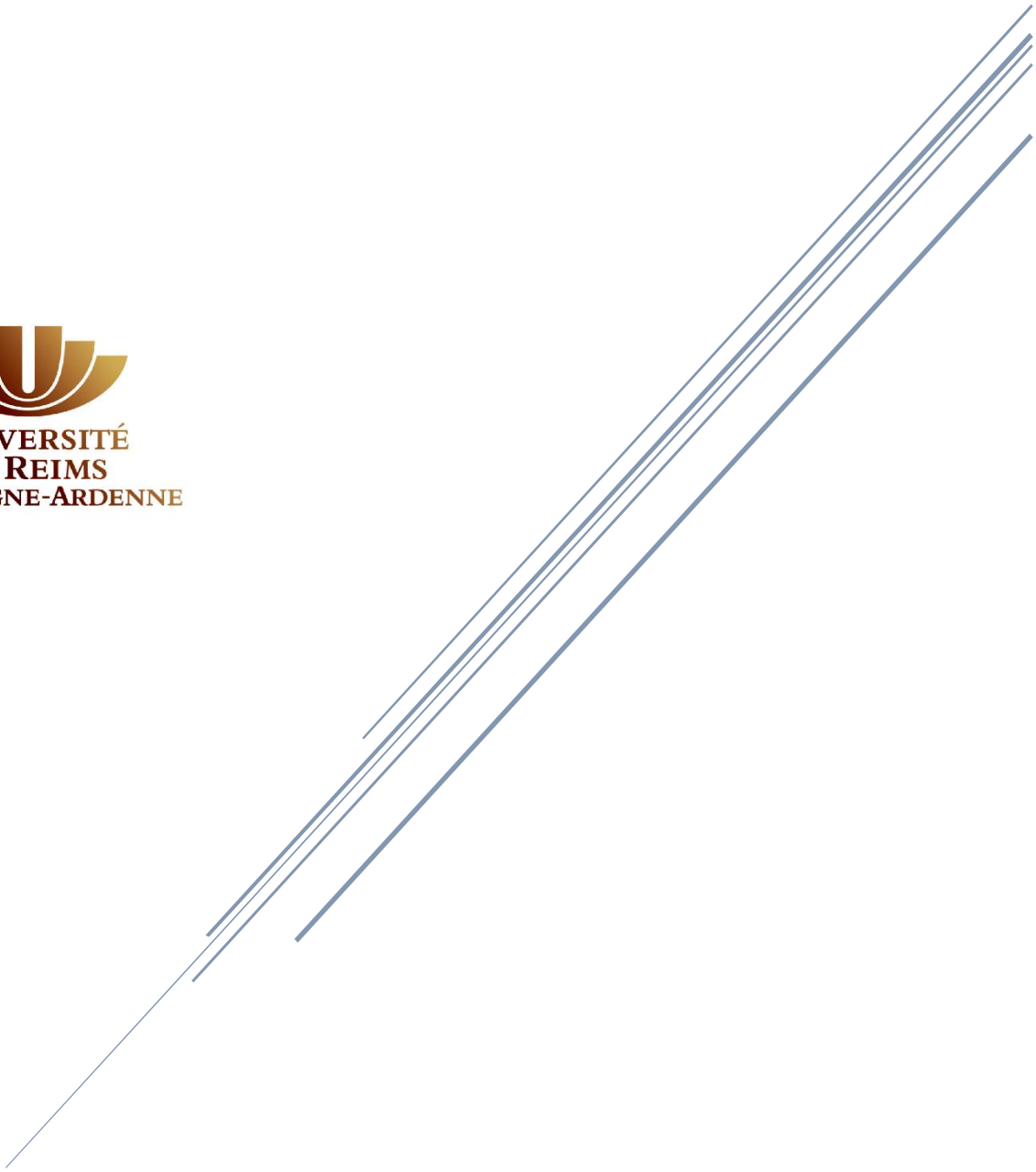
RAPPORT PROJET INFO0806

Application Android pour la détection et l'analyse d'antenne 4G



Soitel Rémi
INFO0807, M1 INFO

Table des matières

I. Introduction

Dans le cadre des matières INFO0806 et des cours de communication en entreprises, il nous a été demandé de coder une application Android avec son Dashboard attitré avec un sujet que l'on pouvait choisir. Dans notre cas nous avons choisis le sujet sur les signaux des antennes 4G/5G (On se concentrera sur les antennes 4G car nous n'avons pas la 5G).

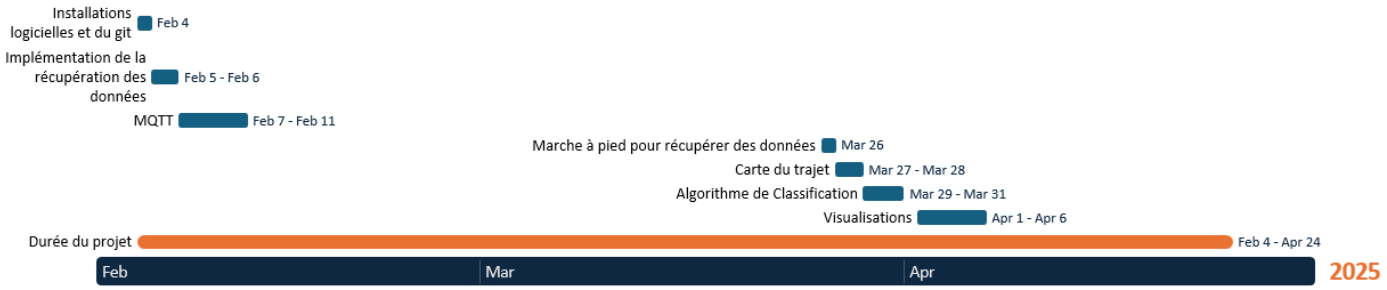
L'objectif principal de ce projet était de concevoir une solution complète de collecte, transmission et visualisation de données liées à la aux signaux 4G capté par un smartphone/tablette. Le projet était donc séparer en 2 points distincts: le développement d'une application Android capable de récupérer et transmettre régulièrement des données comme les coordonnées, la puissance du signal, et autres capteurs; et la création d'un Dashboard permettant d'analyser ces données en temps réel ou en différé avec des méthodes de visualisations. Nous pouvons ajouter aussi qu'il fallait implémenter côté Android une récupération des données via un serveur MQTT et côté Dashboard l'utilisation d'un algorithme de classification qui nous a été fourni.

Bien que ce projet fût réalisé en autonomie, ce projet a été structuré comme un travail en entreprise en intégrant des outils et des méthodes issus du monde de l'entreprise. C'est pour cela que dans ce rapport nous y détailleront le processus de planification de ce projet et de si le plan a été respecter du début à la fin.

Ce rapport résume les différentes étapes de la réalisation du projet, nous y verrons les étapes de planification avant le début du projet pour ensuite voir comment l'application Android fut réaliser pour ensuite voir comment le Dashboard a été codé pour enfin conclure sur tout ça.

II. Planification

Avant de commencer, nous avions planifier le projet pour savoir comment nous allions le réaliser, pour cela, nous nous somme mis d'accord de diviser le projet en 2 points distincts, c'est-à-dire une partie Android Studio et une partie Dashboard streamlit en python. Pour séparer ces deux parties, nous avions établies un diagramme de Gant que voici :



Comme on peut le voir, nous avons séparer le projet en 2 sprints qui chacun définit les 2 parties du projet précédemment énoncé et si le deuxième sprint commencer aussi loin du premier, c'est parce que d'après nos emplois du temps c'était le seul moment où l'on pouvait être libre une journée pour récupérer les données car nous n'habitons pas à Reims de base.

Cette planification étant faite, nous avons lister les élément nécessaire à l'accomplissement du projet, pour cela nous avons installer le logiciel Android Studio pour coder l'application et les différente bibliothèque python nécessaire pour le Dashboard ainsi que l'implémentation du GitHub pour mieux gérer le projet (Consultable ici).

Une fois toute cette mise en place effectuer, il était temps de commencer le projet avec l'implémentation de l'application Android.

III. Codage de l'Application Android

A. Récupération des Données

Il était temps comme indiquer sur le diagramme de Gant d'implémenter la récupération des données via les différent capteurs disponible sur notre téléphone. Pour cela nous avons créé une activité vide sur Android Studio nommé MainActivity.kt et ensuite on commence par initialiser chaque valeur que l'on veut récupérer comme la latitude et la longitude pour les coordonnées où encore la vitesse en km/h que le téléphone parcourt. Pour récupérer les infos des antennes 4G nous avions dû implémenter une bibliothèque Android appelé TelephonyManager qui nous retournera toute les infos nécessaire c'est-à-dire le eNbID qui est l'id de l'antenne, le CellID qui est l'id du téléphone, le TAC et le Signal transmis avec l'antenne. Pour que ces méthodes fonctionnent il a fallu aussi modifier le manifest du projet Android en ajoutant ces lignes :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

Elles permettent tous simplement d'autoriser ou de demander des autorisations pour accéder à certaines fonctionnalité, on y retrouve surtout pour notre projet des demande de localisation et des infos internet.

Après ça, nous créons la méthodes OnCreate qui est appeler dès que l'application est lancé, elle permet de demander les permissions nécessaire aux bons fonctionnement de l'application notamment avec des méthodes comme RequestPermissions et requestPermissionsIfNeeded. Mais OnCreate permet surtout d'initialiser et mettre à jour les différente données récupérer par les capteurs. Pour cela on utilise une liste de String collectedData dans lequel on y insère chaque valeur récupérer toutes les 3 secondes. Pour la majorité des valeurs juste une méthode Getter de base suffisait mais pour les données internet et des antennes 4G il fallait coder nos propres Getter. Nous avions donc codé getNetworkStats qui récupère les valeurs d'upload et de download totaux de nos capteur et ensuite on fait un calcul de moyenne de ces données pour avoir une vitesse d'upload et de download mise-à-jour toutes les 3 secondes toujours; Puis nous avons fait getCellTowerInfo qui utilise la bibliothèque précédemment mentionner pour récupérer les infos des antennes 4G.

Une fois ceci fait nous avions fait l'affichage de l'application avec la méthode DataScreen. Ici on vérifie si les permissions sont bien accessible et on y affiche les différentes données capter en temps réels comme nous pouvons le voir sur l'image à côté. Mais maintenant que nous avons nos valeurs il faut trouver un moyen de les récupérer et pour cela nous avant de faire le MQTT nous avons pensé à exporter un .csv qui aurait pour chaque ligne la valeurs de nos capteurs toutes les 3 secondes toujours. Pour cela on utilise une méthode sauvegarderCSV qui défini le nom de fichier donnees_capteurs.csv et qui exportent toutes les données du csv dans le dossier Téléchargement du téléphone et grâce à collectedData initialiser précédemment, on y met dans le csv toutes les données récupérer depuis le lancement de l'application. Il nous suffisait ensuite d'ajouter le bouton pour télécharger le csv dans la méthode DataScreen et l'on peut enfin récupérer les données retourner par nos capteurs. Il est a noté que l'application ne nous a pas pris comme prévu 2 jours mais plutôt 4 jours.

L'application Android étant finaliser, il nous rester plus qu'a implémenter le côté MQTT de l'application que nous allons voir maintenant.

B. Implémentation du MQTT

Dans ce projet l'une des deux conditions obligatoire était d'implémenter un échange des données récupérer avec un serveur MQTT. MQTT étant un système qui s'implémente en python, nous avons donc ajouter au sein de notre projet un fichier script.py qui implémentera le bon fonctionnement du serveur.

La première galère que nous avons dû faire était d'implémenter la possibilité d'exécuter des scripts python dans notre application. Pour cela on a utilisé la bibliothèque Chaquopy qui permet de démarrer l'interpréteur python dans notre application, on modifie donc la méthode OnCreate pour démarrer une instance python avec comme module « script » qui est le nom de notre fichier, puis on envoie un Log.d pour voir si la liaison se fait bel et bien. Après ça nous pouvons commencer le codage de script.py.

Dans notre script.py, on commence évidemment à implémenter la bibliothèque MQTT car c'est une bibliothèque qui permet très simplement des connexions à des serveurs. On initialise ensuite l'adresse IP dit Broker du serveur avec son port et son topic (ici « antenne ») et après ça on connecte notre client. On écrit donc une méthode on_connect qui va « subscribe » le client à son topic. Il ne nous reste plus qu'à faire la gestion des messages, pour cela nous avons 3 méthodes, on_publish qui une fois un message est envoyé le fait notifier via un print; on_message qui récupère le dernier message et le montre; et enfin nous avons la méthode send_mqtt_message qui renvoie les données sous forme d'un JSON. Et avec tout ça nous avons un MQTT fonctionnelle, il suffit de finir la logique dans notre application Android.

Pour finir l'implémentation, on créer la méthode envoieLigne qui prend nos données, les converties en JSON et les envoie aux serveur MQTT et cette méthode est appeler dans OnCreate toutes les 3 secondes. Et avec ça la gestion de message MQTT est terminer et nous pouvons donc voir les logs pour voir que tout fonctionne avec un affichage du message en entier.

```
--------------------------- PROCESS STARTED (12937) for package com.example.info0806projet ---------------------------
2025-04-12 15:05:03.928 12937-12937 nativeloader            com.example.info0806projet           D  Load /data/app/~~dxxGCOeYSnemX-G5y6CSaQ==/com.example.info0806projet-EVucdc
2025-04-12 15:05:03.930 12937-12937 nativeloader            com.example.info0806projet           D  Load /data/app/~~dxxGCOeYSnemX-G5y6CSaQ==/com.example.info0806projet-EVucdc
2025-04-12 15:05:03.932 12937-12937 nativeloader            com.example.info0806projet           D  Load /data/app/~~dxxGCOeYSnemX-G5y6CSaQ==/com.example.info0806projet-EVucdc
2025-04-12 15:05:03.935 12937-12937 nativeloader            com.example.info0806projet           D  Load /data/app/~~dxxGCOeYSnemX-G5y6CSaQ==/com.example.info0806projet-EVucdc
2025-04-12 15:05:04.476 12937-12937 python.stderr           com.example.info0806projet           W  /data/data/com.example.info0806projet/files/chaquopy/AssetFinder/app/script
2025-04-12 15:05:04.539 12937-12996 python.stdout           com.example.info0806projet           I  Connexion au broker réussie
2025-04-12 15:05:04.893 12937-12996 python.stdout           com.example.info0806projet           I  Message publié avec succès
2025-04-12 15:05:04.913 12937-12996 python.stdout           com.example.info0806projet           I  Message reçu : "{\"Temps\":1744463104882,\"Temps Lisible\":\"2025-04-12 15:
2025-04-12 15:05:07.895 12937-12996 python.stdout           com.example.info0806projet           I  Message publié avec succès
2025-04-12 15:05:07.906 12937-12996 python.stdout           com.example.info0806projet           I  Message reçu : "{\"Temps\":1744463107892,\"Temps Lisible\":\"2025-04-12 15:
2025-04-12 15:05:10.918 12937-12996 python.stdout           com.example.info0806projet           I  Message publié avec succès
2025-04-12 15:05:10.930 12937-12996 python.stdout           com.example.info0806projet           I  Message reçu : "{\"Temps\":1744463110916,\"Temps Lisible\":\"2025-04-12 15:
2025-04-12 15:05:13.928 12937-12996 python.stdout           com.example.info0806projet           I  Message publié avec succès
2025-04-12 15:05:13.966 12937-12996 python.stdout           com.example.info0806projet           I  Message reçu : "{\"Temps\":1744463113926,\"Temps Lisible\":\"2025-04-12 15:
2025-04-12 15:05:16.941 12937-12996 python.stdout           com.example.info0806projet           I  Message publié avec succès
2025-04-12 15:05:16.962 12937-12996 python.stdout           com.example.info0806projet           I  Message reçu : "{\"Temps\":1744463116938,\"Temps Lisible\":\"2025-04-12 15:
```

Finalement nous qui avons initialiser 5 jours pour le MQTT ça s'est fait en 2 jours. Maintenant que l'application est finalisé, nous pouvons donc récupérer nos données et ainsi coder notre Dashboard de visualisation des données.

IV. Codage du Dashboard

A. Carte du Trajet

Nous sommes arrivé aux codage du Dashboard, mais avant ça nous nous somme réservez comme nous le voyons dans le diagramme de Gant la journée du 26 mars 2025 pour récupérer les données, sauf que la mise-à-jour des antennes 4G ne se faisait pas sur le csv. Heureusement que nous avions le lendemain complet aussi pour récupérer les données.

Une fois nos données en poche dans notre fichier, nous commençons à implémenter le Dashboard avec d'abord la lecture du csv pour récupérer toutes les données qui seront ranger dans une liste de valeur de session nommé « data », pour cela on a une méthode load_csv_data qui charge le csv et range les données dans cette variable data. Une fois nos données charger, nous allons donc implémenter la carte de notre trajet avec ses antennes 4G, pour cela on récupère toutes les données de latitude et de longitude et on ajoute un point bleu sur une carte avec la bibliothèque folium. Ensuite on a ajouté un marqueur rouge à chaque fois que dans le csv on s'est connecter à une autre antenne, en cliquant sur le marqueur on peut aussi y voir les informations de l'antenne. Après ça on veut afficher les antennes 4G en eu même, nous avons donc créer un dictionnaire avec les antennes 4G aux alentours de Reims, dans ce dictionnaire on y retrouve pour chaque antenne son eNbID, son TAC, son adresse, sa latitude et sa longitude. Une fois le dictionnaire initialiser et rempli à la main, on affiche des marqueur verts pour chaque antennes sur la map avec leurs informations. Pour finir sur la carte du trajet on ajoute une ligne pointiller violète entres les marqueurs rouges et verts pour mieux voir à quel nouvel antennes

on s'est connecter et nous avons donc ce résultats visible dans l'image à droite. Pour pouvoir ouvrir le Dashboard, il suffit de se mettre dans le dossier ou se trouve dashboard.py et d'exécuter la commande sur un cmd « streamlit run dashboard.py » (Trouvable dans le dossier Info0806Projet \app\src\main\python).

Maintenant que nous avons implémenter une carte du trajet dans les temps que nous avons initialiser et avant de commencer à faire des graphiques de visualisations, nous pouvons déjà réfléchir à comment utiliser l'algorithme de classification.

B. Implémentation de l'Algorithme de Classification

Avant d'implémenter l'algorithme, c'est quel genre d'algorithme de classification ? C'est un algorithme de classification dériver du célèbre algorithme des k plus proches voisins (k-NN) ici appelé k-Iterative Neighbors (kIN). Son but est de classer un point cible en fonction de point déjà étiquetés dans une base de données en se basant sur leur proximité dans un espace de caractéristique. Cette algorithme est divisé en 3 étapes principales :

-Mesure de la distance entre le point cible et tous les autres.

-Sélection des k les plus proches de cette distance.

-Attribution d'une classe au point cible en fonction de la classe majoritaire parmi ses k voisins.

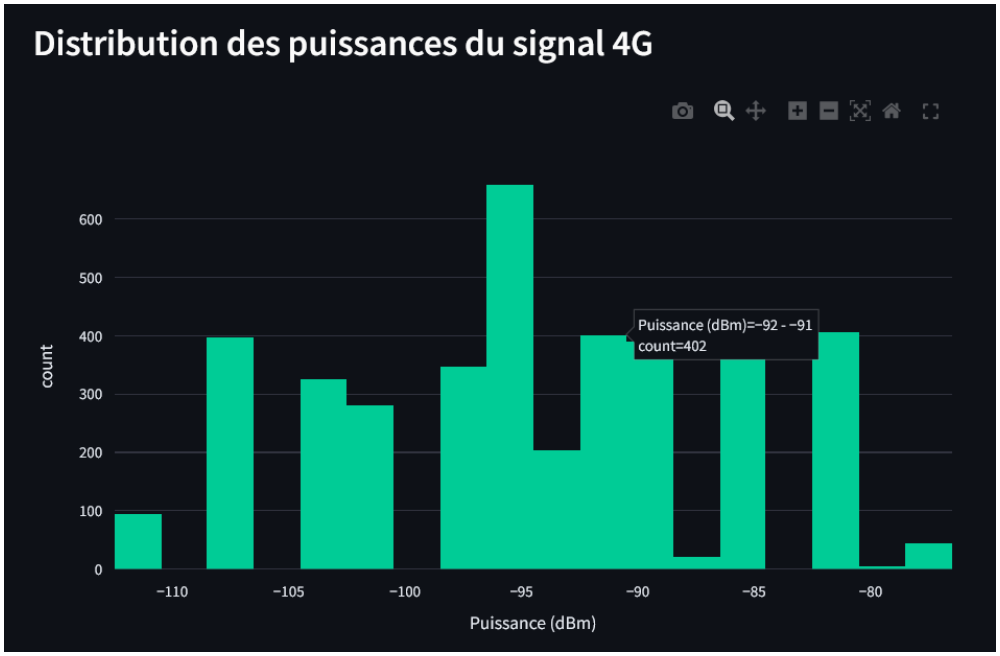
Il est à préciser que la distance qui est utilisé dans cette algorithme est la distance de Minkowski. Et avec ses infos on peut coder une classification des signaux des antennes 4G pour savoir quel endroit du trajet et quels antennes captes le mieux leurs signaux. Pour cela on implémente l'algorithme avec d'abord une méthode distance_minkowski qui retourne la fameuse distance et ensuite on a kIN_classification qui implémente l'algorithme. La fonction prend en paramètre le DataFrame contenant les données, le point_cible qui sera un signal aléatoire dans le jeu de données, k qui est le nombre de voisin pris en compte (Ici 5) et m qui est le paramètre de distance (Ici 2). Une fois l'algorithme implémenter nous pouvons afficher ses résultats avec une cartes similaire à celui implémenter précédemment mais à la place de juste mettre des points bleus pour représenter le trajet, on mettre des points de couleurs diviser en 3 niveaux : Rouge pour un mauvais signal, orange pour un signal moyen et vert pour un bon signal. Le rangement des signaux se fait avec l'algorithme et nous retournes donc ce résultats visible dans l'image de gauche. Cela nous permet de voir sans trop de surprise que les signaux vers le centre-ville sont mauvais et que plus on est loin de celui-ci, plus le signal est meilleurs. Et l'on peut remarquer que ce n'est pas parce que nous somme proches d'une antenne que le signal est meilleur (Y'a des endroits ou c'est vrai mais ce n'est clairement pas souvent le cas). Nous pouvons aussi voir que même en passant à côté d'une antenne 4G, ça ne déclenche pas forcément la connexion vers celle-

ci, même si le signal est mauvais (Nous pouvons le voir en bas à gauche de la carte). Nous pouvons noter que cette partie du projet à pu être implémenter dans les temps énoncé par notre diagramme.

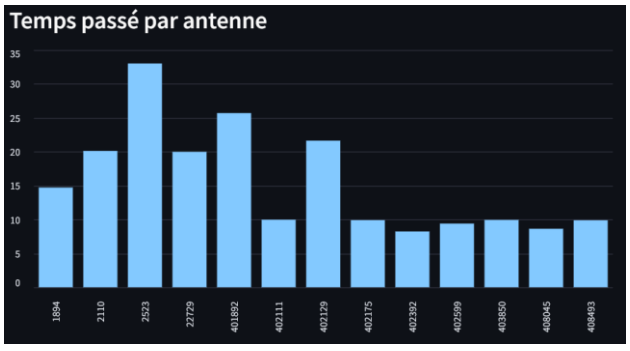
Et pour mieux analyser nos résultats, nous allons implémenter des graphique pour visualiser nos données.

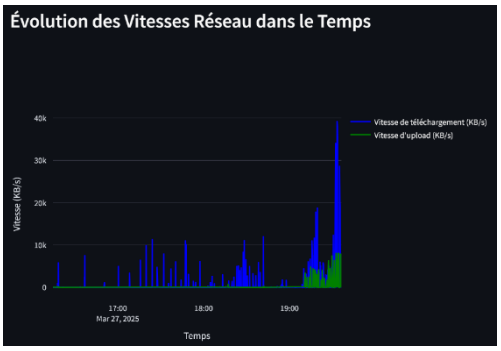
C. Visualisations

Il était temps d'implémenter des graphiques pour mieux observé nos données, nous avons commencé comme premier graphique qui nous semblez pertinent avec un histogramme de la puissance du signal avec le nombre de fois qu'ils apparaissent. Quand on passe la souris sur une des barres on peut voir plus précisément les données (Ici la puissance et le nombre de fois qu'il apparait).



Nous pouvons voir que le signal moyen est entre -96 et -95 mais qu'en globalité, les différent signaux apparaissent avec à peu près le même nombre de coups sauf certains cas assez extrême. En autre graphique implémenter nous avons aussi mis le temps passé par antennes et un graphique pour afficher les vitesse de téléchargement et d'upload :

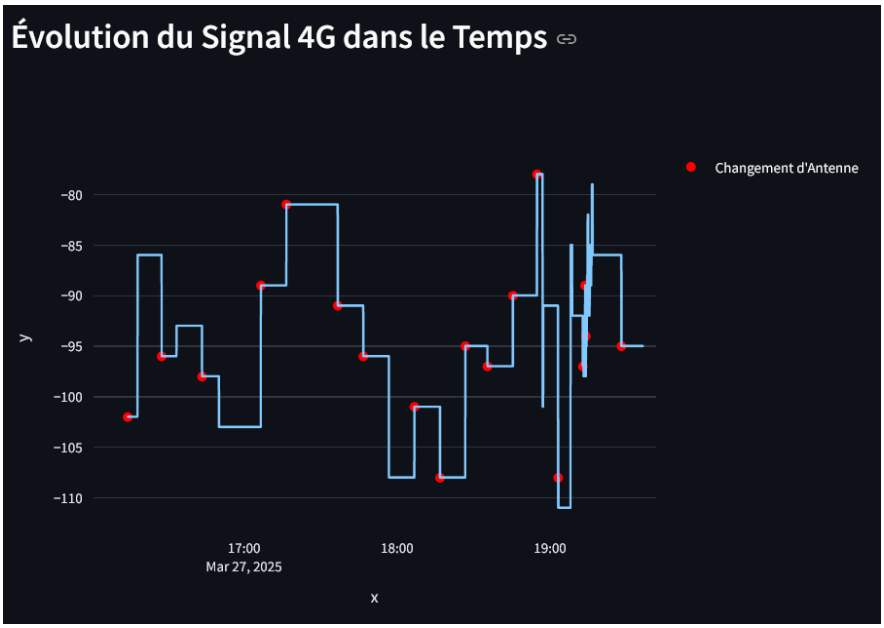




Au cours de mon trajet j'ai fait des speed test sur le speed test de google car c'était un des seuls qui pouvait s'exécuter en arrière-plan sur mon téléphone, chaque barre bleu c'est quand je lancer un speed test, or on voit qu'à la fin de ce graphique les pics explosent, c'est surement une erreur ou du moins c'est une autre méthode car j'avais tellement fait de demande de speed test que Google ne voulait pas que j'en fasse plus, alors je suis aller sur un autre site mais qui sembler télécharger beaucoup plus de données, donc les derniers pics sont à prendre avec des pincettes. Nous pouvons

voir sur le graphique de gauche que l'antenne que j'ai passer le plus de temps c'est celui avec l'eNbID 2523 car je m'y connecte 2 fois à celle-ci et qu'elle semble occuper sur la carte une large zone.

Enfin nous avons implémenter un dernier graphique qui est l'évolution du signal au cours du trajet que voici :



Quand on passe la souris sur un point rouge on peut voir c'est quel antenne suivit de ses coordonnées et quand on passe la souris sur un point de la courbe on voit la date et l'heure, le signal et ses coordonnées. Nous pouvons facilement voir que l'antenne la plus rapide et stable et la 2110 qui est l'antenne la plus à gauche de la carte et qui était afficher en vert du début à la fin par l'algo de classification, on peut dire que cette antenne 4G est la meilleur de la ville. A l'inverse la pire antenne c'est-à-dire celle avec un mauvais signal et qui est instable semble soit être la 402392 ou la 401892. Mais l'antenne qui est stable avec le moins bon signal est la 408493.

En termes de délai de planification, cette étape fut relativement rapide et à pu être compléter en 2 jours au lieu des 5 annoncés.

V. Conclusion

Ce projet nous a permis de mettre en œuvre une solution complète allant de la collecte de données sur Android jusqu'à leur visualisation et leur analyse sur un Dashboard en Python. En développant une application Android capable de capter et transmettre régulièrement des informations telles que le GPS, le signal des antennes 4G, ou encore les vitesses d'upload et de download, nous avons pu générer un jeu de données exploitable en conditions réelles.

L'intégration du protocole MQTT via Chaquopy a offert une nouvelle opportunité concrète d'exploiter des échanges de données en temps réel entre l'application et un serveur. Côté Dashboard, l'implémentation d'une carte interactive, d'un algorithme de classification k-Iterative Neighbors et de graphiques pertinents nous a permis d'analyser efficacement la qualité des signaux et les comportements des antennes tout au long du trajet.

En respectant au mieux notre planification initiale, nous avons pu livrer un projet structuré, complet et fidèle aux attentes pédagogiques. Cette expérience nous a également permis de nous confronter à une certaine gestion de projet similaire à celle du monde professionnel, bien que nous

soyons seul dans ce projet. Pour conclure, ce projet a été très formateur, aussi bien sur le plan technique que dans la manière d'organiser le travail.