Rapport projet
DRONE-GPS

Antoine Vidal-Mazuy
Antoine Huot-Marchand
Loïc Madern

Sommaire

Introduction

Ce projet a pour but de créer une application de montre et téléphone de drone GPS. Depuis la démocratisation du drone de loisir sont venues des réglementations. Dont certaines qui interdisent le vol de drone dans certaines zones pour des raisons de vie privée, d'écologie, etc…
Il nous est donc venue l'idée de créer une application de montre et de téléphone qui indiquent les zones où il est permis de voler. Ces applications sont interconnectées et les géo positions de l'utilisateur sont sauvegardées dans une base de données. Un client web existe également pour en consulter l'historique.

Fiche technique

Constructeur et intitulé de la cible : Samsung - Galaxy Watch 4 40mm

Alimentation:

- batterie : 247 mAh
- Durée d'utilisation : 40 heures

Mémoires :

- Mémoire vive : 1.5 Go
- Stockage interne : 16 Go

CPUs :

- CPU : Samsung Exynos W920 1.18Ghz 2 coeurs
- GPU : ARM Mali-G68

Modules de communications :

- Bluetooth 5.0
- WLAN
- NFC

Capteurs :

- Position : GPS, Beidou, Glonass, Galileo
- Altimètre Barométrique
- Accéléromètre
- Capteur d'impédance bioélectrique
- Capteur de fréquence cardiaque électronique ECG
- Moniteur cardiaque optique
- Capteur d'oxygène dans le sang
- Boussole
- Capteur de luminosité
- Gyroscope
- Tensiomètre
- Micro

Actionneurs :

- Ecran tactile
- Boutons physiques
- Haut-parleur

Système d'exploitation :

- Wear OS 3

Environnement de développement :

- Langage Kotlin :
  - Version : 1.7.10
  - IDE: Android Studio
- Langage Javascript :
  - Version : ES6
  - IDE : WebStom
- API Drone GPS :
  - URL : https://www.geoportail.gouv.fr

Nom **Caractéristiques téléphone**	**Loic**	**Antoine H**	**Antoine**
marque	xiaomi	Apple/Iphone SE	Apple/Iphone 13 Pro
système exploitation	android	IOS	IOS
version	11 RKQ1.201022.002	16.0	16.0
surcouche système d'exploitation	MIUI Global 12.0.1	/	/
accéléromètre	oui	oui	oui
boussole électronique	oui	oui	oui
capteur de lumière ambiante	oui	oui	oui

capteur de proximité	oui	oui	oui
gyroscope	oui	oui	oui
GPS	oui	oui	oui
Baromètre	non	oui	oui
Biométrie/ Touch Id	oui	oui	non
Reconnaissance faciale / Face Id	non	non	oui
micro	oui	oui	oui
caméra	oui	oui	oui
lidar	non	non	oui
doc	https://www.mobilewithdrivers.com/download-adb/xiaomi-redmi-note-10-64gb-4gb-ram https://www.tutoriels-android.net/2017/07/liste-des-codes-secrets-pour-xiaomi.html	https://support.apple.com/kb/SP820?locale=fr_FR	https://www.apple.com/sn/iphone-13-pro/specs/
plateforme pour coder	Android studio SDK 11	XCode	XCode

Compte tenu de cette fiche technique, nous avons remarqué que nos téléphones reposent sur différents OS au sein du groupe. De plus aucun de nos téléphones n'est de la marque Samsung et il nous a donc été impossible de lier correctement l'application physique avec la montre connectée et nos téléphones. Cependant, nous avons pu développer notre application multi-dispositifs à l'aide des émulateurs disponibles sur Android Studio.

Application téléphone

L'application déployée sur le téléphone permet à l'utilisateur de visualiser précisément les différentes zones pour lesquelles le vol de drone est autorisé. L'utilisation du téléphone nécessite de lier sa montre avec l'appareil.

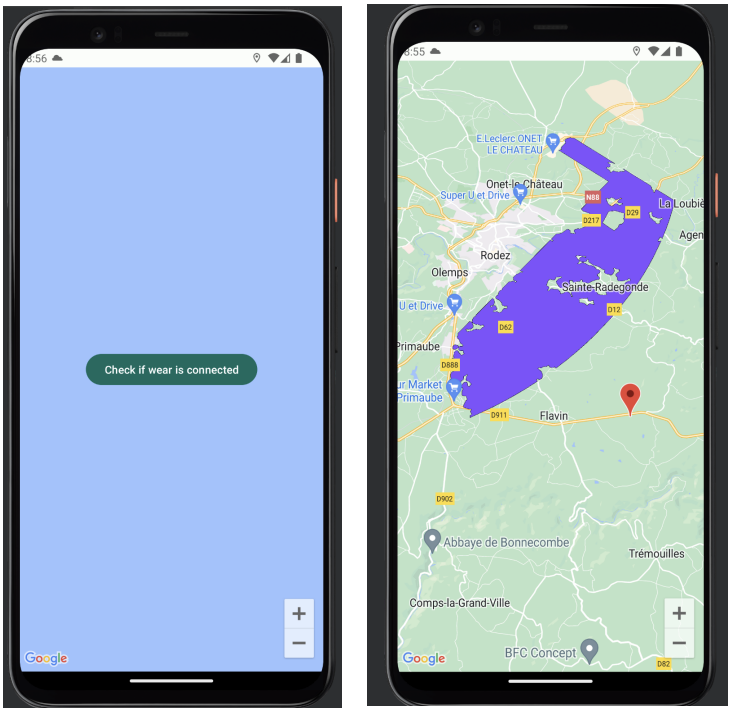
Pour cela, nous utilisons l'API GoogleMaps permettant de voir la carte et les différents points de repères géographiques. Il est possible d'adapter l'affichage de la carte selon la position de l'utilisateur (latitude, longitude) du device. Grâce au capteur GPS présent dans le téléphone, on peut très facilement récupérer ses données.

Afin d'afficher les zones de vol autorisés à proximité, il est nécessaire de faire une requête à une API dont l'utilisation est expliquée dans ce repository : https://github.com/cquest/drone_as_api

On peut, à l'aide des requêtes, connaître la liste des points géographiques où le vol de drone est possible selon un rayon et une hauteur de vol définie par l'utilisateur en plus de sa localisation.

Par la suite, il suffit de transformer la réponse au format geojson pour ainsi construire des polygones illustrant les différentes zones de vol sur GoogleMap.

Pour les différentes requêtes, la librairie **Fuel** a été utilisée. En outre, les différentes librairies de **GoogleMaps** comme **GeoJsonLayer** ont pu permettre l'affichage dynamique sur la carte via le téléphone.



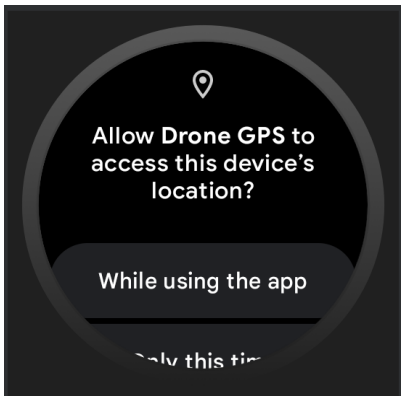
Application montre

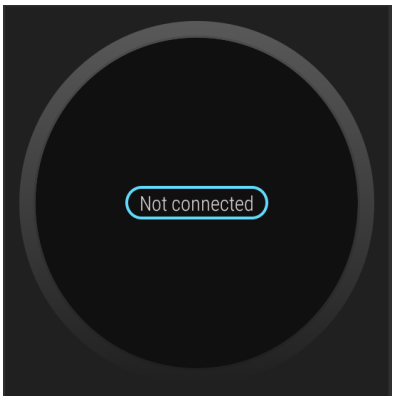
L'application de la montre est une application ambiante. L'affichage de la montre permet à l'utilisateur de savoir selon là où il se trouve, s'il à droit de voler en drone. Les différents écrans sont volontairement simples, car cela reste un support qui ne se contente que d'afficher les informations essentiels.

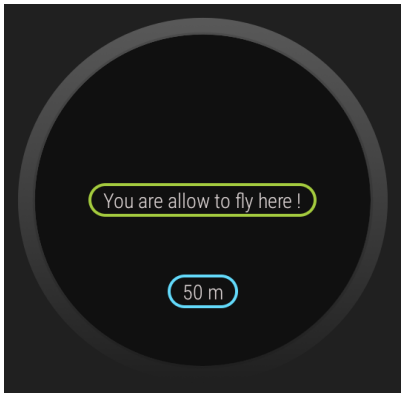
On retrouve les différentes informations strictement nécessaires au bon fonctionnement de l'application. 3 écrans différents sont proposés, un qui précise que l'application de la montre n'est pas connectée, et deux autres qui précisent si l'utilisateur à le droit de voler dans la zone où il se trouve.

La montre va de manière périodique, récupérer la position GPS et l'envoyer au téléphone. Les données sont traitées et envoyées par le téléphone. Si celles-ci sont utiles à la montre, c'est-à-dire qu'elles vont changer les conditions de vols de drone, le téléphone renverra les bonnes informations à la montre, et ainsi changer l'affichage. De plus cette localisation sera stockée dans la base de donnée en ligne MongoDB.

Afin d'économiser la batterie de la montre, l'application est développée en mode ambiant. Celle-ci se met en veille, mais continue à afficher les informations à l'utilisateur. Si l'application est quittée pendant l'exécution, à la réouverture, elle se connectera automatiquement au téléphone si celui-ci à toujours l'application en cours d'exécution.





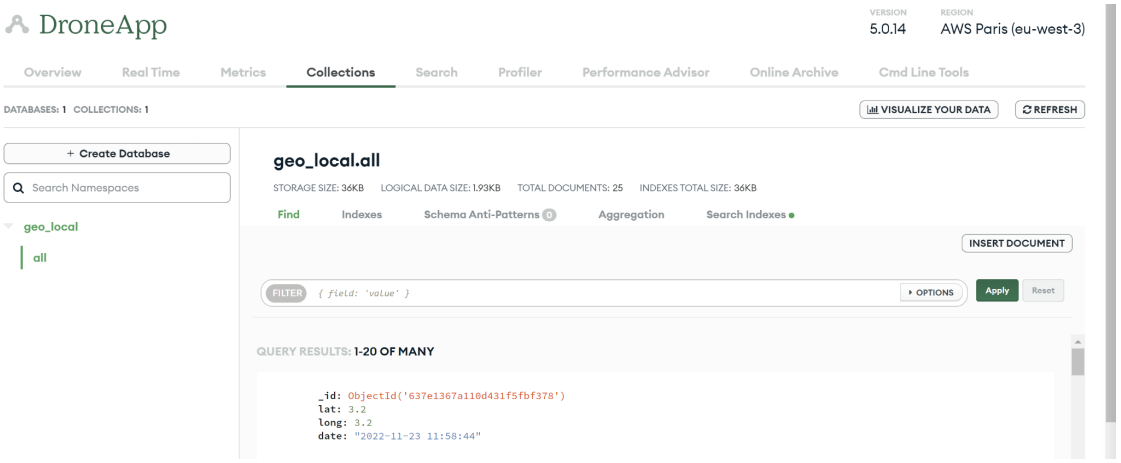


Client Base de données

Le type de base de données utilisée a été une base de données NoSQL MongoDB.
Les données sont envoyées depuis le téléphone à la base de données via un client kotlin grâce à la librairie **Fuel.** Chaque fois que la donnée est actualisée la donnée est stockée.Plus exactement la position ainsi que la date à laquelle la mise à jour a eu lieu.

On distingue 4 champs qui sont stockées dans la base de données :

- un champ **long** et un champ **lat** deux double qui correspond respectivement à la longitude et latitude de la position stockée.
- un champ string **date** qui indique à la seconde près la date à laquelle la position a été stockée.
- un champ **id** pour identifier l'objet stocké lui-même.

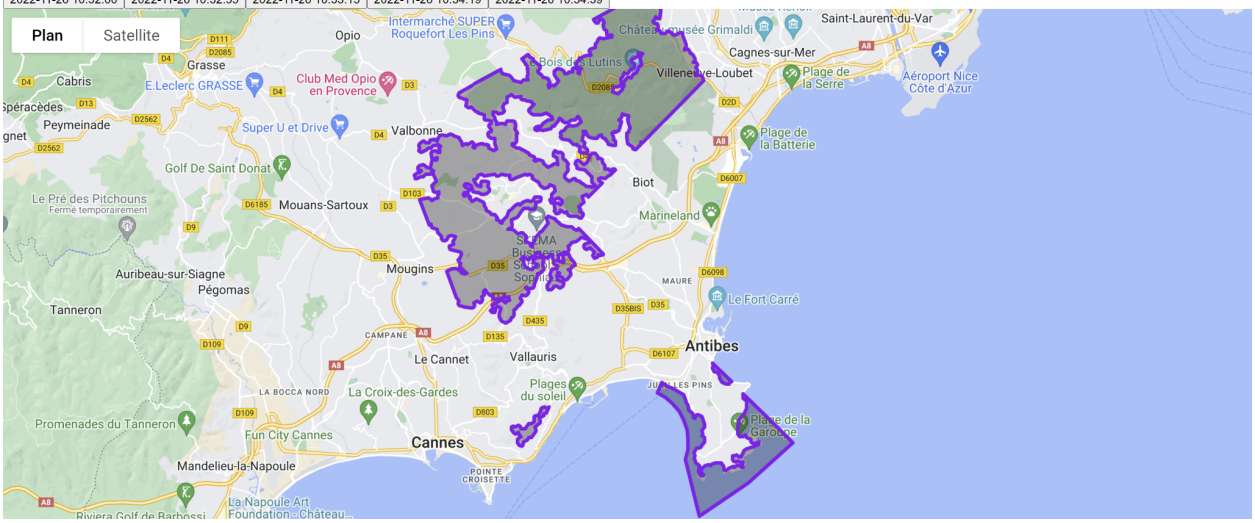


Enfin en ce qui concerne le client web de la base de données, il s'agit d'une simple page web qui affiche les dates des positions stockées. Il suffit de cliquer sur une date donnée pour afficher les zones sur volables de la position correspondante.

Historique des lieux par date

Cliquer sur une date pour afficher le lieu correspondant

2022-11-23 11:58:44	2022-11-23 13:36:11	2022-11-23 14:12:46	2022-11-25 18:31:58	2022-11-25 18:32:28	2022-11-25 20:18:13	2022-11-25 20:19:03	2022-11-25 20:19:13	2022-11-25 20:19:33	2022-11-25 20:19:53
2022-11-26 09:49:13	2022-11-26 09:54:19	2022-11-26 09:54:59	2022-11-26 10:00:01	2022-11-26 10:02:20	2022-11-26 10:03:29	2022-11-26 10:05:39	2022-11-26 10:28:29	2022-11-26 10:28:54	2022-11-26 10:31:30
2022-11-26 10:32:00	2022-11-26 10:32:55	2022-11-26 10:33:15	2022-11-26 10:34:19	2022-11-26 10:34:39					



Répartition du travail

Le travail a été équitablement réparti notamment en trois parties:

- Une partie application montre et envoie des données depuis le tel sur la montre. faite par Antoine Vidal-Mazuy.

- Une partie base de données ou les données sont stockées, et consommées par un client web réalisée par Loïc Madern.

- Une partie récupération des données geojson et affichage via une application Google Map pour indiquer les zones volables faite par Antoine Huot-Marchand.

Toutes ces parties indépendamment développées ont été fusionnées pour aboutir à la réalisation finale du projet.

Conclusion

Ce projet nous a permis de développer avec Kotlin un langage récemment découvert. Nous avons pu découvrir un nouvel environnement notamment avec de nouvelles librairies comme Fuel et une nouvelles façons de coder.
Développer ces applications nous a appris à faire communiquer deux appareils entre eux, notamment entre le téléphone et la montre connectée. Enfin nous nous sommes familiarisés avec le stockage de données relevées depuis un capteur.