INSA de Lyon

Computer Sciences Department

Software Engineering & UML

GL-UML Lab
AirWatcher

Autors:
Melisse Cochet
Saad ElGhissassi
Jassir Habba
Sekyu Lee
Simon Perret

Professors:
Mrs. Laforest
Mr. Hasan

May 3, 2024

Contents

1 Conception

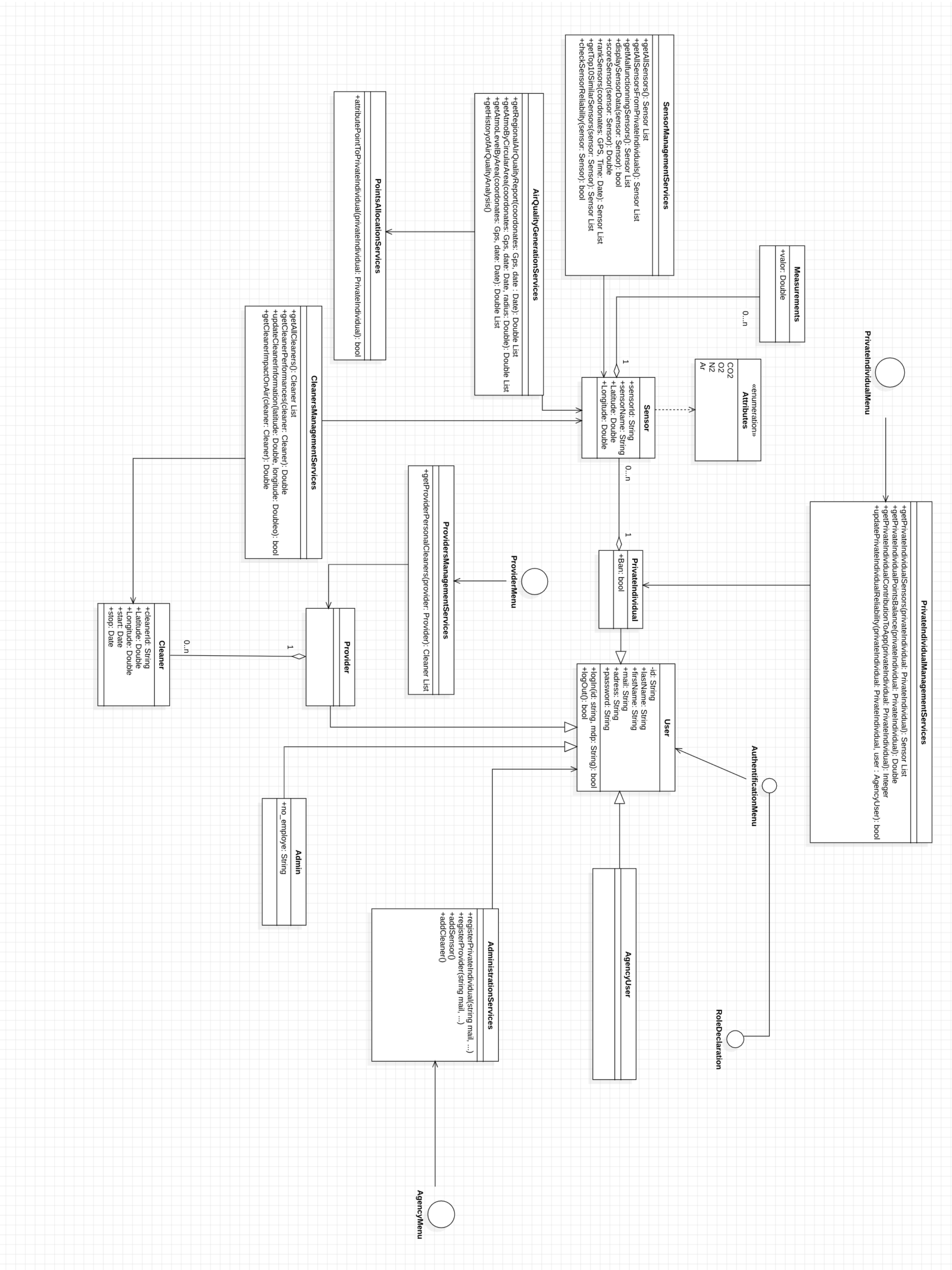
1.1 Architecture

We have chosen to develop our application on a layered based architecture. This enables a concrete Separation of Concerns while developing (Interface Layer - Services Layer - Data Layer) and a better coding organization. The decoupling between layers also allows easier maintenance and updating of the system and a better scalability as the system could handle large amounts of data. Reusability was also a crucial point for choosing this architecture as components developed within each layer, especially within the business logic and data access layers, can be reused across different parts of the application or even in different projects within the same organization. Eventually, this architecture ensures improved security of our application by isolating the layers, making sensitive data operations abstracted away from user interfaces, which was one of our security requirements.

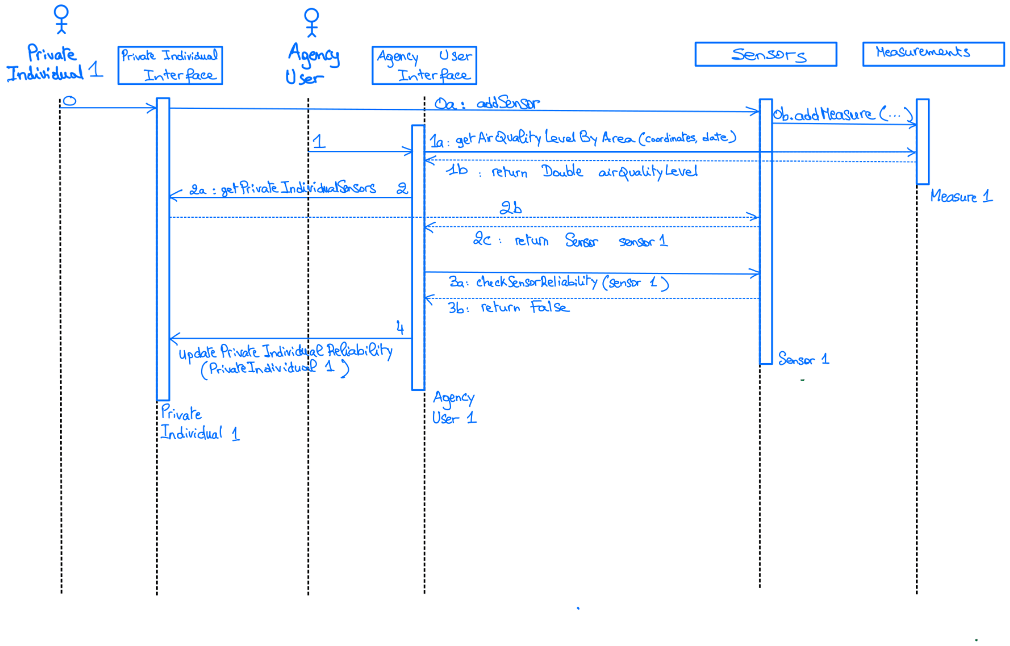
The Repository architecture was also considered. It would have probably been easier to manage, but at high costs : Handling concurrent access to a central repository, especially when the data is stored in CSV files, can be complex and inefficient without the sophisticated concurrency control mechanisms typically provided by a DBMS.

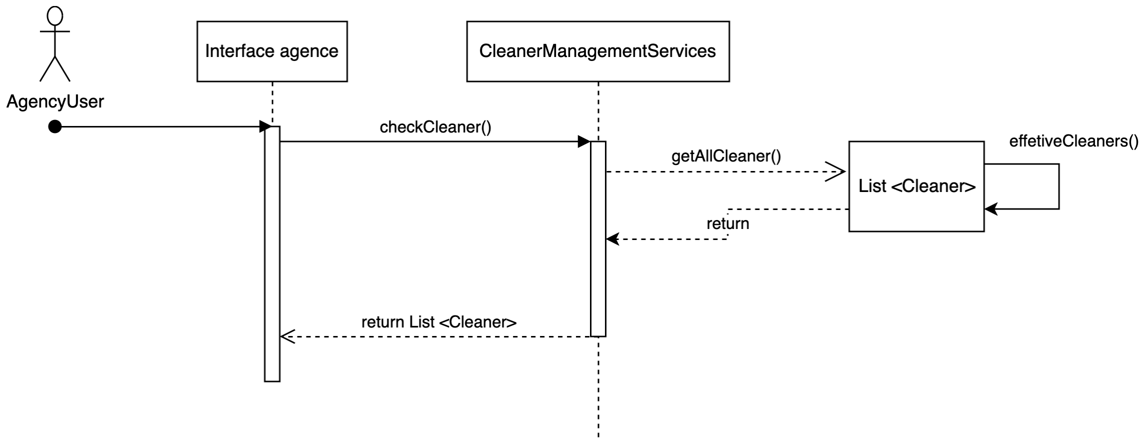
Other architectures seemed totally irrelevant, such as the MVC, as AirWatcher's core functionality revolves more around data processing and analysis rather than user interaction, which might not benefit significantly from the MVC's strong UI orientation.

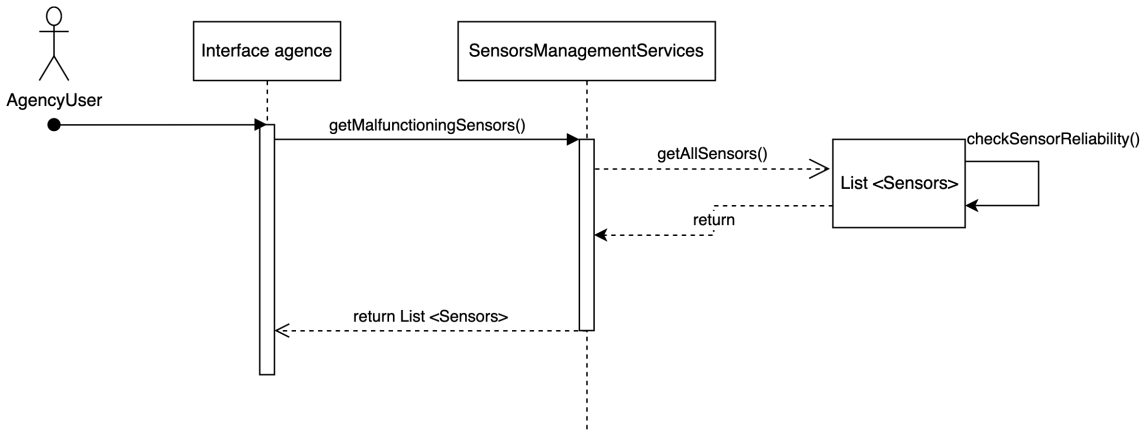
1.2 Class diagram



1.3 Sequences diagram







1.4 Description and pseudo-code

We chose to develop 3 main algorithms such as verifying that the sensors are working correctly, that the air cleaners are effective and finally to verify the veracity of private user sensors.

1.4.1 Analyze data to make sure sensors function correctly

We will select measurements from the sensor provided over the last week and calculate the means for each concentration (O3, SO2, NO2, PM10). Next, we will identify the five closest sensors and assign them weights based on their distance from the selected sensor. Then, we will calculate the means for each concentration during the same period with these weights. Finally, we will compare each concentration with those from the selected sensor, and if the margin of error exceeds 10% for at least one concentration, the sensor will be deemed malfunctioning.

Algorithm 1 Check the operation of a sensor

```
Require: sensorSelected
Ensure: boolean (false if the sensor is faulty)
checking ← true
meanO3, meanSO2, meanNO2, meanPM10 ← calculateMeans(sensorSelected)
nearbySensors ← empty list
for i ← 1 to 5 do
    sensor ← closestSensor(sensorSelected, nearbySensors)
    nearbySensors.append(sensor)
end for
weightedMeanO3, weightedMeanSO2, weightedMeanNO2, weightedMeanPM10 ←
calculateWeightedMeans(sensorSelected, nearbySensors, 7)
for concentration ∈ {meanO3, meanSO2, meanNO2, meanPM10} do
    marginError ← calculateMarginError(concentration, weightedConcentration)
    if marginError > 0.1 then
        checking ← false
    end if
end for
return checking
```

Required functions

Algorithm 2 distance

```
Require: sensor1 sensor2
Ensure: double
return distanceEuclidienne
```

Algorithm 3 closestSensor

```
Require: sensor1, excludedSensor                    ▷ List of sensors to exclude
Ensure: Returns the closest sensor
dist ← +∞
closestSensor ← 0
for sensor not in excludedSensor do
    if Distance(sensor1, sensor) < dist then
        dist ← Distance(sensor1, sensor)
        closestSensor ← sensor
    end if
end for
return closestSensor
```

Algorithm 4 calculateMeans

Require: $Periodisthetimespanoverwhichwewanttoperformthecalculation(uptonow), indays(integer)$
Ensure: return the means for each concentration (O3,SO2,NO2,PM10)

for each $measurement$ in mesures of $sensor$ **do**
    $period$ ←select the period
    avg ←average of concentration
end for
return $mean03, meanSO2, meanNO2, meanPM10$

Algorithm 5 Calculate Weighted Means of Concentrations

Require: $selected\_sensor$, $nearby\_sensors$, $period$ ▷ Selected sensor, list of nearby sensors, and the period in days
Ensure: Returns the weighted averages for each concentration (O3, SO2, NO2, PM10)

$weight\_tot \leftarrow 0$
$meanO3 \leftarrow 0$
$meanSO2 \leftarrow 0$
$meanNO2 \leftarrow 0$
$meanPM10 \leftarrow 0$
for each $sensor$ in $nearby\_sensors$ **do**
    $weight \leftarrow \text{Distance}(selected\_sensor, sensor)$
    $weight\_tot \leftarrow weight\_tot + weight$
    $tempMeanO3, tempMeanSO2, tempMeanNO2, tempMeanPM10 \leftarrow \text{Calculate\_means}(sensor, period)$
    $meanO3 \leftarrow meanO3 + weight \times tempMeanO3$
    $meanSO2 \leftarrow meanSO2 + weight \times tempMeanSO2$
    $meanNO2 \leftarrow meanNO2 + weight \times tempMeanNO2$
    $meanPM10 \leftarrow meanPM10 + weight \times tempMeanPM10$
end for
if $weight\_tot > 0$ **then**
    $meanO3 \leftarrow meanO3/weight\_tot$
    $meanSO2 \leftarrow meanSO2/weight\_tot$
    $meanNO2 \leftarrow meanNO2/weight\_tot$
    $meanPM10 \leftarrow meanPM10/weight\_tot$
end if
return $(meanO3, meanSO2, meanNO2, meanPM10)$

1.4.2 Observe the impact of cleaners on air quality

The algorithm aims to evaluate the effectiveness of air cleaners by analyzing air quality data from sensors located around the cleaner's operational area. It focuses on comparing air quality data from before and during the cleaner's operation to ascertain any improvements.

Algorithm 6 Evaluate the effectiveness of an air cleaner

```
Require: selectedCleaner
Ensure: boolean (true if the cleaner is effective, false otherwise)
closestSensors[3]                                   ▷ Table to store the closest sensors
i ← 0
check ← 0
result ← false
while i < 3 do
    sensor ← closestSensor(selectedCleaner, closestSensors)
    if sensor ∉ closestSensors then
        closestSensors[i] ← sensor
        i ← i + 1
    end if
end while
for i ← 0 to 2 do
    periodAnte ← definePeriodBefore(selectedCleaner)
    periodPost ← definePeriodAfter(selectedCleaner)
    meanAnte ← calculateMeans(closestSensors[i], periodAnte)
    meanPost ← calculateMeans(closestSensors[i], periodPost)
    if meanPost < meanAnte then
        check ← check + 1
    end if
end for
if check == 3 then
    result ← true
end if
return result
```

1.4.3 Classify Individuals as Unreliable

The algorithm determines the reliability of data provided by users by comparing their sensor measurements with those of the three geographically closest sensors on the same date. If a user's sensor data consistently deviates from the norm by more than 20% for more than 10 measurements and the nearest sensor is within 5 km, the user is marked as unreliable and banished.

Algorithm 7 Classify Sensor Data Reliability

```
Require: UserId, SensorId                                          ▷ From users.csv
Ensure: Updated reliability status in measurements.csv and users.csv
user_sensor_data ← getUserSensorData(UserId)
for each data_point in user_sensor_data do
    closest_distance ← findMostClosestSensorDistance(data_point.SensorId)
    if closest_distance > 5 then
        continue
    end if
    deviations ← 0
    closest_sensors ← findClosestSensors(data_point.SensorId, 3)
    for each date_measure in data_point.measures do
        average ← getAverageMeasurements(closest_sensors, date_measure.date)
        if |date_measure.value − average| /average > 0.20 then
            deviations ← deviations + 1
        end if
    end for
    if deviations > 10 then
        updateUserStatus(UserId, "BANNED")
        updateDataValidity(data_point.SensorId, False)
    end if
end for
```

1.5 Unit testing

1.5.1 checkSensor

- If selectedSensor exists and ineffective, checkSensor(selectedSensor) : True
- If selectedSensor exists and effective, checkSensor(selectedSensor) : False
- If selectedSensor doesn't exist, checkSensor(selectedSensor) : False

1.5.2 distance

- If two sensors exist, Distance(sensor1,sensor2) : dist
- If sensor3 doesn't exist, Distance(sensor1,sensor3) : 0

1.5.3 calculateMeans

- If excludedSensor=[] ,closestSensor(sensor1,excludedSensor) : closestSensor
- If excludedSensor=[sensor2] with sensor2 the closest with sensor1 , closestSensor(sensor1,excludedSensor) : (second)closestSensor
- If excludedSensor = [allSensors], closestSensor(sensor1,excludedSensor) : 0

1.5.4 calculateWeightedMeans

- If selectedSensor doesn't exist, calculateWeightedMeans(selectedSensor, nearbySensors ,5) : 0
- If nearbySensors is empty, calculateWeightedMeans(selectedSensor, nearbySensors ,5) : 0
- If selectedSensor exist and nearbySensors is not empty, calculateWeightedMeans(selectedSensor, nearbySensors , 5.5) : 0
- If selectedSensor exist and nearbySensors is not empty but there are not datas on the period, calculateWeightedMeans(selectedSensor, nearbySensors , 5) : 0
- If selectedSensor exist and nearbySensors is not empty with datas on the period, calculateWeightedMeans(selectedSensor, nearbySensors , 5) : weightedMean03, weightedMeanSO2, weightedMeanNO2, weightedMeanPM10

1.5.5 checkCleaner

- If selectedCleaner exists and not faulty, effectiveCleaner(selectedCleaner): True
- If selectedCleaner exists and faulty, effectiveCleaner(selectedCleaner): False
- If selectedCleaner does not exist, effectiveCleaner(selectedCleaner): False

1.5.6 checkPrivateSensor

- Reliable User: classifyUnreliable(UserId) should return no changes if the user's data is within tolerance.
- Unreliable User: For a user with consistently deviant sensor data, the function should mark their status as BANNED.
- Mixed Reliability: A user with both reliable and unreliable sensors should still be banned if the unreliable data meets the criteria.
- Insufficient Proximity: Users without any nearby sensors (less than 5 km away) are marked reliable, but a warning is displayed.
- Date Mismatch: If no matching date data exists, display a warning about the inability to perform verification.