

Projet Beacons

Micro-localisation d'objets

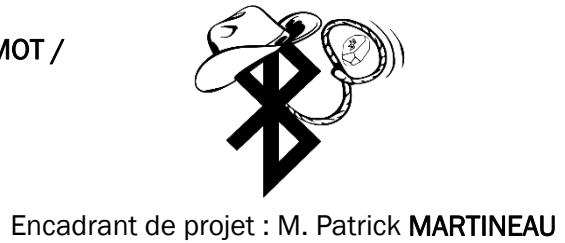


Rapport
de projet

Projet Collectif
4^{ème} année DII

Karim EL GHARBI / Julien GIDEL / Brandon SIMON-VERMOT /
Quentin ARCICAULT / Damien LE LANN / Axel EYRAUD

Promotion 2016-2019
DII 4A



Encadrant de projet : M. Patrick MARTINEAU

Sommaire

Remerciements	7
Introduction	8
1 Cahier des charges.....	9
1.1 Descriptif du projet.....	9
1.1.1 L'objectif général du projet.....	9
1.1.2 Les connaissances requises	10
1.1.3 Exigences d'aboutissement	10
1.1.4 Livrable du projet	10
1.2 Liste des tâches	12
1.3 Bilan des exigences / objectifs.....	14
1.4 Livrables du projet.....	14
2 Méthodologie	15
2.1 Carte mentale	15
2.2 Planning	18
2.2.1. Cadrage du projet	18
2.2.2. Planning prévisionnel	18
2.3 Étude de faisabilité et identification des risques	20
2.3.1 Besoins du projet	20
2.3.2 Scénarios (matrice SWOT)	21
2.3.3 Identification des risques.....	22
2.4 Méthode de gestion de projet.....	24
2.4.1. Cycle en V	24
2.4.2. Méthodes agiles.....	25
2.5 Répartition des tâches	26
2.6 Outils utilisés	27
2.6.1 Microsoft Office	27
2.6.2 Trello	28
2.6.3 Discord	28
3 Étude de l'existant.....	31
3.1 Rappel du contexte	31
3.2 Qu'est-ce qu'un Beacon ?.....	31

3.3	Étude du besoin	32
3.4	Matériel de départ.....	33
3.5	Veille technique	34
3.5.1	Cadres d'application des Beacons.....	34
3.5.2	Limites des systèmes	35
3.5.3	OS mobiles	35
4	Études et réalisations techniques	36
4.1	iBeacon ou Eddystone ?	36
4.1.1	iBeacon.....	36
4.1.2	Eddystone	36
4.2	Recherche de Beacons	37
4.2.1	Avant-propos.....	37
4.2.2	Gimbal.....	37
4.2.3	RadBeacons	38
4.2.4	Kontakt.io	40
4.2.5	Gateway.....	41
4.2.6	BNB Beacons	42
4.2.7	Conclusion de l'étude.....	44
4.3	Prise en main et configuration du matériel	44
4.3.1	Beacons Gimbal	44
4.3.2	Beacons BnB	46
4.4	Mise en place d'une stratégie de localisation	50
4.4.1	Les différentes stratégies	50
4.4.2	Choix de la stratégie	51
4.4.3	Triangulation ou trilateration ?	52
4.5	Architecture matérielle	53
4.5.1	Aspect « hardware ».....	53
4.5.2	Protocole(s) de communication.....	53
4.6	Recherche de solutions techniques	54
4.6.1	Langage de développement & API d'interfaçage	54
4.6.2	Choix des balises fixes	56
4.7	Conception et développement de l'application « chaud-froid ».....	57
4.7.1	Analyse	57

4.7.2	Conception	58
4.7.3	Développement.....	60
4.7.4	Développement de l'interface graphique	61
4.8	Conception et développement de l'application « Localisation ».....	63
4.8.1	Analyse.....	63
4.8.2	Conception	69
4.8.3	Développement.....	80
4.8.4	Interfaçage des sous-ensembles	94
4.8.5	Optimisation des programmes.....	96
4.9	Tests & expérimentation.....	96
4.9.1	Précision des Beacons	96
4.9.2	Précision des balises fixes	98
4.9.3	Expérimentation de la trilateration	100
4.10	Limites de la preuve de concept.....	103
4.11	Pistes d'amélioration.....	104
4.11.1	Solution de positionnement	104
4.11.2	Étude sur le traitement du signal Bluetooth	104
4.12	Outils utilisés	108
5	Présentation des résultats	109
6	Analyse des résultats	111
	Conclusion.....	113
	Avis Personnels	114
	Karim.....	114
	Julien	115
	Brandon	116
	Quentin	117
	Damien	118
	Axel	119
	Table des illustrations.....	120
	Annexes	125
	Annexe 1 : Bilan financier	125
	Annexe 2 : Configuration des Beacons Gimbal et programmation d'une application.....	126
7.1.1	Gimbal Beacon Manager.....	126

7.1.2	Programmation	128
Annexe 3 :	Shared Preferences.....	131
7.1.3	Fonctionnement.....	131
7.1.4	Intégration	131
Annexe 4 :	GITHUB	133
7.1.5	Présentation de Git	133
7.1.6	Présentation de GitHub	135
7.1.7	Présentation du logiciel GitKraken	136
7.1.8	Mise en œuvre de GitKraken.....	137

Remerciements

Nous tenons à remercier toutes les personnes qui ont pu contribuer à la réalisation de ce projet. Nous adressons principalement nos remerciements à Patrick MARTINEAU, notre encadrant de projet, de nous avoir proposé ce sujet et de nous avoir accompagné. Monsieur MARTINEAU a su rester à notre écoute pour répondre à nos interrogations et pour nous aider face aux difficultés. Il nous a également suivis à travers des revues d'avancement régulières, ce qui a contribué à la réussite de ce projet.

Introduction

Dans le cadre de notre formation d'ingénieur en informatique industrielle, des projets collectifs ont été mis en place pour permettre d'appliquer nos compétences acquises en cours, dans un projet concret et en équipe. L'objectif étant d'acquérir une première expérience dans un véritable travail de groupe impliquant plusieurs personnes.

Notre projet consiste en **la mise en œuvre de la géolocalisation d'objets** à l'aide de balises appelées Beacons, émettant des signaux Bluetooth. Ces objets sont utilisés dans le commerce où lors d'événements (salons, expositions...) dans le but de donner des informations à un utilisateur selon sa position ou tout simplement pour connaître sa position. Il existe un large cadre d'application pour les Beacons.

L'objectif de ce projet est de mettre en place la géolocalisation d'objets **fixes et mobiles**. Ce travail se décompose en deux parties. La première consiste à développer **une application mobile sous forme de jeu**, permettant à un utilisateur de trouver sa cible à la manière d'un jeu **chaud-froid**. La seconde consiste à **mettre en place une solution matérielle et logicielle** permettant de détecter plusieurs cibles fixes et mobiles. Il s'agit de développer une application mobile permettant de **localiser une ou plusieurs cibles sur un plan**. La finalité de ce projet est de fournir **une preuve de concept englobant les deux sous-ensembles** et de conclure sur la faisabilité de ce projet. Une étude solide est donc attendue.

Nous verrons dans un premier temps notre **cahier des charges** avec le descriptif du projet et les livrables, puis dans un second temps notre **méthodologie** avec la carte mentale, le planning ainsi que l'étude de faisabilité. Enfin, nous finirons par voir **l'étude d'avant-projet** suivie des **études et réalisations** composées d'une partie technique et une partie expérimentale.

Une analyse des résultats clôturera ce rapport.



Figure 1 : Beacons

1 Cahier des charges

Comme dans tout projet, le livrable est décrit dans un document formalisé : **le cahier des charges**. C'est le premier document que nous avons réalisé pour démarrer notre projet.

1.1 Descriptif du projet

La première partie du cahier des charges est un descriptif de ce que l'on attend du projet dans sa finalité.

1.1.1 L'objectif général du projet

1.1.1.1 Contexte

Au moins deux projets ont déjà été réalisés au sein de Polytech autours des Beacons : une application de localisation des personnes à l'intérieur d'un bâtiment (en général) et une autre de localisation de plusieurs personnes dans un salon d'exposition. La mairie de Tours, intéressée par ce type de technologie, a permis la mise en situation du système lors d'un évènement. L'objectif de ces projets était de suivre une personne à partir de Beacons répartis dans l'espace.

1.1.1.2 Demande client

Dans notre cas, on nous propose d'inverser l'usage : les balises seront positionnées sur l'objet à suivre et à géolocaliser. L'utilisateur pourra utiliser un smartphone (ou un autre système mobile) pour retrouver l'objet. Deux cadres d'application nous ont été proposés à titre d'exemple : suivi de matériel dans un atelier de production, géolocalisation de lits au sein d'un établissement hospitalier ...

1.1.1.3 Contraintes

La principale contrainte est le fait que l'objet doit pouvoir être mobile. Cela complexifie le problème qui est de récupérer la distance par rapport à un appareil. Une solution consisterait à placer des balises fixes et d'utiliser le principe de la triangulation (ou la trilateration) pour déterminer la position de ces Beacons mobiles. Cette solution doit être étudiée et justifiée au cours du projet. Il sera nécessaire de faire un choix technologique permettant de répondre à cette problématique (ex : Arduino, Balise WiFi...).

Concernant le design de l'application il n'existe pas de contrainte. Le choix devra être fait le moment venu lors de la conception de l'interface graphique.

1.1.1.4 Démarche

Ce projet collectif consiste plus spécifiquement à concevoir une application mobile ou web permettant de **localiser des Beacons** dans un espace clos. C'est le point de vue inverse par rapport aux projets précédents.

L'étude consiste à **considérer l'existant, établir une stratégie de localisation, prendre en main le matériel et rechercher des solutions techniques**.

Le développement de l'application doit être composé d'une **analyse logicielle, d'une phase de conception de l'architecture logicielle et matérielle, de la phase de codage et de tests suivi de la validation**. La méthode agile Scrum pourra être *privilégiée car c'est à priori la méthode qui correspond au contexte actuel*.

La mise en place de ce projet est elle-même constituée de plusieurs phases : commencer par **communiquer avec un Beacon et le configurer, récupérer la distance** (de façon à guider l'utilisateur vers sa cible), mettre en place une **interface graphique** pour le guidage.

Cela constitue la base du projet et permet de réaliser les autres étapes.

1.1.2 Les connaissances requises

- Principes fondamentaux des normes et réseaux
- Programmation C/C++
- Programmation Java
- Connaissances en Bluetooth, Wi-Fi...
- Connaissances en Web (JavaScript, PHP, HTML, CSS...)
- Principes mathématiques : triangulation, puissance d'un signal, effet doppler
- Transmission de l'information

1.1.3 Exigences d'aboutissement

- Application mobile fonctionnelle regroupant le jeu « chaud-froid » et l'application de géolocalisation des balises
- Documentation utilisateur : notice d'utilisation

1.1.4 Livrable du projet

Le livrable de ce projet se décompose en différents niveaux du minimum au maximum. Si le minimum est garanti, le cahier des charges est rempli (3/4).

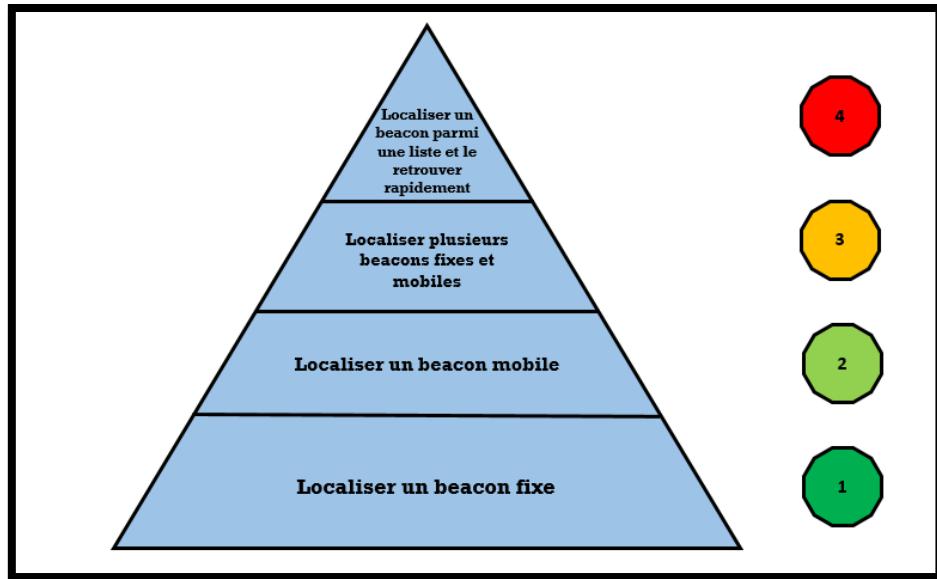


Figure 2 : Objectifs du projet

La pyramide ci-dessus présente les niveaux d'exigences du projet. Une application fournissant une solution de détection valide le cahier des charges. Néanmoins, la conclusion prédomine sur la solution dans notre démarche de **preuve de concept**. Le cœur du projet est de fournir une réponse au client sur la faisabilité de sa demande.

1.2 Liste des tâches

Tâches	Livrable	Durée
Effectuer une étude de l'existant	Bilan des domaines d'application	5 semaines
Établir une stratégie de localisation	Document comparatif/bilan justifiant de la solution adoptée	6 semaines
Prendre en main le matériel	Beacons fonctionnels et rapport d'essais	6 semaines
Rechercher des solutions techniques	Tableau comparatif des langages / environnements de développement et rapports justifiant les choix	6 semaines
Effectuer une analyse logicielle	Diagrammes UML (diagramme des cas d'utilisation, diagrammes de séquence, diagrammes des classes...)	3 semaines

Définir l'architecture logicielle	Schémas décrivant le fonctionnement de l'application	4 semaines
Coder l'application et effectuer des tests	Application fonctionnelle permettant de détecter des Beacons fixes et mobiles	29 semaines
Mise au point et tests	Pas de livrable	3 semaines
Validation	Pas de livrable	1 semaine
Rédiger la documentation utilisateur	Fournir une notice pour l'utilisation de l'application	2 semaines

Figure 3 : Liste des tâches

Ci-dessus une estimation des premières tâches du projet avec des prévisions temporelles. La création de ce tableau précède la création du planning permettant ainsi de préparer le travail.

1.3 Bilan des exigences / objectifs

MUST	COULD
<ul style="list-style-type: none"> <i>Développer un jeu de type chaud-froid</i> <i>Mettre en œuvre la localisation de balises fixes / mobiles</i> 	<ul style="list-style-type: none"> <i>Avoir une application finie prête à être utilisée</i> <i>Mise en place d'un serveur</i> <i>Création d'une base de données</i> <i>Optimisation du programme (évolutivité)</i>
SHOULD	WON'T
<ul style="list-style-type: none"> <i>Intégrer les fonctionnalités sous forme de jeu</i> <i>Améliorer la précision de la localisation</i> 	<ul style="list-style-type: none"> <i>Guidage de l'utilisateur jusqu'à la cible (localisation de l'utilisateur)</i> <i>Localisation précise au mètre près</i>

Figure 4 : MOSCOW

Le tableau ci-dessus répertorie les tâches générales du projet : celles qui doivent être mises en place, celles qui peuvent si le temps le permet et enfin celles qui ne doivent pas du tout être traitées. Au cours de ce projet, nous nous sommes concentrés principalement sur le **MUST** et le **SHOULD** et ponctuellement sur le **COULD**.

1.4 Livrables du projet

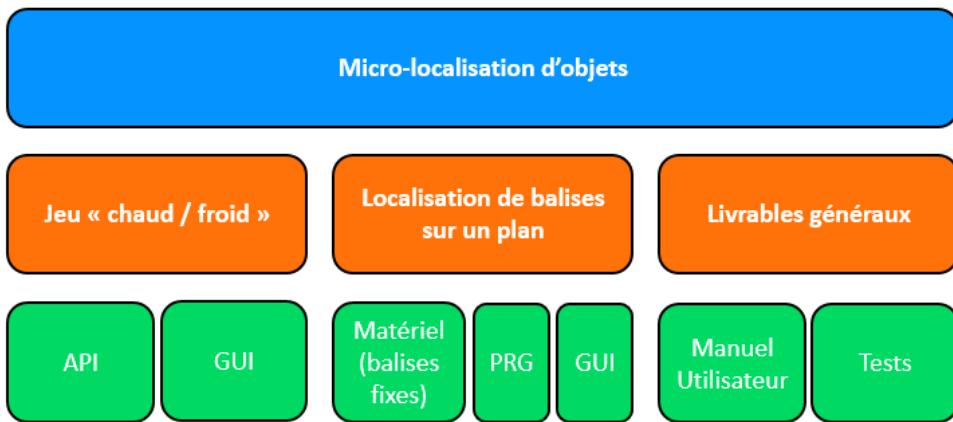


Figure 5 : SDP

La **structure de découpage du projet** ci-dessus décompose le projet en livrables et en sous-ensembles de manière à bien établir une vue globale de toutes les composantes du projet.

2 Méthodologie

Maintenant que nous avons bien exposé les objectifs, nous pouvons étudier la méthodologie en mettant l'accent sur l'aspect gestion de projet.

2.1 Carte mentale

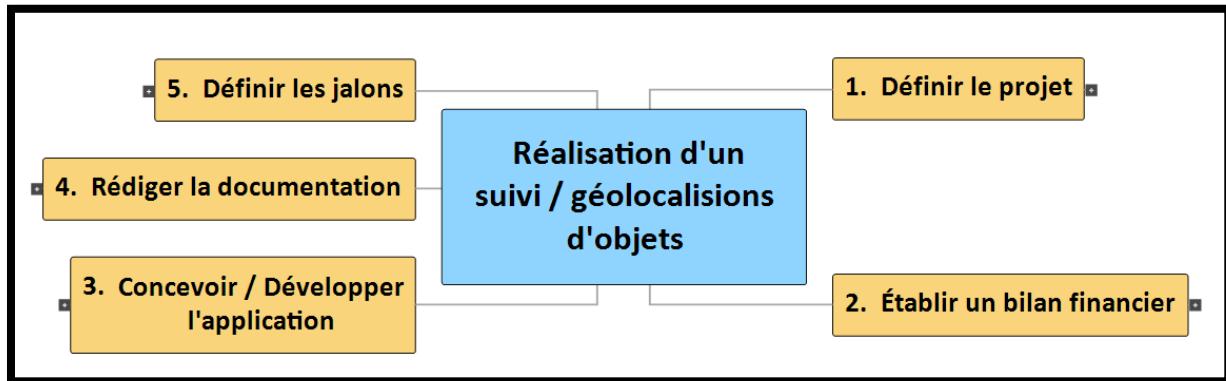


Figure 6 : Carte mentale générale

La réalisation de ce projet a nécessité l'élaboration d'une méthode de travail. La première étape, nécessaire, consiste à **définir le projet en fixant les objectifs**, permettant ainsi de **décomposer le projet en fonctions**. L'étude menée a mis en avant les technologies à mettre en œuvre. La gestion de projet a été respectée par la création d'un **planning** et d'une **répartition des tâches** affichant une vue d'ensemble de la chronologie. **La mise en œuvre du système** suivie par **des réunions d'avancement** régulières, ont permis d'évaluer l'état du projet jusqu'à la livraison de l'application.

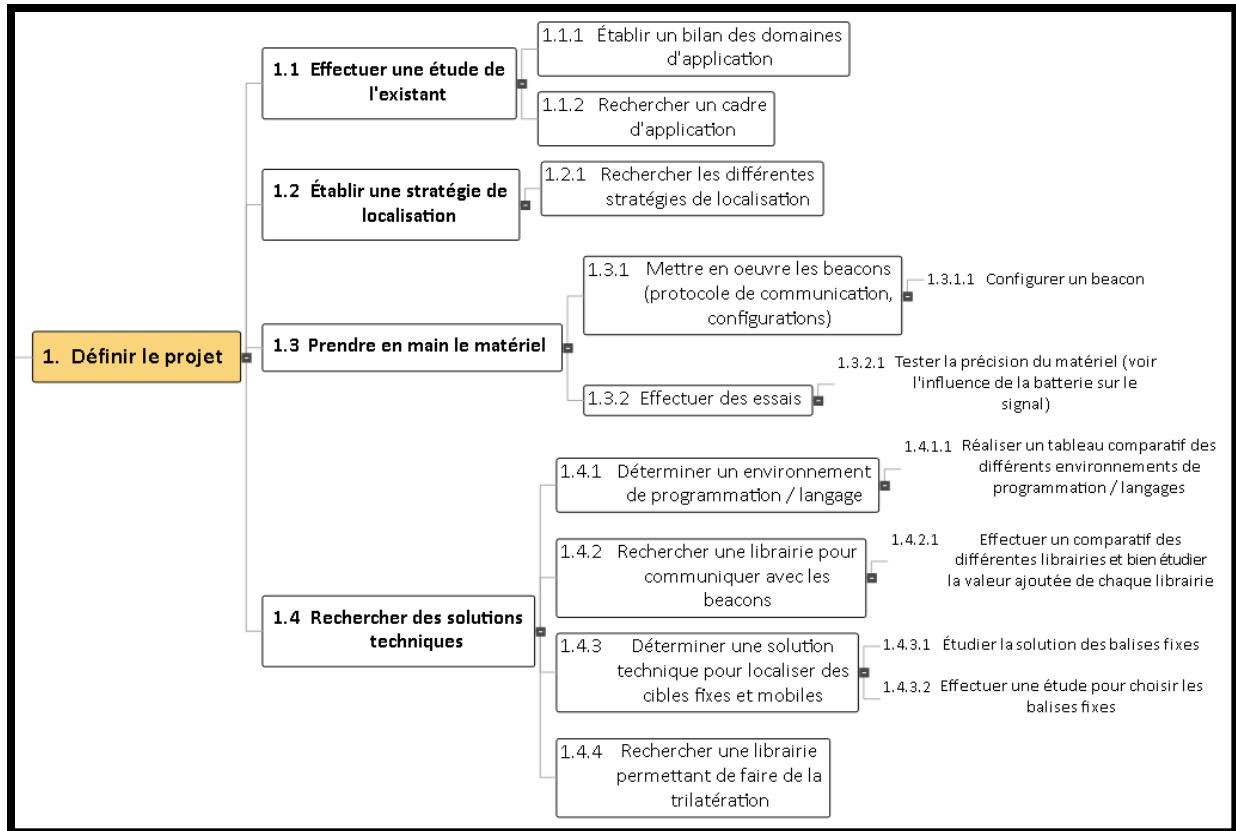


Figure 7 : Carte mentale définir le projet

Notre premier objectif a été de prendre en main le projet et de découvrir ce qui a été fait dans les précédents projets. Suite à cela, **un cahier des charges** a été rédigé compte tenu des informations recueillies lors de la phase **d'avant-projet**. Sur le plan technique, la phase de définition du projet nous a permis de faire des recherches pour savoir comment mettre en œuvre le système. Les choix effectués concernent **les outils et les méthodes à utiliser ainsi que le matériel à employer**. Cela a également permis d'anticiper sur la mise en œuvre. Une fois cette première étape terminée, nous avons effectué une analyse plus approfondie de la gestion de projet.

Tout projet nécessite **une étude de faisabilité**. L'objectif de cette étape, est de vérifier que le cahier des charges est réaliste, c'est-à-dire qu'il faut vérifier si l'on dispose des compétences nécessaires mais surtout si le projet est réalisable dans le temps imparti. Une fois cette étape passée, il faut **identifier l'ampleur des risques** afin de prévenir les éventuelles menaces pour éviter une dérive du planning.

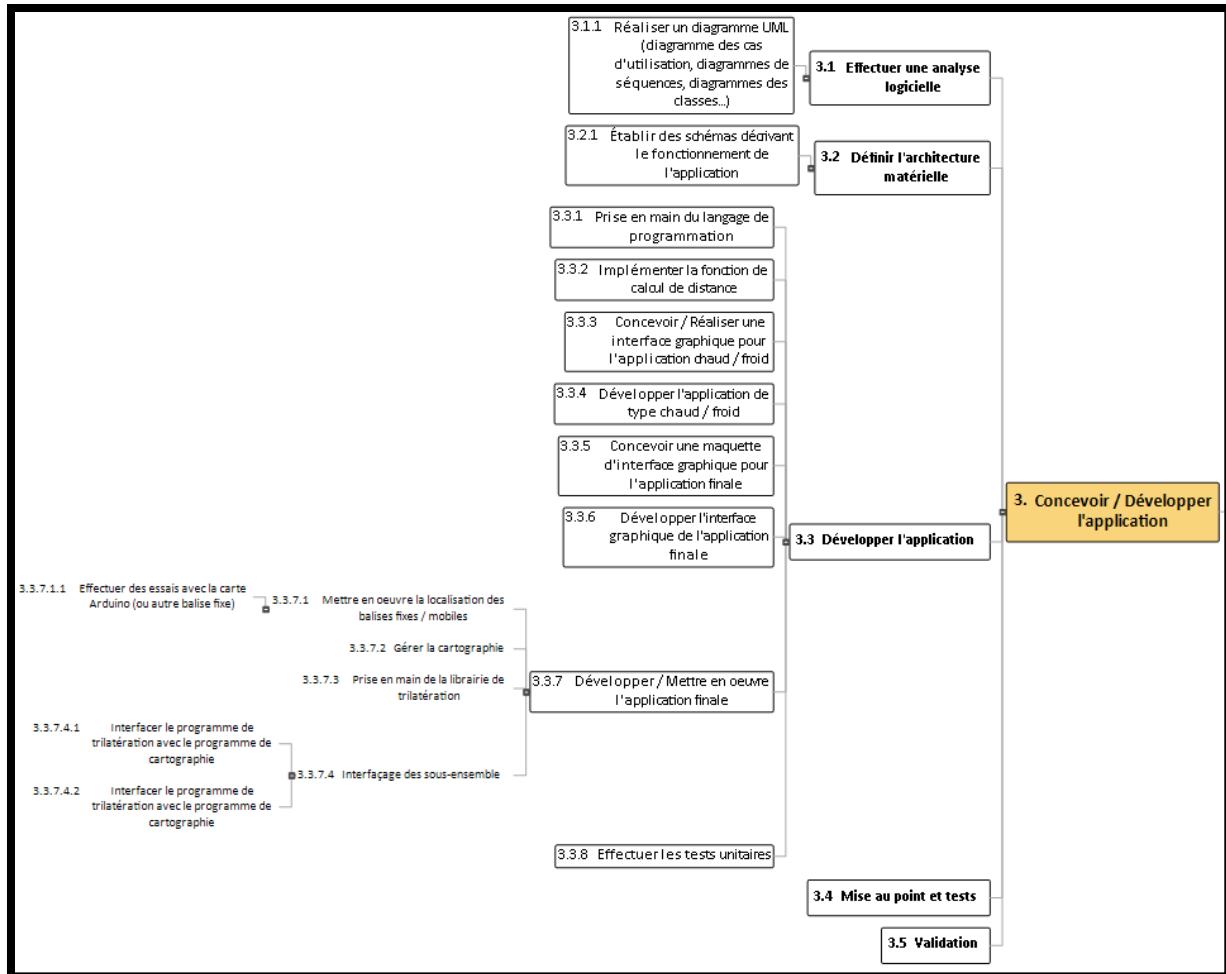


Figure 8 : Carte mentale concevoir / développer l'application

Ensuite, la phase de conception / développement a débuté par **une analyse logicielle et architecturale**. Cela permet de définir le système et d'avoir une structure logicielle et matérielle solide. Une fois la phase d'étude terminée, l'application mobile « chaud-froid » a été mise en place à travers la prise en main du matériel, l'implémentation du calcul de distance et la conception de l'interface graphique. **L'application finale** a été développée à travers la mise en œuvre de la localisation des balises fixes et mobiles et d'une interface graphique. Il s'agissait ensuite d'effectuer **l'intégration et les tests de validation** et éventuellement des **mises au point**.

Remarque : cette démarche semble à première vue très structurée tel un **cycle en V**.

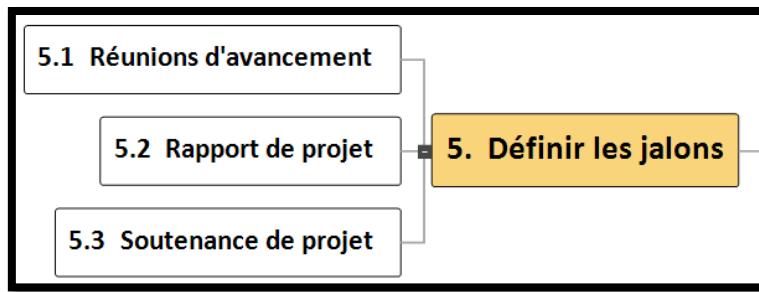


Figure 9 : Carte mentale définir les jalons

Au début du projet, nous avons fait en sorte de bien identifier les principaux jalons dont les **réunions d'avancements** et les formalités du projet (rapport de projet et soutenance de projet). De même, nous avons fait en sorte de fournir une partie de notre projet (chaud-froid) pour les portes ouvertes qui se sont déroulées le 17 février 2018.

2.2 Planning

2.2.1. Cadrage du projet

- Déroulement du projet : S39 – S23
 - Début du projet le 25 Septembre 2017
 - Fin du projet le 06 / 06 / 2018 (Soutenance)

2.2.2. Planning prévisionnel

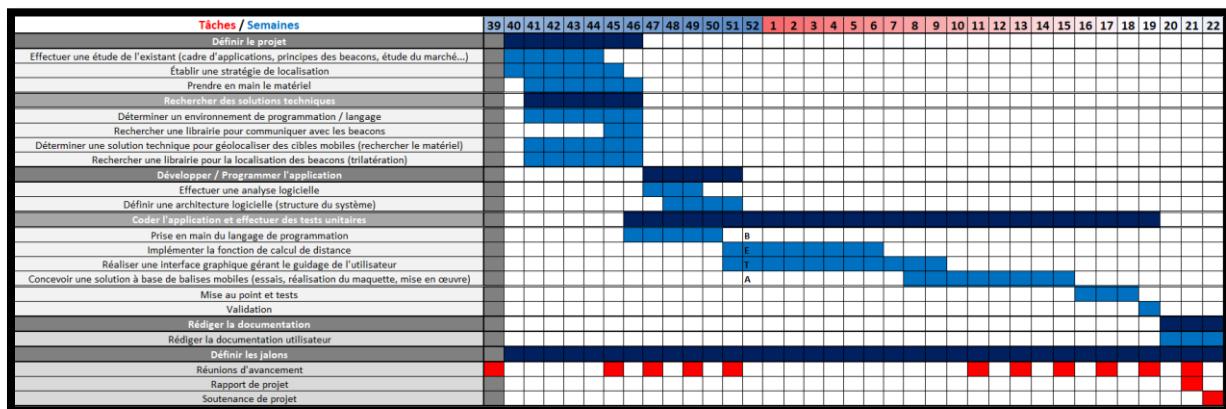


Figure 10 : Planning prévisionnel

Le planning prévisionnel, a été évalué de la manière suivante : l'étude et la **définition du projet** ont été estimées à sept semaines, ce qui permettait de faire une analyse précise du cahier des charges et des solutions potentielles. Cela facilite l'avancement de la partie technique. Nous pouvons voir ci-dessus que la partie **mise en œuvre** est la plus longue. Cette dernière contient toutes les tâches liées à la conception du produit. Il a été décidé que la mise en œuvre de la partie matérielle (balises fixes) pouvait se faire à la suite de la première partie du projet qui est entièrement

« software » (développement du « chaud-froid »). Cela explique l'illustration ci-dessus. Les points difficiles à estimer sont ceux qui composent les parties techniques. N'ayant pas d'expérience dans ce domaine, il n'est pas facile d'estimer le temps à allouer. Par conséquent, beaucoup de tâches ont été rééstimées au cours du projet. Enfin, ce planning regroupe également **les différents jalons**. Nous avons pris soin de programmer à l'avance nos réunions d'avancement avec notre encadrant.

La démarche de **planification** a permis de se projeter dans le temps en effectuant une première estimation. Ainsi, cela nous a donné une idée plus ou moins précise du temps qu'il est possible de consacrer à chaque tâche.

Suite à cette première estimation temporelle, une autre estimation a été effectuée en respectant les liens de précédence et la complexité de chaque tâche.

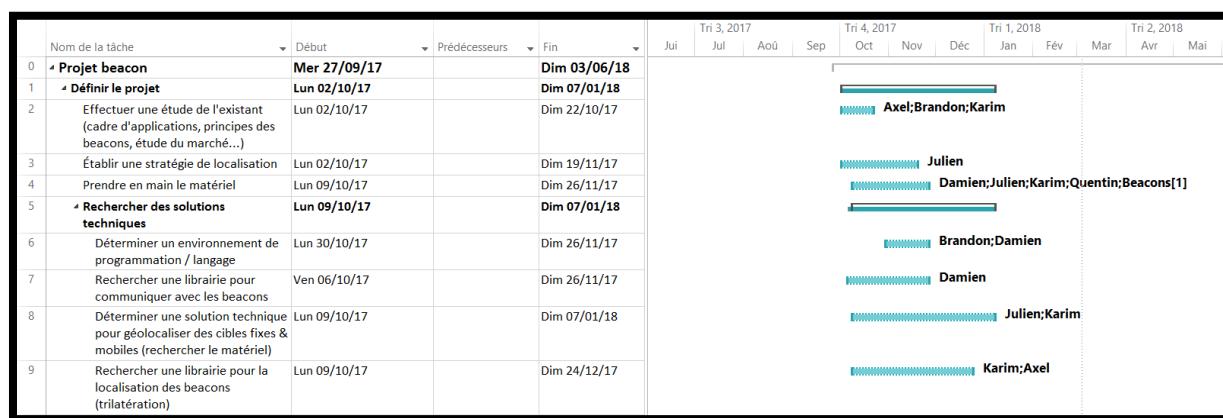


Figure 11 : Planning prévisionnel phase de spécification

Voici ci-dessus un planning prévisionnel de la **phase de spécification du projet**. On retrouve globalement le même résultat que précédemment. L'avantage d'avoir utilisé cette méthode est que l'on a pu gérer les ressources avec aisance. De plus, il est bien plus facile de visualiser l'avancement des tâches grâce à cette méthode.

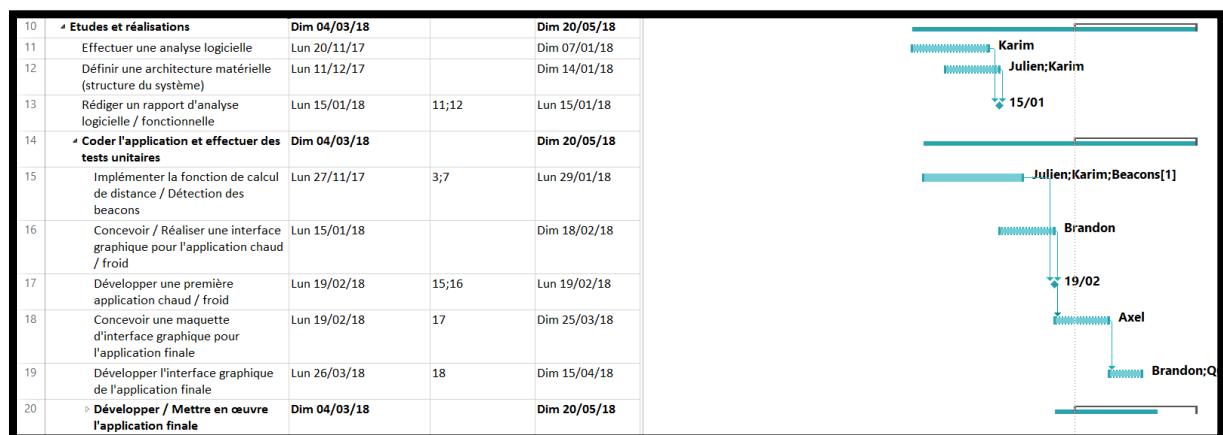


Figure 12 : Planning prévisionnel phase de développement

Voici le planning global prévisionnel du projet. Nous avons également prévu des phases de **mises au point** pour rattraper les éventuelles dérives de planning.



Figure 13 : Bilan des Jalons

Voici ci-dessus l'ensemble des **jalons du projet** dont les réunions d'avancement, le rapport et la soutenance suivie de la recette client.

Remarque : il a été décidé d'utiliser un outil plus professionnel pour la gestion de projet de manière à faire une gestion plus fine au niveau de l'avancement des tâches.

2.3 Étude de faisabilité et identification des risques

Dans un premier temps, une étude a été menée pour vérifier la faisabilité du projet et les risques pouvant s'opposer à sa réussite.

D'une part, cette étude consiste à vérifier que nous possédons toutes les **ressources (matérielles, compétences...)** nécessaires à la réalisation du projet.

D'autre part, **l'étude des risques** consiste à **isoler les risques potentiels** pour savoir **s'opposer à d'éventuels imprévus**.

Pour réaliser cette étude, il faut d'abord commencer par isoler **les besoins** du projet :

2.3.1 Besoins du projet

- Ordinateur avec un/plusieurs environnement(s) de développement
- Informations techniques : échanges Bluetooth, bornes wifi, triangulation (problématiques de localisation), effet doppler, propagation des ondes...
- Auto-formation à un nouveau langage (si nécessaire)

2.3.2 Scénarios (matrice SWOT)

Forces du projet	Faiblesses
<p><i>Compétences en informatique solides</i></p> <p><i>Projet intéressant qui peut venir répondre à un besoin commun</i></p>	<p><i>Nouvelles notions techniques non maîtrisées (propagation des ondes, notions mathématiques...)</i></p>
Opportunités	Menaces
<p><i>Se faire connaître</i></p> <p><i>Publier l'application</i></p> <p><i>Réaliser un projet qui peut être utile dans plusieurs cadres d'application</i></p>	<p><i>Mauvais choix de stratégie impliquant un retour en arrière dans la démarche</i></p> <p><i>Problèmes de matériel</i></p>

Figure 14 : Matrice SWOT

2.3.3 Identification des risques

Nature de risque	Description	Gravité	Responsable	Actions préventives	Actions correctrices
Humain	Un des membres du groupe est indisponible (raisons personnelles ou professionnelles)	3	Karim	Faire des points réguliers entre les membres du groupe pour que chacun ait connaissance de ce qui a été fait.	Demander à un autre membre du groupe de relayer la personne indisponible
Humain	Les résultats des essais ne correspondent pas à ceux attendus	3	Karim	Prévoir une phase de mise au point pour anticiper les retards éventuels	Faire remonter les résultats des essais et proposer des solutions pour débloquer la situation
Technique	Le matériel s'avère non adapté ou non fonctionnel au cours du développement du projet	4	Damien & Quentin	Tester rapidement les points négatifs au début du projet, les valider ou alors rechercher de nouvelles solutions	Faire remonter le problème et proposer d'autres solutions intégrant du nouveau matériel

Figure 15 : Identification des risques (1)

Nature de risque	Description	Gravité	Responsable	Actions préventives	Actions correctrices
Technique	Nouvelles notions techniques non enseignées	2	Karim	Auto-formation pour acquérir les outils nécessaires au projet	Demander des conseils auprès des enseignants concernés de façon à monter en compétences
Technique	Mauvais choix de stratégie (localisation) impliquant un retour en arrière dans la démarche	4	Julien	Effectuer une étude solide et un comparatif des différentes méthodes / procédés	Prévoir d'autres solutions de secours faciles à mettre en œuvre
Organisational	Mauvaise analyse du projet (décomposition, répartition...)	2	Karim	Analyser les problématiques au début du projet et anticiper.	Effectuer des revues d'avancement avec le client/tuteur pour être sûr que le projet est sur la bonne voie

Figure 16 : Identification des risques (2)

Ainsi, grâce à cette étude initiale, le projet a été validé car **le besoin est présent, la faisabilité a été validée et les risques sont gérables**. Tout au long du projet nous avons fait en sorte d'anticiper les risques et de réagir le plus rapidement possible pour trouver une solution et ainsi respecter les jalons.

2.4 Méthode de gestion de projet

Pour gérer ce projet, nous avons choisi de confronter deux méthodes beaucoup utilisées dans le monde professionnel. Il s'agit du **cyclen en V et de la méthode agile**.

2.4.1. Cycle en V

Le cycle en V est une méthode traditionnelle. Il s'agit d'une méthode qui permet **peu de changement** du fait de son cycle qui correspond à une démarche **peu propices aux évolutions**.

Certaines entreprises se refusent à abandonner cette méthode pour éviter les retours en arrière qui peuvent être lourds de conséquences dans un projet.

En ce qui concerne **l'organisation**, c'est sans doute le cycle en V qui prédomine. Lorsqu'il s'agit d'un projet « **from scratch** »¹, la démarche est dans un premier temps de décrire les spécifications, d'effectuer une analyse, de concevoir et enfin de commencer à développer. Ainsi s'en suivent les tests unitaires, les tests systèmes et les tests d'intégration. Toute cette procédure est longue et pas toujours adaptée.

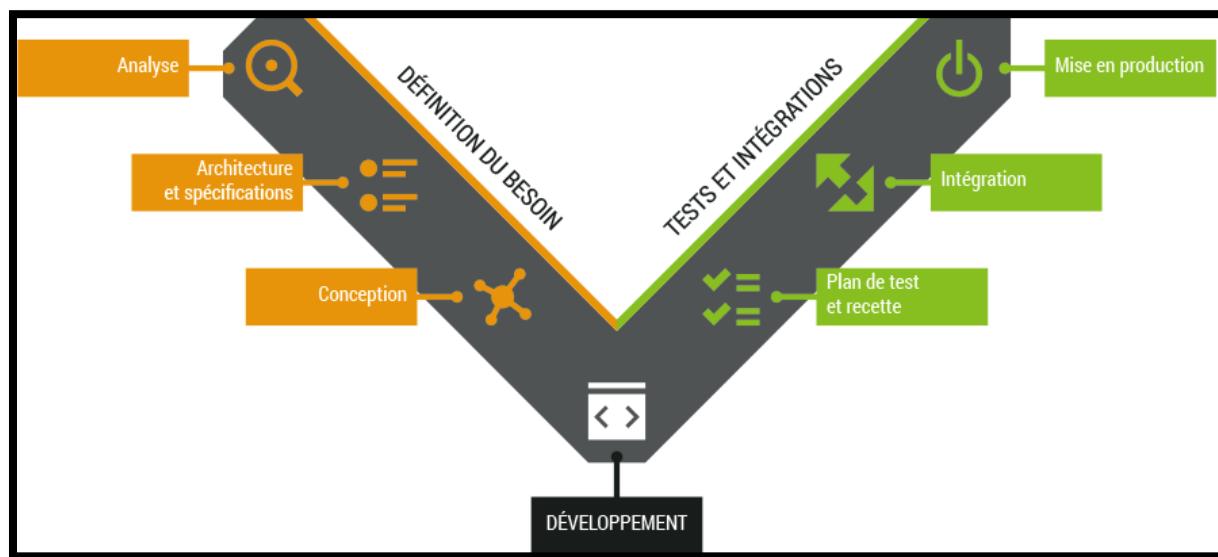


Figure 17 : Cycle en V

Dans le cadre de notre projet, nous avons fait le choix du cycle en V car il s'agissait dans un premier temps de **tout spécifier, tout planifier** à l'avance et ainsi suivre une **démarche prédéfinie** dans la réalisation et les tests du prototype (« Proof of concept »). Néanmoins, nous nous sommes aperçus que cette méthode n'était pas flexible et ne répondait pas à la manière de travailler de chacun. Nous nous sommes tournés vers la méthode agile.

¹ From scratch : « à partir de rien »

2.4.2. Méthodes agiles

Les méthodes agiles sont plutôt une **approche itérative**, ce qui permet de **réagir rapidement** à toute sorte de changement (technique, besoin client...). De même, cette méthode **favorise les échanges** entre les différents acteurs du projet. Cette méthode de gestion de projet permet de modifier les spécifications au cours des différentes sessions de travail. En effet, si jamais un client exprime le besoin d'une nouvelle fonctionnalité, cette méthode permet de réagir rapidement et d'intégrer cette tâche aux objectifs du prochain « **sprint** ».

Les méthodes agiles reposent aussi sur un processus itératif où l'on travaille par sous-ensemble que l'on vient intégrer au système au fur et à mesure des sessions de travail. Ces sessions ne sont pas forcément très courtes mais peuvent être plus longues pour pouvoir effectuer des correctifs.

Néanmoins, il n'est absolument pas nécessaire voir inutile d'utiliser les méthodes agiles pour des projets qui ne sont pas du tout flexibles avec des exigences fixées et peu d'étapes menant à la réalisation. Dans ce cas, il peut être préférable d'adopter le cycle en V.

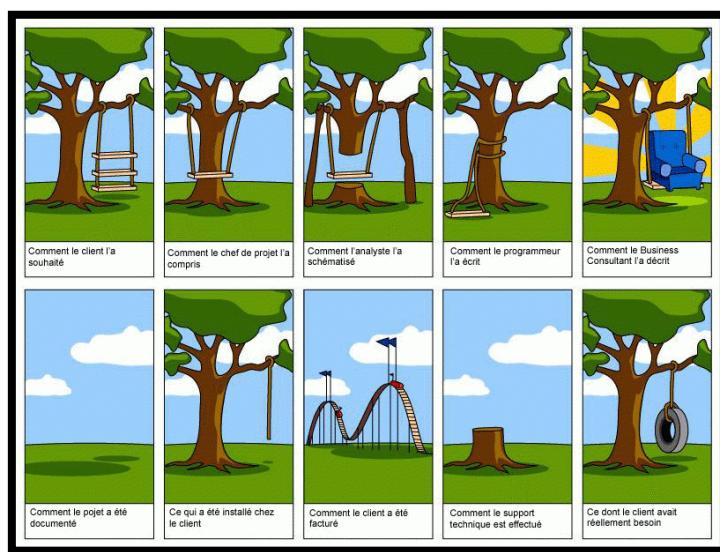


Figure 18 : La gestion de projet

En France, une grande partie des projets informatiques sont abandonnés ou tout simplement non utilisés pour la simple raison qu'ils ne répondent pas forcément aux besoins qui a soit été mal exprimé, soit mal été interprété. Pour éviter que cela se produise, il peut être intéressant d'utiliser les méthodes agiles qui mettent en place la méthode **Scrum**. Ces méthodes privilégient notamment **l'implication constante du client**, de façon à ce que les exigences soient bien prises en compte. De même, le client dispose d'une vue rapide et concise sur le projet, ce qui favorise encore l'efficacité. Eventuellement, si le client a une remarque, celle-ci peut être rapidement prise en compte. Si jamais cette démarche venait à ne pas être respectée, les méthodes agiles n'ont plus de sens.

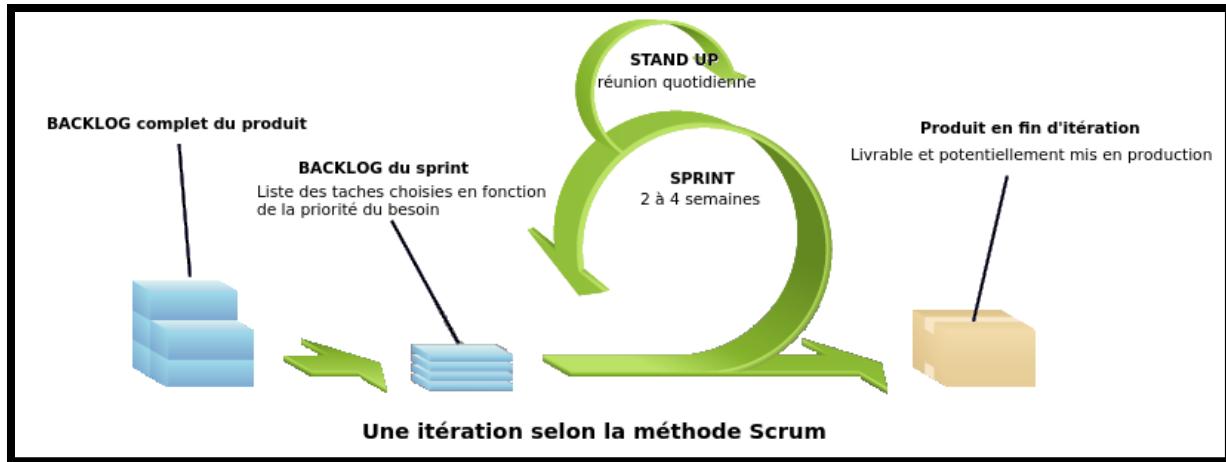


Figure 19 : Méthode Scrum

Le seul désavantage de cette méthode est le fait que la préparation des sprints est difficile et contraignante, surtout au niveau temps. Néanmoins, son efficacité a largement été démontrée.

Dans le cadre de ce projet, nous avons choisi d'adopter la méthode agile Scrum premièrement pour tous les arguments explicités ci-dessus. De même, cette méthode permet de travailler en décomposant le groupe en **lots**. Il est possible de **répartir les sous-ensembles** à développer à de sous équipes, d'où l'avantage de cette méthode agile qui démontre encore une fois sa flexibilité. Ainsi cette méthode a permis un suivi optimal sur l'avancement des groupes et des tâches. De plus, une méthode agile a été choisie car pour la plupart du temps, cette méthode est motivante contrairement au cycle en V qui ne laisse paraître un résultat que lors de la phase de tests et d'intégration. Dans le cas présent, le fait d'avoir utilisé une méthode agile nous a permis de valider le projet point à point afin de se rassurer sur la faisabilité des tâches les plus fragiles. De même, il a fallu s'adapter aux différents profils de membres de l'équipe. Ainsi, pour motiver les personnes, cette méthode paraissait la meilleure par son **caractère humain**.

2.5 Répartition des tâches

En ce qui concerne la gestion des ressources, celle-ci s'est réalisée par préférence et par spécialité. Nous avons choisi de décomposer le groupe en équipes pour réaliser plusieurs tâches. Une des particularités de la méthode agile Scrum est le **travail en binôme**. Nous avons donc choisi de suivre ce modèle-là pour faciliter la mise en œuvre. Néanmoins, la documentation n'a pas du tout été délaissée. Il a été fait en sorte de documenter toutes les tâches réalisées malgré les sprints. Cela a permis de pallier à l'inconvénient des méthodes agiles.

La répartition des tâches a été réalisée de la manière suivante :

		Karim	Julien	Brandon	Damien	Quentin	Axel	P.Martineau
Définir le projet	Effectuer une étude de l'existant (cadre d'applications, principes des beacons, étude du marché...)	R		A			A	I
	Établir une stratégie de localisation	R	A					I
	Prendre en main les beacons	R	S		A	A		I
	Rechercher des solutions techniques	R						
	Déterminer un environnement de programmation / langage	R	S	A	A	S		I
	Rechercher une librairie pour communiquer avec les beacons	R			A	A		I
	Déterminer une solution technique pour géolocaliser des cibles fixes & mobiles (rechercher le matériel)	R	A					I
	Rechercher une librairie pour la localisation des beacons (trilateration)	R			A		S	I
		R						
		R						
Études et réalisations	Effectuer une analyse logicielle	R						I
	Définir une architecture matérielle (structure du système)	R	A	S	S	S	A	I
	Rédiger un rapport d'analyse logicielle / fonctionnelle	R						I
	Développer une première application chaud / froid	R	A	A		S		I
	Concevoir une solution à base de balises mobiles (prise en main, essais, traitement des données)	R	A					I
	Réaliser l'interface graphique de l'application finale (guidage de l'utilisateur)	R		A			S	I
	R&D : Effectuer une étude permettant de trouver des beacons répondant au CDC de manière optimale	R			A			I
	Mise en service (tests unitaires)	R	A	A				I
	Validation	R						I

Figure 20 : Répartition des tâches

Globalement, Quentin et Damien étaient responsables de la partie matérielle, Axel et Brandon étaient en charge d'une partie de la veille technique et de tout ce qui est lié aux interfaces graphiques. Enfin Julien était chargé de gérer les aspects théoriques de la localisation. Karim était en charge de la gestion de toutes les tâches ainsi que de la mise en œuvre de la localisation.

2.6 Outils utilisés

2.6.1 Microsoft Office

En ce qui concerne les outils utilisés, nous avons choisi d'effectuer la gestion de projet à l'aide **d'Excel** et de **MS-Project**. Il est utile de définir son projet de manière temporelle sur Excel. Cela permet d'avoir une vue d'ensemble sur le projet simplement et rapidement. Ensuite, il est nécessaire de bien définir chaque tâche, de lui allouer des ressources et surtout de gérer les contraintes de précédence. Par conséquent, **Excel** est un peu limité pour une gestion de projet très fine, d'où le choix d'utiliser **MS-Project**. Cependant, il est très intéressant d'utiliser les deux outils en parallèle. En effet, Excel permet d'avoir un état des lieux global assez rapidement et en un simple coup d'œil tandis que ms-project fournit une solution de gestion sur **le long terme**.



Figure 21 : Excel



Figure 22 : MS-Project

2.6.2 Trello

Pour la gestion des **tâches au quotidien** et le **management personnalisé**, il peut être utile d'avoir un outil qui permet de gérer toutes les tâches de façon à avoir une vue globale sur le projet. Pour répondre à ce besoin, il a été choisi d'utiliser l'outil Trello. C'est un outil de gestion de projet en ligne inspiré par la méthode Kanban (méthode post-it). Il s'agit d'un tableau avec des tâches que l'on vient affecter au membre de l'équipe. Cela nous a également permis de faire le compte des heures de chacun sur chaque tâche qui compose le projet.

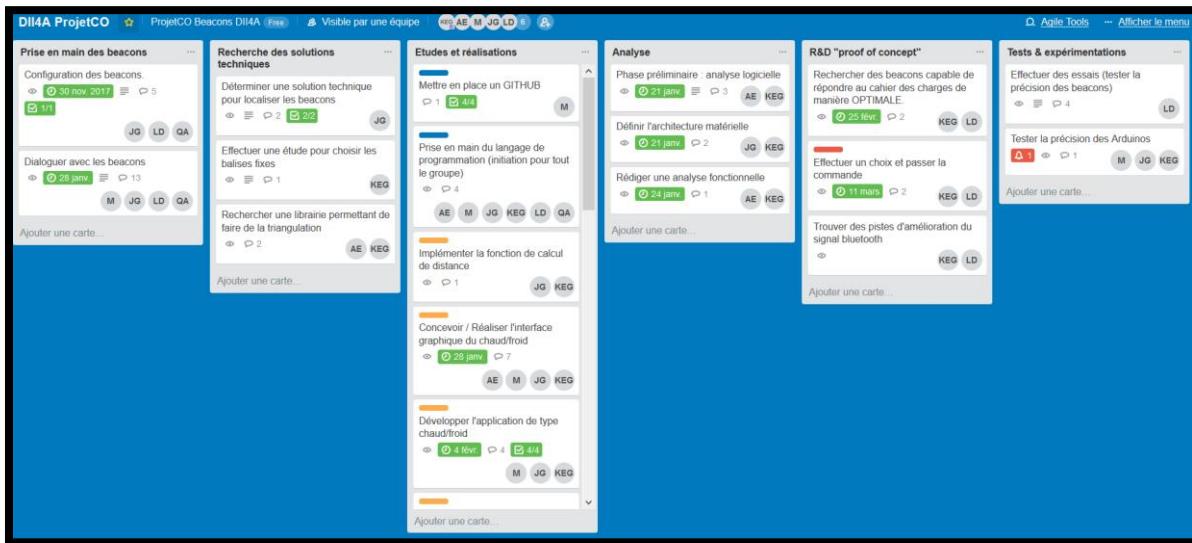


Figure 23 : Tableau Trello

2.6.3 Discord



Figure 24 : Discord

Pour communiquer tout le long du projet nous avons utilisé Discord, qui est un **logiciel de voix sur IP**, comme Skype, conçu principalement pour les communautés de joueurs. Logiciel multiplateforme créé en 2015, il utilise des technologies Web. Ce logiciel est totalement gratuit.

Il est possible de rejoindre ou créer un **serveur personnel**. Pour rejoindre un serveur il faut obtenir un lien vers le serveur en question. Si on souhaite créer un serveur, il doit posséder un nom ainsi qu'une image (optionnel) servant à identifier le serveur. Il est nécessaire de sélectionner la région du serveur, dans le but d'obtenir la meilleure connexion.



Figure 25 : Création d'un salon vocal

Il est possible dans ces serveurs de créer des salons vocaux ou textuels. Une fois le salon créé il est prêt à être utilisé.

Il est possible de manager le serveur, créer différents rôles mis sous forme de groupe d'utilisateurs, il est donc possible d'attribuer des permissions particulières à un groupe particulier, il est possible de consulter l'historique du serveur, et de faire bien d'autres actions...

En conclusion, Discord ne se limite pas seulement aux communautés de jeux vidéo. Avec ces différents outils nous avons pu créer un serveur dédié à notre projet. Autre avantage, ce logiciel est beaucoup moins intrusif que Skype. De plus chaque utilisateur peut modifier ses paramètres très facilement.

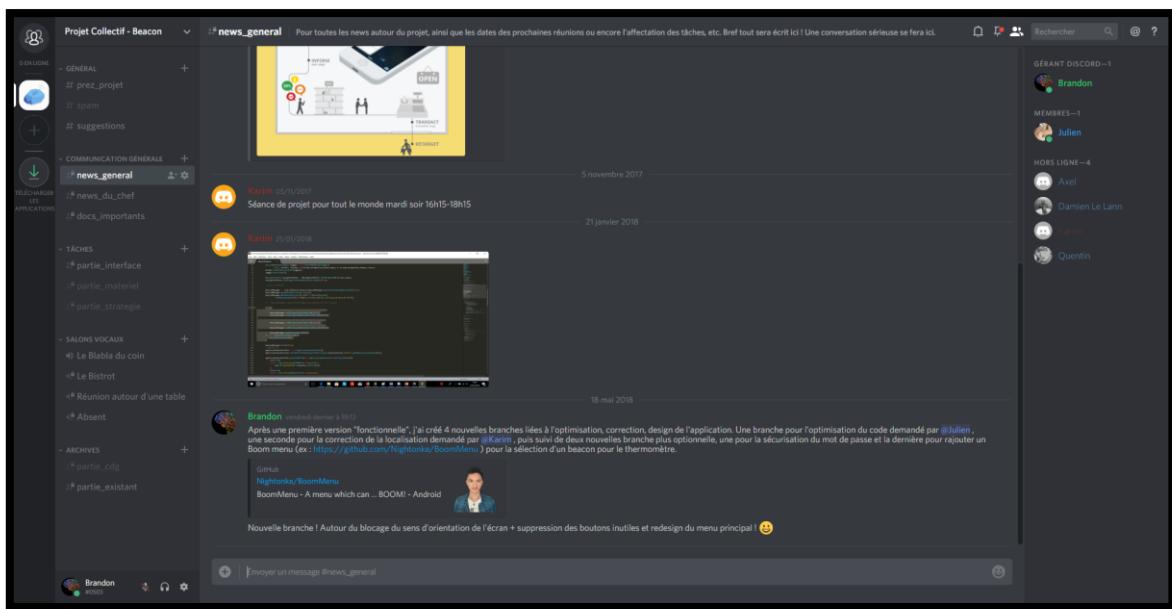


Figure 26 : Serveur Discord

Ci-dessus se trouve une capture d'écran de notre serveur Discord, ce serveur nous a été utile principalement pour pouvoir discuter et garder un historique des différents sujets.

Suite au travail de décomposition des tâches de Karim, nous avons pu créer différents salons textuels pour parler d'un sujet spécifique dans un salon qui lui ait propre (visible via la colonne de gauche, en dessous du nom de notre serveur « Projet Collectif – Beacon »).

Nous avons séparé ces différents salons afin d'avoir quelque chose de lisible qui permet donc de retrouver les informations que l'on souhaite rapidement (sans prendre en compte la fonctionnalité de recherche implémenté dans le logiciel, qui permet de parcourir tout l'historique de chaque salons textuels).

Via les salons vocaux, nous avons pu organiser des réunions ou encore des séances de travail. Cela a permis de résoudre le problème de distance.

Discord nous permet aussi en dehors des serveurs de communiquer en privé et de partager notre écran afin d'obtenir une aide supplémentaire.

Pour conclure, ce logiciel proposé par Brandon et Julien nous a permis garder un contact aisément tout le long du projet avec tous les membres constituant ce dernier.

3 Étude de l'existant

3.1 Rappel du contexte

On souhaite répondre aux exigences d'un client quelconque qui souhaite localiser des **objets fixes et mobiles** dans un **espace quelconque**. Les besoins sont divers et peuvent s'appliquer à **beaucoup de cadres d'application**. Par exemple, il est possible de mettre en place un tel système dans un magasin pour les rayons, dans un entrepôt de colis pour localiser des emplacements mais encore dans un hôpital pour localiser des lits. Dans le cadre de ce projet, on souhaite réaliser ce système avec des Beacons.

3.2 Qu'est-ce qu'un Beacon ?

Un Beacon est une **balise** émettant des **signaux Bluetooth** avec une portée de 30m théorique (BLE : Bluetooth low energy). Les Beacons ont été présentés par **Apple** (première utilisation à grande échelle dans 250 Apple Store Américains) en 2013 comme une nouvelle catégorie de transmetteurs à basse consommation. Ils sont de plus en plus utilisés dans le cadre de **développement d'applications mobiles** plus précisément autour de la géolocalisation ("géolocalisation intérieure principalement¹"). Il y a beaucoup d'applications autour du commerce, notamment pour localiser un client dans un magasin (envoi de notifications, promotions, paiement...). Cela peut permettre de contextualiser les messages en fonction de la position du client.

D'une part, les Beacons transmettent **un identifiant universel** qui peut être reçu par des périphériques (ex : mobiles iOS, Android...) pour convertir cette donnée en localisation. Les périphériques compatibles sont les terminaux IOS, Android et les ordinateurs Apple. Les principales limites sont le fait que le Bluetooth doit être activé mais aussi que l'application en question soit installée (problématique d'intrusion). Une balise Beacon est caractérisée par un identifiant universel unique (**UUID** d'une longueur de 16 octets). On différentie les balises à l'aide des valeurs **Minor/Major**. Il est possible d'associer au major une zone d'un espace et au minor une partie de cette zone.

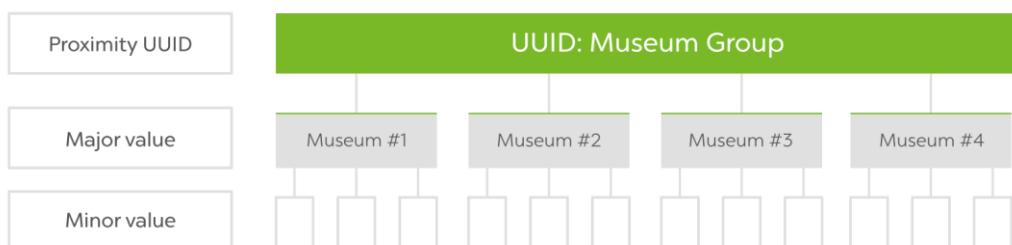


Figure 27 : Caractéristiques d'un beacon

Il est possible de capter le signal provenant du Beacon pour en déduire la distance par rapport à l'appareil (smartphone, tablette...). La distance peut être déduite par rapport à la puissance du signal émise (cf Annexe 2 > Comprendre la valeur du RSSI) que l'on peut facilement acquérir (**RSSI**).

De nos jours, les Beacons ont **plusieurs domaines d'application** : commerce, aéroports, transports, paiements, publicité, maison, événements, sport, musées, handicap, tourisme, éducation...

Il existe plusieurs avantages à utiliser les Beacons Bluetooth. Tout d'abord, la **consommation de batterie** est inférieure par rapport à l'utilisation de la technologie GPS. Ensuite, cette solution est **peu coûteuse** et offre des cas d'application divers, d'où son choix par le passé. L'autre critère est également la **simplicité d'utilisation**.

3.3 Étude du besoin

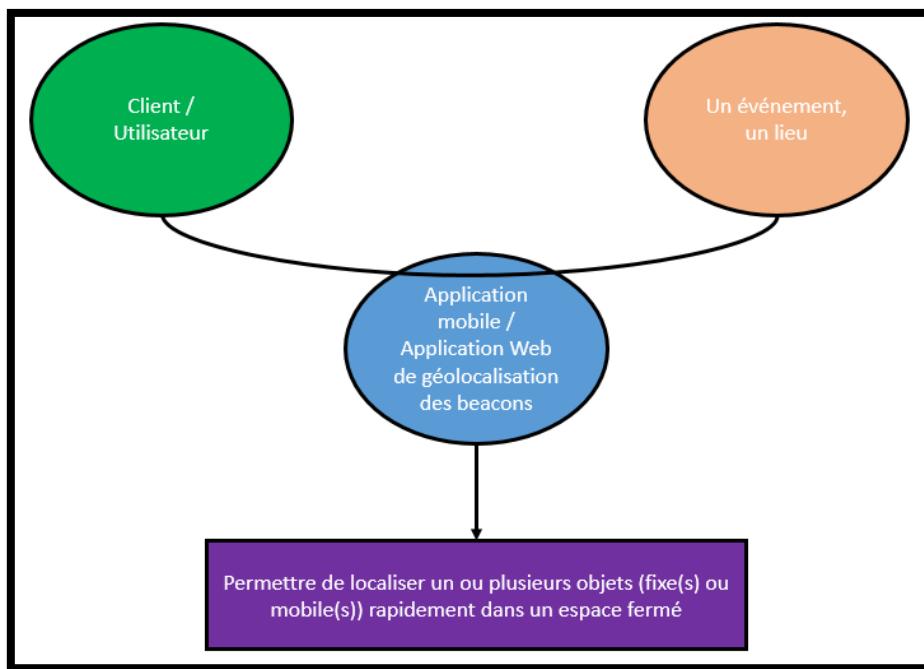


Figure 28 : Bête à cornes

Le diagramme ci-dessus (« Bête à cornes ») représente le besoin exprimé de la part de l'utilisateur. Notre application vient agir dans le cadre d'un événement (salons, sport...) ou d'un lieu (magasins, musées...) de façon à offrir la possibilité à l'utilisateur, de localiser un ou plusieurs objets.

Actuellement, le produit existe grâce à l'évolution des nouvelles technologies et de l'**internet des objets** (IOT), les personnes ont de plus en plus besoin de se simplifier la vie ou alors de la rendre plus confortable. De nos jours, le smartphone est devenu un objet indispensable dans le quotidien. De ce fait, les entreprises développent de plus en plus de technologies en rapport avec les smartphones. C'est le cas des Beacons qui ont été conçus notamment pour le domaine du commerce de façon à

envoyer des notifications personnalisées pour inciter à l'achat. De façon plus générale, cet appareil Bluetooth sert à localiser un objet de manière plus ou moins précise exclusivement dans des **espaces intérieurs** (localisation indoor).

Le besoin pourrait évoluer si jamais on aurait besoin d'une localisation très précise dans un espace vaste. Dans ce cas, une technologie plus développée devra être envisagée. Le besoin ne disparaîtra que s'il n'existe plus la nécessité de géolocaliser des objets (par exemple des clés, qui peuvent se perdre facilement).

Grâce à cette étude, nous avons pu nous convaincre que le besoin est existant, d'où l'utilité de notre projet.

3.4 Matériel de départ

Lors du démarrage du projet, il nous a été donné des Beacons fournis par Gimbal (Serie 10 Beacon).



Figure 29 : Beacon Gimbal

Ces Beacons ont été choisi par les groupes précédents pour les raisons suivantes :

Gimbal propose un SDK (cf Annexe 2) sur Android et iOS, ce qui peut être utile dans le cadre de la mise en œuvre. De même, cette solution est avantageuse de par les ressources qui existent : il existe une **communauté importante** avec des tutoriels ainsi qu'une documentation officielle. Ces Beacons ont été choisis pour une question de coût : environ **5\$** pour un **Beacon**. Le seul inconvénient étant **la dépendance à Gimbal** avec l'association des Beacons à un compte utilisateur. En résumé, ces Beacons sont une alternative intéressante dans le cadre d'une preuve de concept.

QUEL BEACON **CHOISIR** ?

	PUSH/PULL	GEOFENCING	CMS	SDK IOS	SDK ANDROID	PRÉCISION	COÛT	SECURISATION	GESTION DE LA BATTERIE
ESTIMOTE	X		X	X	X	***	\$\$\$		X
ROXIMITY	X		X	X	X	****	\$\$\$\$		
GIMBAL	X	X	X	X	X	****	\$		
BLUECATS	X		X	X	X	**	\$\$		
UBUDU	X	X	X	X	X	****	\$\$\$	X	X

Figure 30 : Comparatif constructeur

Pour effectuer un comparatif par rapports aux différents Beacons présents sur le marché, voici un tableau regroupant les principaux fournisseurs. On peut remarquer que Gimbal fournit des Beacons avec une **précision importante pour le moindre coût.**

3.5 Veille technique

3.5.1 Cadres d'application des Beacons

3.5.1.1 Applications déjà existantes

- MLB
- Travel Radar
- Virgin Atlantic
- Nivea
- Social Retail & ESSEC Business School
- Chipolo
- Wistiki

Voici quelques exemples d'entreprises utilisant la technologie **iBeacon**, afin de repérer différents Beacons disposés dans des espaces, soit pour repérer une place dans un stade (MLB), pour guider dans un aéroport, se situer par rapport à différentes zones, repérer une valise (Travel Radar, Virgin Atlantic)…

Il est possible aussi de recevoir une notification pour l'éloignement d'un enfant qui porte une balise BLE (Nivea), aider à trouver l'emplacement d'une salle de classe pour le prochain cours (Social Retail & ESSEC Business School) ou encore dans un cas plus général, repérer tout ce que l'on souhaite en accrochant une balise, comme la balise Chipolo ou la balise Wistiki, sur les clés, le portefeuille…

3.5.1.2 Cas d'utilisations

- Musée (affiche des infos en lien avec l'œuvre devant nous)
- Parking (retrouver sa voiture)
- Centre commercial
- Spectacles (rendre plus facile l'accès à sa place)
- Chasses aux trésors
- Horaires des bus
- Rappel de la liste de courses devant un magasin
- Émargement automatique pour les cours
- Retrouver un objet

Beaucoup de cas d'utilisation sont en lien avec les Beacons, mais dans ~90% des cas, ce sont les Beacons qui repèrent le téléphone, or ce que nous souhaitons, c'est réaliser l'inverse et de **repérer les Beacons**.

3.5.2 Limites des systèmes

La plupart des systèmes qui existent fonctionnent en direct, c'est-à-dire qu'il n'y a pas d'intermédiaire entre le smartphone et le Beacon. Par conséquent, il semblerait que le moyen le plus facile de trouver un Beacon est un « chaud-froid ». Il est possible d'estimer la distance entre un téléphone et un Beacon. Cependant, un mur, un objet liquide, un corps humain peut **fausser la donnée**, ou la rendre moins certaine, ce qui est un énorme point faible pour ce genre de système.

3.5.3 OS mobiles

Étant donné que l'objectif est de réaliser une application mobile, nous avons jugé intéressant d'effectuer une veille technique pour savoir si nous avions fait le bon choix mais surtout pour décider du système d'exploitation cible.

Les graphiques ci-dessous montrent les parts de marchés dont dispose le système d'exploitation Android ainsi que sa présence dans le monde. On peut voir que selon les prévisions et le bilan actuel, **Android est le système d'exploitation le plus répandu aujourd'hui** et le restera pour les prochaines années, au moins jusqu'en 2021. On ne peut donc pas se soucier de la durabilité de notre application. On peut également justifier ce choix par le fait que c'est un système assez ouvert notamment pour le développement. De même, il existe énormément de ressources vis-à-vis d'Android. Ce choix paraît donc être la meilleure solution.

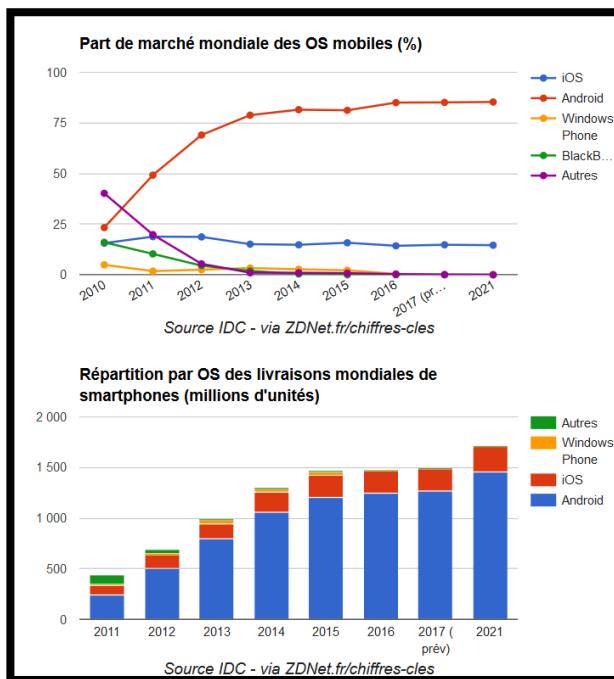


Figure 31 : Veille technique

À l'issue de cette veille technique, nous avons pu confirmer le choix de travailler sur Android.

4 Études et réalisations techniques

4.1 iBeacon ou Eddystone ?

Chaque Beacon fonctionne selon un protocole de communication : la partie matérielle gère la compatibilité avec un ou plusieurs principes d'échanges de données. Chaque protocole dispose de son propre format de trame.

4.1.1 iBeacon

iBeacon est une marque et une **norme** pour les Beacons BLE. Cette technologie a été développée par Apple en 2013. Il s'agit d'un système de positionnement intérieur qui permet d'envoyer des notifications vers les périphériques. iBeacon s'apparente à un protocole de communication.

iBeacon est actuellement **le format le plus répandu** pour les Beacons. De plus cette technologie est plus « mûre » que celles récemment développées. Par conséquent, nous avons fait le choix d'adopter ces technologies grâce aux ressources des années précédentes. Néanmoins, il peut être intéressant de s'intéresser à des technologies alternatives.



Figure 32 : iBeacon

4.1.2 Eddystone

Eddystone est une alternative à iBeacon mais cette fois-ci développée par Google. Cela permet la compatibilité avec de nouvelles interfaces entre les téléphones et les Beacons. La particularité d'Eddystone est l'envoi d'une URL et de données télémétriques en plus des données iBeacon. Cela n'est pas utile dans notre cas.



Figure 33 : Eddystone

4.2 Recherche de Beacons

Dans le cadre de la preuve de concept une étude a été effectuée pour avoir un avis sur les Beacons à utiliser. Cela permet de savoir s'il existe d'autres modèles sur le marché répondant aux exigences du cahier des charges. L'objectif est également de trouver des technologies pouvant apporter une meilleure précision ou un avantage quelconque au système à développer.

Dans cette étude nous nous sommes attachés à sélectionner des Beacons capable d'émettre des trames de type **iBeacon** et capable d'être utilisés sans SDK propriétaire qui occasionnerait des développements supplémentaires et un surcout.

4.2.1 Avant-propos

Il est important de définir quelques termes avant de pouvoir comparer toutes les solutions des différents constructeurs.

Autonomie : durée maximale durant laquelle le Beacon peut fonctionner sur sa source d'énergie.

Portée : distance maximale (idéale) à laquelle peuvent être reçues les trames BLE sans aucun obstacle.

Delta température : plage de température que le Beacon capable de mesurer. (Paramètre secondaire)

Plage d'émission : plage d'émission que le Beacon est capable d'émettre. Une valeur négative correspond au signal nominal atténué, une valeur positive correspond au signal nominal amplifié. Cette valeur s'exprime en Décibel (dB).

4.2.2 Gimbal

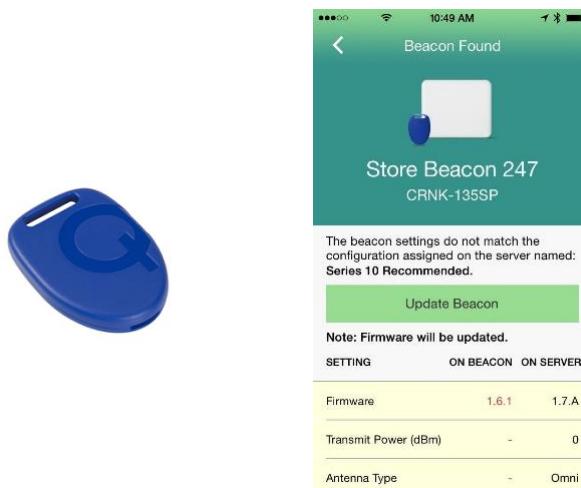


Figure 34 : Beacon Gimbal & Application de configuration

Critères	Valeurs
Autonomie	4 mois
Portée	50 mètres
Delta température	20 à 60 °C avec une précision de 2°C à 4°C
Plage d'émission	0 à -20 dbm
Fréquence d'émission	200 ms
Prix	5 \$

Figure 35 : Caractéristiques Gimbal

Remarque : les autres modèles de la marque (21 et 22) ne présentent *que peu d'intérêt (Antenne unidirectionnelle)*. En effet la plage d'émission de l'antenne reste similaire.

Points forts :

- Prix faible
- Solution compacte
- Application de gestion

Points faibles :

- Très faible précision

4.2.3 RadBeacons



Figure 36 : RadBeacon

Critères	Valeurs
Autonomie	18 mois
Portée	5 à 50m
Delta température	-4 °C à 60°C
Plage d'émission	+4dBm a -20dBm
Fréquence d'émission	100ms - 200ms - 1000ms
Prix	24\$

Figure 37 : Caractéristiques RadBeacon

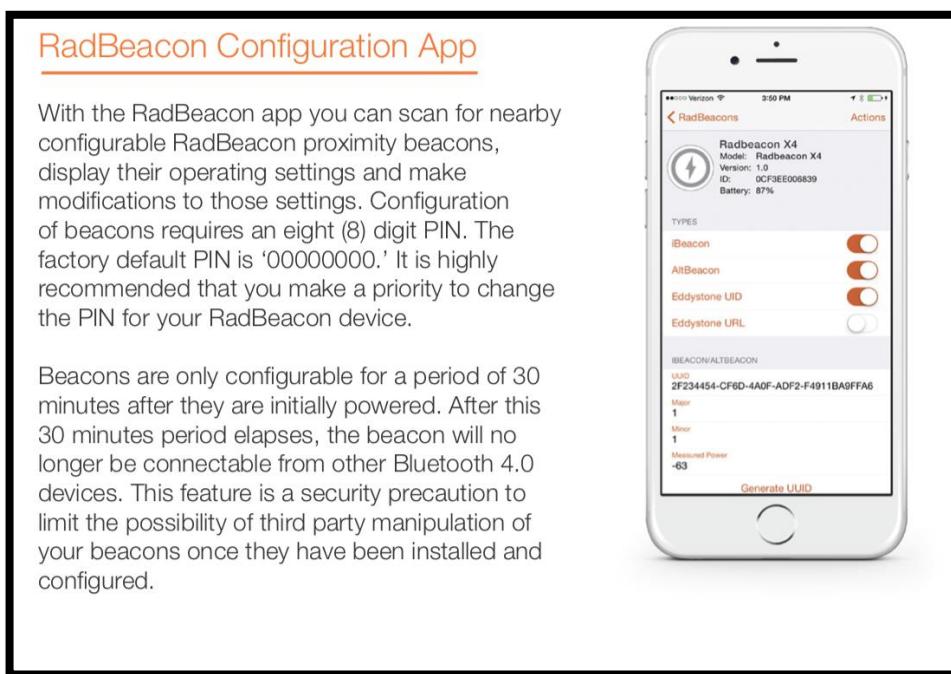


Figure 38 : Application de configuration RadBeacon

Points forts :

- Application de gestion
- Compatibilité avec **altbeacon**
- Portée élevée
- Puissance d'émission

Points faibles :

- Prix élevé

RadBeacon est vendu par l'entreprise assurant le développement de la librairie **AltBeacon**. La plage d'émission est légèrement supérieure à celle des Beacons Gimbal. De plus, au sein de la documentation de librairie AltBeacon il est souvent fait référence à ces Beacons garantissant une compatibilité avec cette solution.

4.2.4 Kontakt.io

4.2.4.1 Beacon Pro

Critères	Valeurs
Autonomie	5 ans
Portée	80m
Delta température	-20°C à 60°C
Plage d'émission	4 à -20 dbm
Fréquence d'émission	100ms à 10000ms
Prix	30\$

Figure 39 : Caractéristiques Kontakt

Capacités supplémentaires :

- Humidité de 0% à 95%
- Accéléromètre et capteur de luminosité (Possibilité de gérer l'extinction en fonction de la luminosité)
- Puce NFC
- Bluetooth 5 (Portée accrue) mais nécessite un terminal compatible



Figure 40 : Beacons Kontakt.io & Application de configuration

Points forts :

- Application de gestion
- Puissance d'émission
- Portée très élevée

Points faibles :

- Prix élevé

4.2.5 Gateway



Figure 41 : Gateway Kontakt

La marque Kontakt.io propose une balise dite « Gateway » pouvant permettre une localisation précise. Néanmoins celle-ci nécessite l'utilisation d'un SDK et d'un service BACK END propriétaire.

Il est tout de même à noter que les balises peuvent être utilisées indépendamment du SDK propriétaire comme les prouve de nombreux exemples sur GITHUB :

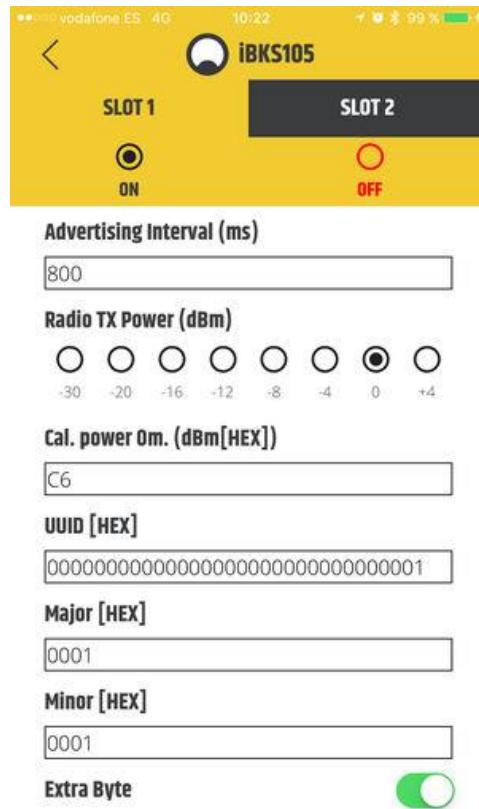
<https://github.com/mlomnicki/Android-Kontakt-Beacons>

Les Beacons Kontakt proposent une puissance d'émission supérieure ainsi qu'une portée intéressante. Néanmoins, ils détiennent un coût relativement important. De plus, il est nécessaire de réaliser cet achat à l'étranger.

4.2.6 BNB Beacons



Figure 42 : BNB beacons & Application de configuration



Critères	Valeurs
Autonomie	10 ans
Portée	100m
Delta température	-40°C à 85°C
Plage d'émission	+4 à -30 dbm
Fréquence d'émission	100ms à 10000ms
Prix	29€ H.T

Figure 43 : Caractéristiques BNB

Points forts :

- Application de gestion
- Puissance d'émission
- Portée très élevée

Points faibles :

- Prix élevé

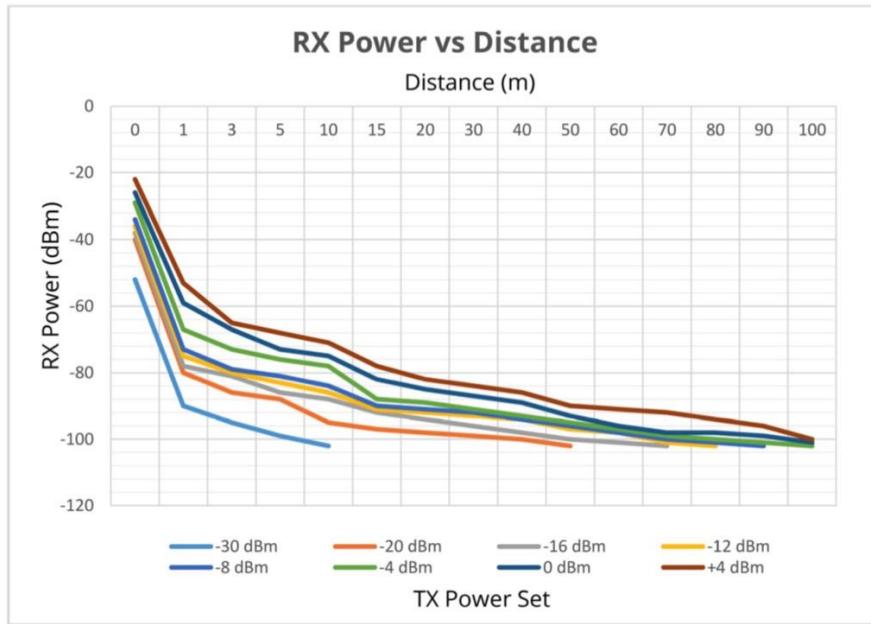


Figure 44 : Puissance reçue en fonction de la distance

Distance (m)	TX Power (dBm)							
	-30	-20	-16	-12	-8	-4	0	+4
0	-52	-40	-38	-36	-34	-29	-26	-22
1	-90	-80	-78	-75	-73	-67	-59	-53
3	-95	-86	-81	-80	-79	-73	-67	-65
5	-99	-88	-86	-83	-81	-76	-73	-68
10	-102	-95	-88	-86	-84	-78	-75	-71
15	-97	-92	-91	-90	-88	-82	-78	-78
20	-98	-94	-92	-91	-89	-85	-82	-82
30	-99	-96	-93	-92	-91	-87	-84	-84
40	-100	-98	-94	-94	-93	-89	-86	-86
50	-102	-100	-97	-96	-95	-93	-90	-90
60		-101	-98	-98	-97	-96	-91	-91
70		-102	-101	-100	-99	-98	-92	-92
80			-102	-101	-100	-98	-94	-94
90				-102	-101	-99	-96	-96
100					-102	-101	-100	-100

Figure 45 : Relevés de puissance en fonction de la puissance émise et de la distance

Ces Beacons ont l'avantage de provenir d'une société de la région garantissant ainsi un suivi après-vente de qualité. De plus ils font partie des Beacons proposant une **puissance d'émission importante (+4db)** comme le démontre les illustrations ci-dessus.

4.2.7 Conclusion de l'étude

Les Beacons BNB semblent être la solution la plus pertinente car ils proposent la **portée** la plus importante et la **puissance d'émission** la plus importante. De plus, ils font appel à une entreprise de la région garantissant un service après-vente de qualité.

Néanmoins il est à noter que cette étude a été difficile à mener. En effet, même si la puissance d'émission est une donnée factuelle ne pouvant pas être remise en cause, il n'y a pas de norme claire et précise pouvant garantir une comparaison factuelle entre chaque Beacon pour les autres caractéristiques.

Un autre point important est celui de la **sensibilité**, il n'est en effet jamais fait référence au gap minimal de distance pouvant être mesuré par le Beacon. La sensibilité est pourtant une valeur primordiale dans un calcul de distance.

4.3 Prise en main et configuration du matériel

4.3.1 Beacons Gimbal

Les premiers Beacons que nous avons eus à notre disposition sont les Beacons Gimbal. Pour les mettre en place nous avons dû utiliser le site Gimbal associé à un compte.

Dans un premier temps, il faut aller dans l'onglet Beacon management et y ajouter la clé présente dans le Beacon.



Figure 46 : Configurations Gimbal (1)

Dans un second temps il faut créer une configuration comprenant UUID, minor, major, technologie de la trame.

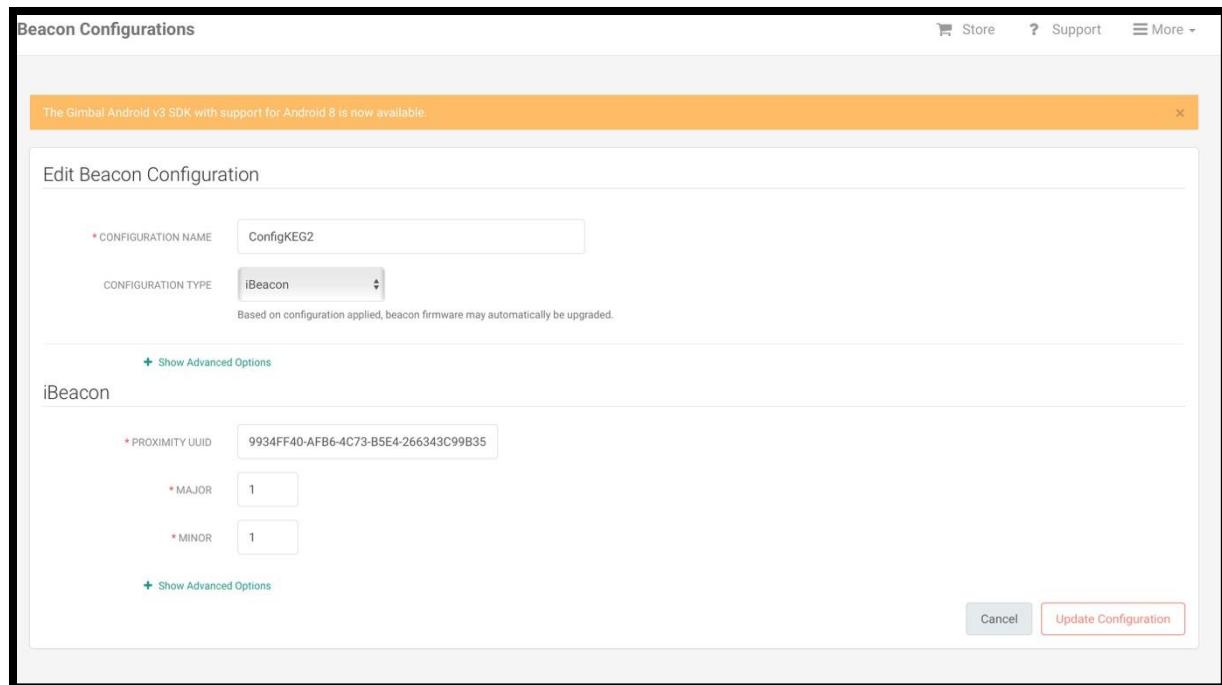


Figure 47 : Configurations Gimbal (2)

Enfin, il suffit d'associer la configuration au Beacon dans l'onglet Beacon management.

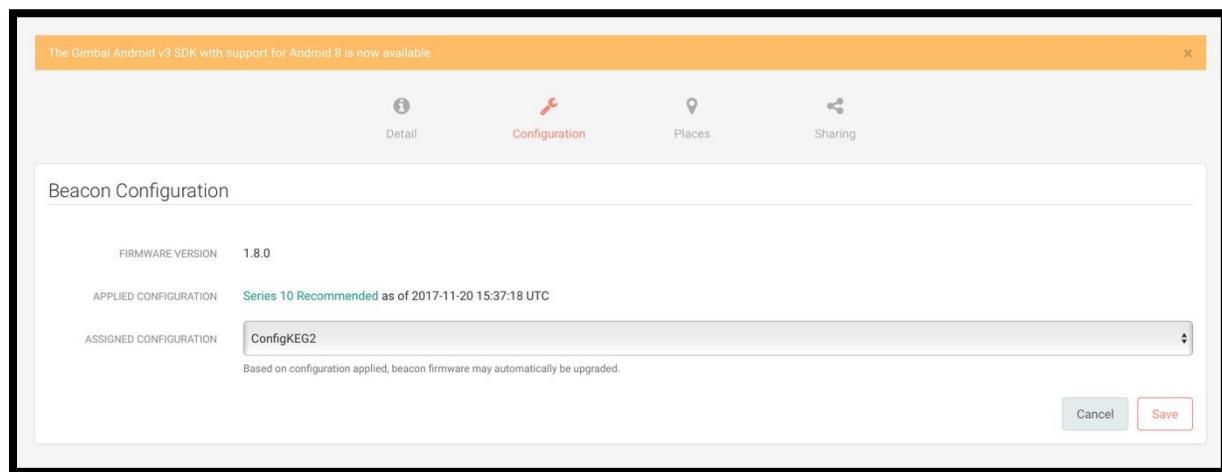


Figure 48 : Configurations Gimbal (3)

4.3.2 Beacons BnB

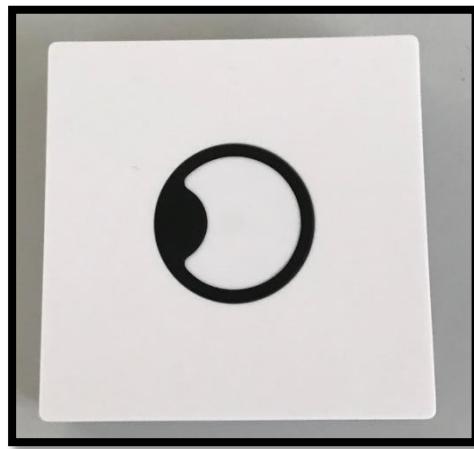


Figure 49 : Beacons BNB

À la réception des nouveaux Beacons BNB, nous avons entamé une démarche de mise en service de ceux-ci pour obtenir une configuration similaire à celle des Beacons Gimbal.

Pour cela nous avons utilisé l'application recommandée par le fabricant à savoir **iBKSCConfigTool**. Cette application est disponible sur Android et IOS.

L'application se présente ainsi :

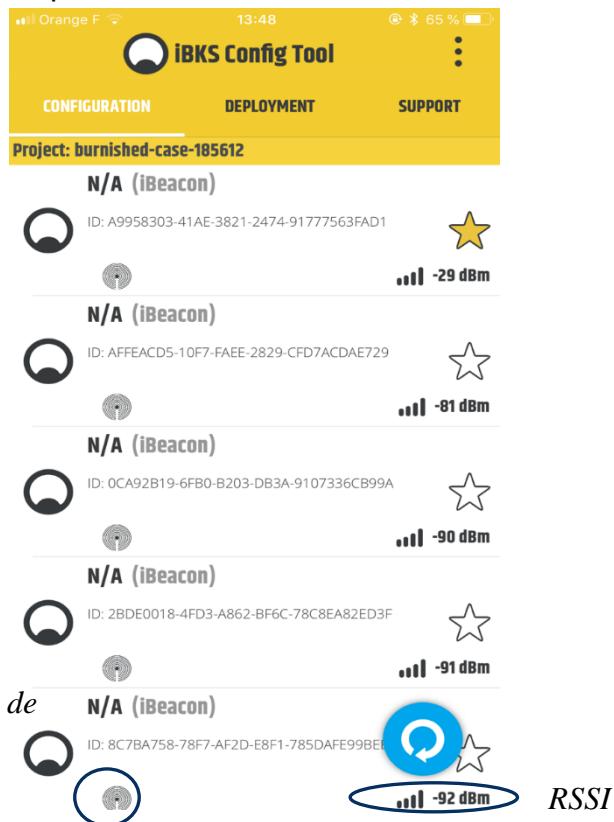


Figure 50 : Configurations BNB (1)

Sur le premier volet, on retrouve tout les objets BLE présent autour du téléphone ainsi que le RSSI relevé. Un petit logo permet de différencier si celui-ci émet des trames iBeacon ou Eddystone.

Pour configurer un Beacon, il faut autoriser l'accès. Pour cela, il existe un bouton sur la face avant du Beacon pour autoriser l'accès à la configuration du Beacon. L'accès reste ouvert 30 secondes après l'appui de celui-ci. Une led verte représente cette durée.

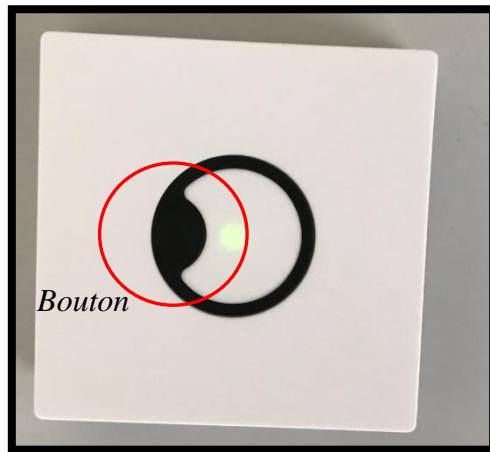


Figure 51 : Beacon BNB

Après avoir choisi le Beacon désiré un volet s'ouvre donnant accès à la configuration. En haut à droite l'onglet « ... » permet de modifier le nom du Beacon et son mot de passe.

Ensuite trois menus deviennent accessibles : le premier permet de mettre en place une configuration Eddystone, le second pour une configuration iBeacon, le dernier permet de modifier la configuration générale du Beacon.

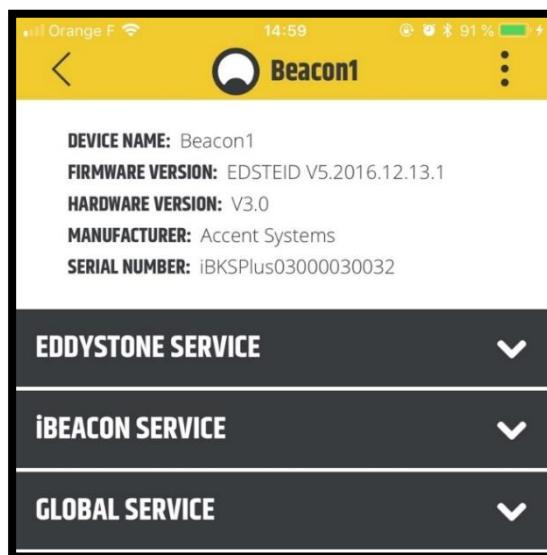


Figure 52 : Configurations BNB (2)

En ce qui concerne la configuration iBeacon, il est possible de modifier l'intervalle entre l'envoi de chaque trame (standard : 100 ms min), la puissance d'émission du signal (+4db max), l'UUID et enfin le major et le minor.

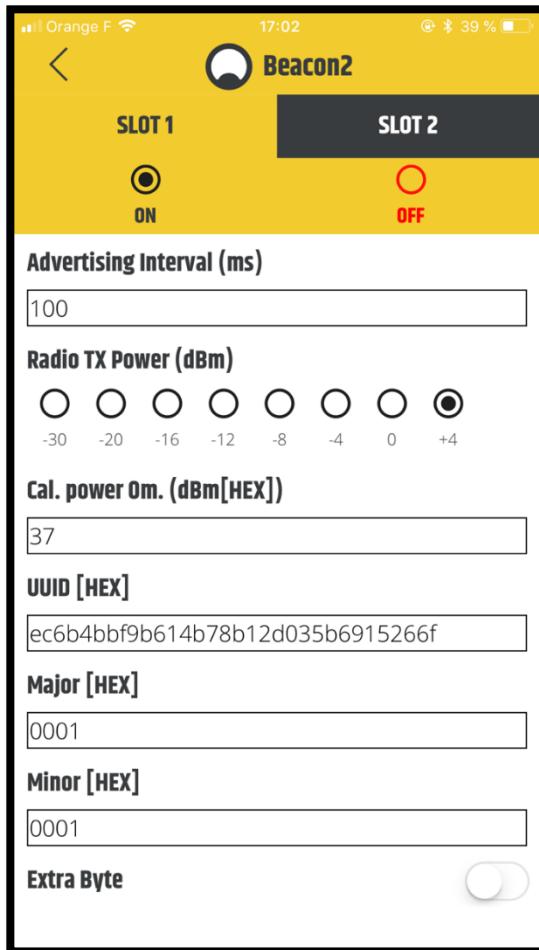


Figure 53 : Configurations BNB (3)

Une fois la configuration souhaitée choisie, il est nécessaire calibrer le Beacon. Pour cela, la méthode est assez simple :

- Se placer à 1 mètre du Beacon
- Relever la valeur du RSSI reçue

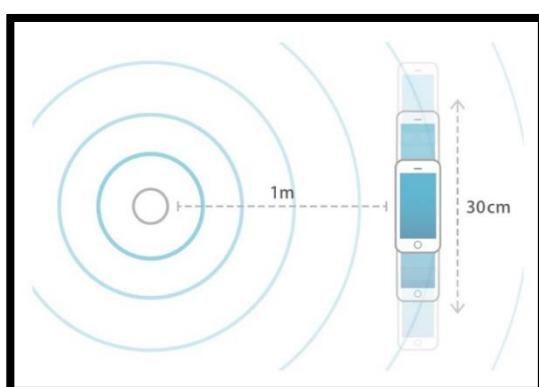


Figure 54 : Calibration RSSI (1)

Il faut ensuite insérer la valeur ainsi obtenue dans le champ « cal.power 0.m ».

Pour vérifier que la calibration a bien été réussie, il a été effectué un test de validation. Ce test consiste à vérifier **l'équidistance** des différentes distances calibrées en relevant le RSSI. Dans le cas présent, les trois valeurs doivent être identiques.

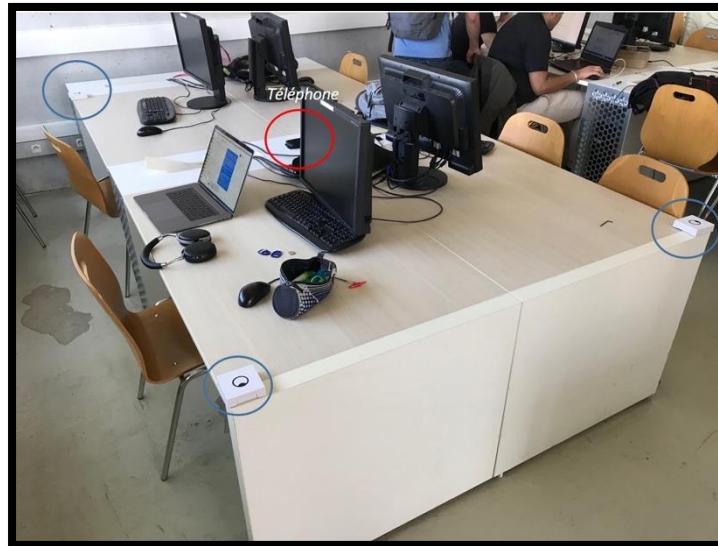


Figure 55 : Calibration RSSI (2)

4.4 Mise en place d'une stratégie de localisation

4.4.1 Les différentes stratégies

Pour ce projet, nous avons pensé à plusieurs stratégies afin de localiser les Beacons. Nous avons fait une réunion afin de recueillir l'ensemble des idées.

La première idée qui est ressortie est un « chaud-froid ». Cela consiste en la récupération du **RSSI** pour calculer la distance du Beacon et l'indiquer à l'utilisateur par un élément graphique de type **thermomètre**.



Figure 56 : Chaud / Froid

La seconde idée est de localiser le Beacon grâce aux différentes valeurs de sa distance que nous **historisons**. Le téléphone disposant d'un GPS, il nous serait possible de connaître sa position et sa direction, puis grâce à la distance du Beacon, savoir dans quelle direction le Beacon se dirige.

Pour détailler, lorsque deux cibles se déplacent dans la même direction, dans des sens opposés, les vitesses s'additionnent. Ces vitesses sont réduites lorsque les cibles vont dans le même sens, dans la même direction. Il serait donc possible, par une historisation des données, de connaître la position des Beacons. Cependant, cela demande de bien connaître le système et un développement important.

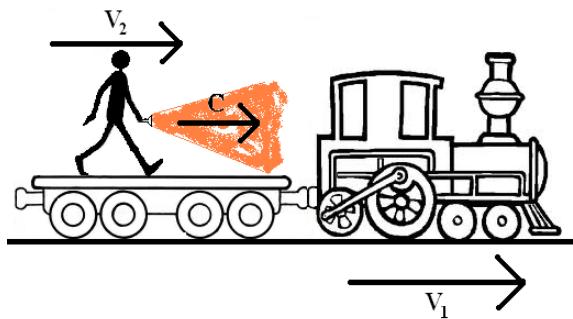


Figure 57 : Historisation des données

La troisième idée était de se servir de l'**effet Doppler** pour connaître les directions des Beacons. L'effet Doppler permet de savoir si une cible s'éloigne de la source d'un signal ou si elle s'en rapproche par des calculs simples. En y ajoutant la distance à un Beacon, nous pensions pouvoir estimer un angle et donc une position.

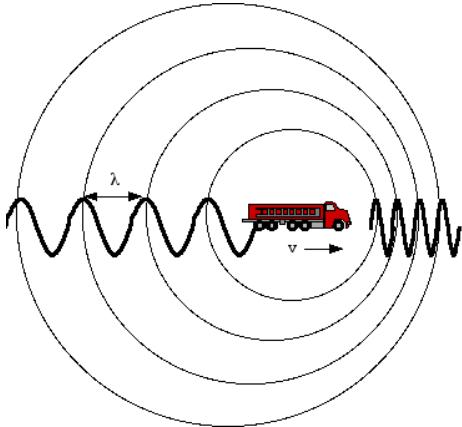


Figure 58 : Effet Doppler

Enfin, nous avons pensé à une dernière solution qui est l'utilisation de balises fixes et de la trilateration pour pouvoir définir des positions. Chaque balise fixe récupère la distance du Beacon et l'envoie à un terminal qui concentre l'ensemble des données, lesquelles sont utilisées par un calcul de trilateration qui définit un point précis duquel nous pouvons trouver le Beacon.

4.4.2 Choix de la stratégie

Pour le choix final, nous avons opté pour la solution des balises fixes et de la trilateration pour des raisons évidentes de temps de développement mais surtout, de précision des données. Cependant, le chaud-froid nous a permis de tester nos Beacons et est toujours intégré à l'application pour y ajouter un petit jeu.

La seconde idée, la localisation par l'**historisation des données**, n'était au final pas claire et n'a pas été utilisée pour ce manque de compréhension, mais aussi parce que l'idée n'était pas assez précise et nous n'étions pas sûr d'arriver à détecter un Beacon précisément.

Pour la troisième stratégie, nous pouvions avoir l'éloignement ou l'approche du Beacon, mais pas une distance et une position précise, elle n'a donc pas été utilisée.

4.4.3 Triangulation ou trilateration ?

Dans le but de réaliser un calcul de position nous devions trouver un algorithme de calcul. Deux choix se sont offerts à nous : la trilateration et la triangulation.

La triangulation utilise les **angles** entre le point connu et le point recherché en traçant une droite vers le point inconnu. On répète l'opération avec tous les points connus et on obtient la position de la cible. Plus il y a de points connus plus le positionnement est précis.

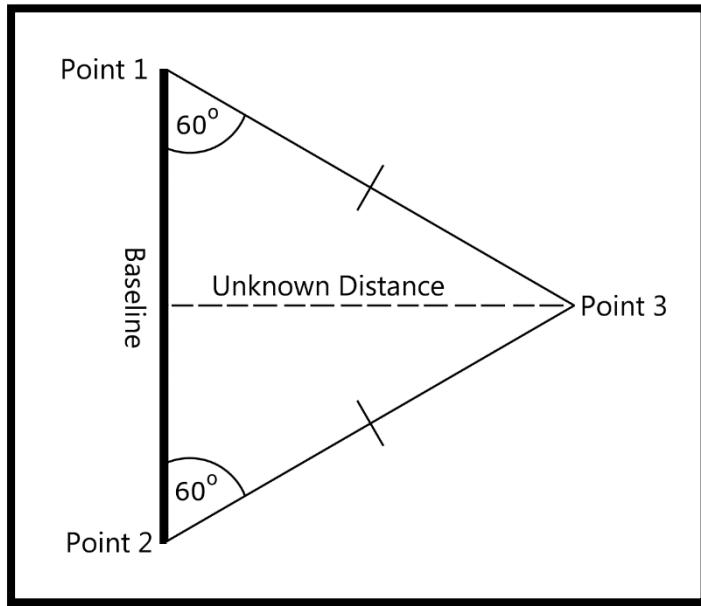


Figure 59 : Triangulation

La trilateration utilise les **distances** avec le point à positionner à partir de tous les points déjà connus. Tout comme la triangulation : plus le nombre de points connus est important plus le positionnement est précis.

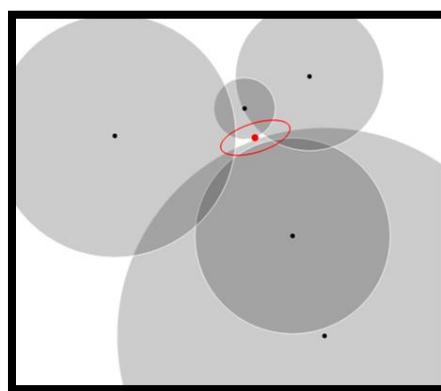


Figure 60 : Triangulation

Nous avons choisi d'utiliser l'algorithme de trilateration car n'ayant pas de mesure d'angle nous ne pouvons pas utiliser l'algorithme de triangulation.

4.5 Architecture matérielle

4.5.1 Aspect « hardware »

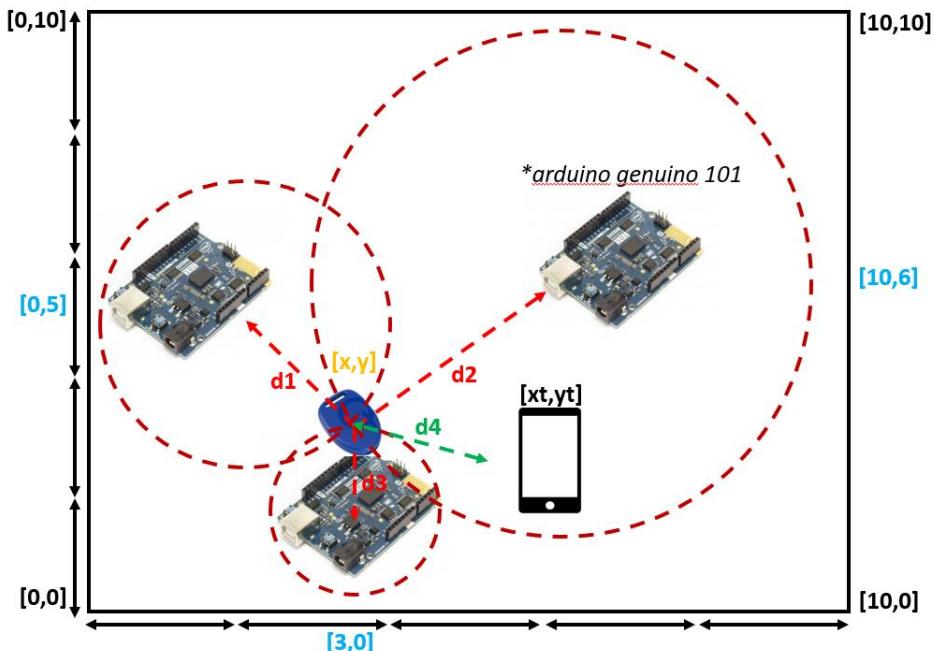


Figure 61 : Architecture matérielle

Nous avons choisi de mettre en œuvre une architecture comme celle décrite ci-dessus. Pour cela, nous appliquons notre stratégie de localisation introduite ci-dessus : la trilateration. Pour cela, il est nécessaire de mettre en œuvre **au moins trois balises fixes**. Pour réussir à mettre en œuvre ce principe, il faut récupérer la distance entre la cible et chaque balise. Ainsi, par le principe de trilateration, il est possible de déterminer une position.

4.5.2 Protocole(s) de communication

La communication entre les balises fixes et les Beacons se fait en BLE par défaut : les Beacons envoient les données en Bluetooth par **broadcast**. Par conséquent, il n'était pas possible de changer de protocole de communication. En revanche, nous avions le choix pour la communication entre les balises fixes et le téléphone. Il nous est offert de mettre en place différents protocoles de communication pour restituer les informations à l'utilisateur. Il a été choisi de garder le même protocole pour une question de simplicité et d'uniformisation. La seule limite de ce choix est la portée qui aurait pu être augmentée par une autre solution tel que le wifi.

4.6 Recherche de solutions techniques

4.6.1 Langage de développement & API d'interfaçage

Pour développer notre application, il a été décidé d'utiliser l'environnement de développement **Android Studio** étant donné notre choix. Par conséquent, le langage choisi est **Java** car nous avons eu des cours en 4A et que les plateformes Android supportent nativement ce langage. Néanmoins, nous avons jugé utile de faire l'étude de plusieurs environnements et langages de développement. De même, il peut être utile de comparer les solutions destinées à des cibles différentes d'Android.

Pour communiquer avec les Beacons, il est nécessaire d'avoir **une librairie (API)**, de manière à pouvoir détecter les Beacons et récupérer les informations diffusées. Nous avons donc fait en sorte de croiser les deux études pour déterminer un langage de développement en fonction de la meilleure API disponible.

Librairie Beacons	Langage	Avantages	Inconvénients
Gimbal SDK	Java /Objectif C/Swift	<ul style="list-style-type: none"> Gimbal est la librairie native de nos Beacons Communauté relativement importante 	<ul style="list-style-type: none"> Utilisation de la librairie peut être parfois fastidieuse
Estimote	Java /Objectif C/Swift	<ul style="list-style-type: none"> Librairie multi-langage Communauté relativement importante Beacons de bonne facture (Précision élevé) 	<ul style="list-style-type: none"> Framework limité aux Beacons de la marque (Prix prohibitif)
Eddystone + Neaby Api	Java	<ul style="list-style-type: none"> L'alternative de Google à iBeacons Ouvert à la majorité de constructeurs de Beacons 	<ul style="list-style-type: none"> Nécessité de passer par l'ide dans le Cloud Google Développer console (Quid de l'utilisation). Communauté assez faible Librairie portée sur les

			notifications à champ proche plus que la localisation
AltBeacons	Java	<ul style="list-style-type: none"> • Open source et indépendante • Grande communauté (abondance de tutoriels et aide en tout genre) • Librairie multiplate-forme (windows /Linux /raspberry Pi) • Possibilité de s'appuyer sur les travaux des années précédentes 	

Figure 62 : Langage de programmation

À travers cette étude, nous avons pu en conclure que la meilleure solution était la librairie **AltBeacon** de par sa simplicité d'utilisation mais également grâce au nombre important de ressources issues des projets précédents. De même, cette librairie est totalement compatible avec Android.

Remarque : une autre solution possible aurait pu être de développer une application avec la bibliothèque .NET grâce à Xamarin. Cependant, le manque de compétences *dans la création d'application mobile grâce cette technologie n'a pas fait de ce moyen une solution prioritaire.*

4.6.2 Choix des balises fixes

En ce qui concerne les balises fixes, il a été utile d'effectuer une étude dans le but de déterminer la technologie à adopter pour mettre en place les balises fixes nécessaires à la trilateration. Pour cela, plusieurs solutions se sont offertes à nous : Raspberry, Arduino, Udo, mbed...

Nous avons décidé d'étudier de plus près deux technologies pour en déterminer une dans le cadre d'une **preuve de concept**.

4.6.2.1 *RaspBerry Pi*



Figure 63 : Raspberry PI

Ce « mini-ordinateur » permet des déploiements rapides. Cependant, le système est lourd et a besoin d'un **système d'exploitation** pour fonctionner. Malgré sa simplicité, nous avons jugé cette solution trop complexe pour notre besoin.

4.6.2.2 *Arduino/Genuino 101*



Figure 64 : Arduino Genuino 101

Cette carte programmable permet de réaliser rapidement des **prototypes** et dans notre cas, de déployer une preuve de concept. Réputée pour leur **facilité** et les librairies disponibles sur internet, cette solution nous a parue évidente pour les raisons suivantes :

- Connaissances de l'environnement arduino dans le groupe
- Existence d'une librairie BLE trouvée après des recherches : **CurieBLE**
- Intégration d'un module Bluetooth « nativement » (Intel Curie)
- Présence d'exemples pour des tests et une mise en œuvre rapide.

Ces quatre arguments nous ont permis de statuer sur la Genuino 101.

4.7 Conception et développement de l'application « chaud-froid »

Pour répondre au point du cahier des charges lié à la localisation d'un Beacon fixe, la méthode choisie a été celle du chaud-froid. Cela a permis de concevoir puis de développer une première application validant la première exigence. Cela nous a offert la possibilité de prendre en main le matériel et l'environnement de développement.

4.7.1 Analyse

4.7.1.1 Analyse fonctionnelle

En ce qui concerne l'aspect fonctionnel, il a été choisi de développer une fonction permettant de récupérer la puissance du signal du Beacon fixe et d'en déduire la distance entre le mobile et le Beacon.



Figure 65 : Fonction de calcul de la distance

Cette fonction devra donc effectuer un calcul qui sera définir par la suite. De plus, une API est nécessaire pour permettre au téléphone de communiquer avec les Beacons environnants.

4.7.1.2 Analyse logicielle

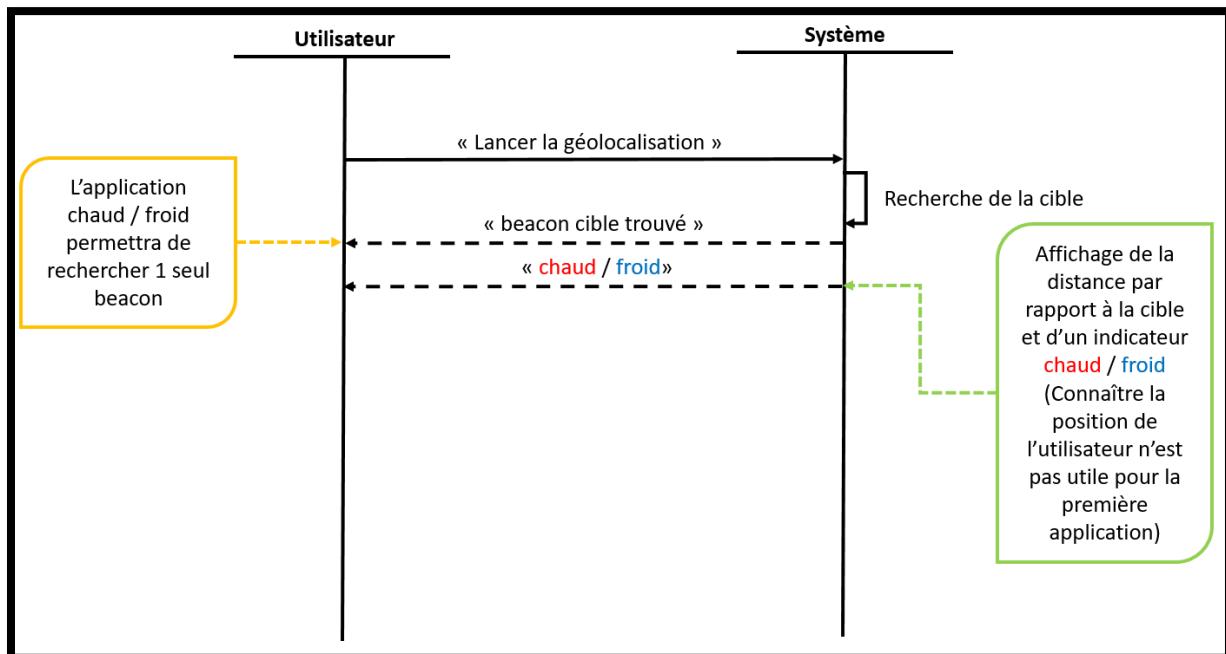


Figure 66 : Diagramme de séquence chaud-froid

Le diagramme de séquence ci-dessus représente les interactions entre l'utilisateur et le système dans le cadre du jeu chaud-froid.

Remarque : *l'acquittement du système « Beacon cible trouvé » n'a pas été mis en place car nous n'avons pas jugé cela utile pour le fonctionnement.*

4.7.2 Conception

4.7.2.1 Interface téléphone / Beacon

Pour réaliser l'interface entre le téléphone et le Beacon recherché, la communication se fait en Bluetooth à travers l'API **AltBeacon** permettant de lire les paquets BLE diffusés par les Beacons.



AltBeacon

Figure 67 : AltBeacon

AltBeacon permet de récupérer les données provenant de plusieurs Beacons à une **fréquence** d'environ **1Hz**. Cette librairie permet également de récupérer des notifications. Cela nous a permis de mettre en place une **notification** lorsqu'un Beacon s'approche dans le champ de scan du téléphone.

4.7.2.2 Conversion de la puissance du signal en distance

Une fois le RSSI reçu, le téléphone récupère la valeur pour le traitement. Le RSSI est changé en distance réelle grâce à un calcul trouvé dans les anciens rapports de ce projet :

$$distance = \begin{cases} \frac{RSSI}{txPower}^{10}, & RSSI < txPower \\ 0.89976 \frac{RSSI}{txPower}^{7.7095} + 0.111, & RSSI \geq txPower \end{cases}$$

Figure 68 : Calcul de la distance en fonction du RSSI

En entrée, le RSSI et en sorti, la distance en mètres. Le RSSI de référence est de **55dBm**, valeur configurée sur les Beacons.

Remarque : cette valeur a été configurée pour le téléphone Asus Zenfone Go

4.7.2.1 Conception de l'interface graphique

En ce qui concerne l'aspect graphique, nous avons fait le choix de représenter le chaud-froid sous forme d'un **thermomètre** dont la valeur varie en fonction de la distance.



Figure 69 : Croquis chaud-froid

Comme énoncé précédemment, le mode chaud-froid est le premier applicatif que nous avons développé dans le but de prendre en mains les Beacons et de répondre aux exigences client.

Le mode de détection du chaud-froid est axé sur la détection d'un seul Beacon. Plus la puissance du signal reçue par l'application est forte, plus l'utilisateur est vu comme près du Beacon et inversement.

Du plus éloigné au plus proche, les différents paliers de distances sont les suivants :

- ✓ Il fait -8000°C
- ✓ Il neige !
- ✓ Tu es gelé
- ✓ Il fait froid !
- ✓ Tiède
- ✓ Tu y es presque
- ✓ Il est où ??
- ✓ Tu le vois
- ✓ Allumez ! Le feu !

Cet applicatif n'a pas pour but la localisation du Beacons, mais l'approximation de sa position dans le référentiel de recherche. Il permet à l'utilisateur de réduire la zone de recherche, mais ne donne en aucun cas une position précise du Beacons recherché.

4.7.3 Développement

4.7.3.1 Prise en main de l'API AltBeacon

De façon à mettre en place la communication, plusieurs étapes sont à suivre. Premièrement, il faut donner des informations à l'application permettant de savoir quels types de Beacons il faut rechercher. Cela se définit par un layout prédéfini pour capter les Beacons iBeacon.

```
beaconManager.getBeaconParsers().add(new BeaconParser().
    setBeaconLayout("m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24,d:25-25"));
```

Figure 70 : Prise en main de l'API AltBeacon

4.7.3.2 Implémentation du calcul de distance

Voici ci-dessous, l'extrait du programme permettant de convertir la puissance du signal en distance exploitable :

```
RSSI=(short) ((result[1] << 8) | (result[0] & 0xFF));
ratio = RSSI*1.0f/RSSI_Init;
if (ratio < 1.0f) {
    return distance = (float) Math.pow(ratio,10);
}
else {
    return distance = (float) ((0.89976)*Math.pow(ratio,7.7095) + 0.111);
}
```

Figure 71 : Calcul de la distance

4.7.4 Développement de l'interface graphique

4.7.4.1 Calibration

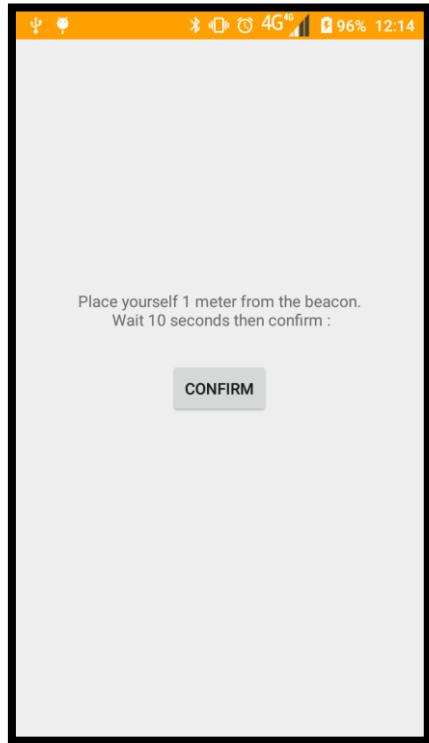


Figure 72 : IHM Calibration

Cette interface est sûrement la plus minimaliste de notre application pourtant elle possède un rôle important au bon traitement de la suite. Le **layout** possède un **TextView** indiquant à l'utilisateur de se placer à un mètre des Beacons et d'attendre un certain temps avant d'appuyer sur un bouton « confirmer ». Cela permet de récupérer le RSSI à 1 mètre nécessaire au calcul de la distance (RSSI_Init).

A. Guidage

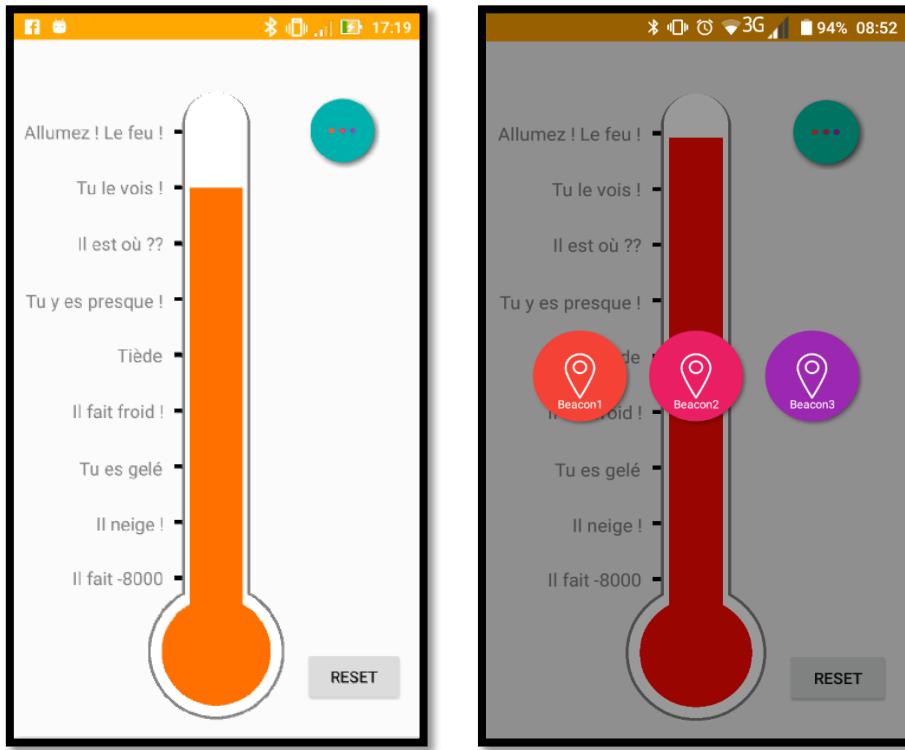


Figure 73 : IHM Chaud-Froid

Après une calibration, nous pouvons afficher ce thermomètre composé d'une couleur qui change automatiquement en fonction de la distance qui sépare le téléphone du Beacon recherché. Cette couleur varie entre le bleu et le rouge. Cette variation est réalisée grâce à un passage de la couleur en **format HSV** (« Hue/Saturation/Value » en anglais, ou en français « Teinte Saturation Valeur »). Elle est appliquée à différentes formes géométriques. Effectivement le thermomètre est composé principalement de plusieurs cercles et de rectangles, dont un prévu pour augmenter sa largeur en fonction de la distance.

Un bouton « reset » est présent pour effectuer la même manipulation que la calibration seulement, cela est transparent pour l'utilisateur.

Un **Boom Menu** (visible avec le cercle de couleur vert turquoise) a également été mis en place. Il permet de sélectionner le Beacon que l'on souhaite rechercher. Par défaut le Beacon 1 est recherché. Une animation accompagne ce menu : elle affiche les boutons d'une manière à faire croire à l'utilisateur que les petites boules de couleurs présentes, surgissent du cercle représentant le Boom Menu. A chaque scan du Bluetooth, nous changeons les Beacons disponibles à la sélection.

B. Conclusion

Suite au développement de cette application, des remarques ont pu être effectuées. Nous avons pu remarquer que la distance calculée n'était **pas assez précise**. En effet, des fluctuations importantes ont été observées. Ces résultats nous ont permis de confirmer nos conclusions sur les limites des Beacons Gimbal. Par conséquent, nous avons décidé d'initier une étude pour effectuer le choix de nouveaux Beacons plus performants, d'où l'achat des Beacons BNB. De même, ce sont ces résultats qui nous ont menés à effectuer un lissage des données.

4.8 Conception et développement de l'application « Localisation »

4.8.1 Analyse

4.8.1.1 Analyse fonctionnelle

En ce qui concerne l'aspect fonctionnel de l'application localisation, voici comment elle se décompose :



Figure 74 : Fonction de calcul de la position

Le schéma ci-dessus décrit les données d'entrées et la donnée de sortie de la fonction de trilateration. Cela répond en partie aux trois dernières problématiques du cahier des charges.

4.8.1.2 Analyse logicielle

A. Diagramme des cas d'utilisation

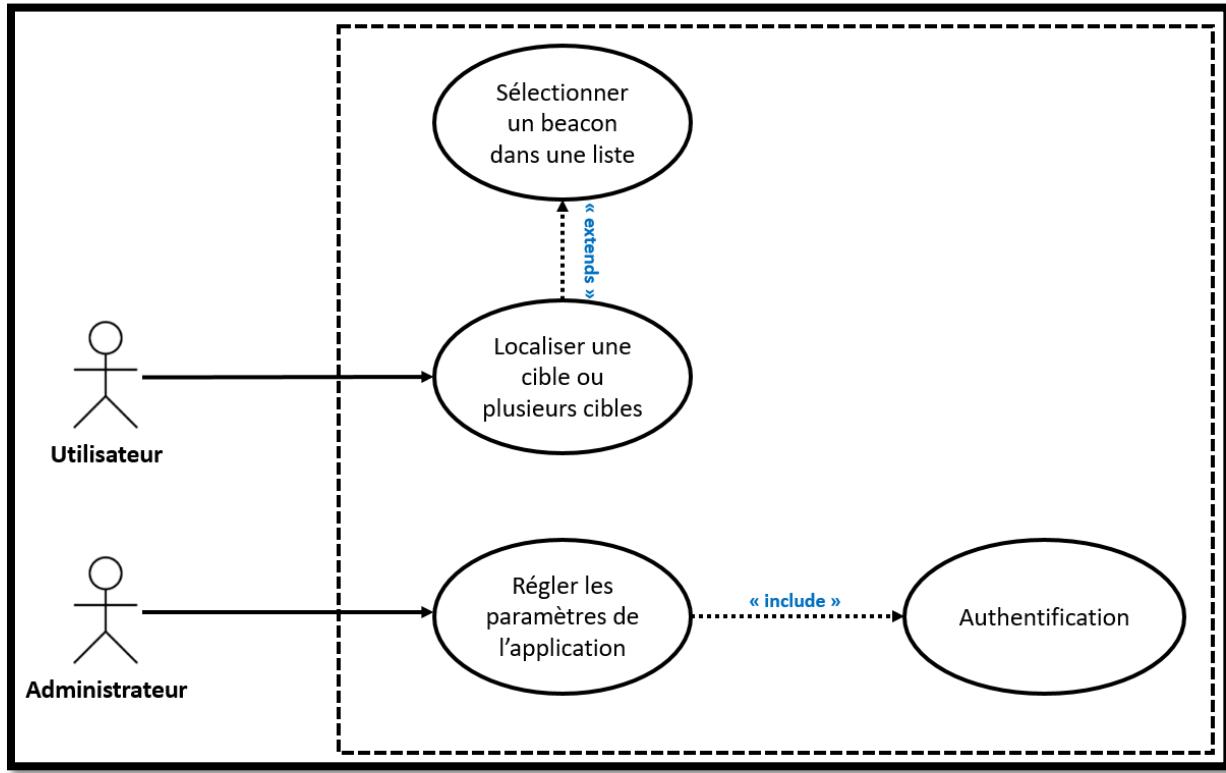


Figure 75 : Diagramme des cas d'utilisation

Le **diagramme des cas d'utilisation** ci-dessus représente toutes les **fonctionnalités** auxquelles doit répondre l'application. D'une part, l'application de géolocalisation doit permettre de localiser une ou plusieurs cibles sur un plan. Par conséquent, cela implique que l'application doit proposer un menu pour sélectionner des cibles. D'autre part, l'application doit permettre de régler les paramètres de l'application. Cela ne peut se faire qu'avec une authentification.

B. Diagrammes de séquence

a. Cas général

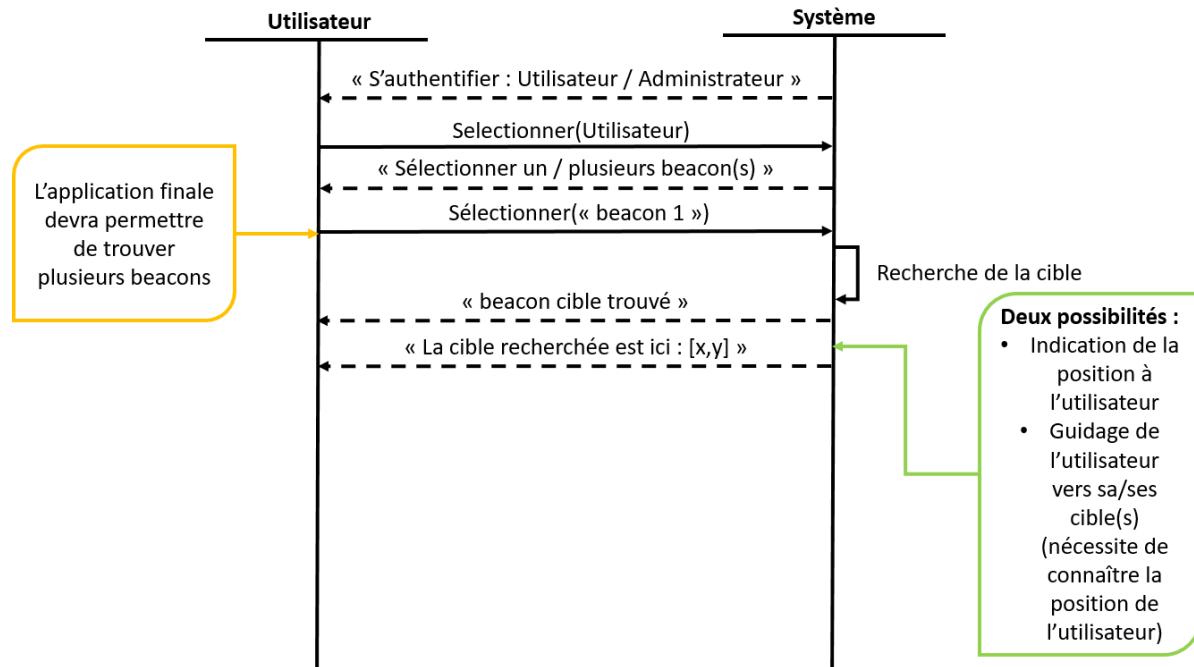


Figure 76 : Diagramme de séquence « localiser un beacon »

b. Cas où il y a une erreur

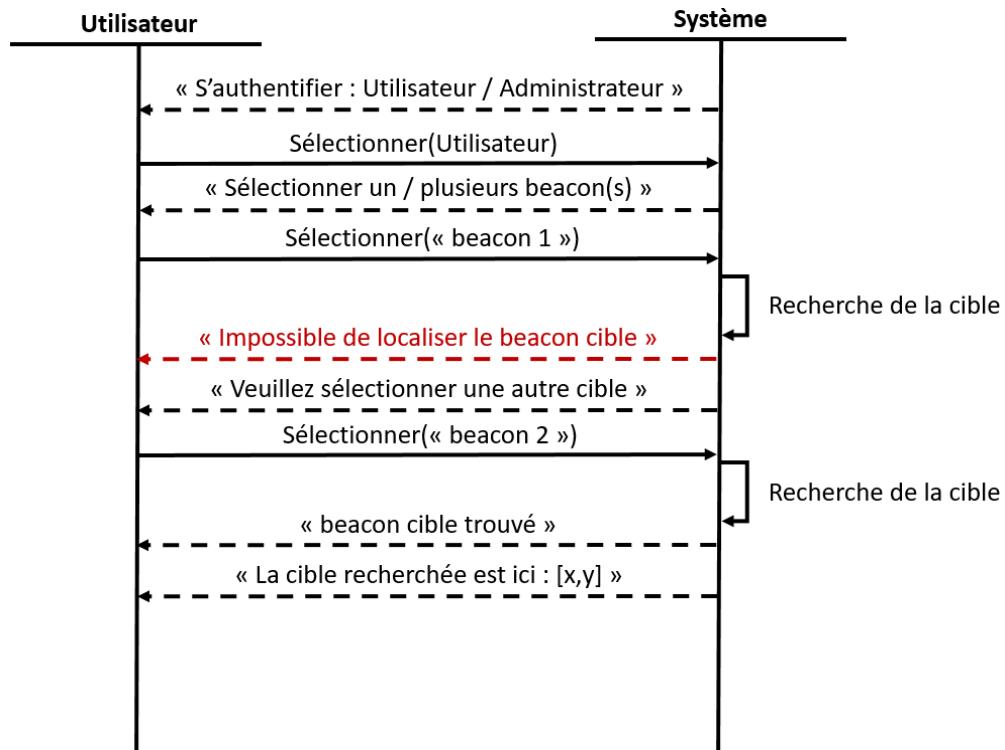


Figure 77 : Diagramme de séquence « localiser un beacon avec erreur »

c. Recherche de plusieurs Beacons (avec erreur)

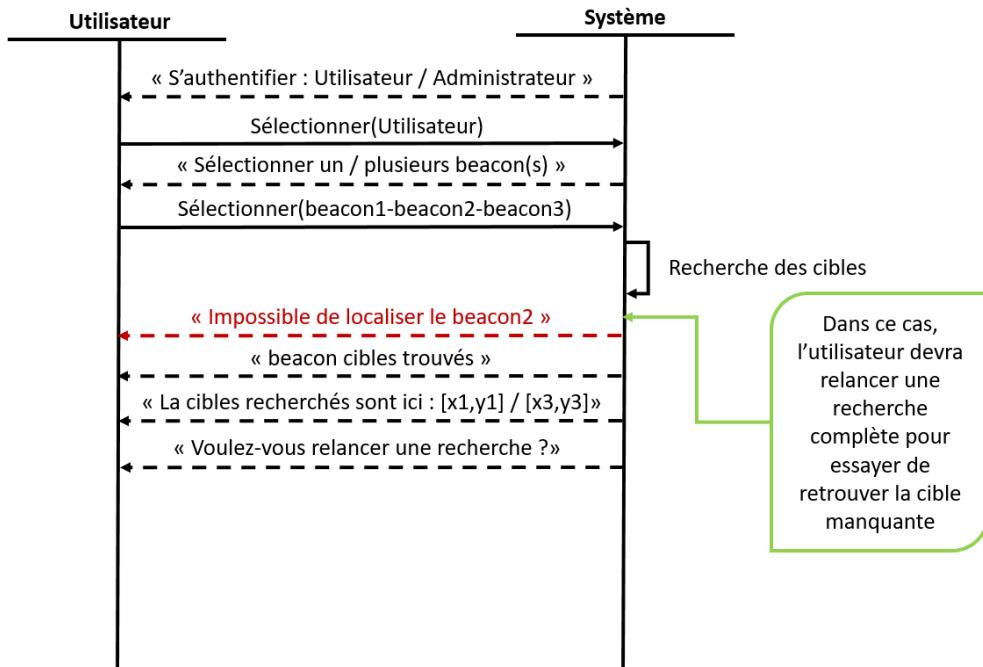


Figure 78 : Diagramme de séquence « localiser plusieurs beacons »

d. Administration du système (ajout d'une balise)

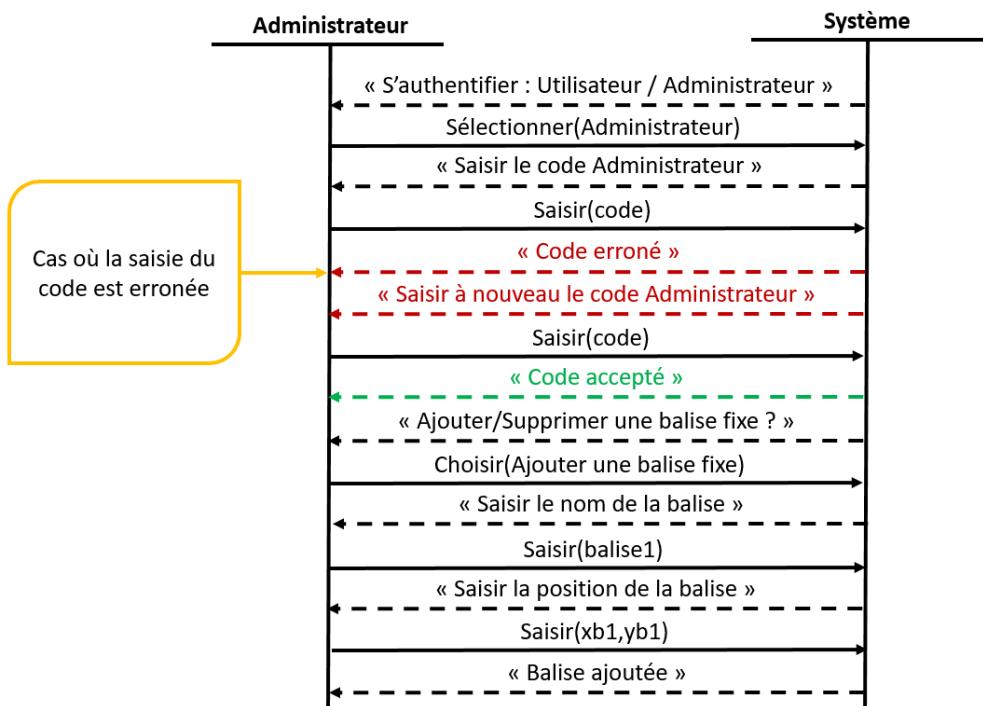


Figure 79 : Diagramme de séquence « administration du système »

e. Administration du système (suppression d'une balise)

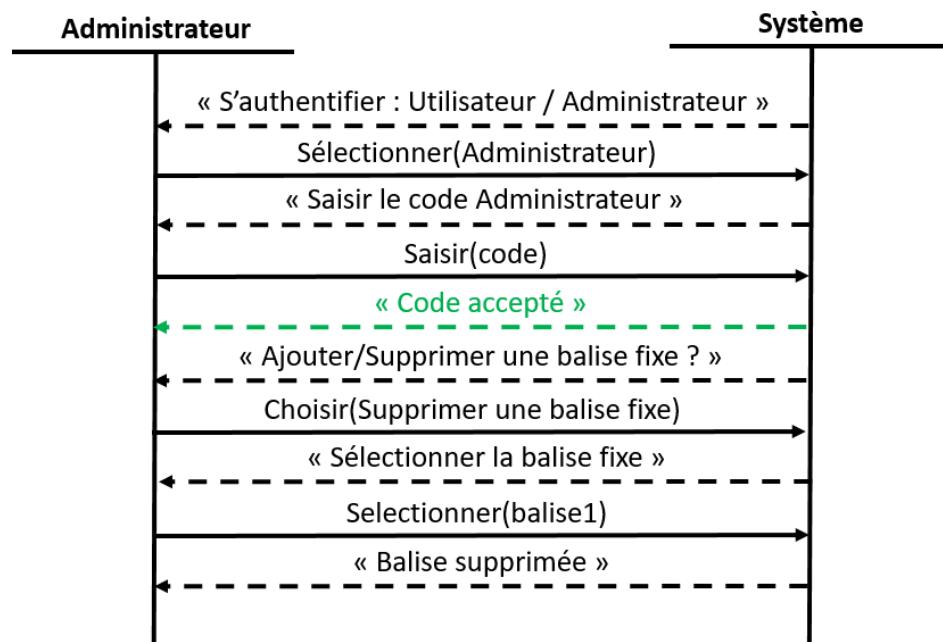


Figure 80 : Diagramme de séquence « suppression d'une balise »

f. Administration du système (reconfiguration d'une balise)

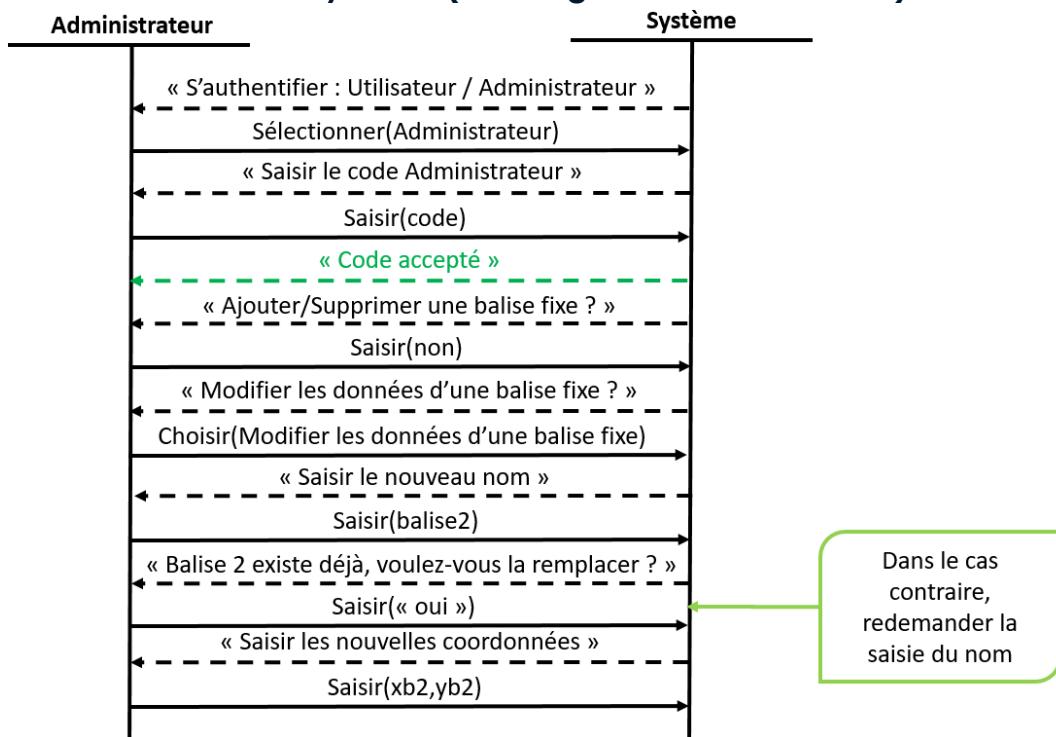


Figure 81 : Diagramme de séquence « reconfiguration d'une balise »

L'ensemble des échanges ci-dessus doit avoir lieu entre l'utilisateur et le système. Les cas d'utilisation qui ont été décrits sont **l'authentification**, **l'administration** et **la recherche d'un ou de plusieurs Beacons** avec réussite où échec. Ces diagrammes ont été conçus pour organiser le développement. Certaines de ces interactions ne seront pas développées.

C. Diagrammes des classes

Enfin, pour clôturer l'analyse, **un diagramme des classes** a été réalisé provisoirement (*ébauche). Cela permet de spécifier le programme qui va être développé. De même, cette méthode permet d'avoir un programme structuré et bien analysé.

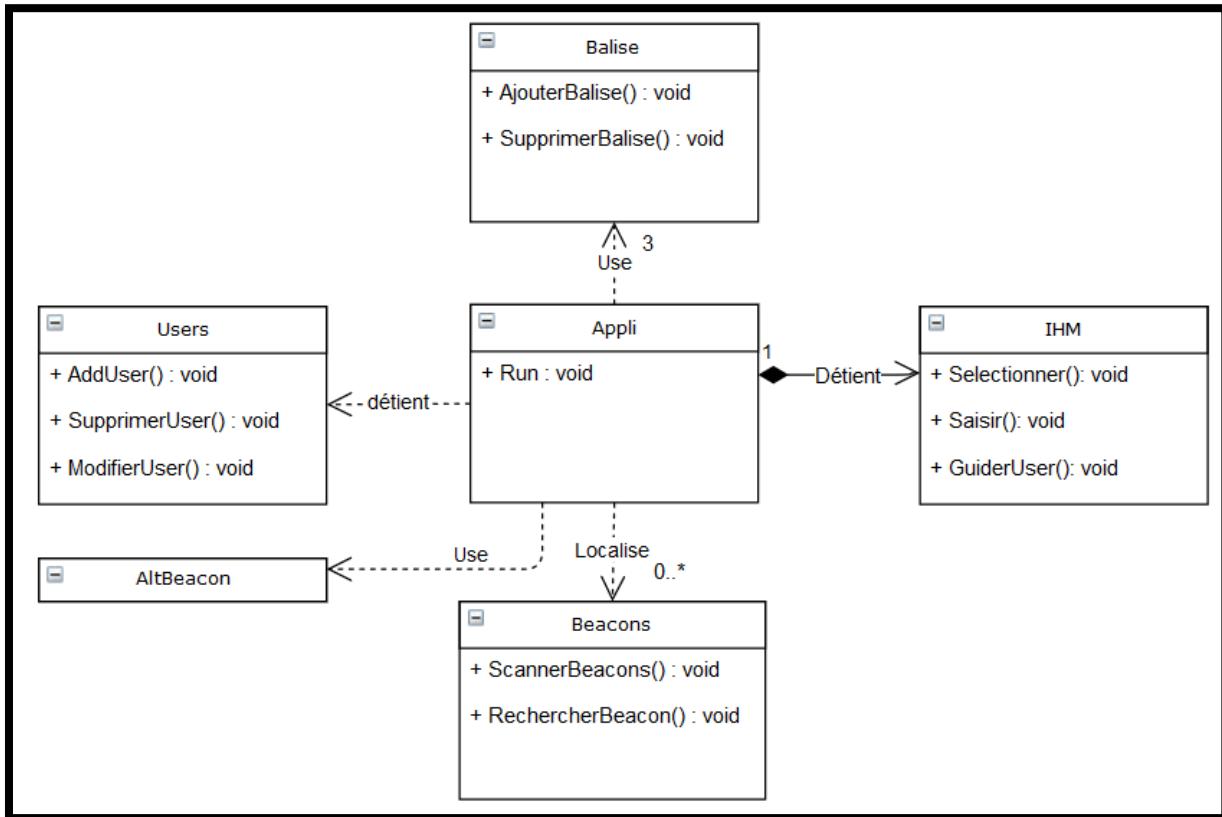


Figure 82 : Diagramme des classes

4.8.2 Conception

4.8.2.1 Interface téléphone / balises fixes

A. Principes fondamentaux du Bluetooth Low Energy

Le Bluetooth Low Energy a été créé par Nokia en 2006 dans le but de compléter le Bluetooth. Cependant, celui-ci ne peut remplacer les performances de ce dernier.

Le BLE devient un outil spécifique, surtout pour l'embarqué dans le domaine du sport, de l'automobile et des appareils de surveillances (médicale et sécurité). Le BLE est surtout utilisé dans la technologie Beacon et plus généralement dans la localisation (dans des configurations pouvant s'affranchir du GPS).

Comparé au Bluetooth bien connu, le BLE dispose d'une bande passante limitée et d'une distance moindre. Cependant, il permet une **consommation 10 fois inférieure** et une **transmission des données mieux organisée**.

a. Réseau

Le BLE dispose d'une philosophie qui peut sembler différente des autres protocoles de communication. Pour cela, nous allons voir dans cette partie les différents besoins ainsi que le fonctionnement du Bluetooth Low Energy.

Des appareils supportant le BLE peuvent avoir deux rôles distincts. Ces rôles sont « **Centrale** » ou « **Périphérique** ».

Les centrales sont généralement des téléphones portables qui ont besoin de communiquer avec ce qu'on appelle les périphériques. C'est le « master » du réseau car il dispose de meilleures performances hardwares.

Les périphériques sont donc souvent des capteurs ou des Beacons qui émettent des trames afin d'être traité par la centrale.

b. Données

Les données envoyées par le protocole BLE peuvent être des « **Advertisements** » ou des « **Answers** ». Les « answers » permettent aux périphériques (et parfois à la centrale) d'acquitter de leur présence sur le réseau. Ils indiquent que le périphérique est toujours vivant. Cependant, un périphérique qui reçoit des données peut demander une réponse.

c. Types d'envoi

En BLE il existe les envois en **Broadcast** et en **Connexions**.

Le broadcast permet d'envoyer des données à différents périphériques ou centrales. Lorsque l'appareil supportant le BLE envoie en broadcast, il envoie des paquets périodiquement aux différents « Observateurs » (centrales ou périphériques) qui peuvent ou non choisir de les lire. Cette façon est la seule possible en BLE pour envoyer des données à plusieurs appareils BLE. C'est sur ce principe que reposent les Beacons lorsqu'ils émettent vers l'extérieur.

Voici un schéma donné pour imaginer le broadcast (MikroElektronika) :

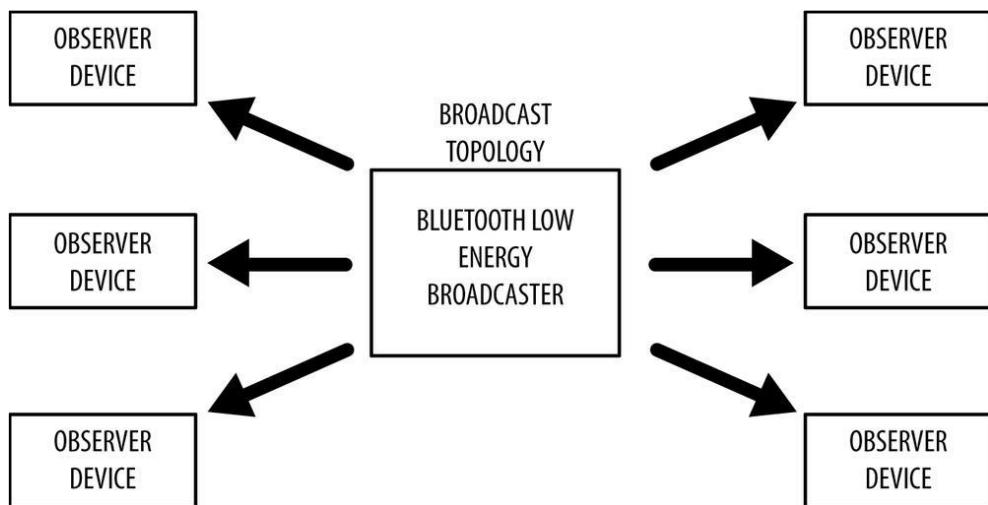


Figure 83 : Envoi en broadcast

Les connexions, dans le cadre du projet, semblent ne pas être intéressantes mais peuvent simplement être abordées :

Une connexion permet l'échange de données entre deux appareils BLE. Lorsque la centrale détecte un objet connectable, elle lui envoie une « advertisement » et initie la connexion. Lorsque cette connexion est établie, c'est la centrale qui gère entièrement l'échange (horloge, demande, réception...). Il est possible de définir un « Événement de connexion » qui permet de lancer des connexions à certains moments définis (toutes les 10 secondes par exemple) et ainsi, de consommer moins.

Voici un schéma donné pour imager les connexions (MikroElektronika) :

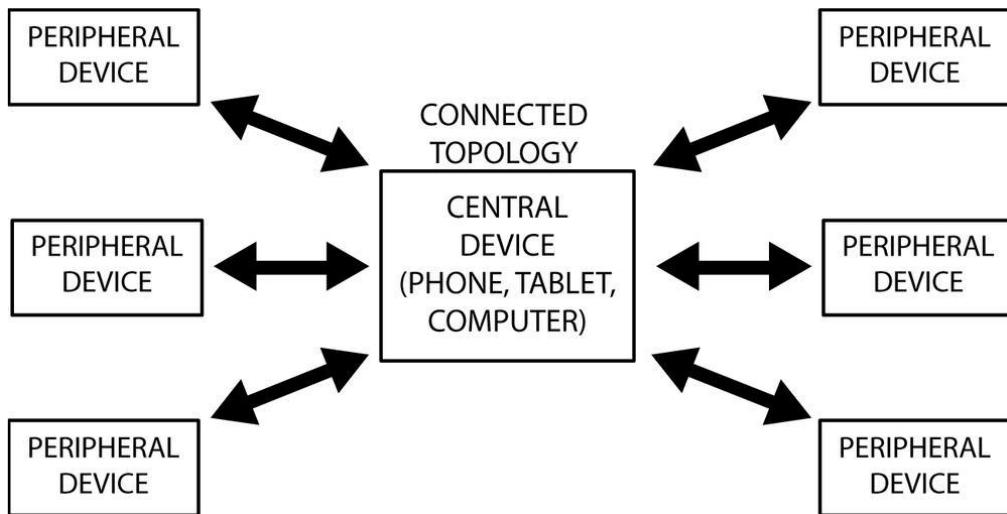


Figure 84 : Connexion à un périphérique (échanges de caractéristiques)

Remarque : il n'existe cependant aucun moyen de faire de la connexion et du broadcast en même temps.

d. Les couches BLE

Il existe trois principales couches dans le protocole BLE :

- **L'Application**, qui est la couche la plus haute et qui porte bien son nom, est la couche de gestion Front end.
- **L'Host** qui contient les sous-couches suivantes :
 - **Le GAP (Generic Access Profile)**
C'est la plus haute couche, elle permet de gérer les connexions et les « advertisements ».
 - **Le GATT (Generic Attribute Profile)**
Cette couche définit comment les données sont organisées et comment elles sont envoyées entre les différentes applications. Le **GATT** est organisé en **Profils** qui contiennent des « **Services** » qui contiennent tous des **Caractéristiques** qui sont-elles même des données de l'utilisateur. Les services se distinguent par un **UUID** unique sur 16 bits. Les Caractéristiques ont aussi un **UUID** qui a pour but de définir la fin d'une donnée. C'est la couche que l'on voit lorsque l'on développe sous Android avec l'utilisation du BLE.
 - **Le L2CAP (Logical Link control and Adaptation Protocol)**
Cette couche permet de gérer les protocoles des couches supérieures et les transformer en trames BLE. Cette couche s'occupe généralement des protocoles de sécurité et de transmission de l'information. Elle permet entre autres de refragmenter les paquets reçus en plusieurs fois.
 - **Le ATT (Attribute Protocol)**
 - **Le SM**
 - **Le HCI (de l'host)**

➤ **Le Controller** qui contient les sous-couches suivantes :

- **Le HCI (Host Controller Interface)**

C'est cette couche qui permet de mettre à disposition de l'appareil différentes configurations afin d'améliorer la communication avec le CPU.

- **Le LL (Link Layer)**

C'est la partie qui permet de relier le physique au logiciel. Il définit quel rôle l'appareil doit jouer dans le réseau BLE. Il s'occupe de savoir si l'appareil est un Périphérique ou une Centrale, un Master ou un Esclave...

- **Le PHY (Physical Loayer)**

Ou « Physical Layer » est la partie hardware très bas niveau. C'est ici que sont définis les canaux de communication (40 max pour le BLE entre 2.4GHz à 2.4835GHz)

B. Architecture et échanges

La stratégie actuelle est basée sur des balises fixes. Pour bien comprendre son fonctionnement nous avons utilisé le schéma suivant :

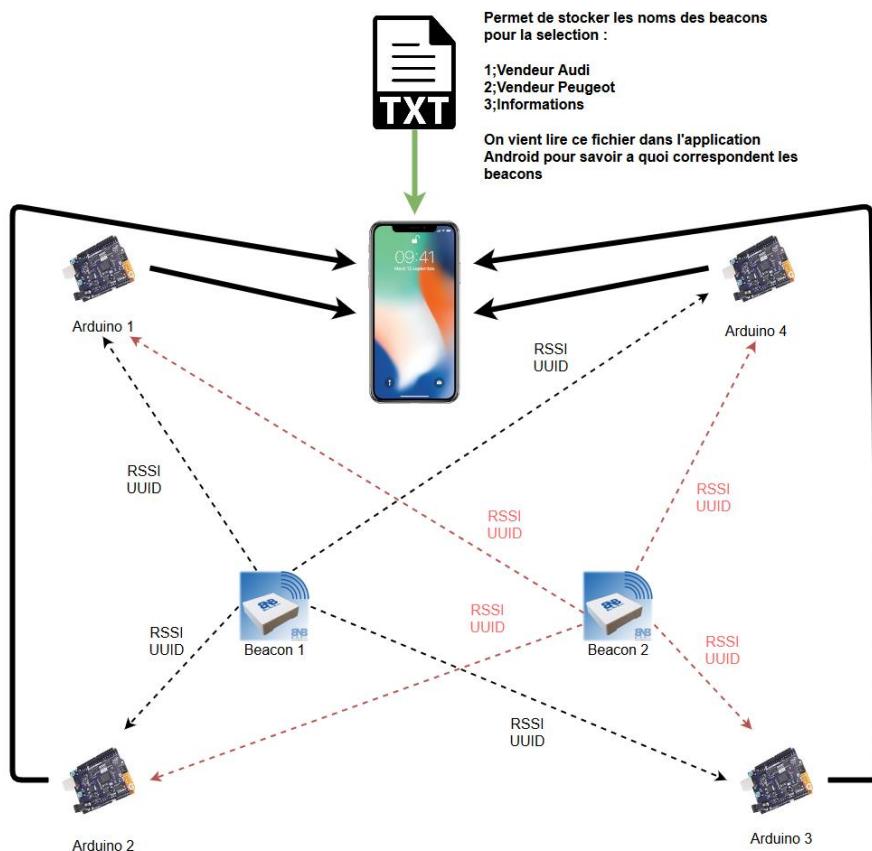


Figure 85 : Architecture mise en œuvre

Les balises fixes sont représentées par des arduinos (et pour le POC, sont des arduinos). Chaque arduinos reçoit le **RSSI** des Beacons (atténuation en dBm) en **broadcast** grâce à un signal BLE. Une fois ce signal reçu, les arduinos renvoient un service « **ScanBeaconArduinoX** » où X représente la balise fixe (1 ;2 ;3 ;4...). Ce même service contient une caractéristique comportant le numéro de l'arduino, le numéro du Beacon et le RSSI du Beacon. Pour ce faire, nous avons imaginé une structure de données simple décrite ci-dessous.

Remarque : il a été fait le choix d'utiliser quatre balises fixes car plus on utilise de balises, plus le résultat est précis. Le fait d'avoir choisi quatre balises permet de les placer aux quatre coins du périmètre.

C. Protocole Applicatif

[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31]

<0:7> : Numéro de l'arduino (jusqu'à 127 arduino)
 <8:15> : Numéro du beacon (jusqu'à 127 beacons, peut changer)
 <16:31> : RSSI du beacon

Figure 86 : Structure de données

Cette structure de donnée permet d'envoyer l'ensemble des informations en un seul et même nombre de **32 bits** qui peut être intégré dans une caractéristique BLE.

Nous pouvons donc avoir au maximum :

- 127 balises fixes
- 127 Beacons
- Un RSSI de 2^{15} dBm

Ces nombres sont arbitraires et peuvent être facilement changés en prenant un peu plus d'espace en réduisant la taille du RSSI.

L'objectif est donc d'intégrer cette structure de donnée dans la caractéristique qui est envoyée par le service émis par l'arduino (« **ScanBeaconArduinoX** »). Ainsi chaque arduino va envoyer dans cette variable ce qu'elle aura elle-même scanné.

D. Lissage des données

Après la réception des informations (cf paragraphe BLE), nous les incluons dans un tableau afin de lisser les valeurs.

Ce tableau correspond au schéma ci-dessous :

TableauLissage =	Beacon 1	Beacon 2	...	Beacon N	Arduino 1
	Beacon 1	Beacon 2	...	Beacon N	Arduino 2
	Beacon 1	Beacon 2	...	Beacon N	Arduino 3
	Beacon 1	Beacon 2	...	Beacon N	Arduino 4
	...				
	Beacon 1	Beacon 2	...	Beacon N	Arduino N

↓

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Moyenne
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	---------

Figure 87 : Tableau de filtrage

Pour chaque arduino, nous stockons 21 valeurs pour chaque Beacon. Dans ce tableau de 21 valeurs, il y a 20 valeurs pour les données de distances récupérées et 1 valeur pour la moyenne de ces valeurs.

Lorsqu'une valeur est récupérée, nous connaissons l'arduino et le Beacon concernés grâce à la structure de donnée qui sont passée en paramètre.

Après avoir récupéré la **moyenne**, nous vérifions que la différence entre la valeur récupérée et la moyenne est inférieure ou égale à 5 ou supérieur ou égale à -5. Cela permet d'exclure les valeurs absurdes que nous pourrions avoir. Enfin, nous recalculons la moyenne des valeurs.

4.8.2.2 Conception de l'interface graphique

Dans cette partie, nous expliquerons les choix que nous avons dû faire lors de la conception de l'IHM pilotant notre application. Lors de sa conception, nous avons pris en compte les différentes fonctionnalités exigées par le cahier des charges.

A. Application orientée jeu

Il a été décidé d'imaginer un soft axé de la même manière qu'un jeu avec un **choix du mode de jeu**. Il nous a été exigé de pouvoir intégrer le mode chaud-froid dans l'application finale. Ce choix de mode de jeu doit également permettre de séparer les deux fonctionnalités suivantes :

- ✓ Recherche d'un seul Beacon
- ✓ Recherche de plusieurs Beacons

De plus lors de nos réunions, P.Martineau nous a orientés vers une application ludique, ce qui nous a confortés dans ce choix.

B. Étude de l'IHM

En amont du développement, il a été décidé de concevoir graphiquement l'application pour faciliter la phase de développement. Pour se faire nous avons utilisé le logiciel de traitement graphique : **Visio**, intégré à la bibliothèque logicielle Microsoft Office.



Figure 88 : Visio

a. Ecran d'accueil



Figure 89 : IHM Admin

La première page affichée à l'écran. Depuis ce menu principal, l'utilisateur a le choix entre le mode administrateur ou alors jouer une partie.

b. Écran d'administration



Figure 89 : IHM Login

Une fois l'utilisateur identifié, le **menu administrateur** permet d'accéder à certains réglages de l'application :

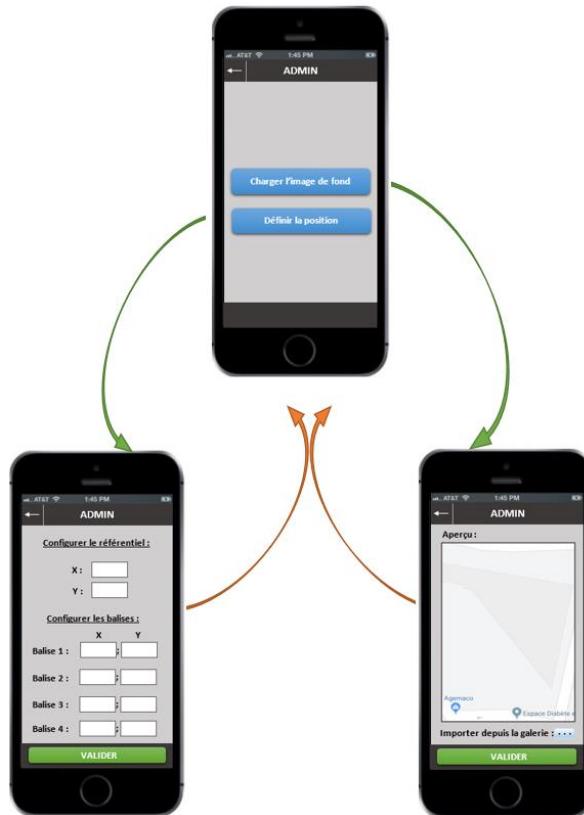


Figure 90 : IHM Navigation

L'idée que nous souhaitions mettre en avant en intégrant une fonctionnalité d'import de la carte est la **flexibilité** de l'application. Cette fonctionnalité permet à l'utilisateur d'utiliser « Indicon join² » n'importe où (à condition qu'il soit équipé des balises fixes). Il suffit de charger la carte du lieu choisi, positionner les balises et de référencer les coordonnées sur la vue « Acquisition des coordonnées ».

Une fois les balises placées, il faut **référencer les coordonnées** dans l'application. **Configurer le référentiel** permet de calibrer la carte en adaptant sa taille et également de définir le référentiel des coordonnées. Une fois que les positions des balises fixes sont entrées, l'utilisateur doit valider la saisie des paramètres. Il peut ensuite commencer à jouer.

c. Mode de jeu



Figure 91 : IHM Game Menu

Lorsque l'utilisateur sélectionne « Jouer » dans le menu principal il accède directement aux 3 modes de jeux que nous avons développés lors du projet :

- ✓ Chaud / froid
- ✓ Rechercher un seul Beacon
- ✓ Rechercher plusieurs Beacons

² Indicon Join : nom de l'application

d. Trouver un seul Beacon



Figure 92 : IHM Sélectionner un beacon

Ce mode de jeu permet à l'utilisateur de localiser précisément la position du Beacon recherché à l'aide des balises fixes. Grâce à l'import de la carte, l'utilisateur peut se repérer plus facilement dans son environnement de recherche. (Voir ci-dessus)

La détection mise en œuvre par ce mode de jeu (trilateration) permet la détection d'un seul Beacon. Dans un premier temps l'utilisateur doit cliquer sur le bouton « Choisir une cible » et sélectionner le Beacon qu'il souhaite rechercher. Une fois sélectionné et validé, l'utilisateur voit apparaître l'emplacement de celui-ci sur la carte.

e. Trouver tous les Beacons



Figure 93 : IHM Sélectionner plusieurs beacons

Cette fonctionnalité est similaire à la précédente. Elle aussi fonctionne avec l'utilisation des balises fixes et l'import de la carte. Et la trilateration permet une détection optimale des Beacons.

Dans ce mode de jeu, toutes les cibles détectées sont affichées sur la carte. L'utilisateur peut choisir une cible prioritaire en cliquant sur « choisir une cible ». Une fois la cible sélectionnée et validée, elle s'affiche en surbrillance pour permettre à l'utilisateur de la différencier des autres présentes sur la carte. (Voir figures ci-dessus)

Remarque : la dernière fonctionnalité explicitée est totalement optionnelle. Il faut retenir que l'application est capable d'afficher la position d'une ou de toutes les balises trouvées si besoin.

4.8.3 Développement

4.8.3.1 Mise en service des balises fixes

Le développement de l'application finale a débuté par le développement des Arduino. Comme expliqué précédemment, c'est le modèle **GATT** qui sera mis en œuvre. Il est possible d'illustrer ce principe par le schéma ci-dessous : le périphérique envoie des services en broadcast à plusieurs utilisateurs qui peuvent venir lire les caractéristiques en se connectant au périphérique.

Remarque : l'application développée est mono-utilisateur.

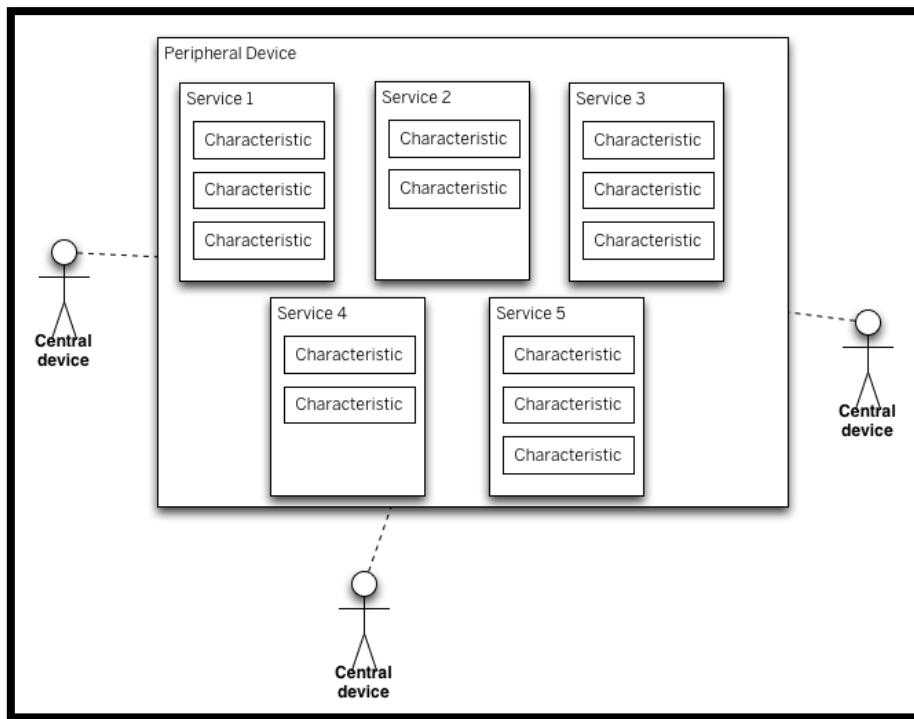


Figure 94 : Principe de fonctionnement du BLE (1)

Pour mettre en œuvre les balises fixes, il faut donc respecter le principe suivant :

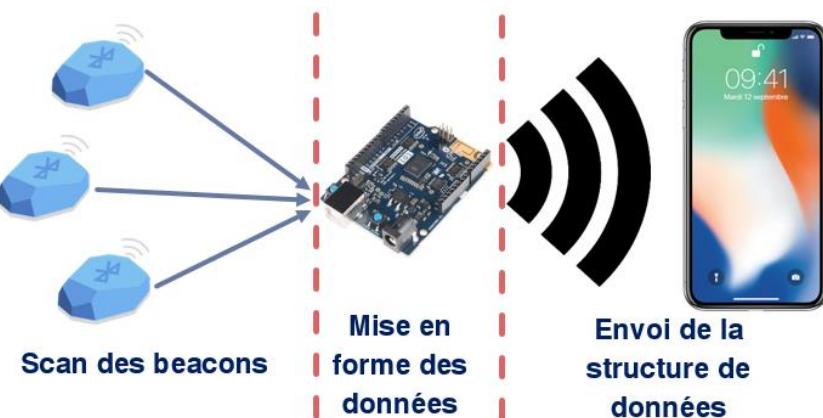


Figure 95 : Principe de fonctionnement du BLE (2)

A. Scan des Beacons

L'objectif est de scanner les Beacons à proximité pour récupérer les RSSI correspondants. La première étape est de faire en sorte d'avoir une interruption lorsqu'une centrale est découverte. Il faut ensuite lancer le scan. Ces deux opérations sont effectuées ci-dessous.

```
// When device is discovered handle centralDiscovered() is called.
BLE.setEventHandler(BLEDiscovered, centralDiscovered);
//Log to change
Serial.println("Arduino 1 starts scanning");
BLE.scan(true);
```

Figure 96 : Extrait de code arduino (1)

Remarque : il faut bien préciser « true » comme paramètre de la fonction scan. Cela permet de générer un événement à chaque fois que l'arduino voit une même centrale. Dans le cas contraire, elle ne le signalera que lors de la première détection.

```
void centralDiscovered(BLEDevice peripheral)
{
    //If the arduino detect a new peripheral
    if (peripheral) {
        if(peripheral.hasLocalName()){
            // print the manufacturer data
            if (peripheral.hasManufacturerData()) {
                unsigned char manu_data[255];
                unsigned char manu_data_length;

                bool success = peripheral.getManufacturerData (manu_data, manu_data_length);

                if (success){
                    int i;
                    char* buf_str = (char*) malloc (2*manu_data_length + 1);
                    char* buf_ptr = buf_str;
                    char* uuidBeacon = (char*) malloc (2*manu_data_length + 1);

                    bzero(uuidBeacon,2*manu_data_length + 1);

                    for (i = 0; i < manu_data_length; i++){
                        buf_ptr += sprintf(buf_ptr, "%02X", (unsigned char)manu_data[i]);
                    }
                    *(buf_ptr + 1) = '\0';

                    //Extracting of the UUID from the manufacturer data
                    extractFromString(uuidBeacon,buf_str);

                    if(buf_str[0]=='4' && buf_str[1]=='C'){
                        // print the local name, if present
                        Serial.print("-----START SCAN-----\n");
                        //Print the UUID
                        Serial.print("UUID : ");
                        Serial.println(uuidBeacon);
                        Serial.print("\n");
                        NumBeacon=peripheral.localName()[6]-48;
                        Serial.print("Beacon Number : ");
                        Serial.println(NumBeacon);
                        Serial.print("\n");

                        // print the RSSI
                        Serial.print("RSSI : ");
                        Serial.println(peripheral.rssi());
                        Serial.print("\n");
                        Serial.print("Characteristic : ");
                        Serial.print("\n");
                        data=(NumArduino<<24)+(NumBeacon<<16)+((short)(peripheral.rssi())*(-1));
                        Serial.print(data);
                        dataCharacteristic.setValue(data);
                        Serial.print("\n");
                        Serial.print("\n");
                        Serial.print("-----END SCAN-----\n");
                    }
                    free(buf_str);
                    free(uuidBeacon);
                }
            }
        }
    }
}
```

Figure 97 : Extrait de code arduino (2)

La fonction ci-dessus est appelée à chaque fois que l'arduino voit une **centrale**. Grâce à l'objet de type `BLEDevice`, on a accès au nom du périphérique et à son RSSI. De même, il est possible de récupérer ce que l'on appelle le « **manufacturer data** ». Ce dernier contient des données dont l'identifiant constructeur et l'UUID. Il a été choisi de ne garder que les trames iBeacon, à savoir ceux dont le manufacturer data commence par les caractères « 4C » (Identifiant Apple).

```
Discovered a peripheral
-----
Address: 40:97:6F:B8:09:52
Manufacturer Data: 4C0002151EFC34D317F74F6595DD78875AA696B6007B01C8BF
RSSI: -81
```

Figure 98 : Résultat du scan Arduino

Cela mène à se poser la question du format des trames iBeacon qui transitent. On peut observer sur la figure ci-dessous que l'identifiant du constructeur est toujours fourni dans la partie appelée **prefix** de la trame. Cela explique également le contenu du manufacturer data (« 4C0215+UUID »).

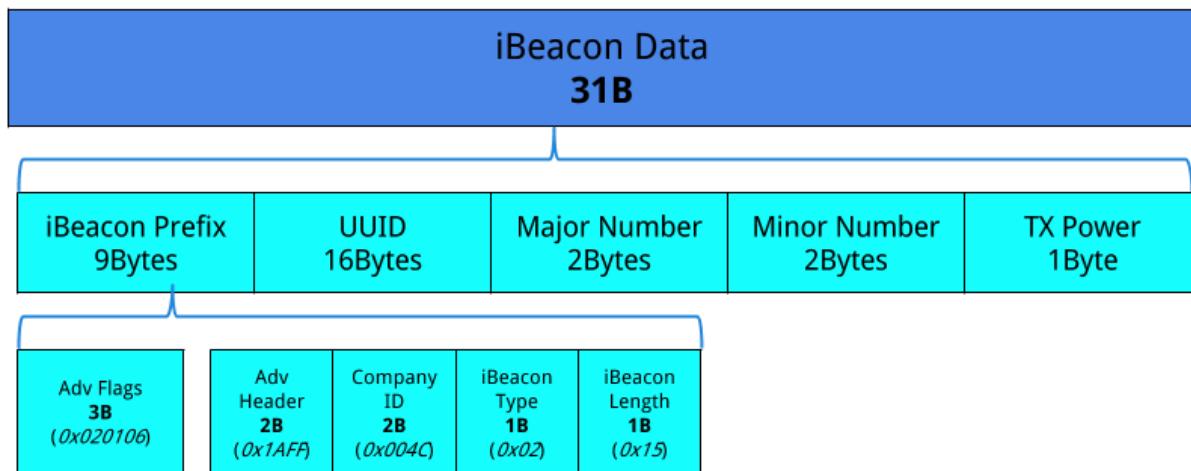


Figure 99 : Format de trame iBeacon

Une fois les données récupérées, il suffit de configurer la balise pour qu'elle génère un service à son tour, contenant la caractéristique qui change de valeur. Pour cela, nous déclarons le service et la/les caractéristique(s) que l'on souhaite diffuser (avec un identifiant).

```
BLEService beaconService ("19B10010-E8F2-537E-4F6C-D104768A1214"); // create service
BLEIntCharacteristic dataCharacteristic("19B10012-E8F2-537E-4F6C-D104768A1214", BLERead | BLENotify);
```

Figure 100 : Déclaration du service et de la caractéristique

Une fois ces opérations réalisées, il faut ajouter le service à l'objet `BLEDevice` courant et lancer l'émission en broadcast. Cela est illustré par la capture ci-dessous.

```

void setup()
{
    //Set of the tx power (from arduino to peripherals)
    BLE.setTxPower(4);

    Serial.begin(9600);
    // BLE object is actually now defined in CurieBLE.h
    BLE.begin();

    Serial.println("BLE Central scan");
    Serial.println(serviceID);

    // When device is discovered handle centralDiscovered() is called.
    BLE.setEventHandler(BLEDiscovered, centralDiscovered);
    //Log to change
    Serial.println("Arduino 1 starts scanning");
    BLE.scan(true);

    // set the local name peripheral advertises
    BLE.setLocalName(serviceID);
    // set the UUID for the service this peripheral advertises:
    BLE.setAdvertisedService(beaconService);

    // add the characteristics to the service
    beaconService.addCharacteristic(dataCharacteristic);

    // add the service
    BLE.addService(beaconService);
    dataCharacteristic.setValue(0);

    // start advertising
    BLE.advertise();
    //Fin du test
}

```

Figure 101 : Extrait de code arduino (3)

L'illustration ci-dessous représente le lancement du scan des données ainsi que la diffusion en broadcast du service associé à la balise fixe.

B. Mise en forme des données

La mise en forme des données consiste à remplir la structure de données définie dans l'analyse du programme.

```
data=(NumArduino<<24)+(NumBeacon<<16)+((short)(peripheral.rssi())*(-1));
```

Figure 102 : Mise en forme des données

L'extrait de programme ci-dessus permet d'empaqueter les données dans une caractéristique de **32 bits**.

C. Réception des données BLE

4.8.3.2 Mise en service de la communication en Bluetooth

Il s'agit maintenant de faire le chemin inverse. L'objectif est maintenant d'intercepter les services associés aux arduinos pour venir lire les caractéristiques dans le but de stocker les données pour la trilateration.

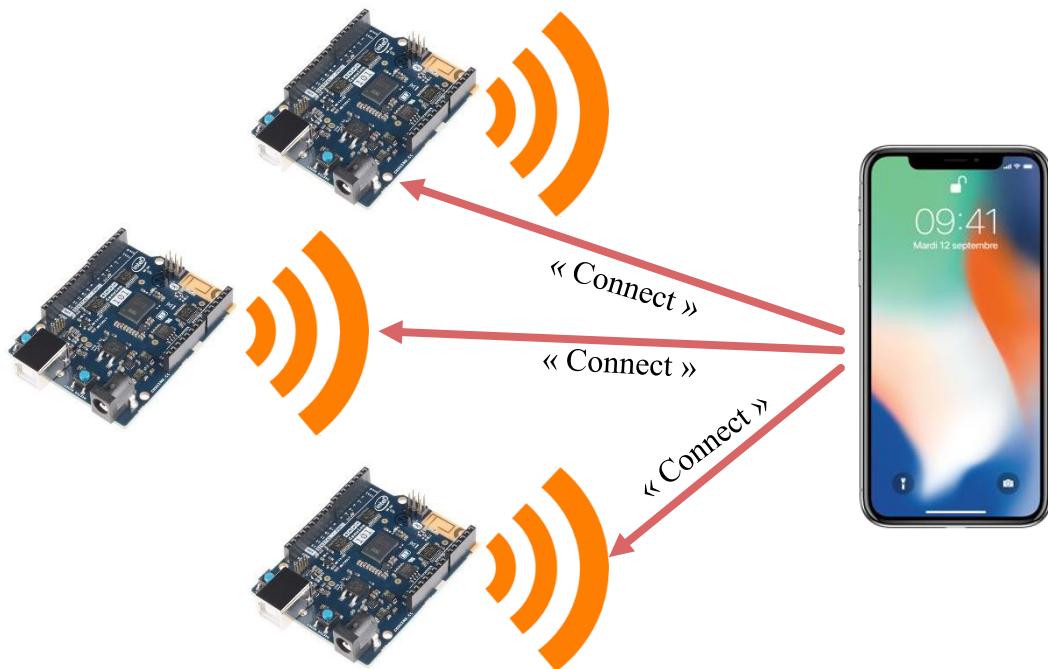


Figure 103 : Connexion aux balises

Pour mettre en œuvre cela, **une machine à états** a été conçue et programmée pour gérer les séquences de lecture.

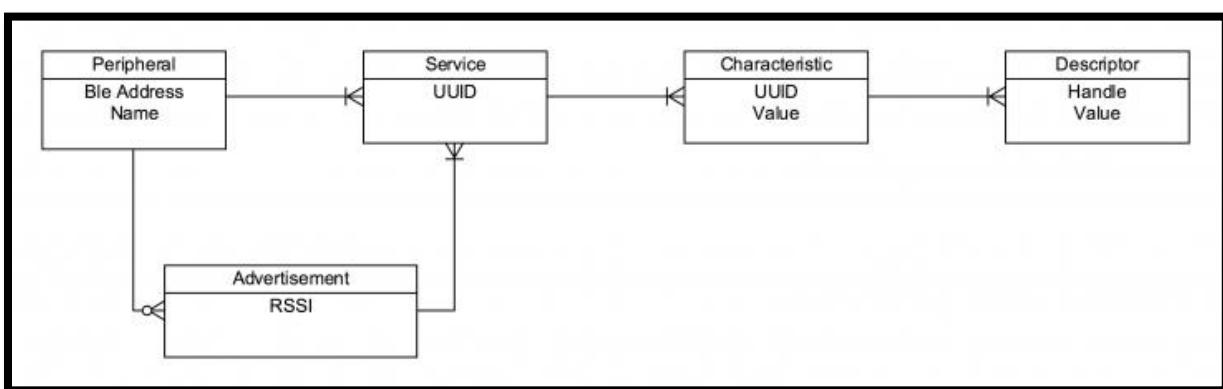


Figure 104 : Protocole de communication GATT

On rappelle le fonctionnement suivant : les balises émettent un ou plusieurs services, ces derniers contiennent des caractéristiques qui sont-elles mêmes décrites par des **descripteurs**.

Pour accéder aux données d'un service, il faut donc implémenter les actions suivantes :

- Scan des services
- Connexion à un périphérique
- Lecture d'une caractéristique

Remarque : pour implémenter ces fonctions, il est nécessaire d'avoir les autorisations liées à la localisation et au GPS (ACCESS_FINE_LOCATION / ACCESS_COARSE_LOCATION / BLUETOOTH / BLUETOOTH_ADMIN)

La schématisation de la machine à états qui gère les actions précédemment décrites :

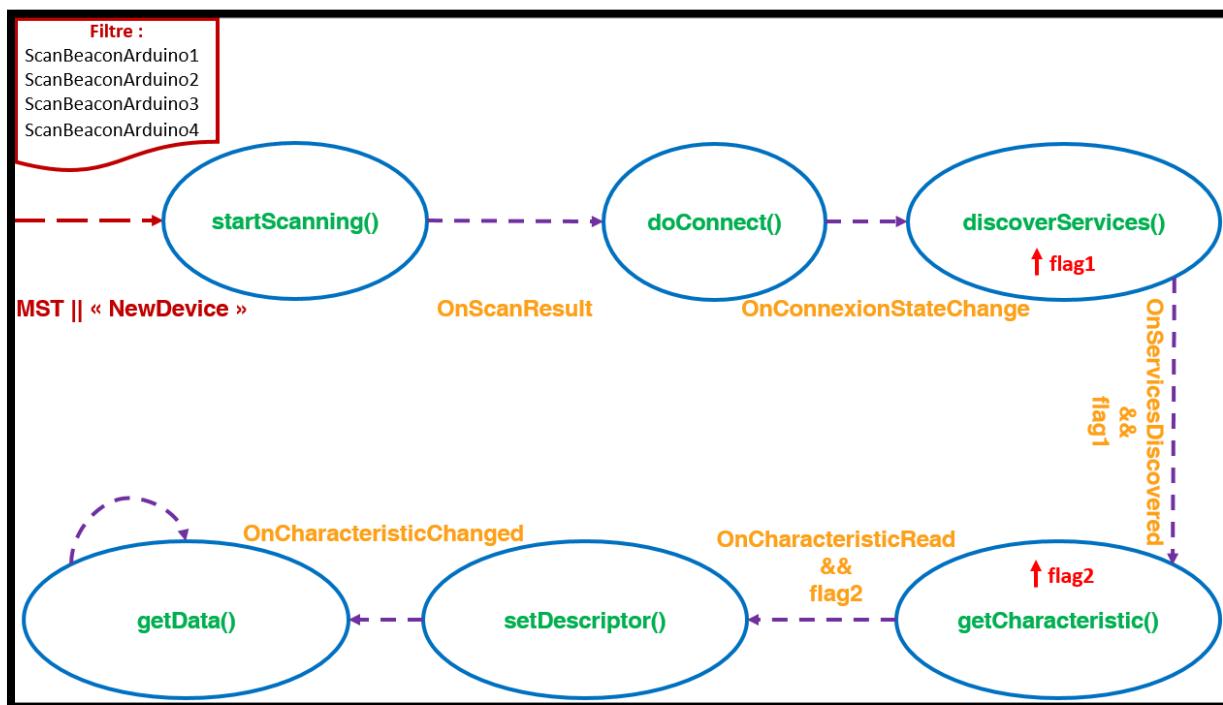


Figure 105 : Machine à états

➤ **startScanning()** :

Cette fonction paramètre le filtre de scan et lance le scan des services environnants. La fréquence de scan choisie est la plus élevée (SCAN_MODE_LOW_LATENCY).

```

public void startScanning(){
    scanner = adapter.getBluetoothLeScanner();
    ScanSettings scanSettings = new ScanSettings.Builder().setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY).build();
    List<ScanFilter> scanFilters = new ArrayList<>();
    scanFilters.add(new ScanFilter.Builder().setDeviceName("ScanBeaconsbyArduino1").build());
    scanFilters.add(new ScanFilter.Builder().setDeviceName("ScanBeaconsbyArduino2").build());
    scanFilters.add(new ScanFilter.Builder().setDeviceName("ScanBeaconsbyArduino3").build());
    scanFilters.add(new ScanFilter.Builder().setDeviceName("ScanBeaconsbyArduino4").build());
    scanner.flushPendingScanResults(scanCallback);
    scanner.startScan(scanFilters, scanSettings, scanCallback);
    System.out.println("Scanner on !");
}

```

Figure 106 : Fonction startScanning()

➤ **doConnect()** :

Cette fonction permet de se connecter aux périphériques précédemment scannés. Le principe d'échange des services et des caractéristiques est appelé **GATT**. Il faut donc créer une instance de la classe BluetoothGatt. C'est de cet objet dont on se sert pour toutes les opérations liées au BLE. Il est nécessaire de temporiser après la connexion. En effet, il faut souvent temporiser entre deux opérations sur l'objet GATT. Cela contribue notamment à éviter des problèmes de connectivité.

```

public void doConnect(ScanResult result){
    device = adapter.getRemoteDevice(result.getDevice().getAddress());
    new Thread(new Runnable() {
        public void run() {
            gatt = device.connectGatt(currentActivity.getApplicationContext(), autoConnect: true, new myGattCallBack());
            try {
                Thread.sleep( millis: 5000 );
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            refreshDeviceCache(gatt);
        }
    }).start();
}

```

Figure 107 : Fonction doConnect()

➤ **discoverServices()** :

Cette fonction permet d'accéder aux services de l'appareil auquel on s'est connecté.

```

public void discoverServices(int newState, BluetoothGatt gatt){
    if (newState == BluetoothProfile.STATE_CONNECTED) {
        Log.i( tag: "TEST", msg: "Connected to GATT peripheral. Attempting to start service discovery");
        gatt.discoverServices();
    } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {
        Log.i( tag: "TEST", msg: "Disconnected from GATT peripheral" );
        gatt.disconnect();
    }
}

```

Figure 108 : Fonction discoverServices()

➤ **getCharacteristic()** :

Cette fonction permet d'accéder aux caractéristiques des services de l'appareil auquel on s'est connecté. Il suffit de lancer la fonction readCharacteristic de l'objet GATT pour y accéder.

```
public void getCharacteristic(int status, BluetoothGatt gatt){  
    if (status == BluetoothGatt.GATT_SUCCESS) {  
        List<BluetoothGattService> services = gatt.getServices();  
        for (BluetoothGattService service : services) {  
            if (service.getUuid().toString().equalsIgnoreCase(uuidService)) {  
                BluetoothGattCharacteristic characteristic = service.getCharacteristics().get(0);  
                gatt.readCharacteristic(characteristic);  
            }  
        }  
    } else {  
        Log.i("TEST", "onServicesDiscovered received: " + status);  
    }  
}
```

Figure 109 : Fonction *getCharacteristic()*

➤ **setDescriptor()** :

Cette fonction permet de mettre en place les notifications. Cela permet de récupérer la valeur de la caractéristique à chaque changement de valeur. Ensuite, il faut modifier le descripteur de caractéristique pour autoriser les notifications.

```
public void setDescriptor(int status, BluetoothGattCharacteristic characteristic, BluetoothGatt gatt){  
    if (status == BluetoothGatt.GATT_SUCCESS) {  
        gatt.setCharacteristicNotification(characteristic, enable: true);  
        BluetoothGattDescriptor descriptor = characteristic.getDescriptors().get(0);  
        descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);  
        gatt.writeDescriptor(descriptor);  
    }  
}
```

Figure 110 : Fonction *setDescriptor()*

➤ **getData() :**

Cette fonction est exécutée à chaque fois que l'on reçoit l'événement « CharacteristicChanged». Ainsi, il est possible de récupérer les données et de les mettre en forme à chaque changement de valeur.

```
public void getData(BluetoothGattCharacteristic characteristic) {
    Log.e( tag: "TEST", msg: "getData");
    Long[] result = {01,01,01,01};
    byte[] data = characteristic.getValue();
    Log.e( tag: "Data", Arrays.toString(data));
    for(int j=0;j<=data.length-1;j++){
        if(data[j]<0) {
            result[j] = (long) data[j] + 256 ;
        }
        else{
            result[j]= (long) data[j];
        }
    }
    dataManager.extractData(result,data);
}
```

Figure 111 : Fonction getData()

C'est dans la fonction getData() que l'on décompose le tableau d'octets data. Il s'agit de la caractéristique que l'on vient lire. Ce tableau contient les informations de la structure de données. Le numéro du Beacon et le numéro de l'arduino sont accessibles au niveau du 3^{ème} et du 4^{ème} octet du tableau. Le RSSI (deux premiers octets) étant codé sur deux octets, nécessite une opération de masquage pour récupérer la valeur.

```
RSSI=(short) ((result[1] << 8) | (result[0] & 0xFF));
```

Figure 112 : Calcul du RSSI

Remarque : pour toutes les valeurs négatives, on effectue une addition pour ajouter +256. Ainsi, cela donne une valeur allant de 0 à 256. Cette opération est nécessaire car un octet en java détient une plage de valeurs allant de -128 à +127.

4.8.3.3 Prise en main de la librairie de trilateration

Rappel : la trilateration est une méthode mathématique issue de la géométrie. Elle permet de déterminer la position relative d'un point en fonction d'un minimum de trois distances. Cette méthode est utilisée pour les systèmes de positionnement de satellites.

Le principe est de calculer l'intersection entre les cercles de distances donnés.

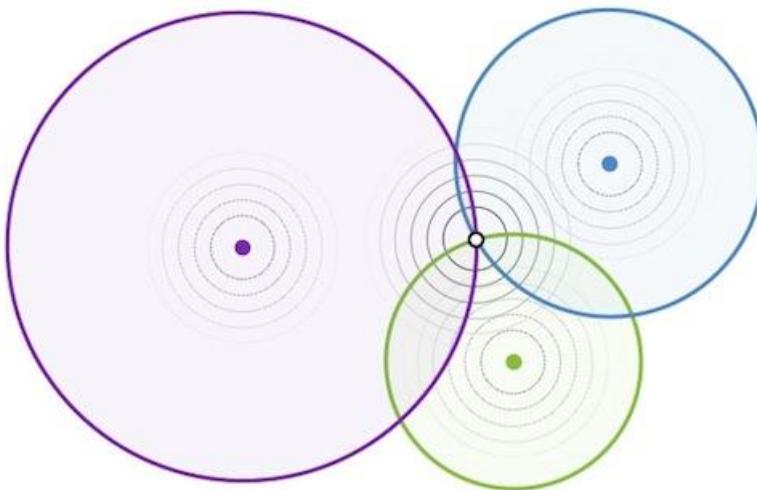


Figure 113 : Trilateration

Grâce à nos quatre balises, nous pouvons obtenir quatre de ces cercles et donc définir un point précis sur la position du Beacon.

Cette position est renvoyée par une librairie de trilateration qui a été choisie lors de l'étude. Il s'agit de la librairie de **Lemmingapex** utilisée lors du projet précédent. Ce choix a été fait étant donné que cette librairie est fonctionnelle et facile à prendre en main.

Cette librairie doit impérativement être utilisée avec des **balises fixes dont on connaît la position** et dans un plan en **deux dimensions**.

```
public void MaFonctionTrilateration() throws Exception {
    double[][][] positions = new double[][][]{{0.0, 0.0}, {20.0, 0.0}, {0.0, 20.0}, {20.0, 20.0}};
    double[] distances = new double[]{2.0, 4.0, 1.0, 3.0};
```

Figure 114 : Librairie trilateration (1)

Ici nous voulons réaliser un positionnement dans un plan à **deux dimensions**, c'est pourquoi nos positions des balises fixes sont des positions [x ; y]. Dans un premier temps nous remplissons le tableau avec la position des Beacons fixes dans le plan choisi au préalable.

Il est idéal par soucis de simplicité, de positionner les Beacons aux quatre coins de la pièce. Néanmoins dans le cas où les Beacons n'auraient pas une portée suffisante il est tout à fait possible de les disséminer.

La précision du calcul résultera de la précision de la distance mesurée.

```
TRILATERATIONFUNCTION trilaterationFunction = new TrilaterationFunction(positions, distances);
LinearLeastSquaresSolver lsSolver = new LinearLeastSquaresSolver(trilaterationFunction);
NonLinearLeastSquaresSolver nlSolver = new NonLinearLeastSquaresSolver(trilaterationFunction, new LevenbergMarquardtOptimizer());
linearCalculatedPosition = lsSolver.solve();
nonLinearOptima = nlSolver.solve();
```

Figure 115 : Librairie trilateration (2)

Ensuite, il faut créer l'objet trilaterationfunction pour y insérer les données du tableau précédemment rempli. Enfin, on demande le traitement avec la méthode solve.

```
double[] calculatedPosition = nonLinearOptimum.getPoint().toArray();
```

Figure 116 : Tableau résultant

Pour récupérer les données, il suffit de lire des données (x,y) dans le tableau de sortie. Ce fonctionnement est parfaitement conforme aux spécifications énoncées lors de l'étude.

A. Développement du lissage de données

```
public void addValue(float new_value, int fixedBeacon, int beacon){

    float average = getAverage(fixedBeacon,beacon);

    if((new_value - average <= 2 && new_value - average >= -2) || Arrays.asList(arrayAverage[fixedBeacon][beacon]).contains(0)){
        float temp;
        for(int i = 18; i > -1; i--){
            temp=arrayAverage[fixedBeacon][beacon][i];
            arrayAverage[fixedBeacon][beacon][i + 1]=temp;
        }
        arrayAverage[fixedBeacon][beacon][0]=new_value;
        setAverage(fixedBeacon,beacon);
    }
}
```

Figure 117 : Développement du lissage de données

Ce système permet d'avoir facilement des valeurs qui se rapprochent de la réalité et ainsi, d'avoir des affichages cohérents.

Ensuite, ces valeurs sont utilisées par l'interface graphique pour configurer la position des Beacons sur la carte grâce à la trilateration.

B. Gestion de l'échelle

Pour pouvoir retranscrire la position d'une cible sur un plan, il est nécessaire d'effectuer une **conversion** de manière à ce que le plan affiché soit le reflet de la réalité mais sous une **échelle réduite**.

Pour cela, il est nécessaire d'utiliser les formules suivantes :

$$positionX(px) = \frac{positionX(m) * largeurÉcran(px)}{largeurPièce} + offsetX(m)$$

$$positionY(px) = \frac{positionY(m) * longueurÉcran(px)}{longueurPièce (m)} + offsetY(m)$$

Figure 118 : Formules de mise à l'échelle

Ces formules sont de simples produits en croix. Ces calculs qui permettent d'avoir la position en pixels des balises en fonction des véritables positions en mètre. Voici ci-dessous le programme qui permet de réaliser cette fonction.

```
public float settingScale(String positionXY, ImageView object, String type){  
    if(type=="x"){  
        return ((Float.parseFloat(positionXY) * deviceWidth / Float.parseFloat(dataTable[1]))  
                + Float.parseFloat(dataTable[2])) - object.getWidth()/2;  
    }  
    else if(type=="y"){  
        return ((Float.parseFloat(positionXY) * deviceHeight / Float.parseFloat(dataTable[0]))  
                + Float.parseFloat(dataTable[3])) - object.getHeight()/2;  
    }  
    else{  
        return -1f;  
    }  
}
```

Figure 119 : Fonction settingScale()

Afin de tester notre application dans le but de valider le fonctionnement, nous avons réalisé le test suivant :

- Fixer la position du Beacon
 - Par exemple : $x = 4 ; y = 4$
- Fixer la distance des balises par rapport au Beacon
 - Par exemple : $d_1 = 1, d_2 = 2, d_3 = 3, d_4 = 4$
- Calculer la position des balises par rapport aux distances et la position du Beacon
 - Par exemple :
 - $X_1 = 3 ; X_2 = 2 ; X_3 = 7 ; X_4 = 4$
 - $Y_1 = 4 ; Y_2 = 4 ; Y_3 = 4 ; Y_4 = 8$
- Une fois ces tâches effectuées, contrôler le résultat de la trilateration qui doit être égal à $(4, 4)$

C. Développement de l'interface graphique

a. Interface administrateur

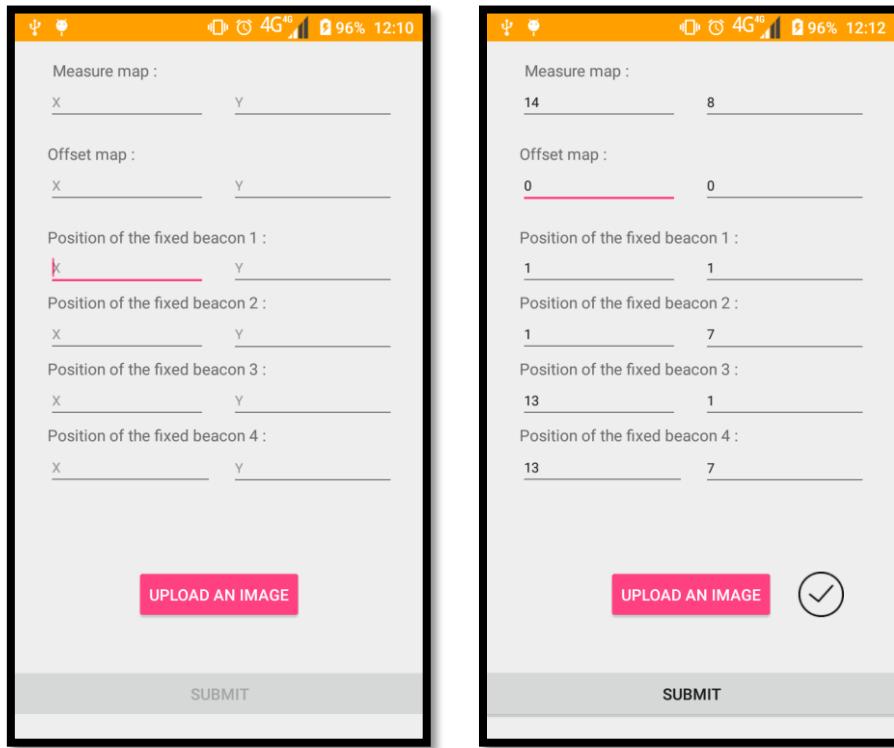


Figure 120 : Interface administrateur

Une fois la connexion valide de l'utilisateur administrateur, nous pouvons charger l'interface administrateur. Cette interface est composée de plusieurs EditText et de TextView associés, ce qui permet à l'administrateur de configurer la partie Map de l'application. Nous devons **sauvegarder** les mesures de la carte, un offset si besoin ainsi que les coordonnées de chaque balise fixe. Ces données sont sauvegardées de la même manière que l'identifiant via un SharedPreferences (cf Annexe 3 : Shared Preferences).

Pour charger une carte, nous avons mis en place un bouton permettant à l'utilisateur de naviguer dans la mémoire de son téléphone pour sélectionner une image qui sera chargée mais aussi sauvegardée dans les données de l'application. Pour cela nous devons vérifier que nous avons bien le droit de sauvegarder dans le téléphone. Pour l'utilisateur, cela est totalement transparent. Une fois l'image sauvegardée, un petit check apparaît et rends le bouton valider disponible. Pour chaque donnée numérique, nous prenons pour la suite du traitement la valeur 0, comme valeur par défaut.

b. Interface « localisation »

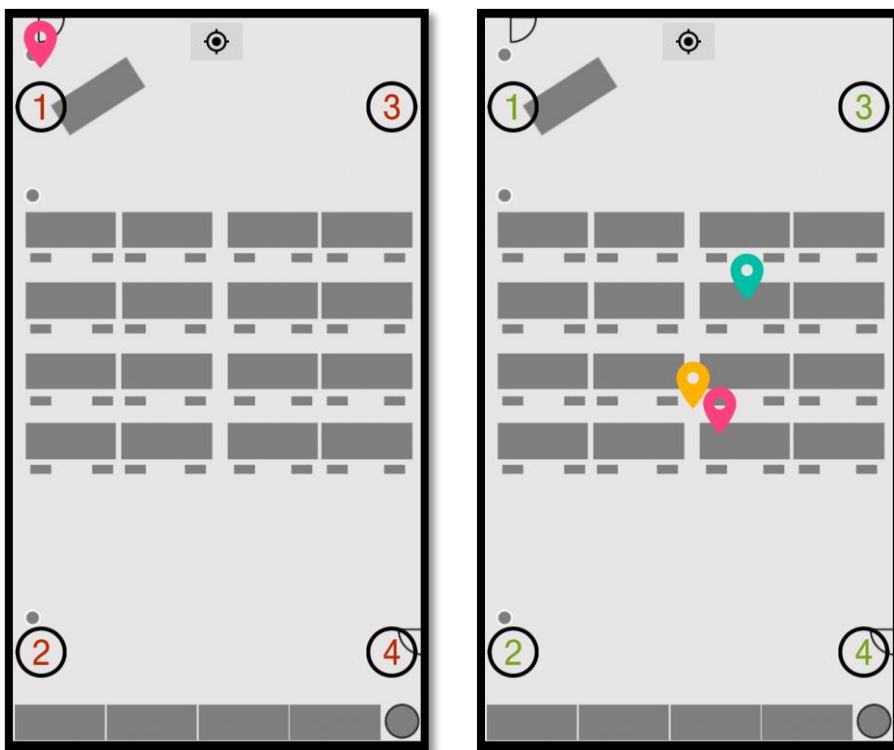


Figure 121 : Interface « localisation »

Suite à une configuration des coordonnées dans le menu administrateur, nous pouvons afficher les différentes positions de chaque balise fixe mais aussi la carte représentant la zone dans laquelle se trouvent les Beacons. Dans le cas contraire nous invitons l'utilisateur à communiquer avec l'administrateur pour configurer l'application.

Les cercles avec un chiffre à l'intérieur correspondent aux balises fixes, ce sont des ImageViews, elles possèdent trois états de couleurs différentes, une couleur **rouge** quand la balise ne détecte aucun Beacon, une couleur **orange** quand elle détecte au moins un Beacon et enfin une couleur **verte** pour indiquer qu'elle détecte tous les Beacons possibles. Nous appliquons juste un changement de couleur bien précis à notre image puisqu'on utilise une seule image par balise.

Ce principe est combiné avec le principe de génération de couleurs pour les Beacons afin d'afficher un nombre indéterminé de Beacon, avec la possibilité d'avoir une couleur différente pour chacun d'entre eux. Un bouton est présent en haut de l'écran qui après un clic, affiche une **boîte de dialogue** avec des Checkbox pour ne sélectionner que les Beacons souhaités afin de n'afficher que ceux sélectionnés sur la carte.

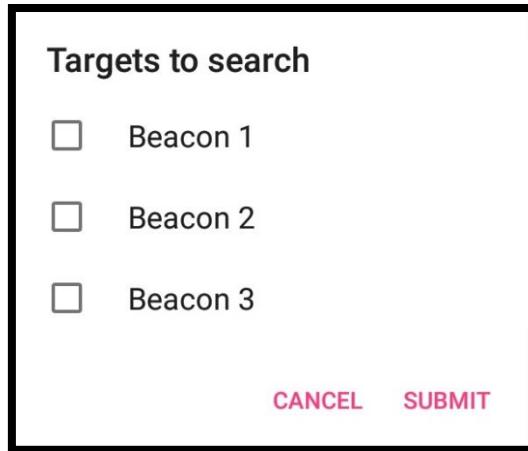


Figure 122 : Boîte de dialogue «sélection des beacons »

4.8.4 Interfaçage des sous-ensembles

Une fois tous les sous-ensembles validés, l'étape suivante était de tout interfacer. Pour cela, il a été choisi de ré-analyser la structure du programme et de faire évoluer **le diagramme des classes** construit lors de l'analyse.

Les classes principales sont :

- **Beacon** : classe représentant un Beacon physique.
- **BleManager** : classe gérant la communication en Bluetooth.
- **DataManager** : classe gérant le traitement des données (ex : lissage).
- **MapActivity** : classe gérant tout ce qui est lié à l'activité Map.
- **ParcBeacon** : classe qui gère le parc de Beacons.
- **Trilateration** : classe qui implémente la librairie de trilateration.

Voici ci-dessous un diagramme des classes qui représente tout ce qui vient d'être cité :

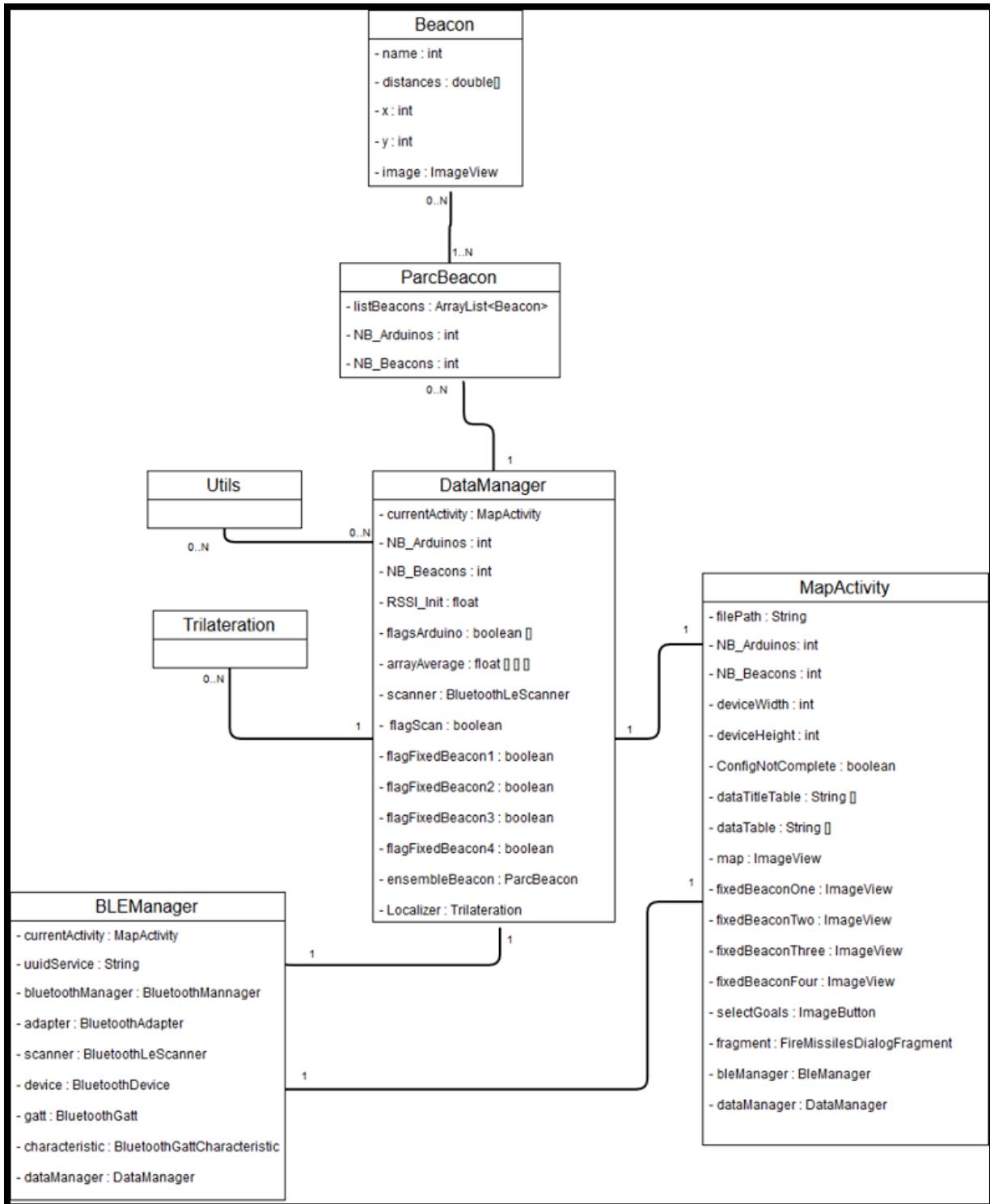


Figure 123 : Nouveau diagramme des classes

4.8.5 Optimisation des programmes

4.8.5.1 Évolutivité du programme

Aujourd’hui, notre programme est évolutif à **75-80%**. Il est de ce fait, possible d’assigner un nombre de Beacon et ainsi, l’application s’adapte au nombre spécifié.

4.8.5.2 Décomposition en threads

Pour mieux structurer l’application, une réflexion a eu lieu sur la gestion des processus de notre application. En effet, il a été choisi de créer un **thread** supplémentaire de façon à gérer la communication et le traitement des données dans un processus à part de l’interface graphique. Ainsi, cela évite toute sorte de **blocage** au niveau de l’interface graphique.

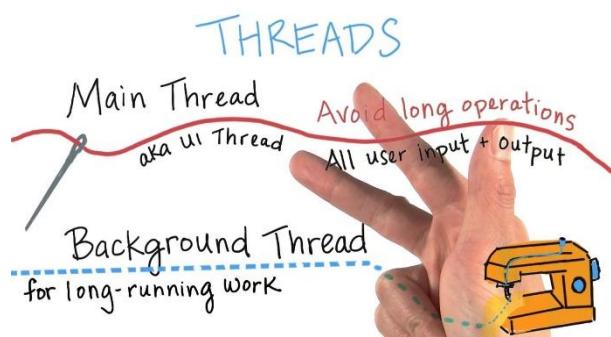


Figure 124 : Threads

Tout ce qui est lié à la communication en Bluetooth est exécuté par ce que l’on appelle des binders. Les **binders**³ servent à effectuer la communication entre des processus. Dans le cas présent, c’est ce mécanisme qui permet de faire communiquer l’application avec le Framework BLE. Par conséquent, il est nécessaire de créer son propre thread pour avoir le contrôle sur l’exécution du programme. En effet, l’objectif est de centraliser toutes les opérations BLE en un seul thread, dans le cas présent : BleManager. Ce problème a été résolu grâce à la **machine à états**.

Remarque : la classe BleManager hérite de la classe thread.

4.9 Tests & expérimentation

4.9.1 Précision des Beacons

Cette étude a pour but dans un premier temps de comparer les **caractéristiques théoriques** des Beacons BNB avec les **caractéristiques en conditions réelles**. Dans un second temps, le but est de montrer des différences notables qui existent entre un Beacon « bas de gamme » et un Beacon de qualité supérieure. (Le Beacon BNB a été configuré avec la puissance d’émission **+4dbm**).

³ Binder : décrit une interface entre deux processus

4.9.1.1 Atténuation du signal (*champ libre*)

Les relevés ont été réalisés en extérieur dans un parking sans obstacle.

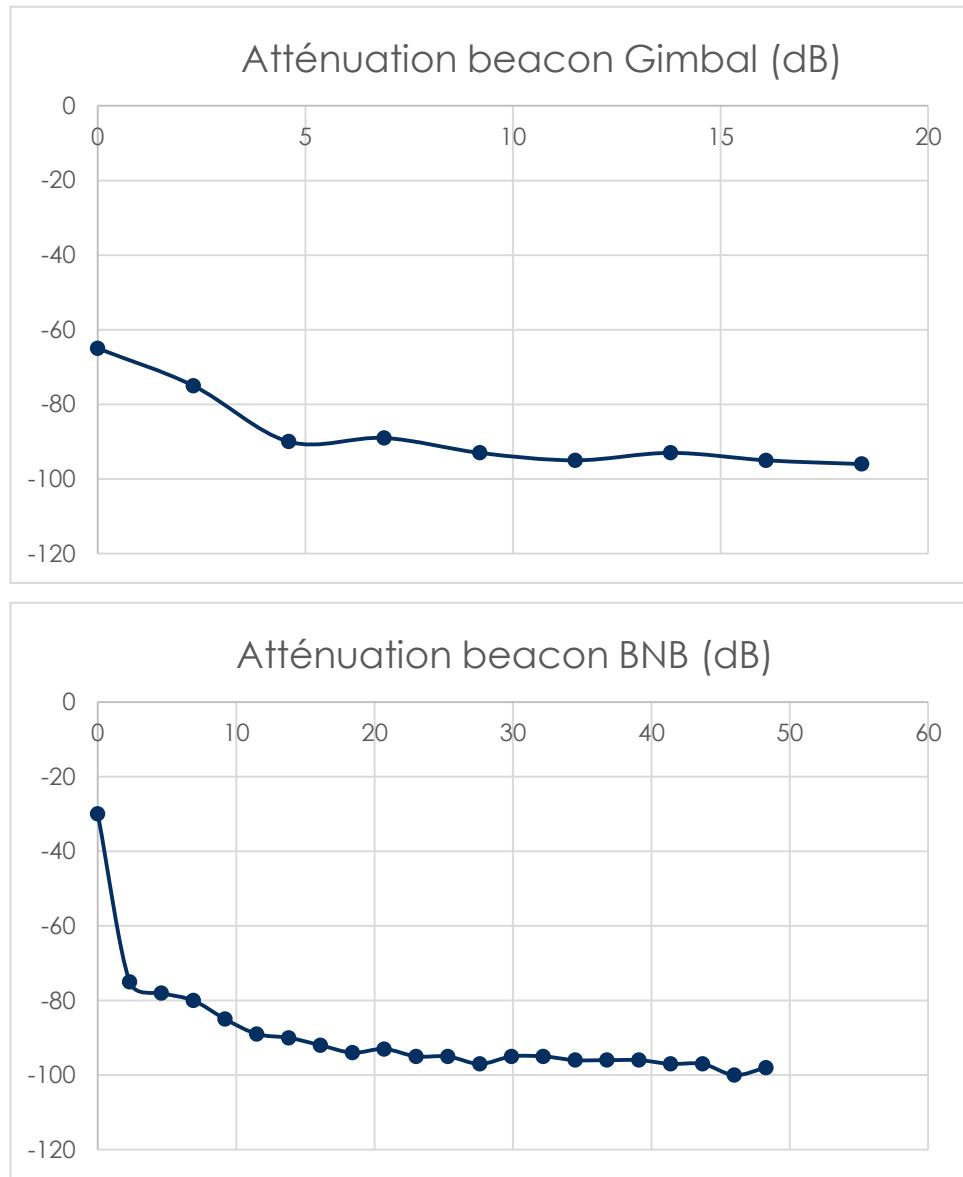


Figure 125 : Atténuation du signal des beacons

À l'issue de ces relevés, nous pouvons premièrement constater que la différence en termes de **portée** est conséquente (théorique : 100m pour BNB / 50m pour Gimbal).

Deuxièmement, nous pouvons constater que la puissance du signal reçue reste assez médiocre tout le long de la variation de la distance pour le Beacon Gimbal. Cependant, l'atténuation varie avec la distance pour le Beacon BNB.

Troisièmement, nous pouvons aussi relever la différence notable entre les valeurs énoncées par le fabricant et celles relevées de manière expérimentale. Les tests effectués par le fabricant se sont déroulés en zone urbaine dégagée. L'espace de travail à un impact important sur les résultats.

4.9.1.2 Atténuation du signal (*champ obstrué*)

Cette étude a été réalisée pour pouvoir comparer la puissance d'émission des deux Beacons à la puissance du signal reçu. Pour cela nous avons relevé une valeur nominale à **1 mètre** sans aucun obstacle puis nous avons mesuré ce même RSSI avec un obstacle de matière variable.

(dB)	Gimbal	BNB
Nominal (1m)	-65	-53
Bois	-70	-54
Placo	-71	-63
Corps humain	-76	-68
Verre	-85	-78

Figure 126 : Atténuation du signal (*avec obstacle*)

Comme nous avons pu voir précédemment, la différence de qualité et de puissance d'émission se fait ressentir notamment entre les deux types de Beacons. Nous pouvons voir que le Beacon BNB est sensiblement meilleur sur toutes sur les surfaces par rapport au Beacon Gimbal.

4.9.2 Précision des balises fixes

Nous avons réalisé des essais afin de relever l'atténuation des signaux des balises fixes (arduinios) dans des environnements différents (un dégagé et un obstrué par des murs). Le but étant de voir si les signaux sont réellement **perturbés**. Ces tests ont été réalisés à 1 mètre, 3 mètres, 5 mètres et 10 mètres.

4.9.2.1 Tests en milieu dégagé

Le test est effectué avec l'antenne vers le téléphone et l'antenne du téléphone vers l'arduino sur un axe du même plan.

Remarque : *la puissance d'émission a été réglée à 55dB pour les arduinos*

* ScanBeaconsbyArduino3 MAC: 98:4F:EE:10:7A:CC Updated: 2018-05-15 11:52:12 RSSI: -52.0db / -56.0db	1 mètre
* ScanBeaconsbyArduino3 MAC: 98:4F:EE:10:7A:CC Updated: 2018-05-15 11:52:34 RSSI: -62.0db / -67.0db	3 mètres
* ScanBeaconsbyArduino3 MAC: 98:4F:EE:10:7A:CC Updated: 2018-05-15 11:52:55 RSSI: -66.0db / -65.0db	5 mètres
* ScanBeaconsbyArduino3 MAC: 98:4F:EE:10:7A:CC Updated: 2018-05-15 11:53:12 RSSI: -75.0db / -73.0db	10 mètres

Figure 127 : Relevés Arduino (1)

Observations

Nous pouvons voir qu'en extérieur, l'atténuation des Beacons reste faible en fonction des mètres parcourus. Cependant, cette même atténuation reste visible et semble normale en prenant pour référence une règle trouvée dans un cours de Jean-Philippe MULLER :

Une règle empirique permet de prévoir « en gros » la portée d'une liaison Bluetooth :

- atténuation de 50 dB pour les 6 premiers mètres
- atténuation de 10 dB par 10 mètres supplémentaires

Figure 128 : Portée d'une liaison Bluetooth

Cela correspond plus ou moins à nos données étant donné que c'est une règle empirique et qui dépend aussi du matériel utilisé.

4.9.2.2 Tests en milieu obstrué

Le test est effectué avec l'antenne vers le téléphone et l'antenne du téléphone vers l'arduino sur un axe du même plan.

ScanBeaconsbyArduino3 MAC: 98:4F:EE:10:7A:CC Updated: 2018-05-15 11:58:52 RSSI: -68.0db / -65.0db	1 mètre
ScanBeaconsbyArduino3 MAC: 98:4F:EE:10:7A:CC Updated: 2018-05-15 11:59:05 RSSI: -76.0db / -73.0db	3 mètres
ScanBeaconsbyArduino3 MAC: 98:4F:EE:10:7A:CC Updated: 2018-05-15 11:59:26 RSSI: -77.0db / -80.0db	5 mètres
ScanBeaconsbyArduino3 MAC: 98:4F:EE:10:7A:CC Updated: 2018-05-15 11:59:45 RSSI: -86.0db / -82.0db	10 mètres

Figure 129 : Relevés Arduino (2)

Observations

Nous pouvons voir que l'atténuation est bien plus forte dans un milieu obstrué. De ce fait, le calcul des distances peut être faussé et peut limiter la localisation des Beacons.

4.9.3 Expérimentation de la trilateration

4.9.3.1 Tests en salle Ada Lovelace

Nos premiers tests en situation réelle ont été effectués en salle Ada Lovelace. Le but étant de tester la portée des arduino et la précision de l'application dans un environnement dégagé et relativement grand.

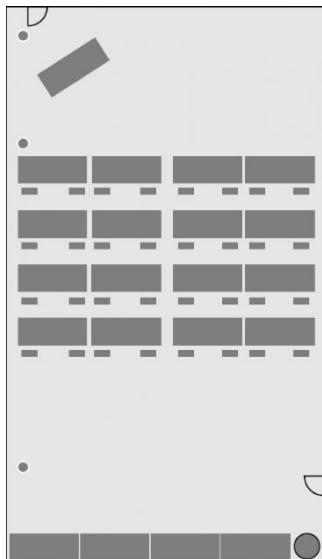


Figure 130 : Salle Lovelace

4.9.3.2 Configuration des tests

Nous avons placé les quatre arduinos aux quatre coins de la pièce avec une alimentation. Nous avons placé les Beacons dans la salle dans le but de vérifier sur l'application que le placement était conforme au plan. Comme présenté sur le panorama, les arduinos sont placées en **hauteur** pour éviter des perturbations venant des tables.

Remarque : il est très important que les balises soient à hauteur des Beacons pour une meilleure précision.



Personne ne doit être présent dans le champ des arduinos. Nous nous sommes donc mit à l'écart de l'espace.

Le plan ci-contre est à l'échelle et correspond à la salle Ada Lovelace. Nous n'avons pas pris en compte le placement des tables et des chaises dans nos tests.

Figure 131 : Croquis Salle Lovelace

4.9.3.3 Observations

Nous avons pu observer plusieurs points qui peuvent bloquer, mais d'autres nous ont permis de soulever des problèmes et de les résoudre.

Ces points sont les suivants :

- Les éléments métalliques créent des **perturbations** qui peuvent être importantes. Nous avons remarqué aussi que les stores en métal dans la salle lovelace pouvaient rendre les signaux très différents (+/- 20 dBm). En remontant les stores, nous avons observé une nette amélioration des résultats de la localisation.
Cependant, cela est inhérent aux ondes et il est impossible d'améliorer ce point. C'est aussi pour cela, que nous avons mis les balises en hauteur pour éviter que les pieds de tables ne perturbent le signal.
- Les Beacons font des « sauts » sur la visualisation, ce qui ne permet pas une localisation précise. Ces « sauts » sont dû au fait que **les signaux émis par les Beacons peuvent être perturbés** par différents paramètres et changent très souvent. En effet, le Bluetooth à la particularité d'être instable et très sensible aux perturbations.
Pour résoudre ce problème, nous avons mis au point un lissage des données, expliqué dans ce rapport.
- Le fait que la connexion entre le téléphone et les balises ne se fait pas du premier coup. Pour résoudre ce problème de stabilité, nous avons changé le raisonnement autour du code au profit d'une machine à états et de l'utilisation des threads.

4.10 Limites de la preuve de concept

Dans ce rapport, nous avons présenté les différents aspects de nos études et de la mise en œuvre de notre preuve de concept. Cependant, il existe des limites qui peuvent être inhérentes à la physique comme la limite de notre développement.

Dans un premier temps, il existe la limite liée aux **ondes des signaux Bluetooth**. Comme nous avons pu le voir tout au long de ce rapport, les ondes émises ne permettent pas une précision inférieure à **5 mètres prêt**. Pour cela, nous devrions faire du **traitement du signal** afin de filtrer et d'améliorer les signaux comme nous l'a précisé le commercial des Beacons BNB. Ce dernier nous a aussi indiqué qu'une entreprise avait réussi à avoir une précision à 50 centimètres prêt en faisant d'importants travaux de recherches et de développement à plein temps. Dans notre cas, il n'était pas envisageable de réussir une telle prouesse technique et nous avons fait avec nos moyens, déjà très bons. Néanmoins, ce contact nous a assuré que la trilateration était la meilleure solution.

Dans un second temps, nous sommes limités à **un seul utilisateur en connexion simultanée**. Cela s'explique par le fait que nous utilisons la diffusion des services BLE par du broadcast, or lorsque nous communiquons avec les arduinos, cela se fait par connexion. Il n'y a donc plus de broadcast et plus de possibilité de connexion avec les arduinos. Malgré tout, nous avons une solution possible qui est la déconnexion à intervalles réguliers aux arduino. Cependant, cela nécessite une synchronisation entre les arduinos, ce qui n'a pas été mis en place dans le cadre de ce projet.

Dans un troisième temps, **la connexion aux arduinos** peut s'avérer difficile dans certaines conditions. Nous pensons que cela est dû à la portée des signaux, tout de même faible, ce qui limite la détection des Beacons. De plus, il existe une limite au niveau de la connexion avec le téléphone. Nous suspectons le cache Bluetooth des téléphones Android. En effet, celui-ci garde en mémoire les connexions. Il est donc nécessaire de le réinitialiser lorsque l'on souhaite relancer une connexion et cela est possible par la désactivation et la réactivation du Bluetooth du téléphone. Pour pouvoir rester dans les « users guidelines » d'Android qui précisent qu'il n'est pas correct de modifier le statut des modules sans en avertir l'utilisateur, nous ne pouvons pas faire cette remise à zéro en restant transparent avec l'utilisateur. De ce fait, il doit de lui-même désactiver puis réactiver son Bluetooth.

Enfin, l'application n'est qu'à **75-80% évolutive**, c'est-à-dire qu'elle ne s'adapte pas encore entièrement au système qui lui est donné. Aujourd'hui, il suffit de préciser un nombre de Beacons pour que le système puisse gérer ce même nombre. Cela n'est pas possible pour les arduinos. Cela s'explique par l'utilisation de « flags » en durs qui nous permettent de savoir lorsque ces dernières sont connectées ou non. La solution sera d'utiliser des tableaux afin de rendre ce système plus évolutif. Néanmoins, nous avons fait en sorte de fournir le système le plus « fini » possible.

4.11 Pistes d'amélioration

4.11.1 Solution de positionnement

Pour revenir au constructeur Kontakt, celui-ci propose des solutions qui peuvent être intéressantes à étudier pour gagner en précision.

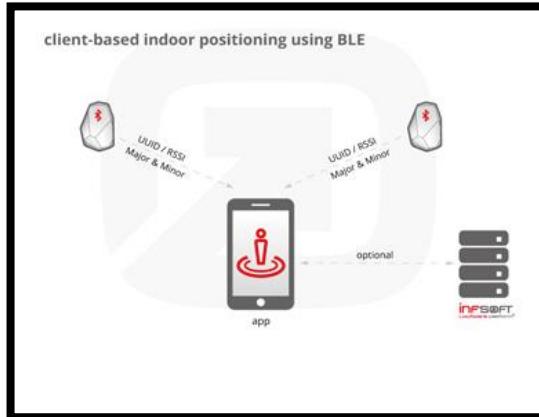


Figure 132 : Solution sans gateway (kontakt)

Cette première solution (ci-dessus) est celle que nous pourrons mettre en place sans la gateway propriétaire.

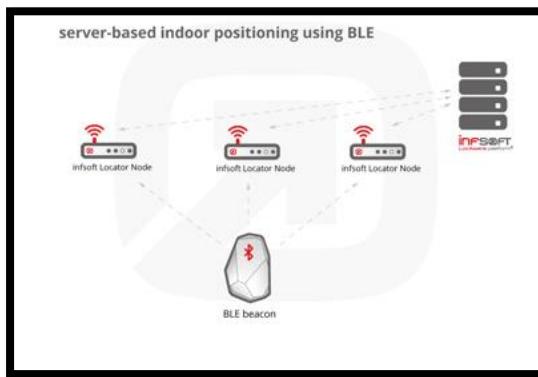


Figure 133 : Solution avec gateway (kontakt)

Cette seconde solution (ci-dessus) est celle qu'il est possible de mettre en place dans le cas où les Beacons ne permettraient toujours pas d'obtenir une précision suffisante. Celle-ci ferait ainsi appel à la balise dite gateway et le SDK imposé.

4.11.2 Étude sur le traitement du signal Bluetooth

Il existe plusieurs **algorithmes pour le calcul de la distance des Beacons**. Android utilise defacto, l'algorithme issue de la norme IEEE Xplore qui se nomme « RSSI-Based Real-Time User Location Tracking System for Indoor and Outdoor Environments ».

Le calcul suivant est réalisé :

$$d = 10^{\frac{ABS(RSSI) - A}{10^n}}$$

d : distance depuis l'émetteur

n : coefficient de perte de trajet (n=2 en environnement ouvert)

RSSI : puissance reçue

Figure 134 : Calcul de la distance (defacto)

Il existe aussi d'autres algorithmes de traitement du signal utilisables. Ceux-là sont basés sur des filtres de traitement du signal :

- Filtre moyenne mobile
- Filtre moyenne pondérée
- Filtre ajustement de courbe

Éventuellement, il est possible de **traiter le signal Bluetooth**, la solution serait de réaliser une carte pour l'atténuation du signal dans la pièce :

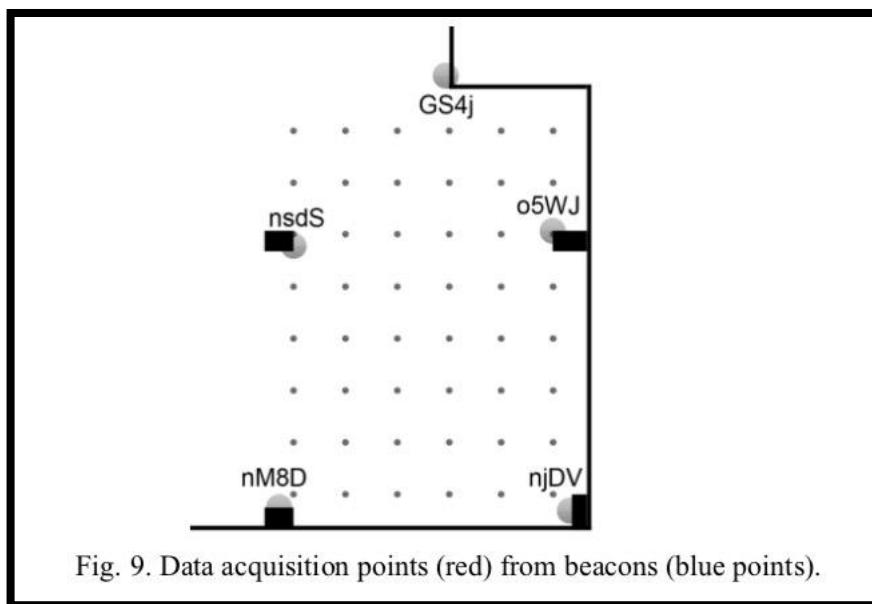


Figure 135 : Atténuation du signal dans une pièce

Cette méthode s'apparente à la création d'un modèle dit « Path-loss ». Nous allons tenter de vulgariser cette méthode, voici la procédure à suivre :

Une grille de point est réalisée dans la pièce, des récepteurs sont disséminés aux extrémités de la pièce. On place ensuite le Beacon sur la chaque point de la matrice de point réalisée plus tôt et on relève la valeur calculée par l'algorithme de trilateration. Ensuite, on compare la position calculée avec la position réel du Beacon et on évalue ainsi l'erreur. Enfin, on dresse une carte d'erreur de la pièce. Cette carte est

très intéressante car elle permet de voir où le signal est fiable et où il y a beaucoup de perturbations dans la pièce.

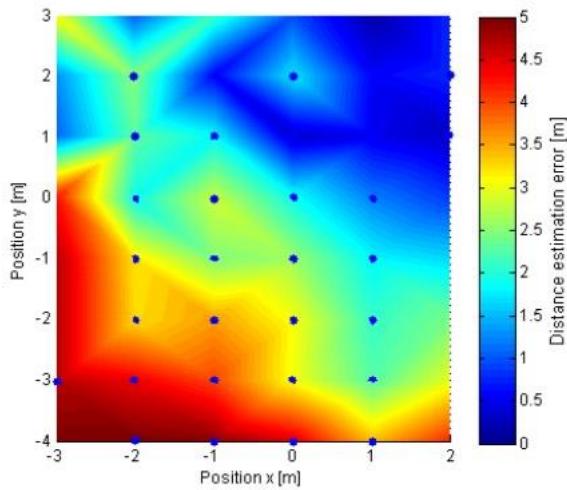


Figure 136 : Exemple d'une carte d'erreur

Il est possible de réaliser par la suite un **post-traitement** en sélectionnant les valeurs de certaines balises en fonction de la carte de bruit dressée précédemment.

Par exemple seulement les valeurs émanant des 3 balises les plus proches :

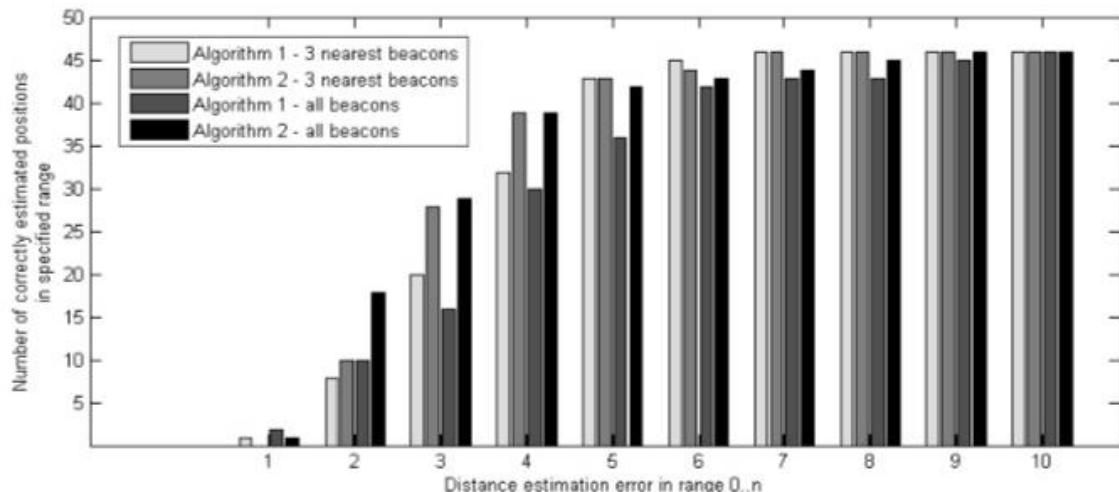


Figure 137 : Estimation de la distance

Une fois le bon algorithme et le bon post-traitement appliqué, les résultats peuvent être probants.

Voici ci-dessous la carte d'erreur de la même pièce avec l'algorithme et le post traitement adéquat.

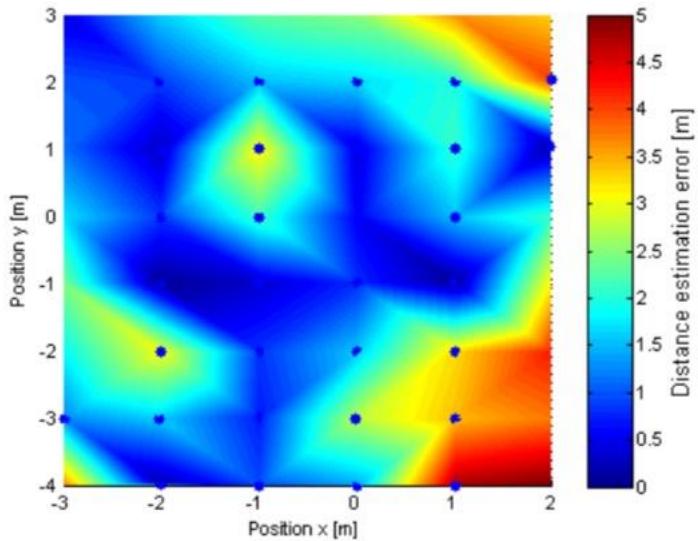


Figure 138 : Carte d'erreur après traitements

Les graphiques et l'analyse sont issus d'une étude complète et concise sur la comparaison entre deux différents algorithmes :

Analysis of Object Location Accuracy for iBeacon Technology based on the RSSI Path Loss Model and Fingerprint Map
Damian E. Grzechca, Piotr Pelczar, and Lukas Chruszczyk

4.12 Outils utilisés

Afin de gérer le **versionning** de notre application, nous avons mis en place un serveur GIT.

Nous avons donc chacun créé un compte **GitKraken** puis lié ce dernier à notre compte **GitHub** afin d'accéder à nos dépôts dans l'application.

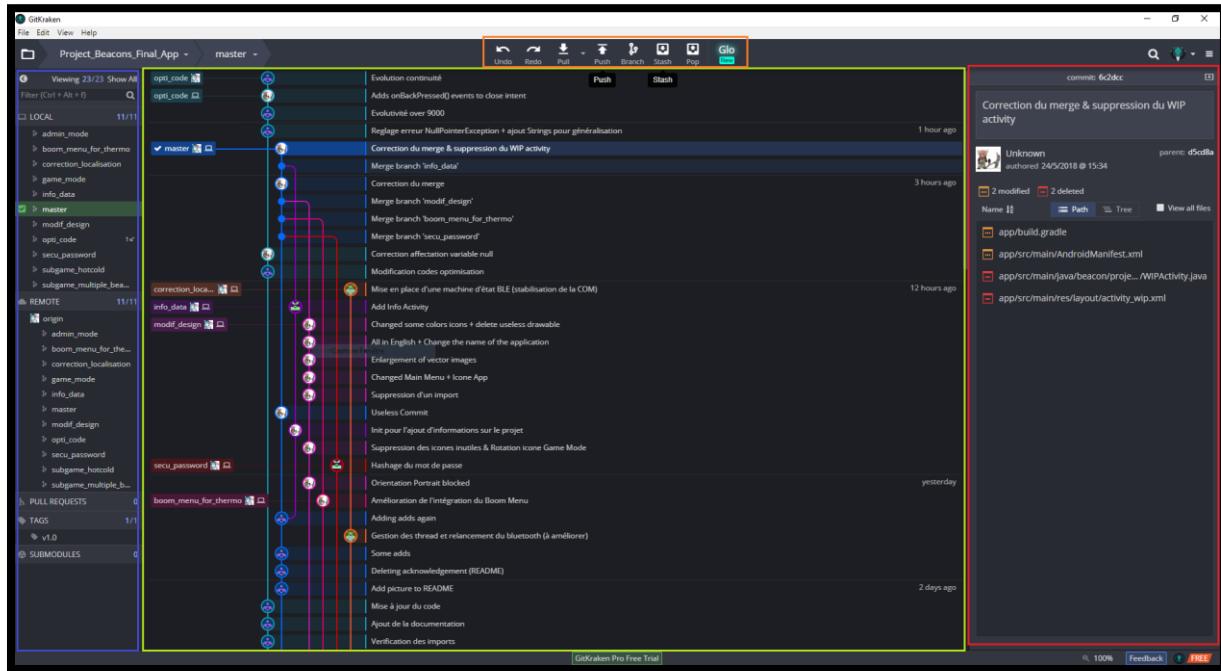


Figure 139 : Serveur GIT du projet

Cet outil a été très utile pour le stockage et l'indexage des programmes de manière à pouvoir travailler en parallèle (cf Annexe 4 : GITHUB).

Lien du GIT : https://github.com/Minipomme/Project_Beacons_Final_App

5 Présentation des résultats

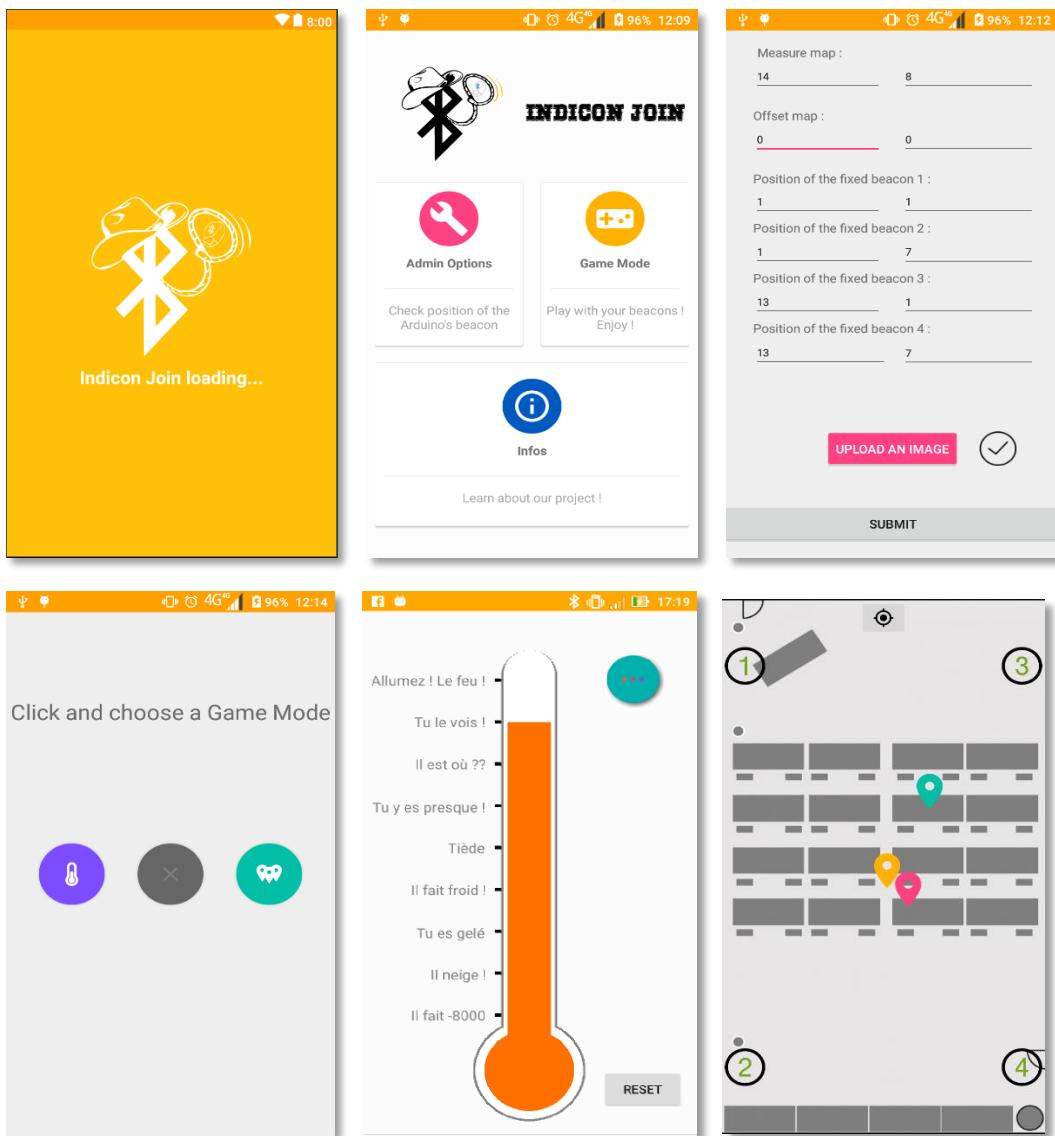


Figure 140 : Application finale

Voici les différentes captures d'écran de notre application et de ses différentes fonctions (non exhaustives). Une vidéo est disponible montrant le fonctionnement en utilisation réelle.

À la fin de ce projet, nous sommes capables de rendre une application finale avec un design qui lui est propre (avec son logo, ses couleurs...) ainsi que l'ensemble des fonctionnalités demandées par notre client M. MARTINEAU.

L'utilisateur peut :

- Créer un compte administrateur local dans l'application
- Modifier la position des balises fixes sur le plan.
- Modifier la taille de la carte

- Utiliser l'application chaud-froid et sélectionner le Beacon voulu
- Voir l'ensemble des Beacons se déplacer sur la carte
- Sélectionner un Beacon à suivre sur la carte

Le suivi actuel des Beacons est tout de même assez précis et nous pouvons assurer à **+/- 2 mètres** la position du Beacon. Avec le lissage des données, nous avons remarqué un décalage d'environ 2 secondes entre la position réelle et la position calculée correspondante au réel. En sachant cela, ce n'est pas handicapant et la précision n'est pas touchée par cela.

Les balises fixes s'affichent en **rouge** (état initial) lorsqu'elles ne reçoivent aucune données, en **orange** lorsqu'au moins un Beacon est capté et en **vert** lorsque tous les Beacons sont détectés. Cela permet à l'utilisateur de savoir si la connexion est établie ou non. Cela palie un peu à l'inconfort d'une interface qui ne réagit pas face aux défaillances.

6 Analyse des résultats

Au départ, ce projet s'annonçait plutôt complexe étant donné que c'est un projet complet et regroupant un certain nombre de nouvelles notions. De plus, à ce stade de la formation il est exigé de fournir un résultat dans les temps impartis. Par conséquent, nous avons à nouveau dû faire face à la problématique de l'apprentissage tout en continuant de faire vivre le projet dans le but de respecter la date limite impérative (**juin 2018**).

Au début du projet, nous avons passé beaucoup de temps à définir le projet et à effectuer une étude solide. Globalement, il n'y a pas eu de difficultés considérables sur ces tâches. Néanmoins, la communication a été très difficile surtout en début de projet.

En revanche, beaucoup de temps a été perdu sur la prise en main et la mise en œuvre du matériel. En effet, cela est notamment dû aux problèmes de communication dans le groupe suivi de longues périodes de partiels qui n'ont fait que décaler le planning. Néanmoins, nous avons su surpasser cette difficulté avec une **prise de recul**.

Comme dans tout projet, il est très important de **bien décomposer les tâches** et les fonctions en différentes parties et d'appliquer des **tests unitaires**. Durant ce projet, nous avons essayé de suivre les démarches qui nous ont été apprises, ce qui nous a été bénéfique. En effet, nous avons été en mesure de travailler en autonomie tout en essayant de répondre aux critères requis par notre formation. En ce qui concerne les analyses, nous avons toujours fait en sorte de les réaliser lors des phases de conception, c'est-à-dire, avant la mise en œuvre. Cependant, il nous est arrivé d'essayer d'avancer sans clairement analyser les problématiques. Ces expériences nous ont à nouveau démontré l'utilité des analyses et nous saurons, à l'avenir, tirer profit de ces expériences.

Concernant le planning, certaines **dérives de planning** ont été constatées. Ces retards s'expliquent par le fait que notre charge de travail est très difficile à répartir et surtout à estimer à l'avance. En effet, il est très difficile de gérer les projets en entreprise, le travail à l'école et les projets polytech. On peut avoir tendance à estimer les tâches sur de courts intervalles mais sans connaître les charges de travail qui nous seront imputées, par conséquent, cela génère des écarts entre les estimations et le réel. Néanmoins, les quelques dérives n'ont pas eu de réel impact sur la réalisation du projet car les quelques tâches qui ont été retardées ont été compensées par la suite. Eventuellement, le temps gagné aurait permis plus d'optimisation. Cependant, tout cela aurait éventuellement pu être évité.

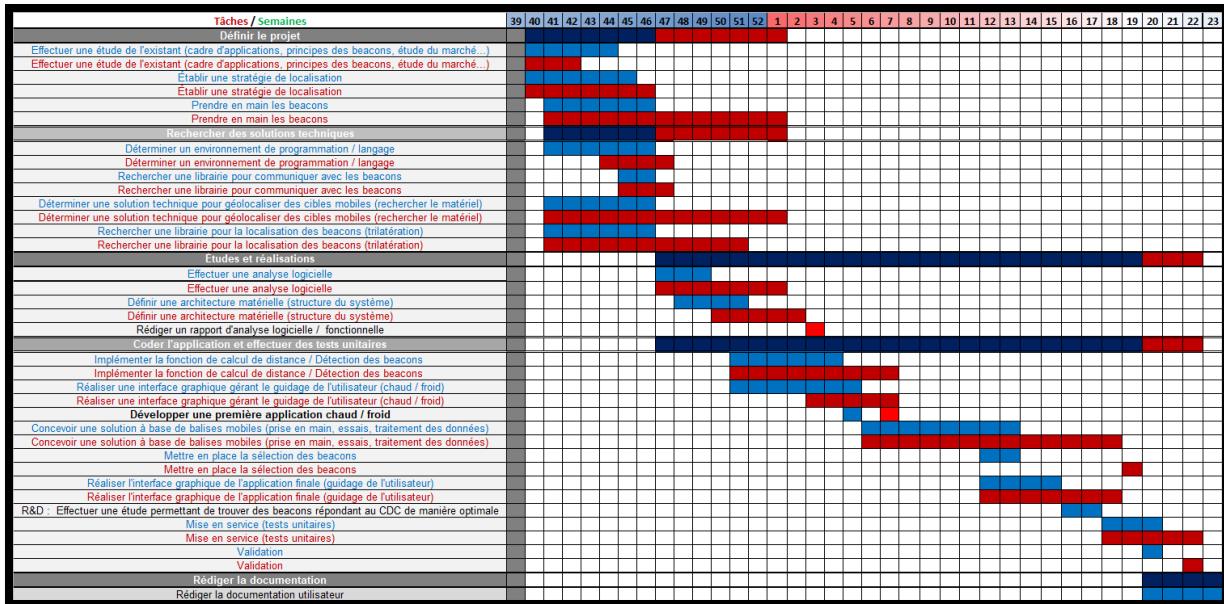


Figure 141 : Planning Prévisionnel / Réel

Voici ci-dessus le planning réel en comparaison avec le planning prévisionnel. On peut faire différentes remarques : la phase de définition de projet a énormément pris de retard par rapport à la prise en main du matériel. Néanmoins, nous avons su réagir vite pour rattraper ce retard. Nous avons été capables de diffuser la première version de l'application (« chaud-froid ») pour les journées portes ouvertes. De plus en ce qui concerne la phase de réalisation, plusieurs retards ont eu lieu à cause de la charge de travail. Cependant, si on regarde sur le long terme, ces retards n'ont qu'un impact mineur sur le déroulement global du projet.

Enfin, les résultats sont satisfaisants. Toutes les exigences du cahier des charges sont présentes. La méthodologie de gestion est restée structurée au cours du projet, même si nous aurions pu améliorer certains points de méthodologie. Sur le plan technique, nous aurions aimé avoir plus de temps pour optimiser la communication.

Certains choix qui ont été faits au cours de l'étude se sont avérés fragiles lors du projet. En effet, nous avons pu nous apercevoir que choisir le Bluetooth comme protocole de communication entre les balises et le téléphone n'était pas forcément le meilleur choix à faire (possibilité de mettre en œuvre le WiFi avec l'Arduino Genuino 101). De même en ce qui concerne les balises. Si nous avions eu plus d'expérience, nous aurions choisi de les mettre en réseau, ce qui aurait grandement optimisé le fonctionnement du système.

Conclusion

Pour conclure, notre projet s'est déroulé en deux axes : **un axe d'étude**, avec l'analyse de l'existant et la conception, ainsi que le cadrage de la preuve de concept. Cette partie se décompose par la prise en main du matériel, la recherche d'une stratégie de localisation. À cela vient s'ajouter **un second axe technique** basé sur la mise en œuvre du système développé autour de la mise en place de la communication en Bluetooth, du traitement des données et de l'expérimentation.

Notre projet a su aboutir sur les conditions initiales du cahier des charges. Cela est en particulier du aux apports théoriques et techniques, mais aussi à la gestion de projet qui a su apporter la méthodologie nécessaire pour mener à terme ce projet dans les temps impartis. Nous avons pu appliquer nos expériences issues des projets antérieurs. Ce travail a été une opportunité pour nous d'appliquer nos compétences dans un projet impliquant plusieurs personnes. Les compétences que nous avons employées s'appuient sur les enseignements des principes fondamentaux des systèmes d'exploitation, du Java mais aussi du traitement de l'information. De même, nous avons dû faire face à de nombreuses difficultés par rapport à android. Néanmoins, nous avons fait en sorte de surpasser ces difficultés, ce qui nous sera bénéfique à l'avenir.

La preuve de concept a été démontrée. Cependant, des extensions sont possibles, comme la mise en place d'une communication WiFi entre les balises et le téléphone ou la création d'un récepteur Bluetooth pour traiter le signal. Cette dernière idée est très ambitieuse et requiert d'importantes connaissances en traitement du signal. Enfin, pour que le produit soit prêt à mettre en service, il faut également penser à trouver une solution de remplacement aux arduinos. En effet, il peut être intéressant de se tourner vers une vraie carte embarquée comme une UDOO.

Avis Personnels

Karim

Personnellement, j'ai trouvé ce projet très intéressant, stimulant et bénéfique pour moi. J'ai beaucoup apprécié mettre en œuvre mes compétences dans un projet concret. En effet, il s'agit de mon premier projet où j'ai eu la réelle opportunité de gérer une équipe dans le cadre de mes études. Je vois réellement ce projet bonne expérience et je suis fier d'avoir pu mettre en œuvre mes compétences de gestion de projet. Je pense que mon investissement a été à la hauteur, ce qui fait que notre projet a bien été mené jusqu'au bout. Nous avons fait tout notre possible pour finir le projet correctement et dans les temps.

Je pense que ce projet m'a notamment permis de consolider mes compétences en Java et en organisation. De même, j'ai pu appliquer des notions vues l'an dernier concrètement. Grâce à cela, j'ai pu acquérir des compétences que je n'avais pas totalement acquises en cours. De plus, mes compétences en gestion de projet s'améliorent de plus en plus et je pense que j'ai su appliquer mon expérience à ce projet, même s'il y a toujours des points à améliorer.

Malgré les bons résultats, nous nous sommes parfois retrouvés en retard par rapport au planning, il faut donc essayer de répartir le travail de manière uniforme sur le planning. Je saurai tirer mes conclusions de cette expérience car c'est un bilan positif. Le grand point à améliorer reste l'estimation de la durée des tâches.

En conclusion, ce projet a été enrichissant, il nous a inculqués de nouvelles compétences et une nouvelle expérience. Ainsi, je saurai donc rester vigilant à l'avenir, pour éviter de faire les mêmes erreurs qui nous ont parfois fait perdre du temps.

Julien

J'ai apprécié travailler avec l'ensemble du groupe. Malgré un début difficile pour fédérer tout le monde, Karim a réussi à prendre en main le projet et à le gérer au mieux. Le rendu final est la preuve que l'ensemble du groupe s'est impliqué afin d'obtenir une application stable et qui répond aux exigences du client.

Ce fut un projet intéressant dans sa technicité. Le Bluetooth Low Energy a été une découverte – plus ou moins heureuse – sur ce que sont les nouveaux moyens de communications d'aujourd'hui. J'ai apprécié travailler sur l'étude de ce moyen de communication qui réserve son lot de surprises lors de sa mise en œuvre et qui nécessite d'être apprivoisé afin d'être maîtrisé.

Je ne pense pas que ce projet m'a appris beaucoup sur la gestion, ou encore sur le travail de groupe en général mais m'a appris à considérer la motivation de chacun sur les tâches qui leur sont assignées et pourquoi pas les aider en cas de besoin. J'ai grâce à cela, été présent dans les différentes phases du projet, de l'analyse fonctionnelle en passant par la mise en œuvre du BLE pour finir sur les tests finaux de l'application, tout en m'intéressant de temps en temps à la gestion du projet (sans y apporter grande aide)

Ce projet est pour moi un projet réussi, avec une deadline respectée et des exigences satisfaites.

Brandon

Un projet qui a su me réconcilier avec Android en ayant un point différent, plus axé design et intégration des différentes fonctionnalités constituant notre application. Ce projet a su me donner de la motivation pour essayer de fournir une application mélangeant un design moderne et fonctionnel.

L'utilisation d'un projet git m'a permis de continuer à me familiariser avec ce logiciel, ainsi que de partager mon intérêt à ce dernier. Il est toujours aussi important pour un développement plus rapide et facile et pour une intégration finale. Notamment lorsque l'on se trouve avec plusieurs personnes potentielles qui peuvent modifier le programme de l'application.

Quentin

Le projet que j'ai pu partager avec mes collègues, a été une très bonne aventure autant sur le plan technique où j'ai pu transmettre certaines connaissances mais aussi apprendre d'eux, que sur le plan organisationnel où nous avons pu élire un chef de projet qui a su nous motiver et mener le projet à son terme, mais aussi à travailler ensemble sur différentes tâches et à communiquer entre nous.

De mon côté je ressors avec un très bon sentiment comme notamment le travail avec les différents membres de l'équipe qui m'a permis de m'améliorer sur le plan technique comme par exemple la création d'IHM sur Android, mais aussi sur certains points particuliers de Git. De plus, l'esprit d'équipe et de cohésion qui était présent tout au long du projet, nous a beaucoup aidés et plus particulièrement lorsque nous avons rencontré des problèmes comme par exemple, au début du projet où nous ne pouvions pas utiliser les beacons. Nous nous sommes tous ensemble réunis afin de chercher des solutions pour accomplir un but commun et ainsi faire avancer le projet.

Cependant j'ai aussi certains points que je souhaite améliorer autant sur ma façon de travailler que personnels avec notamment certains moments où je ne me suis pas libéré assez de temps pour travailler sur le projet, ce qui a ralenti ce dernier mais aussi mon équipe. De même, le travail en collaboration, au début du projet, n'était pas mon point fort, mais j'ai pu améliorer cette compétence au fur et à mesure. Néanmoins, je suis conscient de ces points faibles et je compte travailler sur ceux-là afin de pouvoir devenir un professionnel accompli.

Damien

Dans un premier temps, je pense que le projet est une réussite surtout du point de vue humain. En effet, nous avons réussi à mener à bien un projet avec une charge de travail conséquente en gardant l'ensemble de l'équipe impliquée et en utilisant au mieux les capacités de chacun.

Dans un second temps, je pense que d'un point de vue personnel ce projet m'a beaucoup apporté sur des compétences qui me faisait défauts. J'ai en effet lors de ce projet réalisé un bon nombre d'études et de recherches, ce qui m'a donné de nouvelles capacités pour appréhender ce type de tâche.

Axel

Lors de ce projet collectif, j'ai découvert une nouvelle manière d'appréhender le travail d'équipe sur un projet. Les diverses méthodes et outils de gestion de projet que nous avons utilisé pendant notre travail nous ont permis d'être plus efficace dans la répartition des tâches et dans nos décisions.

D'un point de vue technique ce projet m'a permis d'aborder deux notions que je n'avais jamais mises en œuvre : La communication sans fil (Bluetooth) et Android Studio. Ce projet m'a donné l'occasion de prendre en main et d'approfondir ces deux notions.

Je suis également fier du résultat fourni collectivement par notre groupe : l'application développée et les études menées tout au long du projet sont complètes et me semble en accord avec le travail demandé sur le projet.

Table des illustrations

Figure 1 : Beacons	8
Figure 2 : Objectifs du projet.....	11
Figure 3 : Liste des tâches	13
Figure 4 : MOSCOW	14
Figure 5 : SDP.....	14
Figure 6 : Carte mentale générale	15
Figure 7 : Carte mentale définir le projet	16
Figure 8 : Carte mentale concevoir / développer l'application	17
Figure 9 : Carte mentale définir les jalons	18
Figure 10 : Planning prévisionnel.....	18
Figure 11 : Planning prévisionnel phase de spécification	19
Figure 12 : Planning prévisionnel phase de développement.....	19
Figure 13 : Bilan des Jalons.....	20
Figure 14 : Matrice SWOT	21
Figure 15 : Identification des risques (1).....	22
Figure 16 : Identification des risques (2).....	23
Figure 17 : Cycle en V	24
Figure 18 : La gestion de projet.....	25
Figure 19 : Méthode Scrum	26
Figure 20 : Répartition des tâches	27
Figure 22 : MS-Project.....	27
Figure 21 : Excel.....	27
Figure 23 : Tableau Trello	28
Figure 24 : Discord	28
Figure 25 : Création d'un salon vocal	29
Figure 26 : Serveur Discord	29
Figure 27 : Caractéristiques d'un beacon.....	31
Figure 28 : Bête à cornes	32
Figure 29 : Beacon Gimbal	33
Figure 30 : Comparatif constructeur	33
Figure 31 : Veille technique	35

Figure 32 : iBeacon	36
Figure 33 : Eddystone	36
Figure 34 : Beacon Gimbal & Application de configuration	37
Figure 35 : Caractéristiques Gimbal	38
Figure 36 : RadBeacon	38
Figure 38 : Application de configuration RadBeacon	39
Figure 37 : Caractéristiques RadBeacon	39
Figure 39 : Caractéristiques Kontakt	40
Figure 40 : Beacons Konkact.io & Application de configuration	40
Figure 41 : Gateway Konkact	41
Figure 42 : BNB beacons & Application de configuration	42
Figure 43 : Caractéristiques BNB	42
Figure 44 : Puissance reçue en fonction de la distance	43
Figure 45 : Relevés de puissance en fonction de la puissance émise et de la distance	43
Figure 46 : Configurations Gimbal (1)	44
Figure 47 : Configurations Gimbal (2)	45
Figure 48 : Configurations Gimbal (3)	45
Figure 49 : Beacons BNB	46
Figure 50 : Configurations BNB (1)	46
Figure 51 : Beacon BNB	47
Figure 52 : Configurations BNB (2)	47
Figure 53 : Configurations BNB (3)	48
Figure 54 : Calibration RSSI (1)	48
Figure 55 : Calibration RSSI (2)	49
Figure 56 : Chaud / Froid	50
Figure 57 : Historisation des données	50
Figure 58 : Effet Doppler	51
Figure 59 : Triangulation	52
Figure 60 : Triangulation	52
Figure 61 : Architecture matérielle	53
Figure 62 : Langage de programmation	55
Figure 63 : Raspberry PI	56
Figure 64 : Arduino Genuino 101	56

Figure 65 : Fonction de calcul de la distance.....	57
Figure 66 : Diagramme de séquence chaud-froid.....	57
Figure 67 : AltBeacon	58
Figure 68 : Calcul de la distance en fonction du RSSI	58
Figure 69 : Croquis chaud-froid.....	59
Figure 70 : Prise en main de l'API AltBeacon	60
Figure 71 : Calcul de la distance.....	60
Figure 72 : IHM Calibration	61
Figure 73 : IHM Chaud-Froid	62
Figure 74 : Fonction de calcul de la position.....	63
Figure 75 : Diagramme des cas d'utilisation	64
Figure 76 : Diagramme de séquence « localiser un beacon »	65
Figure 77 : Diagramme de séquence « localiser un beacon avec erreur »	65
Figure 78 : Diagramme de séquence « localiser plusieurs beacons »	66
Figure 79 : Diagramme de séquence « administration du système ».....	66
Figure 80 : Diagramme de séquence « suppression d'une balise »	67
Figure 81 : Diagramme de séquence « reconfiguration d'une balise »	67
Figure 82 : Diagramme des classes	68
Figure 83 : Envoi en broadcast	70
Figure 84 : Connexion à un périphérique (échanges de caractéristiques)	71
Figure 85 : Architecture mise en œuvre.....	72
Figure 86 : Structure de données	73
Figure 87 : Tableau de filtrage.....	74
Figure 88 : Visio	75
Figure 89 : IHM Admin	75
Figure 89 : IHM Login	76
Figure 90 : IHM Navigation	76
Figure 91 : IHM Game Menu	77
Figure 92 : IHM Sélectionner un beacon	78
Figure 93 : IHM Sélectionner plusieurs beacons.....	79
Figure 94 : Principe de fonctionnement du BLE (1)	80
Figure 95 : Principe de fonctionnement du BLE (2)	80
Figure 96 : Extrait de code arduino (1).....	81

Figure 97 : Extrait de code arduino (2).....	81
Figure 98 : Résultat du scan Arduino	82
Figure 99 : Format de trame iBeacon.....	82
Figure 100 : Déclaration du service et de la caractéristique.....	82
Figure 101 : Extrait de code arduino (3)	83
Figure 102 : Mise en forme des données	83
Figure 103 : Connexion aux balises	84
Figure 104 : Protocole de communication GATT	84
Figure 105 : Machine à états	85
Figure 106 : Fonction startScanning()	86
Figure 107 : Fonction doConnect()	86
Figure 108 : Fonction discoverServices()	86
Figure 109 : Fonction getCharacteristic()	87
Figure 110 : Fonction setDescriptor()	87
Figure 111 : Fonction getData()	88
Figure 112 : Calcul du RSSI	88
Figure 113 : Trilateration	89
Figure 114 : Librairie trilateration (1)	89
Figure 115 : Librairie trilateration (2)	89
Figure 116 : Tableau résultant.....	90
Figure 117 : Développement du lissage de données	90
Figure 118 : Formules de mise à l'échelle.....	90
Figure 119 : Fonction settingScale().....	91
Figure 120 : Interface administrateur	92
Figure 121 : Interface « localisation »	93
Figure 122 : Boîte de dialogue «sélection des beacons »	94
Figure 123 : Nouveau diagramme des classes	95
Figure 124 : Threads.....	96
Figure 125 : Atténuation du signal des beacons	97
Figure 126 : Atténuation du signal (avec obstacle)	98
Figure 127 : Relevés Arduino (1)	99
Figure 128 : Portée d'une liaison Bluetooth	99
Figure 129 : Relevés Arduino (2)	100

Figure 130 : Salle Lovelace	100
Figure 131 : Croquis Salle Lovelace.....	101
Figure 132 : Solution sans gateway (kontakt).....	104
Figure 133 : Solution avec gateway (kontakt).....	104
Figure 134 : Calcul de la distance (defacto).....	105
Figure 135 : Atténuation du signal dans une pièce	105
Figure 136 : Exemple d'une carte d'erreur	106
Figure 137 : Estimation de la distance	106
Figure 138 : Carte d'erreur après traitements.....	107
Figure 139 : Serveur GIT du projet	108
Figure 140 : Application finale	109
Figure 141 : Planning Prévisionnel / Réel	112

Annexes

Annexe 1 : Bilan financier

N° nom.	Références	Devis	Désignation produits, prestations	N° inventaire	Montant unitaire H.T.	Quantité	Montant H.T.	Montant T.V.A.	Montant T.T.C.
	96527	BNB BEACONS	Pack 3 X « iBeacon/Eddystone Plus		90,00 €	1	90,00 €	18,00 €	108,00 €
	2520713	FARNELL	Arduinos GENUINO 101		29,00 €	4	116,00 €	23,20 €	139,20 €
Total de la commande									
					206,00 €		41,20 €		247,20 €

Bilan Financier : Ressources Humaines				Brut Ho- raire	Heu res	Salaire Brut
PRESTATION MAIN D'ŒUVRE				8,17 €	350	2859,5 €

Annexe 2 : Configuration des Beacons Gimbal et programmation d'une application

Afin de configurer les Beacons deux outils nous sont indispensables à savoir :

- Gimbal Beacon Manager
- Application et portail web Gimbal permettant l'insertion d'une configuration
- Android Studio afin de programmer sur les smartphones Android

Nous parlerons dans un premier temps de Gimbal Beacon Manager qui est présent sous forme d'application Android/iOS et de portail web permettant l'insertion des configurations dans les Beacons, puis dans un second temps d'Android Studio et de la programmation d'une application qui interagit avec les Beacons.

7.1.1 Gimbal Beacon Manager

7.1.1.1 Observation

Une observation est la réception d'un packet BLE (Bluetooth Low Energy) et celles-ci apparaissent par l'intermédiaire du SDK (Software Development Kit) qui permet de récupérer certaines informations de ce signal comme la valeur du **RSSI (Received Signal Strength Indication)**, le nom du Beacon, l'icône, le niveau de la batterie ainsi que la température. Ces observations sont temps-réel du côté client mais il est aussi possible de recevoir des notifications sur un serveur, afin de procéder à des statistiques par exemple mais ces notifications ne seront pas en temps-réels.

7.1.1.2 Système de visite

Gimbal nous offre un exemple de visite qui permet de montrer et d'expliquer le fonctionnement des Beacons. Un système de visite est l'une des utilisations les plus communes des Beacons et ce système permet de :

- Savoir si un utilisateur est proche d'une localisation,
- Combien de temps l'utilisateur était à une localisation.

Notre système possède donc 3 événements qui vont décrire un utilisateur et une localisation :

- L'arrivée à une localisation,
- Observé (toujours sur place),
- Le départ d'une localisation.

Voici la description d'un utilisateur qui arrive à une localisation :

- Un utilisateur arrive à une localisation et il déclenche donc, sur l'application l'évènement d'arrivé. Cela peut ainsi permettre faire une demande de conseiller ou afficher les informations d'un produit en particulier, etc...
- L'utilisateur est toujours en place et les différentes publicités sont diffusées sur son téléphone ou encore la commande de l'utilisateur se met à jour s'il met des produits dans son panier.
- L'utilisateur part de la localisation et l'application passe en mode globale, affichage de la carte du magasin.

L'arrivée, « l'attente » et le départ d'une zone sont 3 évènements qui sont conditionnés par 3 valeurs à savoir :

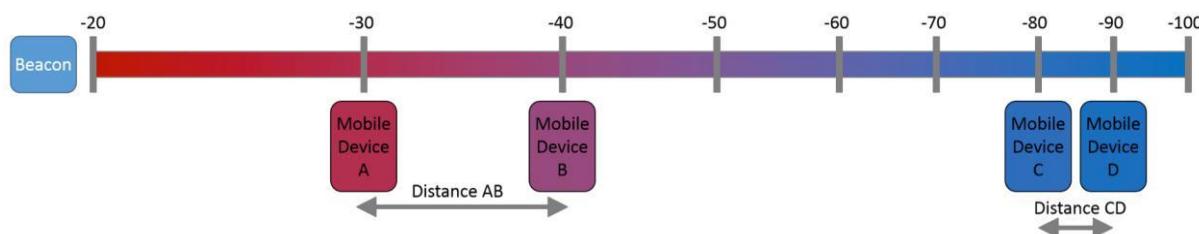
- « Arrival RSSI », valeur qui permettra de savoir si un utilisateur est arrivé sachant que sa valeur de RSSI passera de « +Infini » à la valeur de « Arrival RSSI », lorsque cette valeur sera franchie en partant de l'infini alors l'utilisateur sera identifié comme arrivé à une localisation.
- « Departure RSSI », valeur qui permettra de savoir si un utilisateur part de la zone sachant de celle-ci passera de « 0 » à la valeur de « Departure RSSI », lorsque cette valeur sera dépassée l'utilisateur sera considéré comme parti.
- « Departure Interval », timer permettant de savoir combien de temps un utilisateur est hors de portée dans le but de le considérer comme parti.

7.1.1.3 Comprendre la valeur du RSSI

Le RSSI est une mesure en Decibel-milliwatts permettant de connaître la distance la distance entre un Beacon et un téléphone ou même entre plusieurs téléphones.

Attention, la mesure étant réalisée en DBm, l'échelle est donc logarithme, comme on peut le voir la différence de distance

Aussi la distance que l'on pourrait avoir sera imprécise et sera donc à nuancer.



7.1.1.4 Rules

Permet la notification quand un Beacon est observé par un téléphone au serveur Gimbal qui pourra ensuite notifier un serveur.

7.1.1.5 Beacon Sharing

Permet le partage de Beacons avec d'autres développeurs mais cette fonctionnalité n'est pas intéressante pour nous.

7.1.1.6 Beacon Configuration

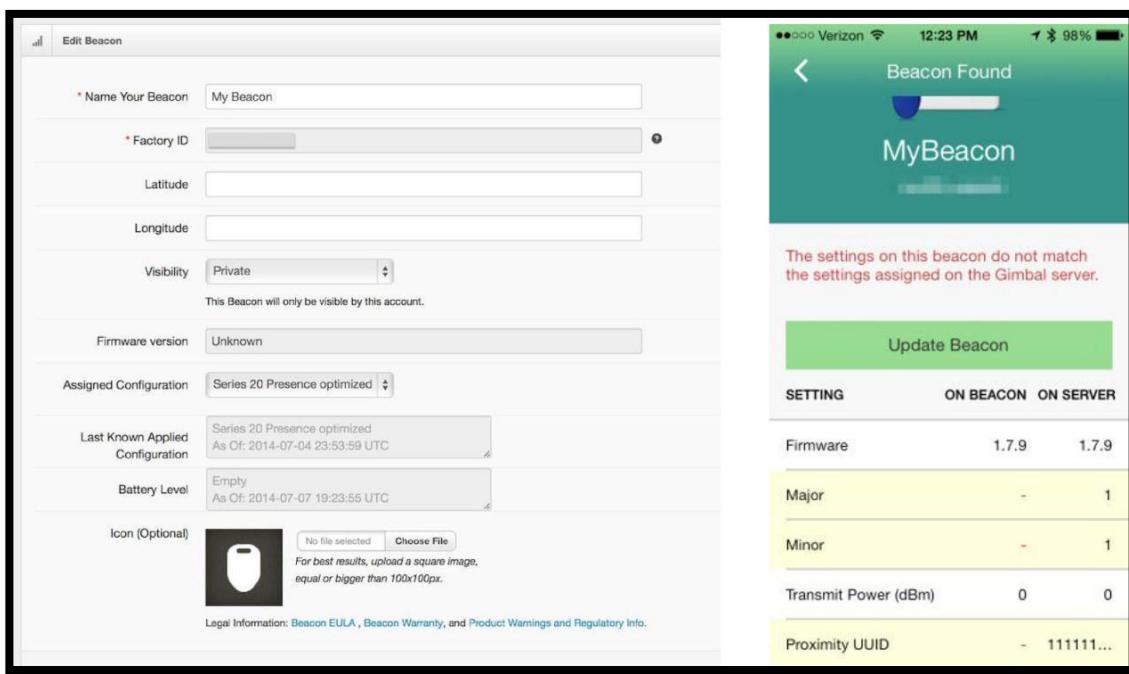
La configuration des Beacons permet la personnalisation des caractéristiques hardware des Beacons comme par exemple la puissance de transmission, le nombre d'émission de signaux par minutes, la visibilité des Beacons (par le compte utilisateur ou tous les développeurs, sa position géographique, son icône et son nom).

Par exemple, ajuster la puissance de transmission permet d'avoir une meilleure distance de détection mais diminue l'autonomie du Beacon.

Il existe deux configurations de base :

- Recommandé : 3 signaux par minutes | Utilisation pour des applications où la zone des Beacons est critiques
- Optimisation de la présence : 1 signal par minutes | Sauvegarde de la batterie & Utilisation quand le téléphone est proche du Beacon.

Voici l'écran d'édition d'une configuration de Beacon qui se déroule sur le portail web de Gimbal et l'écran de mise à jour de la configuration dans l'application Gimbal (Android et iOS) :



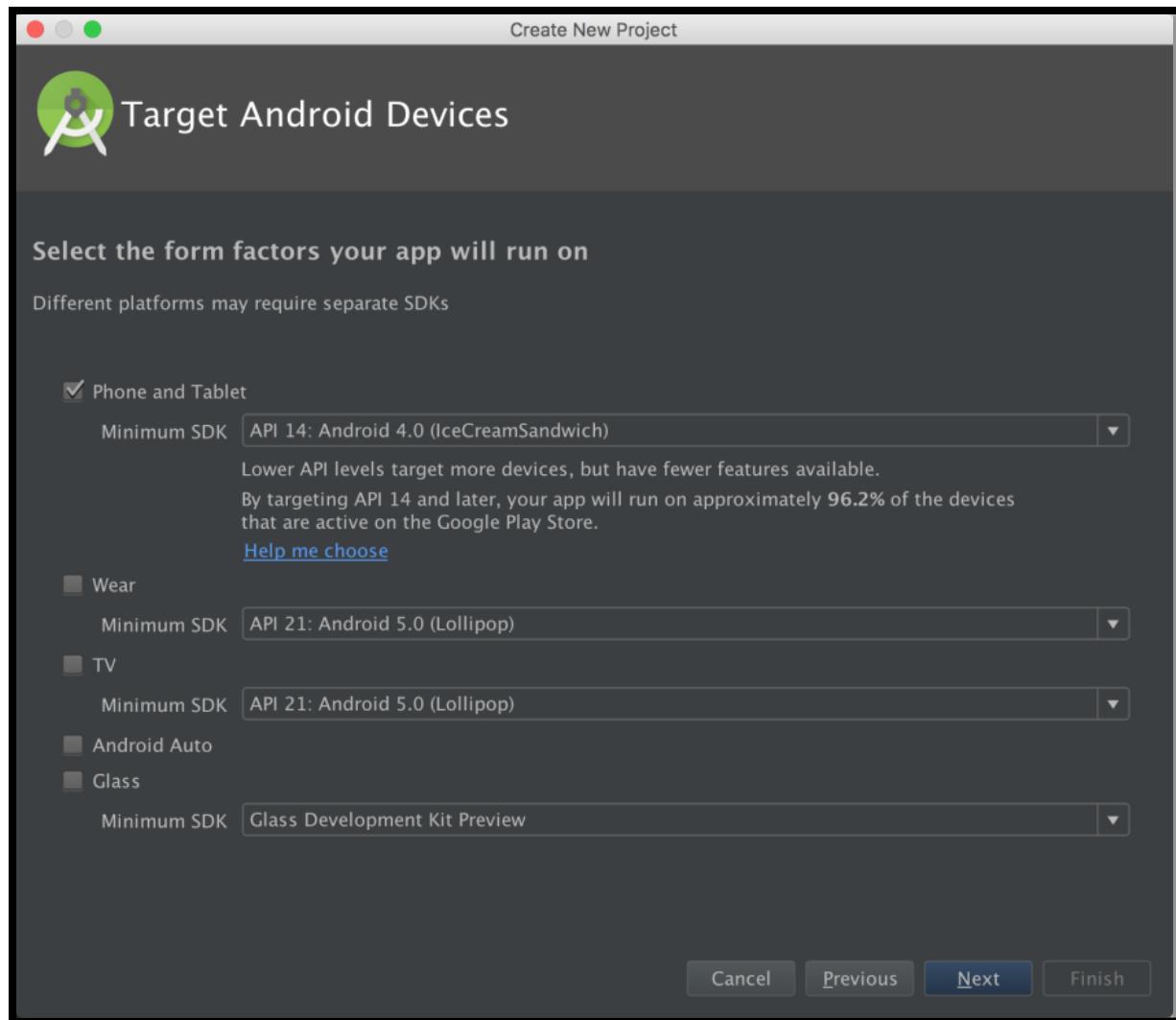
Passons maintenant à la programmation d'une application Android avec le SDK de Gimbal pour détecter et parser les trames BLE provenant des Beacons.

7.1.2 Programmation

Afin d'avoir accès au SDK de Gimbal, il faut créer un compte développeur sur le site de Gimbal. Ensuite il faut se munir d'Android Studio qui est l'IDE développé par Google intégrant tous les outils pour créer des applications Android.

7.1.2.1 Création d'un projet sur Android Studio

Afin de débuter la création d'une application Android, il faut créer un nouveau projet et choisir le type de plateforme (pour nous 'Phone and Tablet') et choisir l'API qui nous convient (ceci conditionnera la version minimum d'Android afin de pouvoir lancer l'application).



Ensuite afin d'inclure le SDK Gimbal, il va faut rajouter ce dernier en tant que dépendance au projet avec les lignes suivantes dans le fichier build.gradle.

```
repositories {  
    ...  
    mavenCentral()  
    ...  
}
```

Et..

```
dependencies {  
    ...  
    compile 'org.slf4j:slf4j-api:1.7.13'  
    compile group: 'com.gimbal.android.v2', name: 'gimbal-sdk', version: '+'  
    compile group: 'com.gimbal.android.v2', name: 'gimbal-slf4j-impl', version  
: '+'  
    ...  
}
```

Afin que notre application puisse avoir accès au Bluetooth, Internet, la localisation, l'état du Wifi, etc.. il est nécessaire de mettre à jour le fichier Manifest qui répertorie les permissions de l'application. Il faut également rajouter les lignes suivantes dans le fichier AndroidManifest.xml.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Gimbal fournit un service qui va permettre de détecter et décoder les paquets BLE et nous renvoyer au final les informations comme l'icône, le nom du Beacon, etc...

Pour que celui-ci soit actif il faut le lancer en tâche de fond. Pour effectuer ceci, rajouter les lignes suivantes dans le fichier AndroidManifest.xml.

```
<service
    android:name="com.gimbal.internal.service.GimbalService"
    android:exported="false" >
    <intent-filter>
        <action android:name="com.gimbal.android.sample.service.GIMBAL_SERVICE" />
    </intent-filter>
</service>

<receiver
    android:name="com.gimbal.internal.service.GimbalServiceStartStopReceiver"
    android:enabled="true" >
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.ACTION_SHUTDOWN" />
    </intent-filter>
</receiver>
```

Annexe 3 : Shared Preferences

7.1.3 Fonctionnement

Les SharedPreferences font partie d'un des moyens proposés par la plateforme Android pour stocker de façon persistante des données dans une application. Sous forme de clés-valeurs, il est ainsi possible de récupérer les valeurs stockées, de contrôler l'existence d'une clé mais aussi de pouvoir modifier la valeur d'une clé.

Il est aussi possible de gérer la portée de nos SharedPreferences avec deux modes : « private » ou « world_readable » ou encore « world_writable », il faut donc faire attention à la portée que l'on utilise pour stocker nos données ainsi qu'aux conséquences.

7.1.4 Intégration

7.1.4.1 Lecture

Pour mettre en place ce système dans une activité il faut dans un premier temps instancier la classe SharedPreferences comme ci-dessous :

```
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
```

Ensuite pour récupérer la valeur d'une clé, nous avons seulement besoin de d'appeler l'une des méthodes suivantes de l'objet que nous venons de créer :

- getAll
- getBoolean
- getFloat
- getInt
- getString
- getStringSet

Sachant que la méthode utilisée correspond au type de données que l'on souhaite lire.

7.1.4.2 Ecriture

Pour insérer de nouvelles données, il faut d'abord créer un objet de type « SharedPreference.Editor » comme ceci :

```
SharedPreferences.Editor editor = preferences.edit();
```

Ensuite il suffit d'utiliser, tout comme pour la lecture, une des méthodes suivantes :

- putBoolean
- putFloat
- putInt
- putString
- putStringSet

Sachant que la méthode utilisée correspond au type de données que l'on souhaite ajouter. Après avoir rajouté vos données, il faut « commiter » les modifications afin que cela soit pris en compte.

```
editor.putString("fav_color", "FFABB4"); // On ajoute la clé « fav_color » avec la valeur « FFABB4 »
```

```
editor.commit(); // On soumet les modifications
```

Annexe 4 : GITHUB

7.1.5 Présentation de Git



Git est un logiciel de gestion de versions décentralisé. Il est open source et a été créé aux alentours de 2005 par Linus Torvalds, développeur du noyau Linux, tout d'abord pour son usage personnel car il n'aimait pas les solutions déjà existantes comme CVS ou SVN.

Git est extrêmement simple d'utilisation, notamment couplé à une interface graphique comme GitHub, il est rapide, gratuit, mais surtout il est possible de travailler hors-ligne puisqu'il n'est pas nécessaire d'avoir un serveur distant pour fonctionner.

7.1.5.1 Réaliser un Commit

Lorsqu'on a réalisé une modification dans notre code, il faut publier cette nouvelle version (réaliser un commit). Un commit doit obligatoirement être accompagné d'une courte description (de préférence en anglais et en quelques mots).

7.1.5.2 Principe des branches

La création d'une nouvelle branche est particulièrement utile lorsqu'on doit avoir plusieurs versions d'une même application ou lorsqu'on souhaite développer une nouvelle fonctionnalité nécessitant des changements majeurs dans le code. Dans notre cas, nous avons défini plusieurs fonctionnalités différentes sur l'application indépendantes entre eux, une branche est donc intéressante pour chaque fonctionnalité, afin que chaque personne travaillant sur le projet avance à son rythme et n'affecte pas les autres utilisateurs.

7.1.5.3 Résoudre un conflit

Il est possible que l'une de nos modifications entre en conflit avec les modifications d'un autre utilisateur et que Git n'arrive pas à fusionner automatiquement vos modifications. Dans ce cas, il nous sera demandé de corriger manuellement le conflit en allant, éditer le(s) fichier(s) concerné(s).

7.1.5.4 Les dépôts distants

La particularité d'un dépôt distant est d'être situé dans le cloud (ou dans un Intranet) et il est donc accessible par tous les utilisateurs à n'importe quel moment. Pour accéder à un dépôt, l'utilisateur doit le « cloner » localement, afin de travailler en local (ce qui inclut la possibilité de travailler en hors-ligne).

7.1.5.5 Utilisation de Git

Après avoir installé Git, il est possible d'utiliser Git en ligne de commande. Cependant cette utilisation peut être relativement compliquée et peu intuitive. Il est préférable une utilisation d'une interface graphique comme GitKraken.

7.1.6 Présentation de GitHub



GitHub est une communauté de millions d'utilisateurs (pas moins de 11 millions). Il y a une multitude de projets qui ont été créé sur la plateforme portant le même nom. Elle permet de faciliter la collaboration de développeurs du monde entier sur un même projet. Pour notre cas, nous avons dû rajouter chacun d'entre nous en tant que collaborateur afin qu'il soit possible de travailler sur notre dépôt distant.

C'est aussi une société d'hébergement, qui propose des solutions de gestion de développement de logiciels, basées sur le protocole Git.

L'utilisation de GitHub est entièrement gratuite dans le cadre d'un développement open source.

GitHub a également développé un logiciel appelé « GitHub Desktop », qui permet de gérer son projet sous Git et d'utiliser toutes les fonctionnalités de ce dernier via une interface graphique.

GitHub est très populaire au sein de plusieurs entreprises/sociétés, même de grande envergure, tel que Microsoft, Twitter, Facebook ou encore Netflix parmi tant d'autres.

7.1.7 Présentation du logiciel GitKraken



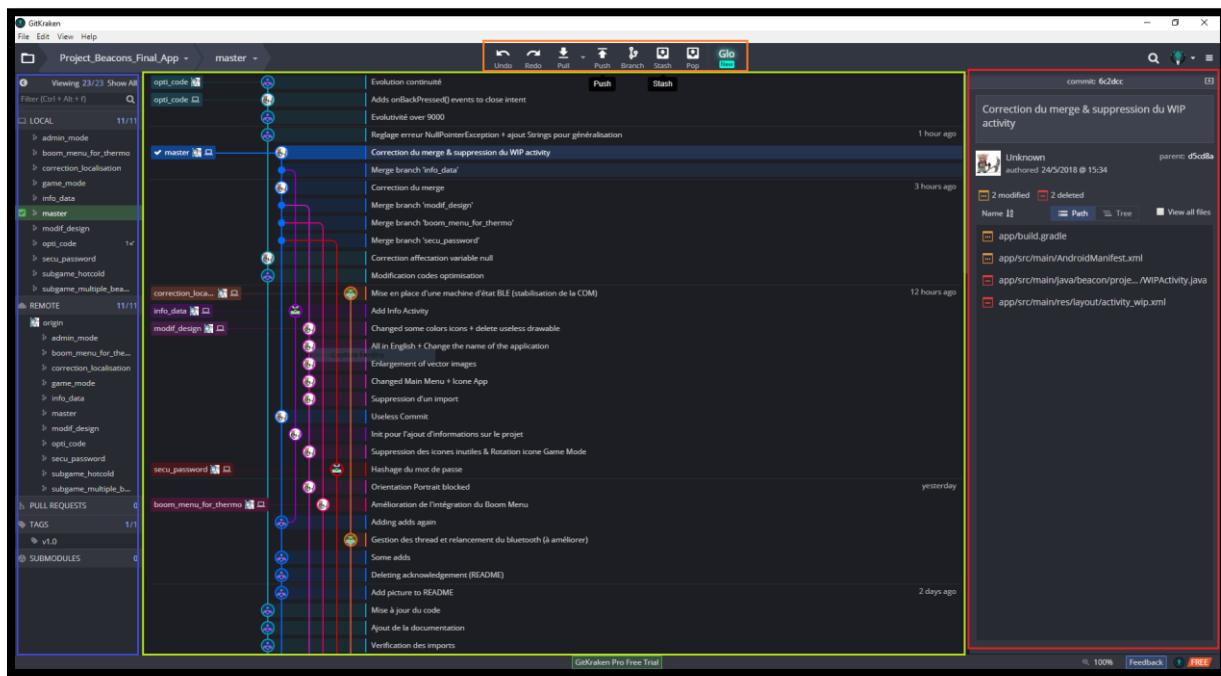
GitKraken est un logiciel créé par Axosoft en 2014, il est donc très récent et sa version 1.0 n'est disponible que depuis mai 2016. C'est un logiciel qui permet de grandement simplifier l'utilisation des logiciels de versionning en particulier Git.

Les avantages de GitKraken sont nombreux, il y a une interface simple d'utilisation et plaisir à utiliser, la possibilité de gérer plusieurs dépôts distants, une parfaite comptabilité sur Windows, Mac et Linux, la gestion de plusieurs profils et il est gratuit pour la version de base.

Cependant un de ses défauts est qu'on est limité si l'on souhaite par exemple héberger son projet sur son propre serveur. On ne peut avoir qu'un seul profil sur cette version. Sur la version payante on peut aussi gérer directement les conflits sous Git-Kraken. Voici les différentes versions de GitKraken :

Version Gratuite	Version Pro	Version Entreprise
Gratuit	\$49 / utilisateur / an \$39 si plus de 10 utilisateurs \$29 si plus de 100 utilisateurs	\$99 / utilisateur / an \$79 si plus de 10 utilisateurs \$69 si plus de 100 utilisateurs
Version de base	Gestion des conflits de code dans l'application Gestion de plusieurs utilisateurs Rajout du GitHub d'entreprise Support courriel	Protégé par un pare-feu Installation sur vos serveurs Gestion en local Suivi informatique complet

7.1.8 Mise en œuvre de GitKraken



La partie bleue présente sur la capture d'écran juste au-dessus, possède une barre de recherche qui permet de rechercher parmi toutes les branches une étape en particulier. On peut aussi voir le nombre de branches en local, ainsi que celles présentes sur le dépôt distant. Il y a aussi les tags associés à une étape importante du projet, ici nous avons sur cette capture qu'un seul tag « v1.0 » qui représente la toute première version finale fonctionnelle, ce tag est présent en local et sur le dépôt.

La partie centrale en vert est composée d'un « graphique » représentant tous les commits qui ont été réalisés sur les différentes branches, par qui le programme a été push, quand et sur quelle branche.

Pour chaque commit, on peut voir tous les détails (par qui, quand, titre, description...). Tous les fichiers modifiés sont indiqués en dessous de tout cela dans la partie en rouge. Il est possible d'avoir le détail exact pour chaque fichier en cliquant dessus mais aussi de passer en mode « arbre » pour voir l'arborescence du projet.

La dernière partie en orange située en haut de l'interface, contient tous les boutons d'actions de GitKraken, des boutons annuler et refaire sont présents, mais aussi un bouton pull qui permet de récupérer la dernière version disponible sur le dépôt distant. Le bouton push permet d'envoyer nos commits d'une branche spécifique au dépôt afin de le mettre à jour. Quant au bouton Branch il permet de créer une branche à l'endroit sélectionner afin d'éviter d'impacter la branche principale souvent nommée « master ». Les deux derniers boutons permettent le remisage de sa branche, nous n'avons pas eu l'occasion d'utiliser ces boutons au cours du projet. Pour finir le dernier bouton est une toute nouvelle fonctionnalité qui permet d'utiliser les fonctionnalités de Trello, qui est un gestionnaire de tâche pour la gestion de projet.