

INTERFAÇAGE D'APPAREILS CONNECTÉS POUR LE SUIVI DE PATIENT·E·S



Thèse de Bachelor présentée par

David LAROCHE

pour l'obtention du titre Bachelor of Science HES-SO en

**Informatique et systèmes de communication avec orientation
Informatique logicielle**

Septembre 2023

Professeur HES responsable

Gregory TROLLIET

En collaboration avec :

Hôpitaux universitaires de Genève

Illustration de couverture : Affichage des mesures cardiovaskulaires dans l'application de suivi de patient à distance.

Source : CARDIOREMOTE, 2023.

TABLE DES MATIÈRES

Remerciements	vi
Liste des acronymes	ix
Liste des illustrations	xi
Liste des tableaux	xiii
Liste des annexes	xiv
Introduction	1
1 Chapitre 1 : Analyse des besoins	3
1.1 Besoins des patients	3
1.2 Besoins des professionnels de santé	3
1.3 Enjeux liés à la pénurie de personnel médical	3
1.4 Interopérabilité des données	4
1.5 Sensibilité des données médicales	4
2 Chapitre 2 : Analyse de l'existant	6
2.1 Tensiomètre connecté	6
a Choix du fabricant	6
b Choix du modèle	7
c Caractéristiques du modèle	8
2.2 La norme FHIR	9
a Définition	9
b Solution HAPI FHIR	10
2.3 Le smartphone comme outil de télésurveillance	11
a Accessibilité et facilité d'utilisation	11
b Transmission efficace des données	11
c Engagement du patient	11
d Personnalisation et adaptabilité	11
3 Chapitre 3 : Architecture	12
3.1 Vue d'ensemble	12
3.2 Back-end : Choix technologiques	13
a Choix du serveur	13
b Langage de programmation et framework	14
c Gestionnaire et dépendances	14
d Base de données	14
e Modèle de conception	14
3.3 Applications clientes : Choix technologiques	15
a Angular et Capacitor	15
b React et React Native	16
3.4 Client mobile	16

a	Choix librairie	16
b	Flux des données	17
3.5	Front-end : flux des données et sécurité	17
4	Chapitre 4 : Conception	18
4.1	Client mobile	18
a	Réception des données Bluetooth	18
b	Interprétation de la charge utile	18
c	Affichage des données	18
d	Authentification	19
4.2	Ressources FHIR	20
a	Définition	20
b	Représentation d'Observation	20
c	Représentation de Patient	20
4.3	Back-end	21
a	Base de données	21
b	L'API REST	23
c	Diagramme de classes	24
4.4	Front-end	25
5	Chapitre 5 : Déploiement	26
5.1	Raspberry Pi	26
a	Client mobile	27
5.2	Conteneurisation avec docker	27
5.3	Cloud Azure	28
6	Chapitre 6 : Résultats	29
6.1	Enregistrement d'un Patient	29
a	Front-end	29
b	Back-end	31
6.2	Relais des mesures médicales	32
a	Tensiomètre connecté	32
b	Client mobile	33
6.3	Affichage des données	34
a	Client mobile	34
b	Front-end	36
Conclusion	39	
Annexes	40	
Références documentaires	43	

REMERCIEMENTS

J'aimerais remercier le docteur Frederic Ehrler ainsi que le professeur HES responsable Gregory Trolliet pour leurs disponibilités et leurs conseils avisés durant le processus de création de ce travail. J'aimerais les remercier de même pour leur confiance en m'offrant la liberté d'accomplir mes objectifs académiques et professionnels.

INTERFAÇAGE D'APPAREILS CONNECTÉS POUR LE SUIVI DE PATIENT·E·S

ORIENTATION : INFORMATIQUE LOGICIELLE

Descriptif :

Dans un monde post-Covid, les suivis médicaux sont de plus en plus demandés et nécessaires alors que les places en milieux hospitaliers manquent et que les déplacements vers les structures sont une contrainte. Pour palier à ces problèmes, ce projet a pour objectif de permettre aux patient·e·s d'utiliser des appareils connectés, tel des capteurs de pression artérielle, afin de reporter des données de signes physiologiques directement au corps médical dans le but d'effectuer un suivi approprié.

Travail demandé :

Dans les limites du temps disponible, les tâches suivantes seront effectuées :

- identifier les appareils appropriés, selon quelques critères tels que l'ouverture de leur code source et matériel, la précision des données récoltées et le prix ;
- mettre en place une architecture permettant la récolte et l'utilisation des données ;
- développer une application sur smartphone permettant la visualisation des données ;
- partager les données de manière confidentielle avec le corps médical ;
- permettre la portabilité du système.

Bien qu'organisé en collaboration avec les HUG, le projet est très flexible, l'étudiant est encouragé à préciser les buts selon ses envies.

Candidat :

LAROCHE DAVID

Filière d'études : ISC

Professeur responsable :

TROLLIET GREGORY

En collaboration avec : HUG

Travail de bachelor soumis à une convention de stage en entreprise : non

Travail de bachelor soumis à un contrat de confidentialité : non

RÉSUMÉ

L'avènement des téléphones intelligents a profondément changé nos sociétés. Leur omniprésence, combinée à leur capacité de connexion, a ouvert la porte à une multitude de solutions innovantes, comme la télésurveillance. Cependant, la sensibilité des données médicales exige de contrôler leur flux. Les Hôpitaux universitaires de Genève cherchent à connaître la faisabilité d'une solution interne. L'objectif principal, avec leur collaboration a donc été de la démontrer. Ainsi la première étape a été d'analyser l'existant pour trouver un tensiomètre connecté dont le protocole est disponible. En effet, il est ainsi possible de s'y connecter directement, évitant le transit des données sensibles sur des solutions cloud proposées par des fabricants. Pour cela, un client mobile a été développé comprenant une fonction double : le transfert des données du tensiomètre et la visualisation de ces dernières sur le même support. Ceci permettra de garder le patient engagé dans la collecte de ses données. Une application web d'architecture classique en trois tiers a également été développée. Elle est composée d'une base de données SQL, d'un serveur API RESTful et d'une interface web pour les professionnels de la santé. Cette application propose la gestion des patients et permet, en particulier, le suivi à distance en affichant l'historique des valeurs automesurées dans leur dossier. Un standard international, définissant les structures et la communication des données médicales, a été implémenté. Il définit l'API du serveur, assurant une interopérabilité avec tout système s'y conformant. Aussi, une connexion sécurisée HTTPS a été ajoutée entre le serveur et l'interface web. Finalement, l'application web a démontré sa portabilité sur le cloud et un micro-ordinateur, grâce à la conteneurisation de ses services.



Candidat :

LAROCHE DAVID

Filière d'études : ISC

Professeur responsable :

TROLLIET GREGORY

En collaboration avec : HUG

Travail de bachelor soumis à une convention de stage en entreprise : non

Travail soumis à un contrat de confidentialité : non

LISTE DES ACRONYMES

- API** Application Programming Interface. 12, 15, 17, 18, 21, 23, 33
- AWS** Amazon Web Service. 28
- B2B** Business-to-Business. 6
- BLE** Bluetooth Low Energy. 16, 17, 18, 33, 39
- CORS** Cross-Origin Resource Sharing. 26
- CRUD** Create, Read, Update, Delete. 14
- FHIR** Fast Healthcare Interoperability Resources. 9, 10, 12, 13, 20, 29, 39, 40
- HL7** Health Level Seven International. 9, 10, 20
- HTTP** Hypertext Transfer Protocol. 9, 10, 26
- HTTPS** Hypertext Transfer Protocol Secure. 17, 26
- HUG** Hôpitaux universitaires de Genève. 1, 7, 15
- ICS** Implementation Conformance Statement. 18
- IoT** Internet of Things. 4
- JAR** Java ARchive. 27
- JDBC** Java Database Connectivity. 21
- JDK** Java Development Kit. 27
- JPA** Java Persistence API. 14, 15, 21
- JRE** Java Runtime Environment. 27
- JSON** JavaScript Object Notation. 9, 17, 20, 29, 36
- LOINC** Logical Observation Identifiers Names and Codes. 20
- ORM** Object Relational Mapping. 14
- OS** Operating System. 16

PKCS Public-Key Cryptography Standards. 27

POC Proof Of Concept. 4

POO Programmation Orientée Objet. 39

RDF Resource Description Framework. 9

REST Representational State Transfer. 9, 12, 17, 23, 33

SGBD Système de Gestion de Base de Données. 12, 14, 21

SQL Structured Query Language. 14, 31

SSL Secure Sockets Layer. 27

TLS Transport Layer Security. 27

UUID Universally Unique Identifier. 18

VM Virtual Machine. 28

XML Extensible Markup Language. 9

LISTE DES ILLUSTRATIONS

1.1	Croissance annuelle composée des données	4
2.1	Diagramme Truc	7
2.2	Carte HAPI	9
2.3	Capture d'écran	10
3.1	Diagramme d'architecture.	12
3.2	Diagramme d'architecture.	13
3.3	Architecture en couches	15
4.1	Diagramme de séquence :	19
4.2	ERD	22
4.3	Diagramme EA	24
4.4	Diagramme de séquence : Front-end	25
5.1	Micro-ordinateur Raspberry Pi	26
5.2	Ressources Azure	28
6.1	Formulaire Patient	29
6.2	Requête formulaire	30
6.3	Réponse formulaire	30
6.4	Terminal Hibernate	31
6.5	Mesures de pression artérielle et rythme cardiaque	32
6.6	Debuggage Mobile	33
6.7	Données Application Mobile	34
6.8	Application Mobile graphe	35
6.9	Affichage dossier patient	36
6.10	Réponse dossier patient	36
6.11	Requête de recherche d'observations	37
6.12	Affichage données patient	37
6.13	Réponse requête observation	38

Références des URL

LISTE DES TABLEAUX

2.1	Lot de données n°2	6
4.2	API Patient.	23
4.3	API Observation.	23
5.1	Caractéristiques Raspberry Pi	27
5.2	Configuration de la VM.	28
6.1	Caractéristiques du tensiomètre	42

LISTE DES ANNEXES

Annexe 1	42
-----------------	-------	----

INTRODUCTION

La médecine a toujours su tirer avantage des progrès technologiques afin d'améliorer la qualité de ses soins. Aujourd'hui, la généralisation des smartphones dans nos sociétés offre de nouvelles opportunités pour le système médical. En effet, leur omniprésence bénéficie d'un contact privilégié avec l'utilisateur. Combiné à ses capacités de connexion à d'autres appareils et d'une puissance de calcul en constante augmentation, ils génèrent un volume important de données toutes aussi précieuses. Ainsi, la médecine tente d'en tirer le meilleur parti en développant de nouvelles approches comme la télésurveillance. Cependant, la nature sensible des données médicales limite l'utilisation de solutions déjà existantes, incitant les institutions de santé à concevoir leurs propres systèmes avec un meilleur contrôle du flux de données. De ce constat, les Hôpitaux universitaires de Genève (HUG) cherchent à développer une application permettant de relayer des mesures physiologiques au moyen de capteurs connectés, afin de suivre au mieux les patients en adaptant leur prise en charge. Il sera donc question de démontrer la faisabilité d'une interprétation à distance des valeurs mesurées d'un tensiomètre connecté.

Un de mes objectifs personnels était la diversification de mes compétences. Le développement d'une application full-stack est donc l'opportunité idéale pour étoffer mes connaissances encore modestes en développement web et pour m'initier à de nouveaux outils employés dans l'industrie. Par ailleurs, le domaine médical a toujours suscité mon intérêt.

Pour arriver à ces objectifs, j'ai d'abords suivi les conseils de prioriser les recherches de l'appareil médical. J'ai été mis en contact avec le Docteur Ehrler et son équipe de développeurs, avec lesquels j'ai pu échanger l'objet de mes recherches et découvrir les technologies de développement qu'ils utilisent. Après nous être entendu sur le modèle du dispositif médical et d'avoir défini plus clairement l'architecture du système à mettre en place, l'objectif a été de démontrer sa faisabilité le plus rapidement possible. Pour cela, j'ai utilisé des solutions existantes et choisi entre autres d'apprendre des technologies avec une courbe d'apprentissage moins abrupte. Après démonstration du concept au Dr Ehrler et au Dr Blondon, j'ai pu bénéficier d'une plus grande flexibilité. Je l'ai ainsi mise au service du développement de mes compétences, en concevant mes propres solutions et en améliorant celles déjà celle déjà mises en place en ajoutant, entre autres, des fonctionnalités demandées par le docteur Blondon. Mes ressources principales étaient les sites internet officiels des technologies utilisées, quelques tu-

toriels pour celles dont je n'avais pas de connaissance préalable et la documentation privée du fabricant pour la communication avec le dispositif médical.

Cette thèse a été structurée de la sorte : d'une analyse des besoins et d'une analyse de l'existant afin de définir le périmètre de ce travail et ne pas répéter ce qui a été conçu auparavant, de l'architecture, justifiant les choix technologiques des composants du système et de leur agencement à l'intérieur de celui-ci. De la conception, expliquant cette fois le fonctionnement même de chacun des composants. D'un chapitre de déploiement, décrivant les diverses étapes pour y parvenir. Et finalement les résultats, prouvant le fonctionnement de notre système.

CHAPITRE 1 : ANALYSE DES BESOINS

1.1. BESOINS DES PATIENTS

Dans le paysage médical actuel, la télésurveillance répond à une variété de besoins essentiels à la fois pour les patients que les professionnels de santé. Pour ces premiers elle offre aux personnes éloignées des services de santé ou ayant des problèmes de locomotion de pouvoir bénéficier de services à distance. Aussi, elle permet de développer la philosophie d'approche centrée sur le patient. En effet, en mesurant lui-même ses données physiologiques et en lui les rendant accessibles au moyen d'une application mobile, cela l'encourage à participer activement à ses soins. Les effets positifs de la démarche sur sa satisfaction et son parcours de soins sont aujourd'hui bien reconnus¹.

1.2. BESOINS DES PROFESSIONNELS DE SANTÉ

L'accès de ces données aux médecins permet une analyse quotidienne de la tension et donc un suivi plus régulier de la tension artérielle, essentiel pour détecter des tendances et des variations. Cette surveillance constante facilite également une prise de décision plus rapide, car les professionnels de santé peuvent identifier et répondre aux anomalies plus efficacement. En outre, ces plates-formes sont souvent bénéfiques pour la collecte et l'analyse de données à long terme, des ressources précieuses pour la recherche médicale et l'amélioration des traitements.

Un autre avantage est de se soustraire du Syndrome de la Blouse Blanche, où un patient peut avoir une tension anormalement haute à cause du stress engendré par la présence du médecin². Une automesure générerait donc des résultats plus exacts en éliminant ce biais.

1.3. ENJEUX LIÉS À LA PÉNURIE DE PERSONNEL MÉDICAL

Face à une pénurie de personnel médical prévue, l'optimisation du temps et la réduction du nombre de visites inutiles sont d'autant plus cruciales. Une étude récente de PwC a alerté sur une aggravation importante de la pénurie de personnel dans le secteur de la santé suisse, prévoyant un déficit de 40 000 infirmières et infirmiers, ainsi que 5 500 médecins d'ici 2040³. Dans ce contexte, des solutions innovantes comme la télésurveillance pourraient jouer un rôle afin d'alléger le fardeau des professionnels de santé.

1.4. INTEROPÉRABILITÉ DES DONNÉES

Les systèmes de santé en Suisse n'échappent pas au cloisonnement de leurs données et à leur manque d'uniformisation. Un exemple pendant le COVID-19 était la difficulté de suivre l'évolution de la pandémie, car le fax était employé afin de communiquer des cas d'infections aux autorités fédérales⁴.

De plus, environ 30 % du volume de données mondial est généré par l'industrie de la santé. D'ici 2025, le taux de croissance annuel composé des données pour la santé atteindra 36 %⁵.

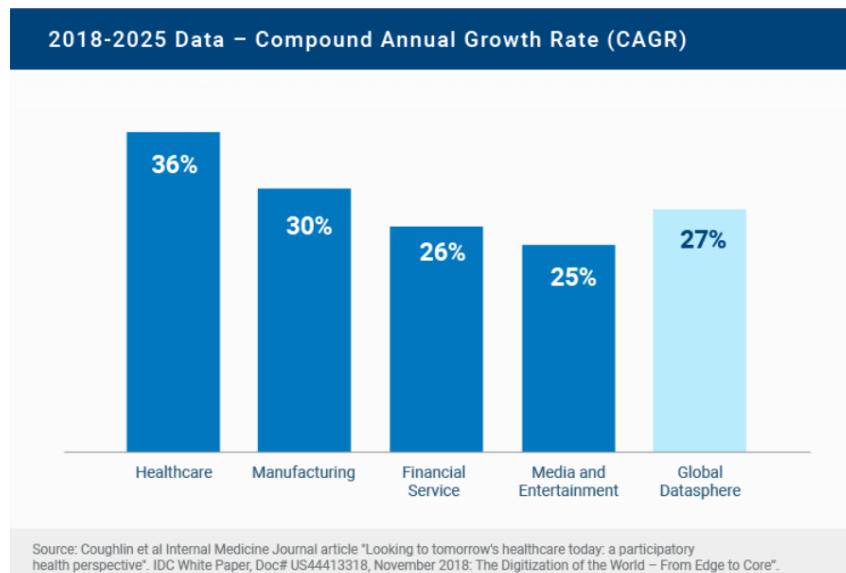


ILLUSTRATION 1.1 – *Croissance annuelle composée des données. Source : rbccm.com .*

Cette augmentation dramatique est due en partie à la généralisation de l'Internet of Things (IoT)⁶, dont le développement de ce Proof Of Concept (POC) qui illustre cette tendance, démontre la nécessité d'évoluer vers des systèmes interopérables. En effet, des économies de temps sont réalisées en rendant les données plus facilement accessibles aux intervenants du système médical et des erreurs pourraient être évitées grâce à des données de qualité. Finalement, cela faciliterait et accélérerait le développement de solutions technologiques au bénéfice du patient.

1.5. SENSIBILITÉ DES DONNÉES MÉDICALES

Bien que des solutions de télémédecine soient déjà disponibles sur le marché, leur adoption est entravée par les préoccupations concernant la sensibilité des données médicales. La gestion sécurisée et la protection de ces données sont primordiales et toute solution envisagée doit

garantir la confidentialité et la sécurité des informations des patients. Sans ces garanties, même les systèmes existants les plus sophistiqués seront ignorés en raison des risques pour la vie privée des patients.

CHAPITRE 2 : ANALYSE DE L'EXISTANT

2.1. TENSIOMÈTRE CONNECTÉ

a. Choix du fabricant

De nombreux appareils mesurant la tension artérielle existent sur le marché, selon mes recherches, plus de 300 sur une plateforme de vente en ligne suisse. Le premier critère est la récupération des données sans fil, ce qui réduit considérablement les choix.

Un second critère important est la maîtrise des flux de données, les données médicales étant particulièrement sensibles, les solutions de fabricants comme iHealth ont été écartées, car elles nécessitent que les données transitent par un cloud tiers. Les produits Omron n'ont pas non plus été retenus, ils proposent un kit de développement pour Android et iOS uniquement dans le cadre d'un échange **Business-to-Business (B2B)**. Nous n'avons donc reçu aucune réponse après notre demande d'acquisition du kit.

Une alternative envisagée est l'utilisation d'un micro-ordinateur Raspberry Pi. Du matériel ainsi que des codes open source existent afin de fabriquer soi-même des appareils médicaux. Bien que le contrôle des données soit total et offre ainsi une liberté accrue en matière de développement, il ne répond pas aux exigences de sûreté et de précision d'un dispositif médical soumis aux normes. Voici une sélection de fabricants majeurs et des critères d'exclusion :

Fabricant	Maîtrise des données	Dispositif medical
iHealth	non ¹	oui
A&D Medical	oui ²	oui
Beurer	oui ²	oui
Omron	inconnu	oui
Raspberry Pi DIY	oui ³	non

TABLEAU 2.1 – ¹ Données sur un cloud tiers. ² Accès au protocole de communication BLE. ³ Logiciels open source.

Notre choix s'est tourné vers deux autres fabricants, A&D Medical et Beurer. Tous deux proposent un accès aux protocoles de communications de leurs appareils médicaux, en échange d'informations et des objectifs de l'organisation voulant s'en servir. Dans le cas d'A&D, l'obtention d'un protocole fut plus laborieuse, nécessitant des échanges avec le vendeur, contrairement à Beurer où après validation automatique du formulaire de demande, nous avons obtenu

tous les protocoles de la gamme du produit. Notre choix s'est porté sur ce dernier, car de plus, il existe un large choix d'appareils rapidement disponibles à la vente.

b. Choix du modèle

Nous avons privilégié un tensiomètre à bras plutôt qu'au poignet. Cette méthode est non seulement considérée comme plus précise et elle permet également une meilleure comparaison des mesures. En effet, en milieu hospitalier, la tension est le plus souvent prise au bras.

Il doit aussi disposer d'une mémoire suffisante afin d'éviter la perte de donnée dans le cas où leur synchronisation serait impossible (réseau, smartphone ou application indisponible par exemple). Le coût est aussi un critère important, en prévision d'une généralisation du concept pour des établissements tels que les HUG, en conséquence l'acquisition en grande quantité de tensiomètres leur engendrerait des coûts significatifs.

Ainsi nous avons choisi le modèle BM54 du fabricant Beurer. Son coût de 50 CHF est un montant se situant dans la fourchette basse pour un appareil offrant ces fonctionnalités.



ILLUSTRATION 2.1 – Dispositif choisi (modèle BM54).

c. Caractéristiques du modèle

L'appareil propose deux profils de 60 mesures chacune. Avec une fréquence de trois mesures quotidiennes, cela correspond à vingt jours par profil sans écraser les mesures précédentes.

L'autonomie de l'appareil alimenté par des piles est en moyenne de 200 mesures, donc environ deux mois pour un profil avec la même fréquence de mesures. Cette autonomie pourrait-être un facteur limitant lors d'un suivi patient sur plusieurs mois, sans compter l'impact environnemental. En cas d'application du concept, il serait judicieux d'envisager un modèle à batterie rechargeable, même si cela implique un coût de départ plus élevé.

L'appareil répond aux normes de dispositif médical suivantes :

- EN60601-1-2 : Garantis que l'appareil ne causera pas d'interférences électromagnétiques et ne sera pas non plus affecté par elles.
- Directive européenne 93/42/EEC sur la sécurité et de performance pour les dispositifs médicaux en Europe.
- EN1060-1 et EN1060-3 : Exigences pour les tensiomètres.
- IEC 80601-2-30 : Exigences sur la sécurité et la performance de tensiomètres non invasifs automatiques.
- Directive européenne RED 2014/53/UE : Garantis la sûreté d'un équipement radio, en utilisant efficacement le spectre sans causer d'interférences.

Le mode d'emploi mentionne que "La précision de ce tensiomètre a été correctement testée et sa durabilité a été conçue en vue d'une utilisation à long terme. Dans le cadre d'une utilisation médicale de l'appareil, des contrôles techniques de mesure doivent être menés avec les moyens appropriés. Pour obtenir des informations précises sur la vérification de la précision de l'appareil, tu peux faire une demande par courrier au service après-vente."

L'appareil présente une précision de ± 3 [mmHg] pour la mesure de la tension artérielle et une marge d'erreur de 5 % pour le pouls par rapport à la valeur affichée. Selon mes recherches sur un site de vente en ligne, 92 % affichent cette même précision pour la tension et 86 % pour le pouls. Il présente donc des caractéristiques similaires à la majorité des tensiomètres sur le marché, leur totalité est détaillée dans l'Annexe 1.

2.2. LA NORME FHIR

a. Définition

La croissance du nombre de données médicales et leur diversité rendent leur gestion de plus en plus complexe, engendrant coûts et gaspillage de temps. En réponse à ce constat, l'équipe Health Level Seven International (HL7) a développé le standard Fast Healthcare Interoperability Resources (FHIR) dans l'objectif de faciliter l'interopérabilité des systèmes informatiques de santé. Ce standard définit les modèles de représentation des données JavaScript Object Notation

THE HAPI FHIR GLOBAL ATLAS



ILLUSTRATION 2.2 – Carte HAPI, source : hapifhir.io

(JSON), Extensible Markup Language (XML) ou Resource Description Framework (RDF), classés en différentes catégories telles que "patient" ou "observation".

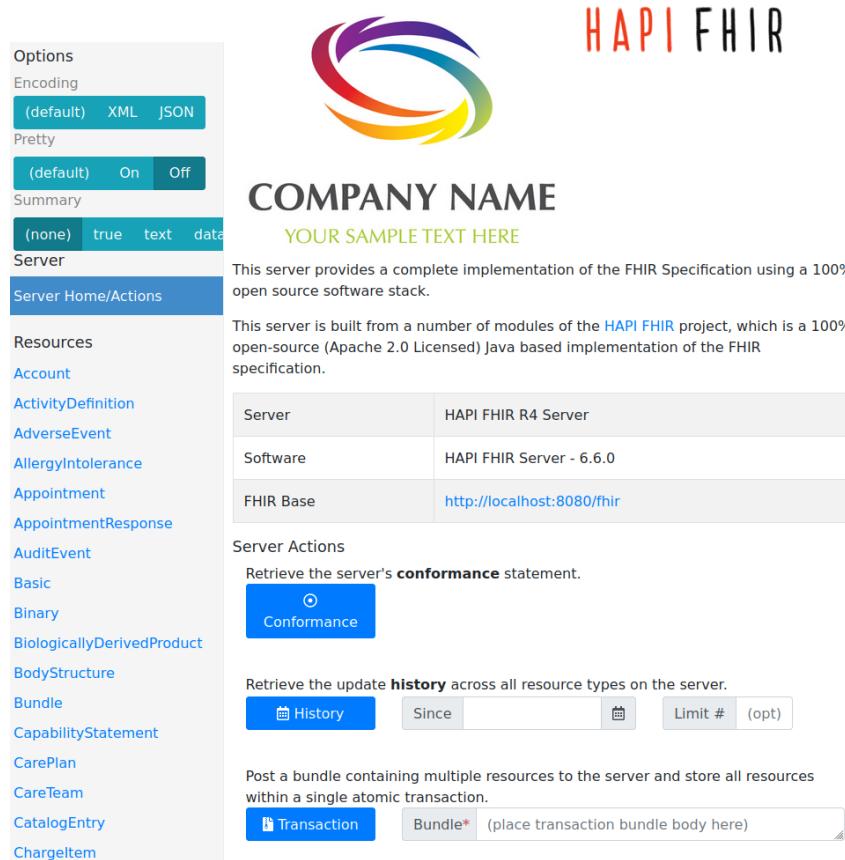
Il définit des protocoles d'échanges Representational State Transfer (REST), rendant les systèmes interopérables sur Internet au travers de requêtes Hypertext Transfer Protocol (HTTP).

FHIR ne se contente pas de faciliter l'interopérabilité entre différents systèmes, il assure également un accès plus rapide aux données de santé. Ainsi, dans le cadre de la télésurveillance médicale, la conformité aux normes telles que FHIR est essentielle non seulement pour garantir des échanges d'information sécurisés et efficaces, mais aussi pour répondre aux besoins actuels et futurs d'accès à des volumes croissants de données.

b. Solution HAPI FHIR

Un serveur FHIR open source a été identifié lors de l'analyse. **HAPI FHIR** est donc une implémentation complète de **HL7 FHIR**. Débuté à l'Université Health Network en 2001, ce projet est aujourd'hui mature et est utilisé par diverses organisations à travers le monde.

Grâce à cette solution, nous avons pu confirmer rapidement la faisabilité en nous focalisant sur les composants clients. Elle a également facilité leur développement conformément aux normes **FHIR**, étant donné que seules les requêtes **HTTP** répondant à ces normes peuvent interagir avec le serveur HAPI. De plus, ce serveur dispose d'une interface web qui nous permet de vérifier l'état des requêtes, de confirmer les enregistrements et de comprendre leur format. Cet outil nous a permis d'effectuer des tests de communication et s'est donc avéré précieux pour notre compréhension du standard **FHIR** pour son implémentation.



The screenshot shows the HAPI FHIR RESTful API interface. At the top left, there's a sidebar with configuration options: Encoding (XML, JSON), Pretty (On), Summary (true), and Server Home/Actions. The main content area has a large logo for "HAPI FHIR" with a colorful circular swoosh graphic. Below the logo, it says "COMPANY NAME" and "YOUR SAMPLE TEXT HERE". A text block states: "This server provides a complete implementation of the FHIR Specification using a 100% open source software stack." Another text block says: "This server is built from a number of modules of the HAPI FHIR project, which is a 100% open-source (Apache 2.0 Licensed) Java based implementation of the FHIR specification." To the right, there's a table showing server details: Server (HAPI FHIR R4 Server), Software (HAPI FHIR Server - 6.6.0), and FHIR Base (<http://localhost:8080/fhir>). Below the table, under "Server Actions", there's a "Conformance" button. Further down, there's a "History" button with filters for "Since" and "Limit # (opt)". At the bottom, there's a "Transaction" button with a placeholder for "Bundle* (place transaction bundle body here)".

ILLUSTRATION 2.3 – Façade offerte par le serveur HAPI

2.3. LE SMARTPHONE COMME OUTIL DE TÉLÉSURVEILLANCE

Selon une étude de Deloitte en 2018, 92% des adultes en Suisse possèdent un smartphone⁷. Son omniprésence dans la vie des suisses en font le candidat idéal permettant la collecte des données de santé mesurées directement par le patient.

a. Accessibilité et facilité d'utilisation

Cette universalité et cette commodité réduisent la complexité pour les patients : il n'y a pas de nouveau matériel à manipuler, ni de risque de perte ou de dysfonctionnement. De plus, cela évite aux systèmes de santé des coûts supplémentaires d'acquisition de matériel spécifique.

b. Transmission efficace des données

Les smartphones actuels, équipés de capacités avancées de connectivité, permettent une transmission fiable et rapide des informations du tensiomètre vers le système de surveillance. La connectivité Bluetooth, par exemple, facilite la communication entre l'appareil et le téléphone, garantissant ainsi que les données soient transmises en temps réel et sans perte.

c. Engagement du patient

Une application mobile offre non seulement la possibilité de collecter et de transmettre des données, mais aussi de les visualiser. En permettant aux patients de voir leurs propres résultats et de suivre l'évolution de leurs indicateurs de santé, ils sont davantage encouragés à rester impliqués et engagés dans le processus de collecte de données. Ceci est important car l'engagement du patient est crucial pour assurer la continuité et la qualité des informations transmises.

d. Personnalisation et adaptabilité

Les applications mobiles offrent également un niveau élevé de personnalisation. Que ce soit en termes d'interface utilisateur, de rappels pour la prise de mesures, ou d'alertes basées sur les résultats, l'application peut être adaptée aux besoins spécifiques de chaque patient, offrant ainsi une expérience utilisateur personnalisée.

CHAPITRE 3 : ARCHITECTURE

3.1. VUE D'ENSEMBLE

Pour l'application web, une base architecturale classique 3-tiers a été choisie afin de répondre aux exigences du projet. Ce modèle est couramment utilisé pour les applications web grâce entre autres à sa flexibilité, sa scalabilité et sa testabilité. Il bénéficie donc d'un couplage faible avec ses composants. Cette popularité a engendré de nombreux outils et de documentation que nous emploierons, facilitant ainsi une mise en œuvre avec fiabilité. Sur cette base solide, nous lui ajoutons le client mobile, voici le schéma ci-dessous, acteurs inclus :

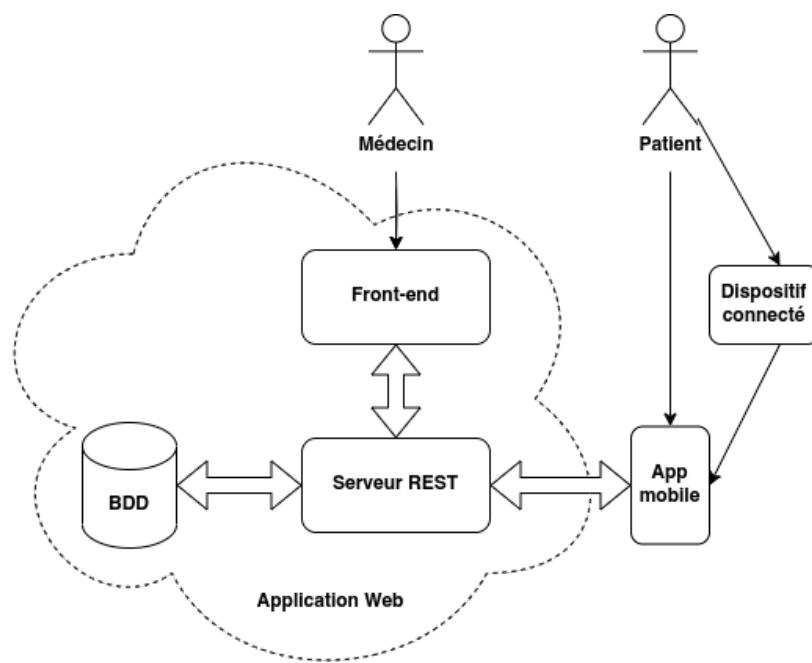


ILLUSTRATION 3.1 – Diagramme d'architecture de notre application.

Notre système entier est constitué des quatre composants suivants :

- un serveur comprenant une Application Programming Interface (API) REST basé sur les recommandations FHIR, chargé de la communication bidirectionnelle des données avec les interfaces clients (mobile et front-end)
- une Système de Gestion de Base de Données (SGBD) pour la persistance des données, communiquant avec le serveur REST.
- une application mobile récupérant les données du tensiomètre connecté et les synchronisant avec le back-end grâce à son API. Elle peut l'interroger à nouveau afin d'afficher

les données nouvellement récupérées.

- un front-end représentant ces données au corps médical afin qu'ils puissent suivre leurs patients à distance.

3.2. BACK-END : CHOIX TECHNOLOGIQUES

a. Choix du serveur

Le standard **FHIR** a été adopté selon les conseils du Dr Ehrler. Comme évoqué dans l'analyse de l'existant, nous avons initialement opté pour le serveur HAPI open source. De ce fait, nous avons pu à la fois garantir la conformité des échanges de données avec les composants clients, mais aussi démontrer plus rapidement la faisabilité de l'application. En l'occurrence, ce serveur dispose d'une interface web qui nous permet de vérifier l'état des requêtes, de confirmer les enregistrements et d'en comprendre la structure.

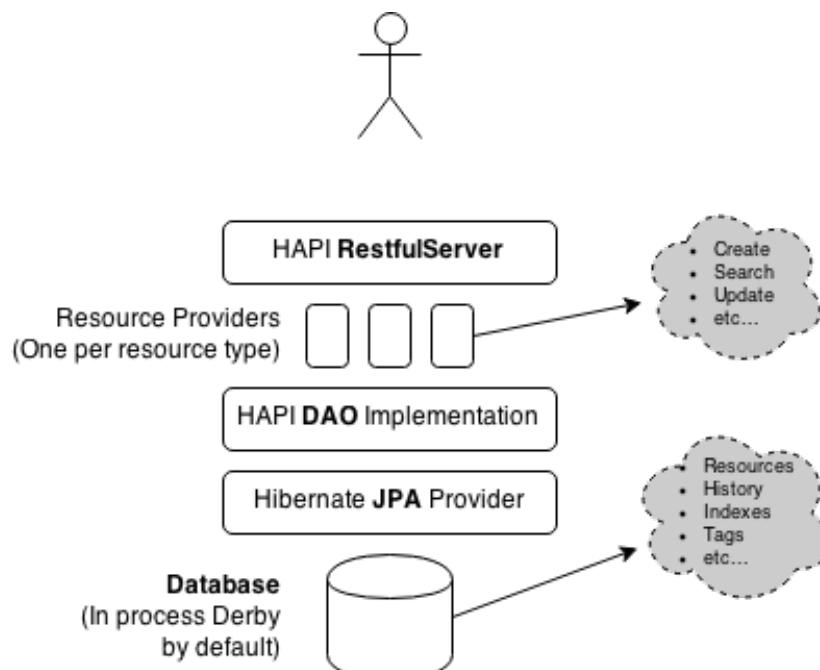


ILLUSTRATION 3.2 – Architecture du serveur HAPI, source : hapifhir.io

Des tentatives de modifications du serveur ou des squelettes proposés par HAPI ont été faites à des fins de personnalisation et de compréhension. Chronophage et peu concluante, la décision de démarrer un nouveau projet Spring Boot pour le back-end a été prise. Il comprend les technologies recommandées par le Dr Ehrler et son équipe, elles sont détaillées ci-dessous.

b. Language de programmation et framework

Les recommandations pour le langage Java et son framework Spring Boot ont été suivies. Ces outils accélèrent et facilitent la mise en production d'application stand-alone, en épargnant entre autres de multiples configurations laborieuses. Nous verrons cela dans le chapitre suivant.

c. Gestionnaire et dépendances

Le gestionnaire de projet Maven a été recommandé, il télécharge automatiquement les dépendances spécifiées et en simplifie la gestion. Il permet la compilation du projet en un fichier exécutable facilitant ainsi son déploiement.

- Java Persistence API (JPA) Est une spécification Java pour l'Object Relational Mapping (ORM), implémentée par Hibernate. En d'autres termes, JPA il définit la conversion d'objets Java en entrées de tables Structured Query Language (SQL) et réciproquement. Toute classe Java a donc la capacité d'avoir sa représentation correspondante sous forme de table dans la base de données. Nous pouvons ainsi nous soustraire de l'écriture de requêtes SQL et gérer les opérations Create, Read, Update, Delete (CRUD) avec Java. Ceci accélère l'implémentation de la base de données tout en améliorant la robustesse et la modularité du système. Un avantage supplémentaire est de pouvoir interchanger aisément de SGBD.
- Pour finir, Apache Tomcat est un choix courant pour les serveurs web. Bien qu'il soit modeste en fonctionnalités, il bénéficie d'une rapidité et d'une légèreté. Notre serveur ne nécessitant pas davantage de fonctionnalités, il peut ainsi bénéficier de cette légèreté.

d. Base de données

Les SGBD PostgreSQL et MongoDB ont été recommandés pour ce projet. Tandis que PostgreSQL est un système relationnel, MongoDB est orienté documents. De plus, PostgreSQL est activement supporté pour son fonctionnement avec le serveur HAPI. Comme il avait déjà été installé dans ce cadre, il a semblé naturel de conserver ce système lors de la transition vers notre serveur personnalisé, qui a nécessité un minimum d'ajustements.

e. Modèle de conception

Un modèle en trois couches a été sélectionné, il est défini par les couches :

- Représentation : comprenant les contrôleurs et gérant les routes vers l'**API**
- Logique métier : où résident les calculs, validations, etc.
- Accès aux données : interagissant avec la base de données grâce à **JPA** et **Hibernate**.

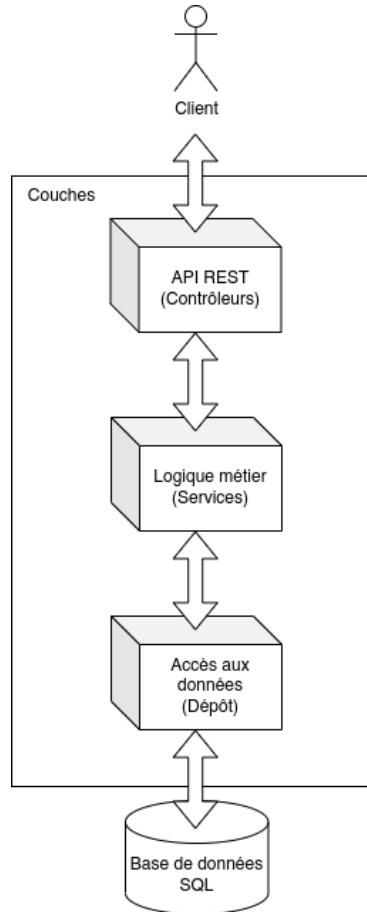


ILLUSTRATION 3.3 – *Architecte trois couches du back-end*

Ce modèle est recommandé pour le développement d'applications web, car il octroie une simple et claire séparation des fonctions améliorant modularité, flexibilité et maintenance.

3.3. APPLICATIONS CLIENTES : CHOIX TECHNOLOGIQUES

a. Angular et Capacitor

Angular et Capacitor ont été initialement proposés comme framework web et plateforme open source respectivement, pour le développement du front-end et de l'application mobile. Ils offrent la possibilité de rédiger un code unique adapté à la fois pour le web et les plateformes mobiles comme iOS ou Android. Ils sont donc des outils puissants et sont employés par les développeurs professionnels des **HUG**. Toutefois, en raison de nos exigences spécifiques (no-

tamment la communication entre l'application mobile et l'appareil connecté), ces outils ne sont pas totalement adaptés à nos besoins.

b. React et React Native

L'un de mes objectifs académiques était de développer mes compétences modestes en JavaScript et en développement web. Ne maîtrisant aucun framework ou librairie web, React était un choix judicieux. Contrairement à d'autres solutions comme Angular qui adopte TypeScript par défaut, React offre une flexibilité dans l'apprentissage des langages web, permettant l'adoption facultative d'extensions syntaxiques telles que JSX ou TypeScript. Sa nature modulaire offre aussi la possibilité d'une évolution progressive de la complexité. Ces avantages accélèrent sa maîtrise et permettent d'arriver à des résultats plus rapidement. Il jouit aussi d'une meilleure popularité, le maîtriser est donc un avantage dans le contexte professionnel.

Quant à React Native, il est la déclinaison mobile de React répondant au besoin de créer une application multi-[Operating System \(OS\)](#) mobile, tout en offrant une expérience utilisateur native. Cette correspondance m'a permis d'approfondir mes compétences en suivant les mêmes principes. Mais aussi d'harmoniser les fonctions communes aux parties clientes, ce qui permet une compréhension plus rapide du code, un gain de temps de développement et d'une amélioration de la modularité.

3.4. CLIENT MOBILE

a. Choix librairie

Afin de communiquer avec le dispositif médical, en Bluetooth Low Energy (BLE) nous utilisons la librairie React Native [react-native-ble-manager](#). Elle possède diverses méthodes multi [OS](#) mobiles, afin de gérer les connexions. Des différences subsistent tout de même entre les deux [OS](#). C'est le cas des autorisations, qui seront détaillées plus tard dans un diagramme de séquence. Mais certaines fonctions ne sont pas communes ou sinon ce sont certains paramètres, tout cela est décrit dans le dépôt Github¹¹. Le concurrent direct est [react-native-ble-plx](#), cependant la dernière version date de 2021, préférant utiliser une librairie maintenue nous avons de ce fait écarté ce choix.

b. Flux des données

L'application mobile est le composant le plus complexe de notre système, car il communique avec le tensiomètre grâce à la librairie **BLE**, mais transfère aussi dans le même temps ces données aux serveur **REST** grâce à la méthode POST. Enfin pour afficher l'historique des mesures du patient, il utilise la requête GET sur ce serveur.

3.5. FRONT-END : FLUX DES DONNÉES ET SÉCURITÉ

Ce composant client communique avec le serveur au moyen de son **API**. Elle permet d'envoyer des ressources **Patient** au format **JSON** via la requête **POST**, mais aussi de les récupérer ainsi que leurs **Observation** qui leur sont associées avec la méthode **GET**. Le diagramme de séquence système suivant décrit les opérations permettant la création d'un patient ou l'affichage des données d'un patient enregistré.

Une connexion sécurisée **Hypertext Transfer Protocol Secure (HTTPS)** a été mise en place à la fois pour l'accès au client et pour la communication entre le client et le serveur.

CHAPITRE 4 : CONCEPTION

4.1. CLIENT MOBILE

a. Réception des données Bluetooth

Afin de récupérer les données du tensiomètre, l'application mobile doit s'y connecter avec le protocole **BLE**. Les protocoles de Beurer et d'A&D medical, bien que propriétaires, sont conformes à l'**Implementation Conformance Statement (ICS)** Proforma Bluetooth® et respectent l'assignement des nombres⁸.

En résumé, lorsque le tensiomètre est en mode publicitaire, il émet ses données relatives comme son nom et l'**Universally Unique Identifier (UUID)** du service "Pression artérielle". Sa partie unique est **0x1810** et sa partie constante est **0x1000-8000-00805f9b34fb**, attestant de sa conformité au protocole. Notre application en mode découverte, si elle repère ces paquets, émet une requête d'abonnement. Celle-ci spécifie ce même **UUID** ainsi que la caractéristique "Blood Pressure Measurement" dont la partie unique est **0x2a35**. En cas d'acceptation, le tensiomètre communique de façon successive la totalité des mesures sauvegardées.

b. Interprétation de la charge utile

Ces données brutes incluent, entre autres, la pression artérielle (systolique et diastolique), la fréquence cardiaque, l'horodatage de ces mesures et le profil utilisateur qui leur sont associés. Le protocole de l'appareil BM54 nous permet de comprendre ces données en décrivant de quelle manière le message est découpé et dans quel ordre interpréter les octets. Il est similaire au protocole d'A&D Medical, mais par respect pour leur confidentialité, nous n'entrerons pas dans les détails.

c. Affichage des données

La partie mobile offre une visualisation des données, encourageant ainsi l'engagement du patient dans le processus de collecte des données. Des boutons permettent ainsi de les afficher sous forme d'un tableau ou d'un diagramme à barres. Leur activation déclenche la méthode **GET** par l'intermédiaire de l'**API** du serveur, permettant de récupérer toutes les **Observations** attribuées au **Patient**.

d. Authentification

Avant d'accéder aux fonctionnalités, l'utilisateur doit s'authentifier. Bien que l'identifiant soit utilisé dans le cadre de l'application, le système d'authentification actuel est une maquette et nécessite donc d'être implémentée.

Voici ci-dessous le diagramme de séquence figurant les étapes expliquées précédemment :

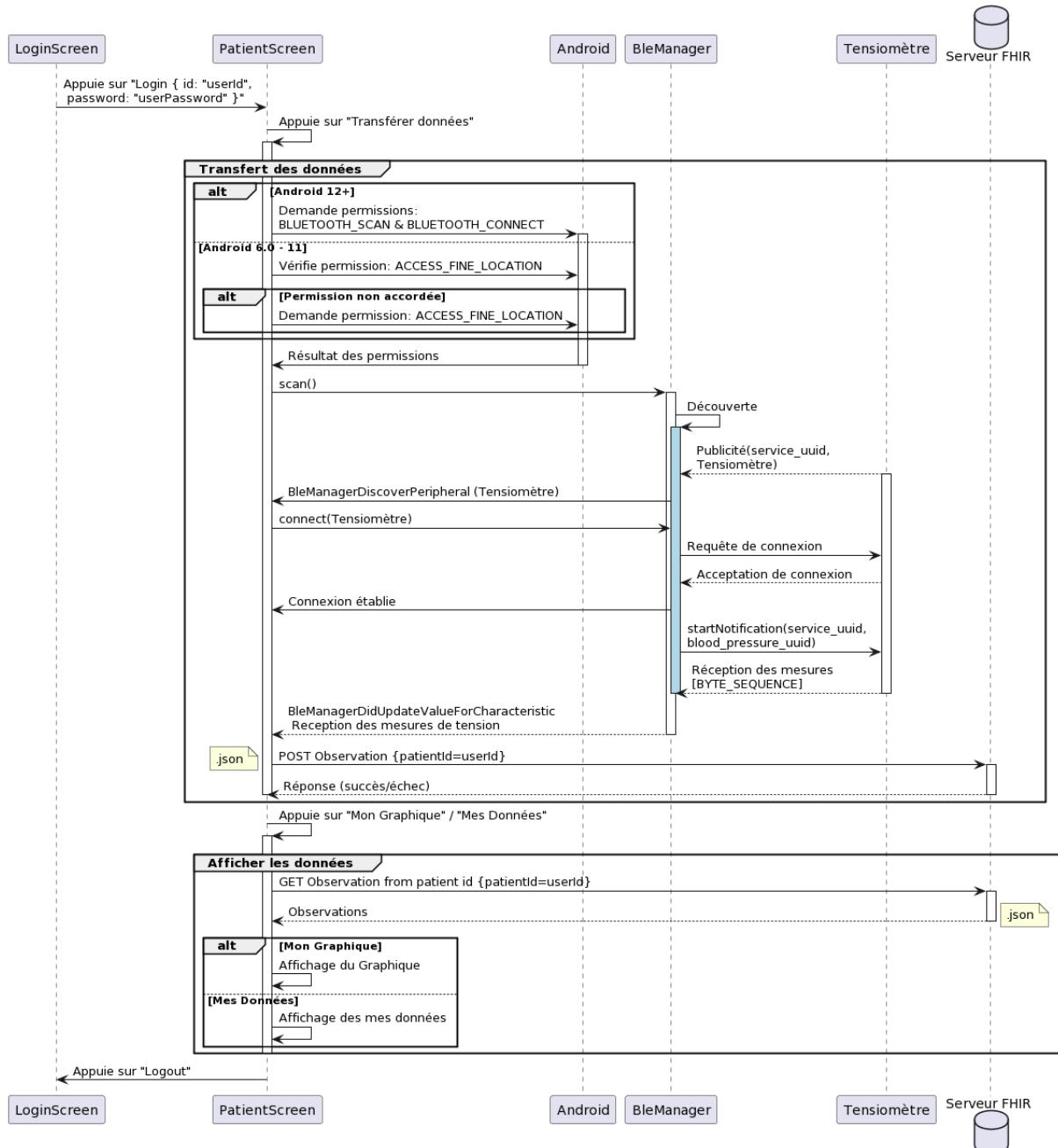


ILLUSTRATION 4.1 – Synchronisation et affichage des données avec le tensiomètre via l'application mobile

4.2. RESSOURCES FHIR

a. Définition

Une Resource selon [HL7](#), est une entité avec une identité spécifique, associée à un type de ressource, contenant des données structurées et un identifiant de version modifiable⁹.

Dans notre projet, nous nécessitons seulement des ressources de type Patient et Observation. La première comprend les informations sur le patient (nom, date de naissance...) et la seconde contient les mesures de santé (tension artérielle et fréquence cardiaque) relatives à un patient.

b. Représentation d'Observation

[Logical Observation Identifiers Names and Codes \(LOINC\)](#) Est un standard international permettant d'identifier les mesures, observations et documents des systèmes de santé¹⁰. HL7 Emploie ces codes d'identification dans ses exemples de structures de données, mais tout autre système de codification peut être utilisé à la place. Voici la codification du type de mesures du tensiomètre :

Mesure	LOINC	Unités
Pression systolique	8480-6	mmHg
Pression diastolique	8462-4	mmHg
Fréquence cardiaque	8867-4	beats/minute

TABLEAU 4.1 – *Codes LOINC des mesures du tensiomètre*

Ces dernières sont incluses dans la catégorie `vital-signs` 85353-1 et la sous-catégorie `Blood pressure panel with all children optional` 85354-9 pour les données de la pression artérielle.

L'ajout de l'identifiant du sujet des mesures et de leur horodatage nous permet de finaliser la représentation de notre Observation au format [JSON](#) en respectant le standard [FHIR](#).

c. Représentation de Patient

Cette ressource comprend les valeurs entrées dans le front-end pour l'enregistrement d'un Patient, en respectant la structure imposée par [FHIR](#).

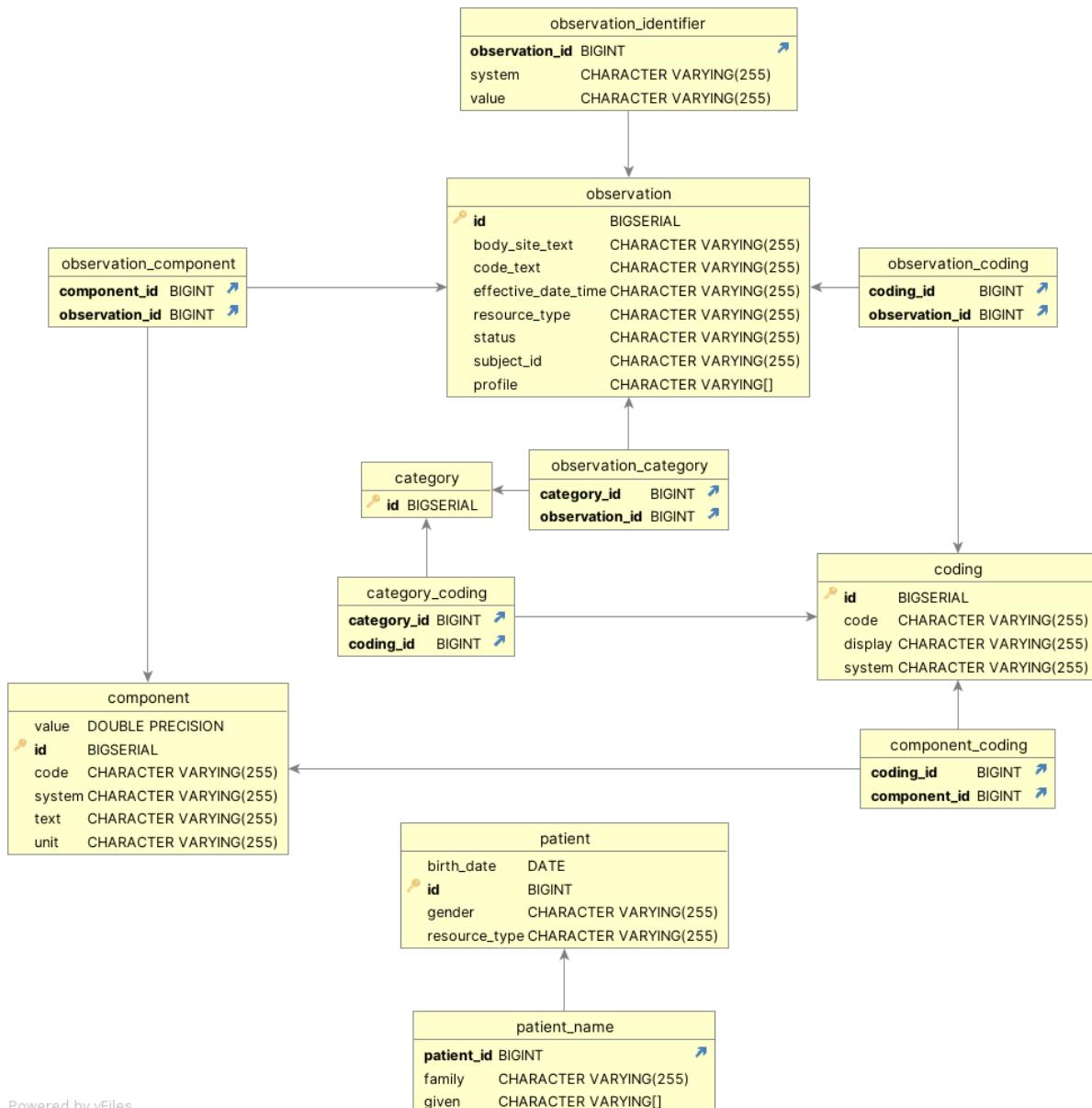
4.3. BACK-END

a. Base de données

La mise en place d'une base de données PostgreSQL implique l'installation du **SGBD**, la création d'une base de données et d'un utilisateur avec des droits d'accès. Ces détails de connexions sont renseignés dans le fichier de configuration Spring Boot application.properties pour l'**API Java Database Connectivity (JDBC)**. Spring boot en automatise la configuration permettant la gestion de la base de données avec **JDBC**.

Hibernate comme expliqué dans le chapitre précédent, génère les tables correspondantes aux classes Java indiquées par l'annotation **@Entity**, grâce au module Spring Data **JPA**.

Voici le diagramme Entité-Relation de notre base de données PostgreSQL :



Powered by yFiles

ILLUSTRATION 4.2 – Diagramme Entité-Relation.

b. L'API REST

Voici les points de terminaison de l'[API RESTful](#) :

Pour la ressource Patient la route de base est '/fhir/Patient', seules les méthodes n°1 et n°3 son nécessaires au fonctionnement de notre système.

N°	Méthode	Chemin	Description
1	POST	/	Enregistre un nouveau patient.
2	GET	/all	Récupère tous les patients.
3	GET	/{patientId}	Récupère un patient par son ID.
4	DELETE	/{patientId}	Supprime un patient par son ID.
5	PUT	/{patientId}	Met à jour un patient spécifique.

TABLEAU 4.2 –
Requêtes API pour la gestion des patients.

Pour la ressource Observation la route de base est '/fhir/Observation', seules les méthodes n°1 et n°4 son nécessaires au fonctionnement de notre système.

N°	Méthode	Chemin	Description
1	POST	/	Enregistre une nouvelle observation.
2	GET	/all	Récupère toutes les observations.
3	GET	/{observationId}	Récupère une observation par son ID.
4	GET	?subject=Patient/{patientId}	Récupère les observations par sujet.
5	DELETE	/{observationId}	Supprime une observation par son ID.

TABLEAU 4.3 – *Requêtes API pour la gestion des observations.*

c. Diagramme de classes

Ce diagramme illustre les relations et leur cardinalité entre les classes Java utilisées pour l'implémentation FHIR de nos ressources. Ainsi nous remarquons la relation des observations avec le patient par l'intermédiaire du sujet de référence, possédant le même identifiant. De cette manière nous pouvons retrouver les mesures d'un patient donné.

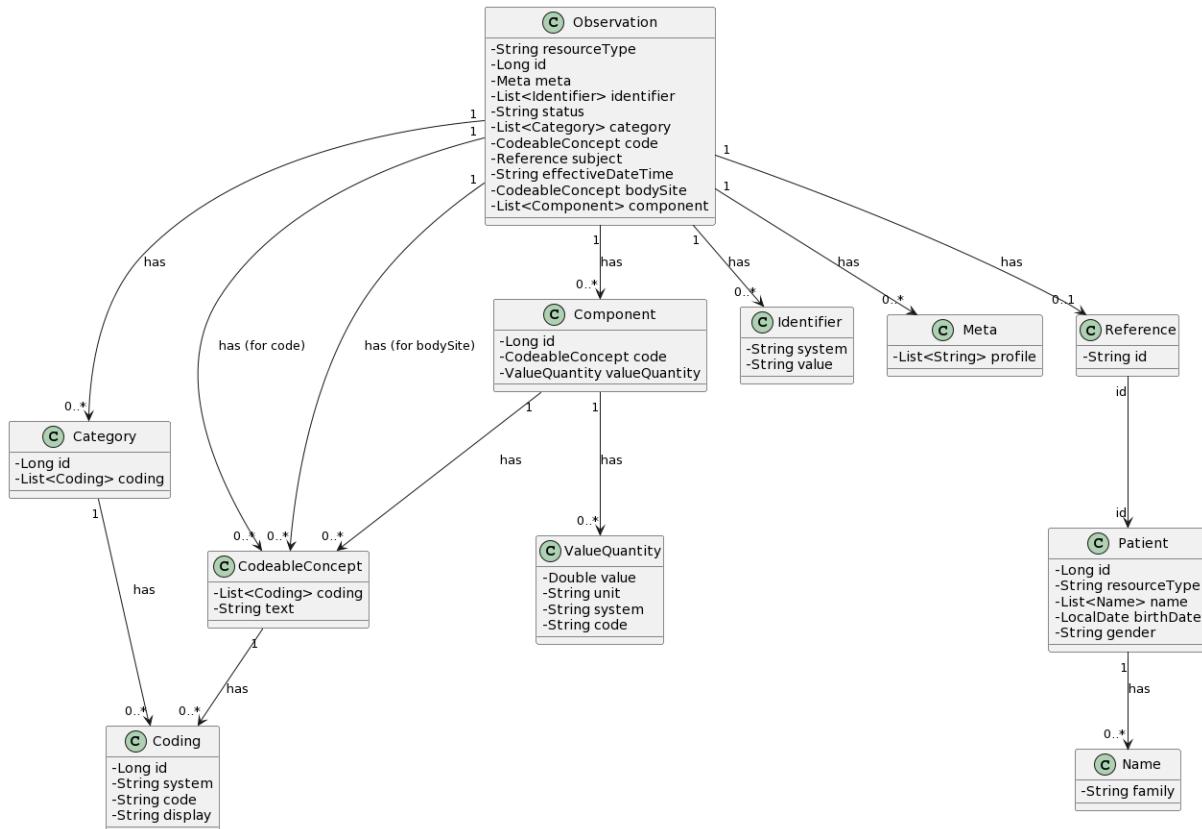


ILLUSTRATION 4.3 – *Diagramme Entité-Association*

Afin de pouvoir manipuler ces données, leurs classes correspondantes ont été implémentées dans notre serveur. Par simplicité, la majorité des attributs non essentiels à notre système n'ont pas été implémentés.

4.4. FRONT-END

L'aspect haut niveau ayant déjà été décrit dans le chapitre précédent, un diagramme de séquence illustrera mieux le fonctionnement de ce composant, le voici :

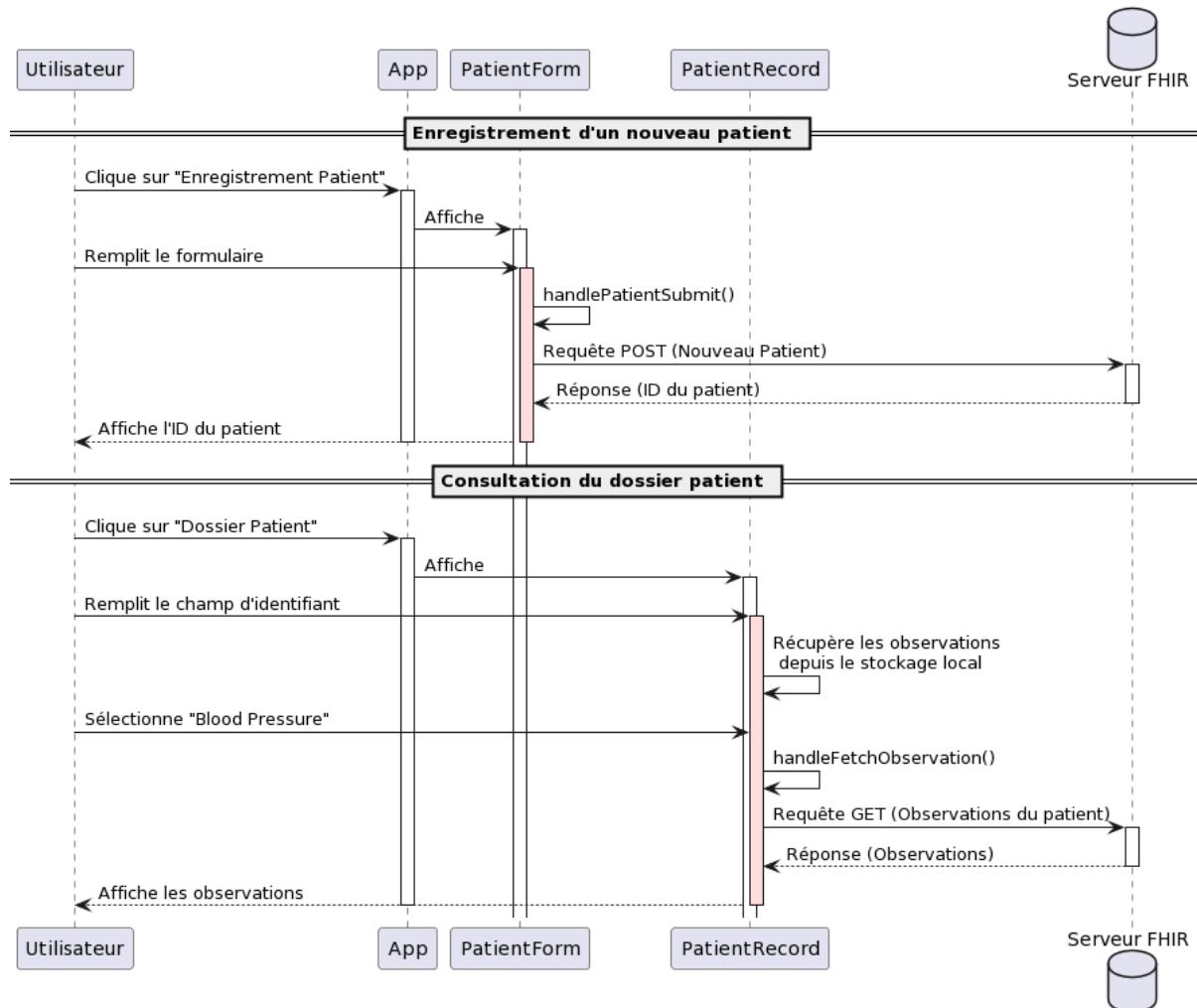


ILLUSTRATION 4.4 – Diagramme de séquence : Nous pouvons voir la division de l'application en deux vues pour la création d'un patient ou la consultation de son dossier. *J'ai remarqué lors de la rédaction qu'il manque la méthode GET récupérant les informations Patient pour la consultation de son dossier.

CHAPITRE 5 : DÉPLOIEMENT

5.1. RASPBERRY PI

Initialement développée en environnement local, nous avons voulu améliorer sa portabilité par étapes. Dans un premier temps, nous avons installé notre serveur et la base de données PostgreSQL sur un micro-ordinateur Raspberry.

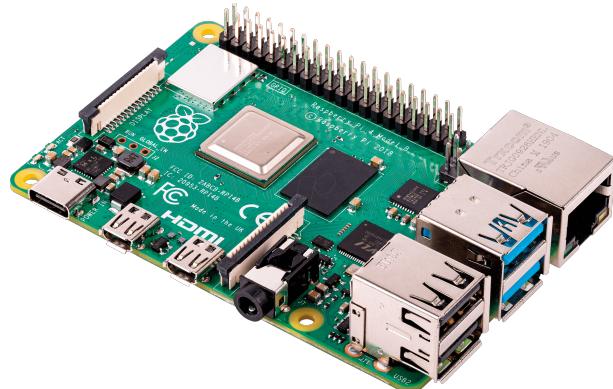


ILLUSTRATION 5.1 – *Micro-ordinateur Raspberry Pi 4 employé pour notre prototype.*

Nous avons alors rencontré des problèmes liés à Cross-Origin Resource Sharing (CORS), en effet, par sécurité, le navigateur bloque la requête de notre front-end vers le back-end, car l'origine de leur domaine n'est plus la même. Nous avons donc installé le front-end sur le Raspberry selon la logique d'une application web complète. Cette fois-ci nous avons rencontré un nouveau problème CORS plus difficile à déterminer. La connexion au front-end étant servie en HTTPS, le navigateur bloque les requêtes HTTP vers le serveur pour des raisons de sécurité. Cette restriction n'est pas apparue en local, peut-être est-ce une indulgence du navigateur Firefox pour faciliter le développement. La solution est donc de définir une connexion sécurisée avec le back-end en partageant la même clé. Il est à préciser qu'il reste à implémenter la connexion HTTPS de l'application mobile au serveur, ce dernier fait une exception sur le port 8080 afin d'y permettre des connexions HTTP depuis cet appareil.

Après configuration du nouveau nom de domaine de notre serveur dans les parties clientes, notre application web est opérationnelle sur l'appareil dont l'architecture processeur est différente, voici ses spécifications :

Notons que notre application fonctionne de base avec Ubuntu desktop 22.04 sur l'architecture x86_64, l'environnement de notre machine locale.

Nom	OS	Architecture	CPU	RAM	Mémoire OS
Raspberry Pi 4 model B	Linux(Raspbian Lite)	ARM64	4	4 GB	128 GB

TABLEAU 5.1 – *Caractéristiques du Raspberry Pi*

a. Client mobile

Une fois installée, l'application mobile fonctionne en stand-alone et permet d'utiliser les services de notre serveur REST pour autant qu'une connexion avec internet soit établie.

Ainsi il est à noter que l'utilisation de `localhost`, dans le contexte de notre application, fait référence au smartphone et non à notre machine de développement.

5.2. CONTENEURISATION AVEC DOCKER

La conteneurisation permet de virtualiser un environnement isolé. Il facilite le développement et le déploiement, car cela évite les problèmes d'adaptation en cas de changement d'environnement extérieur, pour autant que cette technologie soit installée sur ce dernier. Nous employons Docker Compose, car il permet de gérer plusieurs conteneurs à la fois. Dans notre cas, nous avons un conteneur par composant de l'application web : le front-end, le serveur et la base de données. Ces services devant communiquer fonctionnent sur le même réseau virtuel docker.

Nous avons opté pour une image Docker minimale, Eclipse-Temurin : Java Runtime Environment (JRE) 17 Focal, capable d'exécuter notre serveur grâce au fichier Java ARchive (JAR). Une JRE est plus petite qu'une, Java Development Kit (JDK) car elle ne comprend pas les outils de développement Java, elle fait donc une taille de 272MB.

Concernant nos autres applications, les dernières versions docker de Postgres et de Nginx sont utilisées pour la base de données et le front-end respectivement. Ce dernier est servi dorénavant sur Nginx, un serveur adapté à la production, remplaçant Webpack utilisé initialement dans l'environnement de développement React. Un fichier de configuration Nginx a été défini afin d'utiliser le certificat et la clé privée pour établir une connexion Secure Sockets Layer (SSL)/Transport Layer Security (TLS). Ces fichiers sont dérivés de la même paire de clés Public-Key Cryptography Standards (PKCS) utilisée par le serveur Spring Boot, autorisant ainsi une communication sécurisée entre ces deux services. Cependant, l'isolement des applications induit par leur conteneurisation ne permet pas d'accéder à ces données. Il est donc nécessaire de monter les volumes correspondants à l'aide du fichier `docker-compose`.

Le déploiement de notre application conteneurisée fonctionne dès lors en local et sur le

raspberry Pi.

5.3. CLOUD AZURE

Après avoir testé sur le réseau local, nous avons vérifié le fonctionnement de notre application sur Internet. Pour cela nous employons le service Azure de Microsoft, étant plus familier avec son interface. Il permet entre autres, tout comme ses concurrents Amazon Web Service (AWS) ou Google Cloud, le déploiement de Virtual Machine (VM)s sur des serveurs dispersés dans le monde,

Nous avons donc choisi un serveur européen, voici les caractéristiques de la machine :

Nom	OS	Architecture	vCPU	RAM	Mémoire OS
Standard_B1s	Linux (Ubuntu 22.04)	x64	1	1GiB	1023 GiB

TABLEAU 5.2 – *Détails de configuration de la machine virtuelle.*

Voici les ressources Azure actives lors de la mise en route de la VM. La majorité concerne le réseau :

Name ↑↓	Type ↑↓	Resource group ↑↓	Location ↑↓
💻 cardioRemote	Virtual machine	cardioRemote_group	West Europe
🌐 cardioRemote-ip	Public IP address	cardioRemote_group	West Europe
🛡️ cardioRemote-nsg	Network security group	cardioRemote_group	West Europe
🔗 cardioRemote-vnet	Virtual network	cardioRemote_group	West Europe
🌐 cardioremote852	Network Interface	cardioRemote_group	West Europe
💿 cardioRemote_OsDisk_1_1733789dd1f74036a9b1ad7...	Disk	CARDIOREMOTE_GROUP	West Europe
🌐 NetworkWatcher_westeurope	Network Watcher	NetworkWatcherRG	West Europe

ILLUSTRATION 5.2 – *Ressources pour le fonctionnement d'une machine virtuelle sur le cloud Azure*

Après adaptation des hostname pour les interfaces clientes, notre application de télésurveillance fonctionne dans son intégralité. Rappelons que l'objectif principal de ce déploiement était de tester la portabilité du système. Elle n'est donc pas une solution envisagée pour une application réelle en raison de la sensibilité intrinsèque des données médicales.

CHAPITRE 6 : RÉSULTATS

6.1. ENREGISTREMENT D'UN PATIENT

a. Front-end

Il est d'abord nécessaire de créer une ressource Patient afin de pouvoir l'attribuer en tant que sujet dans les futures Observation. Dans l'interface front-end, le formulaire d'enregistrement de patients est prévu à cet effet, l'identifiant logique n°2 nouvellement créé s'affiche dans l'interface :

The screenshot shows the 'CardioRemote' application interface. At the top, there are three tabs: 'CardioRemote' (disabled), 'Dossier Patient' (disabled), and 'Enregistrement Patient' (highlighted in green). The main area contains a form for patient registration:

- First Name: Martin
- Last Name: Müller
- Gender: male
- Date of Birth: 07/08/1990
- Submit Button: Soumettre
- Feedback: Patient créé, identifiant n°2

ILLUSTRATION 6.1 – *Formulaire d'enregistrement d'un patient.*

Nous pouvons observer dans ci-dessous que la requête de type POST a bien été effectuée à l'adresse au port 8443. Ceci correspond bien à l'adresse où notre application est déployée sur le cloud et le port correspond au serveur Spring. Les données transférées sont bien une ressource Patient au format JSON et respectent la structure FHIR.

Status	Method	Domain	File	Type	Trans...	Headers	Cookies	Request
200	OPTIONS	cardioremove.westeuropa.cloudapp.azure.com:8443	Patient	plain	416 B			Filter Request Parameters
200	POST	cardioremove.westeuropa.cloudapp.azure.com:8443	Patient	json	394 B	JSON		
<pre> birthDate: "1990-08-07" gender: "male" ▼ name: [...] ▼ 0: {...} family: "Müller" ▼ given: [...] 0: "Martin" resourceType: "Patient" </pre>								
<p>⌚ 2 requests 123 B / 810 B transferred Finish: 779 ms</p> <p>Filter Output</p> <p>Connecteur (outils) : MozillaWebExtension 20230516 15:16::17 Connecteur (script injecté) : MozillaWebExtension 20230516 15:16::17 ▶ Date Tue Aug 15 2023 18:50:37 GMT+0200 (Central European Summer Time) CHARGEMNT {"resourceType": "Patient", "name": [{"given": ["Martin"], "family": "Müller"}], "gender": "male", "birthDate": "1990-08-07"} 2 </p>								

ILLUSTRATION 6.2 – Requête du formulaire d'enregistrement dans l'outil de développement du navigateur Firefox.

Cette capture d'écran confirme la réception de l'identifiant n°2 et de l'objet patient, attestant de son enregistrement.

Status	Method	Domain	File	Type	Trans...	Headers	Cookies	Request	Response
200	OPTIONS	cardioremove.westeuropa.cloudapp.azure.com:8443	Patient	plain	416 B			Filter properties	
200	POST	cardioremove.westeuropa.cloudapp.azure.com:8443	Patient	json	394 B	JSON			
<pre> id: 2 resourceType: "Patient" ▼ name: [...] ▼ 0: Object {family: "Müller", given: [...] } given: ["Martin"] family: "Müller" birthDate: "1990-08-07" gender: "male" </pre>									
<p>⌚ 2 requests 123 B / 810 B transferred Finish: 779 ms</p> <p>Filter Output</p> <p>Connecteur (outils) : MozillaWebExtension 20230516 15:16::17 Connecteur (script injecté) : MozillaWebExtension 20230516 15:16::17 ▶ Date Tue Aug 15 2023 18:50:37 GMT+0200 (Central European Summer Time) CHARGEMNT {"resourceType": "Patient", "name": [{"given": ["Martin"], "family": "Müller"}], "gender": "male", "birthDate": "1990-08-07"} 2 </p>									

ILLUSTRATION 6.3 – Réponse de la requête d'enregistrement

b. Back-end

Nous pouvons observer dans le terminal ci-dessous le journal SQL d'Hibernate dans notre conteneur spring-app. La requête insère les données du patient et lui attribue un identifiant unique avec la fonction `nextval('patient_sequence')`. Les valeurs des requêtes sont cependant masquées par Hibernate sans doute pour des questions de sécurité.

```
spring-app_1 | Hibernate:
spring-app_1 |   select
spring-app_1 |     p1_0.id,
spring-app_1 |     p1_0.birth_date,
spring-app_1 |     p1_0.gender,
spring-app_1 |     p1_0.resource_type
spring-app_1 |   from
spring-app_1 |     patient p1_0
spring-app_1 |   where
spring-app_1 |     p1_0.id=?
spring-app_1 | Hibernate:
spring-app_1 |   select
spring-app_1 |     nextval('patient_sequence')
spring-app_1 | Hibernate:
spring-app_1 |   insert
spring-app_1 |   into
spring-app_1 |     patient
spring-app_1 |     (birth_date,gender,resource_type,id)
spring-app_1 |   values
spring-app_1 |     (?, ?, ?, ?)
spring-app_1 | Hibernate:
spring-app_1 |   insert
spring-app_1 |   into
spring-app_1 |     patient_name
spring-app_1 |     (patient_id,family,given)
spring-app_1 |   values
spring-app_1 |     (?, ?, ?)
```

ILLUSTRATION 6.4 – *Requêtes SQL d’Hibernate*

6.2. RELAIS DES MESURES MÉDICALES

a. Tensiomètre connecté

Voici les résultats des mesures sur l'appareil, pour simplifier la lecture de nos résultats nous n'en avons seulement deux.



ILLUSTRATION 6.5 – Mesures de pression artérielle et rythme cardiaque

b. Client mobile

Cette capture d'écran montre les informations d'enregistrement (identifiant et mot de passe) de l'application.

Ensuite nous avons les informations de l'[API bluetooth](#), elle repère bien l'appareil BM54 et s'y connecte. Elle s'abonne aux notifications de la caractéristique 0x2a35 et reçoit ainsi deux paquets de données. Nous pouvons voir dans le premier : 112 et 79 pour la deuxième et troisième valeur et dans le second message 121 et 84 aux mêmes indices. Ces valeurs correspondent bien aux valeurs de pression artérielle n°1 et n°2, respectivement, mesurées par le dispositif connecté.

Avant de clore la connexion, l'appareil mobile reçoit deux fois la réponse 201. Ceci nous informe du succès de deux requêtes POST et confirme la création de ces ressources sur notre serveur API REST.

```
LOG  Running "TeleHealth" with {"rootTag":11}
LOG  username: 2
LOG  Password: MyPassword
DEBUG [handleAndroidPermissions] User accepts runtime permissions android 12+
DEBUG BleManager started.
DEBUG [startScan] starting scan...
DEBUG [startScan] scan promise returned successfully.
DEBUG [handleDiscoverPeripheral] 'BM54' found
DEBUG [handleStopScan] scan is stopped.
DEBUG [connectPeripheral] connected to 'BM54'
DEBUG Notification started for characteristic: 00002a35-0000-1000-8000-00805f9b34fb
DEBUG [connectPeripheral] notified on 'BM54'
DEBUG [handleUpdateValueForCharacteristic] Received data from 'FC:D2:B6:57:9D:B9' with
characteristic='00002a35-0000-1000-8000-00805f9b34fb' and value='30,112,0,79,0,0,0,231,7,
8,2,19,11,0,81,0,0,0'.
DEBUG [handleUpdateValueForCharacteristic] Data added.
DEBUG [handleUpdateValueForCharacteristic] Received data from 'FC:D2:B6:57:9D:B9' with
characteristic='00002a35-0000-1000-8000-00805f9b34fb' and value='30,121,0,84,0,0,0,231,7,
8,2,18,50,0,83,0,1,0,0'.
DEBUG [handleUpdateValueForCharacteristic] Data added.
DEBUG [sendData] response received: 201
DEBUG [sendData] response received: 201
DEBUG [handleDisconnectedPeripheral] Device 'FC:D2:B6:57:9D:B9' disconnected.
```

ILLUSTRATION 6.6 – *Information de débogage lors du transfert des données BLE*

6.3. AFFICHAGE DES DONNÉES

a. Client mobile

Nous pouvons dès lors les afficher les données, ci-dessous avec "Mes Données", nous voyons les deux mesures qui sont exactes :

A screenshot of a mobile application interface. At the top, there is a black header bar with white icons for battery level (73%), signal strength, and other status indicators. Below this is a teal-colored navigation bar with white text containing four items: "Systolic", "Diastolic", "Heart Rate", and "Date". The main content area displays a table with two rows of data. The columns are labeled "Systolic", "Diastolic", "Heart Rate", and "Date". The first row shows values 112, 79, 81, and "2.8.2023, 19:11:00" respectively. The second row shows values 121, 84, 83, and "2.8.2023, 18:50:00" respectively.

Systolic	Diastolic	Heart Rate	Date
112	79	81	2.8.2023, 19:11:00
121	84	83	2.8.2023, 18:50:00



ILLUSTRATION 6.7 – Représentation des données sous forme d'un tableau sur le mobile

Un affichage d'un graphique est aussi possible en appuyant sur "Mon Graphique".



ILLUSTRATION 6.8 – *Représentation graphique des données sur le mobile en vue panoramique (l'espacement des données est relatif à l'échelle de temps).*

b. Front-end

Il faut d'abord sélectionner l'identifiant du patient concerné. Après avoir cliqué sur recherche, voici ci-dessous l'affichage du dossier patient avec ses informations relatives.

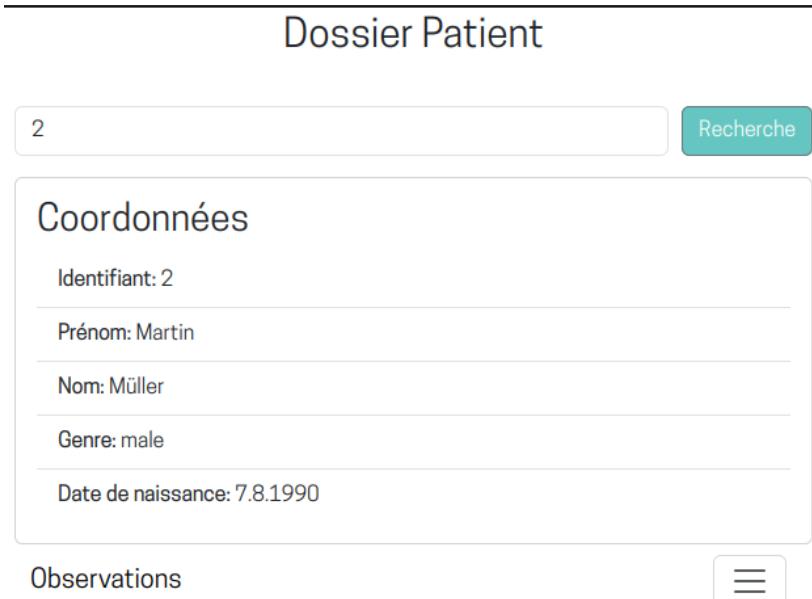


ILLUSTRATION 6.9 – Affichage du dossier Patient recherché.

Nous pouvons confirmer la réception d'un fichier JSON avec la méthode GET à l'adresse du serveur REST. L'affichage de ces données se retrouve dans la capture d'écran ci-dessous :

Status	Method	Domain	File	Type	Tran...	Headers	Cookies	Request	Response
200	GET	cardioremove.westeuropyne.cloudapp.azure.com:8443		2	json	394 B			Filter properties
JSON									
{ id: 2 resourceType: "Patient" name: [{ family: "Müller", given: ["Martin"] }] birthDate: "1990-08-07" gender: "male" }									

1 request | 123 B / 394 B transferred | Finish: 243 ms

ILLUSTRATION 6.10 – Reponse de la requête patient

En ouvrant le menu **Observation** et en sélectionnant le signe vital **Blood Pressure**, une requête est effectuée avec le paramètre **subject** défini à **Patient/2**.

Status	Method	Domain	File	Type	Transferred
200	GET	cardioremove.westeurope.cloudapp.azure.com:8443	Observation?subject=Patient/2	json	3.35 kB
200	GET	cardioremove.westeurope.cloudapp.azure.com:8443	2	json	394 B

ILLUSTRATION 6.11 – *Requête de recherche d'observations du patient 2*

Ceci affiche un en dessous du dossier patient un graphique et un tableau en-dessous :



ILLUSTRATION 6.12 –
Affichage des données du dossier Patient

Finalement, vérification de ces données peut se faire en analysant la charge utile du message reçu. Nous remarquons que nous recevons les deux observations en une fois, la première comprend les valeurs de pression arterielle de 112 et 79, ce qui correspond au jeu n°1 de mesures systolique et diastolique de notre appareil. Son sujet est bien un Patient dont l'identifiant est 2.

```
JSON
▼ entry: [ { ... }, { ... } ]
  ▼ 0: Object { id: 1, resource: { ... } }
    id: 1
    ▼ resource: Object { id: 1, resourceType: "Observation", effectiveDateTime: "2023-08-02T17:11:00.000Z", ... }
      id: 1
      resourceType: "Observation"
      ▶ meta: Object { profile: [ ... ] }
      ▶ identifier: [ { ... } ]
      ▶ category: [ { ... } ]
      ▶ code: Object { text: "Blood pressure systolic & diastolic", coding: [ ... ] }
      ▶ subject: Object { id: "Patient/2" }
      effectiveDateTime: "2023-08-02T17:11:00.000Z"
      ▶ bodySite: Object { coding: [ ... ] }
      ▶ component: [ { ... }, { ... }, { ... } ]
        ▼ 0: Object { code: { ... }, valueQuantity: { ... } }
          ▶ code: Object { coding: [ ... ] }
          ▶ valueQuantity:
            value: 112
            unit: "mmHg"
            system: "http://unitsofmeasure.org"
            code: "mm[Hg]"
        ▼ 1: Object { code: { ... }, valueQuantity: { ... } }
          ▶ code: Object { coding: [ ... ] }
          ▶ valueQuantity:
            value: 79
            unit: "mmHg"
            system: "http://unitsofmeasure.org"
            code: "mm[Hg]"
        ▶ 2: Object { code: { ... }, valueQuantity: { ... } }
      ▶ 1: Object { id: 2, resource: { ... } }
```

ILLUSTRATION 6.13 – Réponse de la requête de recherche des observations associées au patient 2

CONCLUSION

Cette démonstration a mis en lumière la puissance de diverses technologies, nous permettant de concevoir des logiciels plus robustes et de les développer plus rapidement. Il est également évident qu'un même problème peut avoir plusieurs solutions. La solution idéale n'étant pas forcément la plus performante, mais celle qui répond aux exigences dans le temps imparti. Parmi les qualités essentielles du développement logiciel, nous soulignons la modularité et l'interopérabilité. Ces dernières se retrouvent à diverses échelles de notre système. Il est ainsi possible d'interchanger facilement notre back-end avec la solution HAPI sans nécessiter d'ajustement. Les technologies de conteneurisation couplées aux services cloud décuplent cette modularité, conduisant à des systèmes d'une agilité accrue. Fort de ces connaissances et de ces outils, l'éventail des possibilités n'a jamais été aussi grand et ne demande qu'à être exploité par le système médical, comme il a toujours su si bien le faire. Toutefois, certains obstacles demeurent. Les systèmes cloud sont souvent écartés lorsqu'il s'agit de données sensibles, et l'utilisation d'appareils non médicaux est fréquemment exclue pour des raisons évidentes. Inévitablement, des dilemmes surgissent lors de l'intégration de nouvelles technologies, mais ils doivent toujours être résolus dans l'intérêt du patient.

Ce travail m'a permis d'éclairer les zones d'ombres qui restaient sur la forme et la transmission de l'information dans une application web à l'architecture classique. J'ai pu apprendre une approche différente de la construction d'interfaces utilisateurs, avec la librairie React et son système de composants. Il a été intéressant de comprendre le fonctionnement du protocole **BLE**, un standard autant répandu dans nos appareils connectés. J'ai découvert des outils puissants, comme la conversion automatisée entre les objets issus du paradigme de la **Programmation Orientée Objet (POO)** et en entrées dans la base de données, permettant sa manipulation uniquement par un langage de haut niveau. J'ai constaté la difficulté de connaître le moment opportun d'arrêter de persister dans une idée. Devenir alternative moins idéale afin de progresser. J'ai dorénavant la conviction de l'utilité de normes telles que **FHIR**. Bien que contraintes au début, leur adoption permet d'avoir une assurance dans l'intégrité des données et leur transmission. De plus, l'interopérabilité qu'elle propose permet de développer des logiciels plus agiles. Finalement, j'ai pu réaliser à mon échelle, la force des synergies entre ces différentes technologies et les possibilités qu'elles peuvent générer.

Si cette application est amenée à devenir opérationnelle, il sera prioritaire de définir la façon dont les données seront gérées. L'adoption d'un standard, que ce soit **FHIR** ou un autre, souligne l'importance accordée à l'intégrité des données dans l'industrie. Il sera important de préciser quelles données seront partagées entre le système médical et l'application et par quelle méthode. De même, certaines données du patient devront-elle être partagés par le système médical afin de permettre son authentication dans l'application. Conscients de ces enjeux, l'aboutissement de ce projet est encourageant car il ouvre la voie à diverses adaptations de systèmes de télésurveillance, étant donné la facilité de connexion à d'autres dispositifs. Des balances avec impédance, des capteurs de saturation d'oxygène, des glucomètres afin de suivre la glycémie d'un patient atteint du diabète ne sont que quelques exemples possibles.

ANNEXES

ANNEXE 1

Caractéristique	Description
N° du modèle	BM 54
Méthode de mesure	Mesure de la tension artérielle au bras, oscillométrique et non invasive
Plage de mesure	Pression dans la manchette 0 – 250 mmHg, pression systolique 50 – 250 mmHg, pression diastolique 30 – 200 mmHg, pouls 40 – 180 pulsations/minute
Précision de l'indicateur	pression systolique ± 3 mmHg, pression diastolique ± 3 mmHg, pouls $\pm 5\%$ de la valeur affichée
Incertitude de mesure	écart type max. admissible selon des essais cliniques : pression systolique 8 mmHg/ pression diastolique 8 mmHg
Mémoire	2 x 60 emplacements de mémoire
Dimensions	L 139 mm x 194 mm x H 48 mm
Poids	env. 360 g (sans batterie, avec manchette)
Taille de la manchette	de 22 à 44 cm
Alimentation électrique	4 piles 1,5 V AAA
Durée de vie des piles	Pour environ 200 mesures, selon le niveau de tension artérielle, de la pression de gonflage et du nombre de connexions Bluetooth®.
Classement	Alimentation interne, IP21, pas d'AP ni d'APG, utilisation continue, appareil de type BF
Transfert de données par technologie sans fil Bluetooth®	Le tensiomètre utilise la technologie Bluetooth® Low Energy, Bande de fréquence 2402 MHz – 2480 MHz, Puissance d'émission 4 dBm max., Compatible avec les smartphones/tablettes Bluetooth® 4.0

TABLEAU 6.1 – Caractéristiques du tensiomètre

RÉFÉRENCES DOCUMENTAIRES

1. La participation du patient, [sans date] [en ligne]. Disponible à l'adresse : <https://www.hug.ch/rapport-qualite/participation-du-patient> [consulté le 16 août 2023].
2. MARTIN-DU-PAN, Rémy C., 2009. White coat hypertension (syndrome de l'hypertension de la blouse blanche). Rev Med Suisse. Vol. 227, no 43, pp. 2424-2424.
3. PRICEWATERHOUSECOOPERS, [sans date]. Nouvelle étude sur les hôpitaux : inflation, pénurie de personnel qualifié, tarifs rigides – une combinaison insidieuse pour les hôpitaux suisses. PwC [en ligne]. Disponible à l'adresse : <https://www.pwc.ch/fr/centre-de-presse/inflation-penurie-de-personnel-qualifie-tarifs-rigides-une-combination-insidieuse-pour-les-hopitaux-suisses.html> [consulté le 16 août 2023].
4. Digitising clinical data : an uphill road for Switzerland, 2021SWI swissinfo.ch [en ligne]. Disponible à l'adresse : <https://www.swissinfo.ch/eng/business/sanit%C3%A0/digitizing-clinical-data-an-uphill-road-for-switzerland/46363068> [consulté le 16 août 2023].
5. Healthcare's Data Tsunami | Brunswick Group, [sans date] [en ligne]. Disponible à l'adresse : <https://www.brunswickgroup.com/healthcare-data-i20729/> [consulté le 16 août 2023].
6. Unleashing the Power of Data with IoT and Augmented Reality, 2021BCG Global [en ligne]. Disponible à l'adresse : <https://www.bcg.com/publications/2020/unleashing-the-power-of-data-with-iot-and-augmented-reality> [consulté le 16 août 2023].
7. Smartphones are becoming the control centre of people's lives – only 8% of Swiss do not have one, [sans date] Deloitte Switzerland [en ligne]. Disponible à l'adresse : <https://www2.deloitte.com/ch/fr/pages/press-releases/articles/deloitte-in-switzerland-smartphones-become-control-centre.html> [consulté le 16 août 2023].
8. Assigned Numbers, [sans date] Bluetooth® Technology Website [en ligne]. Disponible à l'adresse : <https://www.bluetooth.com/specifications/assigned-numbers/> [consulté le 16 août 2023].
9. Resource - FHIR v5.0.0, [sans date] [en ligne]. Disponible à l'adresse : <https://www.hl7.org/fhir/resource.html> [consulté le 16 août 2023].
10. Home, [sans date] LOINC [en ligne]. Disponible à l'adresse : <https://loinc.org/> [consulté le 16 août 2023].
11. react-native-ble-manager [logiciel] [en ligne]. 15 août 2023. Innove. [consulté le

16 août 2023]. Disponible à l'adresse : <https://github.com/innoveit/react-native-ble-manager> [consulté le 16 août 2023].