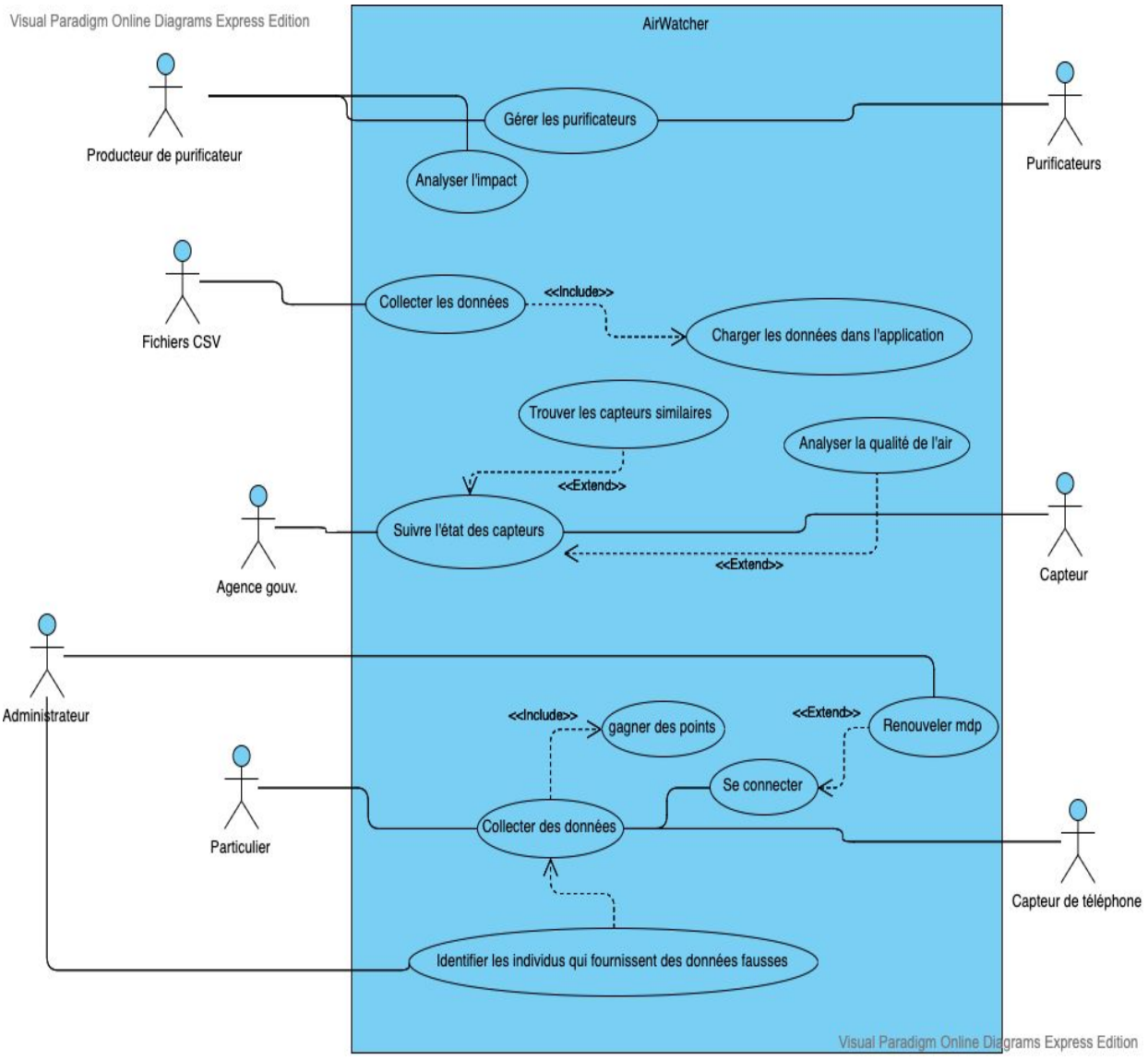
# Projet GL - AirWatcher
# Dossier de Conception



## I. Introduction

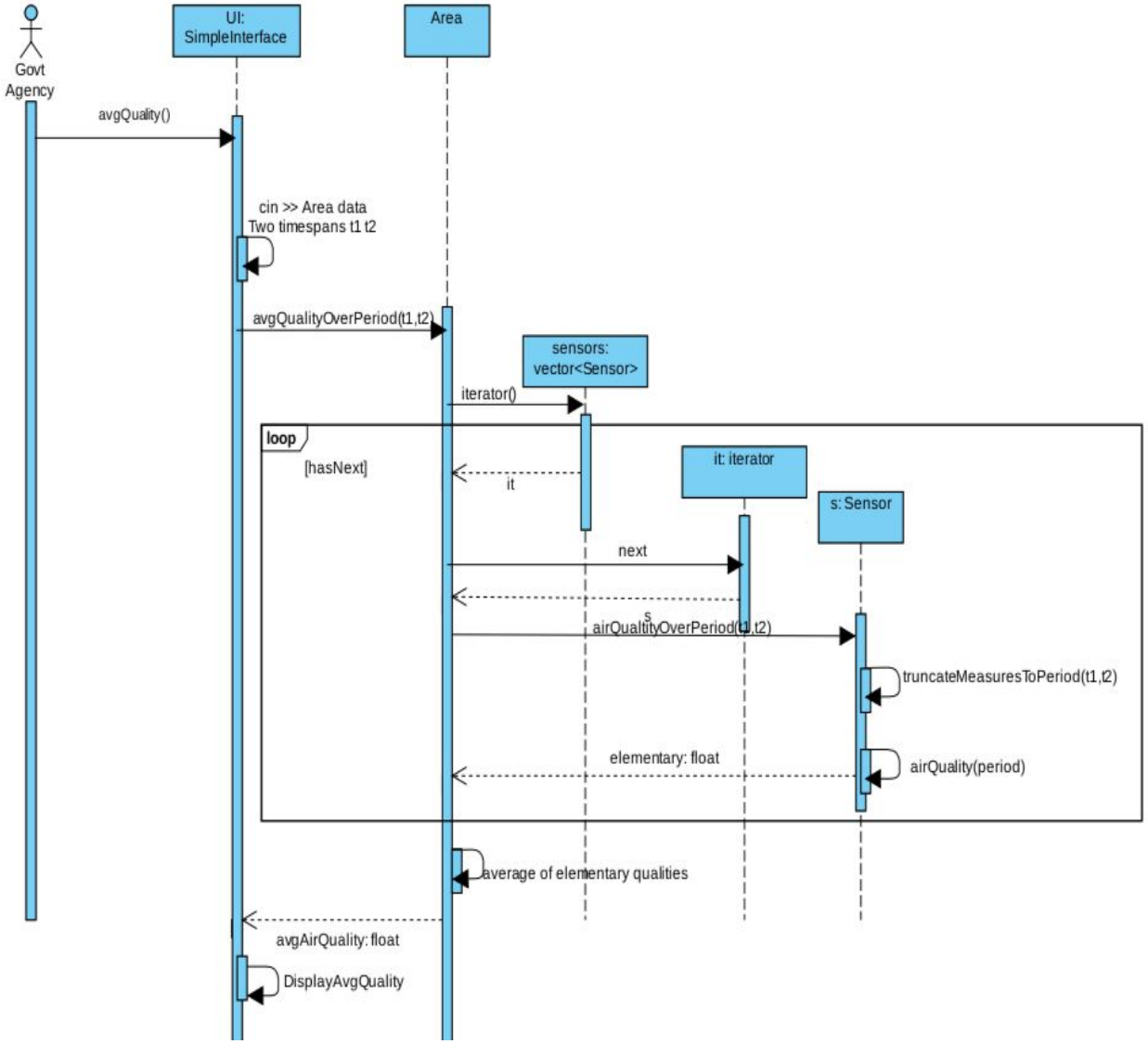
Dans ce document, nous allons aborder la conception de notre logiciel. Cette description sera illustrée par de nombreux diagrammes UML afin de pouvoir détailler de façon précise du fonctionnement de notre application.

# II. Diagramme de cas d'utilisation

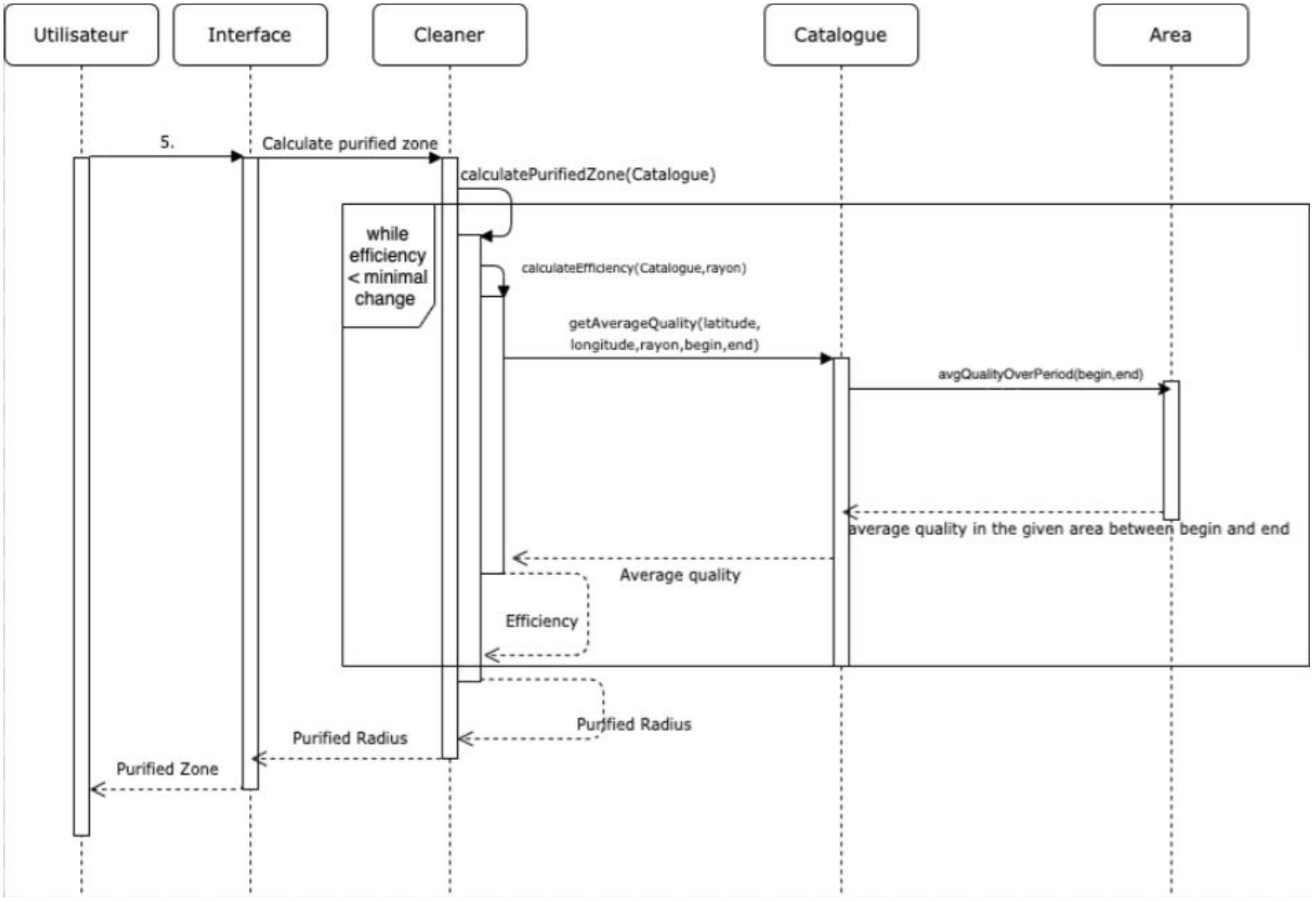


# III. Diagrammes de séquence

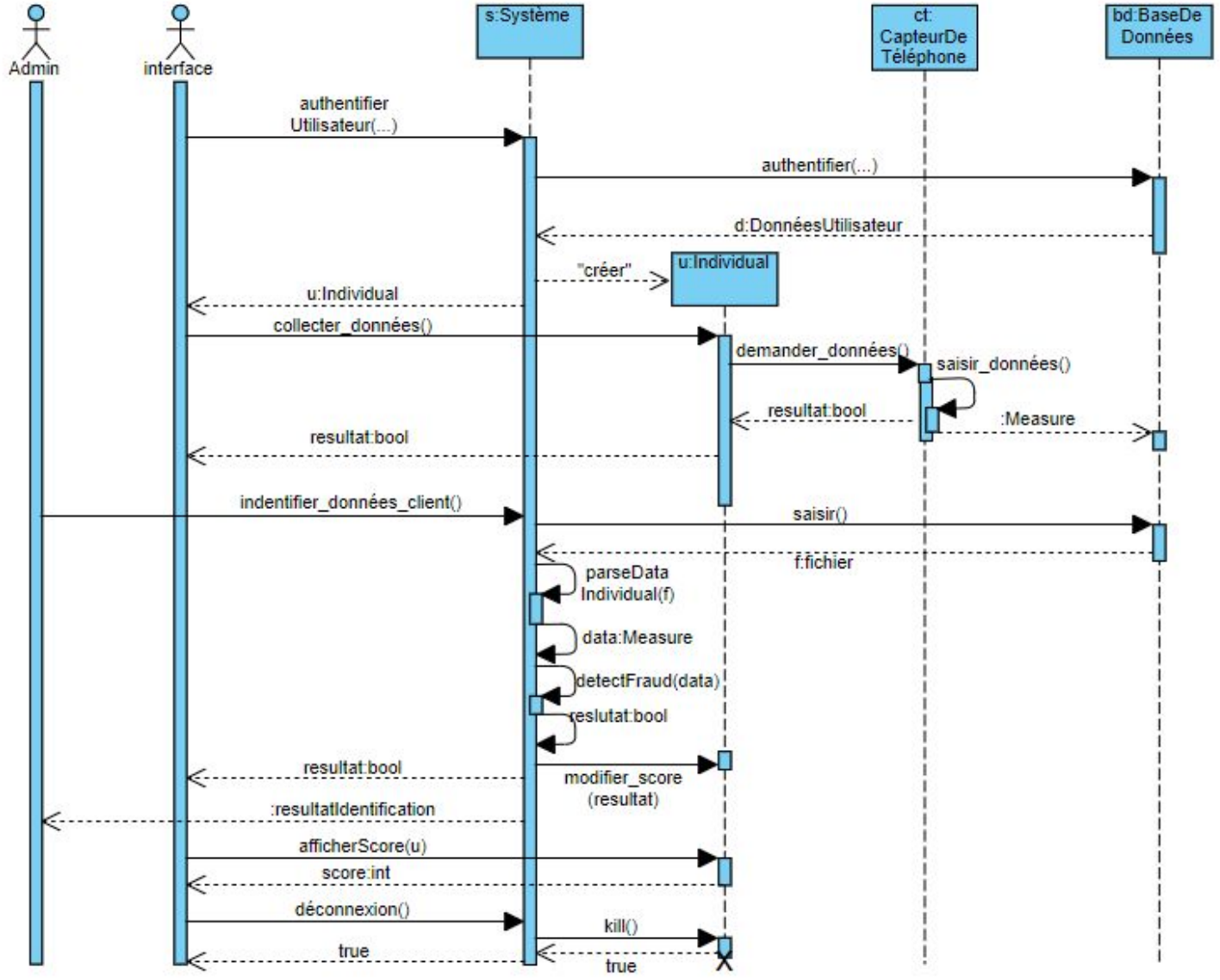
1. Consulter la qualité moyenne de l'air pour une zone donnée pendant une durée donnée



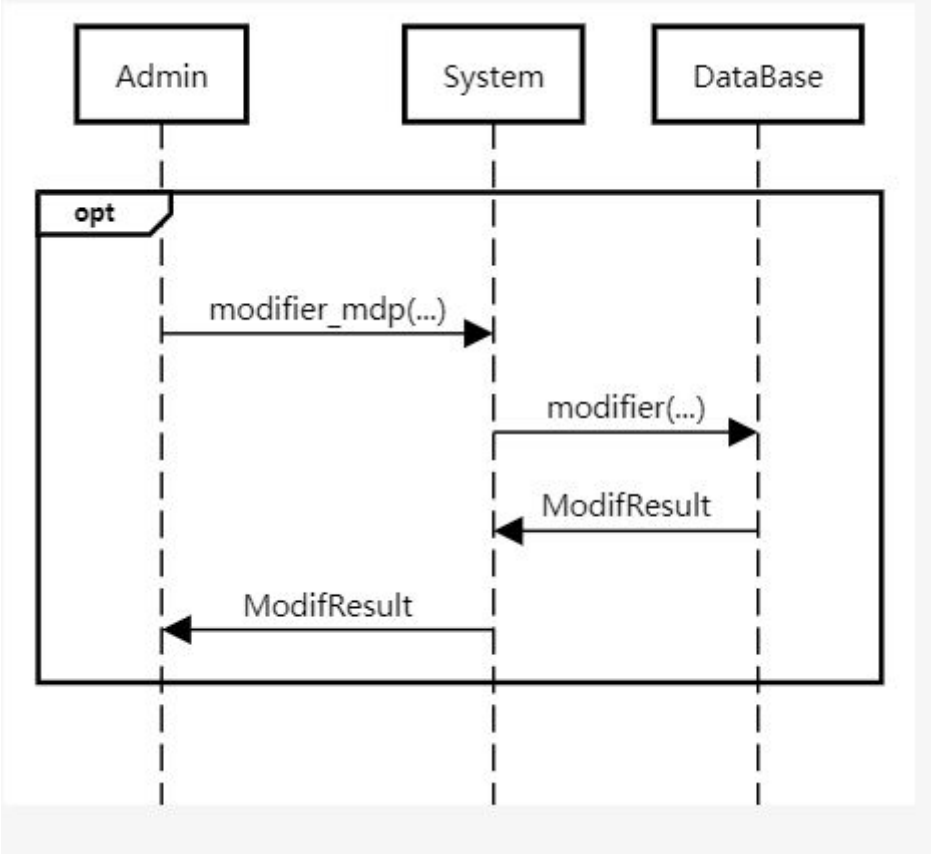
2. Consulter la zone purifiée par une installation donnée



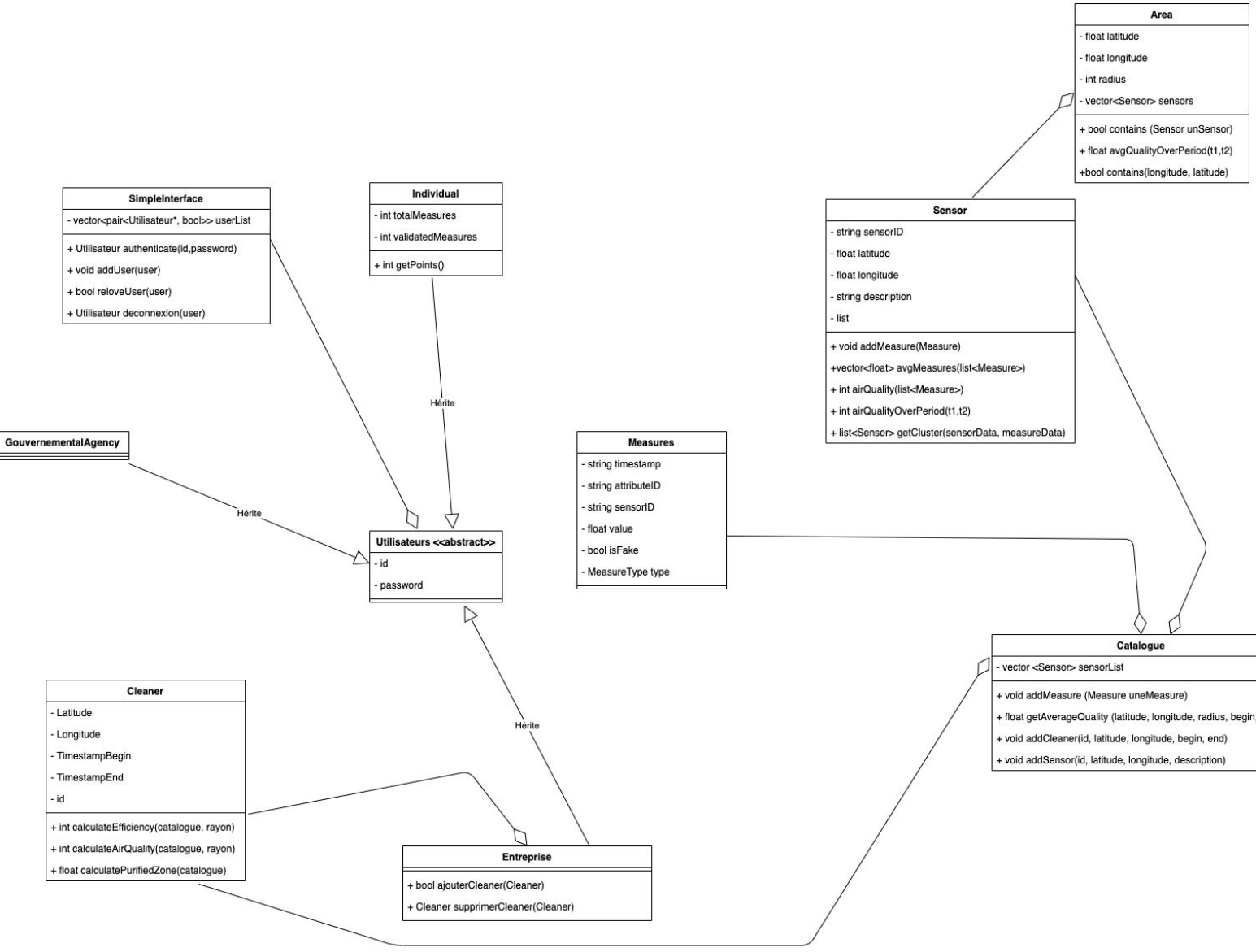
## 3. Utilisateur individuel



4. Modification du mot de passe



# IV. Diagramme de classe



# V. Méthodes principales

- parseSensors(csv file);
- parseReadings(csv file);
- parseMeasureTypes(csv file);
- parseCleaners(csv file);
- parseDataIndividuels(csv file); // get data from individual, check if it's not detected as fraud, if it's good, import it and increment individual score, if it's fraudulent, label it as such
- authentifierUtilisateur(mail, mot de passe);
- deconnexion();
- calculerImpact(cleanerId); // difference in air quality before vs after cleaner added
- calculerZonePurifiee(objet cleaner); // area that has improved after cleaner added
- afficherScore(client); // cout a getter
- detectFraud(Mesure); // get data as input, use some stats to see if it's divergent, in which case label it as fraudulent by setting the flag to true (returning a boolean type thing)
- ajouterAirCleaner(latitude, longitude, description);
- supprimerAirCleaner(cleanerId);
- int obtenirMoyenneQualite(latitude, longitude, timespan); // if timespan ==null, just get general quality of area
- getCluster(sensorId); // e.g. is their values equal or are similar. Returns a list of clusters (lists) i.e sensors sorted into groups based on behavioural similarities (like a score that weighs in their data and localisation etc)
- getSensor(); // gets the sensor for a given Measure, implemented in Measure class