Université de Lorraine
Licence informatique

Année 2018-2019

Rapport de stage

Clinique Kuindo-Magnin
5 Rue Contre-Amiral Joseph du Bouzet
Nouméa, Nouvelle-Calédonie

8 avril – 31 mai

Olejniczak
Matthieu

Avant-propos

Le sujet initial devait porter sur la prédiction de pathologies à venir et leurs évolutions. Cependant, à cause d'un manque de données exploitables vis à vis du dossier patient, on m'a alors proposé de faire la géolocalisation des patients au sein de la clinique.

Le stage a été enrichissant mais peu de tests ont pu être réalisés. Vis-à-vis du matériel actuel, mon stage est une réussite.

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage.

Dans un premier temps, j'adresse mes remerciements à M. Lilian VEDRENNE, mon maître de stage, pour la qualité de son encadrement ainsi que sa disponibilité. Le partage de ses connaissances m'a permis de très bien comprendre chaque point du sujet.

Je saisis cette occasion pour remercier le corps administratif et en particulier M. Emmanuel JEANDEL pour m'avoir accordé ce stage.

Je désire aussi remercier tout le bureau informatique pour leur accueil chaleureux ainsi que pour leur bonne humeur.

Un grand merci à mon père pour m'avoir recommandé cet endroit mais aussi pour son soutien et sa présence.

Pour finir, merci à toutes les personnes qui ont accepté de relire mon rapport et de me donner leur avis sur celui-ci.

Sommaire

Introduction

Du 8 avril au 31 mai, j'ai effectué un stage au sein de la clinique Kuindo-Magnin, située sur le territoire français de la Nouvelle-Calédonie. J'ai déjà séjourné quelques semaines sur ce territoire il y a cinq ans. Ayant la possibilité d'y retourner, j'ai saisi cette occasion pour y chercher un stage. Mon père m'a alors mis en relation avec le bureau informatique qu'il connait.

J'ai de ce fait intégré le service informatique en tant que développeur. Au final, mon but a été la localisation des patients par géolocalisation au sein de la clinique. Ce stage m'a montré une première approche du monde de travail et principalement dans le domaine de la médecine, qui n'est pas le même que dans une vraie société informatique par exemple.

Pour commencer, nous présenterons les grands points de l'entreprise. Nous étudierons ensuite le travail que j'ai réalisé, y compris le contexte et l'environnement, l'objectif de celui-ci. Nous verrons aussi la démarche et les méthodes que j'ai utilisées et les résultats engendrés ainsi que l'état du travail à la fin de ce stage. Au final, nous dresserons un bilan de tout ceci.

Présentation de l'entreprise

L'entreprise est la clinique Kuindo-Magnin. Elle se trouve en Nouvelle-Calédonie et est l'unique clinique du pays. La Nouvelle-Calédonie est une collectivité territoriale de type pays d'outre-mer disposant d'une large autonomie. La santé fait partie des compétences exercées par le gouvernement de Nouvelle-Calédonie. L'entreprise dispose des services de médecine, de chirurgie, d'obstétrique, de soin de suite et de réadaptation. Elle possède 285 lits et places. Elle a ouvert en septembre 2018 et est l'union des trois anciens établissements de soins du territoire.

Travail réalisé

Contexte

Le sujet de stage choisi est la géolocalisation des patients au sein de la clinique.

Il existe un parcours patient selon qu'il reste une journée ou qu'il soit en hospitalisation complète. Par ailleurs, ce parcours amène le patient à beaucoup circuler dans la clinique. Ce parcours représente le chemin du patient durant son séjour dans la clinique. Il est dans sa chambre, puis il est conduit au bloc opératoire, il retourne dans sa chambre…

Le géolocalisation permet par exemple d'aider pour cette situation : un patient est d'abord assigné à une chambre à l'accueil. Cependant, lors de son arrivée en service, il arrive que les services de soins redéfinissent la chambre pour des raisons de commodités. L'information ne circulant pas en temps réel, donc, on ne peut pas savoir quand un patient change de chambre. Un médecin ou un brancardier ou le service d'accueil cherchant un patient, il ne le trouvera pas facilement à l'endroit indiqué dans l'outil.

Environnement de travail

Peu de contraintes m'ont été données durant le stage. Le langage de développement était libre, j'ai donc opté pour java pour des raisons de confort avec l'utilisation de l'environnement de développement IntelliJ. J'ai commencé par utiliser Wampserver pour gérer une base de données sous MySQL en local. Je l'ai ensuite exportée sur la base de données qui m'a été fournie avec le logiciel HeidiSQL. Le système de gestion de base de données MySQL est la seule contrainte qui m'a été imposée. J'ai également utilisé Android Studio pour comprendre le fonctionnement de Kontakt.io ainsi que MQTT dont il sera question plus tard.

Les semaines de mon stage se sont déroulées comme telles :

Semaine 1	- Lecture de documentation
Semaine 2	- Interprétation des informations des bornes
Semaine 3	- Mise en place de la base de données
Semaine 4	- Création des logs
Semaine 5	- Tests Fin du projet
Semaine 6	- Préparation du rapport - Préparation de la soutenance
Semaine 7	- Intégration dans la société informatique ISINC
Semaine 8	- Finalisation du rapport

Objectif

Localisation applicative permettant le suivi du parcours d'un patient via géolocalisation.

Démarche et méthode suivies

Chaque patient serait équipé d'un petit appareil, appelé « beacon », qui est attaché à leur bracelet. Un beacon envoie un signal en BLE (Bluetooth Low Energy, soit Bluetooth à basse consommation) qui peut être intercepté par des bornes, appelées « gateway ». Ces bornes récupèrent des informations sur tous les appareils à leur portée. Ces appareils ont été achetés sur le site de Kontakt.io [1].

Chaque beacon et chaque gateway sont uniques. Les beacons sont reconnus par :

- Un UUID (Universally Unique IDentifier), qui est identique et permet de représenter par exemple le propriétaire.

- Un « major », qui permet d'identifier les beacons par groupe.

- Un « minor » qui permet de distinguer les beacons individuellement par rapport au groupe.

Il existe deux types de beacons. Les « iBeacons », créés par Apple. Ils sont simples d'implémentation mais possèdent moins de fonctionnalités. Contrairement aux « Eddystone » qui sont plus complexes mais possèdent plus d'informations. Il est possible de changer la puissance des ondes émises par les beacons à l'aide de TX power (qui correspond à la force du signal) pour qu'ils soient détectés dans un périmètre plus restreint (voir Annexe 2 pour la configuration des beacons).

Le but est alors le suivant :

- À l'aide de gateways et de beacons, il faut connaître la position d'un patient à un instant t, et le chemin qu'il a parcouru.

Il y a cependant quelques contraintes :

- Un beacon peut être détecté par plusieurs gateways, et à l'inverse une gateway peut trouver plusieurs beacons.

- Les ondes traversent les murs.

Kontakt.io met à disposition une API (Application Programming Interface, soit Interface de Programmation Applicative) pour tout ce qui est géolocalisation. Une API est un ensemble de fonctions et de classes qui permet la communication entre programmes [2]. Cette API donne un accès direct à toutes les ressources disponibles sur leurs services. Il est nécessaire d'utiliser une clé API secrète disponible sur notre compte Kontakt.io pour s'y connecter. Elle fournit certaines commandes en cURL pour récupérer des informations sur le lieu, sur les « triggers » (déclencheur, on en reparlera plus en détail prochainement), sur les UUIDs etc... cURL est une interface en ligne de commande destinée à récupérer le contenu d'une ressource accessible par un réseau informatique [3].

Différentes APK (Android Package Kit, une collection compressée de fichiers pour Android [4]) aident à comprendre le fonctionnement de la géolocalisation en proposant des services comme la localisation d'un beacon, le balayage d'une région, une connexion au cloud Kontakt.io (un cloud est réseau partagé) ainsi que la configuration d'un beacon. On peut aussi trouver, sur la boutique Google Play par exemple, une application pour l'administration du matériel fourni.

Plusieurs méthodes ont été utilisées pour extraire les informations des gateways. La première provient directement du panel du site Kontakt.io (voir Annexe 3). Ce panel permet de gérer tous les produits possédés. Il permet également de gérer des triggers qui déclenchent des actions sous certaines conditions. Par exemple, si un beacon possédant un certain major est détecté

par une gateway précise, alors une action comme « poster un message dans Slack » est déclenchée (Slack est une plate-forme de communication collaborative). Ce message comporte les informations que l'on a souhaitées, comme par exemple l'identifiant (ID) de la gateway, celui du beacon et la proximité. Ceci est géré à l'aide d'un « webhook », tel qu'IFTTT [5] qui a été choisi. Un webhook est une fonction configurée par un utilisateur qui permet d'extraire des informations d'un événement [6].

Ceci aurait pu être très pratique pour récupérer facilement les informations principales, cependant bien que les beacons soient interceptés, aucun déclenchement n'a été effectué. L'idée fut abandonnée en début de projet.

La deuxième méthode, celle qui a été gardée, fut trouvée dans la documentation de développeur Kontakt.io [7]. Il est possible d'extraire des informations d'une gateway à l'aide d'un protocole « WebSocket ». Ce protocole permet d'ouvrir une connexion permanente entre le navigateur et le serveur [8]. J'ai alors choisi d'utiliser le protocole WebSocket MQTT.fx [9]. On crée un profil en utilisant une adresse fournie par Kontakt.io, leur port ainsi qu'une clé secrète liée à notre compte Kontakt.io pour pouvoir se connecter. Il est ensuite possible de s'inscrire à une page pour récupérer les informations fournies par les gateways d'un certain lieu, de forme « stream/:venueID/presence ». Les doubles points indiquent qu'il faut fournir sa propre donnée. L'ID du lieu (représenté donc par venueID) a été trouvé grâce à l'API Kontakt.io.

On peut stocker les informations des gateways dans des fichiers texte en format JSON. Celles-ci sont composées de tous les appareils détectés par les gateways et pour chacune d'entre elles on retrouve :

- la date de détection (format timestamp)
- l'ID de la gateway
- l'ID de l'appareil
- le RSSI (Received Signal Strength Indication, soit l'indication de la force du signal reçu)
- la proximité (qui représente un intervalle de distance)
- l'adresse MAC de l'appareil
- le type de balayage (toujours BLE)
- la distance.

Maintenant qu'un accès aux données est possible, il suffit d'interpréter ces dernières.

J'ai commencé par créer un programme qui me permet de lire un fichier. L'un des soucis avec les fichiers créés, c'est qu'ils ne possèdent qu'une seule très grande ligne. Par conséquent ce

n’est pas possible d’utiliser la lecture ligne par ligne. Cependant, le format permet de bien séparer les appareils. En effet, les fichiers sont de la forme :

[{*Appareil_1*},{*Appareil_2*},{*Appareil_3*},{…}]

Il est alors possible de séparer chaque bloc d’appareil à l’aide des accolades. Chaque bloc est composé de plusieurs informations, telles que décrites dans l’annexe 4. Il faut prendre en considération que tous les appareils qui possèdent du BLE sont interceptés par les bornes. Un filtre est alors nécessaire. Les beacons sont reconnaissables par leur ID dans le champ « trackingId ». Ce champ contient une adresse MAC pour un autre appareil. Etant donné que l’on cherche les patients dans leur chambre, il faut aussi faire un tri sur la distance. Un beacon se trouvant à 10 mètres d’une gateway et qui se trouve quelques chambres plus loin ne nous intéressent pas. Le champ « distance » est alors utile ici. Il peut être de trois valeurs différentes : immediate (si le beacon se trouve sur la borne, à moins d’un mètre de la gateway), near (dans un périmètre de 1 mètre à 3 mètres) et far (entre 3 mètres et jusqu’à 70 mètres selon la force du signal). Dans le cas des chambres, seuls les beacons dans un périmètre de 3 mètres nous intéressent.

Il reste maintenant à exploiter les données reçues. Une base de données sous MySQL a été créée pour ceci. MySQL a été choisi pour sa simplicité et son efficacité. Toutes les lignes de toutes les tables sont déjà créées. Seules des modifications sont effectuées. La volumétrie est alors très correcte. Une table contient autant de lignes que de beacons, une autre autant de lignes que de lieu, et la table principale a aussi une taille en fonction du nombre de beacons. Nous avons d’abord estimé qu’une simple table avec l’ID du beacon, la date, la proximité, la distance ainsi que l’ID de la gateway. Cependant, on a trouvé cette table trop légère et mal exploitée. Il n’y avait aucune association avec d’autres éléments pour une mise en pratique. Deux autres tables ont alors été mises en place : la table Beacon_ipp, composée de l’ID du beacon ainsi que de l’Identifiant Permanent du Patient (IPP), permettant d’associer un patient à un beacon; la table Lieu, composée d’un numéro de chambre et de l’ID de la gateway, permet de définir les lieux présents associés à une gateway. Cette table ne changera jamais. La table principale servant à la géolocalisation contient en plus une pièce active mais aussi la dernière pièce visitée par le patient. L’ID du beacon et l’IPP sont uniques. Le numéro de chambre et l’ID de la gateway le sont aussi. La table de géolocalisation réfère à Beacon_ipp pour l’ID du beacon et à Lieu pour les numéros de chambre et l’ID de la gateway. L’IPP est un numéro unique qui permet d’identifier un patient. Le fait de retenir la dernière pièce visitée par un patient permet d’avoir une idée sur ce qu’il a déjà entrepris. Est-ce que le patient a déjà subi son opération et est retourné dans sa chambre ou est-ce que les retards se sont accumulés et il attend depuis tout ce temps ? Une petite application web de test m’a été fournie pour associer l’IPP d’un patient à un beacon et vérifier si le changement s’effectue correctement.

Comment ceci fonctionne-t-il ?

Un patient est assigné à un beacon dès son arrivée à l'accueil. Une chambre prédéfinie lui est attribuée, où il s'y dirige. Arrivé en service, il y a deux possibilités : le patient va voir les infirmières, qui vont lui assigner, ou non, une nouvelle chambre pour des causes pratiques; ou alors le patient est dirigé vers sa chambre prédéfinie. Hormis le problème que tout le monde n'est pas au courant du changement, maintenant le patient est affiché avec une chambre incorrecte dans le premier cas. Pour remédier à cela, une sorte de chambre provisoire est nécessaire. A force d'être à proximité d'une même gateway, un compteur s'incrémente. A partir d'un certain nombre d'occurrences, la chambre provisoire devient la chambre active du patient. La mise à jour est alors faite dans la base et donc sur les écrans. Le patient arrive dans sa première chambre. Cinq minutes plus tard il n'est encore assigné à aucune chambre. Un patient attend parfois plusieurs heures avant de partir en bloc. Le temps d'assignation n'a donc pas besoin d'être court. Une infirmière arrive et lui demande de changer de chambre, car celle-ci sera plus proche du bloc, donc plus pratique. Dix minutes après son changement de chambre, celle-ci lui est finalement attribuée. Un script PHP vient alors chercher dans la base référence de géolocalisation le numéro de la chambre du patient et l'indiquer sur un écran. Ce script a aussi été écrit par mon tuteur et permet ensuite la liaison avec d'autres logiciels. Il ne fait que récupérer l'information que j'ai placée dans la base de données. Mon application est une valeur physique et sûre. Elle fait foi de référence de localisation.

Pour la sortie, il n'y a rien à gérer concernant la base de données. Après sa procédure de sortie, le patient n'est plus compté dans la clinique. Il n'est alors plus dans une base (base qui contient tous les patients présents dans la clinique), donc l'IPP ne s'y trouve plus non plus, donc l'information ne sera pas recherchée. Il n'y a donc pas de problème de chambre indiquée utilisée mais qui est en réalité vide.

Le plus gros du travail est terminé. Nous avons cherché à optimiser le programme. Par exemple, il faut au préalable s'être connecté avec MQTT.fx pour lancer le programme. Il a fallu chercher comment implémenter ceci directement dans le code pour faciliter l'utilisation du logiciel de géolocalisation. Grâce à Github, on trouve de nombreux schémas, dont ce que nous cherchions [10]. Il restait à adapter selon nos propres besoins. L'implémentation de MQTT dans le programme permet en plus d'éviter l'utilisation de fichiers initialement prévu pour sauvegarder les données JSON récupérées par les gateways. Celles-ci sont désormais directement envoyées en chaîne de caractères.

L'idée d'un système de logs a aussi été énoncée. A un instant T, on récupère les positions de tous les beacons présents dans l'établissement qui seront enregistrées dans une nouvelle ligne d'une table réservée à ceci. Ces logs représentent le chemin patient. Celui-ci peut aider à visualiser le parcours d'un patient afin de pouvoir établir un trafic (comme la mise en place d'une IA pour déterminer le meilleur chemin par exemple), détecter les points chauds de passage (et définir si besoin des relations entre les problèmes médicaux et la localisation).

Pour conclure, une archive exécutable java a été conçue, puis une exécutable à l'aide du logiciel Launch4j [11] (qui permet de convertir une archive java en exécutable) pour une meilleure portabilité et une utilisation simplifiée.

Résultat obtenu

La décision la plus difficile était la disposition des gateways au sein de la clinique. Le périmètre de celles-ci est un cercle et peut s'étendre jusqu'à 70 mètres. On se demande alors combien de gateways faut-il utiliser. Le but principal est quand même de pouvoir localiser un patient dans une pièce. On n'a pas besoin de savoir s'il est dans le couloir.

- La première idée est de placer une gateway au milieu de chaque pièce, et de récupérer les beacons à une distance proche. Cette idée est la plus coûteuse (bien que les appareils ne sont pas spécialement chers), mais aussi la plus simple à mettre en place et la plus efficace. C'est cette idée qui a été retenue actuellement.

- Autrement, on peut placer une gateway devant chaque bloc. Si un patient passe devant, il est donc dans ce bloc jusqu'à ce qu'il soit repéré dans un autre lieu. Le nombre de gateways est radicalement réduit, mais on n'a pas de précision exacte quant à la position du patient. On sait qu'il se trouve dans une chambre, mais laquelle ? De plus, si le patient se retrouve juste à portée de la gateway et fait demi-tour pour une quelconque raison, il est quand même assigné au bloc, alors qu'il n'y sera pas du tout.

- Une dernière idée est d'utiliser la distance d'une gateway pour évaluer la pièce, et de placer d'autres bornes pour évaluer la direction et gérer une précision. Une gateway a une grande portée. Elle peut couvrir plusieurs chambres. Disons pour cet exemple qu'une chambre fait cinq mètres de long. La gateway placée à une extrémité couvrirait jusqu'à six chambres. On utilise les intervalles de la portée pour y placer les salles. D'autres gateways placées dans différentes pièces permettraient de détecter si un patient est plus d'un côté ou non d'une chambre pour éviter la pénalité que peut donner la zone de détection qui est un cercle. Cette idée permet une bien meilleure précision, mais trop complexe à réaliser. Le manque de matériels empêchait n'importe quel test.

La base de données a aussi beaucoup évolué. En plus de rajouter des tables associatives (qui ont grandement facilité la lecture de la table), des champs dans la table principale de géolocalisation sont apparus. La chambre donnée au patient à l'accueil est parfois différente de la vraie chambre qui lui sera attribuée, donc nous avons rajouté un compteur ainsi qu'une chambre provisoire. Sa vraie chambre est indiquée une quinzaine de minutes après qu'il soit resté dans la même.

Au final, on arrive bien à localiser une personne. Si elle s'éloigne d'une borne pour s'en approcher d'une autre, elle est alors assignée à la nouvelle borne.

État du travail et prolongements

Au final, le projet est un succès. La mise en place n'a cependant pas encore eu lieu car le projet n'a pas encore été mis à disposition. Il y a alors un manque de matériels. De plus, la localisation du territoire rend l'importation longue et difficile. Il faut attendre plusieurs semaines avant l'arrivée de matériels.

Un prolongement du sujet serait l'utilisation des adresses MAC détectées par les gateways. Par exemple, si un médecin entre dans une chambre, un message est envoyé sur son téléphone portable contenant directement un lien vers le dossier du patient attribué à la chambre.

Conclusion

Le sujet tel qu'il m'a été donné est un succès. Cependant le projet n'a pas encore été mis en place au sein de la clinique, du fait du manque de matériel. Le positionnement des gateways dans l'établissement est la plus grosse difficulté qui a été rencontrée.

Vis-à-vis de la formation, il manque un regard sur les besoins de l'entreprise et une ouverture à la formation de développeur. Il lui manquerait la notion de fullstack (aussi bien du développement en front-end qu'en back-end) pour être complète et utile. Notre formation propose essentiellement du back-end. Le développement web se fait rare alors qu'il est optimal pour un client léger.

Personnellement, je considère mon stage comme une réussite. Le sujet m'a paru complètement étranger au départ ce qui me donnait l'impression de ne pas avancer en ne faisant que lire de la documentation toute une semaine. Cependant le sujet était intéressant et j'ai pu y découvrir de nouvelles technologies. Je n'ai pas uniquement fait de la programmation. Les notions de WebHook et de WebSocket m'étaient inconnues mais j'ai trouvé ces technologies intéressantes. Les échanges avec mon tuteur furent riches et m'ont permis une très bonne compréhension du sujet. J'ai également élargi mes connaissances dans le domaine médical (en plus du monde de l'entreprise). Je suis entré dans le bloc opératoire pour trouver des solutions de simplification pour les brancardiers (faire un système de gestion de lits par exemple). J'ai également intégré une société informatique qui m'a montré un aspect très différent de la vie de programmeur à la clinique.

Références bibliographiques

Sites visités :

1. **KONTAKT.IO**. *Site officiel de Kontakt.io.* https://kontakt.io/. [En ligne, accès mai 2019]

2. **WIKIPEDIA**. *Définition d'une API.* https://fr.wikipedia.org/wiki/Interface_de_programmation. [En ligne, accès mai 2019]

3. **WIKIPEDIA**. *Définition de cURL.* https://fr.wikipedia.org/wiki/CURL. [En ligne, accès mai 2019]

4. **WIKIPEDIA**. *Définition d'une APK.* https://fr.wikipedia.org/wiki/APK_(format_de_fichier). [En ligne, accès mai 2019]

5. **IFTTT**. *Site officiel IFTTT.* https://ifttt.com/discover. [En ligne, accès mai 2019]

6. **INBENTA**. *Définition d'un webhook*. https://www.inbenta.com/fr/blog/webhook-chatbot-api-tal/. [En ligne, accès mai 2019]

7. **DEVELOPER KONTAKT.IO**. *API Kontakt.io.* https://developer.kontakt.io/api-reference/10/. [En ligne, accès mai 2019]

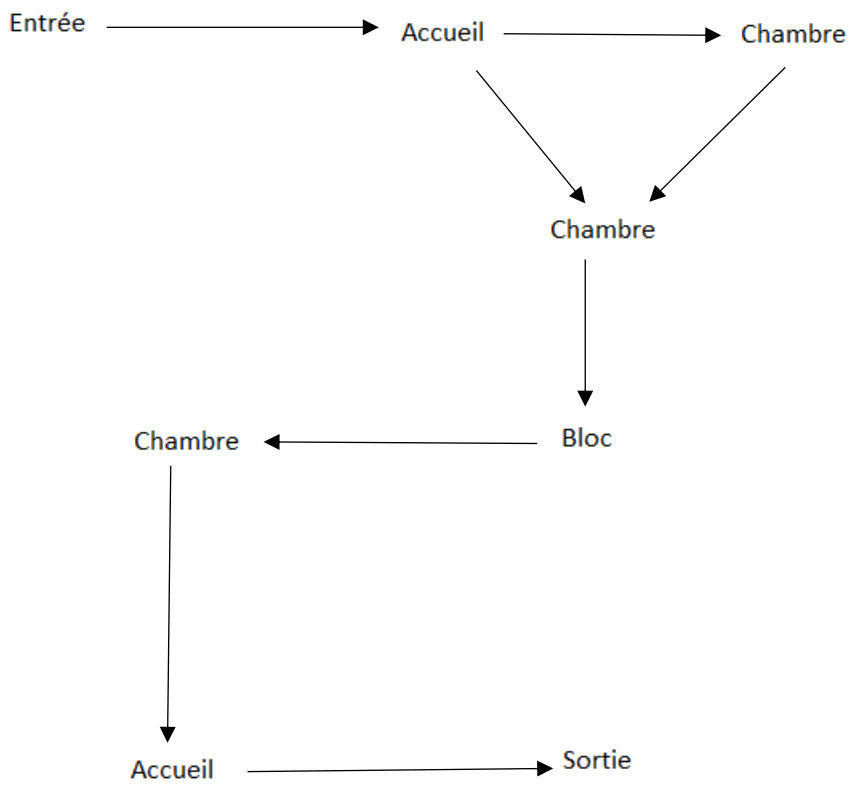
8. **SAMETMAX**. *Définition d'un websocket.* http://sametmax.com/quest-ce-que-les-websockets-et-a-quoi-ca-sert/#tc-comment-title. [En ligne, accès mai 2019]

9. **MQTT.FX**. *Site Officiel.* https://mqttfx.jensd.de/index.php. [En ligne, accès mai 2019]

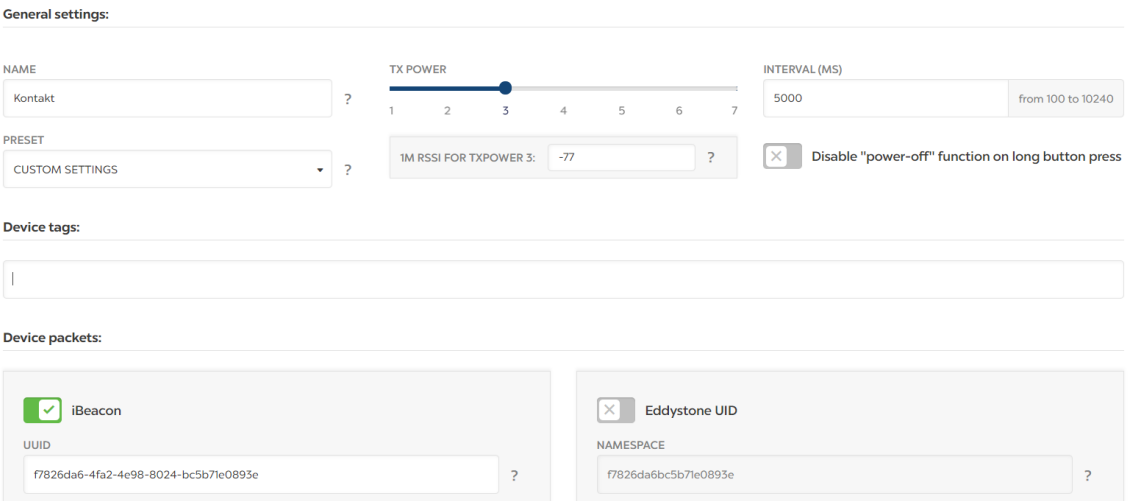
10. **GITHUB**. *Exemple d'un client MQTT.* https://gist.github.com/m2mIO-gister/5275324. [En ligne, accès mai 2019]

11. **LAUNCH4J**. *Site officiel.* http://launch4j.sourceforge.net/. [En ligne, accès mai 2019]

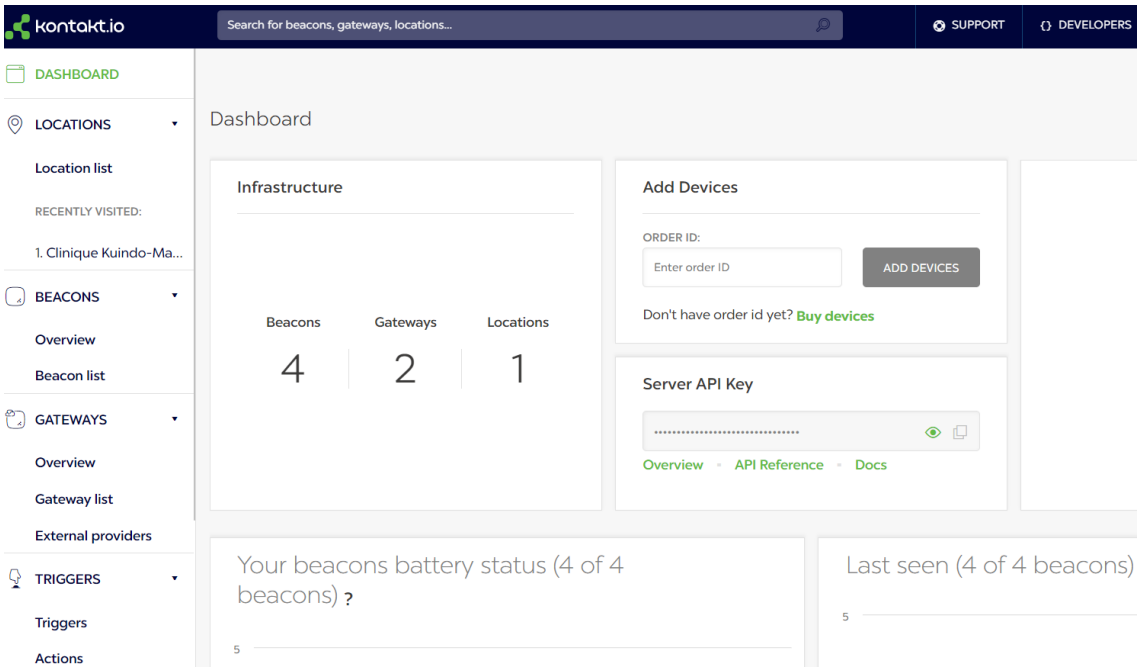
Annexes



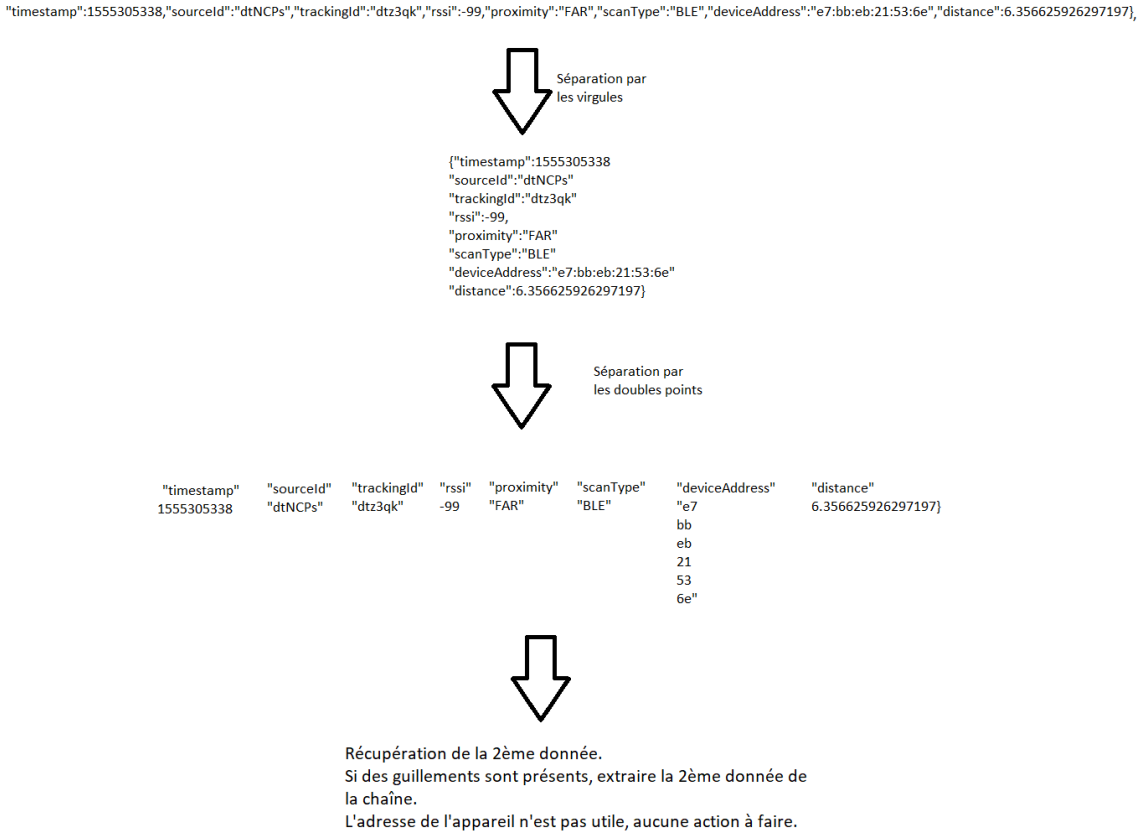
Annexe 1. Exemple d'un parcours patient



Annexe 2. Ecran de configuration d'un beacon



Annexe 3. Panel Kontakt.io



Annexe 4. Séparation du fichier JSON