B3201 : Annie A. & Sophanna N. && B3209 : Louis H. & Sébastien G.

LIVRABLE: PROJET GÉNIE LOGICIEL ET MODÉLISATION UML

---------- sujet : AirWatcher ----------

III - Design Document

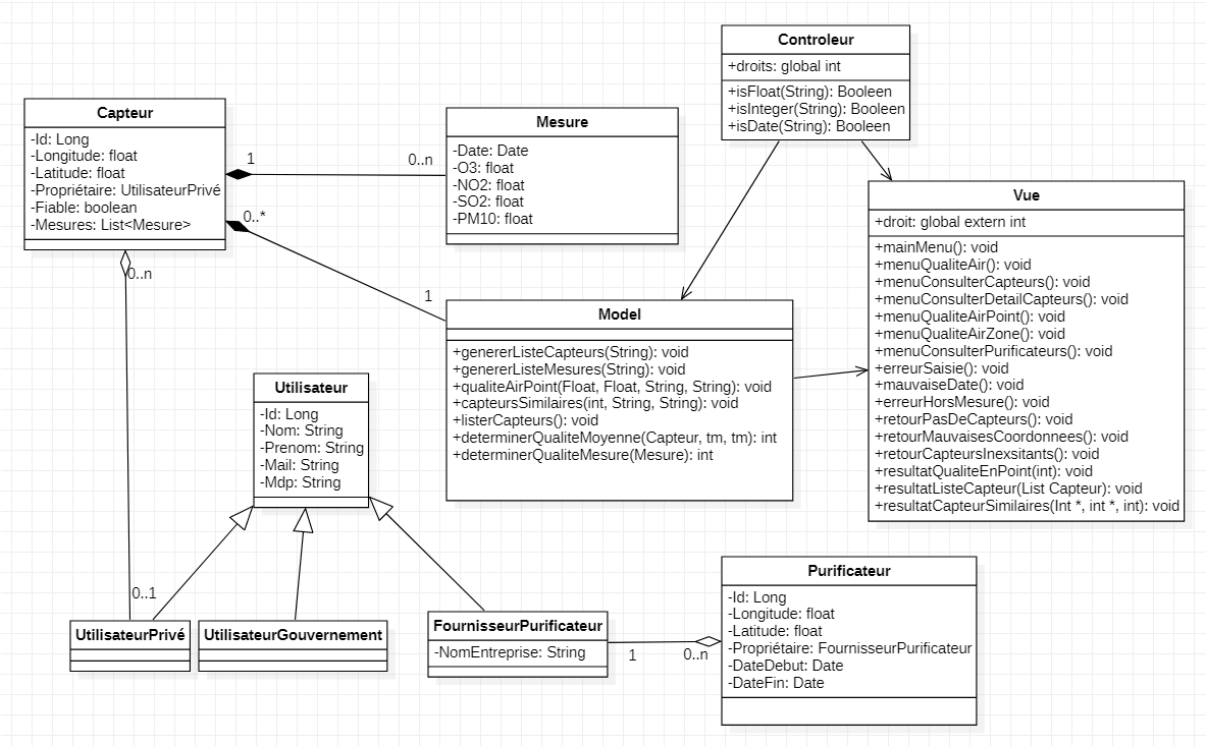
A - Architecture et pattern

Nous avons donc choisi comme architecture le pattern MVC (Model-View-Controller). Cette architecture présente plusieurs avantages dans notre cas :
-La répartition en 3 composants logiques nous permet de travailler indépendamment sur chaque composant et donc une répartition de travail qui nous semble plus naturelle.
-L'isolation du composant "View" permettrait de pouvoir l'améliorer, passer l'affichage en page Web. Sur ce genre de projet court, il nous semble important de rendre le projet facilement améliorable s' il devait être réellement utilisé.
-l'architecture MVC nous semble globalement plus facile de prise en main et présente moins de défauts que les autres.
À noter tout de même qu'étant donné que notre affichage et nos interactions se font sur le terminal, la séparation entre les composants logiques "View" et "Controller" risque de poser problème et devra donc être discutée.

L'utilisation d'une architecture en couches (Layered Architecture) aurait été intéressante pour des projets de plus grande envergure ou des améliorations du système peuvent être envisagées. Dans notre cas, cette architecture ainsi que la répartition en différentes couches nous paraît un peu excessive (Pas d'interface graphique dans notre cas, on utilise que le terminal / Pas de base de données, seulement des fichiers .csv sont utilisés)

Pour l'architecture en silos (Repository Architecture), cette architecture semble adaptée aux logiciels qui disposent d'un grand nombre de fonctionnalités. Pour AirWatcher, le nombre de fonctionnalités à développer étant plutôt faible, l'architecture en silos nous semble peu pertinente. De plus, c'est une architecture qui nous prendrait du temps à maîtriser.

B - Diagramme de classe (UML)



Avec notre choix d'architecture, les méthodes accessibles par l'utilisateur sont séparées des objets qui sont interrogés, ainsi, ces méthodes ne sont pas représentées sur ce diagramme de classes qui se concentre uniquement sur les liens entre les différentes classes que nous allons utiliser. Notons que certaines méthodes implémentées dans les classes (comme les setters/getters et les constructeurs/destructeurs) ne sont pas représentées.

C - Diagramme de séquence des trois scénarios principaux

Diagramme de séquence pour le scénario 1: Qualité de l'air en un point. Les données sont stockées dans les différents .csv

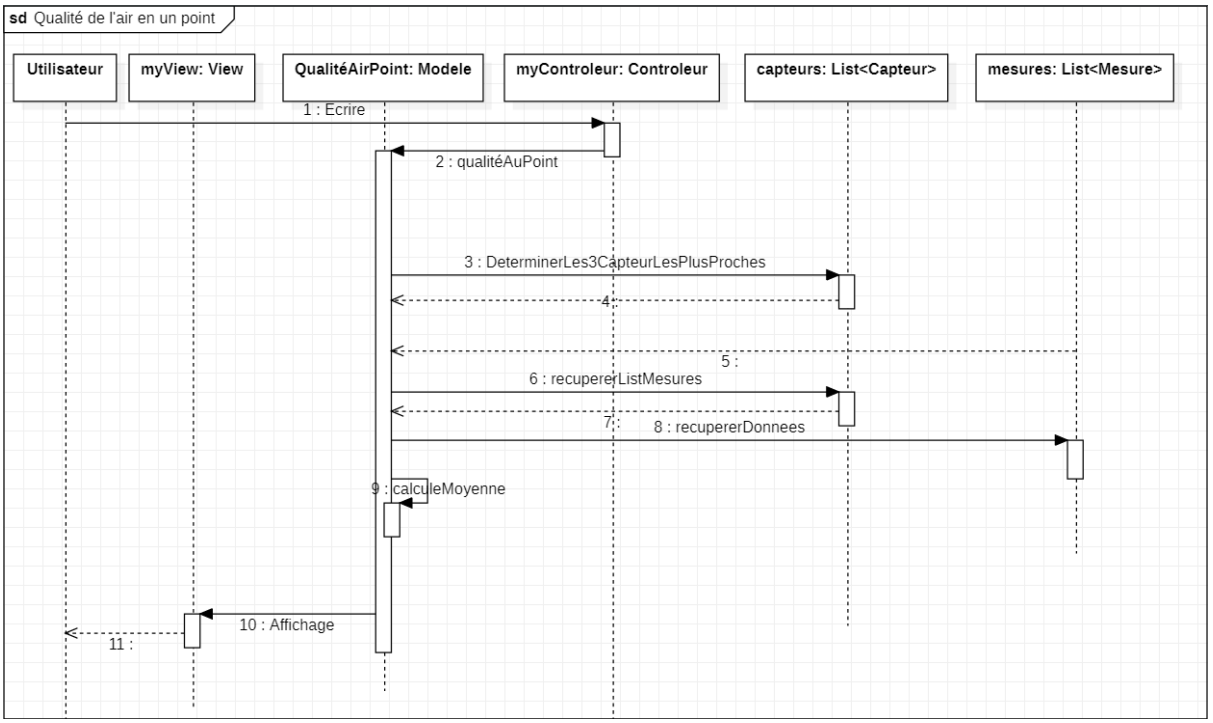


Diagramme de séquence pour la méthode 2: Similarités entre les capteurs: compare le capteur sélectionné par l'utilisateur avec tous les autres capteurs et calcule des valeurs de similarité.

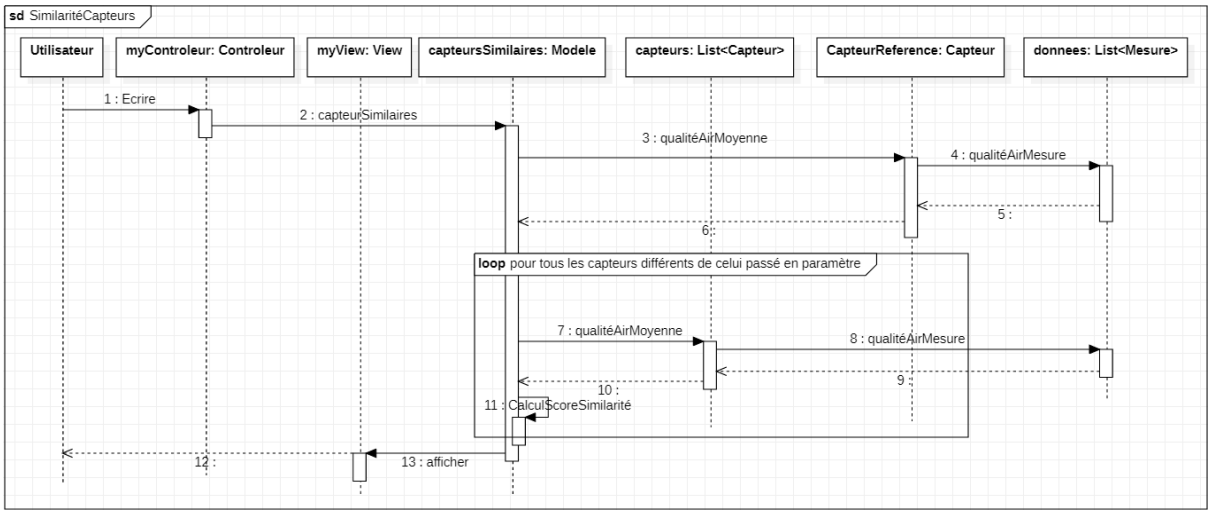
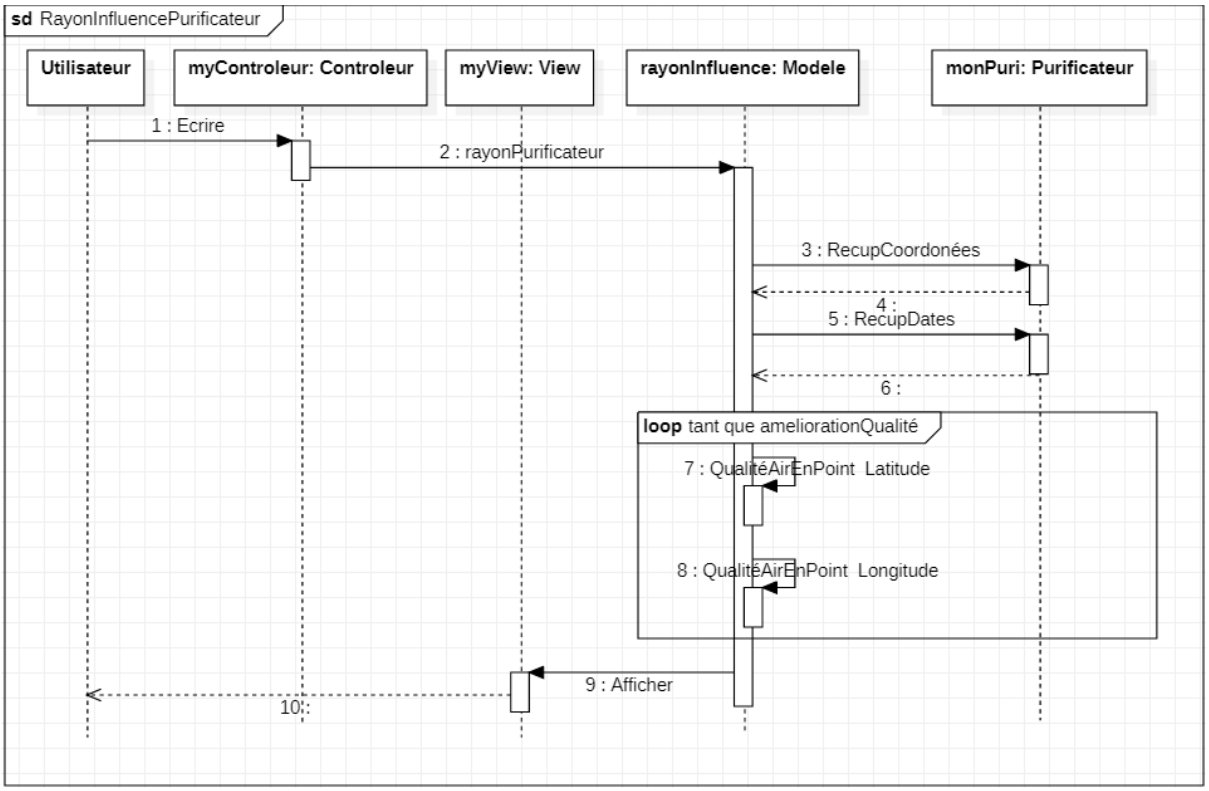


Diagramme de séquence pour le scénario 3: Rayon d'influence des purificateurs



D - Description et pseudo-code des trois algorithmes principaux

Qualité de l'air en un point:

```
Entrée:
point, dateDebut, dateFin

Algo:
Récupérer les informations des capteurs
Trouver les trois capteurs les plus proches du point
Récupérer les mesures des trois capteurs entre dateDebut et dateFin
Pour chaque capteur:
     Calculer les valeurs moyennes des différents gaz mesurés sur
     l'intervalle de temps
Pour chaque gaz:
     Calculer la moyenne pondérée (par la distance) de la valeur
     du gaz au point
Calculer la qualité de l'air (indice ATMO)

Sortie: indice de qualité de l'air
```

Trier les capteurs par similarité:

Entrée:
le capteur: *c, dateDebut, dateFin*

Algo:
Récupérer les informations des capteurs
Récupérer les mesures entre *dateDebut* et *dateFin*
Pour chaque mesure par *c*:
    Calculer l'indice de qualité de l'air Q_{date}
Pour chaque capteur *c'* différent de *c*:
    score= 0
    Pour chaque mesure par c':
        Calculer l'indice de qualité de l'air Q_{date}' de cette mesure
        score+= $|Q_{date}-Q_{date}'|$
Trier les capteurs par score croissant

Sortie: la liste des capteurs et de leurs scores triés

Rayon d'influence des purificateur:

Entrée:
purificateurId

Algo:
Récupérer les informations correspondant à purificateurId, et les mettre dans l'objet *purificateur*
Initialiser *pas*
Initialiser *posX = purificateur.X, posY = purificateur.Y*
Initialiser deltaQualiX = 1, deltaQualiY = 1
Tant que *deltaQualiX* > 0 et *deltaQualiY* > 0:
    posX += pas
    posY += pas
    deltaQualiX=QualitéAirEnPoint(posX,purificateur.Y,purificateur.dateDebut)-QualitéAirEnPoint(posX,purificateur.Y,purificateur.dateFin)
    deltaQualiY=QualitéAirEnPoint(purificateur.X,posY,purificateur.dateDebut)-QualitéAirEnPoint(purificateur.X,posY,purificateur.dateFin)
Rayon =|posX-purificateur.X|-pas
Sortie: Rayon

E - Test plan

Qualité de l'air en un point:

Type de test: Test avec un point normal
Entrée: Latitude; Longitude (Valeurs dans les bornes (à définir) donc suffisamment proche des capteurs pour que la valeur soit fiable); date de début; date de fin
Résultat attendu: Indice de qualité de l'air en ce point

Type de test: Test avec un point inexistant
Entrée: Latitude erronée; Longitude erronée (Valeurs qui sortent des bornes acceptables), date de début; date de fin
Résultat attendu: Erreur: Point inexistant

Type de test: Test avec date de début et date de fin erronées
Entrée: Latitude; Longitude, date de début impossible (date pas encore atteinte); date de fin impossible
Résultat attendu: Erreur: Dates saisies incorrectes

Type de test: Test avec date de début et date de fin très éloignées
Entrée: Latitude (dans la zone mesurée); Longitude (dans la zone mesurée) ; date de début (dans l'intervalle de temps mesuré); date de fin (dans l'intervalle de temps mesuré et très éloigné de la date de début)
Résultat attendu : Une moyenne pondérée par la distance des trois capteurs les plus proches sur tout l'interval de temps
Remarques: ce test évalue la complexité car il oblige à moyenner toutes les valeurs des trois capteurs

Type de test: Test avec un point sur le capteur
Entrée: Longitude et latitude correspondent aux coordonnées d'un des capteurs de la base de données; Date de début comprise dans l'interval de temps ou les capteurs ont mesurés ;Date de fin compris dans l'intervalle de temps où les capteurs ont mesurés et pas excessivement éloignée de la date de début
Résultat attendu : la moyenne de la qualité de l'air du capteur pendant l'intervalle

Type de test: Test avec un point loin d'un capteur
Entrée: Longitude et latitude appartenant à la zone mesurable, mais étant très éloignés de tous les capteurs; date de début; date de fin
Résultat attendu: Indice de la qualité de l'air en ce point, calculé par les capteurs les plus proches
Remarques: On pourra prendre une coordonnée très proche de la limite, ou même au maximum, un point se situant sur la limite de la zone

Similarité des capteurs:

Type de test: Cas normal
Dataset : Usuel, celui fourni pour le projet
Entrée: un capteur quelconque existant;date de début (dans l'interval mesuré); date de fin(dans l'interval mesuré)
Résultat attendu: la liste de tous les capteurs avec un score de similarité pour chacun.

Type de test: Tous les capteurs sont identiques
Dataset : Modifié, tous les capteurs ont des qualité similaires
Entrée: un capteur quelconque existant;date de début (dans l'intervalle mesuré); date de fin(dans l'intervalle mesuré)
Résultat attendu: la liste de tous les capteurs avec le même score pour tout les capteurs

Type de test: Un seul capteur disponible
Dataset : Modifié, tous les capteurs sauf un ont été retirés
Entrée: le capteur restant;date de début (dans l'intervalle mesuré); date de fin(dans l'intervalle mesuré)
Résultat attendu: un message spécifiant qu'il n'y a aucun capteur à comparer.

Type de test: Un capteur très différent des autres
Dataset : Modifié, on ajoute un capteur ayant des qualités très différentes de tous les autres
Entrée: le capteur en question, date de début (dans l'intervalle mesuré); date de fin(dans l'intervalle mesuré)
Résultat attendu: la liste de tous les capteurs avec un score de similarité pour chacun
Remarques: Ces scores seront très mauvais mais il n'y aura pas de message d'erreur

Type de test: Un capteur inexistant
Dataset : Usuel
Entrée: un capteur n'étant pas présent dans la base de données; date de début (dans l'intervalle mesuré); date de fin(dans l'intervalle mesuré)
Résultat attendu: Erreur : Capteur saisi incorrect.

Type de test: Test avec date de début et date de fin erronées
Entrée: Latitude; Longitude, date de début impossible (date trop ancienne); date de fin impossible (date pas encore atteinte)
Résultat attendu: Erreur: Dates saisies incorrectes

Rayon d'influence des purificateurs:

Type de test: Rayon extrêmement petit
Dataset : Modifié, avec un purificateur entouré de mauvais points
Entrée: le purificateur concerné
Résultat attendu: Erreur: Rayon d'influence inexistant
Remarques: le rayon sera minimum de valeur 0

Type de test: Grand rayon: l'air est bon de partout
Dataset : Modifié, avec un purificateur sans obstacle sur toute la carte, le rayon maximal dépassant le bord de la zone limite
Entrée: le purificateur concerné
Résultat attendu: le rayon maximal pour former un cercle, c'est à dire, sans dépasser la zone mesurable

Type de test: Cas usuel: Un point moins bon au milieu
Dataset : Usuel
Entrée: un purificateur bloqué par un point moins bon dans la zone mesurable
Résultat attendu: le rayon minimal entre ce purificateur et ce point

Type de test: Un purificateur en bord de zone mesurée
Dataset : Modifié, avec un purificateur en bord de zone
Entrée: le purificateur en bord de zone
Résultat attendu: Le rayon de la zone purifié ne prenant en compte que les points dans la zone mesurée