INGENIEUR EN SCIENCES INFORMATIQUES

RAPPORT DE STAGE DE QUATRIEME ANNEE

Implémentation d'une application proposant des informations sur les lieux à proximité pour terminal mobile de marque Apple

Stagiaire : LUCAS SOUMILLE (SI4)

Maître de stage : Alexandre CATHALAN

ENTREPRISE : FRONTWARE INTERNATIONAL (Bangkok)

01 juillet 2016 – 31 août 2016

Remerciements

Je tiens à remercier toutes les personnes qui m'ont permis de réaliser ce stage et qui ont contribué à son bon déroulement.

Tout d'abord, j'adresse mes remerciements au PDG de l'entreprise Frontware International, Mr Eric FAIRON pour m'avoir accueilli dans l'entreprise et pour avoir facilité mon intégration.

Ensuite, je remercie vivement Mr Kriangchai PONGKITTISOPA pour ses conseils et son aide qui ont été importants à la réalisation du projet

Enfin, Je remercie les développeurs de l'entreprise en particulier Mr Yann SIMON et Mr Alexandre CATHALAN pour avoir répondu à mes questions et pour avoir partagé leur expérience au quotidien.

Plan du rapport

1. INTRODUCTION

J'ai réalisé mon stage d'une durée de sept semaines au sein de l'entreprise Frontware International qui est située dans la capitale Thaïlandaise, à Bangkok. Dans cette introduction, je vais présenter le cadre de travail qui a été le mien pendant ces deux mois, puis expliciter ma mission ainsi que les raisons pour lesquels j'ai choisi ce sujet. Enfin j'introduirai les autres parties du rapport.

1.1. Cadre de travail

1.1.1. Lieu du stage

Les locaux de l'entreprise sont situés dans un quartier un peu excentré de Bangkok (Figure1), éloigné de l'agitation du centre-ville.

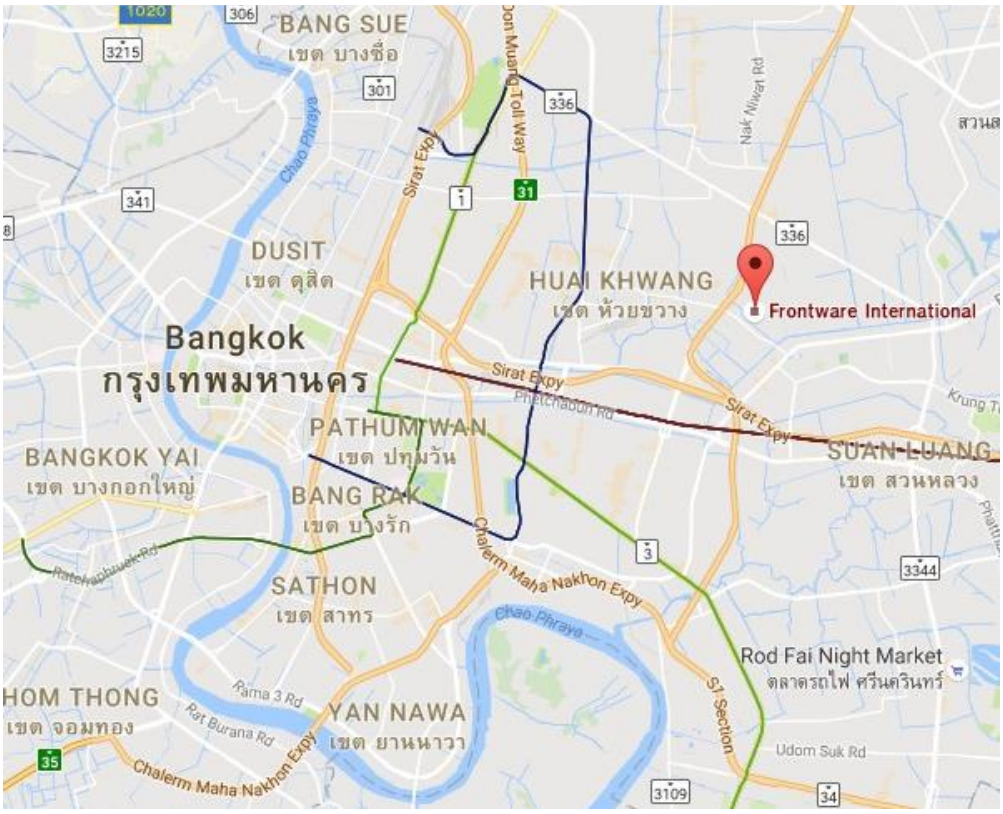

Figure 1 Carte de Bangkok

Au sein du bâtiment, les salariés sont répartis sur deux étages. Au rez-de-chaussée, nous retrouvons l'administrateur système, les testeurs ainsi que les commerciaux de l'entreprise. Enfin l'autre étage est celui réservé aux développeurs. L'entreprise contient sept développeurs. C'est dans cet espace que place m'a été réservée.

1.1.2. Station de travail

Mon poste était équipé d'un ordinateur de bureau avec deux écrans. Le système d'exploitation de ce dernier était un Ubuntu. J'avais aussi à ma disposition un Mac Mini sur lequel je me connectais à distance pour pouvoir travailler et un iPhone afin de pouvoir tester l'application.

1.2. Sujet du stage

Depuis une année, une partie des développeurs de l'entreprise travaille sur un projet nommé « Nearby ». L'application mobile pour Android est déjà disponible sur le magasin d'applications de Google. Mon sujet de stage était dans l'énoncé plutôt simple, continuer le développement de l'application iOS (pour iPhone) et obtenir la même expérience pour l'utilisateur c'est-à-dire avoir les mêmes fonctionnalités. Le côté serveur était déjà implémenté ainsi que la partie présentation l'application, la version iOS devait donc pouvoir s'intégrer avec les entités déjà présentes. Pour ce stage les principaux challenges étaient pour moi d'apprendre à travailler dans un nouvel environnement par conséquent d'apprendre de nouvelles connaissances autour de l'univers Apple et aussi de saisir les différences de points de vue entre Google et Apple.

1.3. Contenu du rapport

La suite de ce rapport sera découpée en deux axes. Dans un premier temps, je vais décrire plus précisément le fonctionnement de l'application et les objectifs que l'on m'a donné ou que je me suis fixé avant de commencer le développement. Ensuite dans un second temps, je détaillerai les différents problèmes que j'ai pu rencontrer et comment j'ai essayé de les résoudre, tout en donnant un planning global de ces sept semaines.

Enfin la conclusion du rapport sera l'occasion de prendre plus de recul afin d'analyser les compétences que j'ai développé et aussi ce que j'aurais pu mieux appréhender.

2. DESCRIPTION DU TRAVAIL PROPOSE

La mission réalisée durant ce stage avait pour but de s'intégrer dans le projet « Nearby ». Je vais commencer cette partie en précisant davantage les spécificités de ce dernier puis ensuite je détaillerai les différentes parties de ma mission, données au début du stage.

2.1. Description globale du projet

« Nearby » est une application qui permet à des magasins, restaurants, musées, aéroport ou tout autre type d'entreprise de diffuser des informations à tous les utilisateurs proches utilisant la technologie Beacon. De plus, ces informations peuvent être adaptées en fonction de chaque utilisateur selon ses goûts grâce aux statistiques collectées et traitées par l'application.

Le projet possède deux types de clients. Les premiers sont les utilisateurs de l'application qui ont installé l'application sur le smartphone et sont à la recherche d'informations et de bons plans en fonction de leur position géographique. Nous retrouvons ensuite les propriétaires d'entreprise qui ont souscrit à « Nearby » et ont installé des Beacons dans leur magasin pour attirer les utilisateurs de l'application mobile, par la suite je nommerai ces commerces localisation « Nearby ».

2.1.1. Principales fonctionnalités de l'application

Dans ce paragraphe, je vais détailler les fonctionnalités qui sont proposées sur l'application mobile.

La fonctionnalité principale est la réception d'information en fonction des commerces voisins, pour cela dès que l'application a été installée, elle traque la position de l'utilisateur et le notifie dès qu'il s'approche d'une localisation « Nearby ». S'il ouvre l'application proche d'une de ces dernières, l'utilisateur peut voir une page d'accueil propre à la localisation. S'il entre dans le commerce l'utilisation du Bluetooth permettra de détecter les Beacons présents et obtient des informations différentes en fonction de sa position dans le magasin. Par exemple, un Beacon pourra être positionné dans chaque section d'un musée et dès qu'un utilisateur entrera dans une nouvelle section il recevra les informations correspondantes.

Aussi, l'installation de l'application sur son smartphone permet d'obtenir un accès wifi gratuit dès que l'on se trouve à proximité d'une localisation « Nearby ».

Enfin, « Nearby » possède une partie sociale afin de pouvoir partager ses lieux préférés ou de prévenir ses amis d'une offre intéressante dans un commerce.

2.1.2. Interface de gestion pour les propriétaires d'entreprises

Dès qu'un propriétaire a souscrit à « Nearby », ils déploient les Beacons dans son établissement. Ensuite, il peut accéder à un espace personnel sur le portail web « Nearby » où il pourra alors configurer chacune des pages affichées par ses Beacons, des modèles lui sont mêmes proposés ou encore la page d'accueil dès qu'un utilisateur arrive dans son commerce.

Depuis cette page web, il sera tout à fait possible de créer une offre promotionnelle afin d'attirer le plus de client durant un intervalle de temps donné. Pour cela les utilisateurs proches recevront une notification de cette spéciale offre.

Des statistiques sont aussi disponibles sur la page pour que le propriétaire ait un retour immédiat sur ce qu'il propose, « Nearby » permet de mieux connaitre ses clients.

2.2. Technologies utilisées

2.2.1. GPS et scan des réseaux wifi

Pour traquer la position de l'utilisateur et savoir s'il se situe proche des localisations « Nearby », il faut réussir à obtenir des informations sur sa position tout en consommant le moins de batterie possible.

Pour cela dans la version Android, il y a une mise à jour régulière de la position en utilisant la puce GPS avec une faible précision (rayon d'environ cinq cents mètres) puis si une localisation « Nearby » n'est pas trop éloigné, un scan des réseaux wifi est lancé en mode passif c'est-à-dire qui ne consomme pas trop d'énergie. Nous savons alors si un utilisateur est dans la localisation « Nearby » s'il est à proximité des réseaux wifi normalement visibles depuis cette dernière.

2.2.2. Beacons

Les Beacons sont de petits boitiers qui transmettent des données utilisant le Bluetooth aux appareils électroniques proches. Cette technologie est reconnue pour gérer les positions des utilisateurs en intérieur, en particulier elle est plus efficace que le GPS qui est peu précis pour détecter les positions à l'intérieur des bâtiments et consomme plus de batterie. Cette technologie est notamment portée par Apple, qui en a équipé dans tous les Apple Stores américains.

Figure 2 Exemple de Beacons

2.2.3. Chargement de pages web pour l'affichage

La partie visuelle de l'application n'est pas directement créer en natif, dans le projet « Nearby » il est utilisé des pages web qui sont chargées dynamiquement des que l'utilisateur clique sur un bouton ou lors de l'affichage de la page d'un Beacon par exemple. Cette pratique a l'avantage d'être indépendante de la plateforme, les mêmes pages sont utilisées sur Android et iOS par contre l'inconvénient est un chargement un peu moins rapide de l'affichage en particulier quand la page possède beaucoup d'éléments.

2.3. Buts de mon travail

Comme je l'ai dit précédemment ma mission était de continuer le développement de l'application iOS en essayant d'utiliser le même design de conception que la version Android.

2.3.1. Résoudre les problèmes bloquants

Lors de mon premier jour dans l'entreprise, l'ingénieur ayant commencé le développement m'a fait état de l'avancement en particulier sur les problèmes qu'il a rencontré qui sont vraiment bloquant pour la suite.

Le premier problème était dû à la gestion de la localisation de l'utilisateur, il est impossible de s'aider des réseaux wifi environnants car Apple ne permet d'utiliser ces informations malgré une demande faite au support. Pour remplacer cela, l'ingénieur utilisait la puce GPS en permanence ce qui provoquait une consommation de batterie excessive (batterie consommée en deux ou trois heures). Sans gestion efficiente de la position de l'utilisateur, l'application n'est pas utilisable.

L'autre problème important concernait l'affichage et le chargement des vues de l'application. Afin que l'affichage (réalisé en HTML et JavaScript) se rafraichisse en fonction des données de l'utilisateur, la page est capable d'appeler une fonction de l'application. Mais cet appel était asynchrone

c'est-à-dire que l'affichage réalise l'appel pour se mettre à jour mais n'attend pas le résultat de la fonction de rafraichissement ce qui du coup ne provoque aucune mise à jour.

2.3.2. Développer les fonctionnalités présentes dans la version Android

Le développement de l'application avait été stoppé sachant que cela allait mon sujet du stage. Du coup bon nombre de fonctionnalités n'était pas présente, ce qui a constitué des tâches à réaliser au cours de ces deux mois.

La partie sociale de l'application n'avait pas été commencée. Il fallait notamment gérer les amis ou encore pouvoir partager les bons plans sur les réseaux sociaux.

L'affichage des pages des Beacons avait aussi reçu une mise à jour pour l'affichage des modèles qui devait être gérer car cela était une avancée importante dans le projet.

Aussi, il fallait gérer les liens profonds c'est-à-dire afficher les pages web de l'application référencées par Google directement dans l'application lors d'un clic dans le navigateur du téléphone.

D'autres fonctionnalités pour améliorer l'expérience de l'utilisateur étaient à faire mais ces dernières n'étaient pas indispensables pour la conception d'un produit minimal.

2.4. Avancement attendu à la fin du stage

Même si aucun état d'avancement du projet attendu au bout des sept semaines ne m'avait été donné. J'ai essayé d'établir un calendrier pour les différentes semaines afin d'avancer le plus efficacement possible tout en ne restant pas bloquer sur une tâche non bloquante.

Numéro de semaine	Taches principales	Taches secondaires
1	Prise en main et compréhension du projet	X
2	Se former à l'univers Apple	Algorithme de Localisation
3	Algorithme de Localisation	Problème chargement vue
4	Connection automatique Wifi	Algorithme de Localisation (Test)
5	Gestion des Beacons	X
6	Partie sociale	X
7	Fonctionnalités qui améliorent l'expérience utilisateur	X

La première partie du stage est réservée pour prendre en main le projet et me former aux outils utilisés. Ensuite, je me consacrerai aux points bloquants et les fonctionnalités qui sont indispensables à l'application. Pour finir avec des tâches qui améliorent l'expérience de l'utilisateur. Au début du projet, j'ai une vision plutôt précise pour les trois premières semaines.

3. DESCRIPTION DU TRAVAIL REALISE

Je vais maintenant détailler les solutions que j'ai implémentées au cours du projet, tout en donnant le temps accordé à chacune.

3.1. Prise en main du projet

Tout d'abord, il a fallu que je prenne en main le projet c'est-à-dire que je comprenne le fonctionnement de l'application, ce qui était attendu et aussi voir l'avancement du développement, puis que j'apprenne à utiliser les outils de l'entreprise et les outils propres au projet, induit par Apple.

3.1.1. Apprentissage du langage Swift

Pour ce stage, j'ai dû apprendre un nouveau langage, le Swift. C'est en autre la possibilité de connaitre ce langage lancé en 2014 que j'ai choisi ce sujet de stage. L'apprentissage a plutôt était rapide car le langage est syntaxiquement assez proche de langage existant avec une composante fonctionnelle tel que le Python ou le JavaScript. En plus, de la documentation fournie par Apple j'ai suivi un tutoriel en ligne avant le stage pour avoir une connaissance de la syntaxe de base comme la gestion des tableaux ou le système de classes et structures. Du coup, même en n'ayant aucune précédente expérience même en Objective-C, l'apprentissage et la compréhension de ce langage n'ont pas été bloquants et ne m'ont pas fait perdre trop de temps.

3.1.2. Spécificité de l'univers Apple

Développer une application en iOS nécessite de travailler sur un Mac et surtout d'utiliser XCode, l'environnement de développement fourni par Apple. Ces outils étant optimisés, sont plutôt faciles en prendre en main et facilite le développement. Ce qui a posé davantage de problème dans le développement est la politique restrictive adoptée par Apple quand l'utilisation des composants du téléphone en particulier lorsque l'application est en tâche de fond ou a été tuée par l'utilisateur. En effet, lorsque l'utilisateur appuie sur le bouton « Home » l'application a alors un maximum de trois minutes pour terminer son traitement. De plus ce temps n'est même pas donné lorsque l'application est suspendue par l'utilisateur.

Vu que dans l'application, nous voulons traquer la position de l'utilisateur pour le notifier des magasins ou commerces. Il a fallu adapter le comportement de l'application pour cette version iOS et pas seulement reproduire le même que la version Android.

3.2. Reproduire la même application dans un environnement différent

3.2.1. Gestion de la localisation de l'utilisateur

Comme énoncé précédemment, il n'est pas possible d'adopter la même stratégie que pour la version Android. De plus, il faut réussir à analyser la position du téléphone quand l'application est en tache de fond ou suspendue.

Lors de la prise en main du projet, la position de l'utilisateur était suivie grâce à l'utilisation du GPS avec la meilleure précision possible. Il est vrai qu'avec cette implémentation la position est toujours connue car les applications utilisant le GPS en permanence ne sont pas suspendues par le noyau du téléphone. Par contre, la consommation de batterie est vraiment excessive, le téléphone tenait environ deux à trois heures seulement ce qui rendait l'application inutilisable.

Pour trouver une solution, j'ai essayé de nombreuses implémentations que je testais chaque soir sur le trajet vers mon appartement afin de voir si le comportement était bon ou à améliorer.

Pour première modification, j'ai diminué la précision du GPS mais cela a eu seulement effet d'économiser un peu de batterie.

Ensuite j'ai développé une stratégie utilisant le changement de « position significative » proposé par Apple, qui réveil l'application dès que le téléphone est déplacé de cinq cents mètres en théorie. Dès qu'une nouvelle position significative était capturée, le GPS était lancé pour trois minutes. Avec ce choix, l'application ne devait pas être autant consommatrice et c'était vrai mais il s'avère que le changement de position significative n'était pas déclenché lorsque l'on se déplace à pied ce qui ne lance jamais le GPS et donc nous ne connaissons jamais la position de l'utilisateur.

Souhaitant avoir des mises à jour plus régulières, à ce stade du projet, j'ai changé complétement le comportement en mettant en place un minuteur en fond pour obtenir une position GPS toutes les trente secondes. Mais pour éviter la suspension de l'application au bout des trois minutes, j'ai laissé allumer le GPS du téléphone avec la précision la plus faible possible (pas de mise à jour de position avant un déplacement de dix kilomètres) et avec un minuteur à intervalle de temps régulier je faisais une requête à la puce GPS pour obtenir seulement une position. Avec cette stratégie, les gains énergétiques étaient conséquents car il fallait trois jours pour vider totalement la batterie du téléphone mais ce n'était toujours pas suffisant.

Du coup, j'ai dû recommencer totalement l'algorithme en utilisant le système des régions c'est-à-dire que l'on crée des cercles correspondants à des régions, chaque cercle correspond à une localisation « Nearby » et le noyau notifie l'application lors de l'entrée et de la sortie d'une région. Cependant, le nombre de régions gérées simultanément est limité (environ à une vingtaine) donc dès qu'une « position significative » est détectée, nous gérons les localisations « Nearby » à une distance prédéfinie, pour mes tests j'avais choisi un kilomètre (Figure 3). Cette stratégie semble la plus optimisée tant par le comportement qui permet de connaitre la position de l'utilisateur que par la consommation de batterie. L'application consomme dix pourcent de batterie en quinze heures de temps. Malheureusement je n'ai pas pu pousser mes tests avant la fin du stage pour pouvoir dire si cette solution est la bonne.

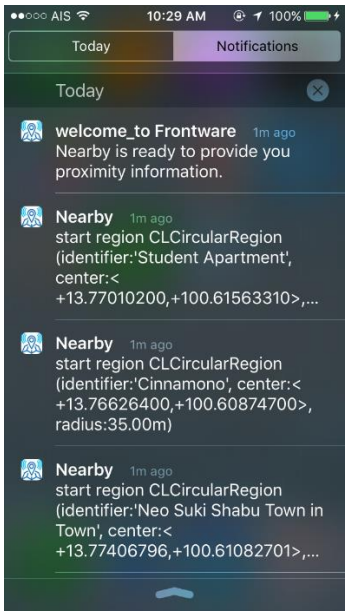

Figure 3 Exemples de régions monitorées

Une autre possibilité que j'aurais voulu implémenter est l'utilisation de la dernière version du podomètre des téléphones d'Apple, mais cette puce (M7) n'était pas disponible sur l'iPhone de test de l'entreprise mais les fonctionnalités que j'ai pu lire dans la documentation sont intéressantes ce qui constitue une piste intéressante si la stratégie des régions ne s'avère pas assez précise.

3.2.2. Détecter des beacons

L'algorithme de détection de beacons avait déjà été commencé dans le projet, mais aucun n'était reconnu par l'application.

La gestion des beacons est plus simple en iOS qu'en Android, pour que le smartphone se mette à les rechercher, il faut lui créer des régions comme pour la détection de la localisation en lui passant l' « id » et le « major » des beacons que l'on recherche. Ainsi dès que l'utilisateur rentre dans une localisation « Nearby », l'application crée une région de recherche de beacon dans cette dernière en passant les informations (id et major) pour la localisation récupérées auprès du serveur.

Il est tout à fait possible qu'une localisation possède plusieurs beacons et que le téléphone en détecte plusieurs simultanément. C'est pour cela qu'il fallait gérer la distance pour déterminer lequel est le plus proche et lancer l'affichage de la page du beacon. Mais la valeur de la distance retournée par n'importe quel beacon était erroné, pour un se trouvant sur mon bureau, à environ un mètre, la distance retournée par le beacon était de vingt-quatre mètres. Pour corriger, j'ai suivi l'implémentation d'une fonction faite en Android pour déterminer la distance en fonction du « Transmission Power » de chaque beacon.

J'ai aussi gérer l'affichage des pages de ces derniers. En effet, lorsqu'un utilisateur se rapproche d'un beacon, il peut voir la page de celui-ci. Cette page peut être une page web ou un modèle rempli par le propriétaire d'une localisation. Depuis peu, ils peuvent depuis le portail de configuration ajouté un modèle ou un style de page à un beacon qu'ils doivent juste remplir. J'ai implémenté le chargement spécifique qu'il était nécessaire de faire pour pouvoir afficher ces pages conçues à partir de modèle (Figure 4).

Ensuite, il est possible à l'utilisateur d'ajouter cette page dans ces favoris pour pouvoir la revoir sans être à proximité du beacon correspondant ou de la partager sur les réseaux sociaux.

Figure 4 Exemple de page modèle d'un beacon

3.2.3. Notification « collante »

Pour implémenter la fonctionnalité d'offre promotionnelle que peux lancer un propriétaire d'une localisation. Ainsi lorsqu'un utilisateur passe à proximité d'une localisation proposant ce type d'offre, il reçoit un type de notification spécial en Android, celle-ci est équivalent à un bon de réduction qui doit explicitement être supprimé du centre de notification par l'utilisateur en cliquant sur le bouton « Cancel ». Cependant ce type de notification n'existe pas en iOS. Afin d'obtenir un comportement équivalent et après avoir reçu la notification sur l'iPhone, lorsque l'utilisateur clique sur la notification, celle-ci est recréer automatiquement. Cette dernière a été un peu complexe à mettre en place car selon l'état de l'application (suspendue, en fond ou en cours d'utilisation) le code exécuté n'est pas le même et aussi éviter la génération infinie de notifications à la suite de la recréation.

3.2.4. Connexion Wifi automatique

L'application Android permet une connexion automatique du téléphone aux réseaux wifi des localisations « Nearby » ainsi dès qu'un utilisateur se rend dans l'une de ces dernières il n'aura pas besoin de renseigner les identifiants wifi. Apple ne donne pas un libre accès à une librairie pour le wifi du téléphone. La librairie « NEHotspotHelper » fournie par Apple et qui pouvait potentiellement être utile pour implémenter cette connexion automatique. Mais la demande faite auprès du support d'Apple pour pouvoir l'utiliser à reçue une réponse négative. Par conséquent cette fonctionnalité, qui n'est pas réalisable pour que l'application soit accessible sur l'Apple Store, ne sera pas disponible dans la version iOS.

3.3. Inclusion des librairies

Dans un projet de cet échelle, l'utilisation de librairie est indispensable afin de gagner du temps que l'on peut consacrer aux taches métiers.

3.3.1. Authentification Digits

Afin de d'obtenir les préférences de l'utilisateur pour créer des statistiques, il est nécessaire que chacun soit authentifié lorsqu'il lance l'application « Nearby ». La librairie choisie ici est « Digits » proposée par Twitter permet une authentification basé sur le numéro de téléphone. Ce choix a été fait car il est vraiment adapté à une cible mobile, plutôt que d'utiliser un email de confirmation « Digits » envoie un SMS avec un code. La documentation est de plus claire et fournie des codes exemples. Malgré cela, j'ai passé plusieurs heures sur cette tâche. Je voulais récupérer l'adresse email en plus du processus d'authentification classique mais même si l'utilisateur remplissait le champ, la valeur de ce dernier ne m'était pas renvoyée. Pour comprendre cette erreur, j'ai contacté le support Twitter pour qu'il me donne des informations. Ils m'ont répondu en précisant qu'il avait corrigé un problème de leur côté qui devait causer cette erreur. Mais je n'ai pas pu tester car mon compte « Digits » de développement est devenu ineffaçable et lié au téléphone de test, une restauration d'usine du téléphone juste après mon départ va être faite et permettra de vérifier que le problème a été résolu.

3.3.2. Utilisation de librairies disponibles sur Github

Github recense un grand nombre de librairies ou de morceaux explicatifs qui m'ont aidé à me former mais aussi pour certaines tâches à inclure directement une librairie pour éviter de perdre trop temps.

Dans un premier temps, j'ai recherché les applications qui permettaient d'exploiter le GPS de l'iPhone pour comparer les résultats obtenus par rapport aux miens et voir s'il n'y avait pas d'implémentation d'algorithme de localisation plus performant que celui que j'ai implémenté. Sur ce sujet, je n'ai rien trouvé d'intéressant les librairies utilisées étaient souvent encore plus consommatrice de batterie.

Sur cette plateforme, j'ai pu trouver des pistes afin de résoudre des problèmes comme par exemple le partage d'information à travers l'application de messagerie LINE. Apple ne propose pas de gestion native de cette dernière au contraire de Facebook et Twitter. Pour tout de même, pouvoir partager des informations avec ses contacts LINE et grâce à un tutoriel disponible sur Github, j'ai pu contourner ce problème en apprenant à utiliser les « URL Schemes » qui permettent à une application d'influencer sur une autre. Ici, depuis « Nearby » lors d'un clic sur le partage à travers LINE, cette dernière s'ouvre avec un message pré rempli.

J'ai bien entendu aussi inclus des librairies directement depuis Github, comme par exemple pour choisir son avatar dans son profil, j'ai utilisé une librairie permettant soit de choisir une photo dans sa photothèque ou d'en prendre une directement depuis l'application. L'utilisation de cette dernière m'a

paru pertinente car elle a permis d'accélérer la réalisation d'une tache qui n'est pas fondamentale pour le fonctionnement minimal de l'application.

Pour justement inclure de nouvelles librairies, j'ai dû apprendre à maitriser le gestionnaire de dépendance CocoaPods qui permet l'ajout facile de ces dernières. Après avoir eu quelques problèmes de configuration, j'ai réussi à inclure, mettre à jour ou supprimer toutes les librairies souhaitées.

3.4. Intégration dans une architecture existantes

Comme j'ai pu le dire précédemment, le développement de cette application devait s'intégrer dans une architecture qui existe déjà, c'est-à-dire que la partie « serveur » était déjà implémentée tout comme la partie « vue ». Je devais par conséquent respecter chacun de leur contrat.

3.4.1. Envoyer les informations aux vues

La partie visible de l'application n'a pas été directement créée en natif afin que l'aspect visuel de l'application Android et iOS soit le même et développé qu'une seul fois.
Le transfert d'informations entre les vues et l'application en elle-même est réalisé grâce à l'appel de fonctions JavaScript par la vue qui sont en réalité des fonctions implémentées dans l'application en Swift.

A mon arrivée dans le projet, ce mécanisme ne fonctionnait pas à cause de l'asynchronisme des fonctions JavaScript qui appelaient la méthode Swift correspondante mais n'attendaient pas le retour d'information. Ainsi il était impossible d'échanger les données et donc d'avoir un affichage dynamique.

Pour résoudre ce problème, j'ai installé une version antérieure de « webview », l'outil qui permet de charger les vues pour que les utilisateurs puissent les voir, tout en créant un nouveau contexte JavaScript pour que les appels deviennent synchrones. La réalisation de cette tâche a été un pas important car sinon il aurait été nécessaire de redévelopper une grande partie du code JavaScript déjà en place.

Pour ajouter une fonction qui fait le pont entre la vue et l'application, il faut ajouter cette fonction dans un protocole qui sera passé à la vue lors de son affichage et qui sera exécuté dans le même contexte et donc synchrone.

3.4.2. Echanges avec le « WebService » et stockage d'informations

Pour ce projet, l'architecture choisie est une architecture REST. J'ai donc dû adapter le développement de l'application pour réussir à communiquer avec le serveur. Du fait de l'utilisation de cette architecture en cours et lors du projet de fin de SI4, j'ai été à l'aise dans la conception de tous les types de requêtes demandées et dans la récupération des informations reçues pour les stocker dans ma base de données locales. Cela a été facilité aussi par la documentation mise en place au sein de l'entreprise qui regroupe l'ensemble des appels « WebService » avec les informations à envoyer et même des exemples de JSON retournés.

Comme je l'ai écrit précédemment je devais stocker les informations dans une base de données locales au téléphone de type SQLLite, grâce au cours sur les bases de données reçues à l'école, je n'ai pas eu de problème particulier pour maitriser le langage de requêtes et donc insérer, récupérer ou supprimer des informations dans celle-ci.

3.5. Planning du stage

Au cours du stage, j'ai tenu un journal de bord (voir Annexe) où j'ai décrit chacune de mes journées avec les taches réalisées et le temps passé.

Dans le graphique ci-dessous (Figure 5), j'ai représenté chacune des fonctionnalités implémentées en fonction du temps total passé à l'entreprise. Nous pouvons nous rendre compte que ce

sont vraiment les taches importantes, qui sont essentielles au fonctionnement de l'application sur lesquels j'ai passé le plus de temps. Je leur ai accordé un temps supérieur car sans détection de la position de l'utilisateur et des beacons il n'y a pas les bases de l'application. Ces deux dernières ont prises plus d'un tiers du stage.

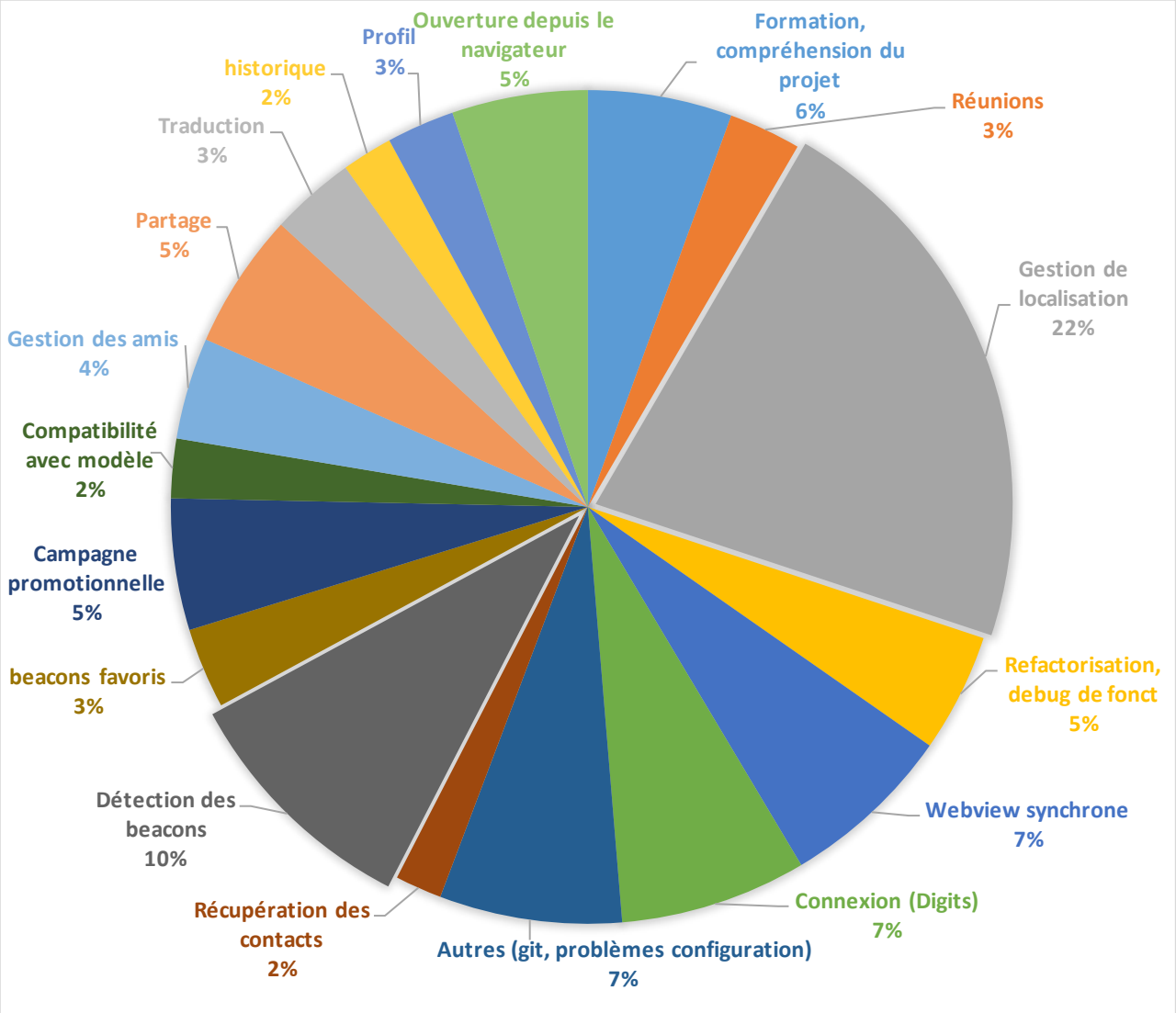

Figure 5 Proportion de temps passé sur chaque tache

4. CONCLUSION

Je vais dans cette partie essayer de prendre du recul sur le travail que j'ai fait, ce que j'aurais pu peut être mieux faire et enfin de l'impact sur mon projet professionnel.

4.1. Enjeux résolus et avancement à la fin du stage

4.1.1. Solutions implémentées ou données aux problèmes initiaux

Lors de mon premier jour dans l'entreprise, après la présentation générale du projet, le développeur de l'entreprise m'avait donné les problèmes bloquants, qu'il fallait résoudre en premier. Tout d'abord, pour ce qui est de la localisation de l'utilisateur, j'ai exploré de nombreuses pistes et je pense que l'implémentation faite au final même si elle reste à perfectionner est fonctionnelle et plus performante que celle initialement utilisée.

Ensuite, grâce à la lecture de la documentation ainsi qu'à la réponse reçue de la part d'Apple, j'ai pu valider le fait que certaines fonctionnalités comme la connexion automatique au Wifi étaient impossibles ou même juste avoir les informations des réseaux à proximité.

Aussi, j'ai pris l'initiative de contacter le support Twitter pour comprendre l'erreur retournée par le service Digits. Même si je n'ai pas pu tester le correctif proposé, le bug est normalement corrigé.

Enfin, j'ai résolu le problème pour l'affichage des vues, en rendant les appels JavaScript synchrones.

Au final, pour chacun des problèmes initiaux j'ai soit apporté une implémentation, donné solution à tester ou invalider la possibilité de réaliser la tâche.

4.1.2. Avancement dans le projet

En plus des problèmes résolus, j'ai développé de nouvelles fonctionnalités qui étaient déjà présentes dans la version Android et aussi j'ai mis à jour certaines parties du code pour gérer les avancées côté serveur ou au niveau de l'affichage comme par exemple l'affichage de la page d'un beacon spécial utilisant les modèles. Ces dernières sont toutes listées dans le planning (Figure 3). La majorité des implémentations ont plutôt était courte ceci a était dû à l'acquisition d'expérience faite après les premières taches bloquantes sur lesquelles j'ai passé plus de temps.

Malheureusement je n'ai pas eu le temps de terminer le développement de l'application. Il est assez difficile pour moi de déterminer s'il était possible dans le temps imparti de terminer l'application avec le peu d'expérience en iOS que j'avais au début du stage. Certes, certaines compétences de mon côté doivent être améliorées mais développer une application dans l'univers Apple requiert une habitude, pour avoir des solutions qui utilisent des chemins différents ou même éviter de perdre du temps sur certains points qui sont restreints par Apple. Du coup, j'ai vraiment pu apprendre des compétences particulières.

4.2. Connaissances validés et acquises

4.2.1. Connaissances techniques

Avec le développement de cette application, j'ai pu valider des compétences vues en cours et aussi en apprendre de nouvelles, ce qui était pour moi le plus important.

Tout d'abord, j'ai pu mettre en pratique ce que j'avais vu en cours comme par exemple l'apprentissage de git en troisième année ou encore les différentes architectures pour construire des applications en quatrième année. Aussi les projets précédents comme celui de fin de quatrième année avec de la programmation Android donc de la programmation mobile m'a permis de mieux appréhender la programmation iOS et de me donner plus d'expérience en programmation mobile.

Mais surtout, j'ai eu l'occasion de me familiariser avec des technologies que je n'avais eu l'occasion d'utiliser. Avant ce stage, je n'avais jamais développé d'application iOS ou même macOS.

De ce fait, j'ai dû me former au langage d'Apple : le Swift. La syntaxe et l'aspect fonctionnel de ce dernier n'ont pas trop été déroutants avec l'utilisation de la documentation. Pour pousser même l'apprentissage de ce dernier, pour gérer les traductions dans l'application, je devais traduire les fichiers XML utilisés en Android en fichier Strings compatible en iOS pour cela j'ai écrit un script en Swift pour réaliser cette conversion.

J'ai bien appris à gérer la politique d'Apple et ces connaissances me seront très utiles lors de mon prochain développement en iOS que ce soit lors de la conception de l'application ou de son implémentation. En particulier, je serai plus à l'aise avec la gestion des taches en fond, mais aussi avec le système de localisation et enfin avec le système de notification.

Le fait d'avoir une base de données côté client était nouveau pour moi, ce qui a nécessité un petit temps d'adaptation pour surtout optimiser c'est-à-dire éviter de faire trop d'appel pour économiser de la batterie.

4.2.2. Gérer un projet en entreprise

Lorsque l'on travaille dans une entreprise, les projets que l'on réalise sont destinés à un client qui a émis ses attentes. Ceci est différent des projets réalisés à l'école où la plus part du temps un sujet complet et détaillé nous est donné, ici il fallait vraiment interroger pour comprendre ce qui étais attendu et pouvoir le réaliser. C'est pour cela que tout au long du stage je n'ai pas hésité à poser des questions dès qu'un point n'était pas clair pour moi.

Ce projet a été vraiment motivant car je savais qu'il y aurait potentiellement des utilisateurs qui verraient le travail réalisé. Aussi, ce projet n'avait pas une date limite de rendu, il était nécessaire de penser au possible évolution puis aussi de permettre à d'autres développeurs de pouvoir comprendre mon code.

De plus, il était important d'avoir toujours le planning du projet en tête et estimer le temps que je devais passer sur une tâche, avoir de la vision pour être certain que le travail prévu est celui qui doit être fait. Aussi lors des réunions, il fallait réussir à expliciter les tâches déjà résolues c'est-à-dire décrire ce qui pourrait peut être changé dans le futur ou présenter des cas de test qui prouve que la fonctionnalité est correcte. Ici, le travail réalisé même si j'étais le seul à travailler sur le projet n'est pas un travail individuel, je me suis intégré dans l'environnement de l'entreprise pour suivre les différentes avancées du projet sur les autres plateformes, ici cela a été facilité par l'utilisation de l'application « Slack » ou même de laisser un historique de ce que j'avais fait avec des explications supplémentaires sur chaque tâche.

4.3. Compétences à développer

Je me suis aperçu durant le stage que j'avais besoin de travailler certaines compétences qui sont indispensables de mon point de vue à un ingénieur.

4.3.1. Estimation des taches

Mon estimation de la durée de chaque tâche n'a pas toujours était la bonne, il suffit d'observer le planning prévu à gros grain et au final le temps passé pour voir les différences. Durant les projets en quatrième année, ce scénario s'était produit au début lors de la conception du planning, le chiffrage n'était pas précis. Pour la réalisation des dernières tâches, le temps estimé était déjà plus proche (erreur d'une demi-journée). Je pense que le développement de cette compétence est très dépendant de l'expérience dans la technologie utilisée car il est plus dur d'avoir une intuition sur le temps que l'on va passer que lorsque l'on a déjà réalisé cette tâche ou une partie de celle-ci pour un autre projet.

4.3.2. Prise en main d'un code existant

Ce projet était le premier que je ne commençais pas depuis le début et j'ai vraiment surestimé ma capacité à maitriser l'ensemble de la base de code déjà faite. Il m'a fallu plus d'une semaine pour avoir une vision globale et distinguer les principales fonctionnalités dans le code. En plus, après il faut réussir à intégrer son code dans l'architecture en la modifiant le moins possible car le développeur qui avait commencé le projet est celui qui doit le maintenir après mon départ.

Raccourcir le temps d'entrée dans le code d'un projet est nécessaire pour être plus rapidement productif et je pense que c'est à cause de cela que je n'ai pas pu avancer le projet autant que je l'aurais souhaité d'où la nécessité de développer cette compétence.

4.4. Impacts sur mon projet professionnel

Il est vrai que développer des applications est plaisant car cela présente des challenges nécessitant des connaissances dans plusieurs domaines, être créatif tout en ayant des compétences techniques solides pour mettre en forme ses idées. Ce stage m'a permis de confirmer que l'informatique est bien le domaine dans lequel je travaillerai dans le futur. Avant le projet de fin de quatrième année et ce stage, je n'avais jamais fait de programmation mobile et c'était un type de développement qui éveillait ma curiosité. Mais après trois mois à développer pour des cibles mobiles, je n'ai pas eu autant l'envie et donc ce stage n'a fait qu'amplifier mon désir de travailler dans la sécurité informatique qui est vraiment la partie de l'informatique qui m'attire le plus.

ANNEXES

Bibliographie

http://ixfocus.com/wp-content/uploads/2015/07/beacons1.jpeg
http://www.frontware.com/Nearby.html/
https://www.nearby.in.th/
https://developer.apple.com/ios/resources/
https://www.google.fr/maps

Journal de bord

Jour 1 : 01/07

8h 8h30 -> présentation de l'entreprise
8h30 -> 9h30 -> explication de ce qui a été déjà fait dans le projet et des problèmes + installation du matériel

9h30 - 17h -> rentrer dans le code gros projet, comprendre les différents modules et mécanismes qui les lient
Tentative de trouver des solutions pour résoudre les problèmes, quelques pistes pour certains

Conclusion : difficulté pour rentrer dans le code (accentué par l'anglais), mais après plusieurs j'ai déjà assez bien compris l'articulation du projet mais besoin de poser encore des questions lundi matin

Jour 2 : 02/07
8h - 9h : réunion avec un ingénieur pour répondre à mes questions et voir ce que j'avais fait vendredi

9h - 12h : recherche d'une solution pour améliorer les performances dû location manager (moins de consommation de batterie, éviter les appels inutiles au gps)
Solution : ajustement de la précision et gps et du pas (distance nécessitant le rafraîchissement)
Envoie d'une requête à Apple pour utiliser une librairie permettant de faire un scan wifi des réseaux aux alentours

13h - 17h : première ligne de code du projet afin d'implementer un location manager qui enregistre les positions de l'utilisateur même si l'application a été tuée
On a besoin de trouver une solution pour faire des tâches en background comme les services en Android mais qui n'existe pas en iOS. Solution envisagée : Lancer les différents traitements (scan wifi / bluetooth)dès lors que la position de l'utilisateur est mise à jour pour faire un genre d'effet background
Comment faire pour que l'application soit active (recenser les positions de l'utilisateur) dès le départ car impossible de lancer l'application au boot. Solution envisagée : peut être notification push

Conclusion : Journée difficile, toujours compliqué de comprendre l'integralité du projet mais ajout des premières lignes de code devrait permettre d'avancer davantage demain

Jour 3 : 08/07
8h - 9h : réunion avec ingénieur pour faire le point

9 - 12h : essayer d'améliorer les réglages gps pour pas que des fonctions soient invoquées alors que l'on pas besoin d'elle notamment lorsque l'application est en background. Elle consomme davantage de batterie en arrière plan qu'en premier plan

13 - 17h : changement de l'algorithme de geolocalisation pour économiser de l'energie. Pour cela on fait des scans régulièrement gérés par l'OS (environ 5 mins ou des qu'il y a un changement de position) puis on lance le tracker gps s'il y a un changement de localisation, pendant 30 secondes pour avoir une vision plus précise des alentours, voir s'il y a des beacons

Conclusion : je commence à comprendre l'environnement iOS, j'ai réussi à comprendre en détail la gestion de la localisation en Swift qui n'est pas autant permissive et modulable qu'en Android, à continuer demain

Jour 4 : 09/07
8h - 12h : refactorisation du code car une classe contenait les fonctions essentielles de l'application et étaient toutes statiques. J'ai donc commencé par créer une classe singleton pour la partie localisation puis j'en créerai d'autres par la suite pour supprimer cette classe dieu. Puis j'ai terminé l'implementation de la nouvelle stratégie de localisation que j'avais commencé la veille

13h - 17h : test pour savoir si lorsque l'application est tuée par l'utilisateur si elle continue à enregistrer les positions de ce dernier. Problème : dès que l'on termine l'execution de l'application le débogguer se déconnecte lui aussi. Pour le résoudre j'insere des données dans la base, j'ai donc dû me familiariser avec le driver de bd (comment créer une table et faire des requêtes). A la fin de la journée je me suis intéressé aux systèmes de webview qui est utilisé pour l'affichage de l'application.

Conclusion : réalisation de ma première feature dans le projet mais le reste du code a aménager reste important ainsi que les améliorations à faire.

Jour 5 : 10/07
8h - 8h30 : point sur ceux qui à été fait la veille + établissement du planning de la journée
8h30 - 9h : tentative de voir les problèmes énoncés par l'ingenieur sur les webviews, problème d'appel asynchrone entre le Javascript et le Swift
9h - 12h : Suite à ma demande la veille pour faire un premier test de l'application en taille réelle, il a fallu que je rentre mes identifiants dans l'application à la place de ceux utilisés pour le développement. Ce changement à fait ressortir des bugs dans la méthode d'inscription de l'application qui utilise l'Api Digits de Twitter.
13h - 15h : à force d'essai l'Api Digits m'a été bloquée ce qui rend impossible l'accès à l'application, mais j'ai quand même eu le temps de trouver des bugs dans la partie front et background end. J'en ai profité donc pour me renseigner sur les webviews en iOS puis pour réorganiser les branches de mon git
Conclusion : Finalement dans cette jounée je n'ai pas fait les tâches que j'avais prévu de faire mais me concentrer sur de la résolution de but d'un code datant de plus de 6 mois

Jour 6 : 08/07
8h - 9h : Recherche du bug d'identification côté serveur. Changement réalisé pour qu'un utilisateur possédant une identification Digits puisse se connecter à l'application
9h - 12h : Recherche du bug Côté front end car lors de la première connexion le mail entré par l'utilisateur n'est pas retenu par l'Api Digits. Le trop grand nombre de requête à entraîné encore une fois le blocage de mon compte, ce qui m'a rendu impossible l'acces à l'application pour le reste de la journée.
12h - 14h : Impossible de build le projet, erreur dans l'édition de lien dès librairies gérées par Pod (équivalent iOS de gradle). Appel à l'aide d'un autre ingénieur pour résoudre le problème du au final au .gitconfig qui était interprété par Pod pour rechercher les dernières versions.
14h - 16h : Écriture de la documentation renseignant ce problème, et tout ceux que j'ai pu rencontré pour trouver des solutions plus rapidement s'ils se reproduisent.
Conclusion: Journée ou j'ai peu avancé du d'une part au blocage de mon compte et d'autre part au bug de l'édition de lien. Le fait d'essayer de trouver le bug dans le module d'identification m'a permis d'apprehender une autre partie du projet

Jour 7 :11/07
9h - 12h : recherche d'une solution pour pouvoir appeler du code Swift en Javascript et vice-versa afin de pouvoir intéragir avec la webview. L'implémentation faite pour le moment est bouchonné car l'appel des méthodes en Javascript est asynchrone ce qui pose problème. Essaie de plusieurs projets liant Javascript et Swift sur des snippets mais pas de solution trouvée.

13h - 15h : Tentative de compréhension du bug de sign in, toujours pas compris pourquoi l'API Twitter ne renvoie pas l'adresse mail mais j'ai trouvé des nouvelles pistes peut être dû aux thermes en Swift.

15h - 17h : librairie trouvée sur github permettant de résoudre le problème de la webview, l'essaie sur un snippet de code est concluant
Conclusion : même si je n'ai toujours pas trouvé de solution pour l'authentification, le problème de la webview sera peut être réglé dans les prochains jours.

Jour 8 : 12/07
9h - 12h : adaptation de la librairie trouvée la veille pour le projet, ce qui a induit beaucoup de changement au niveau des contrôleurs des vues (passage des wkwebview ennui webview). Problème de ponts entre le Javascript et le Swift, toutes les fonctions ne sont pas appellables
13h - 15h : problème du au compilateur Swift qui ne mettait pas le même nom de fonctions que le développeur, en enlevant le bouchon l'application est fonctionnelle avec cette dernière implémentation.
15h - 17h : impossible de rebuild le projet à cause de l'inclusion d'une librairie en objective-c qui n'ai plus reconnu, ceci est dû à un problème de configuration que je n'ai pas encore trouvé.
Conclusion : j'ai terminé aujourd'hui hui une tâche importante qui m'avait été donnée car sans elle toute la partie Javascript de l'application aurait dû être repensée mais encore du temps perdu sur problèmes de configuration.

Jour 9 : 13/07
9h - 12h : résolution du problème de la veille qui a obligé la réinstallation du débogguer de Xcode du à la base à une erreur dans un fichier de config. Validation de la feature d'appel Javascript / Swift donc marge de la branche sur dévelop
13h - 15h : Tentative de résoudre le problème du sign in qui est sûrement du au format des mails utilisés par l'entreprise. Le code erreur renvoyé par Twitter n'est pas documenté ce qui rend la tâche plus compliquée
15h - 17h : préparation de l'application pour tester le système de localisation. Je vais prendre le téléphone à mon appartement pour voir comment la geolocalisation se comporte et voir si le beacon chez moi est reconnu. Pour suivre l'activité, des logos sont enregistrés dans une base locale
Conclusion : Encore beaucoup de soucis de config et le bug récalcitrant du sign in. Le test de la geolocalisation va permettre de savoir si le travail de la semaine dernière est payant ou nécessite des ajustements

Jour 10 : 14/07
9h - 12h : Analyse des résultats obtenus après le déplacement du téléphone. En foreground il y a bien eu les modifications, par contre dès que l'application est en background il n'y pas eu de nouvelles localisations. Tentative de comprendre pourquoi, les changements ne sont pas assez significatifs, configuration de nouveaux paramètres pour pouvoir tester ce soir.
13h 15h : Gestion du rafraîchissement du token par l'api digits pour que les modifications soient reportées dans le web service de l'entreprise. Mais toujours le problème de la première connexion du à une erreur côté Twitter ne pouvant pas enregistrer l'adresse mail
15h 17h : Tentative d'implementer une nouvelle stratégie pour gérer mieux la localisation que je testerai plus tard si la stratégie actuelle n'est pas satisfaisante
Conclusion: les tests de la stratégie sont plutôt décevants et m'oblige à repasser du temps sur une feature qui aurait pu être terminée.

Jour 11 : 15/07
8h 10h : Analyse des résultats de la localisation de la veille, plus de positions mais pas encore assez. Mise en place d'une nouvelle stratégie de geolocalisation qui utilise davantage le gps pour obtenir des positions plus précises. Pour premier test l'application sera laissée en background pendant une semaine et voir si le portable a toujours de la batterie
10h 12h Travail sur un nouveau ticket qui permet de récupérer les informations du répertoire de l'utilisateur lorsqu'il s'authentifie. Durant la matinée, prise en main du framework de gestion de répertoire fourni par Apple
13h 16h30 : Conception de la requête Jason à partir du contrat qui m'avait été donné puis test sur le serveur de développement de l'entreprise

16h30 17 merge de la branche ou j'ai implémenté le ticket sur la branche de développement
Conclusion: première fonctionnalité qui ne pose pas vraiment de problème et qui ne se heurte pas à la politique d'apple. Cette dernière m'a permis de mieux pratiquer le langage Swift

Jour 12 : 25/07
8h 12h : Utilisation d'une librairie trouver sur github pour mieux gérer la geolocalisation du téléphone car les tests des précédentes stratégies s'avérent un peu trop gourmand en énergie ou n'etait pas adaptées à l'utilisation que l'on veut en faire dans l'application car pas assez de paramétrisation du GPS.
13h 15h : Tentative d'authentification à l'application avec un numéro thaïlandais, authentification toujours impossible. Le SMS de confirmation n'a même pas été envoyé par l'API. L'utilisation d'un autre système d'authentification doit peut être être envisagé, il faudra en discuter avec le responsable du projet quand il rentrera
15 16h30 : Tentative de trouver sur github des examples d'utilisation du framework de geolocalisation fourni par Apple. Peut être qu'une stratégie utilisant des Timbers est envisageable mais cela semble peux efficace quand à l'économie de la batterie
16h30 17h : Préparation du téléphone pour le test de geolocalisation de ce soir : purge de la base de données et préparation des requêtes pour le test de la nouvelle librairie
Conclusion : Vu que les résultats fournis par mon code ne sont pas les meilleurs, j'ai installé une librairie tiers afin de voir si leur algorithme est meilleur

Jour 13 : 26/07
8h 11h : Les résultats données par la librairie ne sont pas meilleurs que la stratégie que j'avais précédemment implémentée. Du coup suppression de ce qui avait été fait la veille et utilisation de t'aimer pour avoir une mise à jour périodique de la localisation. Mais dès que l'application est en tache de fond, elle est suspendu du le coup la stratégie t'aimer devient inutile
11h 14h : Tentative d'utiliser le framework de gestion de l'accéloromètre fourni par Apple afin d'éviter d'utiliser le GPS. Mais l'iPhone que possède l'entreprise n'est pas équipé des dernières puces M7 de gestion de mouvement ce qui ne permet pas de tester la dernière version de la librairie
14h 15h : implémentation d'une nouvelle stratégie en utilisant beaucoup le vos et test ce soir voir si elle n'est pas trop consommatrice
15h 17h : Essai de résoudre un bug sur l'écran de chargement. La semaine dernière après un build, le logo n'a plus été chargé alors qu'il était visible dans la Prévisualisation et quand on lance le simulateur mais pas de solution trouvée.
Conclusion : toujours à essayer d'adpater la stratégie de geolocalisation mais les possibilités semblent de plus en plus réduites

Jour 14 : 27/07
8h 8h30 : Analyse des résultats qui sont plutôt convaincant avec davantage de mise à jour de coordonnées et moins de 10% de batterie consommée entre le départ de l'entreprise et mon arrivée ce matin. J'ai augmenté davantage la précision à tester ce soir si le nombre de mise à jour reçu ne sera pas trop consommateur.
8h30 16h : changement dans le code de scanning Bluetooth permettant de détecter les beacons. Le module de gestion du Bluetooth étant important en ligne de code il m'a fallu une heure pour saisir le fonctionnement global. Le changement au niveau de la geolocalisation induit forcément des changements au niveau de la fréquence des scans Bluetooth. Étant donné que le nombre de coordonnées reçues est plus faible pour économiser de la batterie il faut que la détection des beacons soit plus réactive et surtout que les scans de ces derniers soient plus récurrents. Ces changements m'ont pris une grande partie de la journée
16h 17h : Vérification si mon nouveau numéro de téléphone avait été débloqué par Digits, ce fut le cas mais l'authentification ne marche toujours pas mais amélioration ici j'ai reçu un mail de confirmation sur mon adresse mail.

Jour 15 : 28/07
8h 10h : augmentation du nombre de scans pour que la détection soit plus rapide puis réduction de l'intervalle avant d'afficher la page relative au beacons.

10h 14h : suppression de la librairie utilisée pour gérer l'acceloremetre et la gestion des mouvements qui se base en fait sur le GPS. Ces changements m'ont permis de faire gagner 10% de CPU et certainement la batterie également.
14h 16h : j'ai remarqué que l'affichage de la page web était aussi consommatrice en CPU, tentative de voir si il y avait pas moyen d'économiser certain appel à des fonctions mais il semblerait que c'est le site web en lui même qui a un problème du à trop d'appel JavaScript.
16h 17h : Tentative de résoudre le bug de chargement du logo, celui ci s'affiche sur une page sur deux, cela était dû à un problème de hommage

Jour 16 : 29/07
8h 12h : Bug dans le code de geolocalisation testé hier donc les résultats obtenus ne sont pas significatifs. J'ai tout d'abord corrigé ce dernier pour pouvoir le tester la semaine prochaine. J'ai tout de même pu me rendre compte que la vitesse renvoyée par le GPS était complétement fausse par conséquent j'ai implémenté une fonction qui l'a calcule. La vitesse me paraît une métrique importante pour pouvoir adapter la précision du GPS (par exemple plus faible précision si on est en voiture)
13h 15h : résolution du bug empêchant l'affichage du logo.Du à un ancien commit certaines images avait été renommées et du coup le téléphone ne le trouvaient plus. Ce problème n'a pas été visible avant car l'image restée dans le cache du téléphone.
15h 16h : Recherche pour être sur qu'il ne soit pas possible de lancer en background au boot de l'iphone l'application mais il semblerait que cela soit bien interdit par Apple
16h 17h : Création d'un projet à part pour refaire l'installation de Digits pour trouver le bug.

Jour 17 : 01/08
8h - 10h : Correction du bug Digits certainement dû à un problème dans le gestionnaire de dépendance qui meme s'il indiquait que la dernière version était installée, ce n'était pas le cas. Plus mise à jour pour demander le token pour FCM.
10h - 14h : problème de configuration, impossible d'installé l'application du à un renouvellement de certificat pour le mois d'aout qui lors de l'installation avait été corrompu.
14h - 17h : Début d'une nouvelle tâche pour gérer les bookmarks. Tout d'abord besoin de récupérer les informations d'un beacon en faisant un appel au web service puis en le stockant dans la base sqllite

Jour 18 : 02/08
8h - 8h30 : adaptation des valeurs en fonction des résultats obtenus sur la geolocalisation
8h30 - 14h : début de la récupération des informations f'un Beacon auprès du web services puis après l'inserer dans notre base. Ce qui a m'a permis de prendre en main le système de base de données.
14h - 17h : Mise en place des appels web services pour recevoir les campagnes. Mise à jour de la BD en récupérant les tables manquante par rapport à la version Android. Puis création d'une nouvelle table et début de la mise en place du module

Jour 19 : 03/08
8h 8h30 : Suppression des pauses automatiques du GPS pour pouvoir en garder davantage le contrôle.
8h30 - 12h : Fin de l'affichage des bookmarks,code d'appel au web service pour pouvoir en ajouter un et intégration d'une librairie de snack-bar
13h - 17h : Essaie de corriger le bug qui empêche l'affichage de l'action relative au beacon. Problème dans la gestion des distances de ces derniers, du coup calcul à la main en fonction du signal de ce dernier. La partie du code de gestion des beacons et assez complexes et le code est peu lisible (nom de fonctions / nom de variables) ce qui rend ça compréhension difficile

Jour 20 : 04/08
8h - 8h30 : changement dans les réglages du GPS car des qu'il passe en mode low il est trop difficile de le refaire changer de mode
8h30 - 13h30 : essaie de résoudre le problème Digits que j'arrive à faire fonctionner sur un snippet avec le meme code. Puis solution trouvée pour gérer à la fois l'authentification à notre web service et stocker le token envoyé par fcm
13h30 - 17h : Correction des bugs sur le code de la détection des beacons, nouvelle requête base pour récupérer le tx_power d'un beacon à porté afin de déterminer sa distance.

Jour 21 : 05/08
8h - 10h : Merge entre deux branches avec beaucoup de conflits du à des fichiers de configuration dont j'ai retiré le versioning.
10h - 15h : Affichage d'un template quand on clic sur un beacon dans la liste des favoris, pour cela mise à jour des fonctions Swift qui sont appelés par le JavaScript.
15h - 17h : Essayer d'afficher la page de template quand on hit un beacon mais problème dans le chargement du à l'encodage de l'url, puis ensuite la page n'execute pas les bonnes fonctions Javascript (bridge unique)

Jour 22 : 08/07
8h - 10h : réunion pour faire un point sur l'avancement du projet, ce qui a été fait, sur ce qui reste à faire et comment.
10h - 12h : chargement de la page du beacon lorsqu'on le hit
13h - 14h : Ajout d'un bookmark implémentation de la fonction bridge avec le Javascript qui fait l'ajout.
14h - 16h : suppression de bookmark. Requête web service, suppression en base puis méthode Javascript.
16h - 17h : implémentation d'une stratégie utilisant la puce M7 pour réduire la consommation de batterie lors du scan Bluetooth et début de l'exploitation de l'accelerometre.

Jour 23 : 09/07
8h - 10h : Détecteur de mouvement utilisant l'ancienne version de l'accelerometre, essayer de savoir si l'utilisateur se déplace en fonction de trois axes (CMAcceleration)
10h - 12h : requêtes web services pour récupérer les amis ou pour en ajouter + stockage en base.
13h 14h : délecte d'un friend (ws , base, Javascript)
14h 17h : Correction des fonctions JavaScript pour l'affichage des amis, début de la mise en place du partage (envoie de message, d'invitation) avec d'autre application (UIActivity)

15/07 :
8h - 12h : cache sur les infos des beacons
13h - 17h : uialertview pour pouvoir share un beacon à travers les applications de messagerie . Utilisation du framework social d'apple et du chargement d'url pour Line

16/07
8h - 12h : mise en place d'un système de notification côté serveur afin de stopper le GPS pendant la nuit pour économiser la batterie. Par contre problème quand l'application est tuée pas possible de la rereveillée pour continuer les updates que ce soit les notifications locales ou push.
13h - 17h : continuer la fonctionnalité share en ajoutant un bouton others pour ouvrir le uiactivitycontroller si l'utilisateur veut utiliser une autre application. Puis implémentation du share with friends nearby qui envoie directement une notifications avec l'url du Beacon à un autre utilisateur

17/08
8h - 12h : Le système de notification n'a pas été efficace. Mise en place du monitoring par région
13h - 14h : Ajout du bouton share sur la page des beacons qui s'integre. Bug dans l'affichage du template. Debug du taux power dans le spot des beacons

18/08
8h - 11h : les premiers tests du système de région n'a pas été très concluant mais la consommation de batterie est plus faible. J'ai donc essayer d'approfondir le code pour en tirer mieux parti.
11h - 14h : Essais de plusieurs librairie pour transformer les strings.xml en localisable.strings mais pas qui réussisse à 100%.
14h - 17h : réalisation d'un script Swift qui le fait utilisationd'une librairie de gestion de xml et escaping des caractères

19/08
8h - 10h : correction des bugs du script réalisé la veille
10h - 12h : intégration de mon script dans le processus de mise à jour du projet

13h - 15h : Ajout de fonctionnalités dans le système de région pour pouvoir hit les beacons en background
15h 17h : Début de la fonctionalité d'historique des locations

22/08
8h - 10h : Fin de la fonctionalité d'historique
10h - 11h : merge des fonctionnalités précédentes
11h - 12h : Ajout des bridges Javascript pour que les traductions soient visibles dans les webviews
13h - 15h : éviter de hit les beacons si on est pas dans la localisation
15h - 16h : début de la fonctionalité de profil
16h - 17h : mise en place des tests pour tester les nouveautés sur les régions

23/08
8h - 10h : modification sur le code des régions pour avoir plus d'update une fois dans la région.
10h - 14h : mise en place d'un image picker pour modifier l'avater en utilisant un custom url
14h - 17h : Essaie de cropper une image en Swift mais résultat pas convaincant

24/08
8h - 9h : réussie à crop une image
9h - 12h : fonctionnalités campaign handle dès message reçu par gsm + création de la notification reçue
13h - 16h30 : algorithme pour recréer la notification quand l'utilisateur a clicqué dessus. Mais problème pour gérer pour éviter boucle infini
16h30 - 17h : mise à jour de la taille de région plus correction d'un bug sur le scan Bluetooth

25/08
8h- 12h : fin de la mise en place de la notification in tuable
13h 15h : mais problème car message gcm non reçu quand l'app a été tuée
14h - 16h : refactorisation d'une partie du code
16 - 17h : tentative de de comprendre pourquoi la webview ne chargeait pas dès page web

26/08
Correction du bug sur le can share
Refactorisation du code pour avoir toutes les méthodes d'appel a l'api dans la même classe
Après-midi affichage des venues

29/08
8h - 11h : deep Link entre l'application est safari
11h - 17h : gérer le clic hors connexion et la mise à jour de la barre du haut

30/08
8h - 13h : debug des venues (toolbar) et refactorisation du main controller en une hierarchie qui comporte le controller normal de l'application et celui sans log in
13h - 16h : debug et merge des derniers tickets
16h - 17h : changement dans la stratégie de sortie d'une région qui n'est plus la sortie de la location

To validate my engineer school fourth year, I made my internship in Frontware International, a small company in Bangkok. My mission was to continue the development of the iOS version of « Nearby ». This application provide proximity services using Bluetooth to the closest phones. At the beginning of the internship, my main goal was to resolve location and display issue. I gave an answer to these problems and also I implemented several other features like sharing, profile or advertising campaign. I'm disappointed in myself to let the application unfinished, the main reason was my lack of experience in iOS development. Firstly, it was difficult for me to handle and understand the architecture of the code and secondly, I waste a lot of time to avoid Apple restrictions in order to have the same behaviour than the Android application. This internship was the perfect occasion to learn more knowledge like Swift language. Even if that was pleasant to create this application, my professional project stays unchanged that is to work in IT security.