapport de feedback patient - Récapitulatif des alertes - Le planning du corps médical en charge du patient - Statut des installations

2.2.3.5. Classification des alertes

Les acteurs sont alertés en fonction du type d'urgence. Voici les différentes urgences prises en compte par ElderSpy :

Types d'urgence	Métriques concernées	Exemple
Urgence environnement	Qualité de l'air intérieure, Température ambiante	Qualité de l'air excédent les 900ppm, température ambiante dépassant les 26°C ou descendant en dessous de 18°C
Urgence médicale légère	Chute, Température corporelle, Glucomètre	Chute mais le patient se relève, Fièvre légère (37,8 à 38,5°C), Hypothermie légère (35 à 36°C), Hypoglycémie légère à modérée (< à 70mg/dL), Hyperglycémie modérée (180 à 250 mg/dL), Bradycardie légère à modérée (50 à 60 bpm), Tachycardie modérée (100 à 120 bpm)
Urgence médicale grave	Chute, Température corporelle, Glucomètre, Fréquence cardiaque	Chute sans se relever, Fièvre sévère (> à 38,5°C), Hypothermie sévère (< à 35°C), Hypoglycémie légère à modérée (< à 55mg/dL), Hyperglycémie modérée (> à 250 mg/dL), Arrêt cardiaque, Bradycardie sévère (< à 50 bpm), Tachycardie sévère (>120 bpm), Fréquence cardiaque irrégulière

Les valeurs de fréquence cardiaque sont des valeurs par défaut et peuvent être adaptées aux patients.

Les alertes ne sont pas dirigées vers les mêmes acteurs en fonction du type d'urgence comme le montre le tableau ci-dessous :

Types d'urgence	Acteurs alertés
Urgence environnement	Infirmier, Proche
Urgence médicale légère	Infirmier, Proche
Urgence médicale grave	Médecin, Infirmier, Proche

Cependant, toutes les alertes sont ajoutées à un journal d'alerte ce qui permet d'avoir un historique des urgences passées et permet également aux acteurs d'avoir un accès aux alertes pour lesquelles ils n'ont pas été notifiés. En particulier, pour le médecin, lors d'une alerte suite à une urgence médicale légère ou environnement.

2.2.4. Maintenance

Dans le contexte de cette solution, et plus essentiellement des appareils (dispositifs connectés et passerelle), la maintenance des systèmes est cruciale.

2.2.4.1. Objectif

Les besoins s'articulent sur différentes échelles : la gestion des dispositifs et celle du système. Pour les dispositifs, le besoin va concerner les techniciens qui interviennent dans les foyers et les proches du patient pour maintenir les appareils actifs. Et pour le système, il faut maintenir le système et les dispositifs à distance, ce qui s'adresse aux rôles d'administrateur et de technicien.

2.2.4.2. Maintenance présentielle

D'après l'Hypothèse 2, le technicien a besoin d'accéder à un espace de maintenance, lui permettant d'ajouter ou de supprimer des appareils sur la passerelle, vérifier le fonctionnement des dispositifs distants et des capteurs et de pouvoir intervenir directement sur la passerelle en cas de besoin (dysfonctionnement, perte de connexion, mise à jour à forcer).

Du côté des proches du patient, le besoin se situe surtout sur la maintenance d'activité des appareils distants et des capteurs. Le système doit avertir le patient en cas de panne d'un dispositif causé par la perte d'alimentation.

2.2.4.3. Maintenance distancielle

Il est essentiel de pouvoir accéder à distance à toutes les passerelles pour réduire les déplacements et l'intrusion des administrateurs dans les domiciles des patients. Cet accès permet d'intervenir sur l'appareil et de s'assurer de son bon fonctionnement (mise à jour forcée, statut des appareils). Pour ce faire, un reverse SSH tunnelling est utilisé pour piloter les machines à distance.

2.2.5. Gestion utilisateur

Cette partie concerne la gestion de tous les utilisateurs, comprenant les patients, les médecins, les infirmiers et les proches des patients

2.2.5.1. Médecins

D'abord, les médecins sont les managers de leur propre éco-système. C'est à eux que revient la gestion (ajout, suppression, management, planification…) de leurs patients et celle des infirmiers.

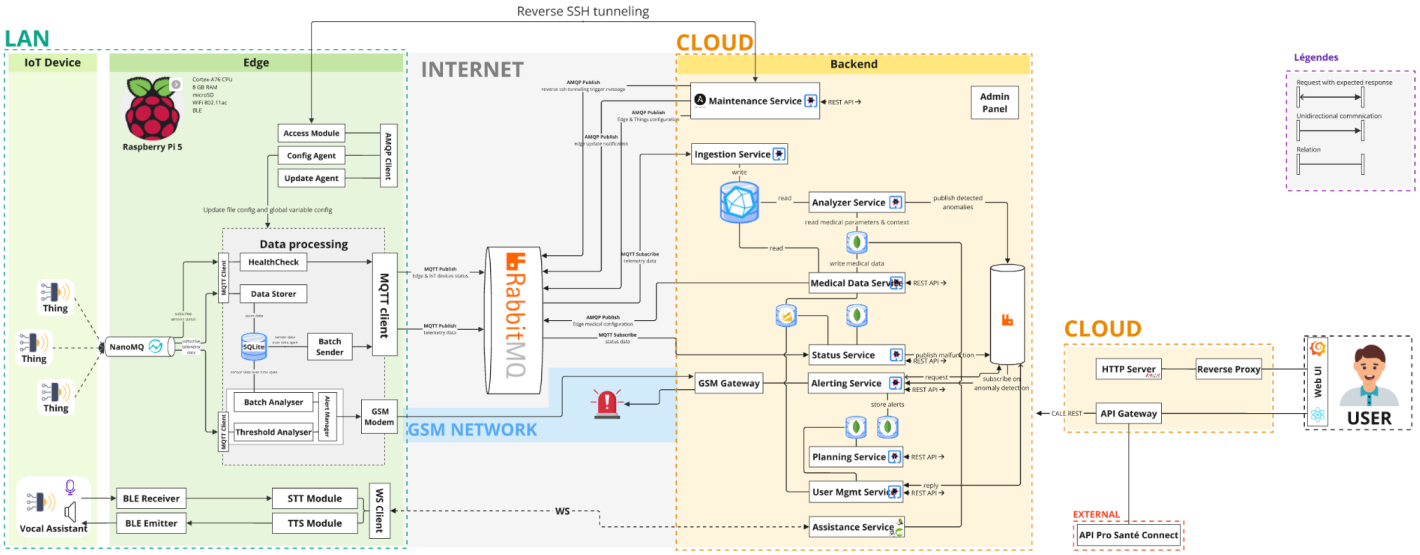
Un médecin qui veut utiliser le système peut se connecter ou créer un compte. Pour s'identifier en tant que médecin certifié, le système utilise le portail de connexion (Hypothèse 5).

2.2.5.2. Proches des patients

Pour répondre aux besoins des proches (suivi du proche), lorsqu'un patient est ajouté au système par le médecin, cela crée un accès partagé par tous les proches du patient (système d'identification). Le système est accessible depuis n'importe quel appareil.

3. Architecture

3.1. Conception générale



3.2. Analyse et choix d'architecture

Cette section aborde les choix et les justifications de choix d'architecture et les choix de technologies et d'interactions entre module de l'architecture.

3.2.1. Architecture des composantes

3.2.1.1. Backend

L'architecture du Backend retenue pour la partie Cloud du système est la microservice. Elle permet la scalabilité (horizontale) aisée des services et donc s'adapter facilement aux variations de charge travail par la gestion dynamique des instances. Elle donne aussi une forte résilience entre les services permettant d'isoler les défaillances des services et augmente donc la disponibilité du système. Il peut même supprimer tout risque d'interruption en déployant des instances en compensation. Dans un contexte de traitement de données de santé, il est aussi important de réduire les accès aux données et la segmentation des services permet de renforcer le respect des réglementations. Dans l'optique d'un déploiement de solution sur le Cloud, cette architecture comprend aussi une compatibilité avec ce type d'environnement spécifique, ce qui facilite aussi les opérations automatisées, la surveillance et le versionning

C'est pourquoi l'architecture microservice a été privilégiée aux autres pour la partie Backend.

3.2.1.2. Edge

Une architecture de Edge Orchestration Architecture a été posée sur le Edge avec des agents (configuration et mise à jour) avec un module de Edge Stream Processing. Cette architecture propose un traitement des données en temps réel, une réduction de la charge réseau en prétraitant les données pour envoyer seulement l'essentiel. Il met en avant une évolutivité simple à mettre en place en ajoutant des modules de traitement supplémentaire et de nouveaux agents. De plus, il était important de traiter la résilience du système pour

permettre au Edge de continuer à fonctionner malgré une déconnexion (coupure réseau) avec le Backend.

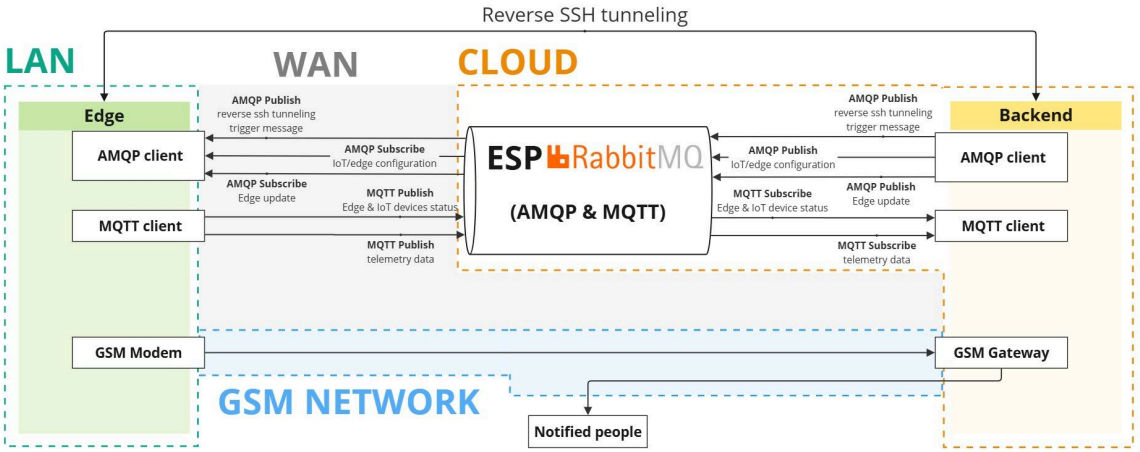
Cette architecture appliquée au passerelle permet réactivité, efficacité réseau et adaptabilité.

3.2.2. Communication et transmission des données

Dans cette partie, il sera question d'analyse et de choix technologique dans la transmission de données entre les différentes composantes de notre système (IoT device, Edge Gateway, Backend).

3.2.2.1. Données entre Edge Gateway et Backend

Dans les communications Edge vers Backend (mais aussi inversement), un grand nombre de données transitent entre ces deux composantes.



Tout d'abord, les communications entre les deux entités sont pensées pour être asynchrone. Puisque la montée des données n'exige pas de réponse du Backend pour le Edge, le but est de réduire le couplage entre ces deux composantes. En le réduisant, le système permet, d'une part, une forte tolérance aux pannes, et d'autre part une meilleure gestion de la charge et, de par l'utilisation de protocole plus léger, optimise la bande passante et s'adapte donc aux ressources limitées des Edges.

Les besoins des communications de données s'articulent entre la transmission des données des appareils IoT et des mises à jour pour les différents appareils chez le patient (Edge, IoT).

Dans la suite, il sera question d'analyse des technologies répondant à notre besoin. Le but est de déterminer celle qui correspond le mieux aux besoins du système.

Technologie	Avantages	Inconvénients	Cas d'utilisation
AMQP	- Fiabilité élevée, tolérance aux interruptions - Supporte des priorités et des accusés de réception - Sécurité robuste avec AMQP	- Plus lourd pour des appareils limités en ressources - Complexité d'intégration	Bon pour des systèmes avec des messages critiques
MQTT	- Léger, peu de consommation en bande passante - Tolérant aux interruptions - Conçu pour des appareils IoT à faibles ressources	- Moins adapté aux messages lourds	Idéal pour des envois fréquents de données simples par les Edge avec peu de puissance
Kafka	- Haute performance pour de grandes quantités de données - Évolutif, permettant de traiter des flux lourds de données	- Plus lourd et exige des ressources importantes - Non conçu pour des appareils IoT en temps réel	Bon pour des backend avec des données en streaming à haute fréquence
CoAP (sur UDP)	- Conçu pour des environnements à faible bande passante - Très léger et sécurisé avec DTLS	- Moins fiable sur des réseaux instables (basé sur UDP) - Limité pour des applications nécessitant de la fiabilité	Utile pour des appareils IoT très limités avec des données peu critiques

Le protocole MQTT, utilisé pour les topics fréquemment utilisés, garantit un faible overhead par message, permet un routage assez fin basé sur des topics et inclut une fonction d'observabilité pour détecter les pertes de connexion et trois niveaux de qualité de services dont "Exactly once". Le protocole AMQP, de son côté, est réservé aux flux de données exceptionnels. Sa fonction de multiplexage permet de gérer plusieurs topics via une seule connexion TCP, ce qui est idéal pour les cas de trafic faible qui sont entre autres les topics de maintenance, de mise à jour système et de configuration…. De plus, le mécanisme de acknowledgment du protocole AMQP garantit la bonne réception des informations et une qualité de service rudimentaire.

Afin de faciliter l'échange des données de télémétrie entre les edges et le cloud, le système adopte donc une architecture pilotée par événements (event-driven) basée sur un broker RabbitMQ. Ce système de messagerie asynchrone, utilisant le modèle pub/sub, est adapté pour l'architecture orientée événements. Il élimine le besoin d'une architecture client-serveur et le besoin de maintenir un registre des adresses IP de chaque équipement du edge, ou bien l'usage de mécanisme de polling. Un point fort de RabbitMQ est qu'il

permet de prendre en charge les protocoles MQTT et AMQP en une seule et même instance, chacun de ces protocoles offre des avantages spécifiques aux cas d'utilisation.

3.2.2.2. Alertes entre Edge Gateway et Backend

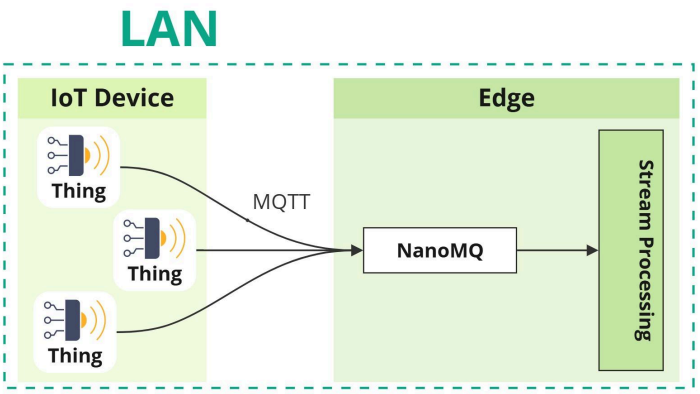
Pour pallier l'intermittence et la faible fiabilité des réseaux privés dans les foyers, nous avons choisi d'utiliser le réseau GSM pour la transmission des alertes, assurant ainsi une connexion fiable et continue entre les équipements edge et le cloud.

3.2.2.3. Connectivité de maintenance entre Edge Gateway et Backend

Pour la maintenance, nous avons mis en place un mécanisme de reverse SSH tunneling pour accéder aux équipements edges. Ce mécanisme repose sur la connexion TCP existante et utilisée par les topics basés sur AMQP. Le topic dédié à la maintenance envoie un signal (trigger) au client pour que celui-ci initie un tunnel SSH depuis le côté edge. Une fois ce tunnel établi, le service de maintenance peut créer un second tunnel au sein du premier, ce qui permet de contourner le proxy du client et de garantir un accès direct aux équipements en périphérie. Cela évite le recours au mécanisme polling côté edge, améliorant ainsi la disponibilité de la connexion ssh entre le backend et le edge.

3.2.2.4. IoT et Edge Gateway

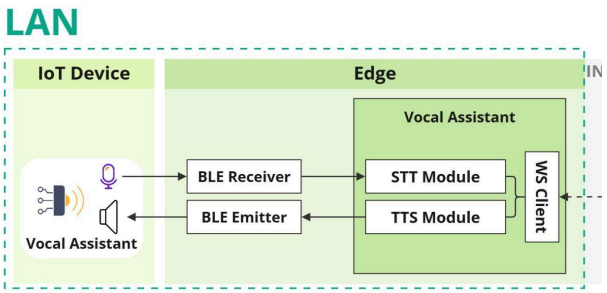
3.2.2.4.1. MQTT



La edge gateway intègre un système permettant de réaliser un minimum de stream processing avant de déléguer des analyses plus complexes aux capacités de calculs élevées du cloud. L'analyse des données repose sur des seuils définis ; par exemple, lorsqu'une fréquence cardiaque est trop basse ou trop élevée, une alerte est générée.

Pour assurer cette fonctionnalité, nous utilisons NanoMQ en version slim, configuré strictement pour agir comme un broker MQTT. Pour optimiser la bande passante et le trafic Internet, les données envoyées au cloud sont d'abord agrégées en paquets regroupant les dernières informations sur une période donnée. Cette approche est possible grâce à l'analyse préalable des données, qui permet d'écarter toute alerte immédiate. Ainsi, les données sont transmises sans la contrainte d'un traitement urgent. Elles peuvent donc être utilisées pour des analyses à plus long terme, telles que la recherche de patterns, le croisement de différentes données et le traitement basé sur l'intelligence artificielle.

3.2.2.4.2. BLE



La edge gateway intègre un deuxième système permettant de traiter les commandes vocales effectuées par les patients.

Afin de traiter cette deuxième fonctionnalité, la technique de communication sans fil BLE (Bluetooth Low Energy) a été mise en place pour la réception et l'émission des commandes. Elle est utile pour échanger des données sur de courtes distances. Deux modules BLE ont été utilisé :

- BLE Receiver : qui est responsable de la réception des données (provenant des dispositifs)
- BLE Emitter : qui a pour rôle de diffuser des informations (du Edge vers les dispositifs)

Ces deux modules assurent une communication fluide et énergétiquement efficace mais permettent aussi au Edge de collecter et de diffuser des informations en temps réel ce qui est particulièrement utile dans le cadre de ElderSpy.

3.2.3. Stockage

Le stockage des données dépend de leur type. Il va exister plusieurs bases de données. Voici donc une étude préalable des différents critères et des technologies comparées pour répondre à nos besoins techniques.

3.2.3.1. Critères

Dans le choix des bases de données qui devront répondre aux différents besoins de notre application, une liste de critères a été établie pour déterminer la technologie suffisante pour répondre à nos besoins.

Sécurité et Confidentialité des Données

Certaines bases devront offrir des mécanismes de chiffrement avancés pour respecter les réglementations, notamment la confidentialité des données de santé (RGPD, HIPAA). Elle doit aussi être configurable pour empêcher tout accès externe non autorisé.
Cela ne concerne pas toutes les données du système.

Fiabilité et Haute Disponibilité

Les données doivent être constamment accessibles, et les bases de données choisies doivent supporter la tolérance aux pannes et la réplication pour éviter toute perte de données critiques.

Performances et Scalabilité

Certaines bases de données doivent pouvoir gérer de grandes quantités de données en temps réel (par exemple, des métriques), tout en maintenant des performances élevées. La scalabilité horizontale ou verticale doit être simple pour adapter les capacités aux besoins.

Simplicité de Maintenance et de Déploiement

La technologie doit être facile à mettre en place, avec des outils de gestion et de monitoring intégrés, pour réduire les efforts de maintenance. La base de données doit également être compatible avec les environnements de conteneurisation ou d'autoscaler pour simplifier les déploiements.

Toutes proportions gardées, la tolérance à la complexité peut être modulée selon les besoins plus ou moins spécifiques des cas d'utilisation.

Type de Données Stockées

Les besoins spécifiques, comme le stockage de métriques, de données relationnelles, de paires clé-valeur et de documents, influencent le choix. Il faut que la technologie propose un modèle correspondant, et qu'elle soit dédiée à ce type de données.

Support pour le Langage de Requête

Les bases de données doivent supporter un langage de requête adapté aux données stockées, et que ce langage soit accessible, ou que la complexité ou la spécificité du langage soit justifiée.

Légèreté

Certaines bases de données, selon son utilisation, se doivent d'être légères, car les ressources disponibles peuvent être limitées.

3.2.3.2. Analyse des bases de données

Premièrement, il faut pouvoir déterminer quel type de base de données il faut pour un besoin particulier, ou plusieurs besoins.

Type	**Description**	**Exemple de Bases de Données**
Relationnel	Tables, clés, relations	PostgreSQL, MySQL
Séries temporelles	Données indexées par le temps	InfluxDB, TimescaleDB
Clé-Valeur	Stockage rapide de paires clé-valeur	RocksDB, Redis
Orienté Document	Données semi-structurées en documents BSON/JSON	MongoDB, CouchDB

Ensuite, il faut déterminer quelle technologie est la plus adaptée à nos cas d'utilisation.

Technologie	Type de Base de Données	Cas d'Utilisation	Avantages	Inconvénients
InfluxDB	Série Temporelle	Données de métriques	Très performant pour séries temporelles, bon support de Flux	Moins adapté pour données non temporelles
PostgreSQL	Relationnel	Données utilisateur	SQL robuste, transactions ACID, haute disponibilité	Peut être lent pour grands volumes de données non structurées
RocksDB	Clé-Valeur	Clés-valeurs rapides	Très rapide, faible latence, embeddable	Ne gère pas les données relationnelles ou complexes
MongoDB	Orienté Document	Alertes, statuts	Flexible pour documents, support JSON/BSON, bonne scalabilité	Moins performant pour transactions ACID
Prometheus	Série Temporelle (Monitoring)	Monitoring et alertes	Spécialisé pour le monitoring, très rapide	Moins adapté pour données historiques importantes
MySQL	Relationnel	Données utilisateurs	Bon support SQL, largement utilisé	Moins de fonctionnalités avancées, transactions moins robustes
Redis	Clé-Valeur en mémoire	Cache rapide, sessions utilisateur	Ultra-rapide, idéal pour cache	Ne stocke pas durablement, nécessite beaucoup de RAM

SQLite	Relationnel	Applications mobiles, systèmes IoT	Extrêmement légère, très portable, facile d'intégration	Pas optimisée pour des bases de données de grande taille ou pour un nombre d'écritures intensif
Firebird Embedded	Relationnel	Applications mobiles, systèmes IoT	Gestion de transactions ACID plus complexe, supporte SQL avancé	Plus lourd que SQLite, nécessitant davantage de ressources

3.2.3.3. Métriques

Pour les métriques, la base de données se doit d'être temporelle pour simplifier la récupération de données pour la visualisation mais aussi pour les traitements qui leur seront appliqués. Il s'agit seulement de stockage de données et n'a aucunement besoin de monitoring (comme pourrait proposer Prometheus). De plus, de part la quantité possiblement importante de données, la scalabilité de la base de données est un critère très important pour ce système.

La technologie la plus adaptée à ce cas d'utilisation est InfluxDB, qui stockera donc les métriques.

3.2.3.4. Mapping de clés cryptées

Pour référer à la partie sur l'Anonymisation, certaines données demandent un format de clé-valeur qui font référence à des clés cryptées. Et pour des raisons de sécurité et de confidentialité importantes, le choix de la base de données s'est orienté sur RockDB.

3.2.3.5. Données personnelles des utilisateurs

La base de données dédiées aux données personnelles des utilisateurs doit proposer le stockage de structures complexes de données et qui respecte les principes ACID. Les données qui y seront stockées sont les comptes utilisateurs avec leur planning. Cette base de données est centrale, c'est-à-dire que plusieurs microservices devront accéder à cette base. Certes, cela les couple plus à la BDD, mais gère le même domaine de données (comme dit précédemment).

Cependant, les microservices réalisent des opérations distinctes, donc les responsabilités peuvent être séparées. Pour se faire, l'utilisation de PostgreSQL permet de répondre aux critères énoncés précédemment.

3.2.3.6. En embarqué

La solution en embarqué (sur les edges, plus précisément) doit surtout répondre au critère de légèreté. Puisque la quantité de données qu'elle aura à gérer reste assez minimes et permet surtout un stockage des données temporaires (pour l'envoi par paquet et pour les

cas de défaillance réseau). Dans les solutions étudiées, la base de données répondant au plus aux critères est SQLite.

3.2.3.7. Autres données

Pour les données n'ayant pas de contraintes particulières et utilisées par un seul microservice doit surtout répondre à des critères de disponibilités, de scalabilité et du modèle de données. Pour répondre à ces critères, MongoDB a été choisi pour les différentes bases de données qui regroupent paramètres médicaux, rapport d'analyse, compte rendu et observation, les statuts d'appareils et les alertes lancées.

3.2.4. Scalabilité

L'infrastructure Cloud nécessite une mise à l'échelle pour pouvoir soutenir la charge du système. Pour cela, le système de scale doit remplir certains critères, selon l'utilisation, dans ce cas-ci, la mise à l'échelle d'une application industrielle.

Orchestration des Conteneurs

Gestion de la distribution, la mise à l'échelle et la supervision des conteneurs.

Autoscaling et Répartition de Charge

Capacités de mise à l'échelle automatique en fonction des besoins et gestion efficace de la répartition de charge.

Portabilité et Compatibilité Cloud

Capacité de déployer sur plusieurs clouds ou en environnement hybride.

Facilité de Gestion et Complexité

Facilité de configuration, de gestion et d'apprentissage pour les équipes.

Technologie	Description	Avantages	Inconvénients	Cas d'Utilisation Idéal
Kubernetes	Orchestrateur de conteneurs open-source	Scalabilité avancée, support multi-cloud, écosystème mature	Complexité élevée, courbe d'apprentissage importante	Applications à grande échelle, multi-cloud, hybride
Docker Swarm	Outil de clustering intégré à Docker	Plus simple à mettre en œuvre que Kubernetes, intégration native avec Docker	Moins puissant pour la mise à l'échelle avancée	Petits clusters, applications simples, prototypage

Nomad (HashiCorp)	Orchestrateur léger pour conteneurs et VMs	Extrêmement flexible (contient Docker, VMs, et plus), facile à configurer	Moins de fonctionnalités d'écosystème par rapport à Kubernetes	Scalabilité pour services légers ou applications distribuées
OpenShift	Distribution Kubernetes avec des outils avancés	Interface utilisateur améliorée, intégration DevOps et CI/CD intégrée	Plus coûteux, complexité d'installation	Grandes entreprises nécessitant support et outils intégrés
Amazon ECS	Service de conteneurs AWS natif	Intégration directe avec l'écosystème AWS, simple à utiliser	Limité à AWS, dépendance aux services AWS	Applications entièrement sur AWS, besoin d'intégration rapide

Kubernetes est la solution retenue pour notre infrastructure car il répond aux questions d'application Cloud à grande échelle.

3.2.5. Technologies et développement

3.2.5.1. Microservices

Pour une architecture microservices, il est important de prendre en considération les critères suivants.

Performances et scalabilité

Les services doivent pouvoir gérer des charges variables et évoluer sans impacter la performance.

Facilité de développement et de maintenance

Les choix technologiques doivent favoriser la rapidité de développement et la facilité de maintenance, surtout dans un environnement en microservices où les composants sont multiples.

Support pour le cloud et la conteneurisation

Les services doivent être compatibles avec les services de conteneurisation et les solutions cloud (Kubernetes ou Docker, par exemple).

Ecosystème

L'accès à une documentation riche peut s'avérer importante lors du développement et de la maintenance du service.

Facilité de surveillance et de gestion

Les frameworks et langages doivent être compatibles avec les outils de monitoring et de gestion des microservices.

Une analyse de framework et d'outils pour le développement de microservices s'impose. L'analyse donnera aussi le langage de développement des services.

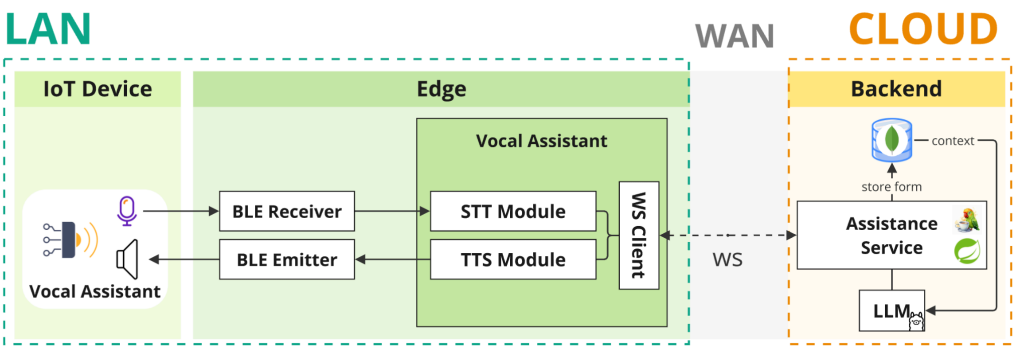
Technologie	Langage	Avantages	Inconvénients	Cas d'Utilisation
Spring Boot	Java / Kotlin	Écosystème mature, gestion avancée des microservices (Spring Cloud), intégration facile avec outils Java	Plus lourd que d'autres solutions, démarrage plus lent	Projets d'entreprise, services nécessitant résilience et intégrations robustes
Quarkus	Java / Kotlin	Optimisé pour le cloud et Kubernetes, démarrage très rapide, faible empreinte mémoire	Moins mature que Spring Boot, support limité pour certaines intégrations	Microservices cloud natifs, environnements Kubernetes
NestJS	TypeScript / JavaScript	Parfait pour les API REST, compatible avec Node.js, rapide et léger, basé sur TypeScript	Moins performant pour des traitements lourds que Java	API légères, microservices pour frontends ou applications temps réel
Express.js	JavaScript	Simplicité et rapidité de développement, très léger, vaste communauté	Pas adapté aux services complexes, support limité pour la scalabilité	Prototypes, microservices simples, API de front-end
Go-Micro	Go	Conçu pour les microservices, haute performance, très léger, rapide	Moins de support pour des fonctionnalités avancées	Microservices orientés performances, services à forte charge de requêtes

L'uniformisation du système n'est pas obligatoire, chaque service peut être développé sur différentes technologies et Frameworks, voire même être développé "from scratch". Pour simplifier le développement et la maintenance des services et avoir une expertise ciblée des équipes, les microservices seront développer sur le même Framework.

Les besoins du système envers les microservices s'articulent sur des designs Cloud et sur la mise à l'échelle des différents services pour subvenir aux charges. Les services n'ont pas le besoin d'une intégration spécifique, mais ont besoin d'être rapidement déployé par Kubernetes, qui va gérer les charges des services.

La solution retenue pour ces services, apart en cas de spécificité (point suivant), se baseront sur la technologie Quarkus qui remplit les critères.

Le microservice de gestion du feedback patient utilise le framework Spring Boot, contrairement aux autres microservices. Ce choix s'explique par la meilleure intégration et la documentation plus complète que Spring Boot offre pour Langchain4j par rapport à Quarkus. En effet, Quarkus doit développer sa propre implémentation de certaines bibliothèques, car il vise une compatibilité optimale avec les environnements natifs. Dans l'exemple de l'assistant vocal, la mise en place d'une IA générative locale a été imaginée pour répondre aux besoins des patients. Les réponses vocales du patient sont converties en texte grâce à une technologie de reconnaissance vocale (speech-to-text) puis transmises vers le cloud pour traitement. Un modèle de langage (LLM), hébergé localement dans le Cloud, génère la suite des questions ou une réponse adaptée, en prenant également en compte le contexte médical du patient pour affiner et personnaliser l'échange. La réponse est ensuite envoyée vers la passerelle, où un module de synthèse vocale (text-to-speech) la restitue sous forme audio via un haut-parleur.



3.2.5.2. Edge

Ici, il faut surtout se baser sur des technologies et des langages de programmation léger et rapide au traitement. Pour ce faire, voici les critères de sélection pour une programmation embarquée.

Légèreté et Efficacité des Ressources

Le langage doit minimiser la consommation de mémoire et CPU pour être adapté aux ressources limitées des dispositifs Edge.

Performances en Temps Réel

Le langage doit offrir de bonnes performances pour traiter les données en temps réel, permettant de détecter des seuils et de lancer des alertes sans latence excessive.

Portabilité et Compatibilité

Il doit être facilement déployable sur des systèmes embarqués et compatible avec les systèmes de communication utilisés pour la montée de données.

Support pour le Traitement de Données

Le langage doit pouvoir gérer efficacement l'analyse et la gestion des seuils avec des bibliothèques adaptées.

Langage	Avantages	Inconvénients	Cas d'Utilisation Edge
Rust	Très performant, gestion fine de la mémoire, léger, sécurisé	Syntaxe complexe, moins de bibliothèques pour traitement de données avancé	Analyse rapide de données, gestion de seuils, alertes temps réel
Go	Légèreté, rapide, supporte la concurrence, idéal pour microservices	Moins adapté pour des traitements de données complexes, moins de bibliothèques pour le calcul scientifique	Montée de données, gestion d'alertes, traitements légers avec seuils
Python	Simple, riche en bibliothèques pour analyse (NumPy, Pandas), rapide à développer	Moins performant et plus lourd en ressources que Rust ou Go	Prototypage rapide, analyse de données de base, traitement avec seuils
C	Ultra performant, très léger, contrôle total des ressources	Complexe, manque de bibliothèques modernes, difficile à maintenir	Applications avec contraintes extrêmes, faible empreinte mémoire
JavaScript (Node.js)	Léger, rapide, large écosystème, bon pour traitements asynchrones	Moins performant pour calculs intensifs, nécessite plus de mémoire que C ou Rust	API légères, traitements de données avec seuils simples, montées de données
C++	Performant, gestion manuelle de la mémoire, programmation orientée objet, nombreuses bibliothèques	Complexité accrue par rapport à C, consommation mémoire parfois élevée	Traitement de données en temps réel, applications modulaires avec logique complexe, gestion d'objets réutilisables

De part son orientation objet et la structure synchrone du Stream Processing, le C++ est le langage remplissant les critères comme support de langage pour de la programmation sur Edge pour ses performances et les bibliothèques disponibles.

3.2.6. Maintenance

Les besoins de maintenance des appareils et du système est un élément clé de l'infrastructure. Pour cela, chaque partie du système à une manière de résoudre les problèmes et de se mettre à jour.
En particulier, la passerelle Edge pourra effectuer des mises à jour suite à des modifications dans le back (voir Maintenance). Les Things seront entretenus par les membres de la famille et le backend utilise des logs pour faciliter la maintenance en cas de problèmes (voir ELK - Log Service).

3.2.7. Visualisation

Le serveur Web donne accès à la visualisation des données aux différents acteurs. Le choix des technologies se porte principalement sur les critères suivants.

Flexibilité et Personnalisation

Besoin d'une interface adaptable et personnalisable pour répondre aux exigences spécifiques des utilisateurs.

Visualisation en Temps Réel

La solution doit pouvoir gérer des données en temps réel pour un suivi continu et rapide des métriques.

Séries Temporelles et Métriques

Nécessité de traiter et de visualiser efficacement des données de séries temporelles pour des métriques.

Développement et Intégration

Le front-end doit s'intégrer avec d'autres composants et services et être facilement extensible.

Interactivité et Expérience Utilisateur

Interface interactive pour améliorer la navigation, l'exploration et l'interaction avec les données.

Évolutivité et Maintenabilité

Solution évolutive et facile à maintenir, pour s'adapter aux changements de besoins.

Technologie	Description	Avantages	Inconvénients	Cas d'Utilisation Idéal
Grafana	Spécialisé dans la visualisation de séries temporelles, intégré aux sources de données	Conçu pour les métriques, plugins de visualisation, alertes	Moins de flexibilité pour des applications dynamiques	Monitoring en temps réel, suivi des métriques IoT
Tableau	Interface drag-and-drop pour visualisation et exploration de données	Très visuel, riche en fonctionnalités, supporte des données variées	Coût élevé, moins adapté pour un suivi en temps réel	Analyses de données complexes, tableaux de bord BI
Power BI	Outil Microsoft pour visualisation de données et reporting	Intégré à l'écosystème Microsoft, facile pour les analyses de BI	Moins adapté pour des métriques en temps réel	Rapports BI, analyses de données, reporting
Apache Superset	Open-source, compatible avec de nombreuses bases de données	Gratuit, flexible, support SQL, nombreuses intégrations	Interface moins intuitive, moins performant pour très grands jeux de données	Analyses BI open-source, intégrations SQL

Pour répondre aux besoins de visualisation des données temporelles, l'outil le plus adapté pour cette utilisation est Grafana, avec un monitoring en temps réel de métriques provenant d'IoT. Il vient répondre au besoin de flexibilité, de suivi et d'évolutivité (avec les plugins).
En intégrant Grafana dans une application web, on ajoute une dimension supplémentaire de personnalisation. L'intégration se fait avec React, bibliothèque open-source pour le développement d'applications micro-frontend.

3.2.8. Hardware et hébergement

3.2.8.1. Hardware

L'appareil choisi pour assumer la partie passerelle du système est la Raspberry Pi 5. Elle permet d'avoir des composants physiques nécessaires au fonctionnement du edge, tel qu'un module WiFi et Bluetooth et des spécifications supérieures aux besoins des traitements et des envois de données pour des questions de performance de traitement et

de stockage de données. La configuration peut être trop élevée pour le besoin, mais permet d'avoir une première base d'intuition sur les choix hardware.

Il faudrait étudier plus en profondeur les besoins réels après le développement des services nécessaires au Edges.

3.2.8.2. Hébergement

L'hébergement est un point crucial. Il doit assurer les normes sur le stockage de données médicales (HDS). Pour cela, un grand choix d'hébergeur Cloud proposant ces services normés est disponible.
Avec le site Esanté du gouvernement français (https://esante.gouv.fr/produits-services/hds), voici une analyse des hébergeurs.

Fournisseur Cloud	Description et Avantages	Inconvénients	Cas d'Utilisation Idéal
Microsoft Azure (France)	Offre une certification HDS pour ses services, ainsi qu'une forte sécurité et des outils avancés de conformité. Intégration avec Power BI pour le traitement de données.	Peut être coûteux selon l'échelle et les configurations requises.	Applications santé multi-services avec besoin de scalabilité.
Amazon Web Services (AWS) en France	AWS a une offre HDS limitée aux services hébergés en région parisienne. Possède des outils avancés pour la sécurité et la conformité (AWS Shield, GuardDuty).	Certification HDS uniquement pour les services en France, complexité d'intégration.	Projets nécessitant des solutions scalables, analytiques.
OVHcloud	Fournisseur cloud européen certifié HDS, avec une infrastructure basée en France. Connu pour son support local et sa conformité stricte avec le RGPD.	Moins de services avancés que AWS ou Azure.	PME/ETI en santé, besoin de souveraineté européenne.
Scaleway	Basé en France et certifié HDS, Scaleway propose des solutions d'hébergement adaptées aux besoins en santé. Offre un bon rapport coût-performance.	Moins d'options d'intégration avec des services tiers que les grandes plateformes internationales.	Projets de petite à moyenne échelle nécessitant une solution HDS flexible.

Google Cloud Platform (GCP) en France	Google propose également des services certifiés HDS en France, avec des outils avancés de machine learning et de data analytics intégrés.	Complexité de configuration, coût élevé pour certaines ressources.	Projets de santé nécessitant des analyses de données avancées et IA.

Pour des raisons de position géographique et de souveraineté politique, le choix d'hébergeur sera OVH, qui permet la scalabilité des services, tout en conservant une norme HDS imposée par le gouvernement européen.

3.2.9. Broker

Dans cette partie, il est question d'argumenter le choix des technologies pour les communications asynchrones : les queues. Selon le segment de communication (IoT<->Edge, Edge<->Back, entre microservices), les critères de sélection change : cela peut demander de la légèreté et des fonctionnalités réduites à l'essentiel sur la partie Edge, ou bien peut demander une flexibilité importante sur les protocoles et les fonctionnalités dans le Cloud.

Segment de Communication	Broker Utilisé	Protocole(s) de Communication	Rôle et Justification
IoT Devices ↔ Edge	NanoMQ	MQTT	NanoMQ via MQTT est utilisé pour transférer les métriques des dispositifs IoT vers le Edge de manière efficace et légère, correspond à l'environnement du Edge avec une faible bande passante entre les entités
Edge ↔ Backend	RabbitMQ	AMQP & MQTT	RabbitMQ avec AMQP & MQTT propose des échanges bidirectionnels entre le Edge et le Backend pour les données (métriques, configuration), garantissant une flexibilité dans les types de message et assurant la fiabilité des communications Le protocole AMQP est spécifiquement utilisé pour les mises à jour des dispositifs et du Edge

Communications internes	RabbitMQ	MQTT	En interne, l'utilisation de RabbitMQ en MQTT assure une distribution des messages rapide et légère pour les communications de statut et de métriques au sein du système

3.3. Appareils connectés

3.3.1. Gestion des données

3.3.1.1. Fréquence d'envoi

La fréquence d'envoi des données dépend du type d'appareil. Cela dépend de l'importance de l'information. Pour mieux comprendre les choix de fréquence d'envoi, suivre la partie Analyse globale sur les Appareils connectés.

3.3.1.2. Format

En plus du format renseigné dans la partie Appareils connectés s'additionnent des données plusieurs informations complémentaires comprenant l'adresse MAC de l'appareil, pour faciliter le monitoring, et le niveau de batterie, pour remonter le statut d'alimentation de l'appareil.

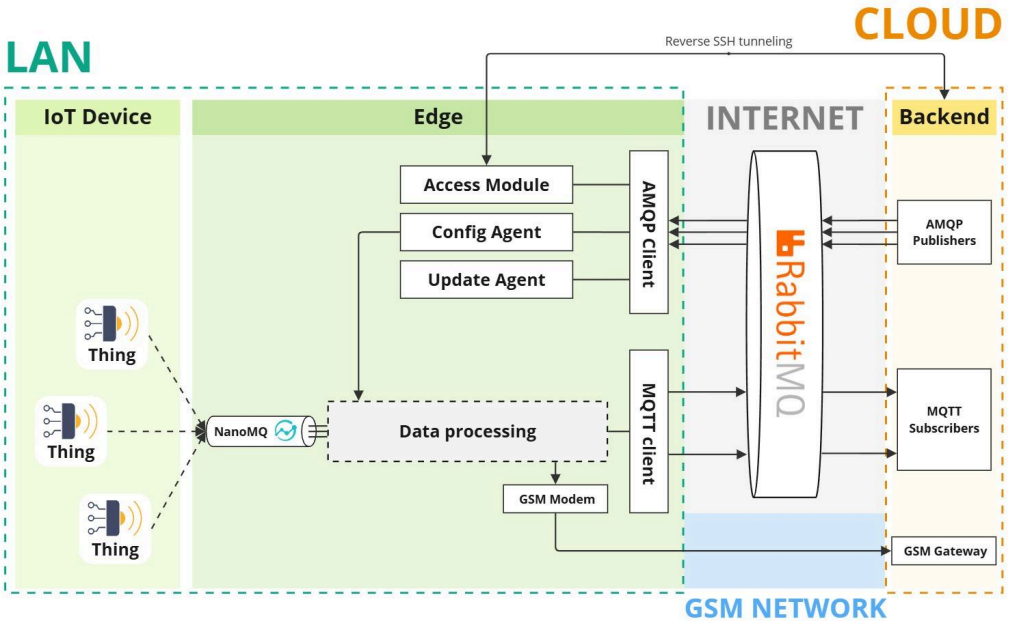
3.3.2. Communications

Les données des appareils sont transmises par le protocole de communication MQTT au Edge. Cela permet de rendre les transmissions de métrique asynchrone et éviter un état bloquant pour l'appareil car certains appareils peuvent envoyer beaucoup de données sur des intervalles de temps très réduit, amenant une surcharge des services du Edge.

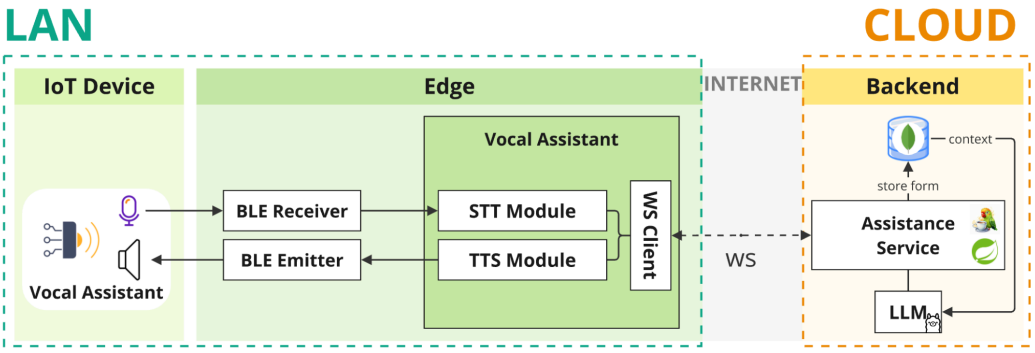
3.4. Gateway - Edge

3.4.1. Structure globale

Le Edge est la passerelle IoT du projet qui permet de collecter, traiter et transmettre les données des capteurs.
Elle se décompose en deux parties dans l'architecture.
La première partie avec le Data Processing :



La deuxième avec l'assistant vocal :

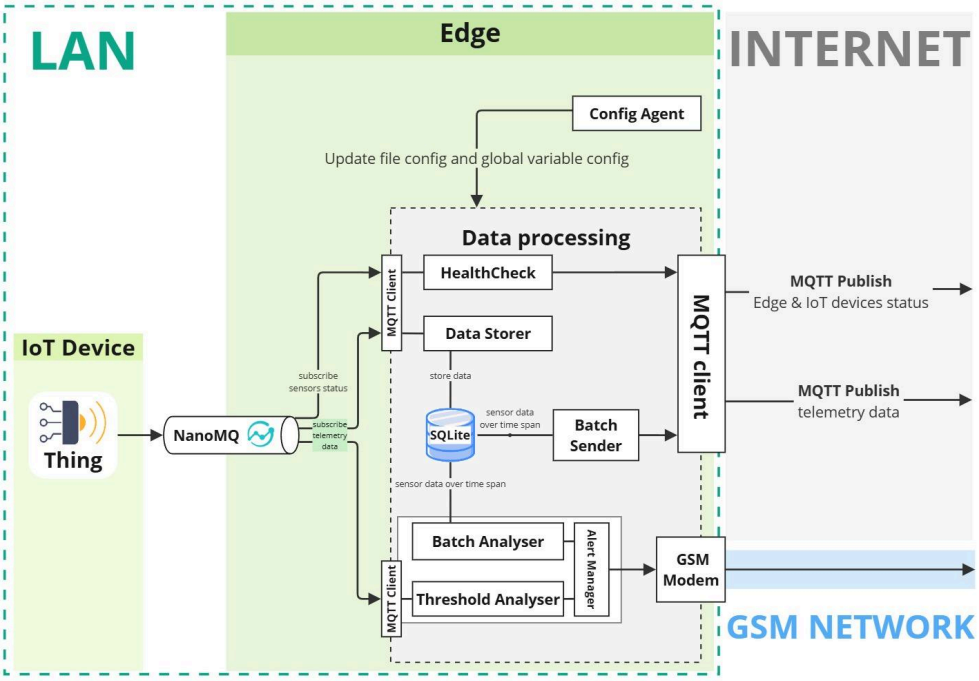


3.4.2. Communications internes

Les composants du Edge communiquent via NanoMQ, un broker MQTT intégré. Cela permet à chaque composant de publier et de s'abonner aux messages selon leurs besoins.

3.4.3. Traitement des données

Les données sont traités dans le Data Processing :



3.4.3.1. Objectif du traitement

Le traitement des données vise à collecter et analyser les données en temps réel. Elles sont ensuite stockées temporairement dans une base de données SQLite pour une conservation locale utile en particulier en cas de perte de connexion avec le backend.

3.4.3.2. Threshold

Ce module de seuil surveille les valeurs critiques de chaque capteur en analysant les données pour détecter des situations nécessitant une alerte. En cas de dépassement de seuil, une alerte est générée et envoyée grâce à l'alerte manager.

3.4.3.3. Détection de pattern

En complément au threshold, le Batch Analyser permet d'analyser les données sur une durée plus longue. Il est utile pour détecter des changements révélateurs d'éventuels problèmes.

3.4.3.4. Surveillance de l'état du système

Le module HealthCheck assure le suivi en temps réel de l'état du backend afin de savoir s'il est opérationnel. Si ce n'est pas le cas, les autres modules du Edge vont s'adapter pour ne pas envoyer de données et donc éviter de les perdre.

3.4.3.5. Gestion locale

Le Data Storer est responsable de la collecte et du stockage temporaire des données générées par les capteurs et les modules du Edge. Il permet avec la base de

données SQLite d'assurer la persistance des données même en cas de coupure avec le backend mais aussi d'optimiser l'envoie des données en les regroupant avec leur transmission.

3.4.3.6. Transmission des données

Le module Batch Sender permet d'optimiser l'utilisation de la bande passante en regroupant les données en paquets avant de les transmettre au backend. Ce regroupement réduit la fréquence des envois, minimisant ainsi la consommation de bande passante.

3.4.3.7. Transcription

Dans le cadre de l'assistance vocale, les données envoyées par le BLE Receiver sont transcrites avec le module STT (Speech-To-Text). En effet, elle transforme les commandes vocales en texte, facilitant l'interaction entre l'utilisateur et le système, avant d'être transmis au Backend via le WebSocket Client.
Les réponses générées par le Backend seront traitées au TTS (Text-To-Speech) via le WS Client. Les données vont à nouveau être transcrites, cette fois, le texte en informations vocales avant d'être transmis au BLE Emitter.

3.4.4. Maintenance

3.4.4.1. Plan de mise à jour

Le système de mise à jour du Edge utilise un modèle de déploiement blue-green avec deux conteneurs distincts (blue et green) pour éviter toute interruption du service lors des mises à jour. La mise à jour est faite dans l'environnement non actif avant d'être basculée en production. Ce modèle permet d'avoir toujours un environnement disponible pour les mises à jour sans interrompre le système tout en assurant une continuité de service et une résilience accrue.

3.4.4.2. Agent de configuration

Le edge contient un fichier de configuration. Il contient les différentes infos des capteurs branchés au Edge (avec le type, le topic, les thresholds..). Lorsque le logiciel est lancé, une variable globale contenant toutes les informations est créée et est disponible pour tous les composants. En cas de changement de paramètres depuis le backend, grâce à un subscribe, l'agent de configuration, Agent Config, met à jour les valeurs de configuration afin d'adapter le comportement du système aux besoins spécifiques de l'environnement.

3.4.4.3. Module de connexion

Le module de connexion assure un accès distant sécurisé via un tunneling SSH inversé. Le reverse SSH tunneling permet de créer un tunnel SSH inversé entre deux machines, où la connexion est initiée depuis la machine distante (client) vers le serveur. Dans ce cas, la machine distante ne peut pas être directement accessible par le serveur en raison de restrictions de réseau (par exemple, derrière un NAT ou un pare-feu).

Ainsi, le backend déclenche l'activation du reverse SSH tunneling en envoyant un message via AMQP. Le composant "Access Module" reçoit ce message et démarre alors le service SSH pour établir la connexion inversée. Une fois cette connexion en place, les administrateurs peuvent se connecter au système distant en passant par le tunnel SSH.

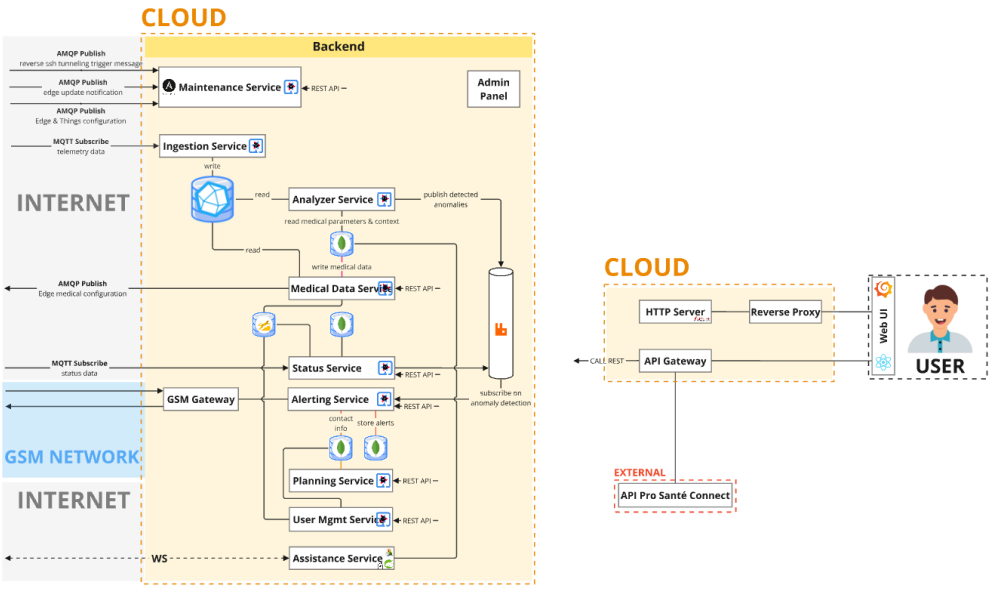
L'accès SSH est évidemment restreint par l'utilisation de clés SSH, garantissant que seules les personnes autorisées peuvent se connecter.

3.4.5. Plan de connexion

La passerelle Edge utilise une combinaison de MQTT et AMQP pour communiquer avec le backend. Le client MQTT publie les données de télémétrie et les alertes générées par le module de traitement des données, tandis que le client AMQP gère les messages de configuration et de mise à jour. Cette architecture permet de gérer efficacement la communication entre les dispositifs IoT, la passerelle, et le backend, assurant une synchronisation et une réactivité en temps réel.

3.5. Backend

3.5.1. Structure globale



Le système se base sur les principes de micro-services. Cela permet de réduire les responsabilités des services en une unique tâche métier ou fonction spécifique. Cette structure permet de répondre à plusieurs aspects : mise à l'échelle, résilience et flexibilité.

3.5.2. Communications internes entre les services

La synchronicité des communications dépend de la connexité entre les services. Pour les services à liaisons synchronisées, cela s'explique par des requêtes requérant une réponse.
Pour les liaisons asynchrones, le but est de garder l'aspect *Quality of Service* pour ne pas bloquer les services s'il n'y a pas de nécessité et éviter des états bloquants inutiles.

3.5.3. Description des services

3.5.3.1. Analyzer Service

Le "*Analyzer Service*" permet de faire des analyses sur les données des patients sur de plus grands échantillons. Ce niveau d'analyse est présent en complément de la première couche sur la passerelle, pour émettre des alertes concernant des anomalies relevables seulement sur le long terme. Pour pouvoir remonter ces alertes, le service peut informer le service dédié "*Alerting Service*".

De plus, l'analyse s'étend aussi à celle des statuts des appareils, pour provenir des alertes sur leurs alimentations. Une liaison entre le *Status Service* et le *Analyzer Service* permet d'activer ou désactiver le suivi des appareils si l'un d'eux est éteint manuellement.

3.5.3.2. Alerting Service

Ce service se couple avec le GSM Gateway. Il est dédié aux traitements des alertes. Il permet de déterminer le niveau d'alerte, de récupérer les informations de contact des proches et de les envoyer.

3.5.3.3. Assistance Service

Le système d'assistance est représenté par le service *Assistance Service*, il gère toutes les demandes en provenance du patient, qui se caractérise par l'assistant vocal, et les enquêtes quotidiennes assisté par l'intelligence artificielle. Il permet de récupérer le compte rendu des patients sur leur ressenti et leur santé. Ce service dispose les données à une base commune avec l'*Analyzer Service* qui s'occupe ensuite de traiter les données pour estimer son état de santé mental. Ces données sont aussi

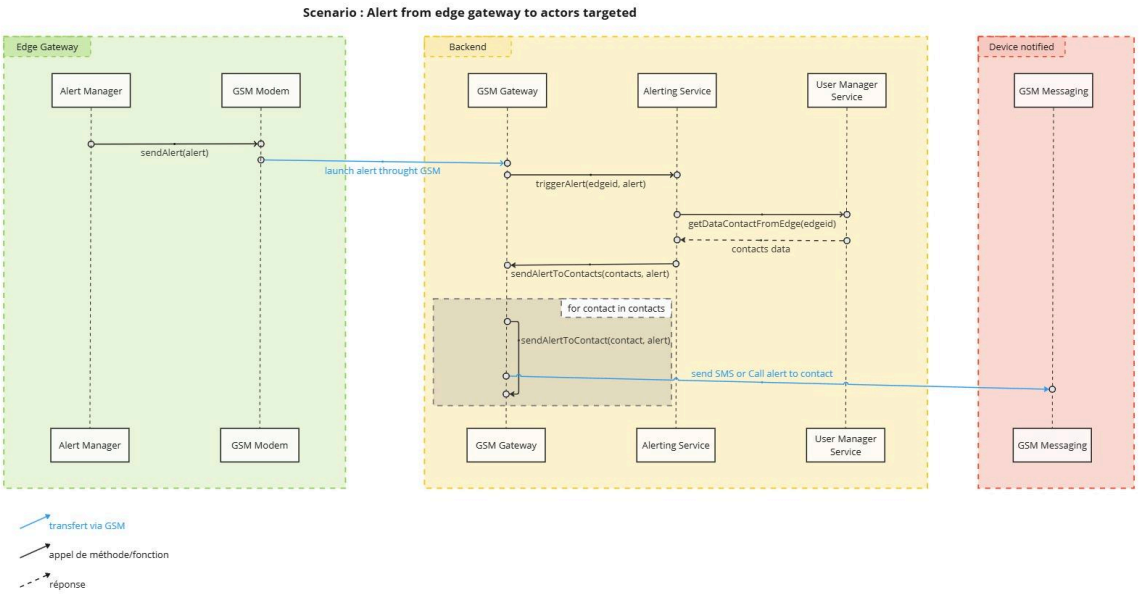
3.5.3.4. Control Access Service

Ce service gère les accès aux données et aux services selon les privilèges utilisateurs et aux besoins (utilisateurs et services). Plus précisément, il sert d'interface entre certains services et de redirection vers le portail de connexion Pro Santé Connect, pour identifier le personnel de santé.

3.5.3.5. GSM Gateway

La *GSM Gateway* ou passerelle GSM se dédie à l'envoi et la réception d'alertes. Le service se base sur la technologie

Dans un premier temps, le service récupère les alertes détectées et levées par les edge gateway, avant de notifier les différents acteurs associés. Ci-dessous un schéma pour expliquer la démarche de lancement des alertes provenant de la edge gateway.



Il permet donc à la fois de recevoir et d'émettre des messages en utilisant le GSM. En utilisant le dispositif GSM, on garantit aussi la réception des alertes provenant des Edge Gateway, même en cas de coupure de connexion internet.

3.5.3.6. Ingestion Service

Le service d'ingestion de données (ou *Ingestion Service*) a pour but d'ingérer les données provenant des edges gateway pour les rendre disponible pour les traitements nécessaires. Il vient ajouter les données à la base de données commune (avec l'*Analyzer Service* et le *Medical Data Service*).

3.5.3.6.1. Récupération des données

Le service d'ingestion est souscrit (MQTT) aux événements de montée de données provenant des Edges Gateway.

L'utilisation de d'une architecture "event driven" permet de servir un traitement asynchrone des publications de données et réduit le couplage entre le Backend et les Edges Gateway.

3.5.3.6.2. Analyse de scalabilité

L'*Ingestion Service* est essentielle à notre système en ajoutant les nouvelles données montantes en provenance des patients. C'est aussi l'un des services les plus sollicités du système, et qui sera ciblé par la scalabilité.

En effet, en partant du principe que le flux de données entrantes pour chaque patient est continu (montée des données régularisée), lorsqu'on ajoute un nouveau patient - et donc un nouveau edge gateway et de nouveaux appareils, cela ajoute une charge constante supplémentaire à notre service d'ingestion. C'est pourquoi il est important de penser sa scalabilité, bien que sa charge soit proportionnelle aux nombres de passerelle.

3.5.3.7. Maintenance Service

Le service de maintenance (ou *Maintenance Service*) est accessible depuis le panneau d'administration. Il permet de déclencher et de contrôler diverses opérations de maintenance telles que la mise à jour des Edge Gateways, la gestion des configurations, et l'établissement de connexions SSH sécurisées.
Les mises à jour des Edges Gateways sont possibles grâce à plusieurs points :

- Le Maintenance Service orchestre l'exécution des playbooks Ansible pour assurer le déploiement automatique des mises à jour sur les Edge Gateways.
- Les images Docker contenant les nouvelles versions sont stockées dans Artifactory, un système de gestion d'artefacts, où elles sont versionnées et accessibles pour les Edge Gateways.
- En publiant sur le topic de mise à jour via AMQP, le Maintenance Service informe l'Agent de Configuration (Agent Config) sur les Edge Gateways de l'arrivée de nouvelles versions. Celui-ci peut alors télécharger et appliquer les mises à jour des images et de leurs dépendances

Les dispositifs sont configurés comme suit : le Maintenance Service gère les changements de configuration en publiant des messages sur le topic de configuration. Cela permet aux Edge Gateways de recevoir et d'appliquer les dernières modifications de configuration automatiquement. Ce processus assure une synchronisation continue entre les configurations du backend et celles des dispositifs, évitant des écarts qui pourraient nuire au bon fonctionnement des Edge Gateways.
C'est également le Maintenance Service qui permet la gestion des connexions SSH comme expliqué dans la Maintenance de la Gateway. De plus, Le Maintenance Service utilise Ansible pour automatiser les déploiements et les configurations via SSH. Cependant, le broker AMQP sert de médiateur pour publier les messages de mise à jour, de configuration et de tunneling SSH, assurant une communication fiable et asynchrone entre le backend et les Edge Gateways.
Enfin, Le Maintenance Service inclut également un système de notifications qui informe les administrateurs de l'état des mises à jour, des changements de configuration, et des connexions SSH. Ce suivi permet de garder un historique des actions réalisées et de réagir rapidement en cas de problème

3.5.3.8. Medical Data Service

Il dessert les informations médicales demandées par l'utilisateur et permet la manipulation des paramètres médicaux concernant les patients. Le service Medical Data a la responsabilité de mettre à jour les seuils d'alertes pour les services sur Cloud et sur le Edge du patient.

3.5.3.9. Planning Service

Il permet au médecin et aux proches du patient de préparer les visites entre le patient et les différents autres acteurs (techniciens pour l'installation de nouveaux appareils, infirmiers pour un suivi continu du patient).

Pour tous les acteurs, il permet de connaître les visites qui leur sont attribuées et pour le patient et ses proches, de connaître les prochaines visites programmées.

3.5.3.10. User Management Service

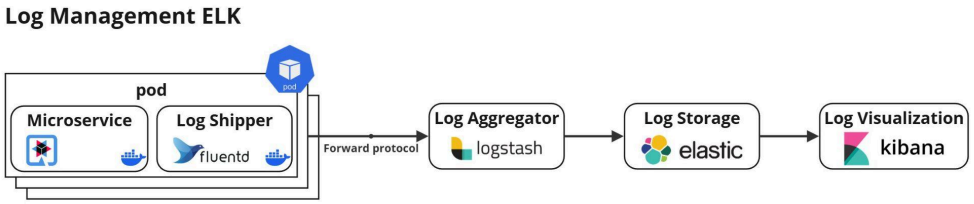
Le *User Management Service* a pour objectif de permettre la création et le management des utilisateurs. Toutes les opérations portant sur ce service sont CRUD, et la création de compte de personnel médical se fait avec la validation des autorités via le service externe Pro Sante Connect.

3.5.3.11. Status Service

Il récupère les données concernant les états des appareils, pouvant comprendre un état courant, le niveau de batterie… Les données concernent les appareils du réseau Mesh chez le patient, et des Edges Gateway.

3.5.3.12. ELK - Log Service

Le service dédié au Log a pour but de rassembler les événements de tous les services pour les centraliser et les rendre plus facile d'accès par ce service.



Le service ELK (Elasticsearch, Logstash, Kibana) est essentiel pour la gestion des logs car il permet de centraliser, stocker et analyser les évènements provenant de tous les microservices du backend. Cette gestion facilite le suivi des opérations effectuées et permet de détecter rapidement les anomalies. Le service se décompose en plusieurs éléments :

- Log Shipper (Fluentd) : intégré au pod du microservice sous la forme d'un side-car container qui collecte les logs et les envoie vers un agrégateur
- Log Aggregator (Logstash): qui reçoit les logs de Fluentd et les transforme si nécessaire avant de les envoyer dans la base de données Elasticsearch
- Log Storage (Elasticsearch) : qui stocke les logs de manière organisée et permet de faire des recherches rapides
- Log Visualization (Kibana) : utilisé pour visualiser les données stockées dans Elasticsearch

ELK permet de gérer et stocker de grands volumes de données , ce qui est particulièrement utile dans le cas des logs puisque cette centralisation laisse la possibilité de visualiser les logs des différents services par les équipes de développement et de maintenance, ici par l'administrateur.

3.5.4. Résilience et mise à l'échelle

3.5.4.1. Mise à l'échelle

En divisant les responsabilités dans des micro-services de traitement et de récupération de données, cela permet d'augmenter les ressources nécessaires (mise à l'échelle horizontale ou verticale). Par exemple, le service d'ingestion des données nécessitera potentiellement une mise à l'échelle (augmentation du nombre de réplicats) plus importante que d'autres services (*Maintenance Service*, par exemple).

Cette mise à l'échelle est automatisée par Kubernetes. En mettant en place un HPA (Horizontal Pod Autoscaler) sur le *Ingestion Service*, cela permet dans un premier temps, de mettre à l'échelle automatiquement le service pour répondre aux besoins des entrées des données. Dans un second temps, la configuration de l'HPA définit les règles de scaling, comme le nombre minimal et maximal de réplicats ou les métriques de charge d'utilisation des réplicats (en définissant une moyenne de charge CPU par exemple).

Tout d'abord, le flux de données entrant, provenant de l'ensemble des Edges Gateway, reste régulier et constant. En effet, chaque passerelle fait remonter les données régulièrement,

3.5.4.2. Résilience

Pour une question de disponibilité des services, ils doivent pouvoir continuer de fonctionner même en cas de défaillance d'une partie des micro-services. Cela permet d'avoir une grande tolérance aux pannes.

La résilience est traitée par Kubernetes. Ici, le Load Balancer permet de rediriger les flux vers un ou des réplicats fonctionnels pour subvenir à la charge et aux services en panne. Il redirige les flux vers les pods (ou réplicats) restants pendant le retour du nouveau pods.

Cette pratique oblige de disposer d'au minimum de deux pods pour chaque service, même si un seul pourrait largement répondre à la charge. Ce système pour prévenir de la résilience amène l'utilisation inutile de ressources.

3.5.5. Traitement des données

En général, le traitement de données dans les services est CRUD. Mais pour détecter des anomalies dans les données et envoyer des alertes, il est essentiel d'en faire un traitement plus ou moins complexe selon le type de données. Cela concerne tout aussi bien des données importantes (fréquence cardiaque avec la détection d'anomalies cardiovasculaires, montée de tension) comme simplement d'environnement (température du foyer). Ces traitements sont réalisés dans le micro service dédié, l'*Analyser Service*, qui lance des alertes avec un niveau de gravité.

3.5.6. Anonymisation

L'anonymisation des données est essentielle pour des données sensibles. Une structure d'encryption et de mapping a été mise en place, pour protéger les données de l'extérieur (ex: fuite de données).

L'encryption miroir permet une anonymisation rigoureuse des données associées au patient. Il fait une association sécurisée entre les identifiants du client à celle de son edge. Le mécanisme mit en place est un chiffrement bidirectionnel sur chaque identifiant, qui permet de créer un lien chiffré dans les deux sens : *E(PatientId) - EdgeId* et *E(EdgeId) - PatientId*.

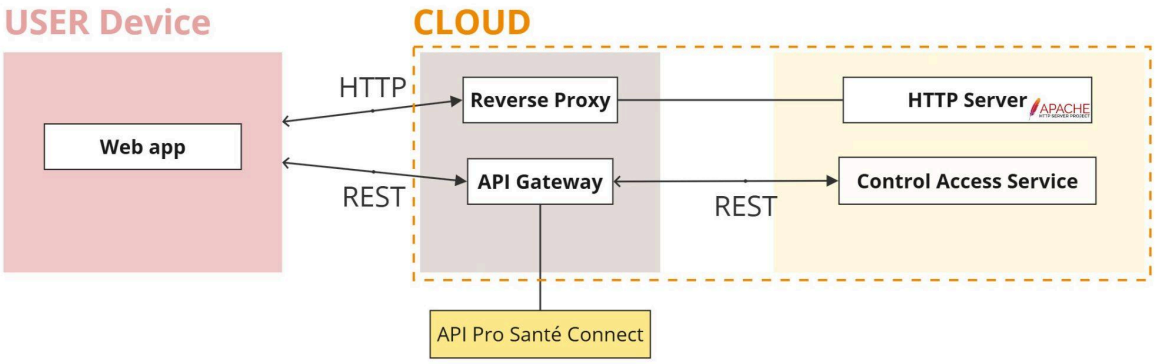
Contrairement aux solutions en mémoire (In-Memory Database) pouvant exposer des clés de chiffrement à des risques d'accès non autorisé, le modèle ne conserve pas les données en mémoire vive, ce qui minimise les points de vulnérabilité.

Cette encryption miroir permet aussi de se conformer aux exigences de confidentialité et de sécurité, définies par les réglementations sur les données de santé.

3.6. Visualisation - Serveur web

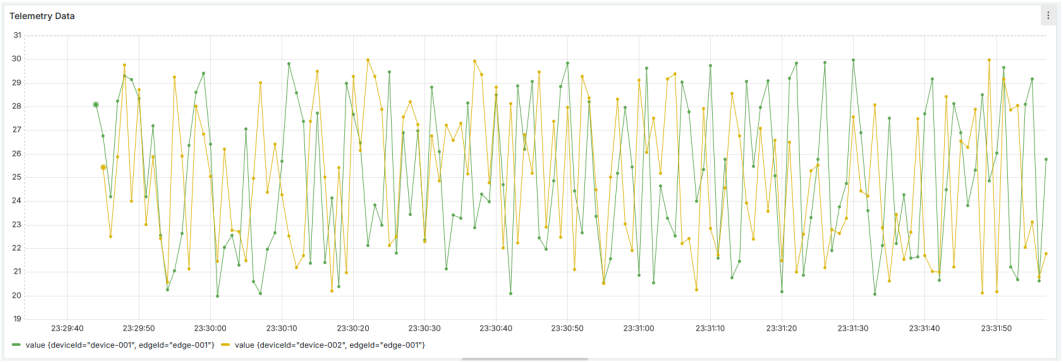
3.6.1. Structure globale

L'entièreté des services liés ou utiles à l'application Web est hébergée sur le Cloud. Seul le portail de connexion Pro Santé Connect est externe au système et permet l'authentification du personnel médical. Ici, on divise, grâce à des principes de micro frontend, le service web des données, en fonction des accès de l'utilisateur.



3.6.2. Visualisation des données

La visualisation des données est proposée par les outils (et plugins) de Grafana. En intégrant les outils de visualisation à l'application Web app pour permettre aux différents utilisateurs de gérer leur propres espaces, selon leurs accès aux données (limité, réduit, entier).
Voici un exemple de plugin de visualisation graphique de données temporelles de température ambiante, avec une oscillation aléatoire.



3.6.3. Accès

Les accès sont géré directement par le micro service du Backend Control Access Service, qui donne les informations à afficher en fonction du niveau d'accès (cf. Analyse de la Visualisation).

3.7. Plan de tests

Afin de garantir la qualité, la fiabilité, et la robustesse de l'ensemble du système, composé de la Gateway et des microservices backend, un plan de tests rigoureux est mis en place. Ce plan est structuré autour de trois types de tests principaux : les tests unitaires, les tests d'intégration, et les tests de bout en bout (End-to-End ou E2E).

3.7.1. Tests unitaires

Les tests unitaires visent à vérifier le bon fonctionnement de chaque composant isolé du système. Ils sont principalement effectués sur les fonctions et modules internes des microservices du backend ainsi que sur les modules individuels de la Gateway. Les tests unitaires ont pour objectif de s'assurer que chaque unité de code (les fonctions, les méthodes ou les modules) produit le résultat attendu.

	Gateway	**Backend**
Scope	Tester les fonctions et modules critiques de la gateway, en se concentrant sur la logique de traitement de données, les protocoles de communication, et la gestion des erreurs	Tester chaque microservice individuellement, en vérifiant le comportement de chaque fonction et méthode. Cela inclut les contrôleurs, les services métiers, et les interactions de base de données
Outils utilisés :	Utilisation de frameworks comme Google Test et Catch2 pour l'exécution de tests unitaires sur le code C++	Pour Quarkus et Spring Boot : JUnit : Principal framework de tests pour Java, recommandé pour tester les microservices. Mockito : Pour les tests avec mock, particulièrement utile pour isoler les dépendances entre composants. Pour Quarkus uniquement : Quarkus JUnit Extension : Pour les services en Quarkus, qui permet de tester les services Quarkus avec des fonctionnalités spécifiques au framework.

3.7.2. Tests d'intégration

Les tests d'intégration permettent de vérifier le bon fonctionnement des interactions entre différents composants du système, sans nécessairement couvrir l'ensemble du système complet. L'objectif est de tester les interfaces entre les microservices ainsi que les communications entre la gateway et le backend pour s'assurer qu'ils échangent correctement les données.

Le scope de ces tests a été définis selon deux grands axes :

- Les interactions entre MicroServices : les appels API entre les services (Quarkus et Spring Boot) sont testés pour garantir la cohérence des échanges de données et le respect des contrats d'API (format de données, types de réponses, gestion des erreurs etc.)
- La communication Gateway-Backend : pour vérifier qu'il est effectivement possible de communiquer entre les deux et correctement. Cette vérification inclut également la vérification des protocoles de communication (WebSocket, AMQP, MQTT, HTTP etc.) et le traitement des réponses et des erreurs.

Les outils utilisés dans le cadre des tests d'intégration sont :

- L'utilisation de Testcontainers pour simuler l'environnement avec des conteneurs Docker, permettant de tester les microservices en interaction avec des bases de données, des files d'attente, etc.
- L'utilisation de Quarkus Dev Services pour les microservices en Quarkus, cette fonctionnalité permet de tester facilement les interactions avec les bases de données et autres services externes.
- La création et l'utilisation de script personnalisé pour tester manuellement la communication entre la gateway et les services backend

3.7.3. Tests E2E

Les tests E2E visent à valider le système complet en simulant des scénarios réels d'utilisation, de bout en bout. Ils permettent de vérifier le flux complet des données et l'interaction de tous les composants du système :

- Vérifier que la gateway collecte correctement les données des dispositifs IoT.
- Confirmer que la gateway transmet bien ces données aux services backend.
- Assurer que les services backend traitent et stockent correctement les données dans les bases de données.
- Valider que les données stockées sont disponibles pour consultation et que les réponses du backend respectent les attentes du système.

Le scope des tests E2E retranscrit les étapes du système de la manière suivante :

1. La collecte des données depuis les Things afin de tester le processus de récupération des données brutes depuis les capteurs ou dispositifs IoT, y compris la gestion des erreurs en cas de données corrompues ou non conformes
2. La transmission et la transformation des données pour vérifier que la gateway envoie bien les données vers les services backend en respectant le format requis, et que le backend est capable de recevoir et d'interpréter ces informations
3. Le stockage et la persistance des données en testant l'insertion, la mise à jour et la suppression des données dans les bases de données utilisées par le backend et la gateway pour garantir que les données sont bien enregistrées et que les bases de données sont correctement mises à jour
4. La consultation des données afin de vérifier que les données dans le backend sont accessibles et consultables par les clients et les interfaces

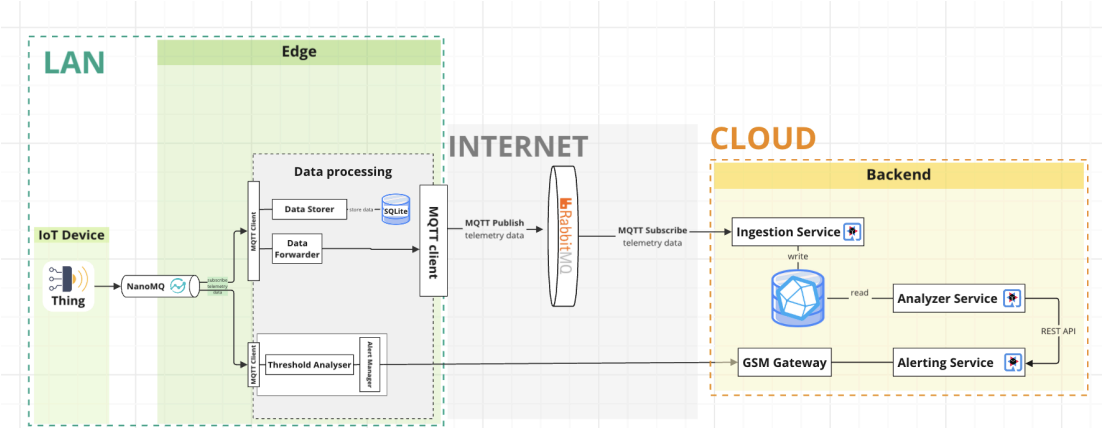
Pour couvrir ce scope, plusieurs outils vont être utilisés :

- L'utilisation de scripts personnalisés à nouveau pour automatiser des scénarios de tests complexes pour simuler la collecte de données de la gateway et les transmettre au backend pour vérifier la transmission mais aussi vérifier que les données sont correctement stockées dans les bases de données.

- La simulation de dispositifs IoT pour émuler les capteurs afin de permettre de tester le flux complet sans dépendre des dispositifs physique et d'avoir la possibilité de créer divers scénarios de collecte de données avec des valeurs variées avec ou sans erreurs
- L'utilisation d'outils de bases de données (MongoDB Compass, pgAdmin, DBeaver etc.) afin de valider que les données sont correctement enregistrées dans les bases de données
- L'automatisation des tests grâce à des Framework de tests E2E (Cypress, Playwright) pour les interfaces utilisateurs

4. Proof of concept

4.1. Équipements de télémétrie



Pour notre POC, nous utilisons Node-RED pour simuler différents capteurs. Ce POC inclut quatre types de capteurs :

- Température du logement
- Température du patient
- Fréquence cardiaque
- Glycémie

Les capteurs connaissent déjà l'IP et le port du broker NanoMQ, ce qui signifie qu'il n'y a pas besoin de service de découverte. Ils publient leurs données de télémétrie directement sur le broker NanoMQ.

4.2. Envoi des données de télémétrie

Contrairement à l'architecture théorique, le edge du POC envoie les données de télémétrie directement au backend, au lieu de les regrouper en batchs. Pour cela, un composant du edge nommé "Data Forwarder" s'abonne aux messages de télémétrie et les transfère au backend via un client MQTT, qui est connecté au broker RabbitMQ. Le service d'ingestion du backend s'abonne aux données de télémétrie sur RabbitMQ et stocke ces données dans la

base de données InfluxDB. Au lieu de fournir une interface client dédiée, nous avons décidé d'afficher les données directement sur Grafana pour ce POC.

4.3. Gestion des alertes

Nous avons également un composant du edge nommé "Threshold Analyzer" qui s'abonne aux messages de télémétrie des capteurs. Il analyse les valeurs pour détecter si elles dépassent des seuils min/max prédéfinis. En cas de dépassement, une alerte est envoyée au backend via un service REST (plutôt que par le réseau GSM, comme nous l'avions envisagé dans notre solution théorique). Différents niveaux d'alerte sont définis : MINOR, SERIOUS, CRITICAL et ENVIRONMENTAL, chaque niveau ayant ses propres seuils pour chaque type de capteur. Lorsqu'une alerte est détectée, le service d'alerte du backend envoie un SMS via l'API de FREE.

Pour des analyses plus approfondies côté backend, un composant appelé "Analyser Service" effectue périodiquement des analyses des données de télémétrie stockées dans le backend. À chaque cycle défini, ce service récupère les données sur une période de temps spécifique, en calcule la moyenne, puis applique une analyse basée sur les seuils de cette moyenne. Si cette analyse révèle une anomalie dépassant les seuils établis, le processus d'envoi d'alerte mentionné précédemment s'active de la même manière. Ce mécanisme permet de compléter les alertes instantanées en détectant des tendances anormales sur la durée.

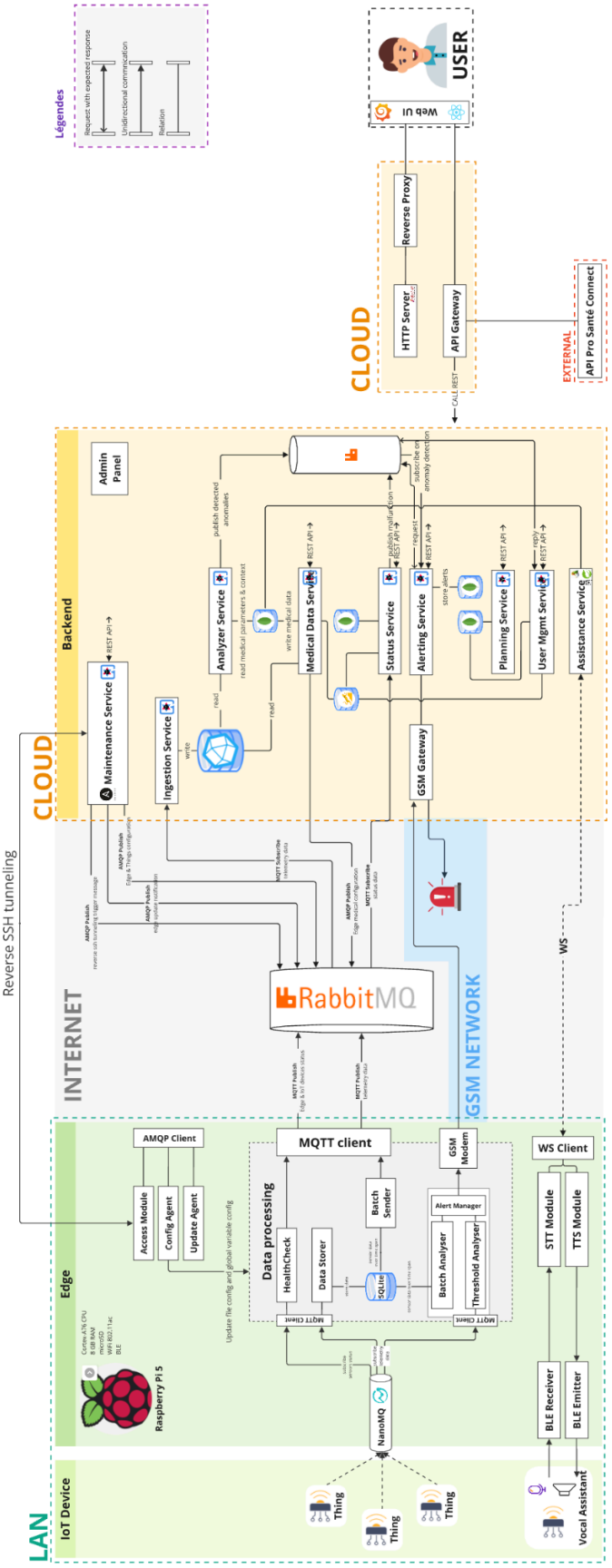
4.4. Des bribes d'implémentation future

Enfin, nous avons dans le edge un composant "Data Storer" qui s'abonne aussi aux messages de télémétrie pour stocker les données dans une base SQLite. Cependant, ces données ne seront pas exploitées dans ce POC. Ce composant aurait eu toute son utilité si nous avions implémenté un envoi des données en batch à intervalles réguliers au lieu de les transférer immédiatement au backend. Il aurait également permis des analyses plus complexes que la simple vérification des seuils.

5. Conclusion

L'architecture mise en place pour répondre aux différents besoins étudiés répond à des problématiques de mise à l'échelle de logiciel, par la gestion de charge. Elle se veut modulable et évolutive, pour l'ajout de nouvelles fonctionnalités futures. Elle est réfléchie pour gérer son propre écosystème, veiller à la sécurité des accès et des données, mettre à jour en continu les appareils distants et proposer des services sans coupure.
Mais par-dessus tout, elle cherche à répondre aux différents besoins utilisateurs, que ce soit pour la visualisation des données, l'envoi d'alertes, la gestion des plannings de visite, des patients et des accès, la maintenance des systèmes distants et veiller à la qualité de vie des patients.

6. Annexes


Légendes
Request with expected response
Unidirectional communication
Relation
USER
Web UI
CLOUD
Reverse Proxy
HTTP Server
API Gateway
EXTERNAL
API Pro Santé Connect
CALL REST
Backend
Admin Panel
Maintenance Service
Ingestion Service
Analyzer Service
Medical Data Service
Status Service
Alerting Service
GSM Gateway
Planning Service
User Mgmt Service
Assistance Service
REST API
publish detected anomalies
read medical parameters & context
write medical data
read
write
store alerts
Reverse SSH tunneling
INTERNET
RabbitMQ
GSM NETWORK
WS
Edge
AMQP Client
MQTT client
GSM Modem
WS Client
Access Module
Config Agent
Update Agent
Data processing
HealthCheck
Data Storer
Batch Sender
Alert Manager
Batch Analyser
Threshold Analyser
SQLite
STT Module
TTS Module
Raspberry Pi 5
NanoMQ
BLE Receiver
BLE Emitter
LAN
IoT Device
Thing
Vocal Assistant