AirWatcher

Design Document

Rédigé par :

Adam Chellaoui

Rim Benzekri

Damien-Joseph Rispal

Alexis Metwalli

Date de création : 03/05/2021
Date de dernière modification : 28/05/2021

Table des matières

I. Architecture choisie

Pour le développement de notre application nous avons choisi d'utiliser l'architecture par couche MVC (Model-View-Controller). C'est une architecture qui sépare le programme en 3 composants logiques qui interagissent entre eux : Model, View et Controller.

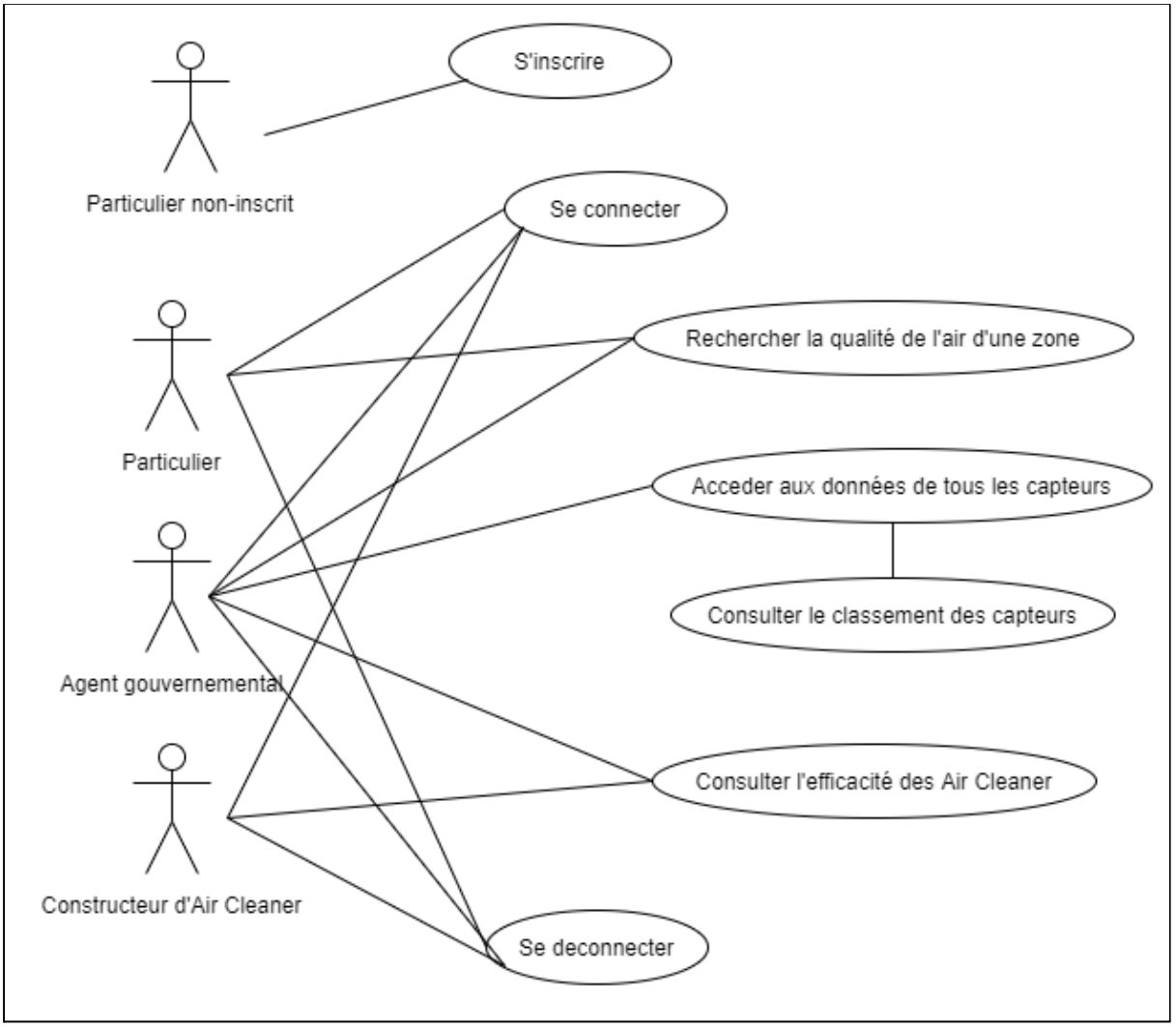
Ce choix a été motivé par la flexibilité octroyée par l'architecture MVC pour effectuer des changements indépendamment de la présentation. Ainsi, si notre application venait à offrir une interface de visualisation graphique à l'utilisateur, alors il ne serait nécessaire de modifier que le code de la classe View.

C'est cet aspect modulaire de l'architecture MVC qui nous a séduit et motivé dans notre choix. L'organisation en package permet de plus d'organiser le code de manière claire et efficace, ce qui facilite sa compréhension et son explication à des personnes extérieures.

II. Choix de l'IDE et du langage

Pour le développement de notre application nous avons choisi d'utiliser l'IDE Visual Studio Code. Ce choix a été motivé par la qualité de l'interface qu'elle propose. Elle nous permettra de visualiser les packages qui structurent notre code, et les classes qu'ils contiennent en permanence. C'est de plus un IDE qui supporte le langage C++ qui est le langage qui nous a été imposé d'utiliser. Ce langage connu pour sa complexité, pourra être débuggé plus facilement avec le terminal de commandes intégré à Visual Studio .

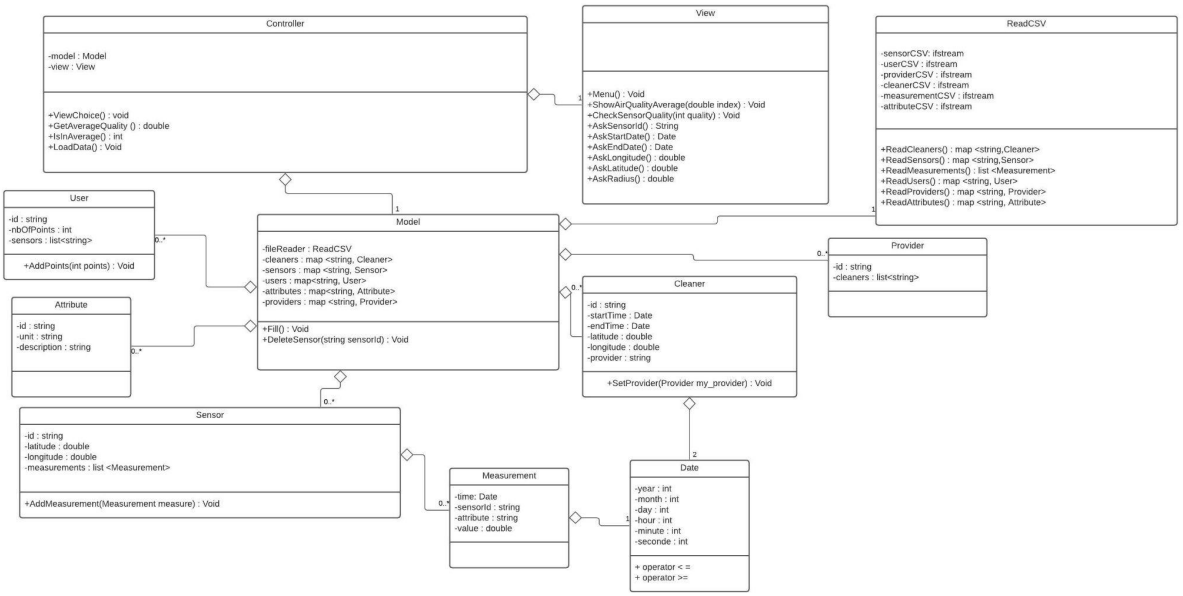
III. Diagramme des cas d'Utilisation - Use case Diagram



Use Case Diagram - "AirWatcher"

IV. Diagramme des classes

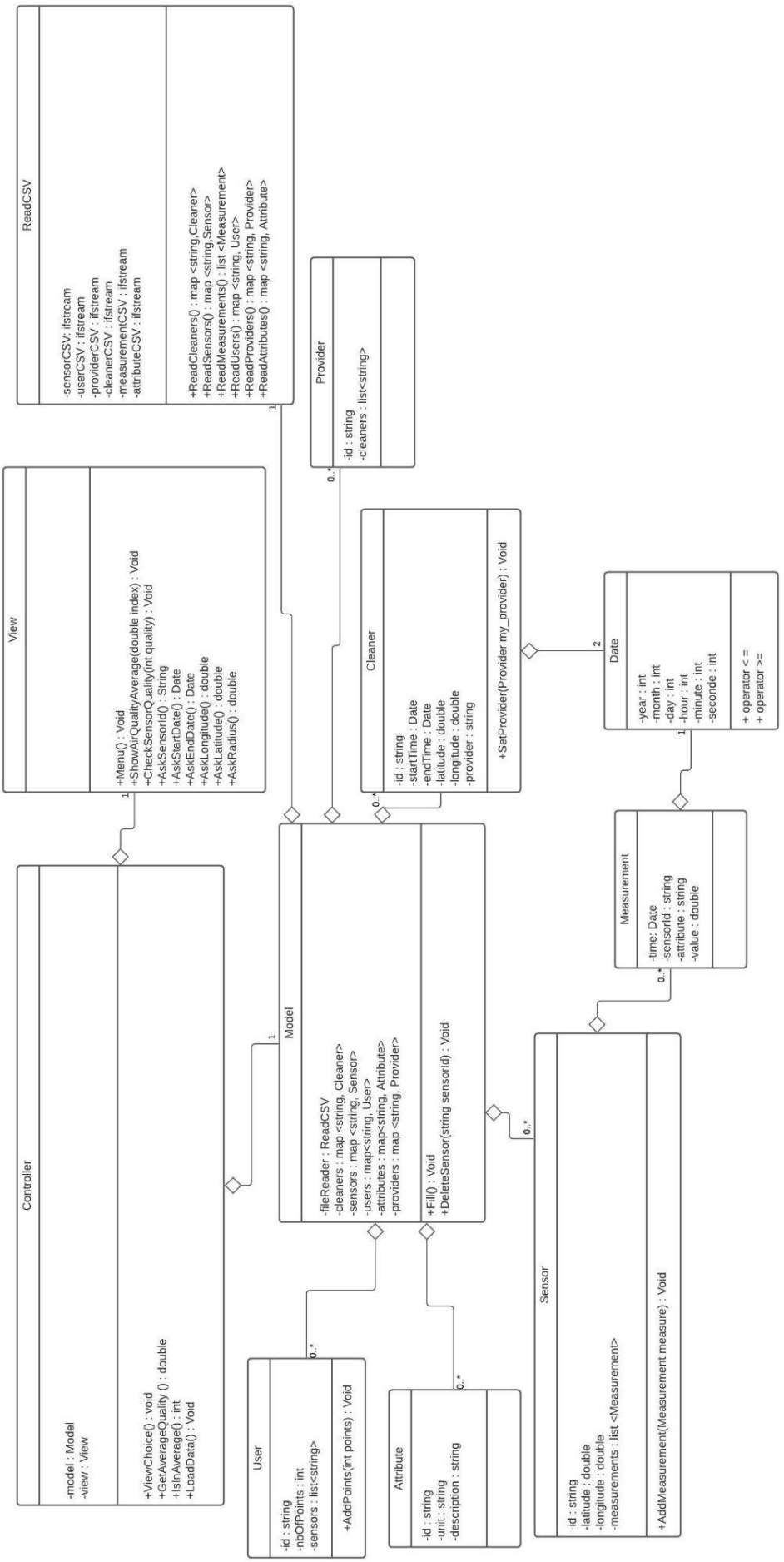
Diagramme UML des classes



Ainsi, sur notre diagramme de classes, on retrouve bien les choix d'architectures qui ont été faits au préalable. Les classes sont regroupées en 3 packages :

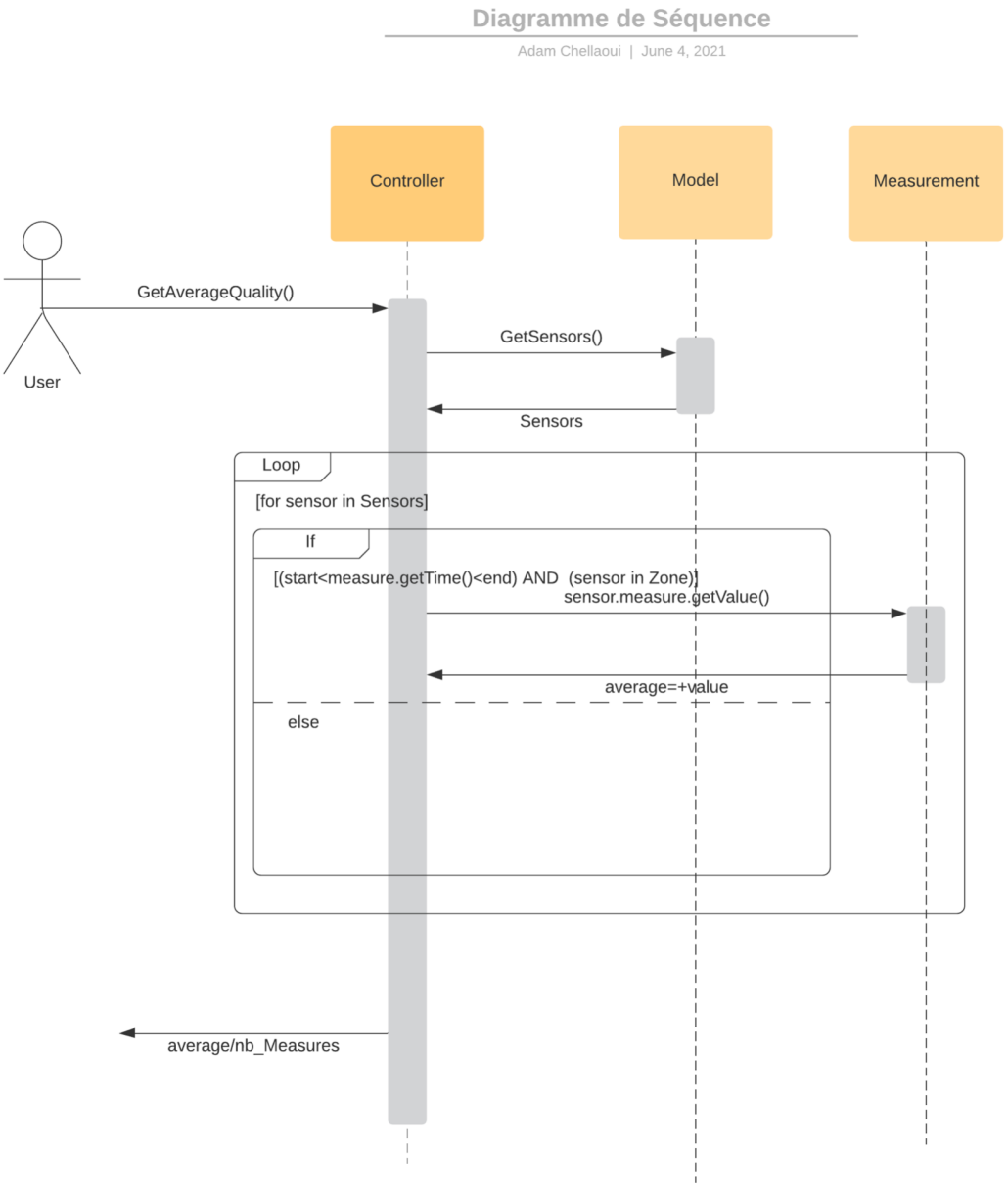
- ➢ Le package Controler : Constitué de la classe Controler
- ➢ Le package Vue : Constitué de la classe View
- ➢ Le package Modèle : Constitué des classes : Model, User, Attribute, Provider, Measurement, Sensor, Cleaner, Date

Sur la page suivante, le diagramme en format paysage pour une meilleure lecture :

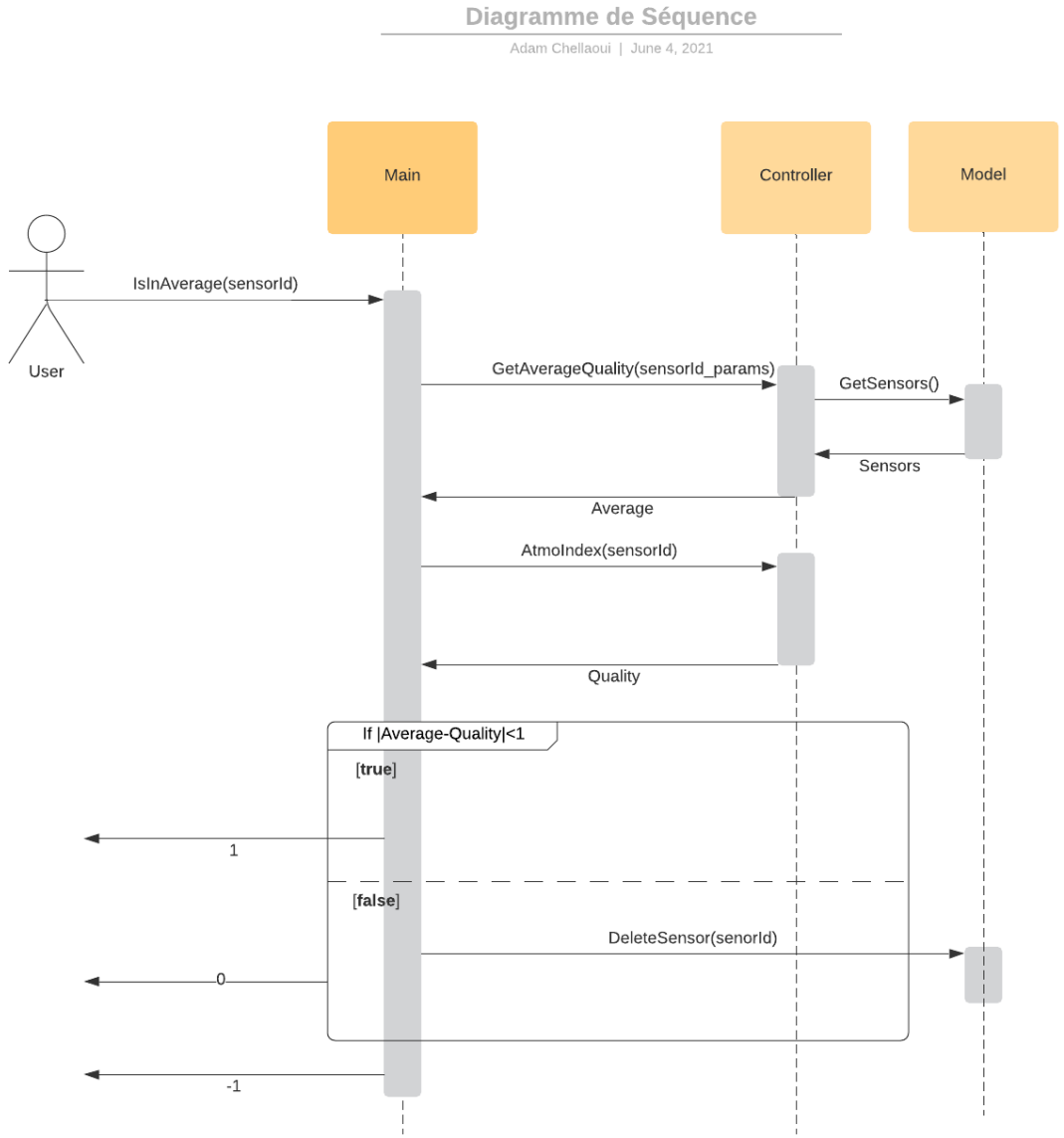

ReadCSV
-sensorCSV: ifstream
-userCSV : ifstream
-providerCSV : ifstream
-cleanerCSV : ifstream
-measurementCSV : ifstream
-attributeCSV : ifstream
+ReadCleaners() : map <string,Cleaner>
+ReadSensors() : map <string,Sensor>
+ReadMeasurements() : list <Measurement>
+ReadUsers() : map <string, User>
+ReadProviders() : map <string, Provider>
+ReadAttributes() : map <string, Attribute>
Provider
-id : string
-cleaners : llist<string>
View
+Menu() : Void
+ShowAirQualityAverage(double index) : Void
+CheckSensorQuality(int quality) : Void
+AskSensorId() : String
+AskStartDate() : Date
+AskEndDate() : Date
+AskLongitude() : double
+AskLatitude() : double
+AskRadius() : double
Cleaner
-id : string
-startTime : Date
-endTime : Date
-latitude : double
-longitude : double
-provider : string
+SetProvider(Provider my_provider) : Void
Date
-year : int
-month : int
-day : int
-hour : int
-minute : int
-seconde : int
+ operator < =
+ operator >=
Measurement
-time: Date
-sensorId : string
-attribute : string
-value : double
Controller
-model : Model
-view : View
+ViewChoice() : void
+GetAverageQuality () : double
+IsInAverage() : int
+LoadData() : Void
Model
-fileReader : ReadCSV
-cleaners : map <string, Cleaner>
-sensors : map <string, Sensor>
-users : map<string, User>
-attributes : map<string, Attribute>
-providers : map <string, Provider>
+Fill() : Void
+DeleteSensor(string sensorId) : Void
User
-id : string
-nbOfPoints : int
-sensors : list<string>
+AddPoints(int points) : Void
Attribute
-id : string
-unit : string
-description : string
Sensor
-id : string
-latitude : double
-longitude : double
-measurements : list <Measurement>
+AddMeasurement(Measurement measure) : Void
1
0..*
2

V. Diagrammes de séquence

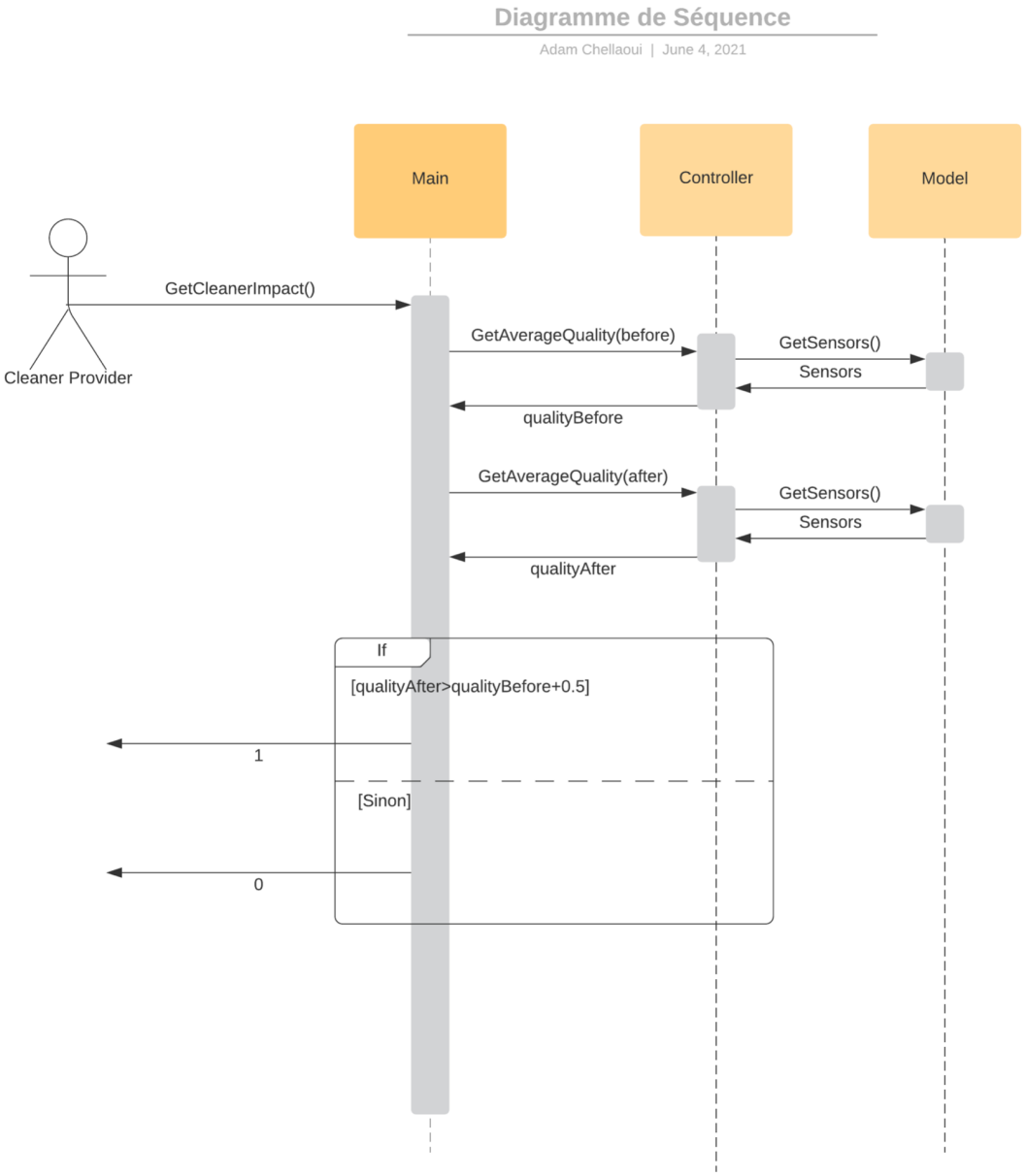
3.1 Fonctionnalité “Calculer la moyenne de l’indicateur de qualité dans un rayon donné “



3.2 Fonctionnalité “Analyse des données collectées par un capteur pour assurer sa fonctionnalité”



3.3 Fonctionnalité “Observer l’impact des cleaners sur la qualité de l’air, selon plusieurs critères”



VI.Pseudo-Code des fonctionnalités

4.1 Fonctionnalité “Calculer la moyenne de l’indicateur de qualité dans un rayon donné “

getQualityRate(latitude,longitude,rayon,dateDebut,dateFin)

Entrée : latitude,longitude,rayon,dateDebut,dateFin

Précondition : Latitude et longitude correspondent à des coordonnées réelles, le rayon est en degré

Sortie : Une note (un réel)

Début :

Somme = 0, nbMesures = 0

Pour chaque Sensor de la liste Sensors :

Si Sensor.latitude ∈ [latitude-rayon;lattitude+rayon]

et Sensor.longitude ∈ [longitude-rayon;longitude+rayon]

Alors Pour chaque Measurement de la liste Sensors.GetMeasurements

Si Measurement.date ∈ [dateDebut;dateFin]

Alors

Somme+=indiceAtmo(Measurement.O3,Measurement.SO2,Measurement.NO2, Measurement.PM10) et nbMesures ++

NoteMoyenne = Somme/nbMesures

Renvoyer : NoteMoyenne

Fin

4.2. Fonctionnalité “Analyse des données collectées par un capteur pour assurer sa fonctionnalité”

assurerFonctionnalite(Capteur)

Entrée : Capteur

Précondition : Le Capteur est affilié à un user Particulier.

Sortie : booléen résultat

Début : Calculer la moyenne de la qualité de l'air des capteurs environnants

Si la qualité donnée par le capteur se situe à moins de 1 Unité près de la moyenne

Renvoyer true

Si la qualité donnée par le capteur est éloignée de la moyenne

Supprimer le Capteur de la liste

Retirer l'User de l'application

Renvoyer false

Fin

4.3. Fonctionnalité "Observer l'impact des cleaners sur la qualité de l'air, selon plusieurs critères"

impactCleaners(provider_id)

Entrée : provider_id

Précondition : provider_id est l'id d'un provider

Sortie : tableau de notes

Début :

Soit tableau un tableau de note ayant la taille du nombre de cleaners du provider.

Pour chaque cleaner de la liste des cleaners tel que le cleaner appartient au provider_id :

On calcule inactif = getQualityRate(localisationCleaner,50km,dateInaction) avec LocalisationCleaner les longitudes et latitudes du cleaner et dateInaction une date où le cleaner est inactif.

On calcule actif = getQualityRate(localisationCleaner,50km,dateAction) avec dateAction une date où le cleaner est actif.

On en déduit une noteCleaner = actif/inactif*10 qui est une note sur 10 correspondant au cleaner et on l'ajoute au tableau des notes

On retourne tableau

Fin

V. Plan de Tests

1. Tests Fonctionnels

1.1. Chargement fichier Measurements normal

1.1.1. Description

Lancement du chargement d'un fichier CSV Measurements

1.1.2. Retour attendu

Affichage de "Fichier de mesures : <chemin> bien chargé !"

1.2. Mauvais chargement d'un fichier

1.2.1. Description

Lancement du chargement d'un fichier Measurements inaccessible

1.2.2. Retour attendu

Affichage de "Erreur lors du chargement du fichier "

1.3. Chargement fichier corrompu

1.3.1. Description

Lancement du chargement d'un fichier avec la mauvaise extension

1.3.2. Retour attendu

Affichage de "Veuillez sélectionner un fichier correct (.csv)"

1.4. Recherche de la conformité des résultats d'un capteur au mauvais identifiant

1.4.1. Description

Lancement d'une recherche de vérification de la cohérence des informations produites par un capteur dont l'identifiant n'existe pas en mémoire.

1.4.2. Retour attendu

Affichage du message : "Cet identifiant ne correspond à aucun capteur"

2. Tests d'Intégration

2.1. Calcul de la moyenne qualité de l'air dans un rayon donné

2.1.1. Description

Lancement d'une recherche de la qualité de l'air produite dans une zone donnée

2.1.2. Retour attendu

Affichage du résultat attendu sous forme de réel entre 1 et 10

2.2. Recherche de la conformité des résultats d'un capteur

2.2.1. Description

Lancement d'une recherche de vérification de la cohérence des informations produites par un capteur.

2.2.2. Retour attendu

Affichage du résultat attendu sous forme de booléen

2.3. Calcul de la moyenne qualité de l'air dans un rayon donné avec un rayon négatif

2.3.1. Description

Lancement d'une recherche de la qualité de l'air produite dans une zone donnée

2.3.2. Retour attendu

Prise en compte de la valeur absolue, donc affichage du résultat attendu sous forme de réel entre 1 et 10

2.4. Calcul de la moyenne qualité de l'air dans une zone sans capteur

2.4.1. Description

Lancement d'une recherche de la qualité de l'air produite dans une zone donnée, mais dans laquelle aucun capteur ne figure

2.4.2. Retour attendu

Message d'erreur

2.5. Recherche de la conformité d'un capteur dans une période sélectionnée sans mesures

2.5.1. Description

Lancement d'une recherche de vérification de l'intégrité des mesures d'un capteur

2.5.2. Retour attendu

Message d'erreur

2.6. Recherche de la conformité d'un capteur dans une période sélectionnée sans mesures

2.6.1. Description

Lancement d'une recherche de vérification de l'intégrité des mesures d'un capteur

2.6.2. Retour attendu

Message d'erreur

2.7. Recherche de la conformité d'un capteur pour un capteur inexistant

2.7.1. Description

Lancement d'une recherche de vérification de l'intégrité des mesures d'un capteur qui n'est pas dans le réseau de capteurs.

2.7.2. Retour attendu

Message d'erreur

3. Tests des exigences non fonctionnelles

3.1. Performance en temps

- Chronométrer le temps mis pour la moyenne de qualité de l'air dans un rayon de 1km
- Chronométrer le temps mis pour la moyenne de qualité de l'air dans un rayon de 5km
- Chronométrer le temps mis pour la moyenne de qualité de l'air dans un rayon de 50km
- Chronométrer le temps mis pour vérifier l'intégrité d'un capteur

3.2. Traçabilité : Recherches en parallèle

Lancer plusieurs recherches de cohérence des valeurs sur plusieurs capteurs à la fois pour tester la solidité du programme