

Master of Science HES-SO in Engineering

Orientation : Technologies de l'information et de la communication (TIC)

SMART BELL

Fait par

Mahmoud Assaf

Sous la direction de

Prof. Dr. Laura Elena Raileanu

A la Haute École d'ingénierie et de gestion du canton de Vaud HEIG-VD

Prof. Monsieur Markus Jatton – Représentant MRU

Lausanne, HES-SO//Master, 9 juin 2017

Table des matières

1	TABLE DES ILLUSTRATIONS	4
2	REMERCIEMENT.....	5
3	ABRÉVIATIONS.....	5
4	RÉSUMÉ.....	6
5	INTRODUCTION	6
5.1	CONTEXTE DU PROJET.....	7
5.2	OBJECTIFS DU PROJET	7
5.3	MATÉRIELS À DISPOSITION.....	7
5.4	CAHIER DES CHARGES	8
5.5	DÉMARCHE PROPOSÉE	8
5.6	OBJECTIFS ET PORTÉE DE L'ÉTUDE	8
5.7	STRUCTURE DU RAPPORT.....	9
6	ANALYSE ET CONCEPTION.....	10
6.1	ETAT DE L'ART.....	10
6.1.1	WI-FI Positioning System (WPS)	10
6.1.2	Recherche de l'existant	11
6.1.2.1	Positionnement, traçage	11
6.1.2.2	Positionnement, traçage et alertes	11
6.2	ENVIRONNEMENT DE TRAVAIL	11
6.3	LES TECHNOLOGIES	11
6.3.1	Les beacons	12
6.3.1.1	Avantages et inconvénients.....	12
6.3.1.2	Comment ça marche	12
6.3.1.2.1	L'émetteur	13
6.3.1.2.2	Le récepteur.....	13
6.3.1.3	Caractéristiques techniques d'un iBeacon	13
6.3.1.4	Comment gérer un beacon.....	14
6.3.1.5	Notre iBeacon « la marque SmartBeacon »	14
6.3.2	Outils et services pour le développement de beacons	14
6.3.3	Choix des technologies.....	15
6.3.3.1	Le Framework cross-plateforme.....	16
6.3.3.2	Le Framework de développement « Solution Stack »	18
6.3.3.2.1	La MEAN Stack - MongoDB, Express, AngularJS et NodeJS.....	19
6.3.3.2.1.1	Les caractéristiques techniques	19
6.3.3.2.1.2	Les langages de programmations prérequis.....	19
6.3.3.2.1.3	Les avantages de la MEAN Stack	19
6.3.3.3	Licences	20
6.3.4	Les cas d'utilisation	21
7	L'APPLICATION, RÉALISATION ET TEST	22
7.1	ARCHITECTURE DE L'APPLICATION	22
7.2	LE RESTFUL API SERVER.....	23
7.2.1	Les APIs.....	24
7.2.1.1	L'API beacons	24
7.2.1.1.1	Le modèle Beacon	25
7.2.1.2	L'API staffbeacons	27
7.2.1.3	L'API alerts.....	28
7.3	LA BASE DE DONNÉE « MONGODB »	28
7.3.1	Description des collections.....	29
7.4	DESCRIPTION DU PROJET ET L'APPLICATION	32
7.4.1	Description du projet Ionic.....	32
7.4.2	Description de l'application	33

SMART BELL - projet d'approfondissement

7.4.2.1	Les fichiers en détails	34
7.4.2.1.1	Le fichier app.js.....	34
7.4.2.1.2	Le fichier service.js.....	35
7.4.2.1.3	Le fichier controller.js	36
7.5	COMMUNICATION CLIENT-SERVEUR « SOCKET.IO »	38
7.6	LES NOTIFICATIONS « PUSH NOTIFICATION »	39
7.7	ALGORITHME DE DISTRIBUTION DES ALERTES	41
7.8	TEST	42
7.9	PROBLÈMES RENCONTRÉS.....	42
8	INSTALLATION ET DÉPLOIEMENT	43
8.1	DÉMARRAGE DU SERVEUR	43
8.1.1	Changement de l'adresse IP	43
8.2	DÉMARRAGE DU CLIENT	44
8.2.1	Modification de l'adresse IP	44
8.3	COMPILATION DE L'APPLICATION	44
9	AMÉLIORATIONS.....	45
9.1	L'ALGORITHME DE DISTRIBUTION DES ALERTES AMÉLIORÉ	45
9.1.1	L'algorithme version 2.....	46
9.1.1.1	Si le cas est prioritaire	46
9.1.1.2	Si le cas est moins prioritaire.....	47
9.2	ESTHÉTIQUE ET INTERACTION	47
10	CONCLUSION	47
10.1	DATE ET SIGNATURE	48
11	RÉFÉRENCES.....	49
11.1	DOCUMENTATIONS	49
12	ANNEXES	50
12.1	LIVRABLES	50
12.1.1	Code Bar patient et staff	50
12.2	LES INTERFACES GRAPHIQUES DE L'APPLICATION	52
12.2.1	SmartBell-patient – projet smartbell	52
12.2.2	SmartBell-staff – projet staff.....	55
12.3	JOURNAL DE TRAVAIL.....	57
12.4	STRUCTOGRAMME DISTRIBUTION DES ALERTES	63
12.5	PLANIFICATION.....	64

1 Table des illustrations

Figure 1: logo Eddystone, iBeacon, AltBeacon. Source : [19].....	12
Figure 2 : émission de signal autour du beacon. Source : [19].....	12
Figure 3 : le beacon relation entre émetteur et récepteur. Source [18]	12
Figure 4 : exemple d'un CMS pour gérer les beacons. Source [20].....	14
Figure 5 : la MEAN Stack. Source [25]	19
Figure 6: AngularJS Two-Way Data Binding. Source [25]	20
Figure 7: Use Case patient – client SmartBell-patient	21
Figure 8: Use Case staff – client SmartBell-staff	22
Figure 9 : aperçu de l'architecture de l'application	23
Figure 10 : arborescence du serveur	23
Figure 11 : l'API beacons (ou patient)	24
Figure 12: l'API staffbeacons (ou staff).....	28
Figure 13 : l'API alerte	28
Figure 14 : aperçu de la collection alerts dans la base de donnée.....	29
Figure 15 : aperçu de la collection beacons dans la base de donnée	30
Figure 16 : aperçu de la collection staffbeacons dans la base de donnée.....	30
Figure 17 : aperçu de la base de donnée avec l'outil Robomongo.....	31
Figure 18 : aperçu du schéma de la base de donnée	31
Figure 19 : arborescence du projet, arborescence du projet smartbell.....	32
Figure 20 : arborescence de l'application smartbell – dossier www	33
Figure 21 : aperçu de la notification push côté staff	39
Figure 22 : aperçu de l'alerte côté staff.....	39
Figure 23 : aperçu de la notification push côté patient	40
Figure 24 : liste des besoins du patient	42
Figure 25 : l'application client (patient et staff) installée sur le Smartphone	44
Figure 26 : paramètres de localisation d'un staff dans la base de donnée	45
Figure 27 : aperçu de la page Accueil et Scan – patient.....	53
Figure 28 : aperçu de la page Beacons et Account - patient.....	54
Figure 29 : aperçu de la page Accueil et Scan - staff	55
Figure 30 : aperçu de la page Beacons et Account - staff	56
Figure 31 : structogramme de l'algorithme de distribution des alertes.....	63
Figure 32 : planification	64

2 Remerciement

Je tiens à remercier prof. Laura Elena Raileanu et l'assistant M. Fabien Dutoit pour leur aide et pour m'avoir guidé tout au long de ce projet. Aussi, je les remercie de m'avoir donné l'occasion de travailler sur ce projet en relation avec le domaine de soin de la santé dont j'ai beaucoup d'intérêt, qui m'a permis de connaître tant de nouvelles choses et d'apprendre de nouvelles technologies.

3 Abréviations

HEIG-VD	Haute Ecole d'ingénierie et de gestion du canton de Vaud
HC	Health Care
MEAN	MongoDB, Express.JS, AngularJs, Node.js
Patient	Patient à l'hôpital
Staff	Personnel de l'hôpital (infirmière, médecin, assistant, aide-soignant, etc.)
Client	Patient ou Staff
SI	Système d'Information
BLE	Bluetooth Low Energy
BD	Base de donnée
IoT	Internet of Thing – Internet des Objets
API	Application Programming Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JSON	Javascript Object Notation
UI	User Interface
CRUD	Create Read Update Delete
WPS	WI-FI Positioning System

4 Résumé

Ce rapport présente les résultats de mon projet d'approfondissement PA que j'ai réalisé dans le cadre de ma formation à la HES-SO Master en Technologies de l'information et de la communication.

Le projet vise le développement d'une application mobile cross-plateforme, précisément pour les appareils mobiles dotés d'un système d'exploitation Android et iOS. L'application est dédiée au secteur sanitaire principalement les hôpitaux. Elle permet de remplacer la sonnette traditionnelle par une sonnette intelligente et rapide, en permettant à un patient d'exprimer ces besoins en cliquant sur bouton spécifique via l'interface graphique de l'application.

Les besoins sont exprimés sous forme d'une liste de message de types différents comme par exemple, j'ai mal, je veux manger, je veux aller au toilette, urgence, etc.

Selon le type du besoin exprimé par le client, le système va déclencher une alerte et l'envoyer vers la personne adéquate (médecin, infirmier, assistant, aide-soignant, etc.).

L'alerte contient les informations nécessaires pour l'intervention, comme par exemple l'identifiant du patient, son prénom, son nom, son âge, le numéro de la chambre, la date et l'heure exacte de l'alerte.

L'application permet par ailleurs d'optimiser le travail du soignant en lui permettant de connaître avant même de se déplacer auprès du patient, l'objet de sa demande d'assistance.

La localisation du patient et du personnel soignant est assurée par l'implantation de plusieurs « beacons » dispersés partout dans l'hôpital.

En revanche, du côté personnel soignant et patient, le système permet de gagner du temps au niveau de l'intervention, et assure l'envoi de la personne adéquate pour une tâche bien précise, il permet aussi d'améliorer la qualité du service hospitalier.

Ce rapport retrace l'ensemble du travail effectué, allant de l'analyse à la réalisation finale du projet. Il décrit également les éventuelles améliorations qui pourraient être apportées à l'application.

5 Introduction

Dans le cadre du projet d'approfondissement à la HES-SO Master j'ai été mandaté à réaliser le développement du projet « Smart Bell » proposé par Dr. Eduardo Sanchez doyen du département des technologies de l'information et de la communication à la HEIG-VD. Le projet est supervisé par Dr. Laura Elena Raileanu professeur à la HEIG-VD et Monsieur Fabien Dutoit assistant à l'institut de recherche appliqué et développement de la HEIG-VD. Lors de l'évaluation, le projet sera évalué par l'expert Monsieur Markus Jatton professeur à la HEIG-VD.

Le projet est en relation avec le domaine sanitaire, principalement les hôpitaux et également les grandes cliniques. Il aborde la notion de traitement du besoin du patient depuis son accueil et le temps d'attente pour l'intervention de celle-ci.

Dans les paragraphes suivants j'aborde le contexte du projet, le cahier des charges initial, les objectifs du projet et la démarche proposée pour la suite.

5.1 Contexte du projet

Lors d'une hospitalisation, le patient dépend des soignants, ce qui peut être parfois lourd à supporter, en particulier lorsqu'il s'agit de les solliciter pour des besoins élémentaires (par exemple aller aux toilettes, boire, manger, être soulagé ou rassuré, etc.). La sonnette actuellement disponible résonne de manière non différenciée, quel que soit le besoin.

Du côté personnel soignant il pourrait gagner du temps en sachant d'avance qui envoyer vers le patient et pour quelle raison. Du côté patient il bénéficiera d'une qualité de service meilleure en lui envoyant la personne adéquate pour une tâche bien précise, en conséquence il gagnera également du temps puisque dès sa requête le personnel soignant connaît d'avance qui lui envoyer pour son spécifique besoin.

Un dispositif permettant l'automatisation de ce processus améliorerait certainement la qualité du service hospitalier et donnerait une satisfaction et un confort pour le patient.

Avec l'essor de l'internet des objets (Internet of Things) nous pourrions profiter de ce progrès technologique en utilisant particulièrement les « beacons » afin de pouvoir localiser le patient et le personnel soignant.

5.2 Objectifs du projet

L'objectif principal du projet est le développement d'une première version de l'application Smart Bell avec les fonctionnalités de bases qui permettent d'automatiser le processus de sonnette intelligente rapide mentionné ci-dessus.

Les principaux objectifs du projet :

- Développer une application cross-plateforme (le patient et le staff pourront installer l'application sur leur appareil mobile).
- Développer un système d'identification (scan d'un Code Bar).
- Implémenter le parrainage du mobile avec le beacon le plus proche.
- Implémenter l'envoi d'une requête du patient.
- Implémenter la réception de la requête du patient par la personne adéquate.
- Implémenter un algorithme permettant de distribuer les requêtes intelligemment.

5.3 Matériels à disposition

Un kit de la marque « SmartBeacon » composé de deux beacons a été mis à notre disposition pour le développement du projet. Le beacon va nous permettre de localiser le patient ainsi que le staff de l'hôpital.

Un paragraphe intitulé « Le beacon » ci-dessous sera consacré pour expliquer en détail ce qu'est un beacon, son intérêt et son principe de fonctionnement.

Concernant le développement et le test de l'application on a utilisé un smartphone de marque Samsung, modèle Galaxy S6 SM-G929F doté de la version Android 7.0.

5.4 Cahier des charges

Cahier des charges initial

Le projet propose le développement d'un système d'information doté de clients fixes et mobiles, et offrant au patient: De connaître l'état d'avancement de sa prise en charge grâce à une remontée d'informations sélectionnées depuis son dossier médical électronique, les renseignements qu'il désire sur son séjour actuel dans l'hôpital, voire des services récréatifs à disposition (musique, radiophonie, etc.).

Un outil lui permettant d'exprimer ses désirs ou ses besoins de manière naturelle et spécifique (usage de Voice mail en conjonction avec la sonnette, par exemple) tout en garantissant un suivi de sa demande.

Des questionnaires qu'il pourrait remplir de manière très flexible, afin d'améliorer les connaissances cliniques de son état de santé.

L'outil permet par ailleurs d'optimiser le travail du soignant en lui permettant de connaître avant même de se déplacer auprès du patient, l'objet de sa demande d'assistance. La requête sera dirigée vers le soignant le plus adéquat pour une tâche donnée, et le suivi de la demande peut être au besoin assuré par le système.

5.5 Démarche proposée

Il n'y a aucun travail préalablement réalisé, on débute avec un projet « from scratch ». Le projet a été divisé en plusieurs étapes, la démarche suivante a été adoptée pour atteindre nos objectifs :

- Recherche des applications similaires existantes.
- Recherche et étude de la technologie beacons.
- Documentation et prise en main du kit beacons à disposition (SmartBeacon)
- Choix des technologies :
- Etude des Framework existants.
 - Choix du Framework de développement cross-plateforme.
 - Choix du Framework Front-End, Middleware, Back-End, Full-Stack, etc.
- Phase d'apprentissage et prise en main des outils choisis (documentations, articles, tutoriels, etc.).
- Développement et implémentation du code.
- Tests et validation.
- Amélioration selon l'avancement du projet.

5.6 Objectifs et portée de l'étude

Le cahier des charges initial est trop riche en termes de fonctionnalités. Nous avons opté pour une stratégie qui consiste à découper les objectifs en plusieurs étapes. La décomposition suivante a été établie pour simplifier les tâches à accomplir :

- Implémentation d'un client patient.
- Implémentation de la détection de beacons à proximité.
- Implémentation d'un serveur.
- Implémentation de la base de donnée.
- Communication client-serveur.
- Implémentation d'un système d'authentification avec la technique scan du Code Bar (le patient et le staff sont identifiés via des Code Bar qui leurs sont attribués).
- Implémentation d'un client staff.
- Implémentation de la communication entre le patient et le staff.
- Implémentation de la notification push.
- Implémentation d'un algorithme simple pour la distribution des alertes aux personnels adéquats.
- Amélioration de l'algorithme de distribution des alertes en le rendant plus intelligent.
- Ajout de nouvelles fonctionnalités selon le temps à disposition.

5.7 Structure du rapport

Le rapport est organisé de la façon suivante :

1. Dans le chapitre Introduction on explique les points suivants :
 - Contexte du projet
 - Les objectifs
 - Le cahier des charges initial.
 - Les matériels à disposition (les beacons).
 - La démarche qu'on a proposée.
 - Les objectifs essentiels et la portée du projet (comment on a découpé le projet en plusieurs étapes).
 - Structure du rapport.
2. Dans le chapitre Analyse et conception on explique les points suivants :
 - Etat de l'art où l'on parle d'application similaire existante actuellement sur le marché.
 - La technologie des beacons et les outils reliés.
 - Le choix des technologies (Framework cross-plateforme, Framework de développement, les langages de programmations et outils qu'on va utiliser, base de donnée, librairies, plugins, etc.).
 - Les cas d'utilisation.
3. Dans le chapitre L'application, Réalisation et test on explique les points suivants :
 - Architecture de l'application.
 - Le serveur RESTful API implémenté avec ExpressJS (les trois APIs beacons, staffbeacons et alerts).
 - La base de donnée implémentée avec MongoDB
 - Description du projet (description du projet Ionic, description et détails dur l'implémentation de l'application).
 - La communication client-serveur et client-client implémentée avec socket.io
 - Les notifications push entre le patient et le staff.
 - L'algorithme de distribution des alertes envoyées par le patient.

4. Dans le chapitre Installation et déploiement on explique les points suivants :
 - Comment installer les outils indispensables.
 - Comment déployer le serveur, la base de donnée et le client.
 - Comment démarrer le système.
 - Comment compiler le projet.
5. Dans le chapitre Améliorations on explique les possibles améliorations à envisager dans le futur (principalement l'algorithme de distribution des alertes).
6. Conclusion.
7. Les références.
8. Les annexes (Code Bar pour patient et staff, captures d'écrans de l'interface graphique de l'application, planning initial et journal de travail).

6 Analyse et conception

6.1 Etat de l'art

Depuis plusieurs années les techniques de positionnement et localisation sont utilisées dans l'industrie. L'évolution de ces techniques est en plein expansion surtout avec l'essor de l'internet des objets IoT et l'apparition de la technologie des « beacon ». Dans le vocabulaire anglophone ces techniques sont connues sous le nom de « Indoor Positioning, Indoor Navigation et Indoor Tracking ». Actuellement ils sont très utilisés dans plusieurs secteurs, les hôpitaux, aéroports, hôtels, station de train et chemin de fer, transport, magasin de vente, entreprise, etc.

Pendant la phase de recherche de l'existant qu'on a consacré pour découvrir les applications similaires et concurrentes à la nôtre, on a cherché à comprendre comment et quelle sont les techniques de géolocalisation utilisés, ceci nous aidera à savoir où l'on se situe par rapport aux concurrents dans le marché.

La majorité des entreprises offrant le services Indoor Positioning combinent entre les deux techniques suivantes:

- Les beacons : explications détaillées dans le paragraphe ci-dessous « Les beacons ».
- WI-FI Positioning System (WPS) : explications ci-dessous.

6.1.1 WI-FI Positioning System (WPS)

Permet de localiser un appareil (smartphone par exemple) en utilisant la détection d'un réseau Wi-Fi.

La technique de positionnement par point d'accès Wi-Fi est basée sur la mesure de l'intensité du signal reçu (Received Signal Strength ou RSS) et sur la méthode d'empreinte. Une empreinte est constituée du RSS, du SSID du point d'accès, et de l'adresse MAC du routeur. Il n'est pas nécessaire de se connecter au réseau.

6.1.2 Recherche de l'existant

L'existant actuel trouvé sur le marché est axé sur les fonctionnalités suivantes :

6.1.2.1 Positionnement, traçage

Il existe plusieurs entreprises qui offrent des services orientés vers la localisation, le traçage et le guide de l'utilisateur pour se déplacer d'un endroit à un autre. Ces entreprises sont nombreuses dans le marché, ci-dessous quelques-unes :

Infsoft.com : basé en Allemagne, ils ont implémenté une solution pour les chemins de fer fédéraux Suisse SBB CFF FFS à Zurich.

Indoor.rs : basé en Autriche, ils ont implémenté une solution pour le réseau de télécommunication Suisse Swisscom.

Navizon.com : basé aux Etats-Unis, ils offrent une solution pour la géolocalisation interne et externe.

Spreo.io : basé aux Etats-Unis, ils offrent une solution orientée IoT pour la géolocalisation interne.

6.1.2.2 Positionnement, traçage et alertes

Il existe des entreprises qui offrent, en plus des services mentionnés ci-dessus, la possibilité d'envoi et de reçu d'alerte.

Ce sont des entreprises qui offrent les mêmes services que la catégorie précédente, mais en plus il offre la possibilité d'envoi et de reçu d'alerte.

Senion.com : une entreprise qui propose des fonctionnalités dont les objectifs sont très similaires à notre application (émission et réception d'alertes).

6.2 Environnement de travail

L'intégralité du projet a été codé sur l'environnement de développement « Sublime Text 3 » Build 3126 tournant sur un PC portable HP Envy disposant de Windows 10 Edition Famille 64 bits (Processeur : Intel Core i7-4700MQ 2.4 GHz / Mémoire : 8Go).

Les tests ont été réalisés sur un appareil Android virtuelle « AVD », qui est intégré dans l'SDK Android, et la version du système Android qui a été choisie est 6.2.1.

Pour des raisons de commodités et rapidité on a continué les tests sur un smartphone Samsung Galaxy S6.

6.3 Les technologies

Dans ce paragraphe on décrit les différentes technologies qu'on va utiliser ainsi que les choix des services et des technologies étudiés pour la suite du projet.

6.3.1 Les beacons

Les beacons c'est une technologie très prometteuse, il existe les trois protocoles suivants :

- iBeacon, proposé par Apple en 2013
- Eddystone, proposé par Google en 2015
- AltBeacon, proposé par Radius Networks en 2014



Figure 1: logo Eddystone, iBeacon, AltBeacon. Source : [19]



Figure 2 : émission de signal autour du beacon. Source : [19]

Un beacon émet tout simplement un signal avec son identifiant autour de lui, de ce fait le but d'un beacon est de l'utiliser pour la détection de proximité.

Pour notre projet nous utilisons un kit de beacon dont le protocole est iBeacon.

6.3.1.1 Avantages et inconvénients

La plus grande force des beacons est son indépendance à une connexion réseau ou Wi-Fi, on a seulement besoin de la technologie Bluetooth. Dans le même sens, sa plus grande faiblesse est la nécessité d'avoir activé Bluetooth sur notre appareil mobile (smartphone ou tablette), d'avoir accepté les services de géolocalisation et de s'être inscrit auprès du beacon pour recevoir les notifications.

6.3.1.2 Comment ça marche

Pour communiquer, un beacon utilise un signal Bluetooth Low Energy (Bluetooth 4.0). L'avantage de celui-ci est qu'il n'y a pas besoin d'appairage, contrairement aux précédentes versions de Bluetooth.

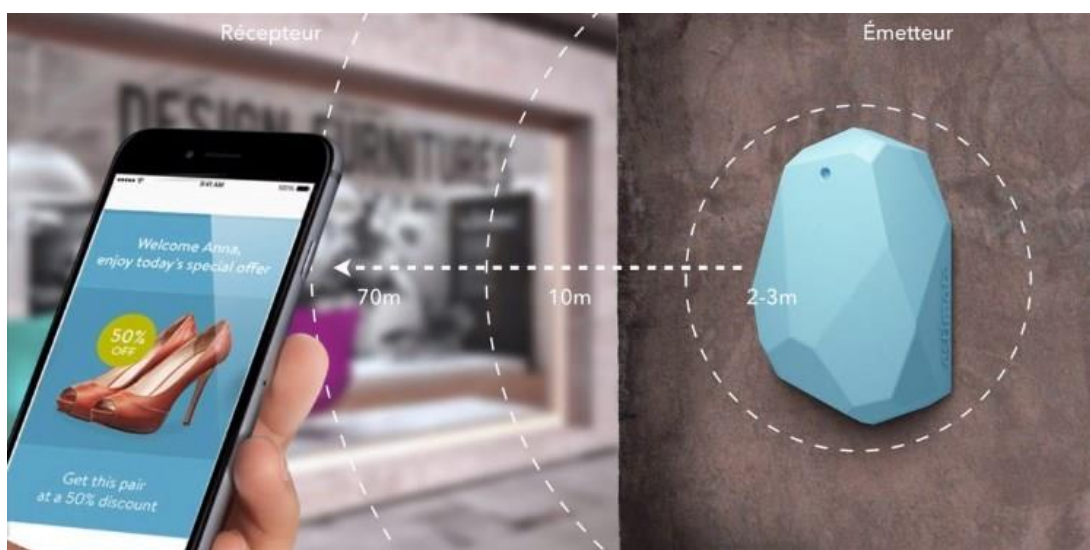


Figure 3 : le beacon relation entre émetteur et récepteur. Source [18]

On peut schématiser le fonctionnement d'un beacon comme étant une relation entre émetteur, et un récepteur :

6.3.1.2.1 L'émetteur

Il transmet au smartphone trois informations importantes permettant l'identification du service proposé :

- L'identifiant général (Enseigne)
- L'identifiant d'un groupe de balises (Lieu de l'enseigne)
- La balise en elle-même (Lieu de balise)

A ces informations s'ajoutent des notions de puissance de signal qui permettront une estimation plus fine de la distance à la balise.

6.3.1.2.2 Le récepteur

Doit être un client ayant déjà installé l'application sur un device smartphone ou tablette. Le device doit être équipé d'une puce BLE pour capter le signal Bluetooth Low Energy de ces balises, le Bluetooth doit être activé dans les préférences de l'appareil mobile, et la réception de messages/push notification doit être autorisée. Doit se situer dans la zone de signal (0 à 50 m) du beacon.

Une fois activé, le Bluetooth ou l'application du client peuvent être réveillés dans la zone de réception.

6.3.1.3 Caractéristiques techniques d'un iBeacon

iBeacon c'est le protocole supporté par le kit SmartBeacon utilisé dans notre projet. Cette borne émettrice est identifiée par la combinaison des trois valeurs suivantes:

- UUID: l'identifiant du producteur
- major: un numéro majeur
- minor: un numéro mineur

C'est en combinant ces trois paramètres que l'on va pouvoir identifier de manière unique chaque « beacon ».

L'API « ng-cordova-beacon » dont on l'utilise dans le projet, pour gérer l'iBeacon permet de nous retourner plusieurs paramètres, dont les principaux sont:

- **Proximity**: nous donne une notion de proximité par rapport à l'iBeacon, c'est à dire de dire si on est très proche ou très éloigné. Ce paramètre peut nous retourner 4 valeurs:
 - Immediate (0-20 cm) (0-8")
 - Near (20cm – 2 m) (8"- 6.5')
 - Far (2 – 70 m) (6.5 – 230')
 - Unknown
- **RSSI** : nous donne la puissance du signal en décibel par rapport à l'iBeacon. Par exemple, plus on s'approche de l'iBeacon plus le RSSI sera élevé, en revanche si on s'éloigne le RSSI sera moins élevé ou plutôt diminue. Si on ne bouge pas le RSSI reste stable.
- **Accuracy**: nous donne une notion de proximité, la différence avec **Proximity** c'est qu'elle permet de définir plus précisément le rayon du cercle du signal, elle est exprimé sous forme de chiffre (exemple 0.28, 6.32, etc.).
- **TX** : nous indique le niveau de puissance du signal à un mètre de l'appareil.

6.3.1.4 Comment gérer un beacon

Lorsqu'on achète un beacon, chaque entreprise nous met à disposition un SDK qu'on pourra le télécharger. Grace à ce SDK on pourra programmer et intégrer les beacon dans les applications qu'on envisage développer. Aussi, l'entreprise nous met à disposition un CMS (Content Management System) pour gérer et visualiser les beacons via leur plateforme. Ce CMS est un outil très pratique pour tracer les utilisateurs et savoir leur position en temps réel.

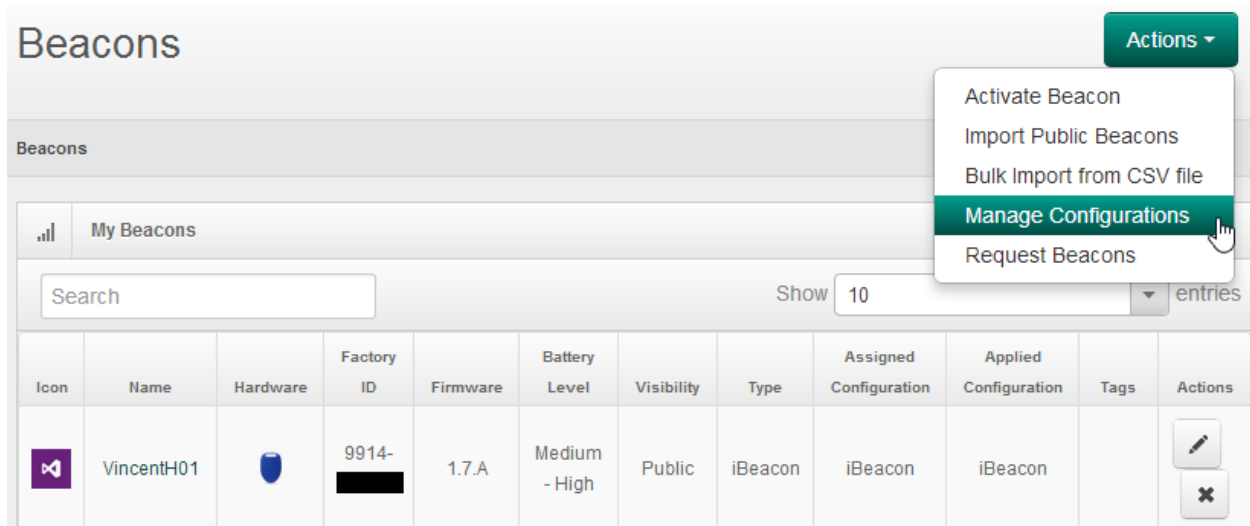


Figure 4 : exemple d'un CMS pour gérer les beacons. Source [20]

6.3.1.5 Notre iBeacon « la marque SmartBeacon »

L'entreprise qui a fabriquée les iBeacons utilisés dans notre projet s'appelle SmartBeacon, leur site web se trouve à l'adresse <http://www.smartbeacon.eu/>. En consultant leur site on s'aperçoit que les informations ne sont plus à jour, leur SDK qui se trouve sur Github à l'adresse <https://github.com/smartbeacon/smartbeacon-android-sdk> n'est plus maintenu et en l'essayant on a constaté qu'il n'est pas fonctionnel. Concernant leur CMS il n'est pas disponible car le lien n'existe plus.

On a essayé de contacter l'entreprise mais malheureusement on a pas réussi à les joindre, apparemment l'entreprise n'est plus opérationnelle.

Pour cette raison, il a fallu qu'on implémente notre propre système pour gérer nos iBeacons. L'API qu'on utilise à ce but est « ng-cordova-beacon », elle est compatible avec les technologies utilisées dont principalement Ionic Framework, Apache Cordova et AngularJS.

6.3.2 Outils et services pour le développement de beacons

Il existe des plateformes qui mettent à disposition des développeurs des APIs pour le développement d'applications permettant de communiquer avec les beacons. Les deux plateformes qui ont été testés sont Evo Things et PubNub.

Evo Things (<https://evthings.com/>) spécialisée dans le domaine d'application embarquée et IoT, propose leur propre IDE avec leur API pour communiquer avec les beacons.

PubNub (<https://www.pubnub.com/>) spécialisé dans le domaine IoT et possède son propre infrastructure de service Publish Subscribe. Il propose leur propre API pour la communication avec les beacons et pour l'envoi et la réception des notifications d'une application à l'autre en mode « two-way communication » en passant par leur serveur.

En effet, les APIs des deux plateformes consiste à prendre l'API de Cordova avec les bibliothèques nécessaires pour chaque protocole de beacons (iBeacon, Eddystone ou AltBeacon) et ajouter une couche supplémentaire pour intégrer leurs propres services.

Pour la raison mentionnée ci-dessus on a décidé de ne pas les utiliser pour les deux principales raisons suivantes :

- Le fait d'ajouter une couche supplémentaire rend l'application complètement dépendante de ces services, et par conséquent le jour où ces services s'arrêtent de fonctionner pour n'importe quelle raison, notre application ne fonctionnera plus.
- Le fait d'utiliser un service externe va influencer sur la performance de l'application.

6.3.3 Choix des technologies

C'est la partie cruciale du projet, le choix des technologies est décisif dans le sens que si le choix est mauvais dès le départ il sera trop tard pour nous par la suite de revenir en arrière, car on risque de dépasser le délai d'échéance du rendu du projet, et on risque de terminer avec une application partiellement fonctionnelle. Raison pour laquelle on a pris soin de consacrer le temps nécessaire pour décider les choix adéquats qui correspondent à nos besoins. Ultérieurement, si le projet fera l'objet d'un travail de Master, le bon choix de technologies facilitera la tâche du successeur car il aura une base solide sur laquelle il pourra avancer sans se soucier des prérequis et sans besoin de refaire le tout « from scratch ».

Lors de choix technologiques plusieurs questions se posent, comme par exemple :

Quelle(s) Framework(s) utilisé ? Quelles APIs et bibliothèques choisir avec ? Est-ce qu'un développement native est-il mieux que le développement cross-plateforme ? Quelle langage(s) de programmation faut-il adopté ? Etc.

La première étape c'était de choisir entre le développement native et cross-plateforme. On a adopté le développement cross-plateforme pour plusieurs raisons :

- L'application fonctionne sur plusieurs plateformes, principalement iOS et Android, avec le même code.
- Plus rapide que le développement native car dans le cas de ce dernier il faut maîtriser d'avance plusieurs langages de programmation, comme Swift ou Objective-C pour la plateforme iOS, et le SDK Android pour la plateforme Android. Tandis que dans le cross-plateforme il suffit de maîtriser des langages standards telle que HTML, CSS, Javascript, JQuery, etc... et un Framework tel que Ionic, PhoneGap, Xamarin, etc. qui sont majoritairement basé sur Javascript, Apache Cordova et autres Framework similaires.
- Moins coûteux en développement et même pour la maintenance puisque dans le cas du développement natif il faut faire le même travail deux fois.

- Dans le cas du développement cross-plateforme, en cas où on aura des problèmes qui empêchent l'application de fonctionner sur les appareils mobile l'application fonctionnera toujours comme Web-App via un navigateur standard, tandis que dans le cas du développement natif l'application ne fonctionnera pas.
- L'application peut être aisément utilisable par tout le monde, précisément par le patient et par le staff, puisqu'actuellement la majorité des gens utilisent des mobiles dotés d'un système d'exploitation Android ou iOS. Dans ce cas l'hôpital n'a plus besoin de mettre à disposition de chaque patient et staff un appareil ou dispositif mobile, et par conséquence elle pourra économiser dans cet investissement.

Il faut bien entendu que nous choisirons un Framework qui est régulièrement mis à jour, qui a une grande communauté, et qui ne risquera pas de s'arrêter dans le futur.

6.3.3.1 Le Framework cross-plateforme

Il s'agit ici de choisir un Framework qui va nous permettre de développer une application qui tournera sur plusieurs plateformes, d'où le terme de « cross-plateforme ».

Les Framework de développement cross-plateforme sont nombreux, il y a plusieurs critères et facteurs pour décider quel Framework choisir. Dans notre projet il ne s'agit pas de faire une étude détaillée en se basant sur tous les critères possibles car en réalité on va choisir un Framework qui correspond à nos besoins et qui permet de couvrir les attentes du contexte du projet. Dans notre cas, certains critères sont prioritaires et indispensables pour le choix de notre Framework, ces critères sont les suivants:

- Supported platforms: les plateformes supportées par le Framework, on s'intéresse principalement à Android et iOS.
- Documentation & community: la qualité de la documentation est primordiale. Par exemple PhoneGap possède une très mauvaise documentation car elle est rarement mise à jour, elle n'est pas bien structurée (parfois on a du mal à trouver ce que l'on recherche), et parfois elle est même erronée. La communauté joue un rôle important, plus elle est grande mieux c'est. Un Framework avec une grande communauté permet d'avoir un support et de l'aide dans les forums et les sites web comme par exemple Stack Overflow (<https://stackoverflow.com/>).
- Access to device-specific features: c'est un facteur très important, notre application a besoin d'avoir accès à la camera pour scanner le Code Bar, elle a besoin également accès à la géolocalisation et le Bluetooth de l'appareil.
- Look and feel: il est important pour nous que le rendu de l'application en terme de UI et design soit d'une bonne qualité.
- Allow me to use my existing development skills: il s'agit de voir si on a les connaissances et les compétences en développement logiciel pour un certain Framework, cela nous permet de gagner plus de temps. Par exemple pour travailler avec Xamarin il nous faut absolument maîtriser C#, or ce n'est pas le cas. S'il existe un autre Framework avec des technologies Web dont on maîtrise, on privilégierait celui-ci.
- Ease of development: certains Framework sont lents au niveau rapidité de développement comme le cas de Unity, d'autre sont plus facile comme Ionic et PhoneGap où il suffit de taper une ligne de commande pour nous générer tout un tas de code prêt à l'exécution.

SMART BELL - projet d'approfondissement

- License and costs: le prix est capital, certains Framework sont totalement gratuits, d'autre partiellement et d'autre complètement gratuit. Quant à la Licence on favorise une Licence gratuite avec droit de propriété.

En se basant sur les critères mentionnés ci-dessus, on a élaboré les tableaux comparatifs suivants qui répondent à nos critères de sélections.

Pour chaque critère on a attribué un score allant de 1 à 5 qui signifie du pire au meilleure respectivement.

Tableau comparatif - licences et open-source :

	Ionic	Meteor	PhoneGap	React Native	Xamarin	Unity
Licence	MIT	MIT	Apache	BSD	MIT	propriétaire
Open-source	OUI	OUI	OUI	OUI	OUI	NON

Tableau comparatif – critères de sélection d'un Framework cross-plateforme :

Technology	Framework					
Critères	Ionic	Meteor	PhoneGap	React Native	Xamarin	Unity
Supported platforms (iOS, Android, Windows Phone)	5	5	5	5	5	5
Documentation & community	4	3	3	2	2	2
Access to device-specific features (camera, GPS, push notification, etc.)	4	3	3	2	2	2
Look and fell	4	2	3	3	2	1
Allow me to use my existing development skills	5	5	5	2	2	2
Ease of development	4	2	4	2	2	1
License and costs	5	5	4	3	5	1
Score	31	25	27	19	20	14

1 +	2 ++	3 +++	4 ++++	5 +++++
-----	------	-------	--------	---------

Le tableau montre que Ionic possède le meilleur score, raison pour laquelle nous l'avons choisi.

Quelques caractéristiques techniques de Ionic :

- Open-source cross-plateforme Framework.
- Basé initialement sur Apache Cordova (qui est open-source et indispensable pour le développement cross-plateforme).
- Permet de créer des applications mobiles hybrides en HTML5.
- Utilise du Javascript standard et AngularJS qui lui-même est basé sur Javascript.
- Le rendu final de l'application en terme de qualité UI et performance est de très haut niveau par rapport aux autres Framework. Source [22]
- Il y a une grande communauté.
- Il est stable et flexible.

La version d'Ionic utilisée est 2.2.2.

6.3.3.2 Le Framework de développement « Solution Stack »

Il s'agit ici de choisir un ou plusieurs Framework qui vont nous permettre de développer les différentes parties de l'application à savoir le Front-End, le Back-End et dans certains cas le Middleware (puisque certains Framework englobe la partie Middleware directement dans le Back-End). On parle de développement côté client, serveur et base de donnée.

Les Framework de développement Front-End, Middleware et Back-End sont très nombreux.

En effet il existe deux alternatives :

- 1 Soit on choisi plusieurs Framework (un pour le Front-End, un pour le Back-End et probablement un pour le Middleware).
- 2 Soit on choisi un package complet. On l'appelle dans jargon informatique Full Stack car le Framework permet de faire le développement entier de l'application (client, serveur, base de donnée, etc.), d'où le terme « full ».

Dans la première alternative, au début il n'est pas toujours évident de savoir quel Framework est compatible avec une telle technologie existante ou un tel Framework existant (comme Ionic puisqu'on la choisi). Il faut essayer le Framework avant de décider de l'adopter définitivement.

Dans la deuxième alternative, le choix des différents Framework est déjà essayé et testé par les experts du domaine, le package est conçu d'une façon pour que les différents Framework composants du package fonctionne entre eux d'une façon compatible, d'où l'avantage par rapport à la première alternative. Un deuxième avantage c'est qu'elle nous évite aussi de faire plusieurs essais pour le choix, et donc on gagnera plus du temps. Un troisième avantage c'est que la cadence du développement en Full Stack est beaucoup plus rapide, et donc on gagnera en terme de productivité.

C'est pour les points positifs mentionnés ci-dessus on a adopté de choisir un Framework Full Stack.

Mais quel package Full Stack Framework choisir?

Notre réponse est toujours selon nos besoins et le contexte du projet. Lors de notre choix du Framework Ionic, explicitement AngularJS nous a été d'office imposé par le Framework lui-même. Donc notre future Full Stack Framework devrait absolument avoir AngularJS intégré. Notre choix est alors la MEAN Stack qui signifie **M**ongoDB, **E**xpress, **A**ngularJ et **N**odeJS dont on détaillera plus ces caractéristiques dans le paragraphe suivant.

6.3.3.2.1 La MEAN Stack - MongoDB, Express, AngularJS et NodeJS

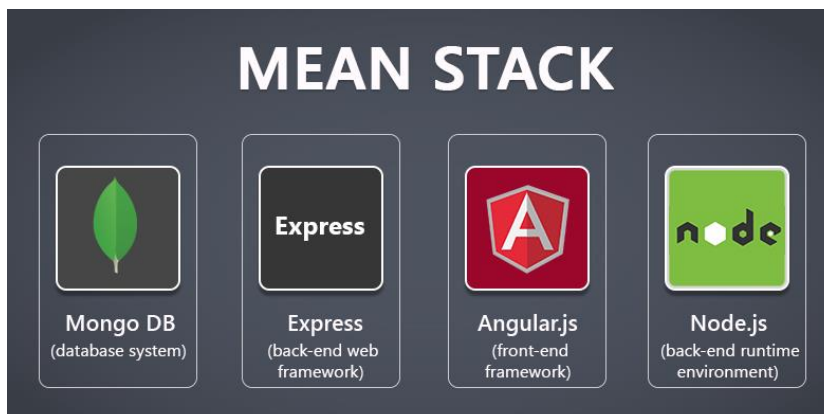


Figure 5 : la MEAN Stack. Source [25]

6.3.3.2.1.1 Les caractéristiques techniques

- MongoDB: Base de donnée NoSQL, agile et scalable.
- Express : Web Application Framework basé sur Node.js, flexible, robuste, permet de construire Hybride Web Application, sachant qu'une application Hybride est une application utilisant le navigateur web intégré du support (Smartphone ou tablette) et les technologies Web (HTML, CSS et Javascript) pour fonctionner sur différents OS (iOS, Android, Windows Phone, etc.).
- AngularJS : Framework Javascript libre et open-source, pour le développement Front-End. La version utilisée est la 1.5.3.
- Node.js: c'est un open-source, server-side et cross-plateforme runtime environnement de développement.

6.3.3.2.1.2 Les langages de programmations prérequis

La MEAN Stack nécessite la connaissance des langages de programmation et des Framework suivants : HTML5, CSS3, Javascript, Node.js, AngularJS (déjà prérequis pour Ionic), Express.js, MongoDB.

Ne connaissant pas tous les technologies mentionnées, une période d'apprentissage et de prise en main a été consacré pour ce but, précisément pour Ionic, Express.js, MongoDB, et AngularJS.

6.3.3.2.1.3 Les avantages de la MEAN Stack

- C'est une technologie très utilisée, très répandue, à la mode, et on trouve derrière une grande communauté.
- Technologie moderne, en expansion, très utilisée, soutenu par une grande communauté.
- Elle est complètement basée sur Javascript (Front-End et Back-End), maîtriser ce langage suffit pour travailler (pas de nécessité d'apprendre de nouveau langage), et nous permet de gagner du temps, d'où l'avantage.
- La base de donnée MongoDB n'est pas compliquée à prendre en main. Elle est extensible, communicable via Javascript pour les opérations CRUD (Create, Read, Update et Delete).
- Opère via AngularJS qui permet la synchronisation entre le modèle et la vue, dans le jargon d'AngularJS ceci est connu sous terme « Two-Way Data binding » :

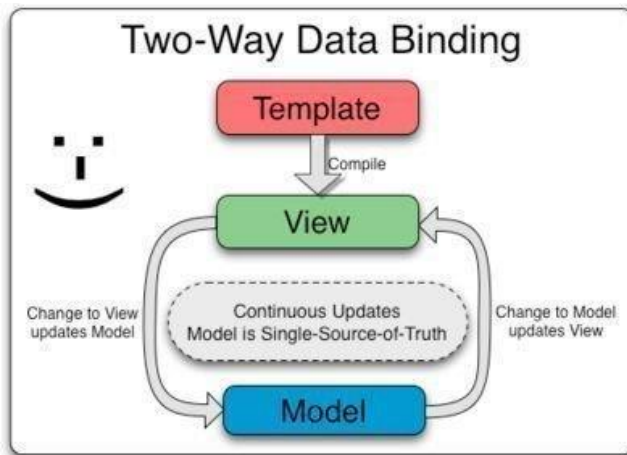


Figure 6: AngularJS Two-Way Data Binding. Source [25]

6.3.3.3 Licences

Dans le tableau ci-dessous nous exposons un récapitulatif des Licences des différents types de des Framework, SDK, logiciels, librairies et plugins utilisés dans le projet.

Framework, SDK, librairie, Plugin			Plugin	
Nom	Licence	Open-source	Nom	Licence
Ionic	MIT	Oui	NgCordova	MIT
AngularJS	MIT	Oui	NgStorage	MIT
ExpressJS	MIT	Oui	PhoneGap Plugin BarcodeScanner	MIT
MongoDB	Apache	Oui	Cordova-plugin-ibeacon	Apacher 2.0
Node.js	MIT	Oui		
Socket.io	MIT	Oui		

6.3.4 Les cas d'utilisation

Le schéma suivant illustre les différents cas d'utilisation du client « patient » :

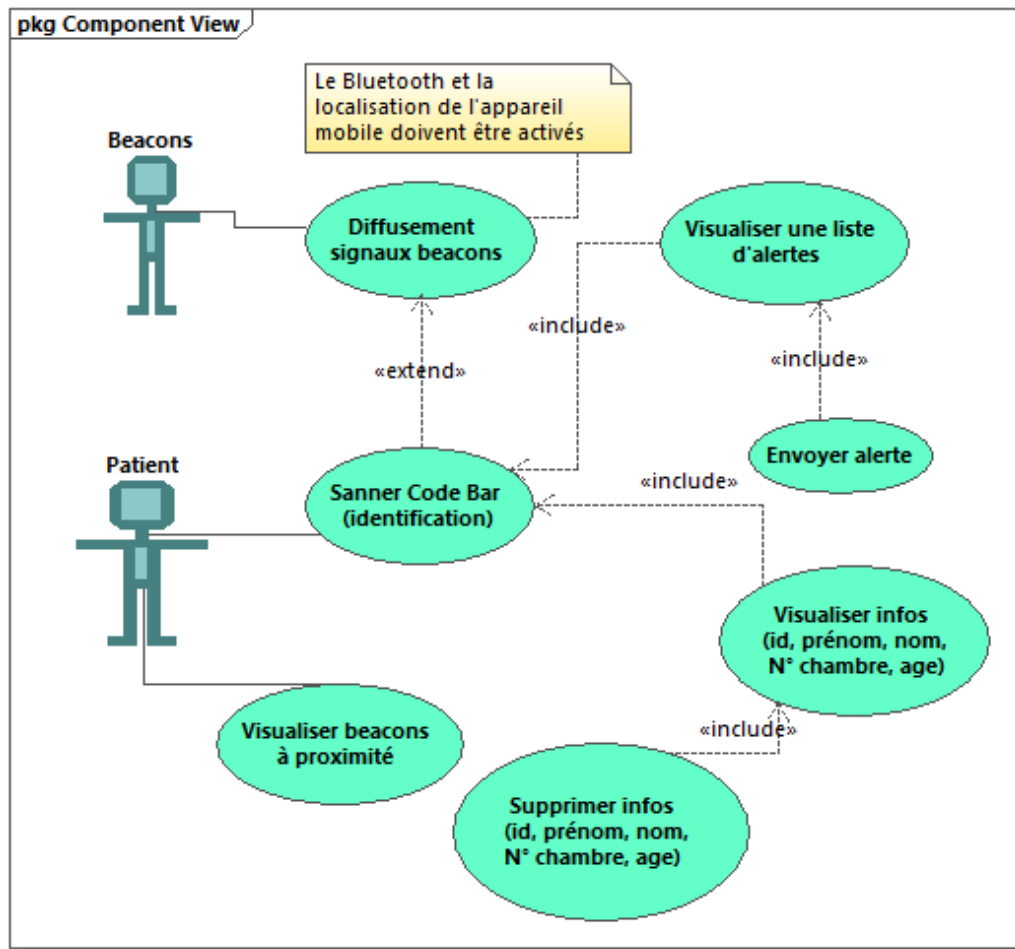


Figure 7: Use Case patient – client SmartBell-patient

Le schéma suivant illustre les différents cas d'utilisation du client « staff » :

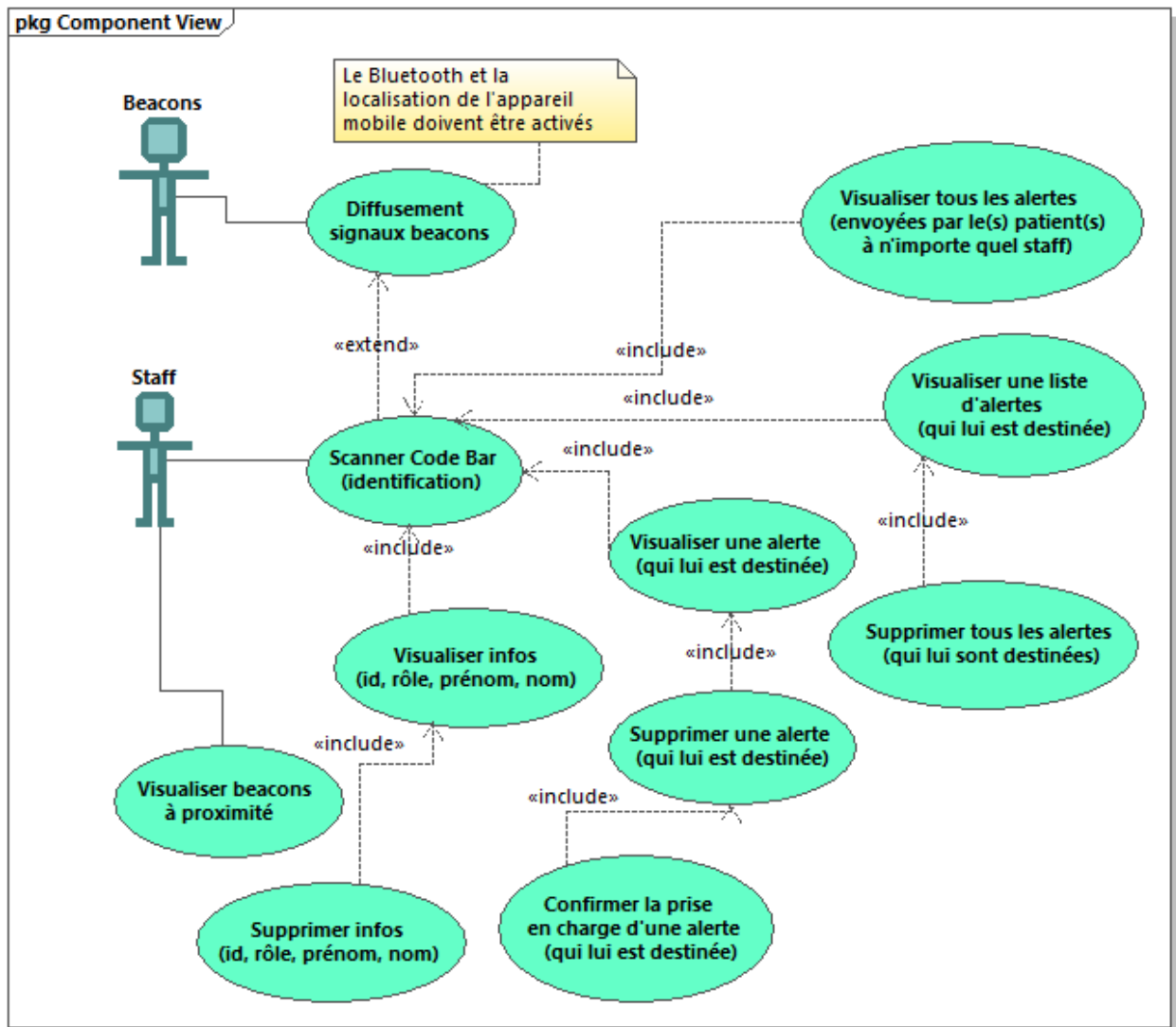


Figure 8: Use Case staff – client SmartBell-staff

7 L'application, réalisation et test

7.1 Architecture de l'application

L'application est composée d'un serveur et de deux clients, un client dédié au patient appelé SmartBell-patient et un autre client dédié pour le staff appelé SmartBell-staff.

Dans la figure ci-dessous nous montrons un aperçu de l'architecture générale de l'application:

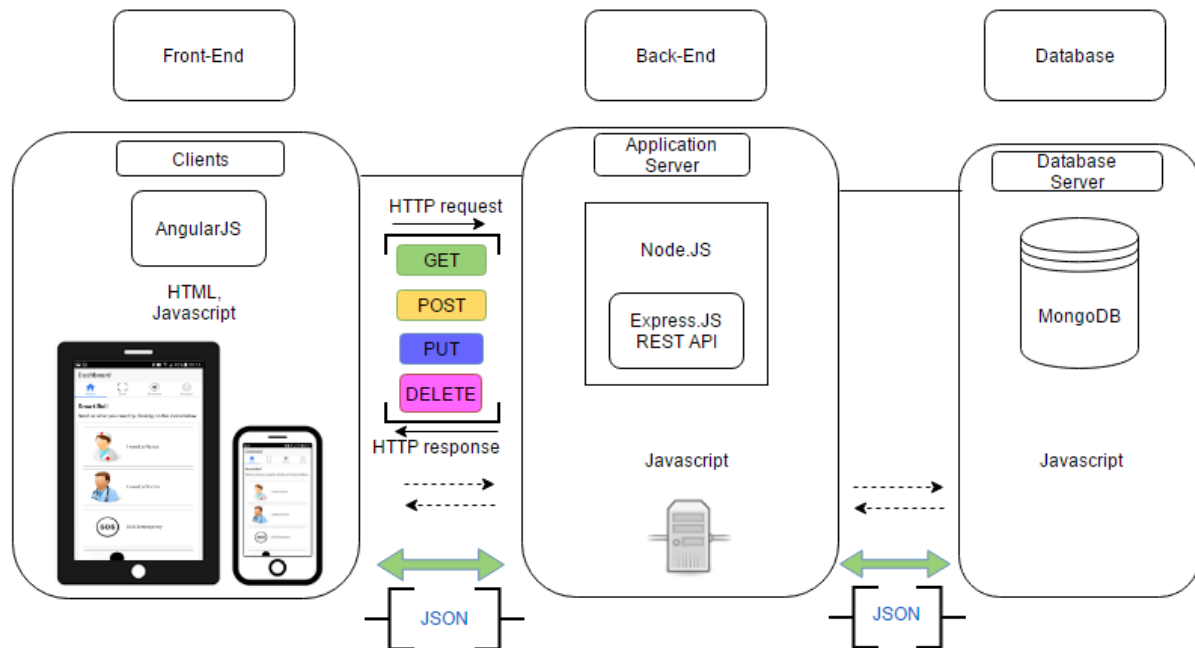


Figure 9 : aperçu de l'architecture de l'application

7.2 Le RESTful API server

Le serveur c'est un RESTful API server crée avec Node.js et Express.js. Il a l'avantage qu'il est extensible, modulable et facilement attaquable via le protocole http et le port 4000. Une fois le serveur est démarré il sera à l'écoute et prêt à recevoir des connections du client.

Dans la hiérarchie du projet il se trouve dans le dossier intitulé « server ».

Il est principalement composé d'un fichier app.js et de trois modèles alerts.js, beacon.js et staff.js représentant chacun le schéma et l'API de respectivement les alertes, les patients et le staff.

Chaque patient ou staff est associé à un beacon.

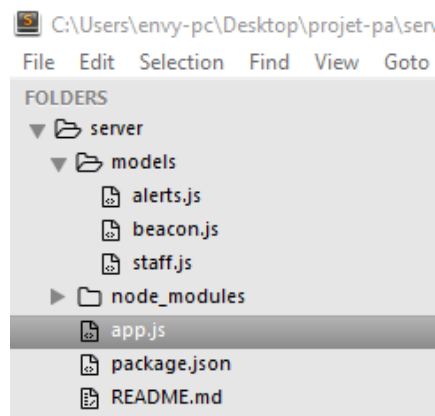


Figure 10 : arborescence du serveur

7.2.1 Les APIs

7.2.1.1 L'API beacons

Elle représente le patient.

L'URL adresse_IP :4000/api/beacons liste tous patients identifiés.

L'URL adresse_IP :4000/api/alerts/id_beacon liste le patient dont l'identifiant est id_beacon.

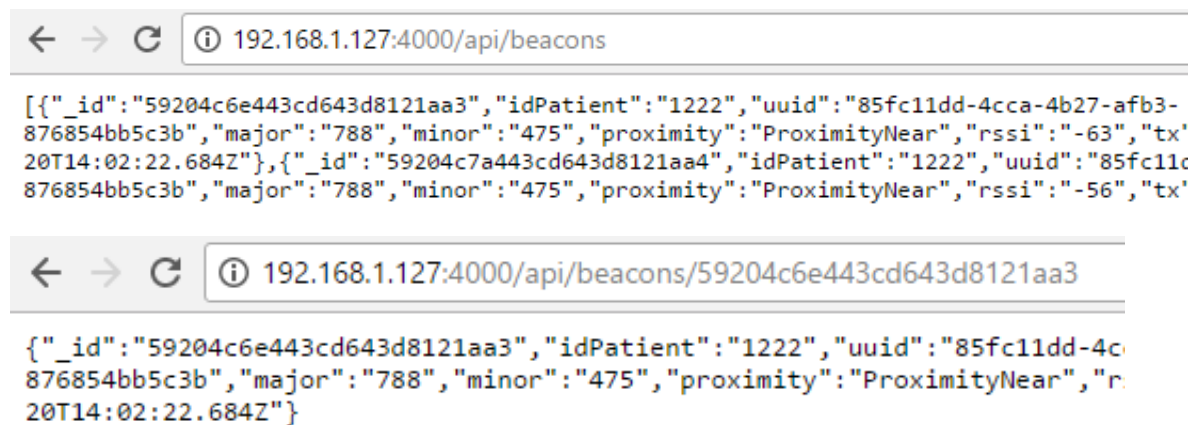


Figure 11 : l'API beacons (ou patient)

Les requêtes CRUD sont implémentées dans le fichier app.js du serveur.

Au début on récupère le modèle « Beacon » qui représente le schéma de la collections « beacons » dans la base de donnée « beaconstore » :

```
//Require Database
Beacon = require('./models/beacon');
```

Puis on effectue les opérations CRUD qui sont en terme de programmation get, post, put et delete. On trouve principalement les requêtes suivantes :

get('/api/beacons'): permet de retourner la liste de tous les beacons (tous les patients) :

```
app.get('/api/beacons', (req, res) => {
  Beacon.getBeacons((err, beacons) => {
    if(err){
      throw err;
    }
    res.json(beacons);
  });
});
```

get('/api/beacons/:_id') : permet de retourner un beacon (un patient) suivant son identifiant passé en paramètre:

```
app.get('/api/beacons/:_id', (req, res) => {
  Beacon.getBeaconById(req.params._id, (err, beacon) => {
    if(err){throw err;}
    res.json(beacon);
  });
});
```


post('/api/beacons') : permet d'ajouter un beacon (un patient) :

```
app.post('/api/beacons', (req, res) => {
  var beacon = req.body;
  Beacon.addBeacon(beacon, (err, beacon) => {
    if(err){
      throw err;
    }
    res.json(beacon);
  });
});
```

put('/api/beacons /:_id') : permet de mettre à jour un beacon (un patient) suivant son identifiant passé en paramètre:

```
app.put('/api/beacons/:_id', (req, res) => {
  var id = req.params._id;
  var beacon = req.body;
  Beacon.updateBeacon(id, beacon, {}, (err, beacon) => {
    if(err){throw err;}
    res.json(beacon);
  });
});
```

delete('/api/beacons /:_id') : permet de supprimer un beacon (un patient) suivant son identifiant passé en paramètre:

```
app.delete('/api/beacons/:_id', (req, res) => {
  var id = req.params._id;
  Beacon.removeBeacon(id, (err, beacon) => {
    if(err){throw err;}
    res.json(beacon);
  });
});
```

7.2.1.1.1 Le modèle Beacon

Techniquement c'est un fichier « beacon.js » dans le répertoire « models » du serveur. Il est constitué du schéma du « beacon » et cinq opérations get, getByld, add, update et delete (ci-dessous en détails).

Le schéma « beacon » c'est un objet JSON constitués de plusieurs attributs :

Un beacon est composé des attributs suivants :

idPatient : identifiant du patient

uuid, major, minor, proximity, rssi, tx, accuracy : les paramètres associés à un beacon (voir paragraphe « Les beacons »).

firstname : prénom du patient

name : nom du patient

age : âge du patient

room : numéro de la chambre du patient

message : le contenu de l'alerte qui l'a envoyée

codeMessage : un code qui peut avoir les trois valeurs suivantes :

- 0 : alerte destinée aux médecins
- 1 : alerte destinée aux infirmières
- : alerte destinée aux assistants

create_date : la date de la création du beacon.

SMART BELL - projet d'approfondissement

```
// Beacon Schema
const beaconSchema = mongoose.Schema({
  idPatient:{
    type: String
  },
  uuid:{
    type: String
  },
  major:{
    type: String
  },
  minor:{
    type: String
  },
  proximity:{
    type: String
  },
  rssi:{
    type: String
  },
  tx:{
    type: String
  },
  accuracy:{
    type: String
  },
  firstname:{
    type: String
  },
  name:{
    type: String
  },
  age:{
    type: String
  },
  room:{
    type: String
  },
  message:{
    type: String
  },
  codeMessage:{
    type: String
  },
  create_date:{
    type: Date,
    default: Date.now
  }
});
```

Les cinq opérations sont les suivantes :

getBeacons permet de récupérer tous les beacons de la base de donnée :

```
// Get Beacons
module.exports.getBeacons = (callback, limit) => {
  Beacon.find(callback).limit(limit);
}
```

getBeaconById permet de récupérer un beacon de la base de donnée selon son identifiant :

```
// Get Beacon
module.exports.getBeaconById = (id, callback) => {
  Beacon.findById(id, callback);
}
```

AddBeacon permet d'ajouter un beacon dans la base de donnée :

```
// Add Beacon
module.exports.addBeacon = (beacon, callback) => {
  Beacon.create(beacon, callback);
}
```

updateBeacon permet de mettre à jour un beacon :

```
// Update Beacon
module.exports.updateBeacon = (id, beacon, options, callback) => {
  var query = {_id: id};
  var update = {
    id: beacon.id,
    idPatient: beacon.idPatient,
    uuid: beacon.uuid,
    major: beacon.major,
    minor: beacon.minor,
    proximity: beacon.proximity,
    rssi: beacon.rssi,
    tx: beacon.tx,
    accuracy: beacon.accuracy,
    firstname: beacon.firstname,
    name: beacon.name,
    age: beacon.age,
    room: beacon.room,
    message: beacon.message,
    codeMessage: beacon.codeMessage
  }
  Beacon.findOneAndUpdate(query, update, options, callback);
}
```

removeBeacon permet de supprimer un beacon de la Base de donnée :

```
// Delete Beacon
module.exports.removeBeacon = (id, callback) => {
  var query = {_id: id};
  Beacon.remove(query, callback);
}
```

7.2.1.2 L'API staffbeacons

Elle représente le staff.

L'URL adresse_IP :4000/api/staffbeacons liste tous les staffs identifiés.

L'URL adresse_IP :4000/api/staffbeacons/id_staff liste tous le staff dont l'identifiant est id_staff.



Les requêtes CRUD sont pareilles que ceux de l'API beacons.

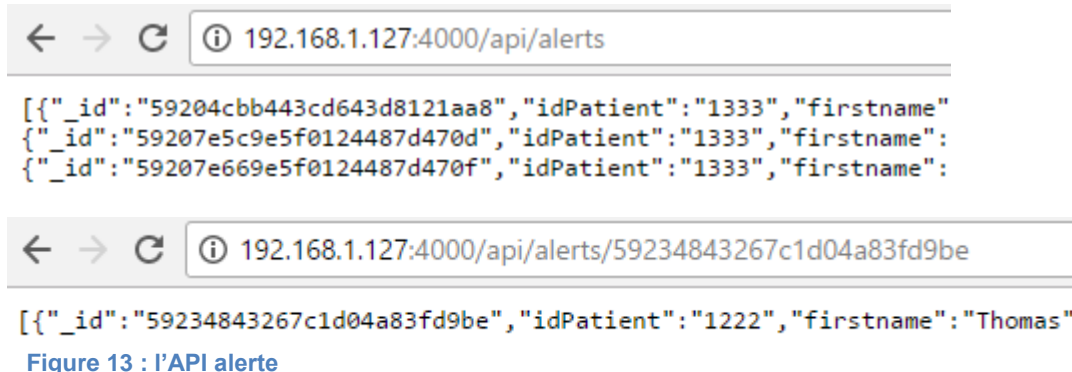
Le modèle « StaffBeacon » est pareil que celui de l'API beacons.

7.2.1.3 L'API alerts

Elle représente une alerte.

L'URL adresse_IP :4000/api/alerts liste tous les alerts signalés par le patient.

L'URL adresse_IP :4000/api/alerts/id_alert liste l'alerte dont l'identifiant est id_alert.



Les requêtes CRUD sont pareilles que ceux de l'API beacons.

Le modèle « Alerts » est pareil que celui de l'API beacons.

7.3 La base de donnée « MongoDB »

La base de donnée MongoDB « beaconstore » est composée de trois collections, alerts, beacons et staffbeacons.

7.3.1 Description des collections

La collection « alerts » :

alerts : représente tous les alertes que le patient envoi via l'application (SmartBell-patient).

Chaque alerte est composée des champs suivants :

- **_id**: identifiant de l'enregistrement dans la BD
- **idPatient**: identifiant du patient qui a envoyé l'alerte
- **firstname**: prénom du patient
- **name**: nom du patient
- **age**: âge du patient
- **room**: le numéro de la chambre du patient
- **message**: le contenu de l'alerte du patient (ex. je veux manger, j'ai mal, etc.)
- **codeMessage** : code du message (0 : besoin d'un médecin ; 1 : besoin d'une infirmière ; 2 : besoin d'un assistant)
- **create_date**: la date de la création de l'enregistrement

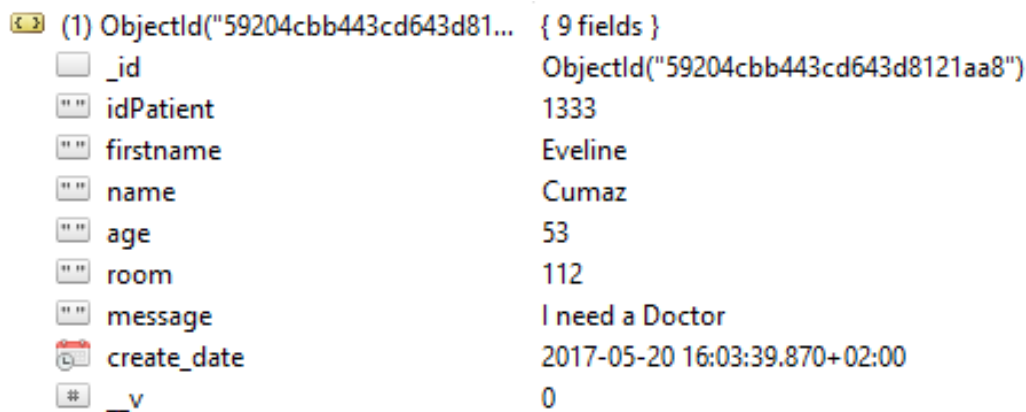


Figure 14 : aperçu de la collection alerts dans la base de donnée

La collection « beacons » :

beacons : représente tous les patients qui sont appariés à un beacon.

Chaque beacon est composé des champs suivants :

- **_id** : identifiant de l'enregistrement dans la BD
- **idPatient** : identifiant du patient
- **uuid** : identifiant univoque du producteur du beacon
- **major** : numéro majeur du beacon
- **minor** : numéro mineur du beacon
- **proximity**: proximité du beacon par rapport au patient
- **rss**: puissance en décibel du signal du beacon
- **tx**: puissance du signal à un mètre de l'appareil
- **accuracy**: rayon du cercle du signal
- **firstname**: prénom du patient
- **name**: nom du patient
- **age**: âge du patient
- **room**: numéro de la chambre du patient
- **create_date**: la date de la création de l'enregistrement











	_id	ObjectId("592b025244fad22c54bac2df")
	idPatient	1333
	firstname	Eveline
	name	Cumaz
	age	53
	room	112
	message	SOS Emergency
	codeMessage	0
	create_date	2017-05-28 19:01:06.411+02:00
	__v	0

Figure 15 : aperçu de la collection beacons dans la base de donnée

La collection « staffbeacons » :

staffbeacons : représente tous les membres du staff qui se sont appariés à un beacon.

Chaque staffbeacon (ou staff) est composé des champs suivants :

- _id : identifiant de l'enregistrement dans la BD
- idStaff : identifiant du staff
- staffRole : role du staff (médecin, infirmière, assistant)
- staffRoleCode : code du role (0 : médecin, 1 : infirmière, 2 : assistant)
- uuid : identifiant univoque du producteur du beacon
- major : numéro majeur du beacon
- minor : numéro mineur du beacon
- proximity: proximité du beacon par rapport au staff
- rssi: puissance en décibel du signal du beacon
- tx: puissance du signal à un mètre de l'appareil
- accuracy: rayon du cercle du signal
- firstname: prénom du staff
- name: nom du staff
- create_date: la date de la création de l'enregistrement

















	_id	ObjectId("592b021c44fad22c54bac2dd")
	idStaff	3222
	staffRole	Nurse
	staffRoleCode	1
	uuid	85fc11dd-4cca-4b27-afb3-876854bb5c3b
	major	662
	minor	359
	proximity	ProximityNear
	rssi	-71
	tx	-59
	accuracy	1.93
	firstname	Angella
	name	Matez
	responseToAlert	
	create_date	2017-05-28 19:00:12.484+02:00
	__v	0

Figure 16 : aperçu de la collection staffbeacons dans la base de donnée

Dans la figure ci-dessous nous montrons un aperçu de la base de donnée visualisé avec l'outil « Robomongo » :

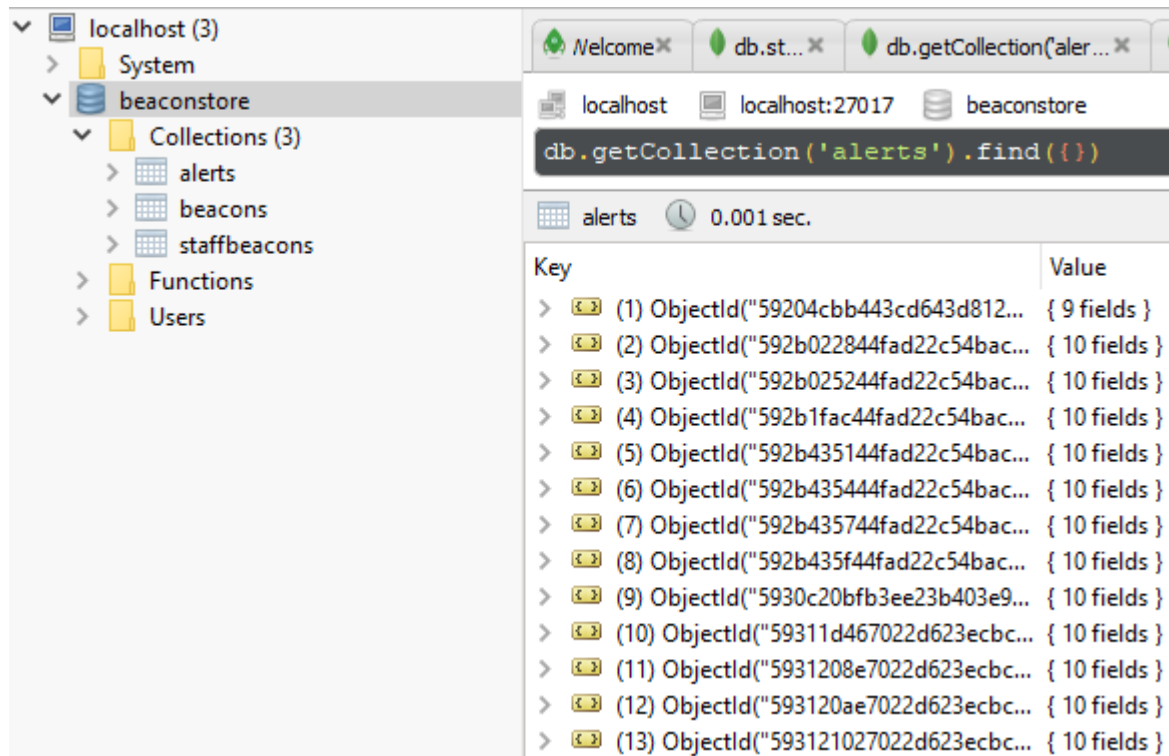


Figure 17 : aperçu de la base de donnée avec l'outil Robomongo

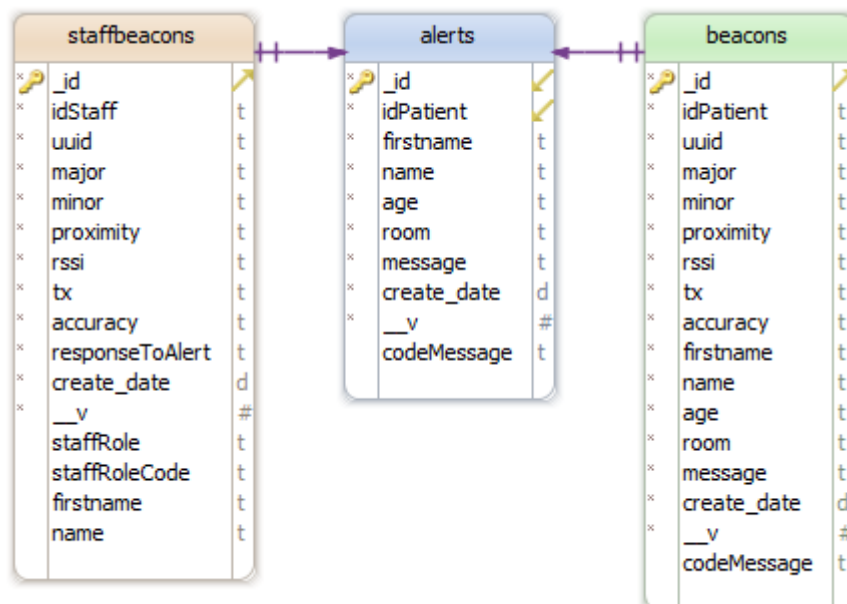


Figure 18 : aperçu du schéma de la base de donnée

7.4 Description du projet et l'application

Dans ce paragraphe nous expliquons la structure et l'arborescence de l'application (le client) développée avec le Framework Ionic. Le projet est composé de deux clients suivants :

- Le patient : nom de l'application SmartBell-patient, nom du projet smartbell.
- Le staff : nom de l'application SmartBell-staff, nom du projet staff.

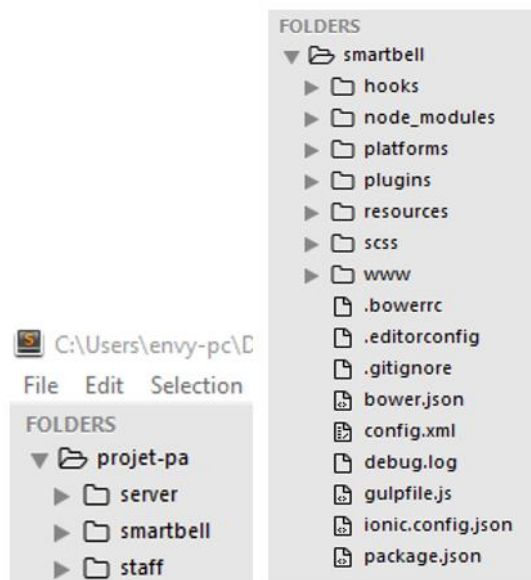


Figure 19 : arborescence du projet, arborescence du projet smartbell

7.4.1 Description du projet Ionic

Le projet « smartbell » :

C'est le dossier intitulé « smartbell ». Il est composé des dossiers et fichiers suivants :

- **node_modules** : regroupe tous les librairies qui ont été installées et utilisées. C'est via le package manager « npm » qu'on peut installer un module (ex. *npm install package_name*). Certains modules sont déjà installés par Ionic, d'autre défini dans le fichier package.json.
- **hooks** : dans ce dossier on peut mettre des scripts qui peuvent s'exécuter à différent stade:
 - **before_build** : ce qu'on veut exécuter avant le build
 - **after_build** : ce qu'on veut exécuter après le build
 - **before_plugin_add** : ce qu'on veut exécuter avant l'ajout de plugin
 - **after_plugin_add** : ce qu'on veut exécuter après l'ajout de plugin
- **platforms** : regroupe les plateformes installées (ex. Android, iOS, etc.)
- **plugins** : regroupe tous les plugins qui ont été installés (ex. plugin caméra, beacon, bare code scanner, etc.)
- **ressources** : les ressources utilisées (ex. icons, splash screen, etc.)
- **scss** : contient les variables par défaut pour le CSS (Cascading Style Sheets) de l'application.
- **www** : c'est l'application elle-même (HTML, CSS, JS, Librairies, Images, etc.)

- **bower (bower.json et .bowerrc)** : Bower est un outil de gestion de paquetage open-source pour le développement Web côté client. Il est facile d'installer et de mettre à jour des bibliothèques et des Framework en utilisant des outils de ligne de commande écrits par Node.js. Il est par défaut installer par Ionic.
- **ionic.config.json** : fichier de configuration pour Ionic (nom de l'application, id de l'application, etc.)
- **debug.log** : contient les logs de l'application en mode debug
- **config.xml** : fichier de configuration de l'application. Il est généré automatiquement par Ionic.
- **gulpfile.js** : c'est un task manager, il permet de réaliser plusieurs actions pour l'automatisation du build de l'application (ex. la génération de CSS depuis SCSS, la vérification de l'installation de git, la customisation du build, voire même du pre-build, etc.)
- **package.json** : regroupe les plugins qui ont été installés localement via *npm install plugin_name*

L'arborescence et la structure du projet staff est pareil que smartbell.

7.4.2 Description de l'application

Ce paragraphe décrit les principaux fichiers en commun de l'application « smartbell » et « staff ».

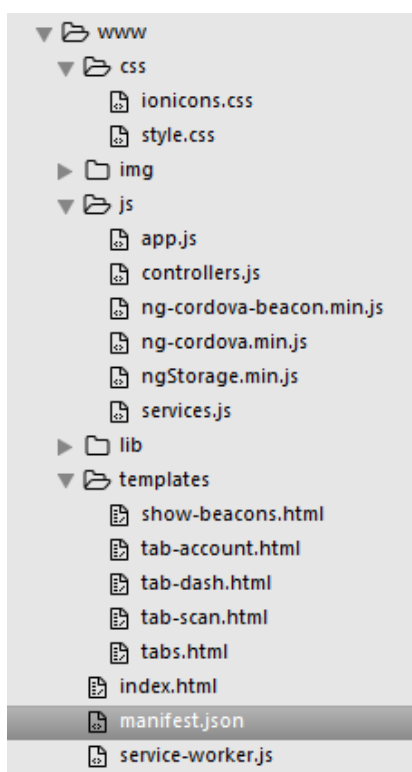


Figure 20 : arborescence de l'application smartbell – dossier www

Les principaux fichiers :

- **style.css** : CSS de l'application.
- **app.js** : fichier starter pour AngularJS, ici on définit les modules, services, contrôleurs, plugins et librairies indispensable à l'application pour fonctionner. Les configurations sont aussi définies dans ce fichier ainsi que le routage pour la navigation entre les différents onglets (ou views) de l'application.

- **controller.js** : c'est le fichier qui regroupe tous les contrôleurs de l'application. Il y a le contrôleur principale « ManageDataCtrl » et le contrôleur pour le scan du code bar « ScanCtrl ».
- **service.js** : ce fichier regroupe les services utilisés dans l'application. Un service en AngularJS sert à partager les informations entre les différents contrôleurs. On a quatre services (factory) :
 - **sessionService** : utilisé pour le local storage (Local Storage).
 - **socket** : utilisé pour l'échange des notification push entre le client patient, le serveur et le client staff.
 - **SharedDataBeaconsController** : utilisé pour partager les informations du beacons qui correspond au patient (idPatient, room, age, name, firstname, uuid, proximity, etc...).
 - **SharedDataCtrl** : utilisé pour partager les beacons dans les alentours et ainsi le beacon le plus proche du patient.
- **ngStorage.min.js** : plugin AngularJS pour le stockage local (Local Storage).
- **ng-cordova.min.js** : librairie Cordova permettant l'interaction avec les composants principaux des devices mobiles (la caméra, le Bluetooth, la localisation du device, etc.).
- **ng-cordova-beacon.min.js** : plugin Cordova permettant l'interaction avec les beacons.
- **index.html** : c'est le fichier HTML principal de l'application.
- **tabs.html** : c'est une région supérieure qui regroupe les différents onglets (ou views).
- **tab-dash.html** : c'est la page Home.
- **tab-scan.html** : c'est la page Scan, elle contient un bouton pour scanner le Code Bar.
- **show-beacons.html** : c'est la page qui affiche les beacons dans les alentours.
- **tab-account.html** : c'est la page qui affiche les informations du patient (Patient ID, Name, Firstname, Room, Age). Il a la possibilité de supprimer ces informations via un bouton intitulé « Delete Local Storage ».
- **manifest.json** : fichier manifeste de l'application, contient des informations statiques comme l'URL de démarrage (start_url), l'icône de l'application, etc.
- **service-worker.js** : c'est un script qui s'exécute dans un thread distinct du reste d'une application. Cela signifie qu'un Service Worker peut effectuer certaines actions en arrière-plan.

L'application staff est pareil, mis à part quelques différences des fonctionnalités dans le contrôleur.

7.4.2.1 Les fichiers en détails

Ce paragraphe décrit les détails des principaux fichiers de l'application. La description concerne les deux application patient et staff. Il s'agit de montrer un aperçu de l'essentiel du code afin de pouvoir comprendre les fonctions capitales de l'application.

Le cœur de l'application se trouve dans le fichier app.js et controller.js et service.js, situé dans l'arborescence suivante : « chemin_vers_le_projet\staff\www\js\ ».

7.4.2.1.1 Le fichier app.js

C'est le module de lancement de l'application. On trouve aussi les configurations de routages :

```
// Each tab has its own nav history stack:
.state('tab.dash', {
  url: '/dash',
  views: {
    'tab-dash': {
      templateUrl: 'templates/tab-dash.html',
      controller: 'ManageDataCtrl'
    }
  }
})
etc...
```

On trouve aussi d'autres configurations comme pour l'adresse IP et le port :

```
.constant('servicesConfig', (function() {
  var urlAPI = 'http://192.168.1.127';
  var port = '4000';
  return {
    urlAPI: urlAPI,
    port: port
  }
}))()
```

7.4.2.1.2 Le fichier service.js

Permet d'échange de données entre différents contrôleurs. On trouve les quatre factories suivants :

SharedDataCtrl : utilisé pour partager les beacons dans les alentours et ainsi le beacon le plus proche du patient.

```
.factory('SharedDataCtrl', function() {
  return {
    beacons: {},
    nearestBeacon: '',
    alerts: {},
    receivedCodeMessage: '',
    latestAlert: ''
  };
})
```

SharedDataBeaconsController : utilisé pour partager les informations du beacon qui correspond au patient (idPatient, room, age, name, firstname, uuid, proximity, etc...) :

```
.factory('SharedDataBeaconsController', function() {
  return {
    idStaff: '',
    staffRole: '',
    staffRoleCode: '',
    uuid: '',
    major: '',
    minor: '',
    proximity: '',
    rssi: '',
    tx: '',
    accuracy: '',
    firstname: '',
    name: '',
    responseToAlert: ''
  };
})
```

socket : utilisé pour l'échange des notification push entre le client patient, le serveur et le client staff.

```
.factory('socket', ['$rootScope', 'servicesConfig', function($rootScope,
servicesConfig) {
  var ipAddress = servicesConfig.urlAPI;
  var socket = io.connect(ipAddress+':4000');
  return {
    on: function(eventName, callback){
      socket.on(eventName, callback);
    },
    emit: function(eventName, data) {
      socket.emit(eventName, data);
    }
  };
}]);
```

sessionService : utilisé pour le local storage (Local Storage) :

```
.factory('sessionService', ['$window', '$http', '$localStorage',
function($window, $http, $localStorage) {
  return {
    set:function(key, value){
      return localStorage.setItem(key,JSON.stringify(value));
    },
    get:function(key) {
      return JSON.parse(localStorage.getItem(key));
    },
    destroy:function(key){
      return localStorage.removeItem(key);
    },
    clear:function(){
      return localStorage.clear();
    },
  };
}]);
```

7.4.2.1.3 Le fichier controller.js

C'est le fichier des contrôleurs de l'application. Composé de deux contrôleurs :

ManageDataCtrl : c'est le contrôleur responsable de la gestion des beacons et du reste des opérations. Principalement on trouve les fonctions suivantes :

initBeaconsPlugin(): permet d'initialiser les beacons :

```
//Search for the neighbor beacons
$scope.initBeaconsPlugin = function() {
  $scope.beacons = {};
  $ionicPlatform.ready(function() {
    $cordovaBeacon.requestWhenInUseAuthorization();
    $rootScope.$on("$cordovaBeacon:didRangeBeaconsInRegion",
    function(event, pluginResult) {
      var uniqueBeaconKey;
      for(var i = 0; i < pluginResult.beacons.length; i++) {
        uniqueBeaconKey = pluginResult.beacons[i].uuid + ":" +
        pluginResult.beacons[i].major + ":" +
        pluginResult.beacons[i].minor;
        $scope.beacons[uniqueBeaconKey] = pluginResult.beacons[i];

        //Store detected beacons
        SharedDataCtrl.beacons = $scope.beacons;
      }
    }
  )
}
```

```

    $scope.$apply();
  });

  $cordovaBeacon.startRangingBeaconsInRegion($cordovaBeacon.createBeaconRegion(
    "iBeacon", "85FC11DD-4CCA-4B27-AFB3-876854BB5C3B"));
  });
};

```

On trouve la fonction `getNearestBeacon` qui permet de trouver le beacon le plus proche :

```

//Get nearest beacon to the patient (must be the beacon in the room)
$scope.getNearestBeacon = function() {
  $scope.mapBeacons = {};
  for (var p in SharedDataCtrl.beacons) {
    $scope.mapBeacons[SharedDataCtrl.beacons[p]["accuracy"]] = p;
  }
  var minOfAccuracy = Math.min.apply(null, Object.keys($scope.mapBeacons));
  var idMinOfAccuracy = $scope.mapBeacons[minOfAccuracy];

  SharedDataBeaconsController.uuid =
  SharedDataCtrl.beacons[idMinOfAccuracy].uuid;
  SharedDataBeaconsController.major =
  SharedDataCtrl.beacons[idMinOfAccuracy].major;
  SharedDataBeaconsController.minor =
  SharedDataCtrl.beacons[idMinOfAccuracy].minor;
  SharedDataBeaconsController.proximity =
  SharedDataCtrl.beacons[idMinOfAccuracy].proximity;
  SharedDataBeaconsController.rssi =
  SharedDataCtrl.beacons[idMinOfAccuracy].rssi;
  SharedDataBeaconsController.tx =
  SharedDataCtrl.beacons[idMinOfAccuracy].tx;
  SharedDataBeaconsController.accuracy =
  SharedDataCtrl.beacons[idMinOfAccuracy].accuracy;
};

```

Le socket pour la réception de notification push :

```

//Get notification
socket.on('notification2', function(data) {
  //alert(data);
  $scope.incomingData = data;
  $scope.pushLocalNotification();
});

```

ScanCtrl : c'est le contrôleur responsable de l'opération du scan di Code Bar. Principalement on a la fonction `Cordova` qui permet de lancer le scan de la caméra :

```

//Scan the Bar-Code
$scope.scanBarcode = function() {
  $cordovaBarcodeScanner.scan().then(function(imageData) {
    $scope.retrieveQrCodeData(imageData);
  }, function(error) {
    console.log("An error happened -> " + error);
  });
};

```

7.5 Communication client-serveur « socket.io »

Il s'agit des connections bidirectionnelles entre le client et le serveur qui permettent une communication en temps réel. On utilise socket.io qui est un module de Node.js permettant de créer des Web Sockets.

1. L'installation se fait via le package manager npm :

```
npm install socket.io
```

2. Nous utilisons la méthode `Server()` du module natif `http` pour créer un serveur `http` et nous utilisons l'application `Express.js` pour gérer le mécanisme requête/réponse :

```
var server = require('http').Server(app);
```

3. Par la suite, nous passons au serveur `Socket.io` qui est associé au même serveur `http` à travers la variable (`server`) :

```
var io = require('socket.io')(server);
```

4. On ajoute le code suivant pour que le socket écoute les connexions qui seront établies à travers le client et servi par `Express.js` :

```
io.on('connection', function(socket) {
  console.log('socket: someone is connected');

  socket.on('disconnect', function() {
    console.log('socket: someone is disconnected');
  });

  //Socket from patient
  socket.on('new-alert-from-patient', function(data) {
    console.log('new-alert-from-patient...!');
    io.emit('notification', data.codeMessage);
    console.log(JSON.stringify(data.codeMessage));
  });

  //Socket from staff
  socket.on('staff-take-alert-coming-from-patient', function(data) {
    console.log('staff-take-alert-coming-from-patient...!');
    io.emit('notification2', data);
    console.log(JSON.stringify(data));
  });
});
```

5. Enfin on lance les "deux serveurs" sur le port 4000 pour que notre application combinée **HTTP + WebSocket** soit disponible à l'adresse `ipExtern`:

```
server.listen(4000, ipExtern, function() {
  var port = server.address().port;
  console.log("Server listening on http://%s:%s", ipExtern, port);
});
```

7.6 Les notifications « Push Notification »

Une notification push est un message envoyé à l'utilisateur d'un smartphone ou d'une tablette, ce message est lié à une application installée sur l'appareil. L'avantage de la notification push c'est que le message est reçu par l'utilisateur même si l'application est fermée. Les notifications push sont importants pour le projet car il ne suffit pas uniquement d'envoyer une alerte sinon le staff risque de ne pas remarquer. L'application fonctionne de la façon suivante :

Lorsque le patient envoie une alerte l'application envoie une notification push au staff.
Lorsque le staff envoie une confirmation de prise en charge du patient en cliquant sur le bouton « Take IT », l'application envoie une notification au patient en indiquant le message ainsi que le rôle, le prénom et le nom du staff qui viendra pour le soigner.
Lorsqu'on clique sur la notification l'application s'ouvre.

Aperçu de la notification chez le staff :

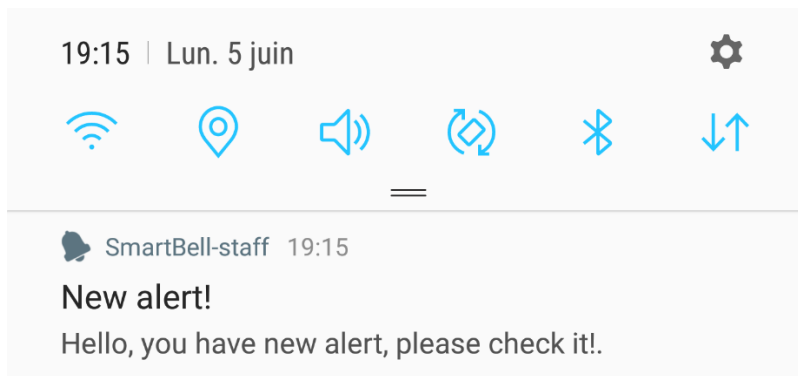


Figure 21 : aperçu de la notification push côté staff

Aperçu de l'alerte chez le staff :

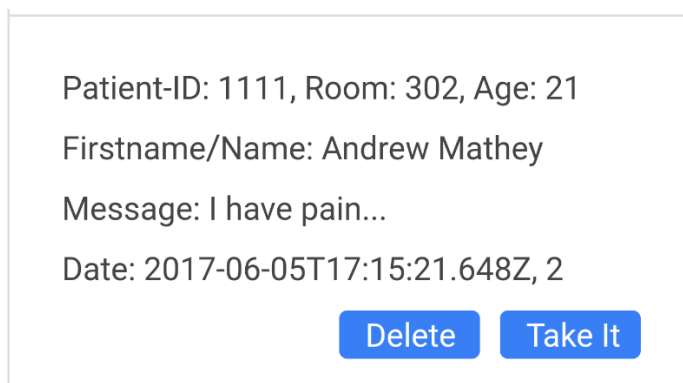


Figure 22 : aperçu de l'alerte côté staff

Aperçu de la notification push côté patient :

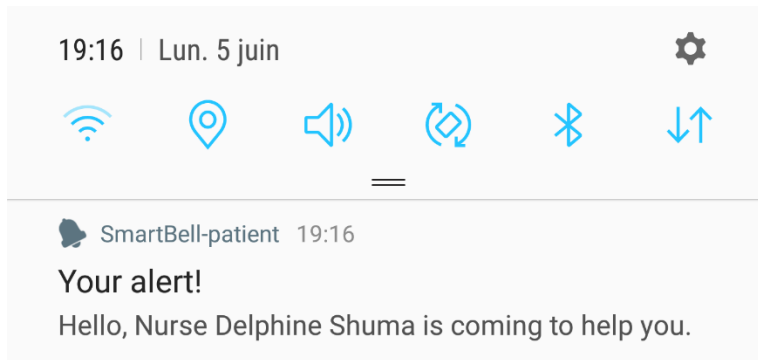


Figure 23 : aperçu de la notification push côté patient

L'implémentation s'est fait avec le plugin Apache Cordova Local Notification. La notification push côté patient est une fonction implémentée dans le contrôleur « ManageDataCtrl » du fichier « controller.js » du projet « smartbell » :

```
$scope.pushLocalNotification = function() {
    $cordovaLocalNotification.isScheduled("123").then(function(isScheduled)
    {
        });
    var alertTime = new Date();
    alertTime.setSeconds(alertTime.getSeconds() + 10);
    $cordovaLocalNotification.add({
        id: "123",
        date: alertTime,
        message: "Hello, " + $scope.incomingData + " is coming to help you.",
        title: "Your alert!",
        autoCancel: true
    }).then(function() {
        console.log("The notificationw was set");
    });
};
```

1. On ajuste le temps en secondes avec la variable alertTime :
alertTime.setSeconds(alertTime.getSeconds() + 10);
2. Via le plugin Cordova on fait les cinq opérations suivantes :
id : identifiant de la notification
date : le temps entre le reçu et le push (c'est la variable alertTime)
message : le message y compris l'identité du staff qui récupérée avec la variable incomingData
title : le titre de la notification push
autoCancel : la notification est automatiquement annulée lorsque le patient clique là-dessus.

La notification push côté staff est une fonction implémentée dans le contrôleur « ManageDataCtrl » du fichier « controller.js » du projet « staff » :


```
//Get notification when alert coming from patient
$scope.getLocalNotification = function() {
    if (JSON.stringify($scope.staffRoleCode) ===
JSON.stringify($scope.receivedCodeMessage)) {

    $cordovaLocalNotification.isScheduled("456").then(function(isScheduled) {
        });
        var alertTime = new Date();
        alertTime.setSeconds(alertTime.getSeconds() + 10);
        $cordovaLocalNotification.add({
            id: "456",
            date: alertTime,
            message: "Hello, you have new alert, please check it!.",
            title: "New alert!",
            autoCancel: true
        }).then(function() {});
    }
};
```

7.7 Algorithme de distribution des alertes

Le but d'une alerte est de renvoyer la personne adéquate pour soigner, soulager et répondre aux besoins du patient. Lorsque le patient envoie une alerte celle-ci est renvoyée vers le staff adéquat. Or chaque staff a un rôle différent qui peut être médecin, infirmier, assistant, aide-soignant, etc. On a choisi de faire de telle sorte qu'un type d'alerte soit associée à un rôle du staff.

Dans une première version on a choisi d'envoyer l'alerte à tous les membres du staff qui ont le même rôle, et qui ont préalablement installés l'application « staff » et sont identifiés.

- La première version de l'algorithme fonctionne de la façon suivante :
- Si le patient a mal on envoie l'alerte à tous les infirmiers identifiés (staff nurse).
- Si le patient demande du secours on envoie l'alerte à tous les médecins identifiés (staff doctor).
- Si le patient veut manger on envoie l'alerte à tous les infirmiers identifiés (staff nurse).
- Si le patient a besoin de divertissement on envoie l'alerte à tous les assistants identifiés (staff assistant).
- Si le patient a besoin d'un infirmier on envoie l'alerte à tous les infirmiers identifiés (staff nurse).
- Si le patient a besoin d'un médecin on envoie l'alerte à tous les médecins identifiés (staff doctor).
- Si le patient a besoin de faire ces besoins (aller aux toilettes) on envoie l'alerte à tous les infirmiers identifiés (staff nurse).

Ces besoins sont affichés sous forme d'une liste cliquable dans l'interface graphique de l'application (figure ci-dessous).

Selon le besoin de l'établissement hospitalier, il est tout à fait possible d'ajouter de nouveaux rôles comme secrétaire, stagiaire, etc.

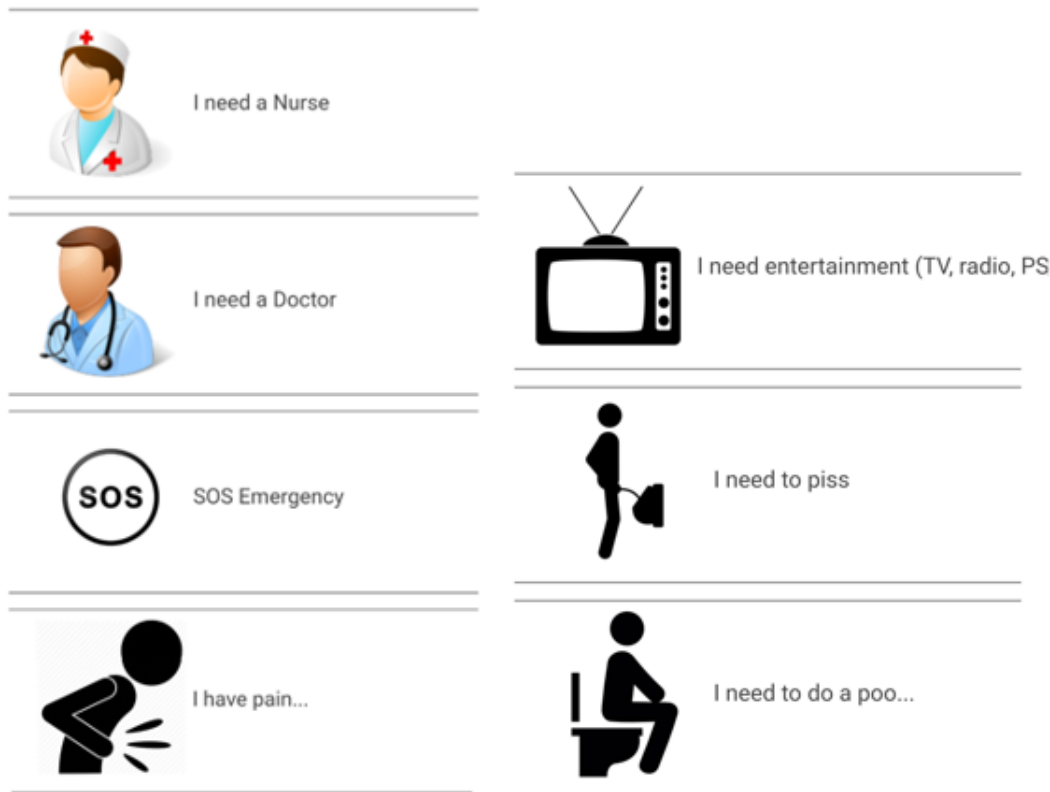


Figure 24 : liste des besoins du patient

7.8 Test

Implicitement des tests ont été réalisés régulièrement avec un émulateur Android, avec un navigateur Web Google Chrome et avec un smartphone Samsung Galaxy S6. En effet, Ionic permet d'effectuer plusieurs types de test en ligne de commande.

Les tests qui ont été réalisés sur le smartphone ont permis de constater plus de bugs que ceux réalisés avec l'émulateur et avec le navigateur.

Les différentes commandes pour tester sont les suivantes :

Navigateur test: *ionic serve -l* (le caractère l signifie livereload)

App native test: *ionic run android -l*

En cas où le smartphone n'est pas relié via le câble USB l'émulateur d'Android démarre.

7.9 Problèmes rencontrés

Ce sont plutôt des difficultés remarquées durant le développement et qui ont été résolues par la suite. Citons par exemple le problème du local storage qui ne fonctionne pas en mode test (*ionic run android -l*), il faut absolument que l'application soit déployée (*ionic run android --save*) sinon le stockage local ne fonctionne pas et on ne se rend pas compte qu'après plusieurs heures.

Mis à part cet difficulté, il n'y a pas eu de grand problème spéciaux à signaler.

8 Installation et déploiement

Afin que le système puisse fonctionner il faut installer les prérequis suivants :

Du côté client il y a aucun prérequis, il suffit juste d'installer l'exécutable (.apk pour Android ou .ipa pour iOS).

Remarque : pour générer l'exécutable .ipa pour la plateforme iOS il faut un ordinateur MacOS car le SDK iOS d'Apple ne fonctionne actuellement pas sur Windows.

Le client et le serveur doivent être dans le même réseau. Si vous changez de réseau il est indispensable que vous changiez l'adresse IP du client (patient et staff) et du serveur. (Voir paragraphe Changement de l'adresse IP dans les paragraphes « Démarrage du serveur » et « Démarrage du client » ci-dessous).

Du côté administrateur système (ou Back-End) il faut installer les outils suivants :

- **Node.js** : télécharger le depuis l'adresse suivante : <https://nodejs.org/en/>
Choisissez la version convenable puis cliquez sur « Download ». Une fois terminer, suivez la procédure d'installation en choisissant les configurations par défaut.
- **MongoDB** : télécharger le depuis l'adresse suivante : <https://www.mongodb.com/download-center?jmp=nav#community>
Choisissez la version convenable puis cliquez sur « Download ». Une fois terminer, suivez la procédure d'installation en choisissant les configurations par défaut.
Afin de pouvoir créer le répertoire de donnée « data directory » (sur Windows C:\data\db) dont MongoDB a besoin pour stocker les données veuillez suivre les instructions sur la page suivante : <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>.
- Un outil de ligne de commande, comme CMD ou PowerShell sur Windows, sinon vous pouvez utiliser Git (<https://git-scm.com/downloads>).

Stocker le dossier « server » dans un répertoire sur votre disque dure.

A ce stade le système est prêt à démarrer !

8.1 Démarrage du serveur

1. Ouvrez votre ligne de commande.
2. Placez-vous dans le répertoire suivant : « C:\Program Files\MongoDB\Server\3.4\bin »

Puis tapez : **mongo**

(Ceci permet de démarrer le serveur de MongoDB)

3. Placez-vous dans le répertoire suivant : « C:\data\db »
Puis tapez : **mongod**
(mongod est le processus de daemon primaire pour le système MongoDB. Il gère les demandes de données, gère l'accès aux données et effectue des opérations de gestion de fond.)
4. Placez-vous à la racine du dossier server, puis tapez **nodemon**.

A ce stade le système fonctionne et le serveur est à l'écoute sur le port 4000.

8.1.1 Changement de l'adresse IP

Pour le serveur il faut modifier l'adresse IP qui se trouve à la ligne 11 du fichier app.js (*var ipExtern*) dans le répertoire suivant : « \server\ ».

8.2 Démarrage du client

Premièrement télécharger l'exécutable (.apk pour Android ou .ipa pour iOS).

Dans le projet l'exécutable d'Android se trouve dans le répertoire suivant :

C:\Users\envy-pc\Desktop\projet-pa\staff\platforms\android\build\outputs\apk\android-debug.apk

Copiez-le sur votre smartphone puis lancez-le et suivez la procédure d'installation par défaut.

L'application est à présent installer il suffit de cliquer là-dessus pour la démarrer (photo ci-dessous).

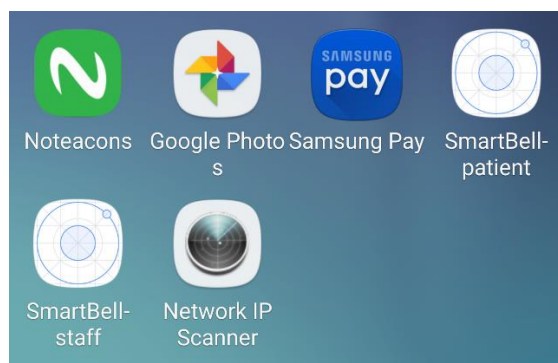


Figure 25 : l'application client (patient et staff) installée sur le Smartphone

8.2.1 Modification de l'adresse IP

Pour le client patient il faut modifier l'adresse IP qui se trouve dans le fichier app.js dans le répertoire suivant : « \staff\www\js\ » (var urlAPI).

Pour le client staff il faut modifier l'adresse IP qui se trouve dans le fichier app.js dans le répertoire suivant : « \smartbell\www\js\ » (var urlAPI).

8.3 Compilation de l'application

Lorsque vous apportez des modifications au code source il est indispensable de refaire un Build. Placez-vous dans le répertoire du projet et ouvrez votre ligne de commande.

Pour générer l'exécutable tapez :

```
Ionic build android
```

L'exécutable se trouve dans le dossier :

```
chemin_vers_la_racine_du_projet\platforms\android\build\outputs\apk\android-debug.apk
```

Pour générer l'exécutable et déployer l'application en une seule commande, tapez :

```
Ionic run android --save
```

9 Améliorations

9.1 L'algorithme de distribution des alertes amélioré

Dans une deuxième version de l'algorithme dont nous n'avons pas pu l'implémenter par manque de temps, nous avons développé la machine d'état de l'algorithme dont on va expliquer dans la suite du paragraphe. Le structogramme de l'algorithme se trouve au paragraphe « Structogramme distribution des alertes » dans les annexes.

Dans cette version nous envoyons l'alerte à un seul staff et non pas à tous les membres du staff. Cette opération est facile car on a déjà les paramètres de localisation du staff qui sont stockés dans la base de données (uuid, major, minor, proximity, rssi, tx et accuracy).

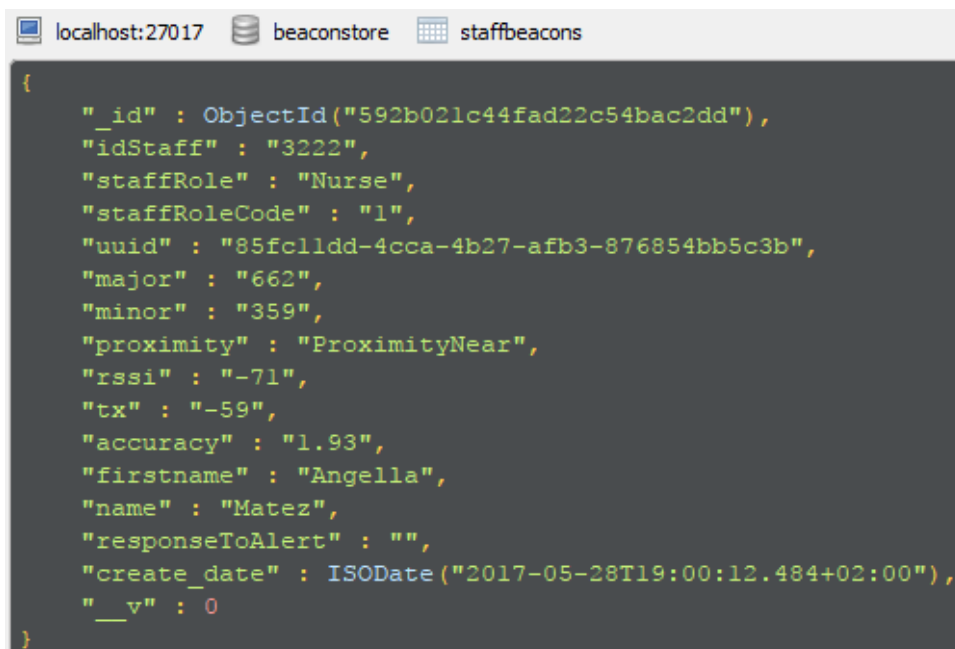


Figure 26 : paramètres de localisation d'un staff dans la base de données

Techniquement, dans le fichier controllers.js du staff et dans le contrôleur « ManageDataCtrl » on met à jour tous les 10 secondes la localisation du staff, cette opération se fait à la ligne suivante :

```
//We update Nearest Beacons  
$interval($scope.getNearestBeacon, 10000);
```

Mais on ne la met pas à jour dans la base de données. Pour la mettre à jour il suffit de récupérer l'identifiant du staff et de le remplacer par « _idStaff » dans le code suivant :

```
$http.post($scope.ipAddress+':4000/api/staffbeacons/:_idStaff',  
$scope.finalBeacon)  
  .success(function(response){  
    //Reload app  
    $window.location.reload();  
  })  
  .error(function (data, status){  
    alert("ERROR when sending message, sorry for this inconvenient, please  
try again or call the Staff");  
  });
```

Cette opération devrait se faire pour le patient aussi, car dans certain cas il peut arriver qu'il change de chambre.

A ce stade la localisation du patient et du staff sera à jour en temps réel.

9.1.1 L'algorithme version 2

Lorsque le patient envoie une alerte (par exemple besoin d'un médecin) on va faire les étapes suivantes selon le degré de priorité.

Cas prioritaire : le patient a mal, le patient demande de secours, etc.

Cas moins prioritaire : le patient a besoin de divertissement, aller aux toilettes, manger, etc.

9.1.1.1 Si le cas est prioritaire

1. Récupérer la localisation récente du patient.
2. Chercher N staffs du même rôle les plus proches du patient (exemple N médecins).
3. Chercher N staffs d'un rôle différent (par exemple infirmier ou aide-soignant) les plus proches du patient (rôle = rôle - 1, on descend dans la hiérarchisation des rôles médecin, médecin stagiaire, infirmier, soignant, aide-soignant, assistant, secrétaire, etc.).
4. Envoyer une alerte au premier médecin :
 - Si pas de confirmation du 1^{er} médecin, attendre 1 minute puis envoyer au deuxième médecin.
 - Si pas de confirmation du 2^{ème} médecin, attendre 1 minute puis envoyer au troisième médecin.
 - Répéter jusqu'au Nième médecin.
 - Si pas de confirmation du Nième médecin, attendre 1 minute puis commencer à envoyer un autre rôle de staff (rôle = rôle - 1).
5. Envoyer une alerte au premier infirmier :
 - Si pas de confirmation du 1^{er} infirmier, attendre 1 minute puis envoyer au deuxième infirmier.
 - Si pas de confirmation du 2^{ème} infirmier, attendre 1 minute puis envoyer au troisième infirmier.
 - Répéter jusqu'au Nième infirmier.
 - Si pas de confirmation du Nième infirmier, attendre 1 minute puis commencer à envoyer un autre rôle de staff (rôle = rôle - 1).
6. Répéter en descendant d'un rôle jusqu'à la secrétaire.
7. Si pas de réponse de confirmation d'aucun on envoie l'alerte à tous les rôles staff (tous les médecins, tous les infirmiers, tous les assistants, etc.).

9.1.1.2 Si le cas est moins prioritaire

1. Récupérer la localisation récente du patient.
2. Chercher N staffs du même rôle les plus proches du patient (exemple N assistants).
3. Envoyer une alerte au premier assistant :
 - Si pas de confirmation du 1^{er} assistant, attendre 1 minute puis envoyer au deuxième assistant.
 - Si pas de confirmation du 2^{ème} assistant, attendre 1 minute puis envoyer au troisième assistant.
 - Répéter jusqu'au Nième assistant.

Rechercher N nouveaux staffs du même rôle les plus proches du patient.

Répéter étapes 1, 2 et 3 jusqu'à la confirmation.

Dans la figure du paragraphe « Structogramme distribution des alertes » nous montrons un structogramme de l'algorithme.

9.2 Esthétique et interaction

L'application actuelle dispose d'interface graphique simple. En effet, on n'était pas focalisé sur l'aspect esthétique mais plutôt sur le côté fonctionnel. Il serait intéressant de donner un peu plus d'élégance en ajoutant un peu plus de CSS. Pour enrichir l'expérience utilisateur, on pourra ajouter plus d'interaction entre le client et l'application, comme les notifications popup, les messages de confirmation, et de l'animation.

Le Framework Ionic dispose d'ores et déjà des bibliothèques nécessaires, il suffit d'implémenter le code.

10 Conclusion

A l'issue de ce projet, on peut tirer un bilan positif, les objectifs fixés par ce dernier sont presque tous acquis. Travailler dans le domaine du développement mobile fut très intéressant, enrichissant et surtout d'une totale nouveauté pour moi. Ce que j'ai le plus apprécié, c'est que ce travail n'était pas axé sur un seul objectif, mais qu'il était à plusieurs tâches, en particulier les beacons, le développement cross-plateforme avec Ionic et le développement d'application mobile moderne avec la MEAN Stack.

Pour finir, autant le cahier des charges que le planning, ont été bien respectés, ce qui est très satisfaisant. Quant à l'algorithme de distribution des alertes, l'application dispose déjà d'une première version de base fonctionnelle, une deuxième version plus intelligente a été mise en place sous forme de structogramme.

Dans l'ensemble, je suis content du déroulement du projet ainsi que de son résultat. J'ai beaucoup apprécié réaliser ce projet.

10.1 Date et signature

Lausanne, le vendredi 9 juin 2017



Mahmoud Assaf

11 Références

11.1 Documentations




- [1] Smart Beacon, <http://www.smartbeacon.eu/>
- [2] iBeacon, <https://developer.apple.com/ibeacon/>
- [3] Eddystone, <https://developers.google.com/beacons/>
- [4] AltBeacon, <http://altbeacon.org/>
- [5] Evo Things, <https://evthings.com/>
- [6] Pub Nub, <https://www.pubnub.com/>
- [7] AngularJS, <https://angularjs.org/>
- [8] Ionic Framework, <https://ionicframework.com/>
- [9] Mean Full Stack, <http://mean.io/>
- [10] Express.js, <http://expressjs.com/>
- [11] Node.js, <https://nodejs.org/en/>
- [12] Cordova, <http://cordova.apache.org/>
- [13] MongoDB, <https://www.mongodb.com/>
- [14] Cordova, <http://cordova.apache.org/>
- [15] Easy Tag, Génération de VCard avec QR Code, <http://www.easytag.pro/generateur-qr-code>
- [16] Balise iBeacon 26 cas concrets d'utilisation !, Christophe Robinet, <https://fr.slideshare.net/christopherobinet/balises-ibeacon-26-cas-concrets-dutilisation>
- [17] Using Local Notifications in Ionic Framework App, NIC RABOY, 2015, <http://devdactic.com/local-notifications-ionic/>
- [18] iBeacons, La technologie digitale qui surclasse minority report, Blog Big Data & Digital, par ERIC VAN CUTSEM, 23 février 2016, <http://blog.businessdecision.com/digital/2016/02/ibeacons-la-technologie-digitale-qui-surclasse-minority-report/>
- [19] Beacons, iBeacons, Eddystone, How it works, Onyx Beacon, <http://www.onyxbeacon.com/>
- [20] Building Cross-Platform iBeacon Apps for iOS, Android and Windows with C# and Xamarin, Vincent Hoogendoorn, 2014, <http://vincenth.net/blog/archive/2014/04/24/building-cross-platform-ibeacon-apps-for-ios-android-and-windows-with-c-and-xamarin.aspx>
- [21] Ionic vs PhoneGap vs Xamarin, stackshare, <https://stackshare.io/stackups/xamarin-vs-phonegap-vs-ionic>
- [22] Ionic vs React Native vs Xamarin, stackshare, <https://stackshare.io/stackups/ionic-vs-react-native-vs-xamarin>
- [23] Comparing cross-platform development approaches for mobile applications, Sebastian Hanschke, http://www.academia.edu/3009209/Comparing_cross-platform_development_approaches_for_mobile_applications
- [24] MEAN Stack RESTful API tutorial – Using MongoDB, ExpressJS, AngularJS and NodeJS together, Learn Coding Tutorials, <https://www.youtube.com/watch?v=kHV7qOHvNdk&t=915s>
- [25] Tutoriel pour la prise en main de la MEAN Stack, Maxime Pawlak, <https://gist.github.com/MaximePawlakFr/d40b46bd81d059318b04>
- [26] Learn to Build Modern Web Apps with MEAN Stack, Eric Simons, <https://thinkster.io/tutorials/mean-stack>
- [27] The Modern Application Stack – Introducing The MEASN Stack, Andrew Morgan, <https://www.mongodb.com/blog/post/the-modern-application-stack-part-1-introducing-the-mean-stack>

12 Annexes

12.1 Livrables








- Le rapport en PDF (dans le répertoire [\projet-pa\Documents\Rapport](#))
- Exécutable .apk patient (dans le répertoire [\projet-pa\Delivrables\Executable\android\android-debug-patient.apk](#))
- Exécutable .apk staff (dans le répertoire [\projet-pa\Delivrables\Executable\android\staff\android-debug-staff.apk](#))
- Code Bar (dans le répertoire [\projet-pa\Delivrables\qr-code\](#))
- Code source :
 - Serveur : dans le répertoire [\projet-pa\server\](#)
 - Client patient : dans le répertoire [\projet-pa\smartbell\](#)
 - Client staff : dans le répertoire [\projet-pa\staff\](#)

12.1.1 Code Bar patient et staff

Patients	Code Bar
patient -1111	
patient -1222	
patient -1333	

Staff	Code Bar
-------	----------

SMART BELL - projet d'approfondissement

doctor - 2111	
doctor - 2222	
doctor - 2333	
nurse - 3111	
nurse - 3222	
nurse - 3333	
assistant - 4111	

assistant - 4222	
assistant - 4333	

12.2 Les interfaces graphiques de l'application

12.2.1 SmartBell-patient – projet smartbell

SMART BELL - projet d'approfondissement

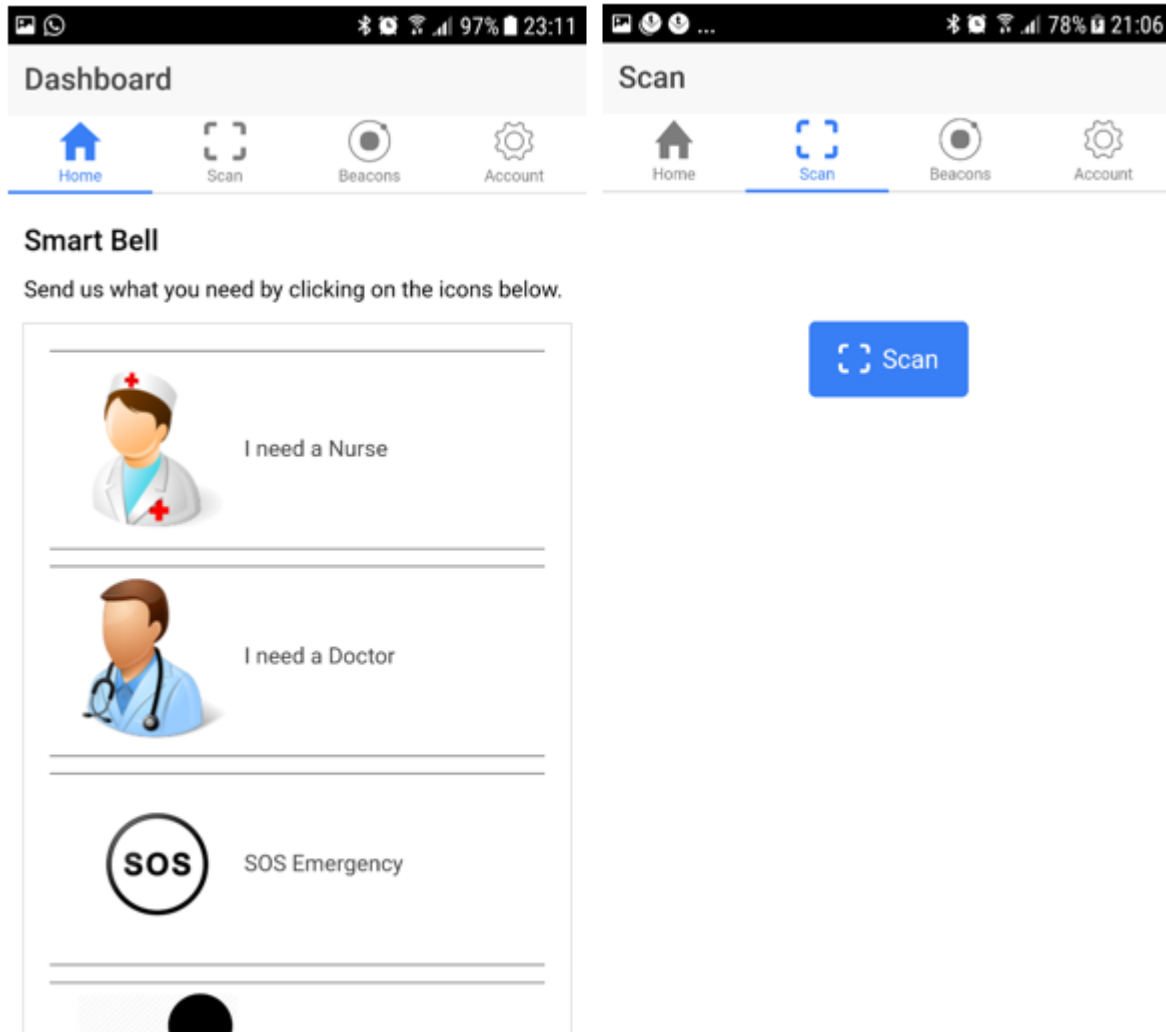


Figure 27 : aperçu de la page Accueil et Scan – patient

SMART BELL - projet d'approfondissement

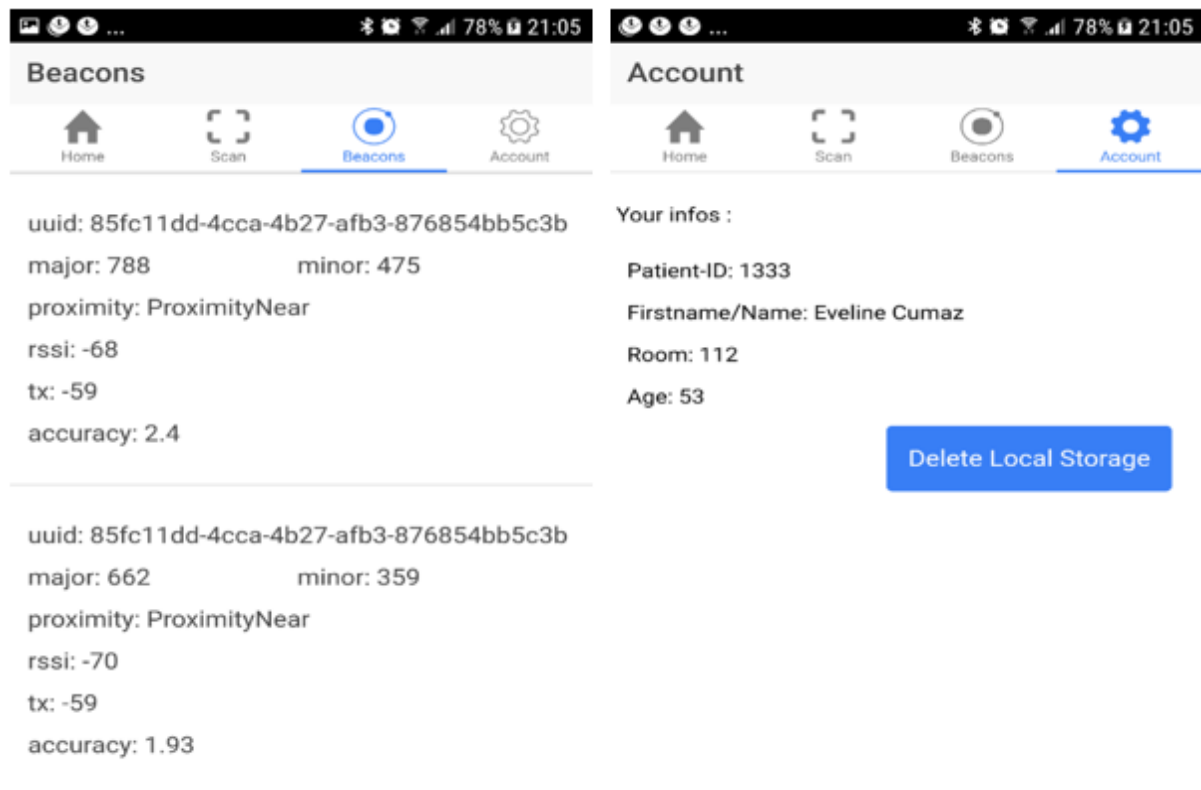


Figure 28 : aperçu de la page Beacons et Account - patient

12.2.2 SmartBell-staff – projet staff

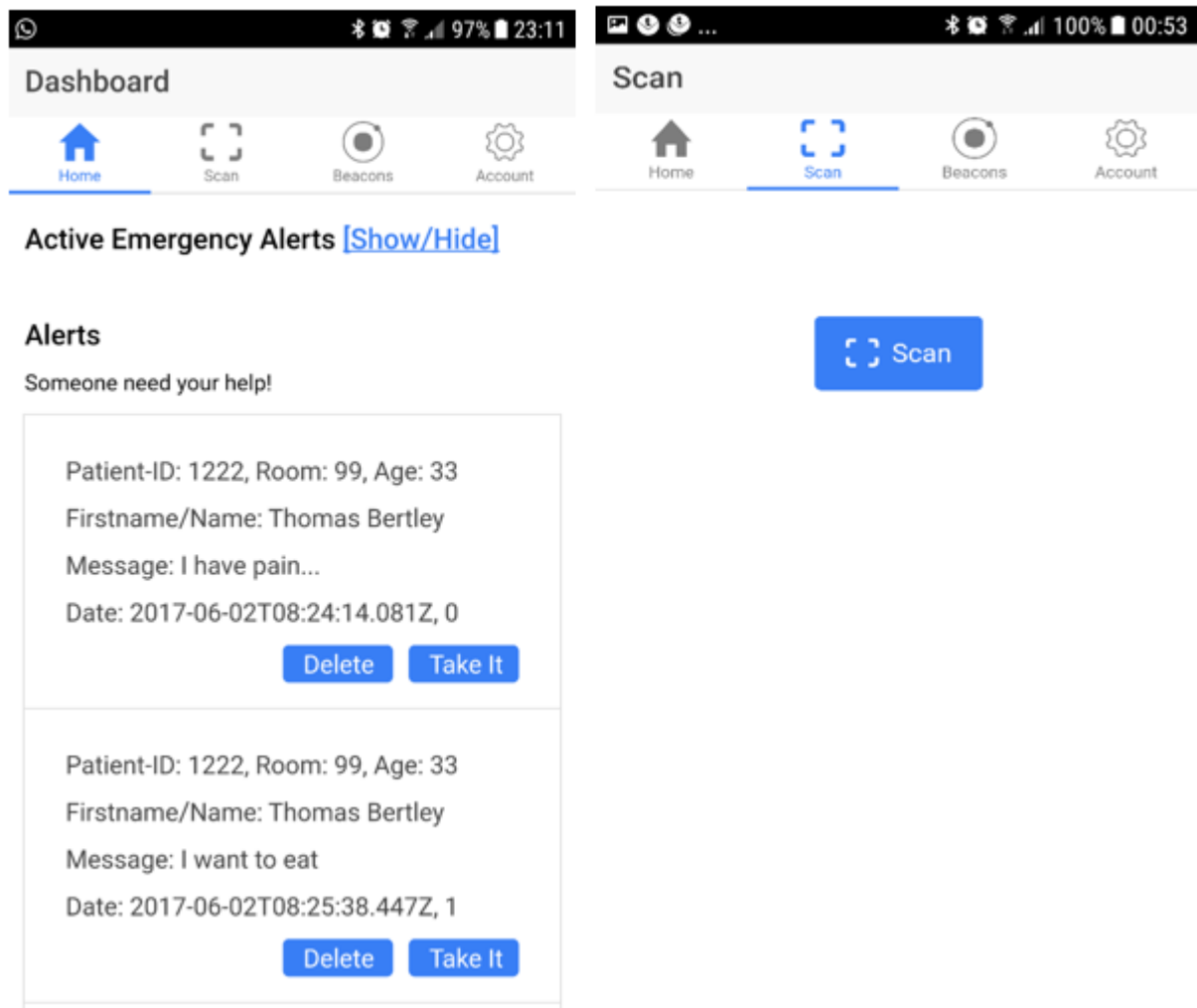


Figure 29 : aperçu de la page Accueil et Scan - staff

SMART BELL - projet d'approfondissement

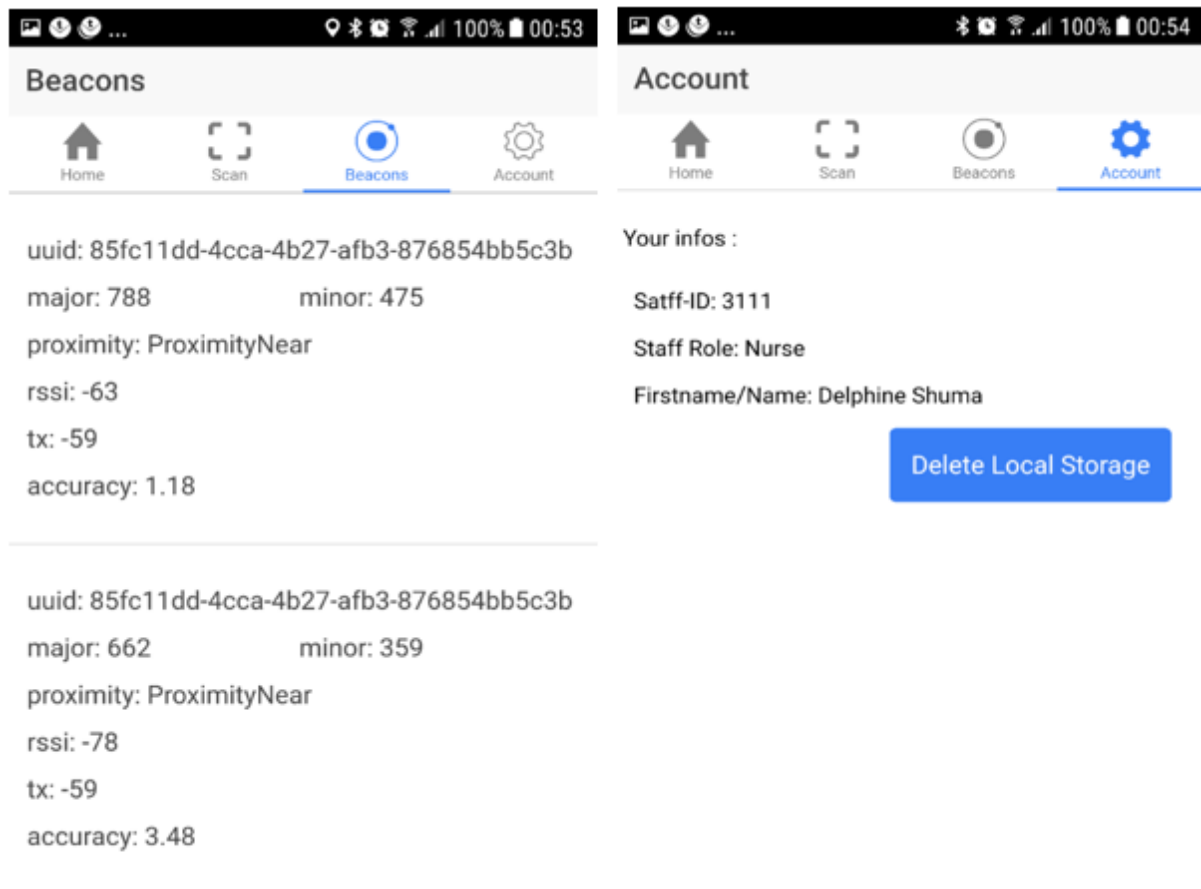


Figure 30 : aperçu de la page Beacons et Account - staff

12.3 Journal de travail

Semaine du 20 Février au 26 Février 2017

Premier rencontre avec prof. Madame Laura Elena Raileanu et l'assistant Monsieur Fabien Dutoit. Discussion à propos du projet d'un point de vue global, objectifs, but, prérequis, cahier des charges, planification et déroulement du projet.

Recherche sur des projets similaires, techniques de localisation interne et externe (Indoor and Outdoor Positioning).

Elaboration d'un planning, et préparation du journal de travail.

Semaine du 27 Février au 5 Mars 2017

Tests et essais de quelques API permettant le positionnement :

- Leaflet: <http://leafletjs.com/>
- Google Indoor Maps: <https://www.google.com/maps/about/partners/indoormaps/>
- Find (Framework for Internal Navigation and Discovery): <https://www.internalpositioning.com/>
- Indoor Atlas: <http://www.indooratlas.com/>

Recherche sur les prérequis pour l'implémentation de ces API.

Recherche sur des Framework et technologies permettant le développement des applications de localisation et positionnement.

Semaine du 6 Mars au 12 Mars 2017

Comparaison entre développement native et développement cross-plateforme (avantages, inconvénients, maintenance, coût, etc.). Bénéfices obtenus en adoptant l'une ou l'autre des 2 alternatives (native ou cross-plateforme).

Recherche sur des Framework pour le développement cross-plateforme :

Ionic, PhoneGap, React Native, Meteor, Xamarin, Crosswalk.

Comparaison entre Framework de développement Full Stack (Meteor, MEAN.io, Express.js, Sails.js, etc.) et Non-Full-Stack Framework (combinaison de plusieurs Framework Front-End, Middleware et Back-End). Avantages et inconvénients des deux alternatives, décider quelle alternative est mieux adaptée pour le projet.

Recherche pour l'installation de l'environnement de travail et outils de développement. Choix de certains outils :

IDE : Sublime Text 3, package manager : Node.js, Ionic Framework, Android Framework y compris ADB (Android Debug Bridge) et Android Studio, Cordova, etc.

Semaine du 13 Mars au 19 Mars 2017

Suivre des tutoriels pour la prise en main des Framework Meteor, Ionic et PhoneGap afin de pouvoir les essayer et les tester. Je décide de choisir Ionic pour la suite du projet (Meteor compliqué à maintenir et demande beaucoup de temps pour le maîtriser, PhoneGap basé sur Cordova et plutôt un Wrapper d'applications en plus la génération des exécutables se fait via le service PhoneGap Build qui n'est pas open-source et comme une boîte noire).

Archivage des recherches effectuées (articles, liens, Framework testés, etc.) et rédaction de quelques mots clés pour le rapport.

Préparation d'un dépôt privé sur Github, mais ça ne marche pas il est toujours public (faut s'inscrire à un abonnement payant), je décide de continuer à travailler en local.

Semaine du 20 Mars au 26 Mars 2017

Rencontre avec prof. Raileanu et M. Dutoit, discussion à propos de tous ce que j'ai fait et pour la prochaine étape. On a décidé de travailler avec les beacons, M. Dutoit m'a fourni un Kit de iBeacon composé de 2 pièces. La localisation du patient et staff dans l'hôpital va être effectuée avec les iBeacons. Discussion de l'architecture de l'application et réfléchir comment gérer les rôles du staff lors de l'envoi d'une alerte. Réfléchir à la façon dont on peut parrainer le smartphone avec l'iBeacon. Discussion à propos comment le client puisse installer l'application et parrainé son smartphone directement lors de son entrée à l'hôpital, on va scanner son le Code-Bar qui lui est fourni (une sorte de bracelet à la main avec un Code-Bar).

Recherche sur les beacons (différents protocoles iBeacons, Eddystone, AltBeacon). Comprendre le principe de fonctionnement.

L'entreprise SmartBeacon :

Recherche pour la configuration et la mise en place des iBeacons. Téléchargement de l'SDK (<https://github.com/smartbeacon/smartbeacon-android-sdk>), essai et test. Recherche de documentations et manuel d'utilisation.

Essai de fonctionner leur CMS pour la gestion à distance des iBeacons (les identifier, définir les notifications, définir leur emplacement, analyser les statistiques) mais le lien ne fonctionne pas. Il faut s'enregistrer et avoir une clé pour pouvoir accéder à la plateforme CMS. Contact de l'entreprise pour le problème, impossible de les rejoindre et aucune réponse de leur part.

Semaine du 27 Mars au 2 Avril 2017

Recherche d'API pour gérer les iBeacons à distance via un CMS par exemple. Impossible de trouver une solution car chaque entreprise qui développe des Beacons fourni son propre SDK et son propre plateforme CMS.

Test de 2 API compatible avec le protocole iBeacon pour gérer les Beacons:

<https://github.com/aemreunal/iBeaconServer>

<https://github.com/Spentas/ibeacon-server>

Les APIs ne fonctionnent pas, librairies utilisées anciennes et pas mis à jour, les deux projets sont abandonnés et ne sont plus maintenus par les auteurs.

Décide de créer from scratch notre propre API.

Choix du MEAN Full Stack Framework pour le projet (MongoDB, Express.js, AngularJS, Node.js).

Suivre plusieurs tutoriels pour la prise en main de la MEAN Stack. Suivre des tutoriels pour la prise en main d'AngularJS, de MongoDB, et de Express.JS.

Développement d'une première version de l'application SmartBell-patient avec Ionic et la MEAN Stack.

Recherche de plugin Ionic pour l'iBeacon.

Recherche de plugin Ionic pour le scan du Code-Bar.

Recherche pour le passage d'alertes entre le patient et le staff (notification push), le plugin officiel d'Ionic nécessite le passage par leur plateforme (<https://apps.ionic.io/login>), et utilise un plugin PhoneGap. Ce n'est pas privée.

Plusieurs services externes existent comme Google Firebase, PubNub, etc. Ce n'est pas privée.

Semaine du 3 Avril au 9 Avril 2017

Essais et test de deux APIs permettant le développement d'application pour iBeacons :

Evo Things: <https://evthings.com/Pub>

Installation de l'IDE Evo Things Studio, implémentation d'une application de base pour iBeacons.

PubNub: <https://www.pubnub.com/>

Implémentation du code concernant de la détection des iBeacons dans les alentours. Affichage des différentes valeurs (proximité, puissance du signal, UUID, major, minor, etc.).

Implémentation du code pour le scan du Code-Bar.

Suivre des tutoriels pour l'implémentation d'une API avec Express.js.

Implémentation du serveur Express.js.

Semaine du 10 Avril au 16 Avril 2017

Suivre des tutoriels pour l'implémentation d'une base de donnée MongoDB.

Implémentation de la base de donnée beaconstore.

Ajout d'une collection beacons (qui représente les patients) dans la base de donnée.

Création de l'API beacons :

- Ajout du code du schéma beacons dans le modèle du serveur.
- Implémentation des méthodes CRUD (Create, Read, Update, Delete) de l'API beacons.

Implémentation des requêtes GET et POST dans le client patient. Test du fonctionnement.

Implémentation d'une fonction permettant de calculer la distance entre le mobile et tous les beacons détectés à proximité et retournant le beacon le plus proche du mobile afin de puisse l'envoyer lors de l'envoi d'une alerte.

Semaine du 17 Avril au 23 Avril 2017

Ajout d'une collection staffbeacons (qui représente le staff) dans la base de donnée.

Création de l'API staffbeacons :

- Ajout du code du schéma staffbeacons dans le modèle du serveur.
- Implémentation des méthodes CRUD (Create, Read, Update, Delete) de l'API staffbeacons.

Implémentation des requêtes GET et POST dans le client staff. Test du fonctionnement.

Recherche concernant « socket.io » pour l'échange de donnée lors d'un évènement. Voir la documentation et suivre un tutoriel pour comprendre le fonctionnement.

Implémentation du socket côté patient et serveur. Test du fonctionnement. Envoi d'alerte vers le serveur via le socket.

Implémentation du socket côté staff, test du fonctionnement.

Semaine du 24 Avril au 30 Avril 2017

Ajout d'une collection alerts (qui représente les alertes envoyées par le patient) dans la base de donnée.

Création de l'API alerts:

- Ajout du code du schéma alerts dans le modèle du serveur.
- Implémentation des méthodes CRUD (Create, Read, Update, Delete) de l'API alerts.

Implémentation des requêtes GET et POST dans le client patient. Test du fonctionnement.

Dans le client staff implémentation d'une zone « Alerts » permettant d'afficher les alertes envoyées au staff.

Implémentation de l'algorithme de distribution des alertes selon le type d'alerte envoyée par le patient, selon le besoin l'alerte sera adressé au staff adéquat (médecin, infirmier ou assistant).

Envoi de plusieurs type d'alertes et test chez le client staff qu'il reçoit l'alerte correspondante.

Semaine du 1 Mai au 7 Mai 2017

Dans le client staff :

- Implémentation du Local Storage pour le stockage de donnée.
- Implémentation d'un onglet « Account » pour afficher les informations du staff : son ID, son rôle (médecin, infirmier, assistant), nom et prénom.
- Ajout d'un bouton Delete Local Storage permettant d'effacer les données.

Dans le client patient :

- Implémentation du Local Storage pour le stockage de donnée.
- Implémentation d'un onglet « Account » pour afficher les informations du staff : son ID, son rôle (médecin, infirmier, assistant), nom et prénom, le numéro de la chambre, son âge.
- Ajout d'un bouton Delete Local Storage permettant d'effacer les données.

Semaine du 8 Mai au 14 Mai 2017

Dans le client staff:

Dans la zone d'affichage « Alerts » ajouter un bouton Delete pour la suppression de l'alerte.

Ajouter un bouton « Clear all alerts » permettant de supprimer tous les alertes qu'il a reçu.

La suppression se fait uniquement côté client et non pas dans la base de donnée.

Améliorer l'esthétique des interfaces des 2 clients patient et staff.

Semaine du 15 Mai au 21 Mai 2017

Préparation et ajout d'une série de donnée concernant le patient et le staff :

Générations de fake data pour le patient (3 patients).

Génération de fake data pour le staff (3 médecins, 3 infirmiers, 3 assistants).

Génération des Code-Bar correspondant depuis le service Easy Tag (<http://www.easytag.pro/generateur-qr-code>).

Dans le client staff implémentation de l'affichage d'une zone « Active Emergency Alerts » qui correspond à tous les alertes envoyées par les patients. Implémentation des requêtes à la base de donné. Ajout d'un bouton « Show/Hide » pour une meilleure visibilité.

Pour le client et le serveur ajout d'une variable « ipExtern » permettant de se connecter à une adresse IP spécifique (autre que le localhost).

Semaine du 22 Mai au 28 Mai 2017

Documentation sur le plugin Ionic Local Notification permettant l'envoi et la réception de notification sans passer par un service externe comme Firebase. Suivre un tutoriel pour l'implémentation.

Implémentation du code de notification push dans les deux clients (patient et staff) et du côté serveur. Ajuster le temps de réponse en secondes pour l'envoi de la notification. Test et déploiement du code sur le mobile.

Dans le client staff ajout d'un bouton « Take It » permettant au staff de notifier le patient qu'il vient pour son aide. Implémentation du code nécessaire.

Semaine du 29 Mai au 4 Juin 2017

Rencontre avec prof. Raileanu et M. Dutoit. Discussion à propos du rapport. Discussion à propos de la procédure de livraison (date de rendu du projet, ce qu'il faut rendre, etc.).

Diverses remarques concernant le rapport, les points suivants sont à améliorer :

- Structure des paragraphes.
- Reformuler certaines phrases.
- Numéroté tous les images et mentionner référence des images (source depuis laquelle les images ont été téléchargées).
- Améliorer le schéma de l'architecture de l'application (ajouter appareils mobiles, flèches de communication entre les composants).
- Ajouter les Use-Case

Création des Use-Case.

Refaire le schéma de l'architecture de l'application.

Semaine du 5 Juin au 11 Juin 2017

Continuation de la rédaction du rapport. Corriger et améliorer les remarques de la semaine passée.

Préparation des rendus.

Nettoyage du code (suppression des commentaires inutiles, console de Debug, etc.).

Test du fonctionnement de l'application avant le rendu.

12.4 Structogramme distribution des alertes

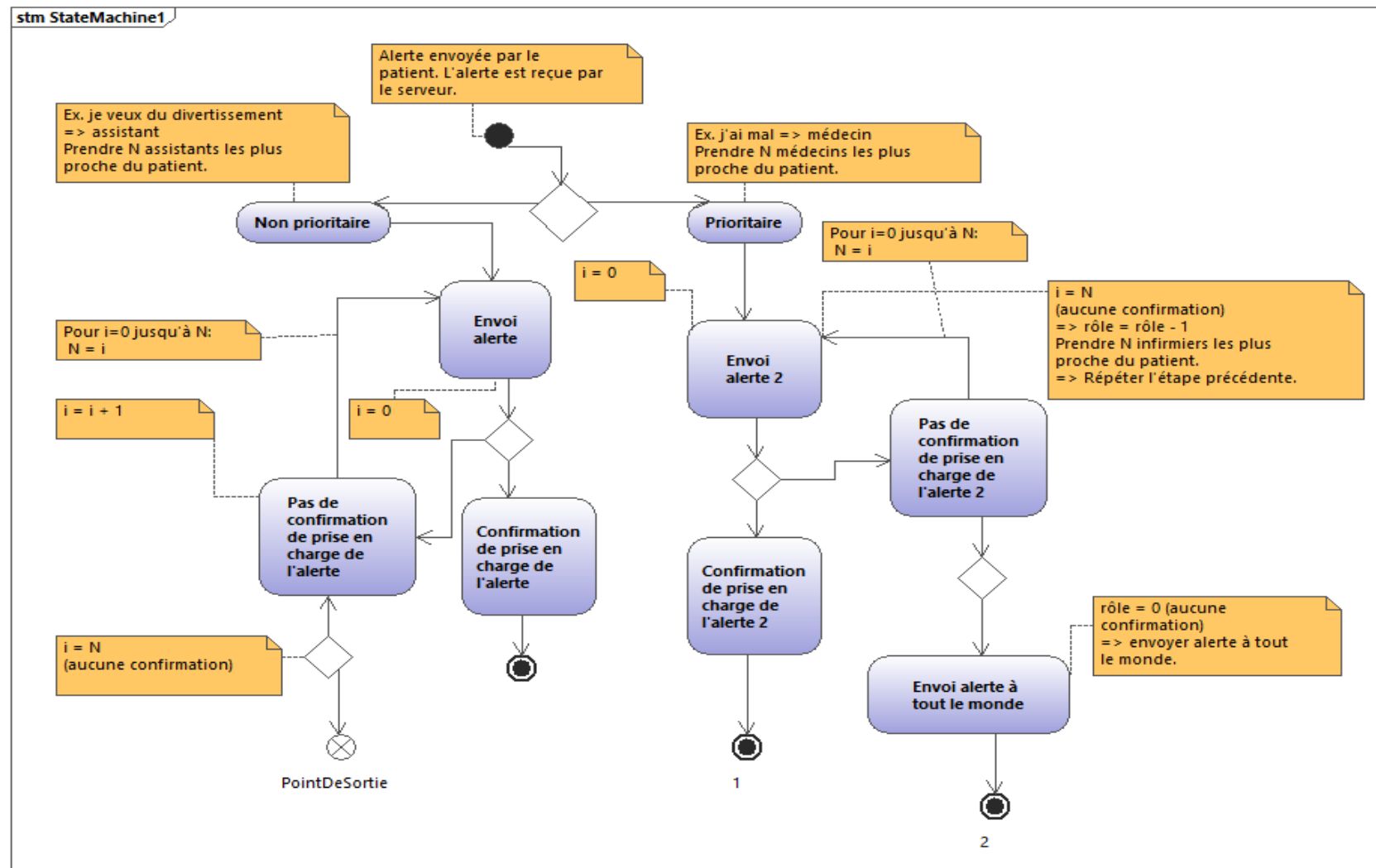


Figure 31 : structogramme de l'algorithme de distribution des alertes

12.5 Planification

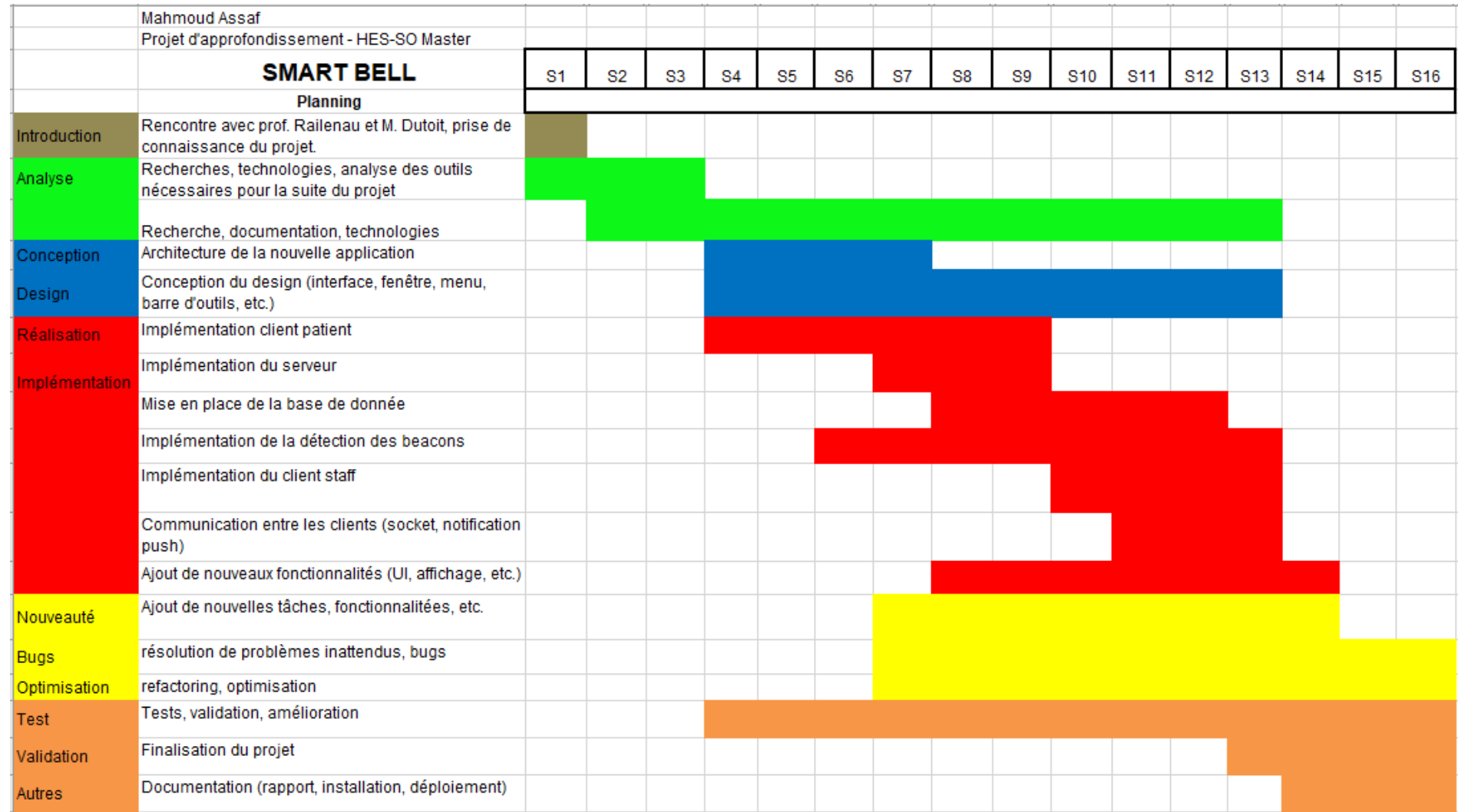


Figure 32 : planification