# Rapport TP Wear

BOURGIN Jérémy

## 1. Les modules :

Lorsque l'on créé une application wear, Android Studio va créer un module pour le côté application mobile, et un module pour le côté application wear. Cependant, il n'y a pas de module pour le code partagé. J'ai donc créé un module afin que les 2 modules puissent partager du code en commun. Par exemple, les 2 applications partagent les entités, les fragments, les adapters, etc… Cependant, elles ne partagent pas les activités (car elles ne sont pas du même type et n'ont pas le même comportement), les services, et les implémentations de code qui leur sont propre.

## 2. Fonctionnalités implémentées :

- afficher la liste des messages (rogné en fonction si la taille de celui-ci est trop grand) : aucune difficulté
- rafraîchir la liste des messages : aucune difficulté
- envoie du message avec coordonnées GPS requises : il a été difficile de récupérer les coordonnées GPS à un moment donné précis. Pour palier à ce problème, je rafraichi les coordonnées GPS toutes les 2 secondes et récupère la dernière position enregistré afin d'être plus ou moins précis
- regarder les messages individuellement avec plus de détail : aucune difficulté
- quitter l'application à tout moment : bien gérer les services lancé
- mode ambient : aucune difficulté

## 3. Communication entre la montre et l'application

Pour ce TP, j'ai décidé de faire l'application sur le mobile et sur la montre (d'où l'importance du module partagé). Cependant, je ne voulais pas que la montre soit une simple copie de l'application mobile. Pour cela, c'est l'application mobile qui est responsable de récupérer et envoyer les messages au serveur (et non la montre). AInsi, lorsque la montre veut récupérer les messages ou en envoyer, elle doit passer par l'application mobile. Pour cela, l'application mobile lance un foreground service (pour lancer ce service il y a un bouton sur l'application) qui va se synchroniser avec la montre afin de pouvoir entrer en communication de façon bi-directionnel avec celle-ci. L'utilité du foreground service, c'est que l'utilisateur pourra fermer l'application, verrouiller son téléphone (et le mettre dans sa poche), et pouvoir profiter des fonctionnalités de la montre (qui pourra donc communiquer avec le foreground service). Cela implique un développement plus lourd côté mobile, mais lorsque l'on souhaite développer le côté wear, une fois tout le système en place (avec le bon éléments de code partagé et les services), cela va très vite.