Département de Génie Informatique et Génie Logiciel

INF8405 – Informatique mobile

Hiver 2018

Rapport du travail pratique nº1

Application de recherche de connexion Wifi via géolocalisation 

Préparé par

Sidy Bouya Ndiaye – 1676529

Antonio Domínguez Trujillo – 1921364

Jordan Ben Mkaddem – 1899011

Soumis à

Mikaela Ngamboé

1 Mars 2018

Introduction

Les appareils mobiles permettent de trouver les connections Wifi captées par ceux-ci et affiche leur nom, la force de leur signal ainsi que le besoin ou non d'entrer un mot de passe. Tous les appareils mobiles ont cette capacité. Cependant leur fonction s'arrête là.
L'application que nous avons développée est faite pour le système d'exploitation Android. Elle permet de géolocaliser les points de connexion Wifi, elle affiche la position du point d'accès Wifi en utilisant la carte de *Google Map* et détermine le chemin pour accéder au point d'accès. Les informations du réseau détecté soient le nom du réseau, l'adresse Mac du point d'accès, une représentation graphique de l'intensité du signal ainsi qu'une description du moyen d'authentification sont affichées par l'application. Toutes les informations des points d'accès détectés pourront être sauvegardées dans une fenêtre des favoris ou partagées avec des contacts téléphoniques d'autres applications.

Présentation technique

1) Description de l'application

L'application a pour nom Wifi Scanner. L'icône de celle-ci a été choisi afin de suggérer à l'utilisateur l'aspect wifi et localisation.

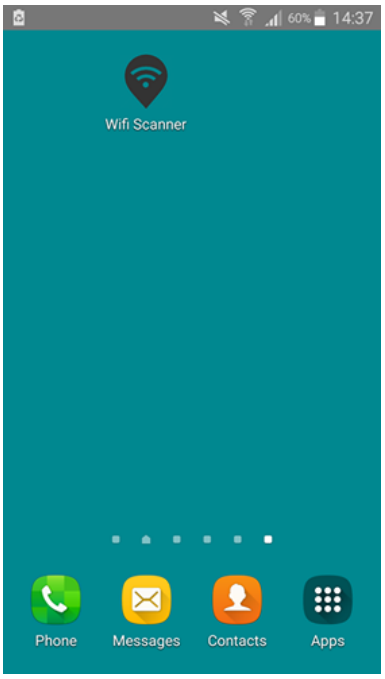

Image 1 : Icône de l'application

Lors du lancement de l'application, le logo de l'application apparaît sur fond bleu pendant quelques secondes puis la carte est affichée.

Image 2 : Logo de l'application

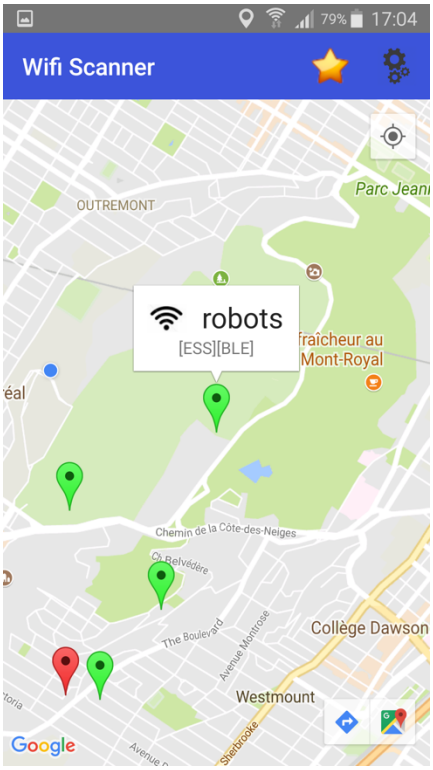

Image 3 : Carte avec la position courante de l'usager

A ce stade, si la localisation est désactivée ou son utilisation interdite par l'OS, une boîte de dialogue apparaît demandant à l'utilisateur de régler cette situation. Celle-ci ne pourra disparaître qu'une fois les données de localisation accessibles par l'application.

La carte est ensuite affichée, avec un zoom sur la position actuelle de l'utilisateur. Les réseaux Wifi sont ensuite placés aléatoirement (dans un rayon donné) sur la carte : un marqueur rouge si le réseau présente une quelconque sécurité, un marqueur vert sinon. L'utilisateur peut accéder au SSID et à l'intensité du signal en cliquant sur le marqueur.

Il peut ensuite afficher l'itinéraire jusqu'à ce point sur GoogleMaps grâce aux icônes en bas à droite de l'écran.

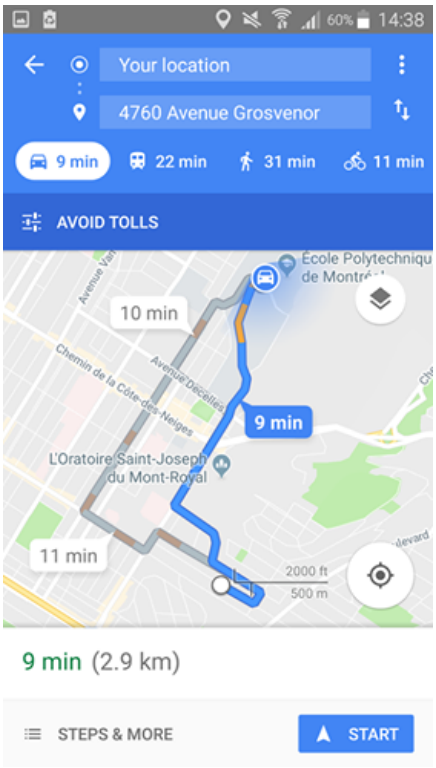

Image 4 : Itinéraire jusqu'au point d'accès

En cliquant sur la fenêtre d'information, il peut également obtenir plus d'informations sur le réseau, l'ajouter aux favoris ou le partager sur les réseaux sociaux.

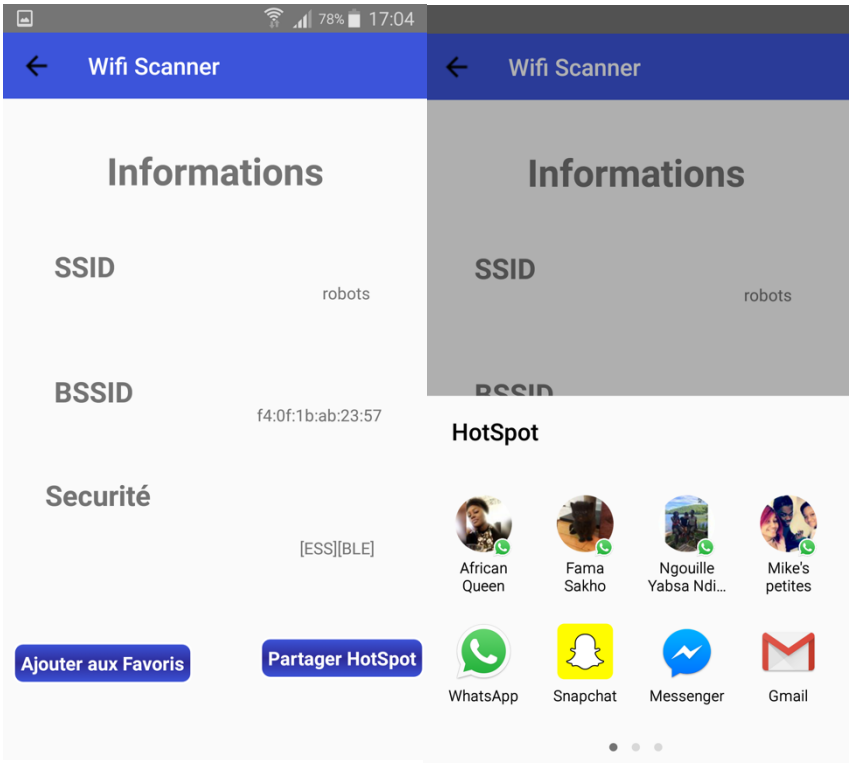

Image 5 : Affichage et partage des Réseaux

En haut de l'écran principal, se trouve une barre d'outils contenant deux icônes, l'un permettant d'accéder à la liste des favoris, l'autre aux réglages de l'application.

La page de la liste des favoris contient deux boutons, "Liste" permet d'afficher uniquement cette liste sans qu'aucune modification ne soit possible tandis que "Supprimer" permet à l'utilisateur de retirer des favoris le réseau qu'il souhaite en cliquant sur le bouton correspondant.

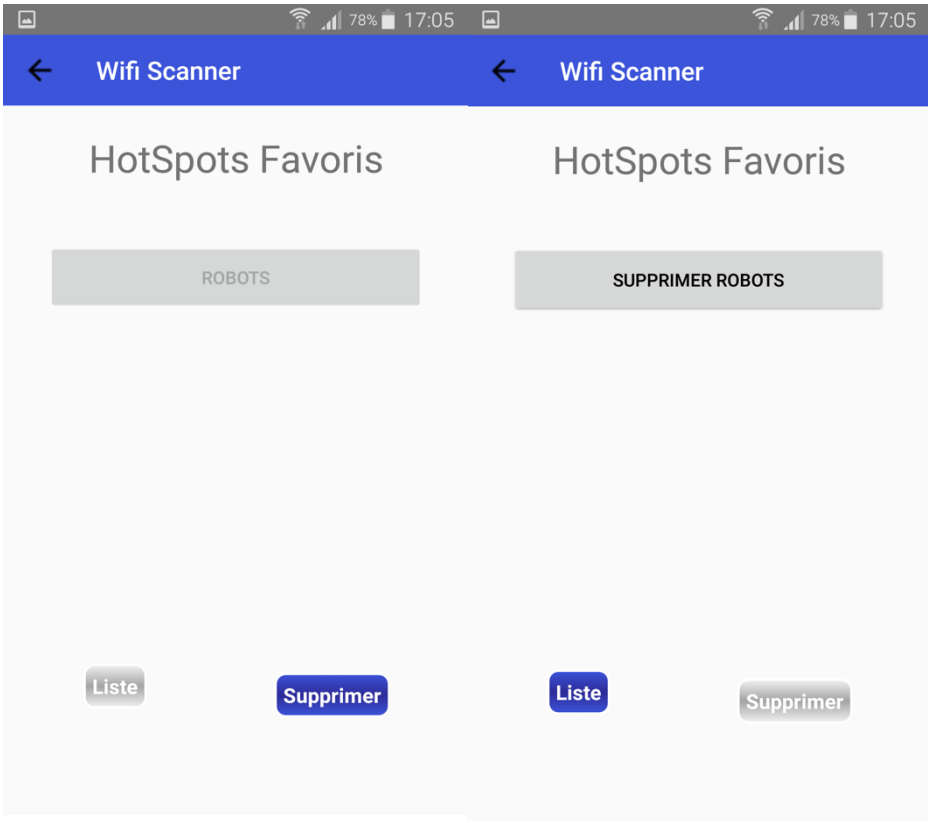

Image 6 : Logo de l'application

Enfin la page Réglage permet de régler la durée entre deux scans WiFi (de 1s à 30s), et également d'effacer tous les favoris.

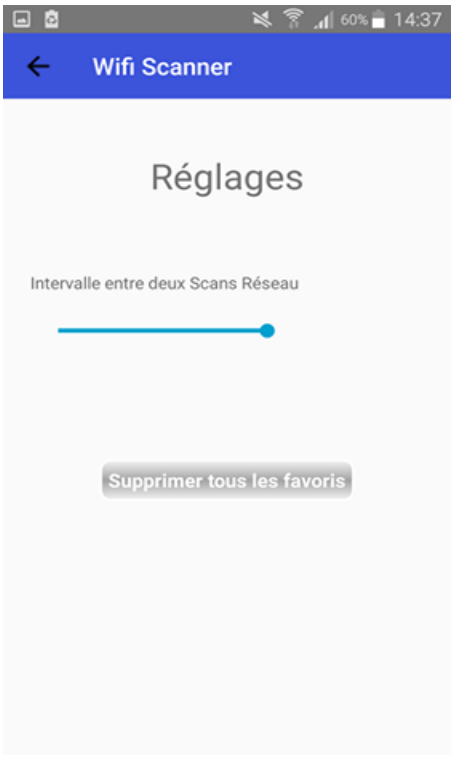

Image 6 : Réglage de l'application

2)Structure de l'application

2.1) Code Java

Tout d'abord, nous nous appuierons sur un diagramme classe *UML* (*image 1*) afin de mieux expliquer la fonctionnalité de chaque composante de *WiFiScanner*.

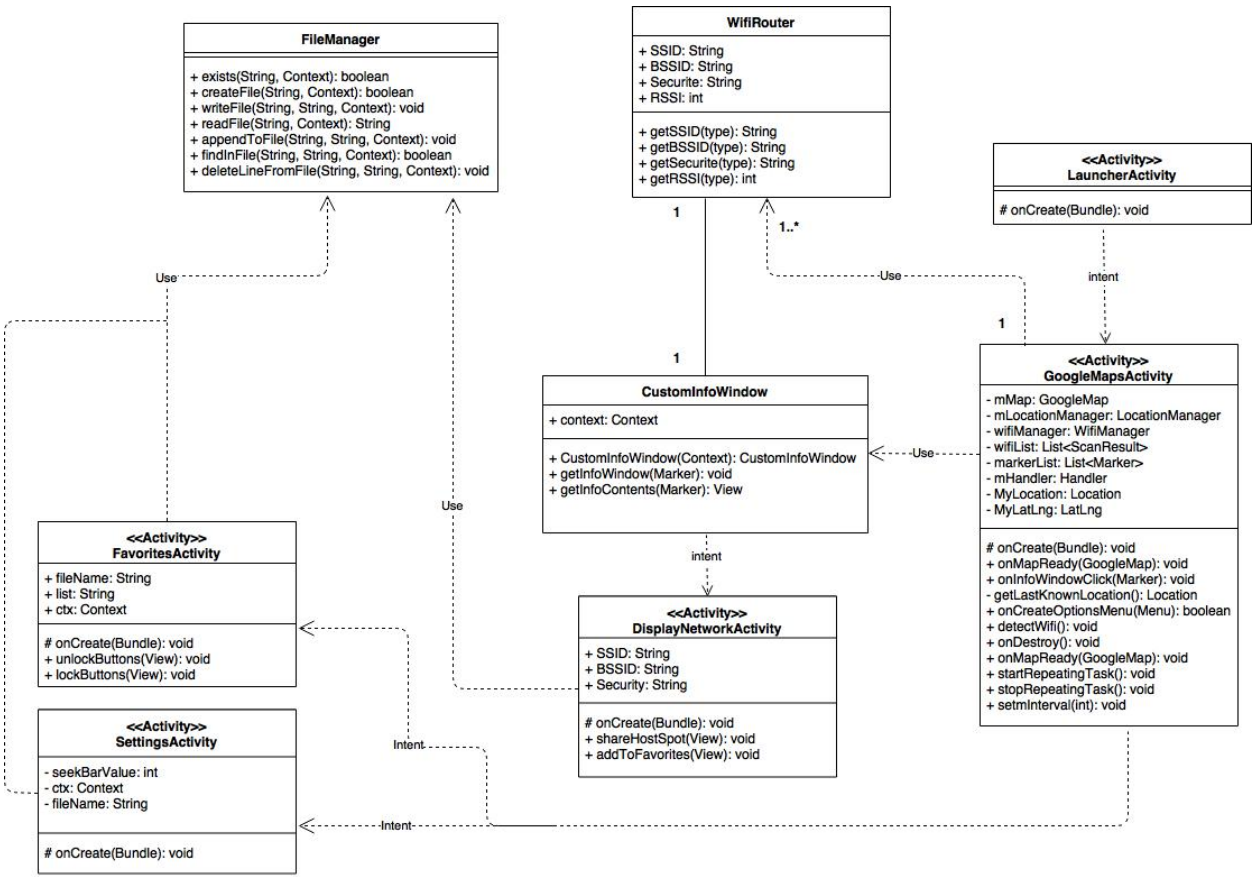

Image 1 : Diagramme de classe de WiFiManager

Ici sont présentées les relations entre les différentes classes, qu'elles correspondent à des activités (nom de classe finissant par *Activity*) ou non (*FileManager, WiFiRouter* et *CustomInfoWindow*).
Voici une description de chaque classe :

- ***LauncherActivity***. Il s'agit de l'activité qui gère la page de démarrage de l'application. C'est la première activité affichée lors du démarrage de l'application. Elle montre l'icône principal de *WiFiScanner* pendant quelques secondes. Après cela, elle crée une *intent* afin de changer d'activité. De cette manière, *GoogleMapsActivity* sera la prochaine activité à être affichée.

- ***GoogleMapsActivity***. C'est l'activité principale de l'application. Elle utilise l'*API* de *Google Maps* afin d'afficher la carte. Une boîte de dialogue est affichée lorsque la localisation est désactivée dans le terminal mobile courant ou si l'utilisateur n'a pas donné les autorisations nécessaires à l'application pour l'accès aux données de localisation. Si celle-ci est fonctionnelle, la carte montre la position de l'utilisateur et commence le scan réseau.
  La recherche des réseaux WiFi est effectuée à l'aide de la bibliothèque *WiFiManager*. Le résultat de ce scan est stocké dans une liste puis dans un HashMap afin de ne conserver qu'un seul routeur par réseau, celui qui a

l'intensité de signal la plus élevée.
Plusieurs objets de type *WifiRouter* sont ensuite créés pour chaque réseau et assigné à un marqueur qui sera placé sur la carte.
Les marqueurs que nous utilisons sont divisés en deux catégories, qu'il s'agisse d'un réseau sécurisé (marqueur rouge) ou non (marqueur vert). En cliquant sur un marqueur, on ouvre une fenêtre *CustomInfoWindow* dont nous parlerons tout de suite. Un bouton avec l'icône de *Google Maps* apparaît alors dans le coin inférieur droit pour permettre à l'usager d'être guidé jusqu'au point de destination grâce à l'application de *Google Maps*, qui se chargera de mener à bien cette tâche.
En tant qu'activité principale de notre application, *GoogleMapsActivity* présente un menu permettant au client de faire usage des autres fonctionnalités, comme par exemple afficher les favoris (icône étoile de la barre supérieure) ou modifier les réglages de l'application. Ces dernières sont deux activités différentes qui seront appelées grâce à l'utilisation des intents.

- ***WifiRouter***. C'est la classe représentant un réseau. Ses attributs correspondent aux informations requises pour l'application. Il encapsule donc le *SSID (Service Set Identifier), BSSID (Basic Service Set Identifier), RSSI (Received Signal Strength Indicator)* et le niveau de sécurité. Ainsi, nous sommes capables de créer plusieurs instances de cette classe, chacune représentant un réseau scanné différent.

- ***CustomInfoWindow***. C'est la classe implémentant la fenêtre d'information GoogleMaps. Elle permet ainsi de personnaliser l'affichage et le contenu de celle-ci. Ces fenêtres sont affichées une fois le marqueur cliqué. Après cela, nous affichons le *SSID* du réseau, le niveau de sécurité ainsi que nous représentons l'intensite du signal grâce à une icône wifi qui change selon 5 niveaux déterminés. Si le client clique sur la fenêtre, il est dirigé vers l'activité *DisplayNetworkActivity*, visant à afficher plus en détail les informations du réseau trouvé.

- ***DisplayNetworkActivity***. Les informations du réseau récupérées grâce à la classe *WifiManager* et enregistrées dans l'objet *WifiRouter* sont affichées. La différence principale réside dans le fait d'afficher également le *BSSID* (adresse *MAC*) ainsi que mettre à disposition des boutons, pour partager le hotspot avec d'autres applications ou pour ajouter ou supprimer le réseau de la liste des favoris. Les méthodes *shareHotSpot* et *addToFavorites* sont chargées d'accomplir ces fonctionnalités.
  Le stockage des favoris est fait à l'aide d'un objet de type *FileManager*, qui s'occupe de la gestion des fichiers rattachés à l'application.

- ***FileManager***. La classe *FileManager* est la classe permettant la gestion des fichiers liés à l'application. Ici, elle permet principalement de gérer la création, la lecture et l'écriture du fichier des favoris, fichier dont le contenu sera préservé même lorsque l'application est quittée. Chaque ligne du fichier contiendra le

SSID, le *BSSID* et le niveau de sécurité d'un réseau, séparés un espace.

- ***FavoritesActivity***. C'est l'activité permettant d'afficher la liste complète des réseaux préférés dans un *widget* de type liste *scrollable*. Cette vue de l'application permet aussi de supprimer un ou plusieurs réseaux de la liste. *FavoritesActivity utilise* un objet *FileManager* afin de pouvoir lire le fichier complet des favoris.

- ***SettingsActivity***. Elle permet de définir la fréquence de la recherche de points d'accès ainsi que l'effacement complet de la liste des favoris en utilisant *FileManager*. En effet, la solution consistant à scanner les réseaux à chaque changement de position GPS nous paraît trop consommatrice en énergie. Il s'agit donc de scanner régulièrement, l'intervalle de temps pouvant être changé par l'utilisateur selon sa vitesse.
  De plus, *SettingsActivity* permet d'afficher le pourcentage de batterie consommée depuis le lancement de l'application contenu dans un attribut de la classe *GoogleMapsActivity.*

2.3) Code XML

Manifest

Ici se trouve la liste de toutes les activités, l'activité *LauncherActivity* étant déclarée comme MAIN et LAUNCHER.
Les permissions nécessaires pour le bon fonctionnement de l'application y sont également déclarées.

Layout

Il existe un fichier xml pour chaque activité gérant l'affichage graphique des éléments. Le nom de ces fichier commence systématiquement par "*activity_*". *info_window.xml* contient les éléments graphiques nécessaires à la personnalisation de la fenêtre d'information GoogleMap.

Drawable

Ce dossier contient toutes les images utilisées dans l'application en format jpeg ou png. Il existe également des fichiers xml :

- *button.xml/button_enabled.xml/button_disabled.xml* permettent de définir un bouton changeant d'aspect selon son état
- *splash.xml* contient ce qui sera dessiné pendant l'activité *Launcher*

Menu

Nous définissons ici un menu qui sera celui utilisé dans la barre d'outils pour naviguer entre l'activité *GoogleMaps*, la liste des favoris et la page Réglages.

Values

Dans *color.xml*, quelques couleurs ont été définies afin de garder une certaine cohérence graphique dans toute l'application.
styles.xml définit un style pour nos boutons qui sera repris dans *themes.xml*

Anim

On définit ici : - les 4 déplacements x d'écran afin de pouvoir effectuer une transition slide entre la fenêtre d'information *GoogleMap* et l'activité affichant plus d'informations sur le réseau.

- fade_in.xml et fade_out.xml afin de faire une transition "fondue" pour la transition vers la liste des favoris et vers les Réglages.

Difficultés rencontrées

Le principal défi lors de la conception de l'application a été de s'adapter aux particularités de la programmation Android comparé au Java classique. En effet la programmation orientée aux évènements ainsi que les multiples méthodes à override requiert une certaine rigueur dans l'organisation du code. De ce fait, le démarrage du projet a été laborieux mais grâce aux nombreuses ressources disponibles sur internet, et le vaste contenu de la bibliothèque d'Android, le challenge a vite été surmonté.

Une des fonctions de l'application est de définir la position du point d'accès du réseau sur la carte, cependant la base de données correspondante est payante. Il a donc été décidé de positionner les réseaux trouvés aléatoirement, dans un rayon donné, sur la carte en manipulant les coordonnées des marqueurs.

De plus, lors de la correction des erreurs du code, nous avons utilisé le débogueur fourni par *Android Studio* en exécutant pas à pas l'application sur un terminal mobile. Cependant, dans certains cas, aucune erreur n'était renvoyée alors que l'application plantait. La plupart du temps, il s'agissait d'une erreur d'affichage que nous ne pouvions pas détecter avec l'outil de débogage d'Android *Studio*.

Critiques et suggestions

La majorité de notre temps de travail a été dédié à effectuer des recherches sur internet afin d'apprendre à développer une application *Android* puisqu'aucun de nous n'avait les connaissances requises.

Nous avons concentré beaucoup de nos efforts afin que l'application réponde parfaitement au mandat (fonctionnel et non fonctionnel). De ce fait, bien que satisfait de l'ergonomie de l'application, elle peut être améliorée.

Une autre difficulté fut de penser l'ergonomie de l'application, afin que celle-ci soit intuitive, nous pensons avoir répondu à cette problématique en utilisant des icônes explicites et en gardant l'interface graphique relativement claire. Il était important pour nous que l'utilisateur n'ait pas à devoir parcourir de nombreux menus avant d'arriver à la carte, c'est pourquoi nous avons décidé de regrouper les fonctionnalités de l'application sur cette dernière, l'utilisateur ayant accès à la plupart des informations des réseaux wifi à partir des marqueurs.

Tout au long du développement, nous avons gardé en tête la consommation d'énergie de l'application, c'est pourquoi il nous semblait important de laisser le choix à l'utilisateur de la fréquence du scan Wifi.

Conclusion

L'application obtenue répond au mandat donné. Une fois les points d'accès Wifi trouvés, leur position est donnée par des marqueurs sur la carte Google Map. Le marqueur est rouge s'il y a un mot de passe, vert s'il n'y en a pas. Lorsque l'utilisateur appuie sur le marqueur une fenêtre avec les informations suivantes est créée : le nom du réseau, l'adresse Mac du point d'accès et une représentation graphique de l'intensité du signal. De cette fenêtre l'utilisateur peut enregistrer le point d'accès sélectionné ou partager ses informations avec ses contacts. Nous avons consacré la majorité de notre temps de travail sur les fonctionnalités de l'application ce qui fait que l'interface graphique puisse encore être améliorée.

Le fait de développer une application Android nous a donné l'opportunité d'acquérir les connaissances de base dont nous avons besoin pour le projet final du cours. De plus, nous avons appris à utiliser la technologie WiFi dans un terminal mobile et l'API de *Google Maps*. De cette manière, avec la réalisation de ce premier travail pratique, nous avons fait une première approche sur le dévl'informatique mobile.