Document de conception d'architecture logicielle

ANTONIOS Camille · EL GHOLABZOURI Hajar · KAHUNGU Jessica · MARTIN Gauthier · MULLER Amélie

Sommaire

Introduction

Ce projet s'inscrit dans le contexte de l'e-santé et du maintien à domicile des personnes âgées. Il a pour ambition d'assurer un suivi continu de la santé des patients, de détecter rapidement les incidents éventuels, de transmettre les informations pertinentes au médecin et de faciliter la préparation des visites des infirmières. Deux objectifs principaux guident cette initiative :

- **Le maintien à domicile**, rendu possible grâce à un système d'alertes permettant de laisser le patient seul sans compromettre sa sécurité ;
- **Le suivi médical**, appuyé sur la collecte automatique de données, réduisant ainsi le nombre de mesures que l'infirmière doit effectuer lors de ses visites.

Le projet se concentre exclusivement sur le monitoring au sein du domicile et s'adresse en priorité aux médecins, qui peuvent prescrire ce dispositif médical à leurs patients.

L'entreprise propose un service de suivi de patients aux médecins. Ce service sera utile pour le médecin car il permettra un suivi plus facile et plus précis de ses patients, par l'intermédiaire d'un tableau de bord, récapitulant les données médicales relevées pour un patient donné. Le médecin pourra aussi lire les résultats d'un formulaire dont il aura paramétré les questions pour un suivi spécifique au patient.

Le médecin peut prescrire le dispositif médical à ses patients âgés dont l'état de santé se dégrade. Le "client" principal de notre entreprise est donc la personne âgée : c'est elle qui paiera pour le service.

Le système se présente au client sous la forme d'une application web (et mobile), pour pouvoir répondre aux questions du formulaire, ainsi que de deux dispositifs physiques. Le premier est un bracelet, que la personne âgée doit porter sur elle lorsqu'elle se trouve dans son domicile, qui relèvera des données médicales. Le deuxième est un boitier, auquel le bracelet se connectera, et qui servira de "Gateway" pour remonter les données médicales jusqu'à la base de données.

Pour le médecin et l'infirmière le système se présente sous la forme d'une application web (et mobile). L'interface du médecin présente un suivi détaillé des données médicales du patient, tandis que celle de l'infirmière ne présente qu'un récapitulatif d'une courte période qu'elle aura défini, afin de l'aider dans sa visite régulière.

1 Analyse des besoins

1.1 Hypothèses

- La surveillance des patients est réalisée uniquement à l'intérieur de leur domicile.
- Les patients répondent aux caractéristiques suivantes :
    - Ils passent la majeure partie de leur temps chez eux
    - Ils vivent seuls et sont les seuls individus monitorés dans leur logement
    - Ils ont des compétences limitées en informatique
    - Leur état de santé est fragile et en déclin, mais ils restent capables d'accomplir certaines actions de manière autonome
    - Ils reçoivent la visite d'une infirmière régulièrement, soit quotidiennement, soit tous les deux jours
    - Ils disposent d'une connexion Internet fonctionnelle, et ont un équipement informatique (mobile ou ordinateur)
- Le système est dimensionné pour gérer jusqu'à 150 000 utilisateurs, répartis comme suit :
    - 100 000 patients
    - 20 000 médecins
    - 30 000 infirmières

1.2 User stories

Liste de nos rôles :

- **Patient** : la personne âgée faisant l'objet du monitoring par le système
- **Médecin** : le médecin ayant prescrit ou demandé le suivi du patient
- **Personnel infirmier** : professionnel de santé responsable du suivi quotidien et des visites auprès du patient.
- **Proche** : membre de la famille ou personne de confiance souhaitant être informé en cas d'alerte critique concernant le patient
- **Administrateur** : responsable de la gestion, du déploiement et de la maintenance du système

Patient :

- **En tant que** patient, je souhaite pouvoir soumettre une demande d'inscription sur le site, **afin que** les administrateurs puissent créer mon espace personnel.

- **En tant que** patient, je souhaite indiquer les coordonnées de mon médecin et de mon infirmière lors de la demande d'inscription, **afin que** les administrateurs puissent établir le lien entre nos comptes.
- **En tant que** patient, je souhaite remplir un formulaire quotidien, **afin de** communiquer l'évolution de mon état de santé au personnel soignant.
- **En tant que** patient, je souhaite pouvoir contacter le support technique en cas de problème, **afin de** recevoir une assistance rapide et adaptée.
- **En tant que** patient, je souhaite vérifier le bon fonctionnement de mon dispositif et de la connexion avec le système, **afin de** m'assurer que mon suivi est toujours actif.
- **En tant que** patient, je souhaite pouvoir retirer et remettre mon bracelet sans que cela perturbe le suivi de ma santé.
- **En tant que** patient, je souhaite pouvoir recharger mon bracelet, **afin de** garantir son bon fonctionnement en continu.

Médecin :

- **En tant que** médecin, je souhaite consulter les données biologiques détaillées d'un patient, **afin de** suivre l'évolution de leur état de santé.
- **En tant que** médecin, je souhaite sélectionner des questions à poser dans un formulaire rempli quotidiennement par le patient, **afin de** m'adapter aux besoins spécifiques de chaque patient.
- **En tant que** médecin, je souhaite accéder facilement aux réponses des formulaires quotidiens, **afin d**'adapter le traitement ou le suivi médical en conséquence.
- **En tant que** médecin, je souhaite recevoir automatiquement un rapport de synthèse pour chaque patient, à une fréquence que je détermine, **afin de** réduire la nécessité de surveiller le profil de chaque patient.

Personnel infirmier :

- **En tant que** personnel infirmier, je souhaite visualiser un bilan simplifié des données, **afin de** détecter rapidement toute anomalie dans l'état de santé du patient et de ne pas avoir à les mesurer manuellement.
- **En tant que** personnel infirmier, je souhaite être notifié en cas d'anomalies sévères dans les indicateurs biologiques du patient, **afin de** pouvoir agir en conséquence.

Proche :

- **En tant que** proche, je souhaite être alerté en cas de problème de santé critique, **afin d**'être informé rapidement et pouvoir réagir en conséquence.

Administrateur :

- **En tant qu**'administrateur, je souhaite créer les comptes des patients, personnel infirmier et médecins, **afin de** gérer leurs droits d'accès au système.
- **En tant qu**'administrateur, je souhaite lier les différents utilisateurs entre eux (patient – médecin – personnel infirmier), **afin d**'assurer un suivi coordonné et personnalisé.
- **En tant qu**'administrateur, je souhaite pouvoir modifier à tout moment les informations associées à chaque compte, **afin de** maintenir la cohérence du système en cas de changement de coordonnées.
- **En tant qu**'administrateur, je souhaite consulter et mettre à jour les identifiants des équipements associés aux patients, **afin d**'assurer la gestion, la maintenance et le suivi du matériel déployé.
- **En tant qu**'administrateur, je souhaite déployer des mises à jour logicielles sur les équipements, **afin de** garantir leur bon fonctionnement et la sécurité du système.

2. Solution proposée

Comme expliqué précédemment, l'entreprise propose aux médecins une plateforme de suivi médical permettant un suivi simplifié et précis des patients grâce à un tableau de bord centralisant les données collectées par les dispositifs connectés.

2.1 Prescription et mise en place

Au moment où le médecin juge l'utilisation du système nécessaire, il le prescrit au patient. Le patient (aidé si besoin de l'un de ses proches) doit alors se rendre sur le site de l'entreprise pour effectuer une demande d'abonnement, en renseignant ses informations personnelles ainsi que celles de son médecin et de son infirmière, des numéros de contact de proches et enfin, la prescription du médecin.

Notre équipe reçoit alors la demande et :

- Si le médecin et l'infirmière possèdent déjà un compte, on crée simplement le compte du patient
- Sinon, on crée les comptes manquants (du médecin et/ou de l'infirmière) et on leur envoie un mail pour finaliser leur compte

On effectue ensuite les liaisons médecin ↔ patient ↔ infirmière avant de contacter le patient pour prévoir l'installation du système à domicile. L'entreprise fournit au patient tous les équipements nécessaires au bon fonctionnement du système :

- Le bracelet connecté, qui effectuera les mesures nécessaires pour le suivi médical, ainsi que la détection de chutes, de crises cardiaques, et d'insuffisances respiratoires ;
- Et le boitier, qui servira d'intermédiaire entre le bracelet et nos serveurs de gestion des données, ainsi qu'à prévenir les personnes concernées en cas d'incident grave (chute, crise cardiaque, ...) par SMS via sa carte SIM intégrée.

Lors de l'installation, un technicien se chargera de faire l'appairage entre le bracelet et \le boitier, ainsi que d'expliquer au patient comment le système fonctionne, comment s'en servir, et de lui présenter l'application pour répondre aux formulaires.

2.2 Utilisation courante

Une fois le système mis en place, le médecin peut configurer différents paramètres du système pour un patient donné. Il a la possibilité de sélectionner les questions et la fréquence d'envoi du formulaire adressé au patient. Il peut puiser dans une banque de questions préétablie, constituée lors de la création du système à partir de questions génériques, ou y ajouter ses propres questions afin de l'enrichir progressivement. Les réponses du patient se limitent à trois options : positive, neutre ou négative.

Le but du formulaire est de fournir un moyen au médecin de suivre de façon personnalisée son patient (ex. suivi de la douleur pour ajuster les antalgiques, appétit, niveau de fatigue, etc). Le médecin peut aussi choisir la fréquence de génération des rapports, pour avoir un suivi adapté aux différents besoins de chaque patient.

L'infirmière, elle, peut simplement configurer la durée de récapitulatif des données, pour l'aligner avec son rythme de visites (dernières 24, 48h par exemple). Elle pourra ensuite pendant chaque visite consulter les données des dernières heures pour faciliter sa visite.

Au quotidien, le patient, lui, n'interagit que très peu avec le système. En effet, le bracelet effectue automatiquement les différentes mesures de santé et les communiques au

boitier, sans besoin d'intervention du patient. Ainsi, à part le formulaire auquel il doit répondre selon la fréquence fixée par le médecin via l'application, le patient doit simplement recharger son bracelet lorsque celui-ci est proche de ne plus avoir de batterie, indiqué par l'écran du bracelet.

2.3 Incident grave

En cas d'incident grave, comme une chute, ou une crise cardiaque par exemple, le bracelet relaie directement l'information au boitier, qui aussitôt envoie un message d'urgence aux proches renseignés à l'inscription, ainsi qu'à l'infirmière. Cet incident est en deuxième lieu enregistré par le système afin que le médecin et l'infirmière puissent accéder à cette donnée via l'interface.

2.4 Maintenance

Dans le cas où le médecin ou l'infirmière constateraient des données incohérentes, signe de mal fonction du bracelet, ces derniers peuvent tous-deux contacter le Service Après-Vente via notre site internet ou un numéro de téléphone, afin qu'un technicien puisse être envoyé au domicile du patient pour effectuer un dépannage, voir remplacer le matériel fourni. Le patient peut lui aussi signaler un problème s'il en constate un.

3. Domain Driven Design

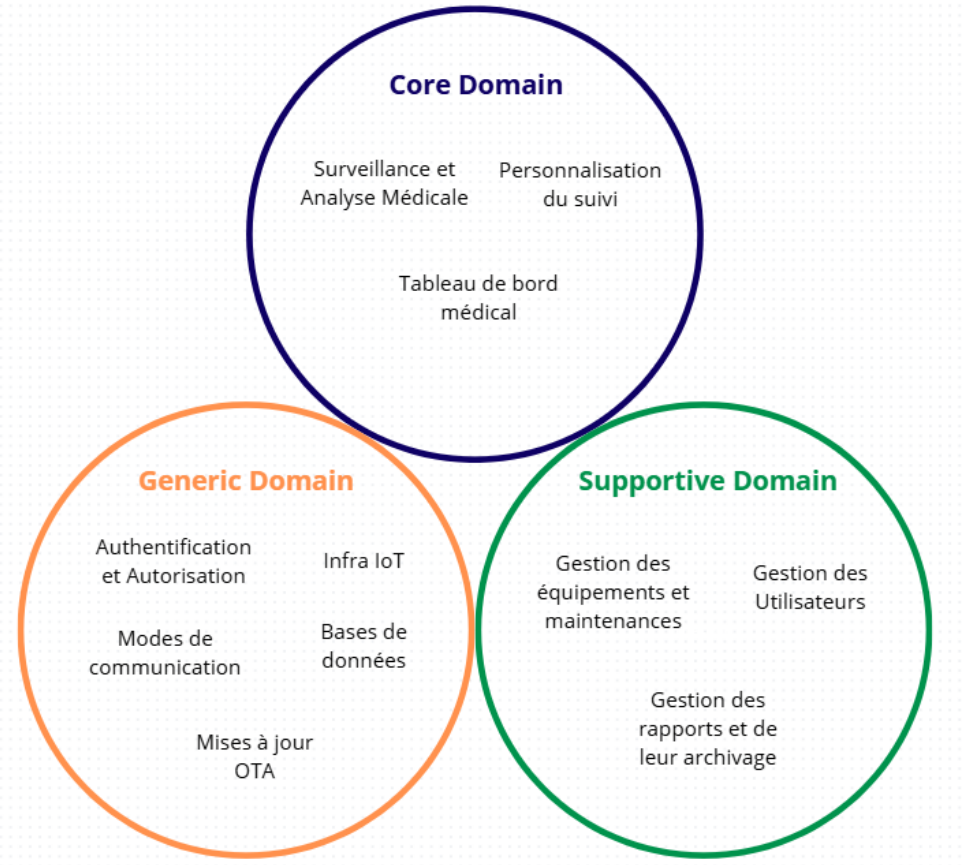

Figure 1 : Diagramme de vue d'ensemble du Domain Driven Design

3.1 Core Domain

Le Core Domain représente tout ce qui, dans notre système, apporte de la valeur. On y retrouve :

1. La surveillance et l'analyse médicale :

- Collecte des mesures
- Envoi des mesures au cloud
- Filtrage et validation des données
- Gestion des seuils d'alerte
- Détection des situations critiques et notification en cas d'alerte
- Génération de rapports médicaux périodiques

2. La personnalisation du suivi du patient :

- Définition des formulaires quotidiens

- Récupération des réponses du patient

3. Le tableau de bord médical :

- Vue globale des patients suivi par un médecin
- Accès aux mesures, tendances, historique d'un patient
- Visualisation des données

3.2 Le Supporting Domain

Le supporting domain sont des éléments indispensables au bon fonctionnement du système, mais qui ne sont pas directement différenciateurs sur le plan métier.

On y retrouve :

1. La gestion des utilisateurs et des rôles :

- Comptes médecin, infirmière, patient, administrateur
- Liaisons médecin ↔ patient ↔ infirmière

2. La gestion des équipements et la maintenance :

- Association des numéros de séries (bracelet et boitier) aux patients
- Gestion des maintenances et interventions à domicile

3. La gestion des rapports et leur archivage :

- Stockage des rapports générés
- Historisation, téléchargement

3.3 Le Generic Domain

Le Generic domain regroupe les "briques" techniques communes à énormément de systèmes logiciels, indépendants du domaine de l'e-santé.

On y retrouve :

1. L'authentification et l'autorisation :

- Login, gestion des tokens d'authentification
- Sécurité d'accès basiques

2. L'infrastructure IoT et les modes de communication :

- Communication via Kafka
- Protocoles de communication
- Envois de SMS

3. Le stockage et les bases de données :

- Base Timeseries
- Base relationnelle
- SFTP pour la mise à jour firmware

4. Mise à jour OTA (Over-The-Air) :

- Lecture des versions sur le serveur SFTP
- Comparaison avec les versions locales
- Téléchargement et installations

4. Risques

Dans le but d'assurer la fiabilité et la conformité du système, une analyse des risques de type FMEA (Failure Mode and Effects Analysis) a été réalisée. Cette méthode consiste à identifier, pour chaque étape critique du système, le risque, les modes de défaillance potentiels, leurs causes, leurs effets sur le fonctionnement global, et les actions préventives ou correctives à mettre en place. Concrètement, cette étude permet de lier directement la gestion du risque aux choix architecturaux.

Risque (Failure Effect)	**Étape concernée**	**Mode de défaillance potentiel**	**Causes possibles**	**Effet sur le système / l'utilisateur**	**SEV**	**OCC**	**DET**	**RPN**	**Contrôles existants**	**Actions recommandées**
Perte de données médicales	Transmission boîtier → Cloud / transmission bracelet →boitier	Données non envoyées	Panne boîtier, perte réseau, panne électrique	Données manquantes dans la base, absence d'alerte santé	9	6	3	162	Système détecte l'absence de données pendant X temps	Alerter le support + redondance locale (buffer) pour renvoyer lorsque possible
Altération/ corruption de données	Traitement dans backend Kafka / InfluxDB	Données modifiées ou incomplètes	Bug logiciel, mauvaise sérialisation JSON, bug de timestamp	Rapports faussés, fausses alertes	8	3	4	96	Validation du format et checksum avant insertion	Vérification d'intégrité, validation schéma JSON, tests
Fuite de données (Data breach)	Transfert de données/ stockage	Accès non autorisé, exfiltration	Piratage BDD, faille Cloud, compte admin compromis	Violation RGPD, perte de confiance des utilisateurs	10	3	7	210	Données chiffrées, authentification forte	Rotation des clés (de DB par exemple), audit sécurité, tests de pénétration
Erreur de valeur des données	Données acquises par le capteur	Données bruitées ou fausses	Défaillance capteur	Fausse alerte ou absence d'alerte,	8	6	4	192	Détection de valeurs hors plage, nettoyage	Analyse statistique automatique pour

mesurées (valeurs aberrantes)				rapports faussés					des données impossibles	détecter une fréquence anormale de valeurs incohérentes et contrôle du matériel si besoin
Perte temporaire de connectivité	Envoi BLE ou Wi-Fi	Déconnexion réseau	Éloignement, brouillage, panne boîtier	Données manquantes sur période donnée	6	8	3	144	Reconnexion automatique	Buffer local + Retry
Mauvaise gestion des droits d'accès	API backend / interface web	Mauvaise isolation des rôles	Mauvais mapping rôle-utilisateur, bug Auth	Exposition de données d'autres patients	9	3	4	108	Contrôles JWT	Tests automatisés de permissions
Retard de traitement des données critiques	Pipeline Kafka → Backend	Goulot d'étranglement	Surcharge réseau, partition saturée (trop de messages accumulés)	Alerte médicale reçue en retard	8	3	4	96	Envoi des alertes urgentes via le boîter (carte SIM)	Surveillance du débit Kafka, autoscaling, priorisation des flux critiques grâce à des requêtes synchrones
Mauvaise manipulation admin (suppression)	Interface d'administration	Suppression ou modification involontaire de données critiques	Erreur humaine, absence de double confirmation, bug dans l'interface	Perte définitive de données patients, rupture de cohérence entre comptes et mesures	8	3	3	72	Double confirmation requise avant action critique	Journalisation des actions, sauvegarde automatique temporaire avant suppression
Mise à jour logicielle incomplète (boîtier/bracelet)	Déploiement firmware	Téléchargement interrompu	Coupure d'alimentation, perte réseau pendant la mise à jour	Dispositif inutilisable, données non transmises	7	3	3	63	Mécanisme de rollback, checksum du firmware + télémétrie OTA	Procédure de reflash manuelle (en USB-C, par notre site, grâce à Web-USB)
Non-conformité RGPD (durée de rétention)	Stockage des données	Conservation excessive de données de santé	Mauvaise configuration de la rétention, oubli d'expiration, script cron de nettoyage qui échoue	Risque légal et atteinte à la vie privée	10	3	4	120	Rétention configurée dans TimeSeries/PostgreSQL, supervision du cron	Politique explicite de durée de conservation, alertes automatiques sur volumes anormaux, audit RGPD
Mauvais chiffrement des données en transit	Transmission boîtier → cloud	Données non chiffrées ou TLS obsolète	Certificat expiré, protocole mal configuré	Fuite potentielle de données, interception réseau	10	3	3	90	HTTPS obligatoire, certificats TLS vérifiés	Rotation automatique des certificats, vérification TLS via scanner automatisé

Légende :

- SEV (Severity) : gravité de l'impact si le risque se produit (1 = faible, 10 = critique).
- OCC (Occurrence) : probabilité d'apparition du risque sur la durée de vie du système (1 = rare, 10 = fréquent).
- DET (Detection) : probabilité de détection avant l'impact utilisateur (1 = facilement détectable, 10 = difficile à détecter).

- RPN (Risk Priority Number) = SEV × OCC × DET (plus il est élevé, plus une action est prioritaire).
- Actions recommandées : mesures préventives ou correctives à mettre en œuvre pour réduire la probabilité ou la gravité du risque.

Afin de compléter l'analyse FMEA, une évaluation des risques par la méthode Bowtie a également été réalisée.
Cette approche offre une vision plus globale et causale du risque, en représentant graphiquement :

- Les causes potentielles d'un événement indésirable
- Les barrières préventives qui permettent d'en éviter la survenue
- L'événement redouté (Top Event)
- Les conséquences si celui-ci se produit
- Les mesures d'atténuation qui en limitent les impacts

Cette méthode permet donc d'analyser la robustesse des contrôles existants et d'identifier les failles possibles dans la chaîne de prévention ou de mitigation.
Deux risques majeurs issus de notre FMEA ont été développés selon ce modèle :

- La fuite de données sensibles (data breach) : risque critique de sécurité et de conformité.
- La perte de données médicales : risque opérationnel impactant directement la fiabilité du suivi patient.

Bowtie n°1 : fuite de données sensibles

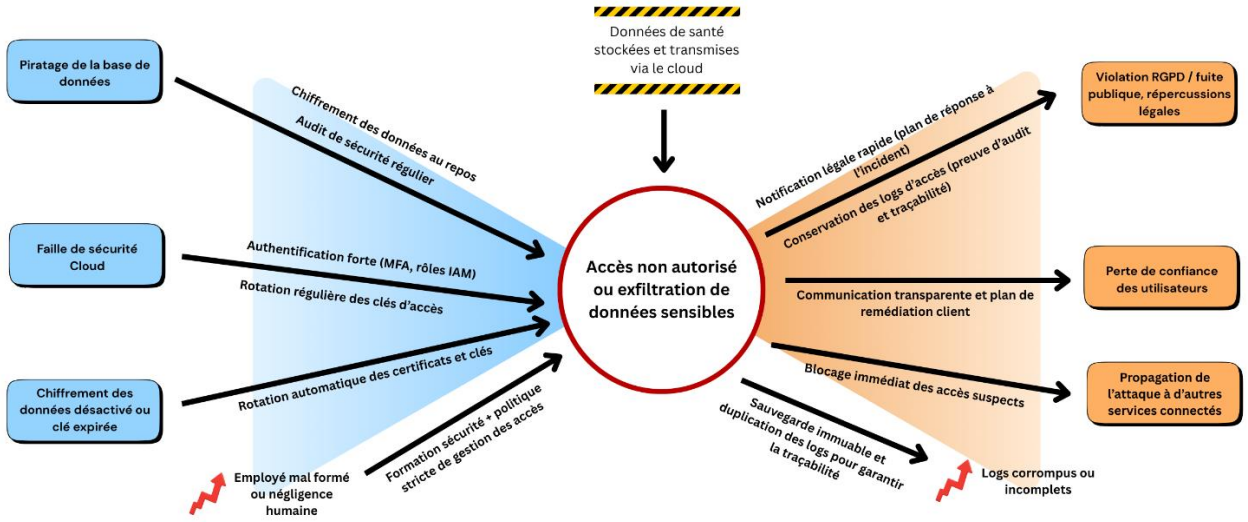

Figure 2 : Diagramme Bowtie du risque “fuite de données sensibles”

Bowtie n°2 : perte de données médicales

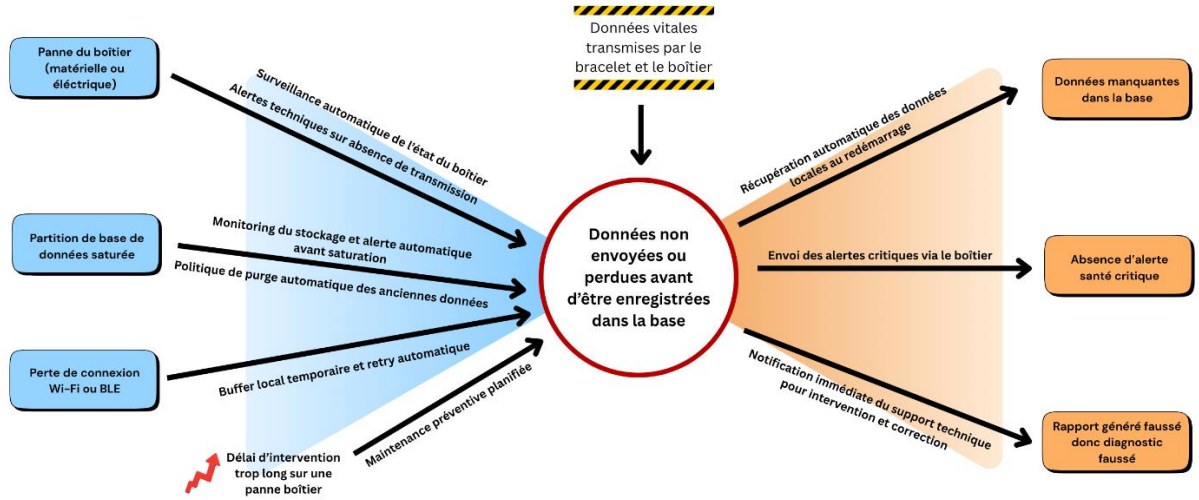

Figure 3 : Diagramme Bowtie du risque “perte de données médicales”

Légende des diagrammes Bowtie :

- **Hazard** *(zone jaune et noire)* :
  Source potentielle de danger, c’est-à-dire un élément du système susceptible de provoquer un dommage**.**
- **Top Event** *(cercle rouge)* :
  Événement central où le risque se concrétise, marquant la perte de maîtrise du système.
- **Preventive Controls** *(encadrés bleus + flèches vers le Top Event depuis la gauche) :* barrières qui empêchent la survenue de l’événement en agissant sur les causes.
- **Mitigating Controls** *(encadrés oranges + flèches depuis le Top Event vers la droite) :* mesures qui réduisent l’impact si l’événement se produit, en agissant avant les conséquences.

- **Escalation Factors** *(flèches rouges)* :
  Circonstances ou faiblesses susceptibles d'affaiblir l'efficacité des contrôles.
- **Escalation Factor Controls** *(texte associé aux flèches rouges)* :
  Moyens mis en place pour prévenir ou neutraliser ces faiblesses**.**

5. Architecture détaillée du système

En prenant en compte nos besoins ainsi que notre analyse de risque, nous avons construit notre architecture selon le schéma ci-dessous :

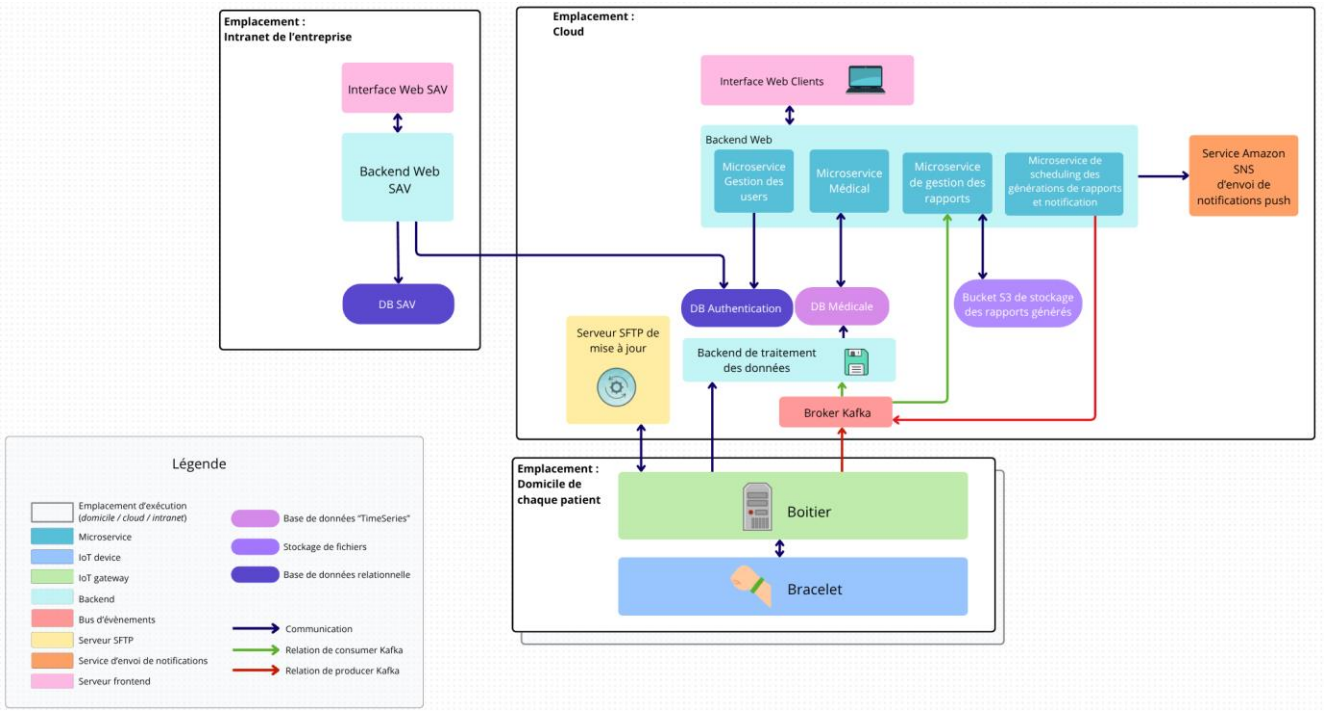

Figure 4 : Diagramme global de l'Architecture du système

Nous détaillons dans cette partie chaque composant de notre architecture.

5.1 Bracelet connecté

Le bracelet ici est notre IoT device.

5.1.1 Composants matériels

Les différents composants électroniques du bracelet sont détaillés ci-dessous :

- **Bouton d'appairage** : connexion avec le boitier.
- **Ecran ePaper** : permet de gérer un affichage simple avec le moins de consommation de batterie possible
- **Micro-contrôleur** (incluant sa mémoire vive et son stockage permanent)
- **Port USB-C** de recharge + de mise à jour de secours
- **Capteur Bluetooth Low Energy**

- **Batterie**
- Ainsi que les capteurs détaillés ci-dessous :
    - **Capteur IMU (accéléromètre, gyroscope, magnétomètre) :**
        - Indice médical mesuré : détection des chutes, nombre de pas
        - Indicateur biologique : accélération 3 axes
    - **Capteur PPG :**
        - Indice médical mesuré : fréquence cardiaque, oxygénation du sang
        - Indicateur biologique : Rythme cardiaque (BPM), oxygénation du sang (SpO2)

5.1.2 Répartition de la mémoire

La mémoire du micro-contrôleur suit un partitionnement permettant la restauration du système en cas d'erreurs lors des mises à jour. Ce partitionnement est détaillé ci-dessous :

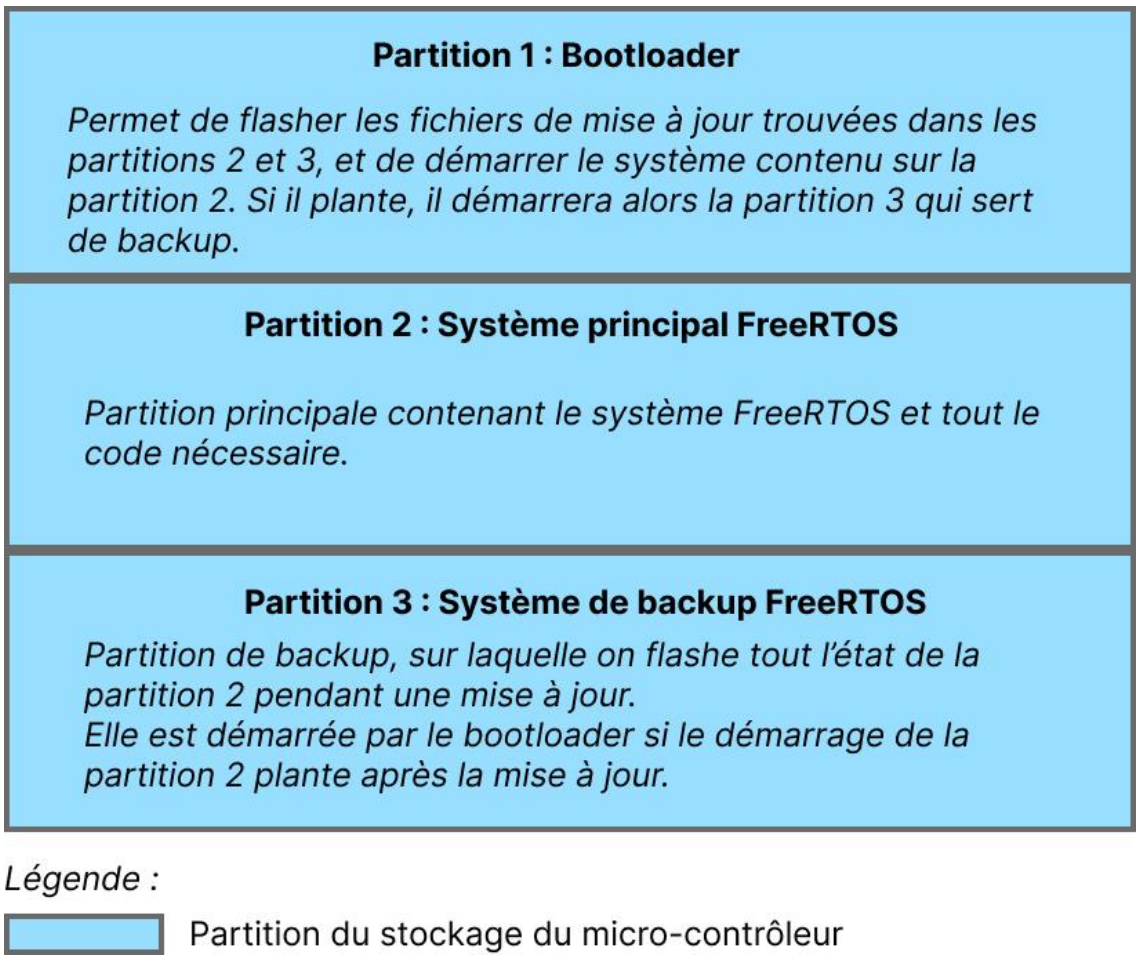

Figure 5 : Schéma de répartition de la mémoire du bracelet

5.1.3 Architecture logicielle

Le micro-contrôleur tourne sur un OS “FreeRTOS”, adapté à l’embarqué et permettant d’exécuter plusieurs “tâches”. Le système exécutera ces différentes tâches :

- Une tâche pour chaque capteur mesurera les données à une fréquence fixée pour chaque donnée (*voir les fréquences définies dans la partie “Data Ingestion Pipeline”)* depuis le capteur, les normalisera, et les enregistrera dans un tableau en RAM.
- Une tâche par capteur réalisera la moyenne du tableau du capteur (lorsque nécessaire), et l’enverra à la fréquence définie au boitier.
- Une tâche écoute continuellement l’appui sur le bouton d’appairage BLE, et effectue l’appairage lorsque demandé.
- Une tâche écoute une requête de mise à jour du bracelet provenant du boitier, et télécharge le fichier de mise à jour lorsque nécessaire. Alors, le bracelet

redémarrera, le bootloader effectuera un backup de la partition 2 sur la partition 3, et appliquera la mise à jour sur la partition 2. L'application d'une mise à jour reflashe complètement la partition. Ainsi, une mise à jour ne dépend pas de l'état précédent du système. Si la mise à jour échoue, le bootloader lance la partition 3, et affichera l'erreur de mise à jour sur l'écran E-paper.

- Une tâche mettra à jour le statut de l'écran E-paper, avec un visuel indiquant que tout fonctionne et un affichage de la batterie.

5.2 Boitier

Le boitier ici nous sert d'IoT gateway, entre le bracelet et le cloud.

5.2.1 Composants matériels

Les différents composants électroniques constituant le boitier sont listés ci-dessous :

- **Capteur BLE**
- **Nano-ordinateur** (type Raspberry Pi) compatible Linux
- **Modem GSM** (pour l'envoi de SMS)
- **Carte SIM intégrée** (pour l'envoi de SMS)
- **Écran E-paper** (pour l'affichage du statut du boitier)
- **Port Ethernet :** permet la connexion au cloud distant
- **Alimentation USB-C**
- **Bouton d'appairage Bluetooth**

5.2.2 Architecture logicielle

Le boitier utilisera un OS basé sur Linux. Ces différents programmes s'exécuteront dessus :

- Un script CRON vérifiera la disponibilité des mises à jour sur le serveur SFTP proposant les fichiers de mise à jour. Si une mise à jour est disponible pour le boitier, elle est appliquée, et si une mise à jour est disponible pour le bracelet, elle est envoyée au bracelet par BLE.
- Un script écoute constamment l'appui sur le bouton d'appairage bluetooth, et effectuera la connexion BLE lorsque demandé
- Un logiciel principal communiquera en Bluetooth avec le bracelet.
    - Si une alerte est envoyée par le bracelet (crise cardiaque, insuffisance respiratoire, chute), le boitier va d'une part envoyer un SMS à l'infirmière et au proche grâce au modem GSM. D'autre part, il va envoyer une requête

HTTP au backend de traitement des données, pour qu'il enregistre l'incident dans la DB TimeSeries.
    - Sinon, les données sont envoyées au broker Kafka, potentiellement plusieurs points à la fois selon le rythme défini dans la partie "Data Ingestion Pipeline".

Ce logiciel met également à jour le statut affiché sur l'écran E-paper, affiche l'heure et une indication que tout fonctionne bien en temps normal, et affiche les erreurs lorsque nécessaire.

5.3 Broker Kafka

Le broker Kafka, hébergé dans le cloud, récupère les messages des boitiers sur ses topics, et permet de stocker toutes les tâches à effectuer dans ses topics.

Il possède deux types de topics :

- Le topic permettant le traitement des données des bracelets. Les boitiers sont des *producers* ("producteurs de messages") de ce topic, et le backend de traitement des données en est un *consumer* ("consommateur" de messages).
- Le topic permettant d'indiquer la liste des rapports à générer. Le micro-service de scheduling de génération des rapports en est un *producer,* et le micro-service de gestions des rapports en est un *consumer.*

5.4 Backend de traitement des données ("Data Processing")

Le backend "Data Processing" est un "consumer" du broker Kafka. Il va s'abonner au topic du broker regroupant les données de bracelet reçues, et traiter les messages petit à petit. Il va recevoir les données, les filtrer lorsque nécessaire, et faire l'enregistrement dans la DB "Médicale".

Il suit une architecture en pipeline, permettant de faire suivre les différents traitements à apporter à la donnée : filtrage de données incohérentes (valeurs négatives ou beaucoup trop grandes), conversion lorsque nécessaire, agrégation lorsque nécessaire, enregistrement en DB.

5.5 Base de données "médicale"

La base de données "médicale" est celle regroupant les mesures récoltées par les bracelets, ainsi que les formulaires, et traitées par le backend "Data Processing". Nous une

base de données de type TimeSeries, pour stocker les données des patients. Cette base de données est particulièrement adaptée au stockage de valeurs temporelles.

Cette base de données est chiffrée afin d'assurer la protection des données qui sont sensibles.

5.6 Backend web

Le backend web suit une architecture micro-services et permet de faire fonctionner le site web et l'application mis à disposition des médecins et infirmières. Il met à disposition une API REST, permettant l'authentification (en utilisant les données de la base de données "Authentication", détaillée plus bas), ainsi que la récupération des données de la DB médicale, et l'enregistrement des réponses au formulaire. Ce backend effectue un traitement minimal sur les données de la DB, et effectue juste le pré-traitement nécessaire au front-end : parsing des données, récupération sur une plage de temps définie ...

Le découpage en micro-services est le suivant :

- Micro-service "médical" : récupération des données de la base de données TimeSeries, parsing, filtrage et traitements utiles à l'affichage. Ce micro-service reçoit également les réponses au formulaire et les enregistre dans la DB Timeseries.
- Micro-service "gestion des users" : chargé des traitements liés aux comptes comme l'authentification, la gestion du profil ...
- Micro-service de scheduling de génération de rapport/des notifications : un micro-service possédant un CRON job appelé toutes les nuits, qui va vérifier pour chaque patient s'il est temps de générer son rapport. Si oui, un message va être ajouté au broker Kafka, demandant la génération du rapport du patient.
  Il possède un autre CRON job qui vérifie pour chaque patient s'il est l'heure de le notifier pour son formulaire. Si oui, il fait un appel au service "Amazon SNS", permettant l'envoi de notifications push.
- Micro-service de gestion des rapports : ce micro-service est un *consumer* du broker Kafka, et va générer le rapport d'un patient lorsque demandé. Il récupère les informations du patient depuis le micro-service médical, puis enregistre le rapport au format PDF dans un bucket S3. Ce micro-service est aussi chargé de fournir au front-end les fichiers de rapport lorsque le médecin le demande.

5.7 Interface web clients

Le serveur frontend "Interface web clients” héberge le site accessible par les médecins, infirmières et patients. Il est conçu comme un seul et unique frontend, qui proposera un affichage différent en fonction du rôle de l’utilisateur. Ce frontend web communique exclusivement avec le backend web.

5.8 Base de données “Authentication”

La base de données “Auth” est une DB relationnelle. Elle stocke les informations des comptes, et permet l’authentification à la plateforme web. Elle est administrable par le SAV, et est également utilisée par le micro-service “gestion des users” du backend web pour gérer l’authentification sur la plateforme web. Celle-ci est chiffrée pour protéger les informations sensibles des utilisateurs.

5.9 Bucket S3 de stockage des rapports

Lorsque les rapports sont générés automatiquement, ils sont stockés dans un Bucket S3 de AWS. C’est une solution efficace et économique, adaptée au stockage d’un grand nombre de fichiers. Ce bucket est uniquement manipulé par le micro-service “gestion des rapports”.

5.10 Serveur SFTP de mise à jour

Les fichiers binaires contenant les mises à jour du boitier et du bracelet sont mis à disposition sur un serveur SFTP. Chaque fichier contient dans son nom le type de mise à jour (bracelet ou boitier) et le numéro de version. Les boitiers peuvent donc à intervalle régulier chercher les mises à jour sur ce serveur, et les télécharger si une nouvelle version est disponible.

5.11 Intranet de l’entreprise

Une plateforme web est mise à disposition des administrateurs (employés de l’entreprise) pour configurer les comptes des médecins, infirmières et patients. Celle-ci est limitée à l’intranet de l’entreprise, pour limiter les failles de sécurité. Un serveur héberge le frontend, et un autre héberge le backend. Ce backend est ainsi relié à la DB d’authentification, ainsi

qu'à la DB d'administration permettant de stocker les informations de connexion des administrateurs.

Le frontend est utilisable par les administrateurs pour effectuer toutes les opérations d'administration : changer les liens entre médecins/infirmières/patients, créer les profils ou les supprimer...

6. Flux de données

Notre système repose sur un pipeline de données complète assurant la collecte, le traitement et l'exploitation des informations issues des capteurs physiologiques embarqués dans le bracelet.

Pour les diagrammes de flux de données présents dans cette section, nous nous réfèrerons à la légende décrite ci-dessous :

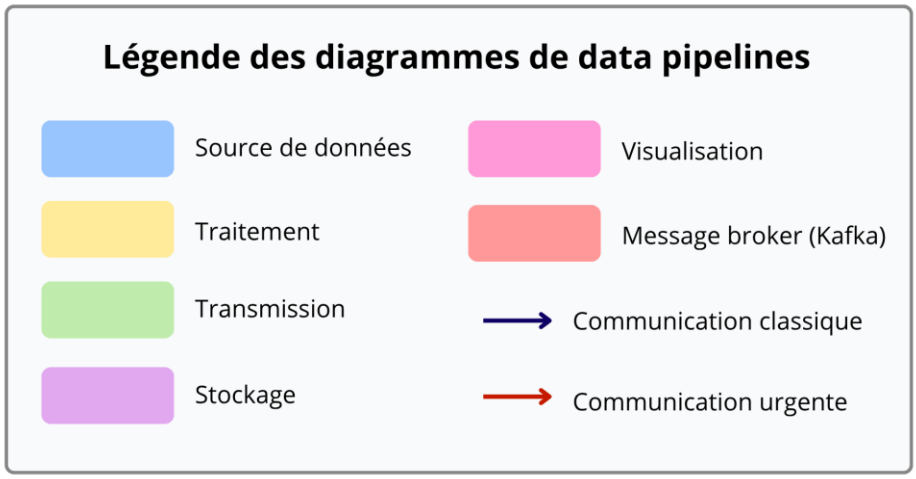

Figure 6 : Légende des diagrammes de flux de données

6.1 Collecte et intégration des données

La phase de *data ingestion* correspond à la collecte, au prétraitement et à la transmission des différentes données physiologiques mesurées par le bracelet.
Notre système a été conçu pour gérer plusieurs types de données hétérogènes, chacune possédant un mode de traitement et de transmission spécifique, adapté à sa criticité.
Certaines informations, comme les mesures courantes de rythme cardiaque ou de pas, sont envoyées de manière asynchrone via le broker Kafka, tandis que les événements

critiques (crise cardiaque, chute, insuffisance respiratoire) déclenchent des transmissions synchrones et immédiates vers le backend afin d'assurer une transmission immédiate.

6.1.1 Rythme cardiaque ou BPM

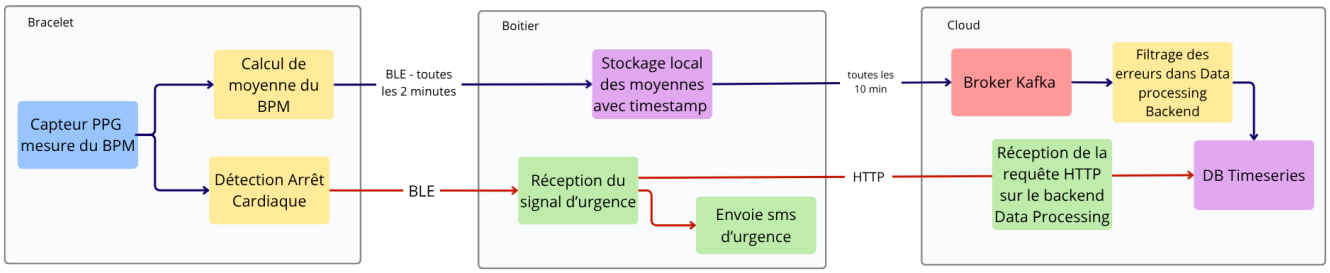

Figure 6 : Diagramme de flux du Rythme Cardiaque

Le rythme cardiaque est mesuré en continu par le capteur PPG (Photo-pléthysmographie) intégré au bracelet.
Chaque battement détecté est comptabilisé localement afin d'évaluer la fréquence cardiaque instantanée. Le bracelet exécute en parallèle un algorithme de détection d'arrêt cardiaque. En cas de crise cardiaque détectée, un signal d'urgence est immédiatement transmis au boîtier via la communication BLE (Bluetooth Low Energy). Le boîtier réagit en envoyant un SMS d'urgence aux contacts des proches de la personne âgée préenregistrés, puis transmet une requête HTTP synchrone au backend *Data Processing* afin d'enregistrer l'événement dans la base de données de séries temporelles (DB Timeseries).

En conditions normales, le bracelet calcule et envoie une moyenne du BPM toutes les deux minutes au boîtier. À la réception de ces valeurs, le boîtier les stocke localement avec un horodatage (timestamp) pour garantir la traçabilité même en cas de perte de connectivité.
Toutes les dix minutes, les cinq dernières valeurs moyennes sont regroupées et envoyées au broker Kafka. Le module *Data Processing* consomme alors les messages Kafka, filtre les anomalies ou erreurs de mesure, puis insère les données validées dans la DB médicale Timeseries.

6.1.2 Détection de chute

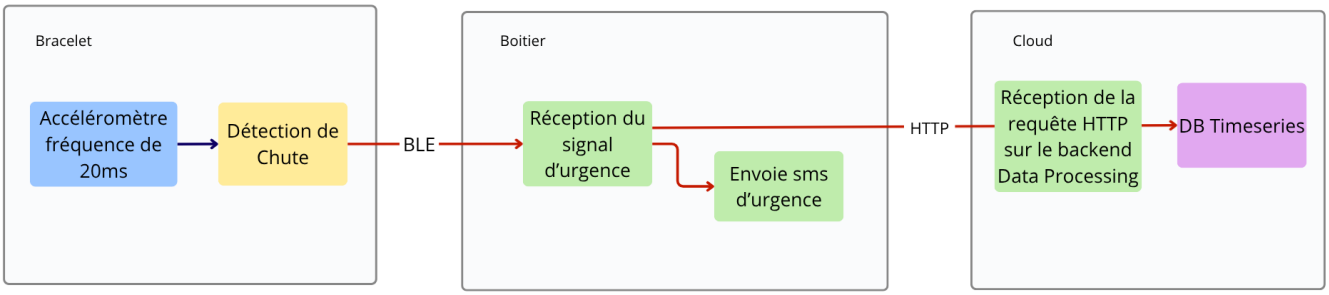

Figure 7 : Diagramme de flux du signal de chute

La détection de chute repose sur les mesures de l'accéléromètre embarqué dans le bracelet.
Celui-ci vérifie l'accélération toutes les 10 millisecondes pour identifier les variations brusques de mouvement caractéristiques d'une chute.

Lorsqu'une chute est détectée, le bracelet envoie immédiatement un signal d'urgence BLE au boîtier. Ce dernier envoie un SMS d'alerte aux personnes concernées, puis transmet une requête HTTP synchrone au backend *Data Processing* pour enregistrer l'événement dans la DB Timeseries.

6.1.3 Nombre de pas

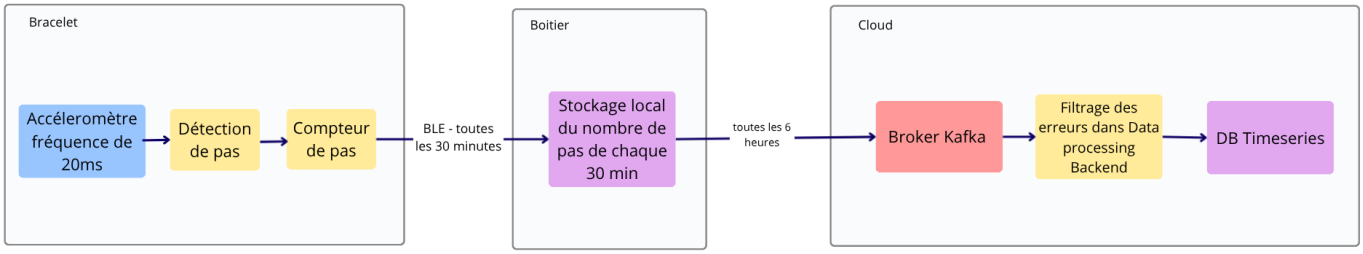

Figure 8 : Diagramme de flux du Nombre de Pas

Le suivi du nombre de pas repose également sur les données issues de l'accéléromètre, analysées avec une granularité de 20 millisecondes.
Le bracelet détecte chaque pic d'accélération supérieur à un seuil de 1,2 g, qu'il interprète comme un pas. Toutes les 30 minutes, le bracelet transmet au boîtier le nombre total de pas détectés.

Le boîtier regroupe ensuite les comptages des six dernières heures et les envoie de manière asynchrone au broker Kafka.

6.1.4 Saturation en oxygène (SpO_2)

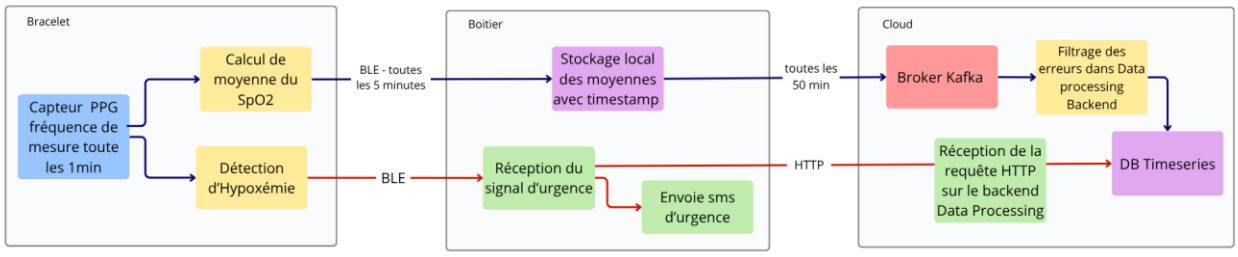

Figure 9 : Diagramme de flux de la saturation en oxygène (SpO_2)

Le bracelet mesure la saturation en oxygène du sang (SpO_2) toutes les minutes. Si la moyenne sur une minute indique une valeur inférieure au seuil d'hypoxémie, le bracelet déclenche un signal d'urgence. Le boîtier envoie alors un SMS d'urgence et transmet une requête HTTP synchrone au backend *Data Processing* afin d'enregistrer cette insuffisance respiratoire dans la DB Timeseries.

En l'absence d'anomalie, les moyennes de SpO_2 sont envoyées au boîtier toutes les cinq minutes, puis regroupées par paquets de dix mesures avant d'être transmises de manière asynchrone au broker Kafka. Le backend filtre ensuite ces données et les stocke dans la DB Timeseries après validation.

6.1.5 Réponses du formulaire

Les réponses aux formulaires constituent un second type de données collectées par le système, distinctes des mesures physiologiques.
Elles proviennent de l'application web ou mobile utilisée par le patient, et permettent au médecin de suivre l'évolution subjective de l'état de santé (fatigue, appétit, douleur, humeur, etc.).

Chaque formulaire est généré automatiquement selon la configuration définie par le médecin (choix des questions, fréquence d'envoi). Lorsqu'un patient répond à un formulaire, ses réponses sont immédiatement envoyées au backend web via une requête HTTP sécurisée (HTTPS).
Le micro-service de gestion médicale réceptionne ces données, les valide (vérification du

format, des identifiants patient et médecin, cohérence temporelle), puis les enregistre dans la base de données médicale (TimeSeries), au même titre que les mesures physiologiques.

Le système distingue donc deux types de flux de données. Les flux critiques et synchrones correspondant aux crises cardiaques, aux chutes et aux insuffisances respiratoires, sont transmis dès leur réception dans le boitier par des sms aux proches et à l'infirmière, ainsi que par requêtes HTTP directes vers le backend afin d'assurer une réaction immédiate et fiable, ces événements ayant un caractère urgent et peu fréquent. Les flux continus et asynchrones, tels que les mesures de BPM moyen, le nombre de pas ou la SpO_2 normale, sont quant à eux envoyés au broker Kafka pour un traitement différé et réparti.

Cette organisation permet de combiner la réactivité du temps réel pour les urgences avec une ingestion efficace et robuste pour les données courantes.

6.2 Exploitation des données

Une fois les données médicales collectées, filtrées et stockées dans les bases de données associées, la phase d'exploitation des données permet leur visualisation pour faciliter l'analyse du médecin et de l'infirmière. Dans cette partie nous expliquerons le fonctionnement des flux d'exploitation de données à travers deux mécanismes principaux : l'affichage des mesures en temps réel sur l'interface web et la génération automatique de rapports médicaux.

6.2.1 Affichage des données dans l'interface web

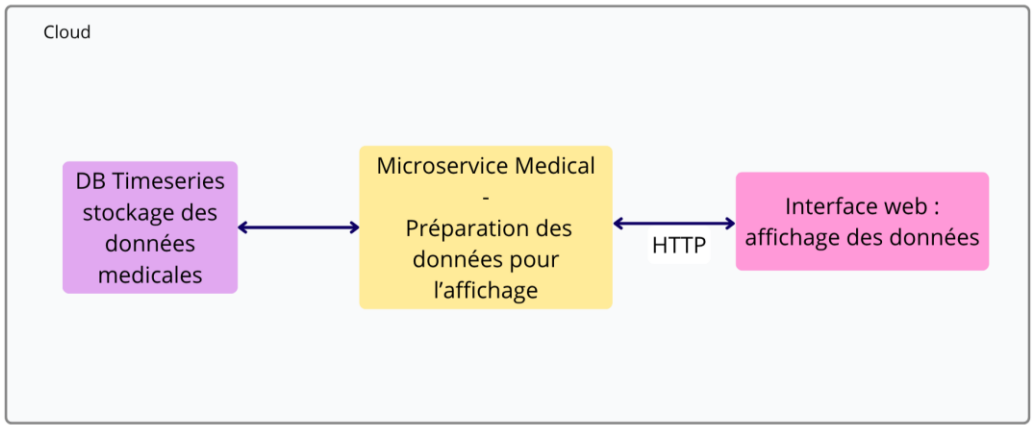

Figure 10 : Diagramme de flux lié à l'affichage des données

Les données physiologiques validées (rythme cardiaque, SpO_2, activité, etc.) sont stockées dans la base de données de séries temporelles (DB Timeseries).
Un micro-service médical interagit directement avec cette base afin de préparer les données avant leur envoi à l'interface utilisateur. Ce service réalise les agrégations, filtrages et normalisations nécessaires pour que les informations soient exploitables et présentées de manière claire et fluide.

Lorsque l'utilisateur consulte son tableau de bord, l'interface web envoie une requête HTTP au micro-service médical, qui récupère les données les plus récentes dans la DB Timeseries et retourne une réponse structurée (généralement au format JSON).
L'interface web se charge ensuite de l'affichage dynamique des indicateurs de santé sous forme de graphiques et de visualisations temporelles.

6.2.2 Génération automatique des rapports médicaux

En complément de l'affichage direct des mesures, le système permet la génération périodique de rapports médicaux synthétiques, destinés aux utilisateurs ou aux professionnels de santé. Ce processus débute par le micro-service de *scheduling*, chargé de déterminer la nécessité de générer un rapport selon une fréquence définie (quotidienne, hebdomadaire, etc.). Lorsque la génération est requise, le service contacte le micro-service de gestion des utilisateurs pour vérifier les droits d'accès et l'authentification auprès de la base d'authentification. Une fois la vérification validée, la demande est transmise au broker Kafka, qui assure l'envoi asynchrone du message vers le micro-service de gestion des rapports, dans le but de répartir la génération de rapport dans le temps. Ce dernier récupère les données pertinentes dans la base médicale et procède à la création du rapport. Une fois généré, celui-ci est stocké dans un bucket S3, garantissant sa persistance et son accessibilité.
Enfin, le micro-service d'obtention des rapports permet à l'interface web de récupérer le rapport stocké via une requête HTTP pour en assurer l'affichage ou le téléchargement par l'utilisateur.

Cette chaîne de génération repose sur une architecture découplée et résiliente, où Kafka gère les demandes asynchrones, tandis que les micro-services spécialisés assurent la sécurité, la modularité et la scalabilité du processus.

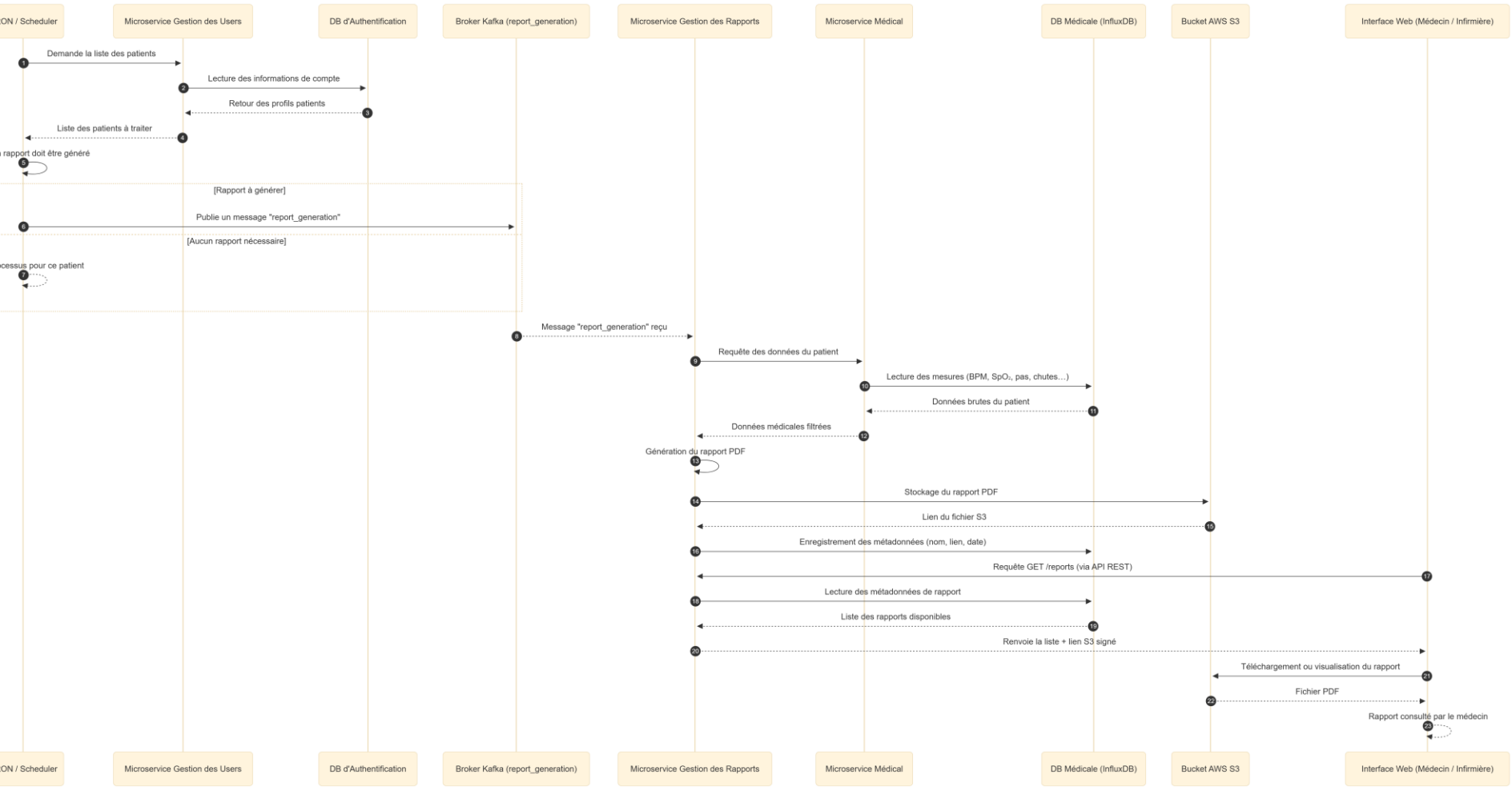
ON / Scheduler
Microservice Gestion des Users
DB d'Authentification
Broker Kafka (report_generation)
Microservice Gestion des Rapports
Microservice Médical
DB Médicale (InfluxDB)
Bucket AWS S3
Interface Web (Médecin / Infirmière)
1 Demande la liste des patients
2 Lecture des informations de compte
3 Retour des profils patients
4 Liste des patients à traiter
rapport doit être généré
5
[Rapport à générer]
6 Publie un message "report_generation"
[Aucun rapport nécessaire]
cessus pour ce patient
7
8 Message "report_generation" reçu
9 Requête des données du patient
10 Lecture des mesures (BPM, SpO₂, pas, chutes…)
11 Données brutes du patient
12 Données médicales filtrées
13 Génération du rapport PDF
14 Stockage du rapport PDF
15 Lien du fichier S3
16 Enregistrement des métadonnées (nom, lien, date)
17 Requête GET /reports (via API REST)
18 Lecture des métadonnées de rapport
19 Liste des rapports disponibles
20 Renvoie la liste + lien S3 signé
21 Téléchargement ou visualisation du rapport
22 Fichier PDF
23 Rapport consulté par le médecin
ON / Scheduler
Microservice Gestion des Users
DB d'Authentification
Broker Kafka (report_generation)
Microservice Gestion des Rapports
Microservice Médical
DB Médicale (InfluxDB)
Bucket AWS S3
Interface Web (Médecin / Infirmière)

Figure 11 : Diagramme de séquence de la génération de rapports

7. Choix de l'hébergement Cloud

Afin d'assurer la disponibilité et la sécurité du système de monitoring médicale, on choisit de déployer nos serveurs sur le cloud sur la plateforme Amazon Web Services (AWS).

7.1 DB Authentification : base de données relationnelle

- Usage : stockage des données d'utilisateurs
    - Nom, prénom, médecin traitant, infirmière assignée, et ensemble des rapports générés, et les formulaires remplis par le patient.
- Population cible : 150 000, dont 100 000 patients qui remplissent quotidiennement des formulaires.
- Volume estimé : environs 50 Go de données
- Instance: db.tg4.medium (2 vCPU, 8 GiB RAM, stockage 50 Go SSD).
- Le coût total est estimé à 59.21$
    - Disponibilité à 100% : 1 instance(s) x 0.072 USD horaire x (100 / 100 utilisés/mois) x 730 heures dans un mois = 52.5 USD
    - 50Go de SSD gp3 : 50 Go par mois x 0,133 USD x 1 instances = 6,65 USD/mois (coût de stockage)

C'est le choix le plus économique possible pour nos besoins actuels tout en maintenant, une capacité suffisante pour 150 000 comptes et une sécurité managée (sauvegarde automatique, chiffrement des données, mise à jour automatiques).

7.2 Interface Web Clients :

- Usage : hébergement de l'application web statique
    - Le site permet la consultation des rapports patients, la visualisation des métriques de suivi et la connexion des différents acteurs (médecins, infirmières, administrateurs).
    - Les fichiers stockés sur S3 sont de type HTML, CSS, JavaScript et images, compilés depuis le framework React ou Angular.
    - L'application étant principalement consultative, la charge en lecture reste modérée et adaptée à un hébergement statique.

    - Volume estimé : environ 10 Go de données (fichiers du site, ressources graphiques et historiques de déploiement).

- Service utilisé : Amazon S3 (Simple Storage Service) en classe de stockage S3 Standard
- Coût estimé : 4.25$ / mois
    - Stockage standard S3 : 10 Go/mois × 0,023 $ = 0,23 $
    - Requêtes PUT/POST/LIST (déploiements du site) : 1 000 requêtes/mois × 0,005 USD / 1 000 = 0,01 $
    - Requêtes GET (consultation du site) : 10 000 000 requêtes/mois × 0,0004 USD / 1 000 = 4 $

7.3 Backend web :

- Usage : hébergement du backend principal assurant le fonctionnement des différents micro-services de l'application, responsables de la gestion des utilisateurs, du traitement des données médicales, de la génération des rapports et de la planification des notifications et rapports périodiques. Contient 4 conteneurs dockers, un par micro-service.
- Instance : EC2, t4g.xlarge (4 vCPU / 16 GiB)
    - 4cpu, 16Go de RAM pour avoir une puissance de calcul suffisante pour exécuter simultanément les 4 micro-services du backend dans les conteneurs docker distincts, et pour avoir une bonne réactivité face au trafic régulier des 100 000 patients qui utilisent l'application/le site web.
- Stockage : 100 Go SSD gp3
    - Pour assurer des temps d'accès rapide adaptés aux opérations fréquentes de lecture et d'écriture sur les bases de données, et la gestion des rapports.
- Système : Ubuntu Server 24.04 LTS
- Disponibilité à 99.99%
- Cout estimé : Total : 73.59 $/mois

7.4 DB Médicale Timeseries :

- Usage : base de données de séries temporelles (TimeSeries) destinée au stockage, à la consultation et à l'analyse des mesures physiologiques collectées en continu par les bracelets médicaux des patients (BPM, SpO_2, nombre de pas, détection de

chutes, alertes d'arrêt cardiaque). Les données sont regroupées et envoyées par lots toutes les dix minutes, puis traitées et insérées dans la base pour un suivi précis et historisé de chaque patient.

- Instance: Amazon Timestream
    - Configuration ajustée pour supporter un flux d'environ 1 200 écritures par seconde, correspondant à la remontée périodique des données issues de 100 000 patients actifs
    - Le moteur Timestream gère automatiquement la mise à l'échelle, la réplication et la répartition entre stockage en mémoire (hot store) et stockage magnétique (cold store), garantissant une disponibilité élevée et une faible latence pour les requêtes récentes.
- Stockage : séparation entre stockage en mémoire (données récentes à accès rapide) et stockage magnétique (données archivées pour l'analyse historique).
    - Stockage en mémoire : données des dernières 72H
    - Stockage magnétique : données de la dernière année
- Coût estimé :
    - Cout total : 4505.59 $/mois (écritures en mémoire : 318.04$, écritures magnétiques 318.04$, stockage magnétique 421.40$, stockage en mémoire 1,515.63$, nombre total de requêtes 1,932.48$)

7.5 Backend de traitement des données Data Processing:

- Usage : hébergement et exécution du module Data Processing, responsable du traitement en flux continu des données issues des bracelets connectés, Le service agit comme consommateur du broker Kafka, en s'abonnant au topic regroupant les mesures physiologiques des patients et traite et filtre les données avant de les enregistrer dans la base de données médicale (Timestream).
- Instance : Amazon ECS Fargate (service managé de conteneurs Docker)
    - Le module Data Processing est déployé sous forme de conteneur Docker exécuté sur ECS Fargate, permettant une exécution serverless : AWS gère automatiquement le provisioning, la montée en charge et la reprise après panne.

    - Chaque conteneur dispose de 2 vCPU et 4 Go de mémoire, suffisants pour traiter un flux de ≈1 200 messages par seconde en provenance de Kafka
- Système : Le conteneur s'exécute sur Ubuntu (image Docker)
- Coût estimé : 66.33 $/mois

7.6 Broker Kafka

- Usage : Assure le transport fiable et asynchrone des données entre les différents composants du système (boîtiers patients → backend Data Processing → base Timeseries).
- Service: Amazon Managed Streaming for Apache Kafka (Amazon MSK)
- Configuration : 3 brokers hébergés sur une instance kafka.t3.small (2 vCPU, 2 GiB RAM) suffisants pour traiter un flux de ≈1 200 messages par seconde en provenance de Kafka avec 24h de rétention des messages, et un stockage attaché de 100Go EBS gp3 pour assurant une persistance des messages en cas de panne temporaire du backend.
- Coût estimé : 148.02 $/mois

7.7 Bucket Amazon S3 – Stockage des rapports générés

- Usage : Stocker les rapports PDF générés périodiquement pour chaque patient.
- Service: Amazon S3
- Configuration : 150Go/mois (nombre de patients x plusieurs rapports mensuels) et une estimation de 400 000 PUT/LIST et 2 000 000 GET.
- Coût estimé : 34.59 $/mois

7.8 Serveur SFTP – Mise à jour des boîtiers

- Usage : Déployer les mises à jour logicielles sur les boîtiers connectés tous les 4 mois.
- Service: AWS Transfer Family (protocole SFTP)
- Configuration : 1 serveur SFTP managé, 2,4 To de transferts sortants mensuels estimés (100 000 boîtiers).
- Coût estimé : 257.20 $/mois

7.9 Service de Notification

- Usage : Envoi automatique de notifications par SMS à la réception d'une alerte médicale critique
- Service: Amazon SNS (Simple Notification Service)
- Configuration : Environ 2 000 SMS envoyés par mois (alertes critiques).
- Coût estimé :
    - Requêtes et notifications SNS : 0,54 $/mois
    - Notifications push mobiles : 0,67 $/mois
    - SMS envoyés : 2 000 × 0,04 $ = ≈ 80 $/mois

Coût total de l'architecture cloud estimé à 5230$/mois.

8. Passage à l'échelle

Notre système est conçu de manière à être scalable. Ainsi, nous pourrons adapter les performances (et donc les coûts) de notre architecture en fonction de notre masse d'utilisateurs.

Nous pouvons commencer par constater plusieurs éléments :

- Un des points sollicitant le plus de performance côté cloud est la réception et le traitement des données des bracelets et des formulaires.
- Les données de bracelet sont envoyées à toutes les heures de la journée, tous les jours. Ainsi, il y aura très peu de différences de trafic entre la nuit, le jour, ou entre chaque jour, semaine...
- Il n'y a pas de raison que la masse d'utilisateurs évolue brusquement (excepté au lancement du service les premières semaines).

D'après ces éléments, il ne serait donc pas forcément nécessaire que notre architecture supporte de l'auto-scaling très réactif (pour ce qui est du traitement des données des bracelets et des formulaires en tout cas). Le passage à l'échelle peut se faire de manière plus progressive et lente, et ne devrait pas avoir besoin de se faire en 15 minutes.

En revanche, il est quand même nécessaire que notre système puisse passer à l'échelle, notamment pour la plateforme web, ou pour l'évolution long-terme du nombre de patients.

L'utilisation d'EKS (Elastic Kubernetes Service), proposé par AWS, permettra d'assurer la scalabilité de notre système de manière fiable et progressive. Grâce à l'orchestration

Kubernetes, chaque composant de notre architecture, qu'il s'agisse des microservices, du backend monolithique ou des traitements de données, pourra être répliqué et ajusté en fonction de la charge réelle.

EKS permettra de définir, pour chaque service, des règles d'adaptation adaptées à son usage. Par exemple, les services traitant les données des bracelets pourront évoluer lentement et régulièrement, tandis que la plateforme web bénéficiera d'une montée en charge plus réactive, afin de répondre rapidement à un afflux d'utilisateurs.

La répartition du trafic entre les différentes instances sera gérée automatiquement grâce au load balancing intégré, garantissant la continuité du service même en cas de panne ou de surcharge.

La montée ou la descente en charge pourra se faire à deux niveaux :

- D'une part, en ajustant automatiquement le nombre d'instances d'un service selon la charge observée
- D'autre part, en adaptant la capacité globale du cluster lorsque la demande évolue à long terme.

Ainsi, EKS offrira une scalabilité fluide et maîtrisée, permettant au système de s'adapter à la croissance du nombre d'utilisateurs tout en optimisant les coûts et la performance.

9. CI/CD et DevOps

Les développeurs disposent de 3 environnements reproduisant l'architecture :

- "test" : les développeurs peuvent librement déployer des versions présentes sur le repo de développement pour chaque service. Il est absolument permis que l'environnement de test soit non-fonctionnel, c'est un outil de développement.
- "pre-prod" : cet environnement permet de tester le bon fonctionnement de tout le système avant de déployer la mise à jour d'un service en prod.
- "prod" : l'environnement réellement utilisé par notre système.

Les déploiements sur ces différents environnements sont réalisés grâce à des pipelines sur notre repository de code. Lors d'un déploiement sur la pre-prod, le contenu des données de production est récupéré, anonymisé, puis remplace le contenu des DB de pre-

prod. Ainsi, à chaque déploiement, la pre-prod possède des contenus en DB extrêmement proches de ceux en production (à ceci près qu'ils sont anonymisés).

Chaque environnement est monitoré grâce à l'outil "NewRelic", permettant de superviser tous les services pour tous les environnements, autant d'un point de vue logs, anomalies, ou performance.

Conclusion

Le projet présenté propose une architecture logicielle complète et structurée pour un système de suivi médical à distance, destiné principalement aux personnes âgées vivant à domicile. L'objectif initial était double : permettre le maintien à domicile en toute sécurité grâce à un système d'alerte, et faciliter le suivi médical continu à travers la collecte automatique de données physiologiques et de formulaires quotidiens.

L'architecture repose sur une chaîne de traitement claire, depuis la collecte des données par le bracelet connecté jusqu'à leur visualisation sur les interfaces web des médecins et infirmières. Le bracelet effectue les mesures physiologiques (rythme cardiaque, SpO_2, nombre de pas, détection de chute), tandis que le boîtier agit comme passerelle entre l'objet connecté et le cloud, en assurant la transmission sécurisée des informations et l'envoi d'alertes en cas d'incident critique.

Les données sont ensuite traitées par le backend "Data Processing", stockées dans la base de données médicale TimeSeries, et exploitées par le backend web via des microservices spécialisés. Le broker Kafka joue un rôle central dans la gestion asynchrone des flux de données et dans la génération automatique des rapports médicaux, garantissant la fiabilité et la modularité du système.

L'analyse des domaines métier selon le Domain Driven Design a permis d'identifier les composantes essentielles (Core Domain), les services de support (Supporting Domain) et les briques techniques génériques (Generic Domain), offrant ainsi une vision claire et hiérarchisée du système. De plus, l'étude des risques (FMEA et Bowtie) a mis en évidence les défaillances potentielles, leurs causes et les actions correctives à mettre en place, renforçant la robustesse et la conformité du dispositif.

Le déploiement sur le cloud AWS assure une infrastructure performante, sécurisée et adaptée aux besoins de stockage et de calcul du système. Les choix d'hébergement (bases de données, backend, interface web) ont été faits pour garantir la disponibilité, la sécurité et l'évolutivité du

service. Enfin, l'intégration d'EKS (Elastic Kubernetes Service) permet d'assurer le passage à l'échelle en fonction de la croissance du nombre d'utilisateurs, tout en maîtrisant les coûts.

Ainsi, ce projet aboutit à une solution complète, fiable et évolutive, répondant aux exigences du suivi médical à distance : collecte automatisée des données, transmission sécurisée, traitement efficace et restitution claire des informations aux professionnels de santé. L'ensemble des choix techniques et organisationnels présentés garantit un système cohérent, conforme et adapté aux besoins des patients, des médecins et du personnel infirmier.