Livrable I : cartographie wifi et bluetooth

Sommaire

I Introduction

1. Description

Dans le cadre d'un projet scolaire en informatique à l'école Télécom SudParis, nous réaliserons une application mobile permettant la localisation d'appareils Wi-Fi et Bluetooth.

2. Équipe

- Augustin Bresset
- Pierre Chambet
- Guillaume De Terline
- Zacharie March

Encadrant

Rémy Grünblatt

3. Développement

Trois pôles ont été retenus pour le développement de l'application :

Frontend

Création de la carte

- Leaflet
- React
- geodjango

Application mobile

Développement de l'application pour smartphone sous android.

- Kotlin
- Android Studio

Backend

Traitement des données, communication avec les serveurs.

- Django (Python)

Supplémentaires

Déploiement des APIs à l'aide de REST. Développement de l'algorithme de géolocalisation.

II Analyse des besoins

1. Cahier des charges

Backend

Établir un serveur qui recueille les données transmises par l'application mobile.

- Établissement du modèle de la base de données avec django : Les modèles de base de données avec Django sont des classes python. Ces classes définissent les champs de chaque table.
- Préciser les informations à recueillir :

    1. le signal bluetooth
    2. le signal WI-FI

    Cette étape prend forme lors de la déclaration des champs. Les signaux recueillis ont plusieurs attributs, il faut donc choisir ceux d'intérêts. Par exemple, les informations utiles peuvent être la puissance du signal reçu et l'heure à laquelle ce signal est reçu.

- Mettre en relation les classes avec les clés étrangères : Afin de pouvoir comparer et utiliser les deux sources d'informations (bluetooth/WIFI).

- Traitement des données : Pour pouvoir employer les données, il faut tout d'abord les traiter dans un format adapté. Il nous faut également des méthodes de classe afin d'exploiter ces données. On peut alors effectuer des opérations, des calculs sur ces données. C'est ici que se fera la partie mathémaqtiques du projet. Il s'agira de traduire les données recoltées pour en faire des informations utiles (distances, coordonnées...)

- Modifier la base de données : Une fois les résultats obtenus, il faut effectuer une migration django afin de mettre à jour les modèles. Cette mise à jour constituera les informations finales qui pourront alors être exploitées par la partie cartographie.

- Envoyer les résultats : Une fois le traitement des données effectué, il faut les publier afin de pouvoir afficher les points GPS sur la carte créée. Pour cela, on peut par exemple créer un fichier CSV à partir de la nouvelle base de données issue de la migration Django. On peut également déployer des APIs REST en utilisant Django REST Framework. Cette problématique représente un travail supplémentaire, à ne faire qu'une fois tous les points précédents effectués.

Développement mobile : Kotlin sur Android Studio

Présenter une application mobile développée sur Android affichant une carte repérant les positions des différents appareils WIFI/Bluetooth émettant un signal perceptible par le smartphone.

- Conception de l'interface utilisateur de l'application via l'éditeur d'interface d'Android Studio et en utilisant la carte après réalisation de la partie Frontend.

- Ajout du code Kotlin permettant d'établir l'échange de données avec le serveur (Backend), et d'indiquer les points WIFI et Bluetooth à proximité. Les données en question concernent les puissances des signaux WIFI et Bluetooth reçus par le smartphone support de l'application.

- Test et débogage de l'application.

Frontend

Réalisation d'une carte où sont pointés les localisations des appareils WIFI/Bluetooth. On peut voir cette carte sur l'application mobile développée.

- Utilisation de la bibliothèque Leaflet pour créer une carte interactive à mettre sur l'application mobile.

- Pointer les localisations des différents appareils WIFI/Bluetooth sur la carte.

- (Utilisation de la bibliothèque React pour créer une interface utilisateur interactive.)

- Éventuellement s'il nous reste du temps: pointer la position de l'utilisateur en fonction de sa proximité aux différents points WIFI par triangulation.

2. Liste des fonctionnalités de l'application

Affichage des points d'accès sur une carte interactive

Les points d'accès seront représentés par des marqueurs sur la carte. Les points d'accès Bluetooth et WIFI seront differenciés par la couleurs des marqueurs. Les utilisateurs pourront interagir avec eux en cliquant

dessus pour disposés des informations locales de chaque marqueur.

Filtrage des résultats

Les utilisateurs pourront filtrer les résultats en fonction de leur position, de la puissance du signal reçu et d'autres critères pertinents.

Ajout de nouveaux points d'accès

Les utilisateurs pourront ajouter de nouveaux points d'accès à la carte en fournissant les informations pertinentes, telles que l'emplacement et la puissance du signal reçu.

III Plan de charge

1. Plan de charge

Plan prévisionnel

Description de l'activité	Charge en %	Charge en heure	Augustin Bresset	Guillaume Macquart De Terline	Pierre Chambet	Zacharie March
Total	100	200	50	50	50	50
Gestion de projet	24	48	12	12	12	12
Réunion de lancement	2	4	1	1	1	1
Planning prévisionnel et Suivi des activités	2	4	1	1	1	1
Réunions de suivi	12	24	6	6	6	6
Rédaction	4	8	2	2	2	2
Outil collaboratifs (Google drive, Github)	4	8	2	2	2	2
Spécification	4	8	2	2	2	2
Définition des fonctionnalités	4	8	2	2	2	2
Conception préliminaire	14	28	7	7	7	7
Définition des données	2	4	1	1	1	1
Définition des formats des données	2	4	1	1	1	1
Définition des modèles des données	2	4	1	1	1	1
Définir les vues sur django	2	4	1	1	1	1
Créer des templates pour la présentation des données	2	4	1	1	1	1
Mise en place des protocoles de triangulation	4	8	2	2	2	2
Conception détaillé	18	36	9	9	9	9
Auto-formation	4	8	2	2	2	2

Plan prévisionnel						
Définition des classes (Kotlin, JavaScript)	4	8	2	2	2	2
Définition des clés étrangères (Django)	2	4	1	1	1	1
Définition des méthodes (Kotlin, JavaScript)	4	8	2	2	2	2
Migration Django	2	4	1	1	1	1
Configurer les URL	2	4	1	1	1	1
Codage	24	48	12	12	12	12
Codage des classes	2	4	1	1	1	1
Codage des clés étrangères	2	4	1	1	1	1
Codage des méthodes	8	16	4	4	4	4
Implémenter la triangulation	4	8	2	2	2	2
Codage de la carte avec la bibliothèque Leaflet	2	4	1	1	1	1
Codage des tests unitaires	6	12	3	3	3	3
Intégration	8	16	4	4	4	4
Intégration des modules	4	8	2	2	2	2
Test d'intégration	2	4	1	1	1	1
Test de l'application	2	4	1	1	1	1
Soutenance	8	16	4	4	4	4
Préparation de la soutenance	6	12	3	3	3	3
Soutenance	2	4	1	1	1	1

2. Suivi des activités

Suivid'activité

Description de l'activité	Charge en %	Charge en heure	Augustin Bresset	Guillaume MACQUART DE TERLINE	Pierre Chambet	Zacharie March
Total		133				
Gestion de projet	10	19	6	6	6	1
Réunion de lancement	1,5	3	1	1	1	0
Planning prévisionnel et Suivi des activités	2	4	1	1	2	0
Réunions de suivi	0	0	0	0	0	0
Rédaction	4,5	8	2	3	2	0
Outil collaboratifs (Google drive, Github)	2	5	2	1	1	1
Conception détaillé	18	48	13	13	13	9
Auto-formation	4	32	10	10	10	2
Définition des classes (Kotlin, JavaScript)	4	6	0	4	0	2
Définition des models, vues (Django)	2	4	2	0	2	1
Définition des méthodes (Kotlin, JavaScript)	4	8	0	1	0	2
Migration Django	2	4	1	0	1	1
Configurer les URL	2	4	1	1	1	1
Codage	24	66	18	18	18	12
Codage des classes	2	4	2	2	3	1
Codage des models/vues	2	4	2	2	4	1

	Suivid'activité					
Codage des méthodes	8	16	2	4	6	4
Implémenter la triangulation	4	8	8	0	4	2
Codage de la carte avec la bibliothèque Leafl	2	4	0	8	0	1
Codage des tests unitaires	6	12	4	2	1	3
Intégration pas encore rempli	8	0	4	4	4	4
Intégration des modules	4	8	2	2	2	2
Test d'intégration	2	4	1	1	1	1
Test de l'application	2	4	1	1	1	1
Soutenance pas encore rempli	8	0	4	4	4	4
Préparation de la soutenance	6	12	3	3	3	3
Soutenance	2	4	1	1	1	1

3. Planning prévisionnel

- 24/02 - 31/03 : Réalisation des conceptions préliminaires et détaillées. Auto-formations en Kotlin, Django et JavaScript
- 31/03 - 17/05 : Réalisation du codage et des phases d’intégration. Validation du bon fonctionnement de l’application.
- 17/05 - 30/05 : Préparation de la soutenance
- 30/05 - 31/05 : Soutenance

IV Conception préliminaire

Première version du plan du rapport final

1. Introduction
2. Cahier des charges
3. Développement
    1. Analyse du problème et spécification fonctionnelle
    2. Conception préliminaire
    3. Conception détaillée
    4. Codage
    5. Tests unitaires
    6. Tests d'intégration
    7. Tests de validation
4. Etude des résultats
    1. Réalisation des tests
    2. Pertinence des résultats
    3. Critique
    4. Suggestion d'Amélioration
5. Conclusion

V Conclusion du livrable 1

1. Récapitulatifs des résultats obtenus dans le livrable

Nous faisons ici le constat de la charge de travail et sa répartition. Une idée plus concrète des étapes du projet s'est formée. Nous avons alors esquissé un travail plus individuel à fournir, tout en restant à l'écoute des objectifs de chacun.

La répartition du travail est présentée de manière uniforme mais nous savons que ce ne sera pas le cas pour les étapes de développement qui seront réparties par pôles. En effet chacun se spécialisera sur sa partie, mais de nombreuses réunions inter-groupe permettront d'expliquer l'avancement du projet et les compétences developpées à chacun.

Le temps d'apprentissage compris dans *auto-formation* est biaisé car répartie sur l'ensemble du projet.

2. Perspective pour la suite du projet

Les prochaines étapes consistent en la mise en place de l'architecture de notre logiciel, le développement de l'application et la création de la carte, la définition des objets employés et des structures de bases de données, et une approche mathématique de triangulation pour estimer la localisation.

Livrable 3

Nous arrivons aux termes du livrable 3, faisons donc un point sur ce qui a été fait et ce qui reste à faire.

Kotlin | Leaflet

Á l'aide d'android studio, une application a pu être produite, d'un autre côté une carte a réussi à être chargé. Enfin les deux ont réussi à être réuni car la carte se charge bien sur le simulateur de l'application intégré à android studio.

Localisation

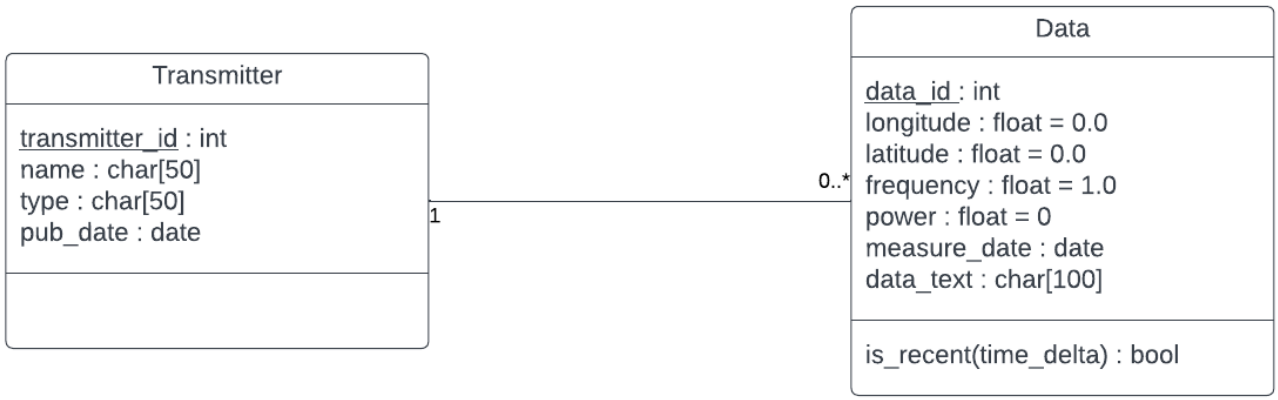
Une librairie locolisation a été créée et publié sur testpypi. Elle permet de triangulariser et donc de localiser un point selon des sources qui en sont son intersections.

Pour l'installer grâce à pip :

```
pip install -i https://test.pypi.org/simple/ localisation
```

Database Django

Une database a été réalisé, composé seulement de deux tables, elle permet de décrire les signaux recu mais aussi son propre signal, c'est pourquoi la localisation du transmitter est dépendante du temps et donc inscrit sur chaque data. Faire ceci permettra éventuellement de localiser des accès selon la position de l'utilisateur.



L'algorithme a été importé sur l'application django et une vue lui est dédié. Actuellement dans la database, il y a un ensemble de données tests qui y résident et montre le bon fonctionnement de l'algorithme ainsi que de la base de donnée.