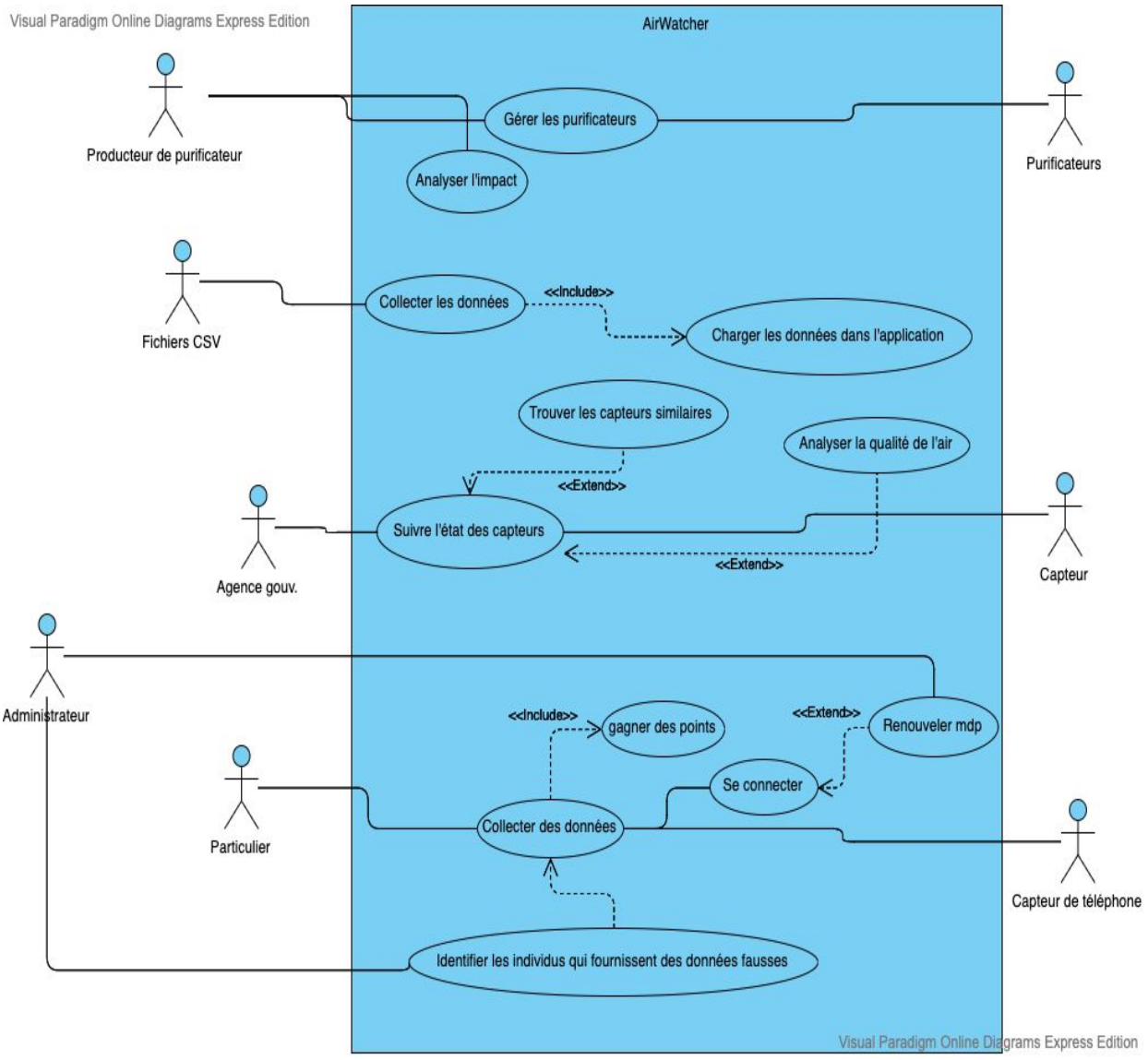
Projet GL - AirWatcher
Dossier de Conception

I. Introduction

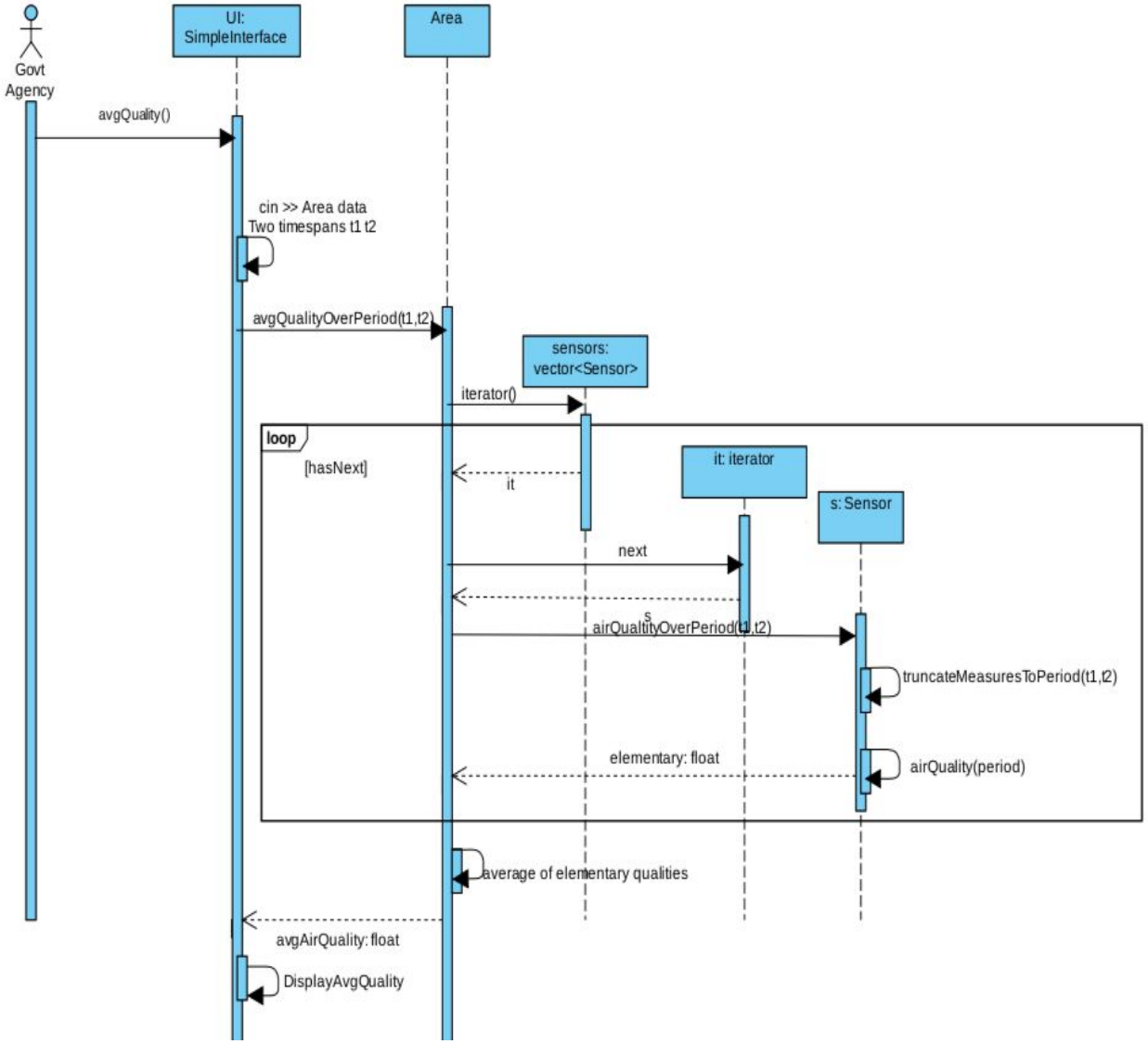
Ce document liste un ensemble de diagrammes UML (cas d'utilisation, de séquence et de classe) permettant de mieux conceptualiser l'architecture et le fonctionnement de notre logiciel.

II. Diagramme de cas d'utilisation

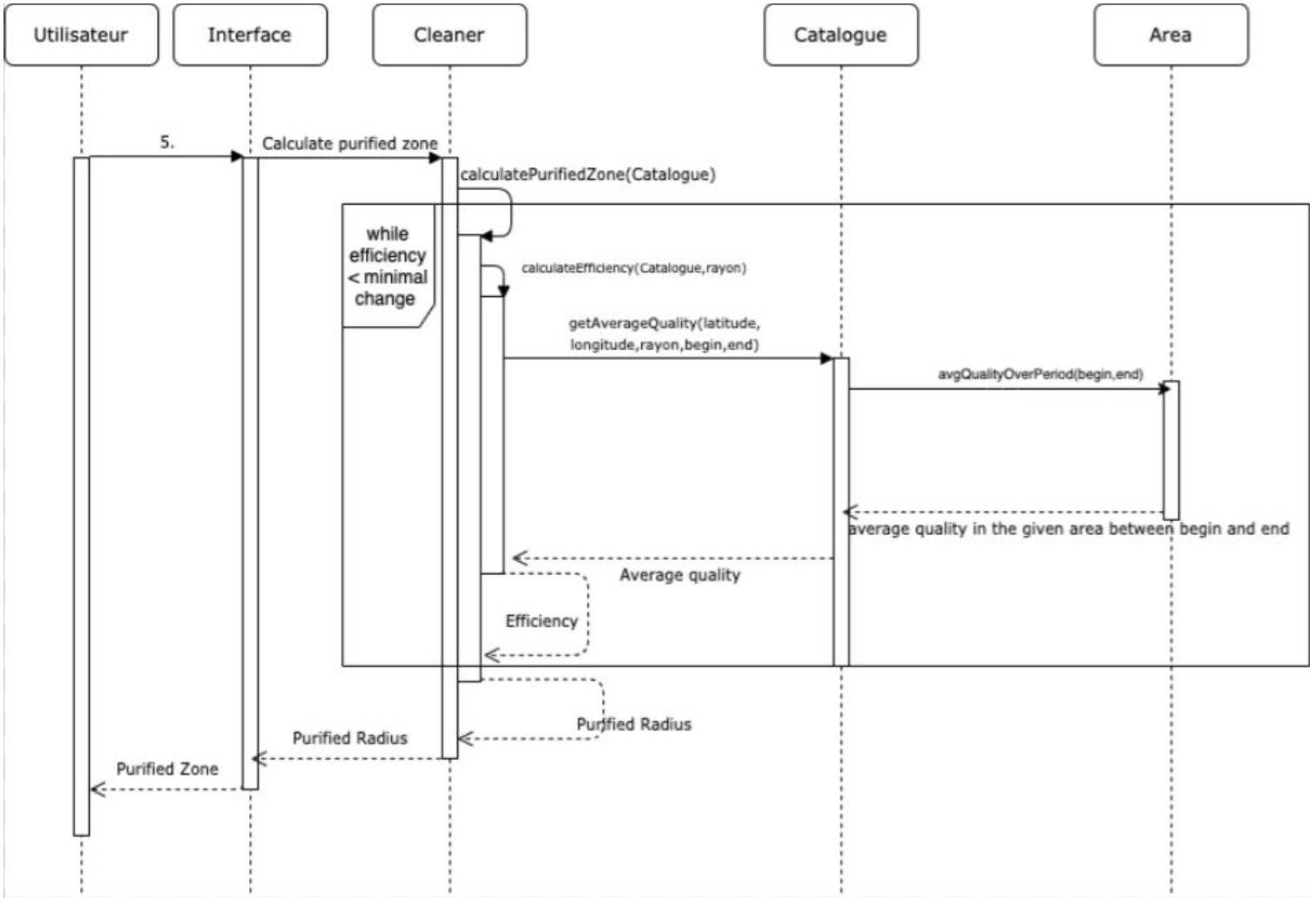


III. Diagrammes de séquence

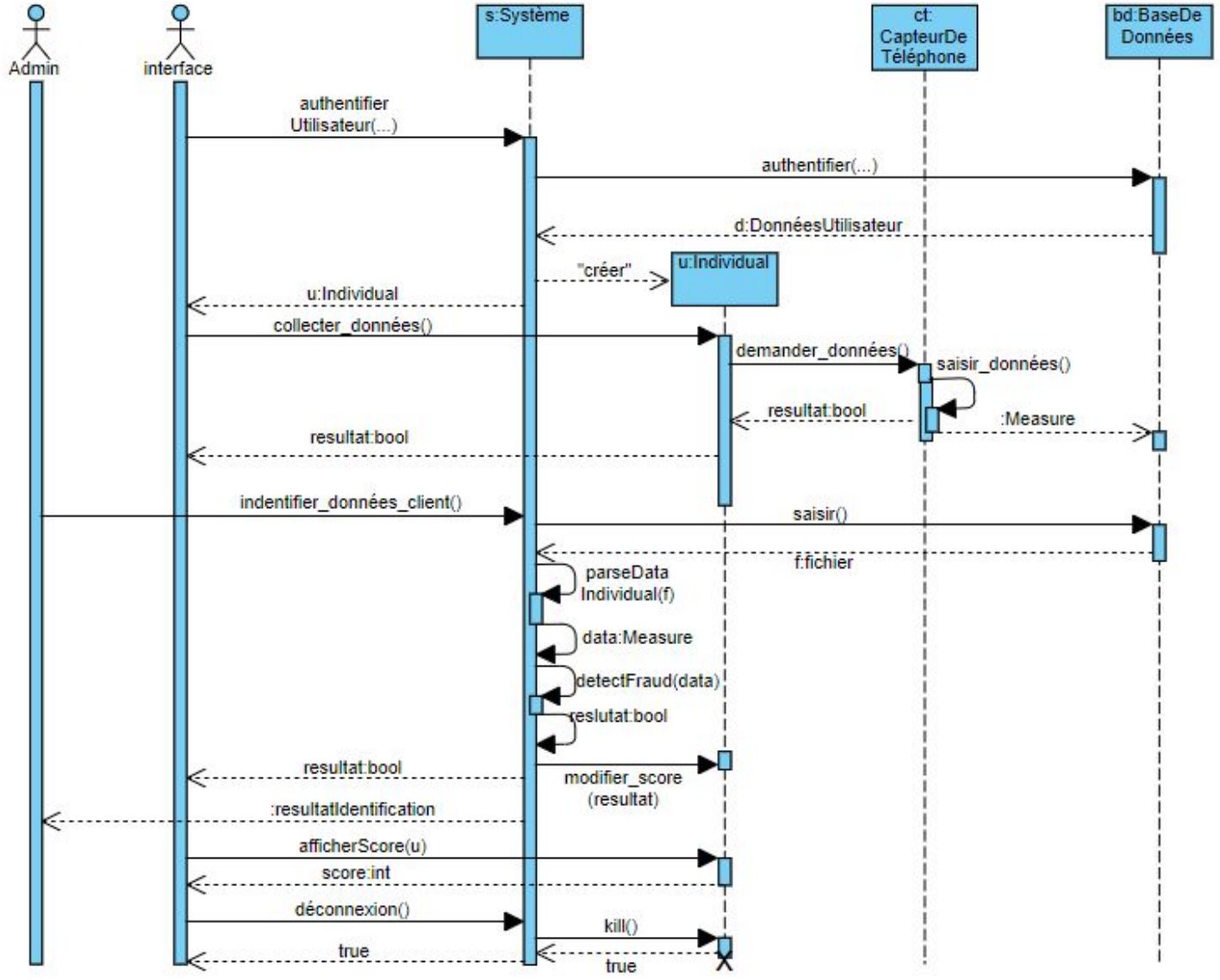
1. Consulter la qualité moyenne de l'air pour une zone donnée pendant une durée donnée



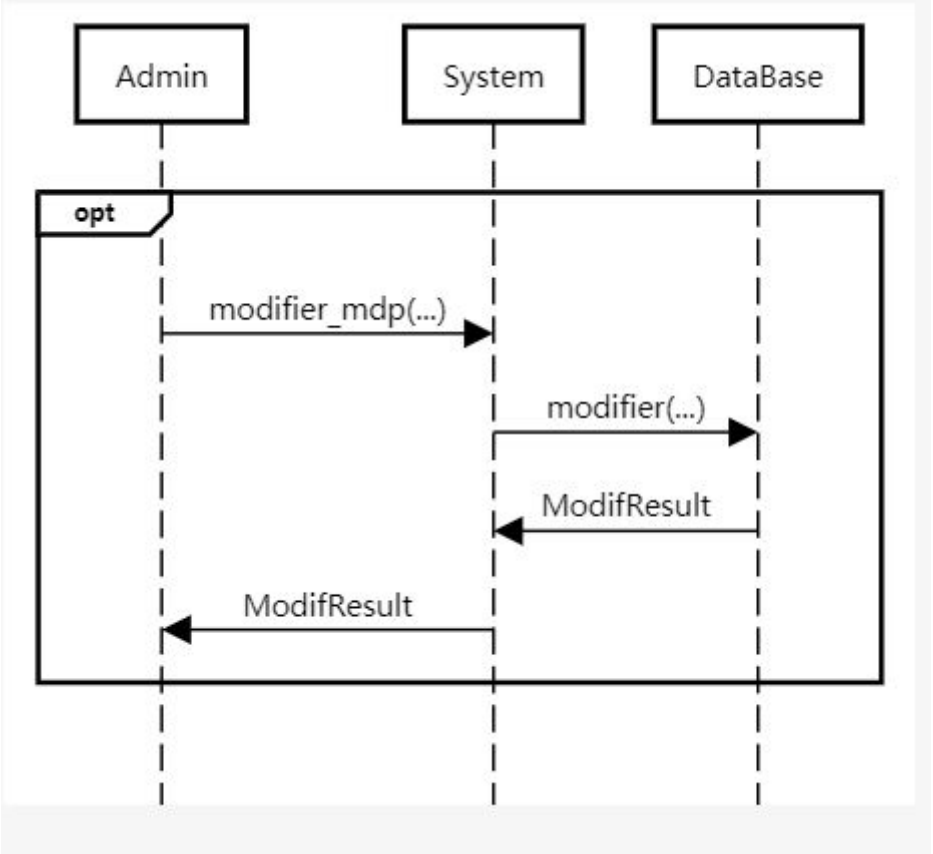
2. Consulter la zone purifiée par une installation donnée



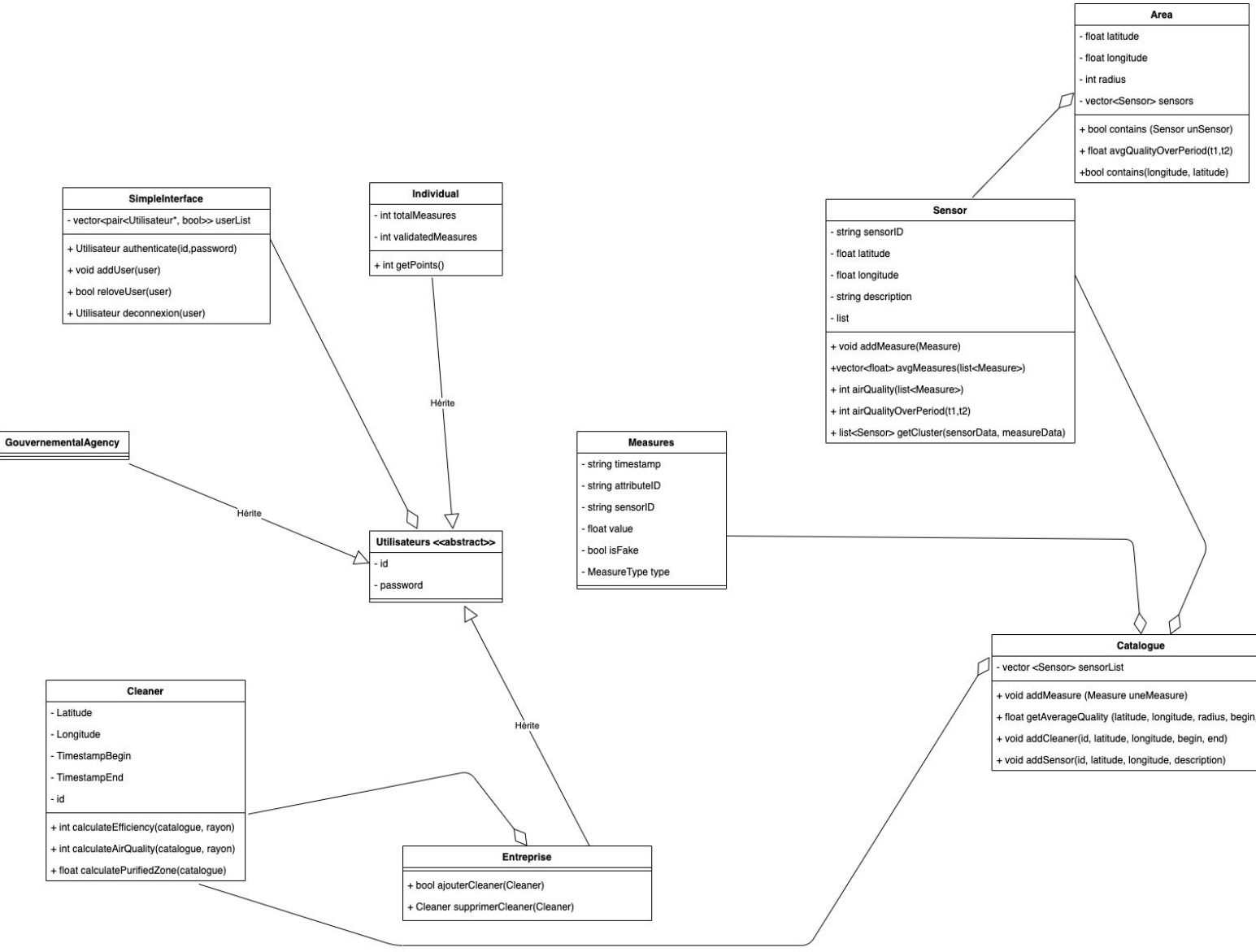
3. Utilisateur individuel



4. Modification du mot de passe



IV. Diagramme de classe



V. Méthodes principales

- float Area::avgQualityOverPeriod (string t1, string t2){}
  Returns the average air quality in area over a given period. Calls quality method on each of the sensors in area and averages the results.
- void Catalogue::detectFraud(Measure* m) {}
  Scans input data against existing record of measures to determine the likeliness of input being compromised.
- float Cleaner::calculateEfficiency(Catalogue* cat, int rayon) const {}
  Algorithm that outputs the difference in air quality before vs after cleaner added.
- float Cleaner::calculatePurifiedZone(Catalogue* cat) const {}
  Algorithm to compute area that has improved after cleaner added.
- vector<float> Sensor::avgMeasures(list<Measure> mesures) {}
  Calculates the average measures for each measure type after looping through measure list embedded in sensor.
- float Sensor::airQuality(list<Measure> data) {}
  Calculates air quality based on the different measures.
- list<Measure> Sensor::truncateMeasuresToPeriod(string t1, string t2) {}
  Restricts measure list to a given period.
- list<Sensor> Sensor::getCluster(vector<Sensor> sensorData) {}
  Returns cluster of sensor, i.e other sensors with similar properties (geographical setting and recorded data).