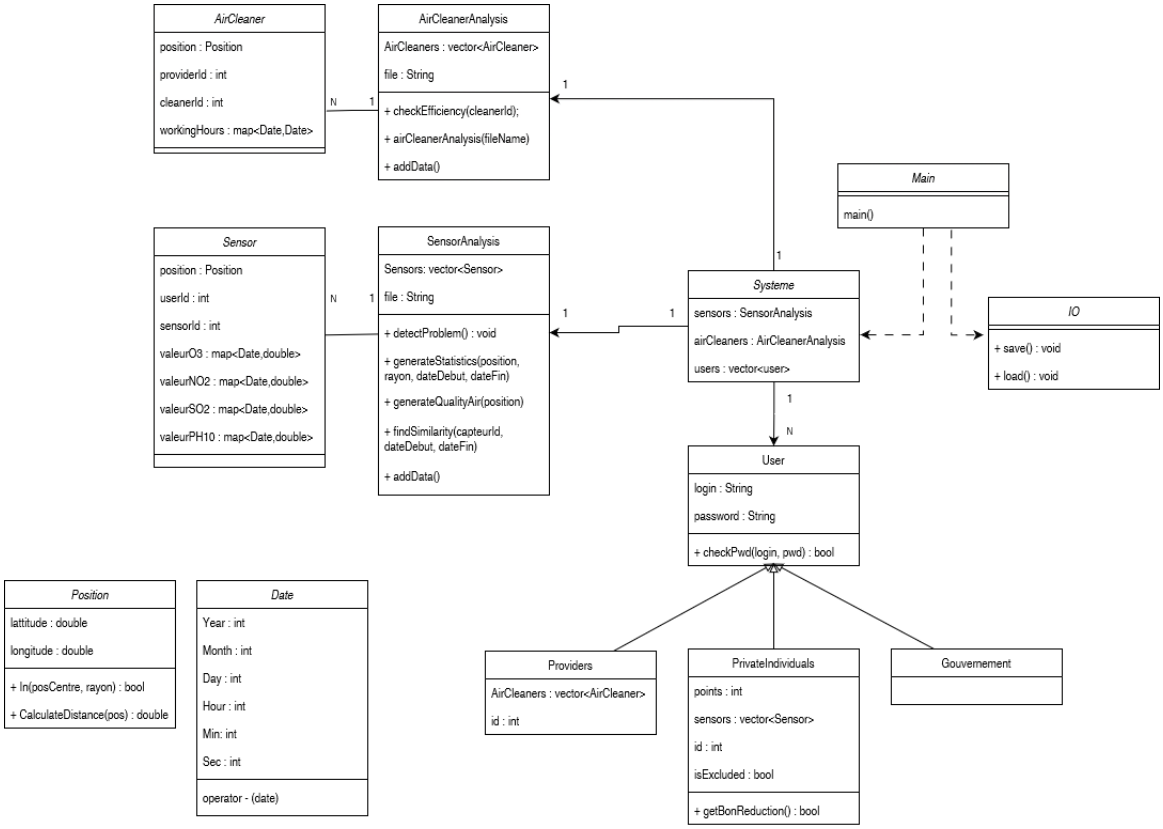
Auteurs : BIAUD, GIRAUDON, MAILLARD, WARIN

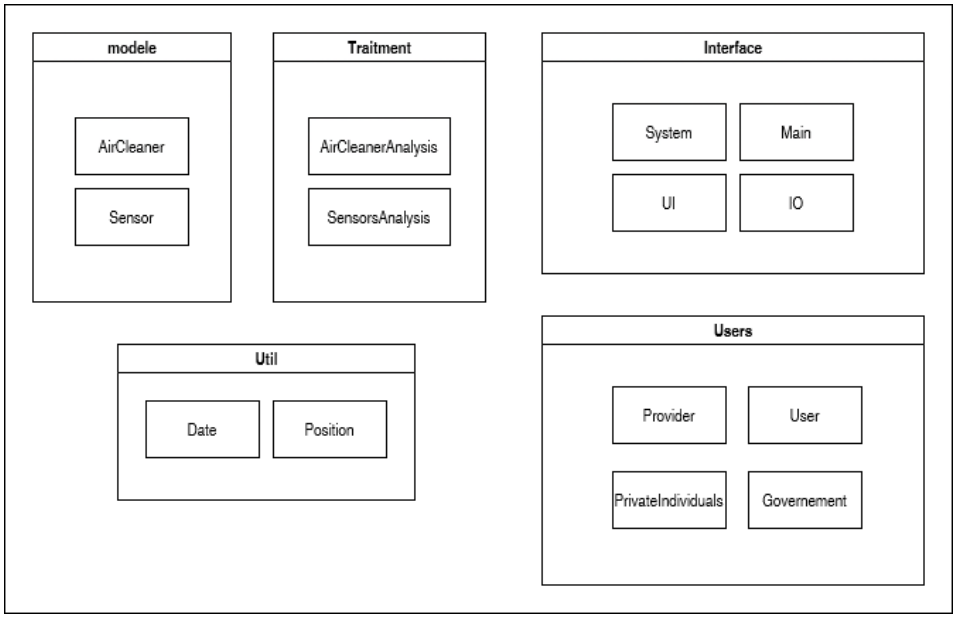
Design Document

I. Diagramme de classe



Auteurs : BIAUD, GIRAUDON, MAILLARD, WARIN

II. Décomposition modulaire

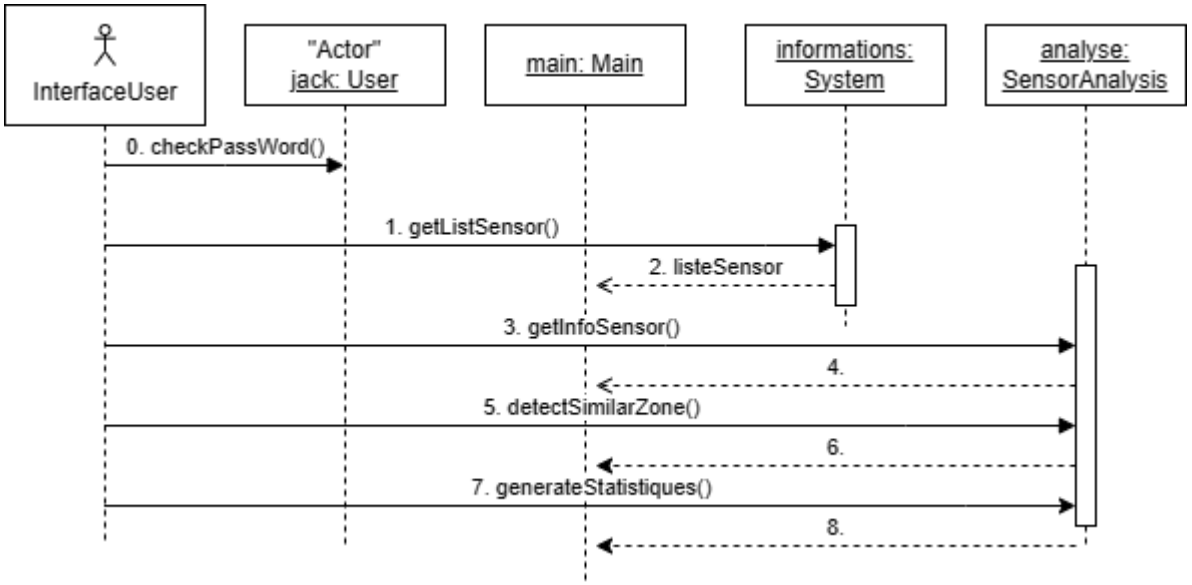


III. Diagrammes de séquences

Les 3 scénarios majeurs

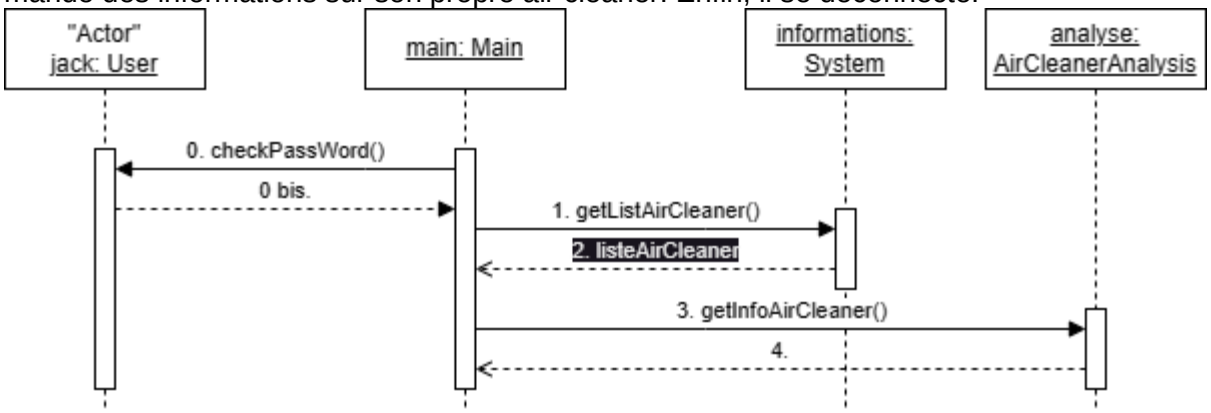
1.

Un utilisateur arrive et se connecte. Puis il demande la liste des capteurs afin d'ensuite en choisir un pour avoir plus d'informations sur celui-ci. Le capteur est annoncé fiable, donc l'utilisateur demande dans l'ordre la qualité de l'air à un instant donné puis la liste des capteurs captant la même qualité d'air. Enfin, il décide de demander les statistiques les plus récentes à sa position. Puis, il se déconnecte.



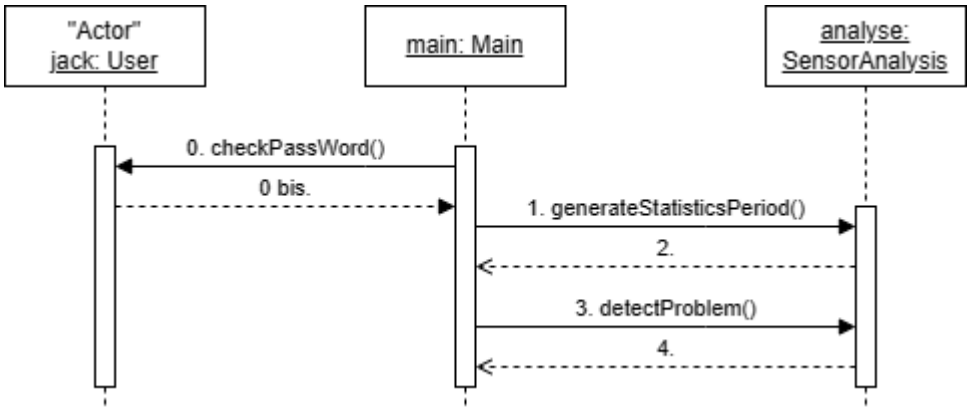
2.

Un provider arrive sur l'application et se connecte. Il demande la liste des airs cleaners puis demande des informations sur son propre air cleaner. Enfin, il se déconnecte.



3.

Un membre du gouvernement se connecte. Il cherche à obtenir les statistiques sur la qualité de Paris de cette année puis il fait un scan des capteurs avant de se déconnecter.



IV. Principaux algorithmes

```
detectProblem() {
        Si la mesure n et n+1 a un écart supérieur à 15% sur une des quatres données
                mettre la variable true à doute
        Si la mesure n et n+1 a de nouveau un écart supérieur à 15% sur une des quatres données et doute = true
                Return true
        Return false
}

findSimilarity(capteurId, dateDebut, dateFin) {
        Récupérer la liste de tous les capteurs
        parcourir la liste des capteurs
                Si un capteur possède la même moyenne et écart type sur la période de temps avec un écart de 5%
et distance > 200m
                        alors ajouter le capteur à la liste similar.
        Return similar
}

//Renvoie la zone d'éfficacité du aircleaner
checkEfficiency(Aircleaner aAircleaner) {
```

```
        Récupérer la liste de tous les capteurs

        Pour chaque workingHours
                efficace = true
                distance = 100mètres
                Tant que Efficace
                        on choisi le capteur qui maximise la distance avec le air cleaner mais < distance
                        Si la mesure Avant la worling hours > 15% de la mesure après les workingHours
                                alors distance+=100
                        Sinon
                                Efficace = false
                somme += distance
                nb++

        Return somme/nb
}
```

V. Fichier de sauvegarde

Nous devons créer des fichiers de sauvegarde afin de se remémorer les login et mot de passe des utilisateurs. Ces derniers ne seront pas cryptés.

Il s'agit du fichiers data.csv. Il contient les données dans le format suivant suivant :

LOGIN ; MDP ; TYPE ; [NB de point] ; [NB de capteurs/airCleaners] ; [ID du capteur/airCleaner 1] ; [ID du capteur/airCleaner 2]…

Le type est 1 si c'est un utilisateur disposant d'un capteur ou non, 2 si c'est un fournisseur d'airCleaner ou 3 si c'est un membre du gouvernement.

Les valeurs suivantes sont présentes pour les utilisateurs et les fournisseurs uniquement.

Pour un utilisateur, il s'agit du nombre de point qu'il a accumulé, puis du nombre de capteurs qu'il possède et enfin liste des id de ces capteurs.

Pour les fournisseurs, il s'agit uniquement du nombre de airCleaner déployé ainsi que la liste des id de ces derniers.

VI. Tests unitaires

User checkpasswd(String login, String password)

Doit renvoyer user si l'utilisateur est dans le fichier data.csv

nullptr sinon

String getBonReduction()

Si le user.getPoints() est suffisant : Renvoyer vrai

Sinon : Renvoyer faux

bool detectProblem()

Si le capteur possède une anomalie (par exemple un écart de valeur > 10% par heure) on renvoie True

et on appelle aSensor.setIsValid(false)

Sinon on renvoie False

String generateStatistics(pos, startTime, endTime)

Renvoie les statistiques(voir Partie 1) sous forme d'une chaîne de caractères

Si pas de données viables, renvoyer "Pas de données"

String generateStatistics(pos, r, startTime, endTime)

Renvoie les statistiques(voir Partie 1, pas de capteur à moins de 50km) d'une zone sous forme d'une chaîne de caractères

Si pas de données viables, renvoyer "Pas de données"

vector<Sensor> detectSimilarZones(Sensor aSensor, startTime, endTime)

Renvoie un vecteur de capteurs étant similaire (même moyenne et écart-type) triés par ordre décroissant de similarité.

Renvoie un vecteur vide s'il n'existe pas de capteurs similaires.

double checkEfficiency(AirCleaner)

Renvoie la zone d'éfficacité en mètre du airCleaner

String getInfosAirCleaner(AirCleaner aAirCleaner)

Renvoie tous les attributs du airCleaner.

double calculateDistance(position)

Renvoie la distance entre l'objet appelant et la position en paramètre

Ne doit pas renvoyer le valeurs négatives

Bool In(position, r)

Renvoie vrai si la position est dans le cercle donné en paramètre

Sinon renvoie faux