

École Polytechnique de Montréal  
Département de génie informatique et logiciel

INF8405  
Informatique Mobile

Projet final - PolyBlue V2.0  
Hiver 2019

Soumis à :  
Mehdi Kadi

Soumis par :  
Francis Laframboise : 1795342  
Charles-Olivier Favreau : 1789479  
Julien Gauthier : 1791257

Section 01  
29 avril 2019

# Table des matières

<b>Introduction</b>	<b>2</b>
<b>Présentation des travaux</b>	<b>2</b>
Géolocalisation et carte interactive	2
Caméra et Lampe de poche	4
Connexion d'équipements mobiles par Wifi	5
Base de données SQLite	7
Statistiques de l'application	7
<b>Difficultés rencontrées</b>	<b>9</b>
<b>Critiques et Améliorations</b>	<b>10</b>
<b>Conclusion</b>	<b>10</b>

# Introduction

Dans le cadre du cours Informatique mobile INF8405, nous avons été amenés à réaliser un projet d'envergure supérieure aux travaux pratiques, afin de mettre en pratique nos connaissances acquises pendant la session. Nous devons développer une application pour Android comportant plusieurs fonctionnalités, certaines nouvelles et d'autres déjà vues dans les travaux pratiques. Il fallait également y inclure des particularités non fonctionnelles pour améliorer l'expérience de l'utilisateur.

Tout d'abord, l'application devait pouvoir utiliser une connexion entre différents équipements mobiles, soit à l'aide de réseaux sans fils, de partage de données par NFC ou toute autre manière. Ensuite, il fallait y inclure deux types de capteurs différents, comme par exemple l'accéléromètre, la caméra, la proximité et autres. Nous devons également appliquer les concepts de géolocalisation et de carte interactive, le plus facilement réalisé avec la Google Maps API utilisée lors du deuxième travail pratique. De plus, l'application devait utiliser une base de données, de préférence SQLite, afin de conserver diverses traces de son utilisation. Finalement, il fallait implémenter une manière pour l'utilisateur d'avoir accès à plusieurs sortes d'informations, notamment sur les différentes utilisation de la batterie et sur la consommation de la bande passante. Il était à noter que l'application devait pouvoir garder son état en cas où la connexion réseau soit perdue pendant au moins 10 secondes.

Le rapport qui suit permettra la présentation des travaux réalisés pour la création de l'application, en plus de toucher aux diverses difficultés rencontrées au cours du processus de création ainsi qu'aux critiques et améliorations pouvant être apportées à l'énoncé et à notre application.

## Présentation des travaux

### Géolocalisation et carte interactive

La vue et la partie centrale de notre application est la Google Maps Activity, que nous avons implémentée dans la classe MapsActivity et le fichier activity\_maps.xml. C'est en effet sur cette carte que les fonctionnalités importantes agissent : l'ajout de marqueurs et de miniatures, le traçage de trajets, la prise de photo et le démarrage de la lampe de poche.

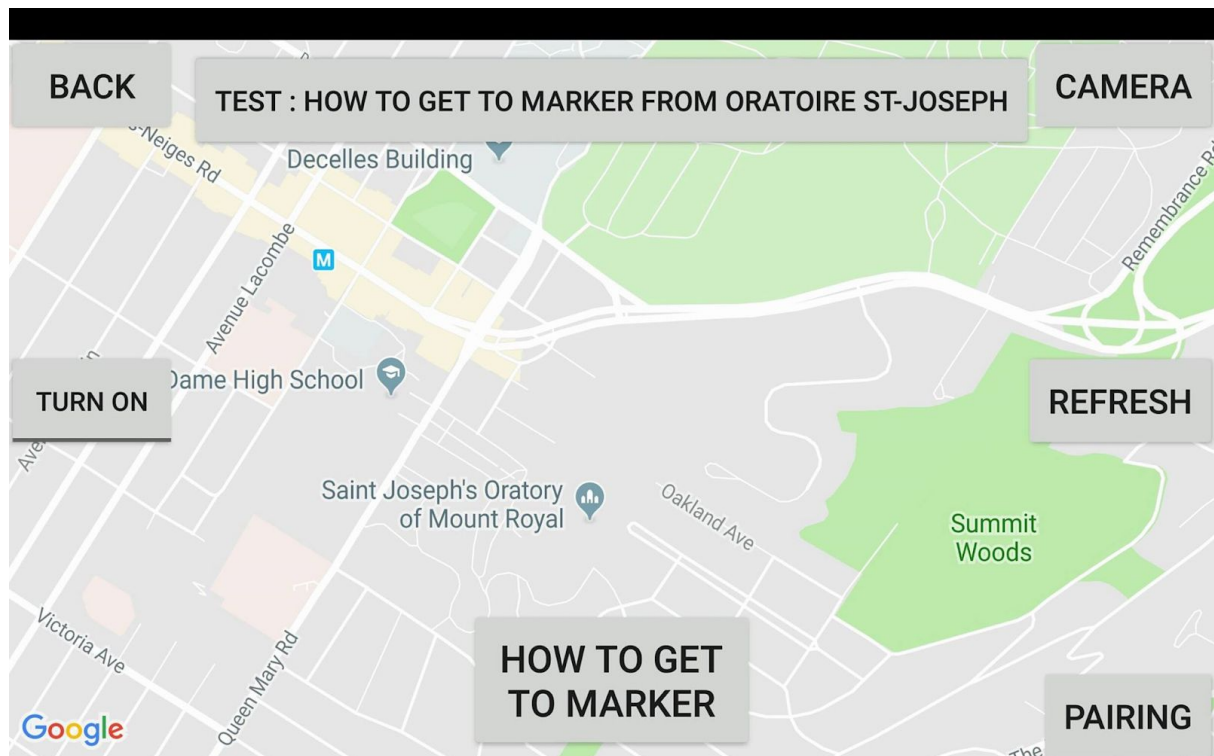


Fig. 1: Vue sur la map de l'application ainsi que les boutons disponibles

Afin d'amener l'application à un point de départ agréable et pratique pour l'utilisateur, nous avons appliqué un zoom maximal sur la carte et centré celle-ci à la position actuelle de l'appareil à la création de la carte. Ici, nous nous sommes fortement inspirés du deuxième travail pratique, alors que nous avons pris le code de localisation déjà écrit et l'avons mis à l'intérieur de la méthode `onMapReady` (héritée de l'interface `OnMapReadyCallback`). La fonction utilisée pour obtenir la géolocalisation de l'appareil est `LocationService.getFusedLocationProviderClient()`. Les photos prises par la suite sont ajoutées à cet endroit sur la carte sous forme d'un marqueur et d'une miniature.

De plus, pour faciliter l'utilisation de l'application, nous avons ajouté une fonctionnalité de "Comment se rendre" à un marker où une photo a été insérée sur la carte précédemment. Effectivement, l'implémentation de cette fonctionnalité effectue une requête `DirectionsApiRequest` à l'API de Google Directions, recevant ainsi les routes possibles de la localisation présente à la localisation du marqueur où l'utilisateur désire de rendre.

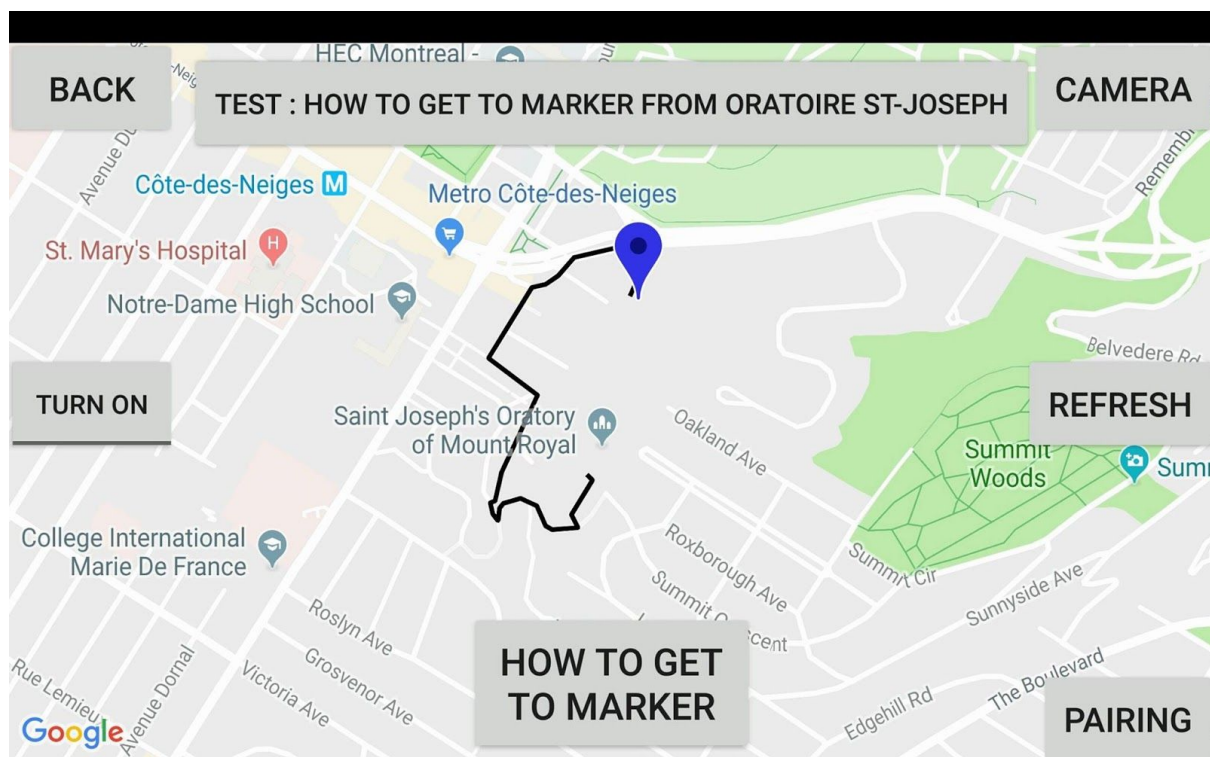


Fig. 2: Vue de la fonctionnalité “Comment s’y rendre”

## Caméra et Lampe de poche

Une des fonctionnalités de l’application est de pouvoir prendre une photo et l’insérer dans la carte, rattachée à un marqueur donnant la position sur la carte où la photo a été prise. Pour implémenter cette fonctionnalité, un bouton sur la vue de la carte permet d’initier un Intent de type ACTION\_IMAGE\_CAPTURE. Cet intent permet de lancer l’application de caméra par défaut. Étant donné que nous voulions utiliser la photo pleine résolution et non seulement la miniature, la caméra lancée par l’intent enregistre l’image dans le storage de l’appareil à l’endroit indiqué par le chemin passé à l’intent. Une fois la photo récupérée dans les fichiers du storage, celle-ci est utilisée pour 2 versions différentes : une est stockée dans les base de données en résolution grande (légèrement moindre que l’image originale, pour des raisons de mémoire) et l’autre est utilisée pour la miniature qui est inséré dans la carte sur un marqueur (la résolution est grandement diminuée).

L’application est aussi dotée d’une fonctionnalité lampe de poche (Flashlight). En effet, l’application permet d’instancier un objet caméra et de lui configurer des paramètres tels que FLASH\_MODE\_TORCH permettant d’allumer le flash de la caméra de l’appareil et de l’éteindre à la demande. Cette fonctionnalité de lampe de poche est uniquement disponible pour les appareils ayant une version du système d’exploitation Android supérieure à API 21. Effectivement, tel qu’il est possible de

voir dans la figure ci-dessous, le bouton d'activation de la lampe de poche est désactivé car l'appareil est équipé d'une version d'Android inférieure à l'API 21.

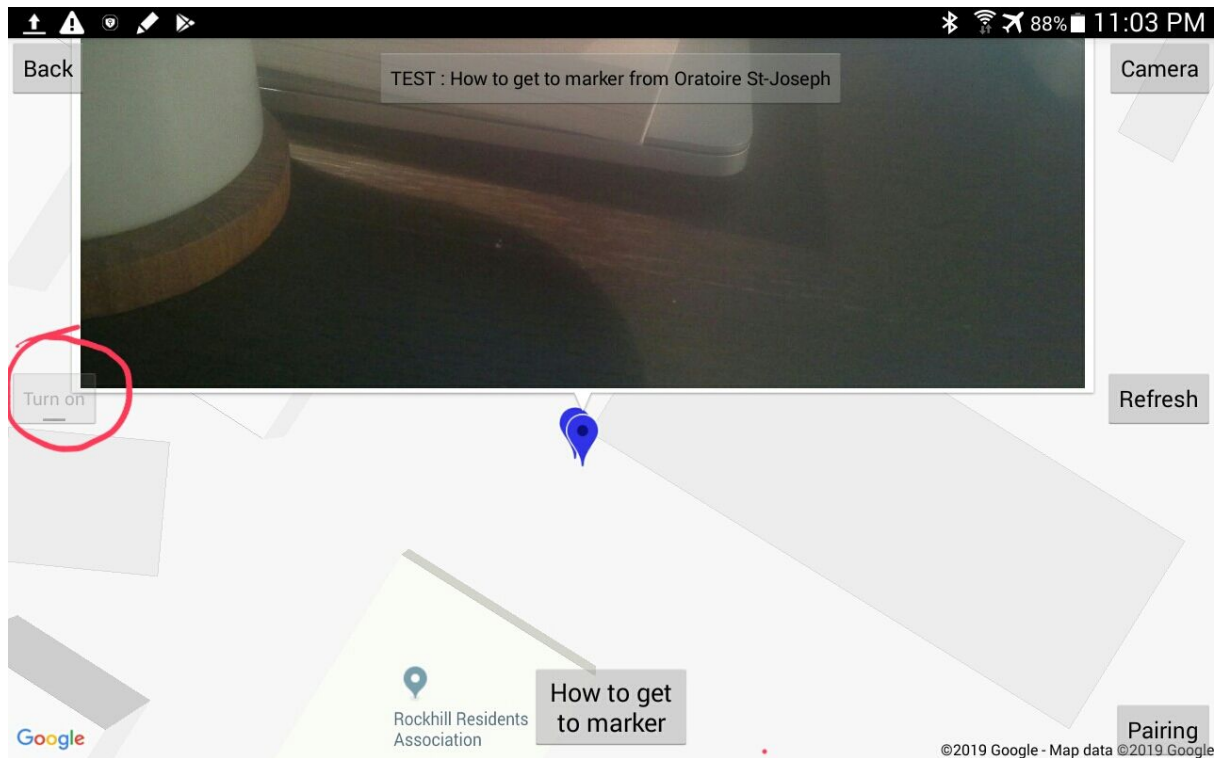


Fig. 3: Option de flash désactivée sur Android inférieur à API 21

## Connexion d'équipements mobiles par Wifi

L'application offre à l'appareil l'option de communiquer avec d'autres appareils sur lesquels l'application est également en marche. Pour ce faire, l'utilisateur peut entrer l'adresse IP de l'autre appareil, disponible à l'écran d'accueil de l'application de celui-ci, dans la fenêtre ouverte par le bouton "Pairing" sur la carte :

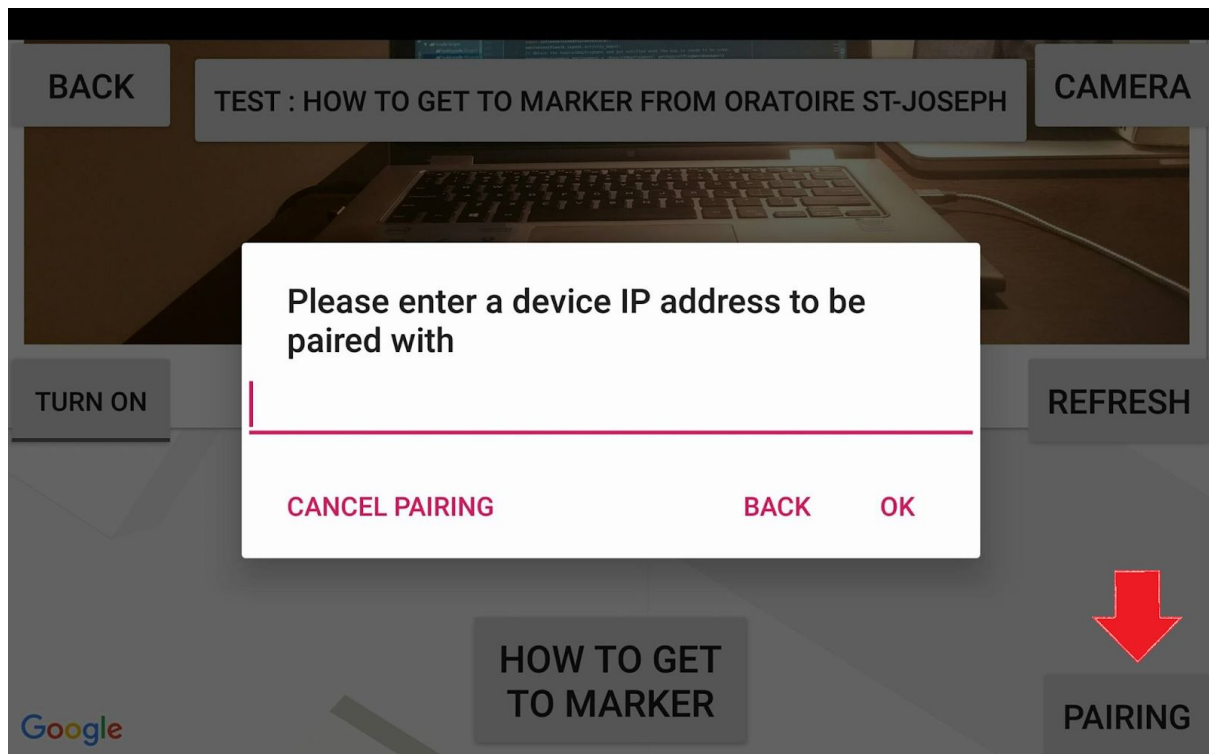


Fig. 4: Option de “Pairing” avec un autre utilisateur

Suite à l'entrée de l'adresse, chaque photo prise par l'utilisateur sera envoyée à l'appareil dont l'adresse IP est semblable. Si l'utilisateur principal désire recevoir les photos prises d'un autre appareil, l'utilisateur de ce dernier doit entrer l'adresse IP de l'utilisateur principal dans la même fenêtre.

La connexion entre deux appareils se fait par des sockets TCP. À l'ouverture de la carte Google, l'application démarre un service d'écoute des connexions par socket (FileListenerService) sur un thread séparée. L'écoute se fait sur le port 8988. Ainsi, si une adresse IP adéquate a été entrée préalablement, la prise de photo enverra des informations par socket au port 8988 de l'application qui écoute. Grâce à la classe FileTransferService (un IntentService, choisi puisqu'il s'agit d'un service à simple utilisation), l'application preneuse de photo ouvre un socket destiné à la bonne adresse et au port 8988 dans lequel elle y met l'image prise (sous forme de String obtenu grâce à l'image en Bitmap) et les coordonnées de géolocalisation, regroupés dans un objet JSON. À la réception de ce paquet, l'appareil récepteur place l'objet Picture extrait de l'objet JSON dans la base de données. Il suffit ensuite d'actualiser la carte par le bouton “Refresh” pour obtenir l'image reçue.

## Base de données SQLite

Pour ce qui est de la base de données locale, nous avons décidé de travailler avec SQLite, en passant par Room. Room permet d'avoir une couche d'abstraction au-dessus de SQLite afin d'en faciliter l'utilisation. Nous avons donc travaillé avec la base de données Room, ses "Data Access Objects" (DAO) et ses Entités.

Pour l'utilisation de la base de données, nous avons décidé de sauvegarder toutes les photos prises, encodées (Base64), en compagnie de leur localisation où elles ont été prises. C'est donc dire que chaque objet photo avait une latitude, une longitude et une "string" contenant l'encodage de la photo comme telle. Chaque fois que la map est ouverte, les photos sont reprises de la base de donnée afin de les replacer sur la map en compagnie d'un marqueur de position. Cette opération est aussi effectuée chaque fois que l'utilisateur appuie sur le bouton "Refresh", après réception d'images venant d'un autre utilisateur. Ces nouvelles images sont aussi sauvegardées dans la base de données et affichées sur la map avec leur marqueur où elle ont été prises.

Lors de l'affichage de ces photos, il y a décodage des images à partir de la base de données, passant des "string" Base64 à des Bitmap.

## Statistiques de l'application

À partir du menu principal, il est possible (pour les versions d'API le permettant), d'atteindre un affichage de statistiques de l'application.





Fig. 5: Vieille version (< API 21)

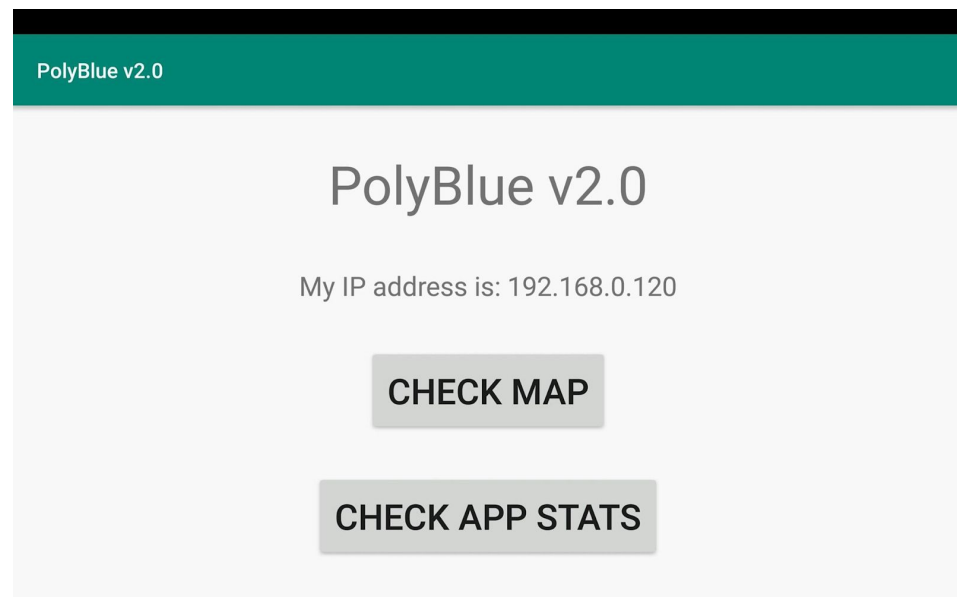
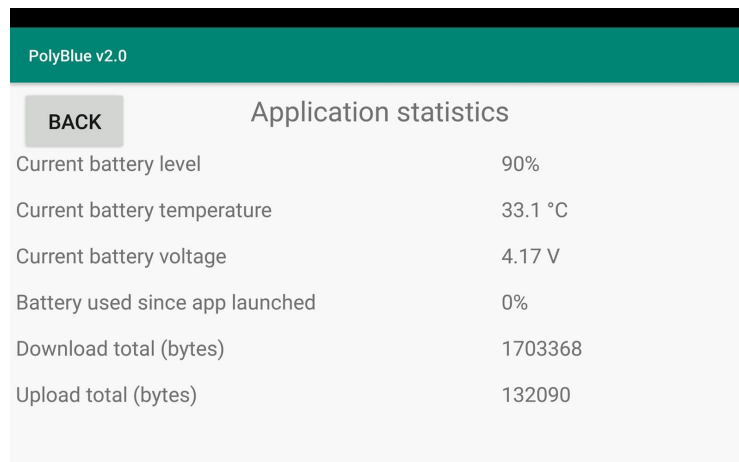


Fig. 6: Version plus récente (> API 21)

En effet, tel que demandé, il est possible de voir certaines stats:

- le niveau de batterie actuel
- **la température actuelle de la batterie** (un des deux capteurs demandés dans les fonctionnalités)
- voltage courant de la batterie
- pourcentage de la batterie utilisé depuis le lancement de l'application
- données sur la consommation de bande passante de l'app (*download/upload*)



The screenshot shows the 'PolyBlue v2.0' application interface. At the top, there is a teal header bar with the text 'PolyBlue v2.0'. Below the header, there is a 'BACK' button on the left and the title 'Application statistics' on the right. The main content area displays a list of statistics in a two-column format:

Application statistics	
Current battery level	90%
Current battery temperature	33.1 °C
Current battery voltage	4.17 V
Battery used since app launched	0%
Download total (bytes)	1703368
Upload total (bytes)	132090

Fig. 7: Vue sur les statistiques disponibles de l'application

La majorité de ces données (les 4 premières lignes) ont été obtenues à partir du "BatteryManager". Celui-ci est ce qui bloque les versions d'APIs < 21, puisqu'il est seulement accessible depuis LOLLIPOP (API 21). En lui passant certaines options, il est possible d'aller chercher les niveaux de batterie, de température et de voltage.

Pour ce qui est des données de consommation de la bande passante, la classe TrafficStats a été utilisée, et retourne les quantités de données téléchargées et téléversées.

## Difficultés rencontrées

Ce projet nous a amené une bonne quantité de difficultés. Malgré le fait que plusieurs des technologies utilisées nous étaient familières, nous avons rencontré de nombreux problèmes qui ont grandement retardé le processus de création.

Tout d'abord, le choix de la méthode de connexion sans fil a été laborieux. Nous pensions d'abord établir les connexions par Bluetooth, mais avons voulu expérimenter avec des technologies qui nous étaient moins familières. De ce fait, nous avons d'abord essayé de paier deux appareils par "Wifi peer-2-peer", puisque ce procédé semblait bien documenté et relativement facile à implémenter. Toutefois, nous ne réussissions pas à passer l'information adéquatement avec cette technologie. Nous essayions en effet de la jumeler avec l'utilisation de socket, mais cela nous a causé plus de problèmes que de bienfaits entre autres dû au fait que le "pairing" ne nous permettait pas d'obtenir l'adresse IP des autres appareils nécessaire pour les sockets, seulement l'adresse MAC. Nous avons finalement opté pour une approche purement WiFi avec des sockets TCP.

Ensuite, nous avons passer énormément de temps à régler les erreurs de “OutOfMemory”, qui survenaient lorsqu’on tentait de passer les images par socket et quand on chargeait la carte à partir de la base de données. Il a fallu expérimenter longuement avec les différents formats d’image. Ainsi, nous avons réduit la taille des images, diminué leur qualité et changé leur extension en JPEG au lieu de PNG afin d’obtenir des images claires mais pas trop volumineuses. Enfin, nous avons trouvé une manière d’augmenter la taille du *heap* alloué à l’application, ce qui nous a permis d’obtenir une application fiable qui peut manipuler une très grosse quantité d’images.

## Critiques et Améliorations

En grande partie, l’énoncé de ce projet final était ouvert et vague pour laisser une certaine flexibilité à toutes les équipes de faire ce qu’elles voulaient comme application finale. Ce fut un point positif, pour laisser chacune de celles-ci travailler et découvrir certains aspects moins travaillés par le passé. Avec une plus grande liberté pour l’implémentation des fonctionnalités demandées, il était aussi possible de mieux s’enligner dans une certaine direction claire pour nous, alors que dans le TP2 par exemple, il pouvait y avoir certaines ambiguïtés sur cet aspect. Un aspect intéressant de ce projet fut aussi la connection de plus d’un appareil, afin de faire un partage de données. Cela nous a permis de revoir nos connaissances requises pour cette fonctionnalité.

Pour ce qui est des améliorations de notre application, nous pourrions tout d’abord automatiser la réception des images venant d’un appareil “paire”. En ce moment, il faut “refresh” manuellement la carte pour faire apparaître les images des appareils connectés. De plus, il pourrait y avoir une liste de connections possibles, au lieu de la possibilité de “paire” un seul autre appareil pour l’envoi d’images. Enfin, nous pourrions utiliser plus que deux capteurs afin d’améliorer l’expérience utilisateur de l’application.

## Conclusion

En conclusion, ce projet final a été, d’après nous, très intéressant pour tout ce qui est de l’exploration de fonctionnalités. En effet, avec des instructions ouvertes comme dans cette situation, il fallait s’informer sur la manière que nous allions nous prendre pour implémenter le tout, avant de s’y mettre. La réutilisation de l’API de Google Maps a aussi été intéressante, et bien plus facile, alors que nous avions de l’expérience grâce au dernier TP.