SAMAIN Luc
NGO Son
MOUREAU Alexandre
LEVRARD Thomas

Design Document

I. Choix de l'architecture

Nous avons choisi l'architecture MVC (Model-View-Controller) pour implémenter l'application AirWatcher.

En effet, l'application propose différentes fonctionnalités qui impliquent différentes manières d'interagir avec les différentes données (notamment dû au fait qu'il y a différents types d'utilisateurs).

Ainsi, notre application sera divisée en 3 composants (ou packages) distincts qui interagissent ensemble : "Controller" "Model" et "View".

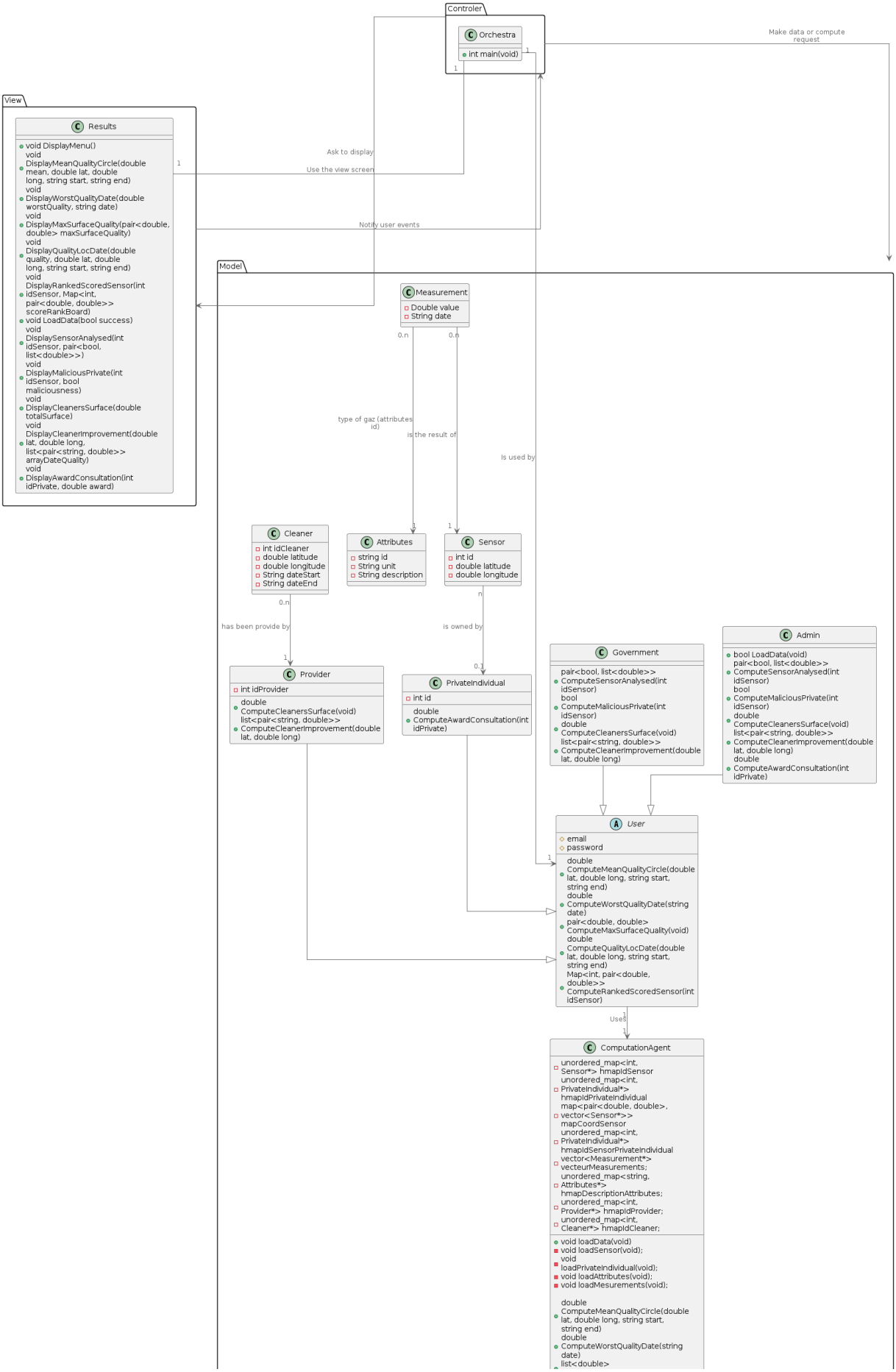
- Le composant **Controller** gère les interactions de l'utilisateur (ex. : saisies au clavier) avec notre application et transmet ces interactions aux autres composants.

- Le composant **View** définit et gère la manière dont les données sont présentées aux utilisateurs.

- Le composant **Model** contient toute la logique métier de l'application. En effet, ce composant fournit les différents services correspondant aux différentes fonctionnalités de l'application. Ce sont ces services qui gèrent et agissent sur les données.

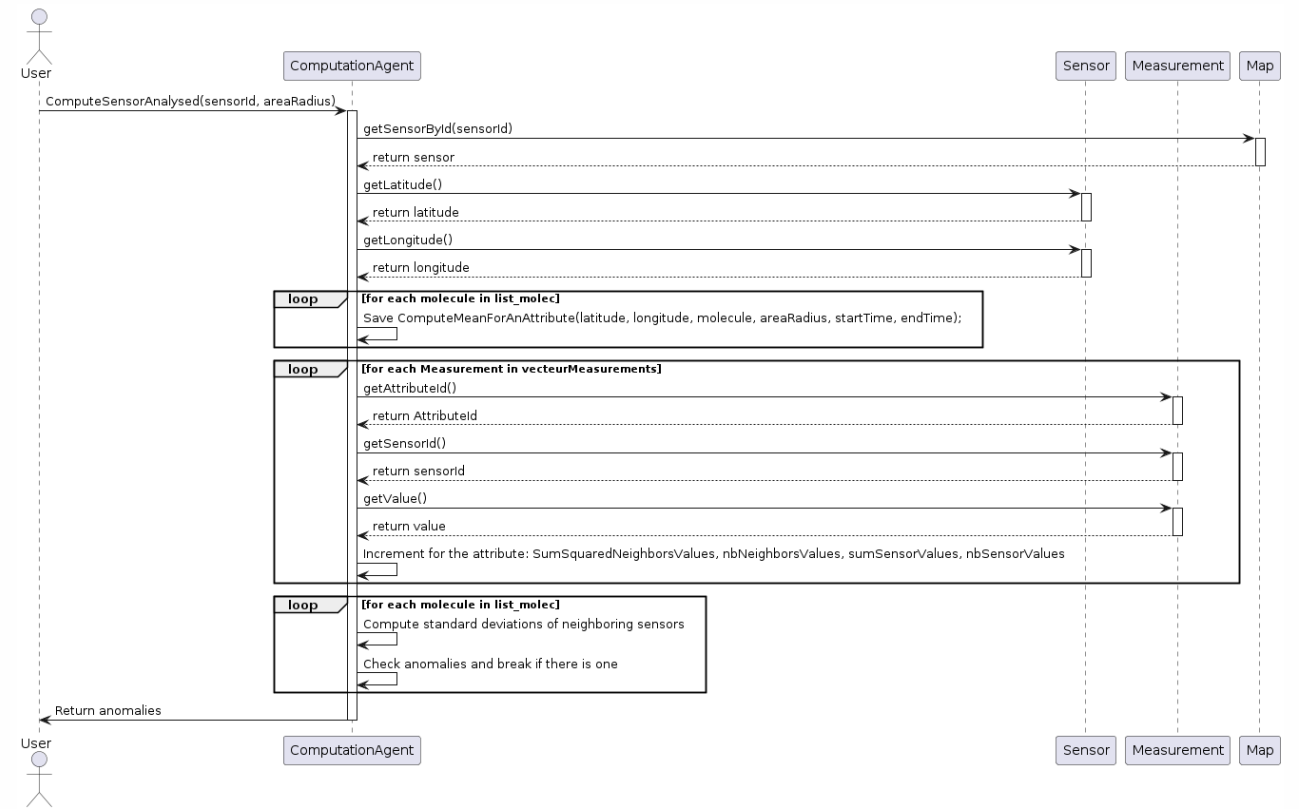
Cette architecture permet donc de séparer la présentation des données et les interactions avec celles-ci.

II. Diagramme de Classe

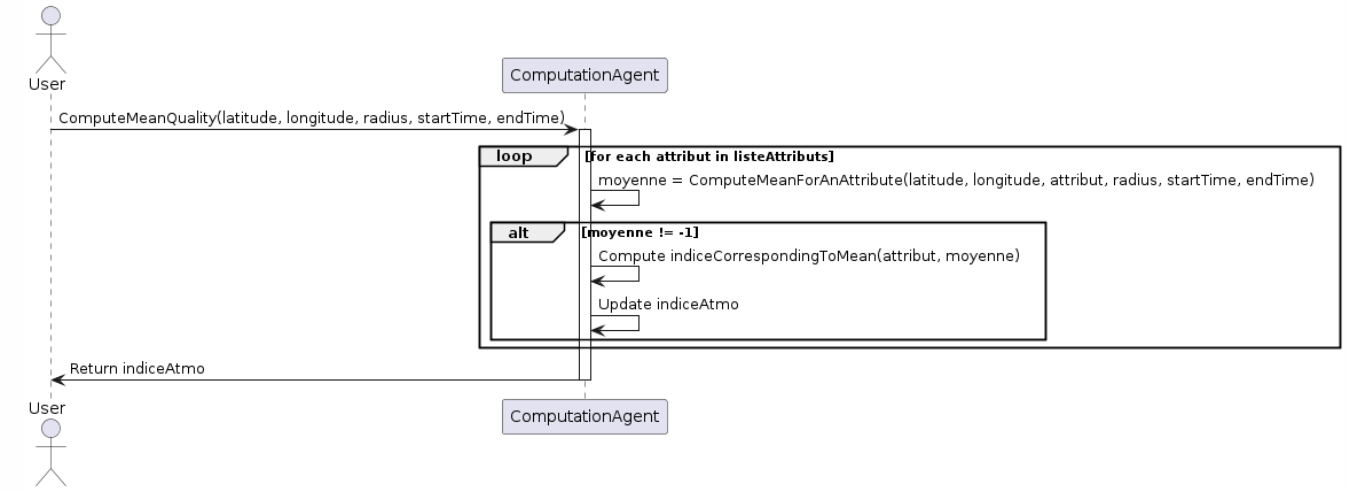


III. Diagrammes de séquence

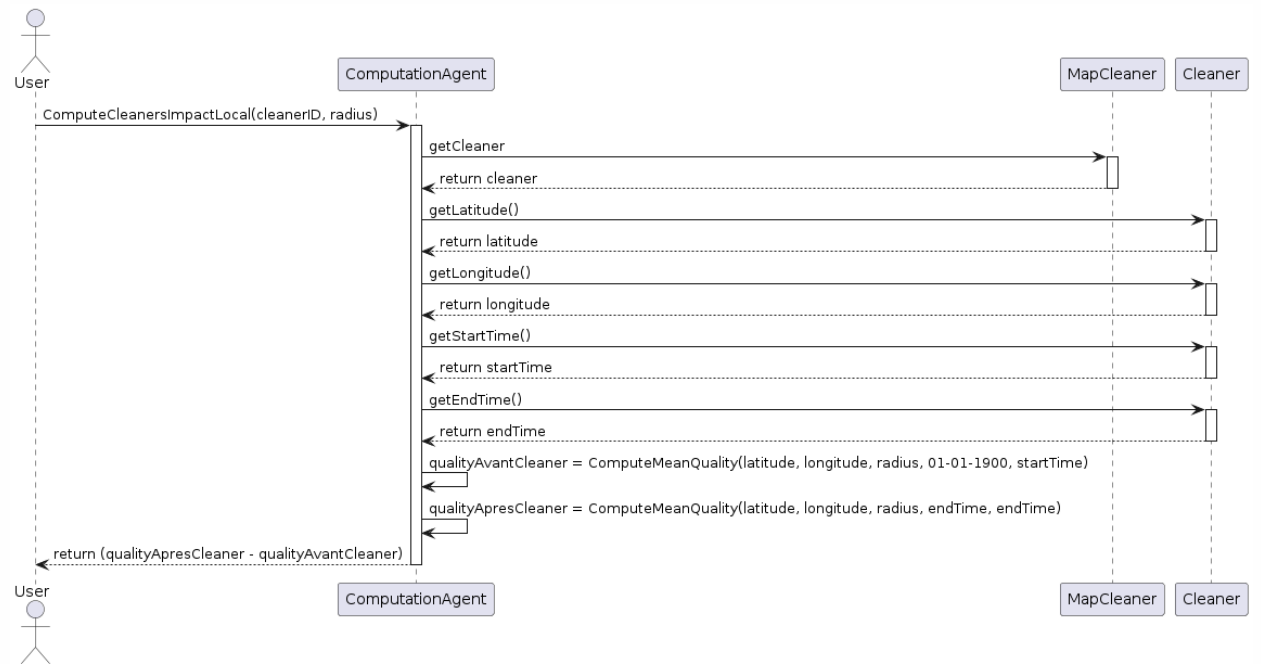
III.1. Scénario 1 : Analyser un capteur



III.2. Scénario 2 : Calculer la moyenne de la qualité de l'air dans une zone géographique donnée et pour une période spécifiée



III.3. Scénario 3 : Analyser l’impact d’un Cleaner



IV. Description et pseudo-code de 3 algorithmes majeurs

IV.1. Algorithme 1 : Analyse des mesures d'un capteur

IV.1.a. Description

Description	Cet algorithme analyse les mesures du capteur, associé à l'id fourni en paramètre, pour déterminer si celui-ci est défaillant ou truqué. Pour ce faire, l'algorithme compare les mesures du capteur avec celles des capteurs qui se trouvent dans un rayon de 'areaRadius' km (paramètre) autour du capteur à analyser. Remarquons qu'il y a différents cas spéciaux. Si notre capteur n'a aucune mesure pour toutes les molécules alors l'application doit spécifier que l'analyse est impossible. Idem s'il n'y a pas de capteurs voisins pour effectuer la comparaison sur au moins une des 4 molécules où le capteur a des mesures. Le diagnostic de l'algorithme est alors renvoyé en sortie, sous la forme d'un booléen : 'true' si le capteur est défaillant (ou truqué) et 'false' sinon.
Entrée	***sensorID*** : Id du capteur à analyser ***areaRadius*** : Rayon en kilomètres représentant la distance maximale à laquelle un capteur voisin peut se trouver pour être pris en compte dans l'analyse
Sortie	***anomalies*** : Booléen qui vaut 'true' si le capteur est défaillant/truqué et 'false' sinon

IV.1.b. Pseudo-code

```
ComputeSensorAnalysed(sensorID, areaRadius):

    anomalies = false

    sensor = rechercherSensorParID(sensorID)

    dicoMeanCapteur = {"O3": 0, "NO2": 0, "SO2": 0, "PM10": 0}

    Créer les dictionnaires dicoMeanAll, dicoSumOfSquaresAll, dicoSdAll, dicoNbValuesAll et dicoNbValuesCapteur

    list_molec = ["O3", "NO2", "SO2", "PM10"]

    // Initialiser les dictionnaires
    Pour chaque molecule dans list_molec faire:
        dicoSumOfSquaresAll[molecule] = dicoNbValuesAll[molecule] = dicoNbValuesCapteurl[molecule] = dicoSdAll[molecule] = dicoMeanAll[molecule] = 0
    Fin Pour

    // Calculer les moyennes globales pour chaque molécule
    Pour chaque molecule dans list_molec faire:
        dicoMeanAll[molecule] = ComputeMeanForAnAttribute(sensor.latitude, sensor.longitude, molecule, areaRadius, 0, 0);
    Fin Pour

    // Lire les mesures et calculer les sommes et les carrés
    Pour chaque mesure dans measurements.csv faire:
        molecule = mesure.attribute

        Si mesure.sensorID == sensorID alors :
            dicoMeanCapteur[molecule] += mesure.value
            dicoNbValuesCapteur[molecule] += 1
        Fin Si

        dicoSumOfSquaresAll[molecule] += (mesure.value - dicoMeanAll[molecule]) *  (mesure.value - dicoMeanAll[molecule])
        dicoNbValueAll[mesure[Attribute]] += 1

    Fin Pour

    Si le capteur n'a pas de mesure pour toutes les molécules alors :
        Exception("Analyse impossible car le capteur n'a aucune mesure")
    Fin Si

    // Calculer les écarts-types et vérifier les anomalies
    Pour chaque molecule dans list_molec faire:
            Si (dicoNbValuesCapteur[molecule] == 0) faire : passer à la molécule suivante
            Fin Si

            Si (dicoNbValueAll[molecule]==dicoNbValuesCapteur[molecule]) faire :
                Exception("Analyse impossible car il n'y a pas de mesure de capteurs voisins pour au moins une molécule ")
            Fin Si

            dicoMeanCapteur[molecule] = dicoMeanCapteur[molecule] / dicoNbValuesCapteur[ molecule ];
            dicoSdAll[molecule] = sqrt(dicoSumOfSquaresAll[molecule] / dicoNbValueAll[molecule])

            Si ((dicoMeanCapteur[molecule] > dicoMeanAll[molecule] + dicoSdAll[molecule]) ou
                dicoMeanCapteur[molecule] < dicoMeanAll[molecule] - dicoSdAll[molecule])) alors :
                anomalies = True
            Fin Si

    Fin Pour

    retourner anomalies
```

IV.2. Algorithme 2 : Calcul de la moyenne de la qualité de l'air dans une zone géographique donnée et pour une période spécifiée

IV.2.a. Description

Description	Cet algorithme permet de calculer la moyenne de la qualité de l'air (indice ATMO de 2014) dans une zone géographique donnée et pour une période de temps spécifiée. La zone géographique est circulaire et elle est définie par son centre (latitude,longitude) et son rayon (en km). La période de temps est définie par deux TimeStamp : dateDebut et dateFin
Entrée	***latitude*** **:** La latitude du centre de la zone géographique ***longitude*** **:** La longitude du centre de la zone géographique ***radius*** **:** Le rayon en kilomètres représentant la distance maximale (à considérer par rapport au centre de la zone) pour inclure les données. ***startTime*** **:** La date de début de la période de temps pour laquelle la moyenne est calculée. ***endTime*** **:** Le date de fin de la période de temps pour laquelle la moyenne est calculée.
Sortie	***indiceAtmo*** **:** entier correspondant à l'indice Atmo calculé (égal à -1 s'il y a eu une erreur)

IV.2.b. Pseudo-code

```
ComputeMeanQuality(latitude, longitude, radius, startTime, endTime):
  list_molec = ["O3", "NO2", "SO2", "PM10"]
  listeMoyennes = []
  indiceATMO = -1
  Pour chaque molecule dans list_molec faire :
    moyenne = ComputeMeanForAnAttribute(latitude, longitude, molecule, radius, startTime, endTime)

    Si (moyenne != 0) alors :
      ajouter(moyenne, listeMoyennes)
      indice = indiceCorrespondingToMean(molecule, moyenne)
      Si (indice > indiceATMO) alors : indiceATMO = indice
    Fin Si
    Sinon
      Traiter Erreur...
      Afficher("Calcul Impossible de l'indiceAtmo")
    Fin Sinon

  Fin Pour

  retourner indiceATMO
```

Fonction ComputeMeanForAnAttribute :

Description	Cette fonction calcule la moyenne des valeurs des mesures pour une molécule donnée (O3, NO2, SO2, PM10), pour une zone géographique donnée et pour une période de temps spécifiée.
Entrée	Idem que **ComputeMeanQuality** mais avec “molecule” en plus ***molecule*** **:** molécule pour laquelle la moyenne va être calculée
Sortie	***moyenne*** **:** La moyenne des valeurs des mesures de la molécule (-1 s’il y a eu un problème ou une erreur)

Remarque : si en entrée on met startTime = endTime = null alors on spécifie qu’on ne veut pas de contrainte sur la date

```
ComputeMeanForAnAttribute(latitude, longitude, molecule, radius, startTime, endTime):
  moyenne = -1
  somme = 0
  nbValues = 0
  Pour chaque mesure dans measurements.csv faire :
    dateMesure = mesure.date
    sensor = mesure.sensor
    Si (mesure.attribute == molecule) alors :
      distance = calculateDistance(latitude, longitude, sensor.latitude, sensor.longitude)

      Si  ((startTime < dateMesure  et dateMesure < endTime) ou (startTime == null et endTime == null)) et
        (distance <= radius) alors :
          somme += mesure.value
          nbValues++
      Fin Si
    Fin Si
  Fin Pour

  moyenne = somme/nbValues
  retourner moyenne
```

Fonction calculateDistance :

Description	Cette fonction utilise la formule de Haversine pour calculer la distance en kilomètres entre deux points géographiques spécifiés par leur latitude et leur longitude.
Entrée	***lat1*** **:** La latitude du premier point ***lon1*** **:** La longitude du premier point ***lat2*** **:** La latitude du deuxième point ***lon2*** **:** La longitude du deuxième point
Sortie	***distance :*** La distance en kilomètres entre les deux points

```
calculateDistance(lat1, lon1, lat2, lon2):
  R =  6371.0  //Rayon de la Terre en km

  lat1 =  radians(lat1)
  lon1 =  radians(lon1)
  lat2 =  radians(lat2)
  lon2 =  radians(lon2)

  dlon = lon2 - lon1
  dlat =  lat2 - lat1
  a =  sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
  c =  2 * atan2(sqrt(a),sqrt(1 - a))
  distance =  R * c

  retourner distance
```

IV.3. Algorithme 3 : Calcul de l'impact d'un cleaner dans une zone

Description	Cette fonction calcule l'impact d'un air cleaner dans une zone circulaire donnée (centre = cleaner et radius en paramètre) en calculant la différence entre la qualité de l'air à la date à laquelle a été arrêté le cleaner (ou la date actuelle s'il n'est pas encore arrêté) et la date à laquelle il a été activé.
Entrée	***cleanerID*** : Id du purificateur à analyser ***radius*** : Rayon en kilomètres représentant la distance maximale à laquelle un capteur peut se trouver pour que ses mesures soient prises en compte dans le calcul de la qualité de l'air
Sortie	Différence entre la qualité de l'air à la date à laquelle a été arrêté le cleaner (ou la date actuelle s'il n'est pas encore arrêté) et la date à laquelle il a été activé

```
ComputeCleanersImpactLocal(cleanerID, radius) :

  cleaner = rechercherCleanerById(cleanerID)

  qualityAvantCleaner = ComputeMeanQuality(cleaner.latitude, cleaner.longitude, radius, 01-01-1900, cleaner.startTime)

  dateCalculApres = La date des dernières mesures effectuées durant la phase où le purificateur a été activé
  qualityApresCleaner = ComputeMeanQuality(cleaner.latitude, cleaner.longitude, radius, dateCalculApres , dateCalculApres )

  Si (qualityAvantCleaner == -1 ou qualityApresCleaner == -1) alors :
    //Traiter les cas d'erreurs
    Afficher ("Impossible de mesurer l'impact du purificateur !")
  Fin Si

  retourner (qualityApresCleaner - qualityAvantCleaner)
```

V. Tests Unitaires

V.1. Test de ComputeMeanQuality

Cas 1 :

Description Cas	Toutes les données sont prises en compte et il y a 2 données par Attribute (molécule)
Données	**Forme = (Sensor,Date,LatCapteur,LongCapteur,Attribute,Unit,Value)** **1 : (Sensor1,2019/01/01,44,-1,O3,µg/m3,10)** **2 : (Sensor1,2019/01/02,44,-1,SO2,µg/m3,10)** **3 : (Sensor1,2019/01/03,44,-1,NO2,µg/m3,10)** **4 : (Sensor1,2019/01/04,44,-1,PM10,µg/m3,10)** **5 : (Sensor2,2019/01/01,44,-1,O3,µg/m3,20)** **6 : (Sensor2,2019/01/02,44,-1,SO2,µg/m3,30)** **7 : (Sensor2,2019/01/03,44,-1,NO2,µg/m3,40)** **8 : (Sensor2,2019/01/04,44,-1,PM10,µg/m3,50)**
Entrée	LatitudeCentre = 44 LongitudeCentre = -1 Rayon = 1 DateDebut = 2019/01/01 DateFin= 2019/02/01
Sortie Attendue	O3 -> m = 15 -> i = 1 SO2-> m = 20 -> i =1 NO2-> m = 25 -> i = 1 PM10-> m = 30 -> i = 5 **ATMO = Max(i) = 5**

Cas 2 :

Description Cas	**Test Filtrage Date :** Des données ne sont pas prises en compte à cause de la date et il y a 2 données par Attribute (molécule)
Données	**Forme = (Sensor,Date,LatCap,LongCap,Attribute,Unit,Value)** **1 : (Sensor1,2019/01/01,44,-1,O3,µg/m3,10)** **2 : (Sensor1,2019/01/02,44,-1,SO2,µg/m3,10)** **3 : (Sensor1,2019/01/03,44,-1,NO2,µg/m3,10)** **4 : (Sensor1,2019/01/04,44,-1,PM10,µg/m3,10)** **5 : (Sensor2,2019/01/01,44,-1,O3,µg/m3,20)** **6 : (Sensor2,2019/01/02,44,-1,SO2,µg/m3,30)** **7 : (Sensor2,2019/01/03,44,-1,NO2,µg/m3,40)** **8 : (Sensor2,2019/01/04,44,-1,PM10,µg/m3,50)** **9 : (Sensor1,2019/03/03,44,-1,O3,µg/m3,1000)** **10 : (Sensor1,2019/03/04,44,-1,SO2,µg/m3,1000)** **11 : (Sensor1,2019/03/03,44,-1,NO2,µg/m3,1000)** **12 : (Sensor1,2019/03/04,44,-1,PM10,µg/m3,1000)**
Entrée	LatitudeCentre = 44 LongitudeCentre = -1 Rayon = 1 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	O3 -> m = 15 -> i = 1 SO2-> m = 20 -> i = 1 NO2-> m = 25 -> i = 1 PM10-> m = 30 -> i = 5 **ATMO = Max(i) = 5**

Cas 3 :

Description Cas	**Test Filtrage Position :** Des données ne sont pas prises en compte à cause de leur position et il y a 2 données par Attribute (molécule)
Données	**Forme = (Sensor,Date,LatCap,LongCap,Attribute,Unit,Value)** **1 : (Sensor1,2019/01/01,44,-1,O3,µg/m3,10)** **2 : (Sensor1,2019/01/02,44,-1,SO2,µg/m3,10)** **3 : (Sensor1,2019/01/03,44,-1,NO2,µg/m3,10)** **4 : (Sensor1,2019/01/04,44,-1,PM10,µg/m3,10)** **5 : (Sensor2,2019/01/01,44,-1,O3,µg/m3,20)** **6 : (Sensor2,2019/01/02,44,-1,SO2,µg/m3,30)** **7 : (Sensor2,2019/01/03,44,-1,NO2,µg/m3,40)** **8 : (Sensor2,2019/01/04,44,-1,PM10,µg/m3,50)** **9 : (Sensor3,2019/01/03,30,-158,O3,µg/m3,1000)** **10 : (Sensor3,2019/01/03,30,-158,SO2,µg/m3,1000)** **11 : (Sensor3,2019/01/03,30,-158,NO2,µg/m3,1000)** **12 : (Sensor3,2019/01/03,30,-158,PM10,µg/m3,1000)**
Entrée	LatitudeCentre = 44 LongitudeCentre = -1 Rayon = 1 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	O3 -> m = 15 -> i = 1 SO2-> m = 20 -> i = 1 NO2-> m = 25 -> i = 1 PM10-> m = 30 -> i = 5 **ATMO = Max(i) = 5**

Cas 4 :

Description Cas	**Test mesures manquantes :** Pour au moins une des molécules (PM10) il n'y a pas de mesure donc cas erreur
Données	**Forme = (DateMes,LatCap,LongCap,AttributeID,Unit,Value)** **1 : (2019/01/01,44,-1,O3,µg/m3,10)** **2 : (2019/01/02,44,-1,SO2,µg/m3,10)** **3 : (2019/01/03,44,-1,NO2,µg/m3,10)** **4 : (2019/01/01,44,-1,O3,µg/m3,20)** **5 : (2019/01/02,44,-1,SO2,µg/m3,30)** **6 : (2019/01/03,44,-1,NO2,µg/m3,40)**
Entrée	LatitudeCentre = 44 LongitudeCentre = -1 Rayon = 1 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**Retourne -1** "Il n'y a pas de mesure pour au moins une des molécules donc calcul de ATMO impossible "

Cas 5:

Description Cas	**Entrée invalide** : Rayon < 0
Données	Peu importe
Entrée	LatitudeCentre = 44 LongitudeCentre = -1 Rayon = - 4 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**Retourne -1** “Le rayon doit être positif “

Cas 6:

Description Cas	**Entrée invalide** : Latitude > 90°
Données	Peu importe
Entrée	LatitudeCentre = 91 LongitudeCentre = -1 Rayon = 4 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**Retourne -1** “La latitude doit être comprise entre -90° et 90°”

Cas 7:

Description Cas	**Entrée invalide** : Longitude > 180°
Données	Peu importe
Entrée	LatitudeCentre = 79 LongitudeCentre = 181 Rayon = 4 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**Retourne -1** “La longitude doit être comprise entre -180° et 180°“

Cas 8:

Description Cas	**Entrée invalide :** DateFin < DateDebut
Données	Peu importe
Entrée	LatitudeCentre = 80 LongitudeCentre = 179 Rayon = 4 DateDebut = 2019/01/01 DateFin = 2018/01/01
Sortie Attendue	**Retourne -1** “La date de fin doit être > à la date du début“

Cas 9:

Description Cas	**Données invalides :** Il y a au moins une valeur de mesure invalide car < 0
Données	**Forme = (Sensor,Date,LatCap,LongCap,Attribute,Unit,Value)** **1 : (Sensor1,2019/01/01,44,-1,O3,µg/m3,-10)** **2 : (Sensor1,2019/01/02,44,-1,SO2,µg/m3,10)** **3 : (Sensor1,2019/01/03,44,-1,NO2,µg/m3,10)** **4 : (Sensor1,2019/01/04,44,-1,PM10,µg/m3,10)** **5 : (Sensor2,2019/01/01,44,-1,O3,µg/m3,20)** **6 : (Sensor2,2019/01/02,44,-1,SO2,µg/m3,30)** **7 : (Sensor2,2019/01/03,44,-1,NO2,µg/m3,40)** **8 : (Sensor2,2019/01/04,44,-1,PM10,µg/m3,50)**
Entrée	LatitudeCentre = 44 LongitudeCentre = -1 Rayon = 1 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**Retourne -1** “Il y a des mesures invalides (<0) donc le calcul est impossible”

V.2. Test de ComputeSensorAnalysed

Cas 1 :

Description Cas	**Capteur défectueux/truqué :** Toutes les données de capteurs sont des données de capteurs voisins et des données sur une seule et même molécule et le capteur est truqué/défectueux
Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)** **(Sensor1,44,-1,2019/01/01,O3,µg/m3,10)** **(Sensor2,44,-1,2019/01/01,O3,µg/m3, 50)** **(Sensor3,44,-1, 2019/01/01,O3,µg/m3, 50)** **(Sensor4,44,-1, 2019/01/01,O3,µg/m3, 50)**
Entrée	sensorID = “Sensor1” areaRadius = 1
Sortie Attendue	**True**

Cas 2 :

Description Cas	**Capteur pas truqué/défectueux :** Toutes les données de capteurs sont des données de capteurs voisins et des données sur la même molécule et le capteur n’est pas truqué/défectueux
Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)** **(Sensor1,44,-1,2019/01/01,O3,µg/m3,10)** **(Sensor2,44,-1,2019/01/01,O3,µg/m3, 9)** **(Sensor3,44,-1, 2019/01/01,O3,µg/m3, 10)** **(Sensor4,44,-1, 2019/01/01,O3,µg/m3, 9)**
Entrée	sensorID = “Sensor1” areaRadius = 1
Sortie Attendue	**False**

Cas 3 :

Description Cas	**Test avec des mesures de capteurs voisins sur d'autres molécules :** Le capteur n'est pas défectueux (il serait défini défectueux si les données qui ne sont pas sur la même molécule étaient utilisées pour l'analyse) et il y a des données de capteurs qui sont voisins mais ne sont pas sur la même molécule
Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)** **(Sensor1,44,-1,2019/01/01,O3,µg/m3,10)** **(Sensor2,44,-1,2019/01/01,O3,µg/m3, 9)** **(Sensor3,44,-1, 2019/01/01,O3,µg/m3, 10)** **(Sensor4,44,-1, 2019/01/01,O3,µg/m3, 9)** **(Sensor5,44,-1,2019/01/01,NO2,µg/m3,100)** **(Sensor6,44,-1,2019/01/01,SO2,µg/m3, 150)** **(Sensor7,44,-1, 2019/01/01,PM10,µg/m3, 118)**
Entrée	sensorID = "Sensor1" areaRadius = 1
Sortie Attendue	**False**

Cas 4:

Description Cas	**Même cas que cas 3 mais défectueux** (il serait défini non défectueux si les données qui ne sont pas sur la même molécule étaient utilisées pour l'analyse)
Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)** **(Sensor1,44,-1,2019/01/01,O3,µg/m3,20)** **(Sensor2,44,-1,2019/01/01,O3,µg/m3, 12)** **(Sensor3,44,-1, 2019/01/01,O3,µg/m3, 13)** **(Sensor4,44,-1, 2019/01/01,O3,µg/m3, 12)** **(Sensor5,44,-1,2019/01/01,NO2,µg/m3,20)** **(Sensor5,44,-1,2019/01/01,NO2,µg/m3,20)** **(Sensor5,44,-1,2019/01/01,NO2,µg/m3,20)** **(Sensor6,44,-1,2019/01/01,SO2,µg/m3, 21)** **(Sensor6,44,-1,2019/01/01,SO2,µg/m3, 21)** **(Sensor6,44,-1,2019/01/01,SO2,µg/m3, 21)** **(Sensor7,44,-1, 2019/01/01,PM10,µg/m3, 20)** **(Sensor7,44,-1, 2019/01/01,PM10,µg/m3, 20)** **(Sensor7,44,-1, 2019/01/01,PM10,µg/m3, 20)**
Entrée	sensorID = "Sensor1" areaRadius = 1
Sortie Attendue	**True**

Cas 5 :

Description Cas	**Test pas défectueux avec des mesures de capteurs pas voisins :** Le capteur n'est pas défectueux (il serait défini comme tel si les données qui ne sont pas dans la zone géographique étaient utilisées pour l'analyse) et il y a des capteurs sur la même molécule mais pas voisins
Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)** **(Sensor1,44,-1,2019/01/01,O3,µg/m3,10)** **(Sensor2,44,-1,2019/01/01,O3,µg/m3, 9)** **(Sensor3,44,-1, 2019/01/01,O3,µg/m3, 10)** **(Sensor4,44,-1, 2019/01/01,O3,µg/m3, 9)** **(Sensor5,-80,190,2019/01/01,O3,µg/m3,100)** **(Sensor6,-80,190,2019/01/01,O3,µg/m3, 100)** **(Sensor7,-80,190, 2019/01/01,O3,µg/m3, 100)** **(Sensor8,-80,190, 2019/01/01,O3,µg/m3, 100)**
Entrée	sensorID = "Sensor1" areaRadius = 1
Sortie Attendue	**False**

Cas 6 :

Description Cas	**Même cas que cas 5 mais défectueux :** (il serait défini comme non défectueux si les données qui ne sont pas sur la même molécule étaient utilisées pour l'analyse)
Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)** **(Sensor1,44,-1,2019/01/01,O3,µg/m3,30)** **(Sensor2,44,-1,2019/01/01,O3,µg/m3, 9)** **(Sensor3,44,-1, 2019/01/01,O3,µg/m3, 10)** **(Sensor4,44,-1, 2019/01/01,O3,µg/m3, 9)** **(Sensor5,-80,190,2019/01/01,O3,µg/m3,30)** **(Sensor6,-80,190,2019/01/01,O3,µg/m3, 30)** **(Sensor7,-80,190, 2019/01/01,O3,µg/m3, 30)** **(Sensor8,-80,190, 2019/01/01,O3,µg/m3, 30)** **(Sensor9,-80,190, 2019/01/01,O3,µg/m3, 30)** **(Sensor10,-80,190, 2019/01/01,O3,µg/m3, 30)**
Entrée	sensorID = "Sensor1" areaRadius = 1
Sortie Attendue	**True**

Cas 7 :

Description Cas	**Cas erreur :** Des capteurs ont des mesures sur la même molécule mais ne sont pas voisins
Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)** **(Sensor1,44,-1, 2019/01/01,O3,µg/m3, 20)** **(Sensor2,-80,190,2019/01/01,O3,µg/m3,30)** **(Sensor3,-80,190,2019/01/01,O3,µg/m3, 30)** **(Sensor4,-80,190, 2019/01/01,O3,µg/m3, 30)**
Entrée	sensorID = “Sensor1” areaRadius = 1
Sortie Attendue	**Exception :** “Analyse impossible car il n’y a pas de mesure de capteurs voisins pour au moins une molécule”

Cas 8 :

Description Cas	**Cas erreur :** Aucun capteur n’a de mesure sur la même molécule
Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)** **(Sensor1,44,-1, 2019/01/01,O3,µg/m3, 20)** **(Sensor2,44,-1,2019/01/01,SO2,µg/m3,30)** **(Sensor3,44,-1,2019/01/01,NO2,µg/m3, 30)** **(Sensor4,44,-1,2019/01/01,PM10,µg/m3, 30)**
Entrée	sensorID = “Sensor1” areaRadius = 1
Sortie Attendue	**Exception :** “Analyse impossible car il n’y a pas de mesure de capteurs voisins pour au moins une molécule”

Cas 9 :

Description Cas	**Entrée invalide :** Rayon < 0
Données	Peu importe
Entrée	sensorID = “Sensor1” areaRadius = -1
Sortie Attendue	**Exception** **“Rayon invalide”**

Cas 10 :

Description Cas	**Entrée invalide :** Le capteur n’existe pas
Données	Sensors.csv : (vide)
Entrée	sensorID = “Sensor1” areaRadius = 1
Sortie Attendue	**Exception** **“Capteur invalide”**

Cas 11 :

Description Cas	**Non Défectueux - Filtrage des capteurs avec areaRadius limite :** Teste si le filtrage selon la zone définie par l'area radius fonctionne bien (capteur non défectueux mais qui serait défini comme défectueux si les données hors de l'area radius étaient prises en compte)

Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)**
	
	(Sensor1,44,-1,2019/01/01,O3,µg/m3,10)
	(Sensor2,44,1.05,2019/01/01,O3,µg/m3, 9) //d<areaRadius
	
	(Sensor5,44,1.30,2019/01/01,O3,µg/m3,100) //hors zone
	(Sensor6,44,1.30,2019/01/01,O3,µg/m3,100) //hors zone
	(Sensor7,44,1.30,2019/01/01,O3,µg/m3,100) //hors zone
	(Sensor8,44,1.30,2019/01/01,O3,µg/m3,100) //hors zone

Entrée	sensorID = "Sensor1" areaRadius = 10
Sortie Attendue	**False**

Cas 12 :

Description Cas	**Défectueux - Filtrage des capteurs avec areaRadius limite :** Teste si le filtrage selon la zone définie par l'area radius fonctionne bien (capteur défectueux mais qui serait défini comme non défectueux si les données hors de l'area radius étaient prises en compte)

Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)**
	
	(Sensor1,44,-1,2019/01/01,O3,µg/m3,100)
	(Sensor2,44,1.05,2019/01/01,O3,µg/m3, 9) //d<areaRadius
	(Sensor3,44,1.08, 2019/01/01,O3,µg/m3, 10) //d<areaRadius
	(Sensor4,44,1.10, 2019/01/01,O3,µg/m3, 9)// d<areaRadius
	
	(Sensor5,44,1.30,2019/01/01,O3,µg/m3,100) //hors zone
	(Sensor6,44,1.30,2019/01/01,O3,µg/m3,100) //hors zone
	(Sensor7,44,1.30,2019/01/01,O3,µg/m3,100) //hors zone
	(Sensor8,44,1.30,2019/01/01,O3,µg/m3,100) //hors zone
	(Sensor9,44,1.30,2019/01/01,O3,µg/m3,100) //hors zone
	(Sensor10,44,1.30,2019/01/01,O3,µg/m3,100) //hors zone

Entrée	sensorID = "Sensor1" areaRadius = 10
Sortie Attendue	**true**

Cas 13 :

Description Cas	**Cas erreur :** Le capteur n'a aucune mesure sur toutes les molécules
Données	**Forme = (Sensor,LatCap,LongCap)** **(Sensor1,45,1)**
Entrée	sensorID = "Sensor1" areaRadius = 10
Sortie Attendue	**Null** "Analyse impossible car le capteur n'a aucune mesure"

Cas 14 :

Description Cas	**Test avec mesures du capteur sur plusieurs molécules - Défectueux :** Le capteur a des mesures pour O3 et NO2 et SO2, pour O3 il est défectueux mais pas pour NO2 et SO, mais il est noté comme défectueux
Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)** **(Sensor1,44,-1,2019/01/01,O3,µg/m3,10)** **(Sensor2,44,-1,2019/01/01,O3,µg/m3, 50)** **(Sensor3,44,-1, 2019/01/01,O3,µg/m3, 50)** **(Sensor4,44,-1, 2019/01/01,O3,µg/m3, 50)** **(Sensor1,44,-1,2019/01/01,NO2,µg/m3,10)** **(Sensor2,44,-1,2019/01/01,NO2,µg/m3, 10)** **(Sensor3,44,-1, 2019/01/01,NO2,µg/m3, 10)** **(Sensor4,44,-1, 2019/01/01,NO2,µg/m3, 10)** **(Sensor1,44,-1,2019/01/01,SO2,µg/m3,10)** **(Sensor2,44,-1,2019/01/01,SO2,µg/m3, 10)** **(Sensor3,44,-1, 2019/01/01,SO2,µg/m3, 10)** **(Sensor4,44,-1, 2019/01/01,SO2,µg/m3, 10)**
Entrée	sensorID = “Sensor1” areaRadius = 10
Sortie Attendue	**true**

Cas 15 :

Description Cas	**Test avec mesures du capteur sur plusieurs molécules - Non Défectueux :** Le capteur a des mesures pour toutes les molécules et est non défectueux
Données	**Forme = (Sensor,LatCap,LongCap,Date,Attribute,Unit,Value)** **(Sensor1,44,-1,2019/01/01,O3,µg/m3,50)** **(Sensor2,44,-1,2019/01/01,O3,µg/m3, 50)** **(Sensor3,44,-1, 2019/01/01,O3,µg/m3, 50)** **(Sensor4,44,-1, 2019/01/01,O3,µg/m3, 50)** **(Sensor1,44,-1,2019/01/01,NO2,µg/m3,10)** **(Sensor2,44,-1,2019/01/01,NO2,µg/m3, 10)** **(Sensor3,44,-1, 2019/01/01,NO2,µg/m3, 10)** **(Sensor4,44,-1, 2019/01/01,NO2,µg/m3, 10)** **(Sensor1,44,-1,2019/01/01,SO2,µg/m3,30)** **(Sensor2,44,-1,2019/01/01,SO2,µg/m3, 30)** **(Sensor3,44,-1, 2019/01/01,SO2,µg/m3, 30)** **(Sensor4,44,-1, 2019/01/01,SO2,µg/m3, 30)** **(Sensor1,44,-1,2019/01/01,PM10,µg/m3,20)** **(Sensor2,44,-1,2019/01/01,PM10,µg/m3, 20)** **(Sensor3,44,-1, 2019/01/01,PM10,µg/m3, 20)** **(Sensor4,44,-1, 2019/01/01,PM10,µg/m3, 20)**
Entrée	sensorID = “Sensor1” areaRadius = 10
Sortie Attendue	**false**

V.3. Test de ComputeCleanersImpactLocal

Cas 1 :

Description Cas	**Amélioration qualité :** Cas où le cleaner a amélioré la qualité de l'air, toutes les mesures proviennent de capteurs dans la zone
Données	**Measurements.csv :** **// Avant** **(2024-05-01 01:00:00, 45, 1, O3, µg/m3, 100, sensorID = Sensor1)** **(2024-05-01 01:00:00, 45, 1, SO2, µg/m3, 100, sensorID = Sensor1)** **(2024-05-01 01:00:00, 45, 1, NO2, µg/m3, 100, sensorID = Sensor1)** **(2024-05-01 01:00:00, 45, 1, PM10, µg/m3, 100, sensorID = Sensor1)** **//Les dernières données durant la phase de marche du cleaner** **(2024-05-03 01:00:00, 45, 1, O3, µg/m3, 10, sensorID = Sensor1)** **(2024-05-03 01:00:00, 45, 1, SO2, µg/m3, 10, sensorID = Sensor1)** **(2024-05-03 01:00:00, 45, 1, NO2, µg/m3, 10, sensorID = Sensor1)** **(2024-05-03 01:00:00, 45, 1, PM10, µg/m3, 10, sensorID = Sensor1)** **Cleaners.csv :** **(Cleaner0,45,1,2024-05-02 00:00:00, 2024-05-03 23:59:59)** **Sensors.csv :** **(Sensor1,45,1)**
Entrée	**(Cleaner = Cleaner0, radius = 1)**
Sortie Attendue	**-8** Car AtmoAvant(O3) = 4 | AtmoAprès(O3) = 1 AtmoAvant(SO2) = 3| AtmoAprès(SO2) = 1 AtmoAvant(NO2) = 4| AtmoAprès(NO2) = 1 AtmoAvant(PM10) = 10| AtmoAprès(PM10) = 2 AtmoAvant = 10 | AtmoAprès = 2

Cas 2 :

Description Cas	**Baisse qualité :** Cas où le cleaner n'a pas eu d'effet et la qualité de l'air a empiré, toutes les mesures proviennent de capteurs dans la zone
Données	**Measurements.csv :** **// Avant** **(2024-05-01 01:00:00, 45, 1, O3, µg/m3, 10, sensorID = Sensor1)** **(2024-05-01 01:00:00, 45, 1, SO2, µg/m3, 10, sensorID = Sensor1)** **(2024-05-01 01:00:00, 45, 1, NO2, µg/m3, 10, sensorID = Sensor1)** **(2024-05-01 01:00:00, 45, 1, PM10, µg/m3, 10, sensorID = Sensor1)** **//Les dernières données durant la phase de marche du cleaner** **(2024-05-03 01:00:00, 45, 1, O3, µg/m3, 100, sensorID = Sensor1)** **(2024-05-03 01:00:00, 45, 1, SO2, µg/m3, 100, sensorID = Sensor1)** **(2024-05-03 01:00:00, 45, 1, NO2, µg/m3, 100, sensorID = Sensor1)** **(2024-05-03 01:00:00, 45, 1, PM10, µg/m3, 100, sensorID = Sensor1)** **Cleaners.csv :** **(Cleaner0,45,1,2024-05-02 00:00:00, 2024-05-03 23:59:59)** **Sensors.csv :** **(Sensor1,45,1)**
Entrée	**(Cleaner0, 1)**
Sortie Attendue	**8**

Cas 3 :

Description Cas	**Filtrage selon le rayon :** Cas où le cleaner a amélioré la qualité de l'air, toutes les mesures ne proviennent pas de capteurs dans la zone (si les mesures des capteurs pas dans la zone avaient été prises en compte alors resultat différent)
Données	**Measurements.csv :** **// Avant** **(2024-05-01 01:00:00, 45, 1, O3, µg/m3, 100, sensorID = Sensor1)** **(2024-05-01 01:00:00, 45, 1, SO2, µg/m3, 100, sensorID = Sensor1)** **(2024-05-01 01:00:00, 45, 1, NO2, µg/m3, 100, sensorID = Sensor1)** **(2024-05-01 01:00:00, 45, 1, PM10, µg/m3, 100, sensorID = Sensor1)** **//Les dernières données durant la phase de marche du cleaner** **(2024-05-03 01:00:00, 45, 1, O3, µg/m3, 10, sensorID = Sensor1)** **(2024-05-03 01:00:00, 45, 1, SO2, µg/m3, 10, sensorID = Sensor1)** **(2024-05-03 01:00:00, 45, 1, NO2, µg/m3, 10, sensorID = Sensor1)** **(2024-05-03 01:00:00, 45, 1, PM10, µg/m3, 10, sensorID = Sensor1)** **//Mesures (dernières données durant phase marche cleaner) d'un capteur pas dans la zone** **(2024-05-03 23:00:00, -90, 150, O3, µg/m3, 200, sensorID = Sensor2)** **(2024-05-03 23:00:00, -90, 150, SO2, µg/m3, 200, sensorID = Sensor2)** **(2024-05-03 23:00:00, -90, 150,NO2, µg/m3, 200, sensorID = Sensor2)** **(2024-05-03 23:00:00, -90, 150,PM10, µg/m3, 200, sensorID = Sensor2)** **Cleaners.csv :** **(Cleaner0,45,1,2024-05-02 00:00:00, 2024-05-03 23:59:59)** **Sensors.csv :** **(Sensor1, 45, 1)** **(Sensor2, -90, 150,)**
Entrée	**(Cleaner = Cleaner0, radius = 1)**
Sortie Attendue	**-8** Car AtmoAvant(O3) = 4 | AtmoAprès(O3) = 1 AtmoAvant(SO2) = 3| AtmoAprès(SO2) = 1 AtmoAvant(NO2) = 4| AtmoAprès(NO2) = 1 AtmoAvant(PM10) = 10| AtmoAprès(PM10) = 2 AtmoAvant = 10 | AtmoAprès = 2

Cas 4 :

Description Cas	**Filtrage date :** Cas où le cleaner a amélioré la qualité de l'air, toutes les mesures dans measurements.csv n'ont pas été faites entre le début activation cleaner et désactivation cleaners (si ces mesures avaient été prises en compte alors résultat aurait été différent)
Données	**Measurements.csv :** **// Avant** **(2024-05-01 01:00:00;Sensor1;O3;100;)** **(2024-05-01 01:00:00;Sensor1;SO2;100;)** **(2024-05-01 01:00:00;Sensor1;NO2;100;)** **(2024-05-01 01:00:00;Sensor1;PM10;100;)** **//Les dernières données durant la phase de marche du cleaner** **(2024-05-03 01:00:00;Sensor1;O3;10;)** **(2024-05-03 01:00:00;Sensor1;SO2;10;)** **(2024-05-03 01:00:00;Sensor1;NO2;10;)** **(2024-05-03 01:00:00;Sensor1;PM10;10;)** **// Mesures pas faites durant phase marche du cleaner** **(2024-11-03 01:00:00;Sensor1;O3;200;)** **(2024-11-03 01:00:00;Sensor1;SO2;200;)** **(2024-11-03 01:00:00;Sensor1;NO2;200;)** **(2024-11-03 01:00:00;Sensor1;PM10;200;)** **Cleaners.csv :** **(Cleaner0,45,1,2024-05-02 00:00:00, 2024-05-03 23:59:59)** **Sensors.csv :** **(Sensor1, 45, 1)**
Entrée	**(Cleaner = Cleaner0, radius = 1)**
Sortie Attendue	**-8** Car AtmoAvant(O3) = 4 | AtmoAprès(O3) = 1 AtmoAvant(SO2) = 3| AtmoAprès(SO2) = 1 AtmoAvant(NO2) = 4| AtmoAprès(NO2) = 1 AtmoAvant(PM10) = 10| AtmoAprès(PM10) = 2 AtmoAvant = 10 | AtmoAprès = 2

Cas 5 :

Description Cas	Cas où le rayon donné en paramètre n'est pas strictement positif
Données	Cleaners.csv : (Cleaner0,45,1,2024-05-02 00:00:00, 2024-05-03 23:59:59)
Entrée	(Cleaner = Cleaner0, radius = -2)
Sortie Attendue	"Le rayon doit être strictement positif"

Cas 6 :

Description Cas	Cas où le cleaner donné en paramètre n'existe pas
Données	Cleaners.csv : (Cleaner0,45,1,2024-05-02 00:00:00, 2024-05-03 23:59:59)
Entrée	(Cleaner = Cleaner2, radius = 1)
Sortie Attendue	"Le Cleaner que vous avez demandé n'existe pas"

Cas 7 :

Description Cas	Cas où il y a des mesures avant la position du cleaner dans la zone mais pas après
Données	Measurements.csv : // Avant (2024-05-01 01:00:00;Sensor1;O3;100;) (2024-05-01 01:00:00;Sensor1;SO2;100;) (2024-05-01 01:00:00;Sensor1;NO2;100;) (2024-05-01 01:00:00;Sensor1;PM10;100;) Cleaners.csv : (Cleaner0,45,1,2024-05-02 00:00:00, 2024-05-03 23:59:59) Sensors.csv : (Sensor1, 45, 1)
Entrée	(Cleaner = Cleaner0, radius = 1)
Sortie Attendue	“Analyse du cleaner impossible : Aucune mesure dans la zone spécifiée après l’activation du cleaner”

Cas 8 :

Description Cas	Cas où il y a des mesures après l’activation du cleaner mais pas avant activation
Données	Measurements.csv : // Après activation cleaner (2024-05-03 01:00:00;Sensor1;O3;10;) (2024-05-03 01:00:00;Sensor1;SO2;10;) (2024-05-03 01:00:00;Sensor1;NO2;10;) (2024-05-03 01:00:00;Sensor1;PM10;10;) Cleaners.csv : (Cleaner0,45,1,2024-05-02 00:00:00, 2024-05-03 23:59:59) Sensors.csv :

	(Sensor1, 45, 1)
Entrée	**(Cleaner = Cleaner0, radius = 1)**
Sortie Attendue	**“Analyse du cleaner impossible : Aucune mesure dans la zone spécifiée avant activation du cleaner”**

V.4. Test de indiceCorrespondingToMean

Cas 1 :

Description Cas	**Test pour O3**
Entrée	Molecule = “O3” Moyenne = 240.0
Sortie Attendue	**10**

Cas 2 :

Description Cas	**Test pour SO2**
Entrée	Molecule = “SO2” Moyenne = 240.0
Sortie Attendue	**6**

Cas 3 :

Description Cas	**Test pour NO2**
Entrée	Molecule = “NO2” Moyenne = 240.0
Sortie Attendue	**8**

Cas 4 :

Description Cas	**Test pour PM10**
Entrée	Molecule = “PM10” Moyenne = 240.0
Sortie Attendue	**10**

Cas 5 :

Description Cas	**Molécule inconnue**
Entrée	Molecule = “TropDeTests” Moyenne = 240.0
Sortie Attendue	**“La molécule spécifiée n’existe pas”**

Cas 6 :

Description Cas	**Moyenne < 0**
Entrée	Molecule = “O3” Moyenne = -240.0
Sortie Attendue	**“La moyenne des valeurs des mesures doit être >=0”**

V.5. Test de ComputeMeanForAnAttribute

Cas 1 :

Description Cas	**Moyenne pour O3 : toutes les données sont prises en compte**
Données	**Measurements.csv :** **(Timestamp;SensorID;AttributeID;Value;)** **(2019/01/01 01:00:00, Sensor1, O3, µg/m3,10)** **(2019/01/01 01:00:00,Sensor1,O3,µg/m3,20)** **(2019/01/02 01:00:00,Sensor1,O3,µg/m3,30)** **(2019/01/03 01:00:00,Sensor1,O3,µg/m3,40)** **(2019/01/04 01:00:00,Sensor1,O3,µg/m3,50)** **Sensors.csv :** **(Sensor1, 45, 1)**
Entrée	Molecule = “O3” LatitudeCentre = 45 LongitudeCentre = 1 Rayon = 1 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**30.0**

Cas 2 :

Description Cas	**Filtrage Date :** Des données ne sont pas prises en compte à cause de la date
Données	**Measurements.csv :** **(Timestamp;SensorID;AttributeID;Value;)** **(2019/01/01 01:00:00;, Sensor1, O3, µg/m3,10)** **(2019/01/01 01:00:00;,Sensor1,O3,µg/m3,20)** **(2019/01/02 01:00:00;,Sensor1,O3,µg/m3,30)** **(2019/01/03 01:00:00;,Sensor1,O3,µg/m3,40)** **(2019/01/04 01:00:00;,Sensor1,O3,µg/m3,50)** **//Pas prises en compte** **(2018/01/04 01:00:00;,Sensor1,O3,µg/m3,200)** **(2018/01/04 01:00:00;,Sensor1,O3,µg/m3,350)** **(2020/01/04 01:00:00;,Sensor1,O3,µg/m3,200)** **(2020/01/04 01:00:00;,Sensor1,O3,µg/m3,350)** **Sensors.csv :** **(Sensor1, 45, 1)**
Entrée	Molecule = “O3” LatitudeCentre = 45 LongitudeCentre = 1 Rayon = 1 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**30.0**

Cas 3 :

Description Cas	**Filtrage selon position :** Des données ne sont pas prises en compte à cause de leur position
Données	**Measurements.csv :** **(Timestamp;SensorID;AttributeID;Value;)** **(2019/01/01 01:00:00;Sensor1, O3, µg/m3,10)** **(2019/01/01 01:00:00;,Sensor1,O3,µg/m3,20)** **(2019/01/02 01:00:00;,Sensor1,O3,µg/m3,30)** **(2019/01/03 01:00:00;,Sensor1,O3,µg/m3,40)** **(2019/01/04 01:00:00;,Sensor1,O3,µg/m3,50)** **//Pas prises en compte** **(2019/01/03 01:00:00,Sensor2,O3,µg/m3,200)** **(2019/01/03 01:00:00,Sensor2,O3,µg/m3,350)** **(2019/01/03 01:00:00,Sensor2,O3,µg/m3,200)** **(2019/01/03 01:00:00,Sensor2,O3,µg/m3,350)** **Sensors.csv :** **(Sensor1, 45, 1)** **(Sensor2, 70, 178)**
Entrée	Molecule = “O3” LatitudeCentre = 45 LongitudeCentre = 1 Rayon = 1 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**30.0**

Cas 4 :

Description Cas	**Filtrage selon molécule :** Des données ne sont pas prises en compte car elles ne sont pas sur la bonne molécule et il n'y a pas de mesure pour la molécule en question
Données	Measurements.csv : //Pas prises en compte (2019/01/03 01:00:00,Sensor1,SO2,µg/m3,200) (2019/01/03 01:00:00,Sensor1,NO2,µg/m3,350) (2019/01/03 01:00:00,Sensor1,NO2,µg/m3,200) (2019/01/03 01:00:00,Sensor1,PM10,µg/m3,10) Sensors.csv : (Sensor1, 45, 1)
Entrée	Molecule = "O3" LatitudeCentre = 45 LongitudeCentre = 1 Rayon = 1 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**-1** "Il n'y a pas de mesure pour la molécule et les contraintes spécifiées"

Cas 5 :

Description Cas	**Entrée invalide : Rayon < 0**
Données	Peu importe
Entrée	Molecule = “O3” LatitudeCentre = 44 LongitudeCentre = -1 Rayon = -4 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**-1** **“ Le rayon doit être positif “**

Cas 6 :

Description Cas	**Entrée invalide : Latitude > 90°**
Données	Peu importe
Entrée	Molecule = “O3” LatitudeCentre = 91 LongitudeCentre = -1 Rayon = 4 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**-1** “La latitude doit être comprise entre -90° et 90°“

Cas 7 :

Description Cas	**Entrée invalide : Longitude > 180°**
Données	Peu importe
Entrée	Molecule = “O3” LatitudeCentre = 79 LongitudeCentre = 181 Rayon = 4 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**-1** “La longitude doit être comprise entre -180° et 180°

Cas 8 :

Description Cas	**Entrée invalide : DateFin < DateDebut**
Données	Peu importe
Entrée	Molecule = “O3” LatitudeCentre = 80 LongitudeCentre = 179 Rayon = 4 DateDebut = 2019/01/01 DateFin = 2018/01/01
Sortie Attendue	**-1** “ La date de fin doit être > à la date du début“

Cas 9 :

Description Cas	**Il y a au moins une valeur de mesure invalide car < 0 pour la molécule spécifiée**
Données	**Measurements.csv :** **(Timestamp;SensorID;AttributeID;Value;)** **(2019/01/01 01:00:00;Sensor1, O3, µg/m3,-10)** **(2019/01/01 01:00:00;,Sensor1,O3,µg/m3,20)** **Sensors.csv :** **(Sensor1, 45, 1)**
Entrée	Molecule = “O3” LatitudeCentre = 45 LongitudeCentre = 1 Rayon = 1 DateDebut = 2019/01/01 DateFin = 2019/02/01
Sortie Attendue	**-1** “Il y a des mesures invalides (<0) donc le calcul est impossible”

Cas 10 :

Description Cas	**Entrée invalide : La molécule spécifiée n’existe pas**
Données	Peu importe
Entrée	Molecule = “TropDeTests” LatitudeCentre = 80 LongitudeCentre = 179 Rayon = 4 DateDebut = 2019/01/01 DateFin = 2018/01/01
Sortie Attendue	**-1** “La molécule spécifiée n’existe pas“

V.6. Test de CalculateDistance

Cas 1 :

Description Cas	**Entrées valides**
Entrée	(lat1, long1) = (1,1) (lat2, long2) = (2,2)
Sortie Attendue	**157.5**

Cas 2 :

Description Cas	**Entrées valides**
Entrée	(lat1, long1) = (67,174) (lat2, long2) = (26,32)
Sortie Attendue	**9198.0**

Cas 3 :

Description Cas	**Entrées valides : 2 points égaux**
Entrée	(lat1, long1) = (1,1) (lat2, long2) = (1,1)
Sortie Attendue	**0.0**

Cas 4 :

Description Cas	**Au moins un des 4 paramètres est invalide (ici lat1 > 90)**
Entrée	(lat1, long1) = (91,1) (lat2, long2) = (1,1)
Sortie Attendue	**Erreur : “La latitude doit être comprise entre -90° et 90°“**

Cas 5 :

Description Cas	Au moins un des 4 paramètres est invalide (ici long2 > 180)
Entrée	(lat1, long1) = (1,1) (lat2, long2) = (1,181)
Sortie Attendue	**Erreur : “ La longitude doit être comprise entre -180° et 180°“**

V.7. Test de LoadData

Cas 1 :

Description Cas	Cas de base
Données	**Sensors.csv :** **(Sensor0, 44, -1)** **(Sensor1, 44, -0.3)** **Providers.csv :** **(Provider0,Cleaner0)** **(Provider1,Cleaner1)** **Cleaners.csv :** **(Cleaner0, 45.333333, 1.333333, 2019-02-01 12:00:00, 2019-03-01 00:00:00)** **(Cleaner1, 46.666667, 3.666667, 2019-02-01 12:00:00, 2019-03-01 00:00:00)** **Measurements.csv :** **(2011-01-01 12:00:00, Sensor0, O3, 50.25)** **(2012-01-01 12:00:00, Sensor0, O3, 53.5)** **(2013-01-01 12:00:00, Sensor0, NO2, 74.5)** **(2014-01-01 12:00:00, Sensor0, NO2, 76.5)** **(2015-01-01 12:00:00, Sensor0, SO2, 41.5)** **(2013-01-01 12:00:00, Sensor0, SO2, 43.5)** **(2017-01-01 12:00:00, Sensor0, PM10, 44.75)** **(2018-01-01 12:00:00, Sensor0, PM10, 45.75)** **(2011-01-01 12:00:00, Sensor1, O3, 60.25)** **(2012-01-01 12:00:00, Sensor1, O3, 63.5)** **(2013-01-01 12:00:00, Sensor1, NO2, 84.5)** **(2014-01-01 12:00:00, Sensor1, NO2, 86.5)** **(2015-01-01 12:00:00, Sensor1, SO2, 51.5)** **(2013-01-01 12:00:00, Sensor1, SO2, 53.5)** **(2017-01-01 12:00:00, Sensor1, PM10, 54.75)** **(2018-01-01 12:00:00, Sensor1, PM10, 55.75)** **Attributes.csv** **(O3,µg/m3,concentration d'ozone)** **(SO2,µg/m3,concentration de dioxyde de soufre)** **(NO2,µg/m3,concentration de dioxyde d'azote)** **(PM10,µg/m3,concentration de particules fines)** **Users.csv** **(User0,Sensor70)** **(User1,Sensor36)**

Entrée	**/**
Sortie Attendue	**Pas de sortie mais on doit avoir ces structures de bien remplies :** **hmapIdSensor** **hmapIdPrivateIndividual** **hmapAttributes** **vecteurMeasurements** **hmapIdCleaner** **hmapIdProvider** **mapCoordSensor;** **hmapIdSensorPrivateIndividual;** **hmapDescriptionAttributes;**