Génie logiciel et modélisation UML

PRÉSENTATION ET DÉMO

SOMMAIRE

- SPÉCIFICATIONS DES EXIGENCES
- CONCEPTION
- DÉMONSTRATION

1.

SPÉCIFICATION DES EXIGENCES

System AirWatcher performance de l'algorithme <<Extend>> Analyser capteurs <<include>> Marquer la fiabilité de l'utilisateur Ajouter un point <<include>> comparer un au particulier capteur aux autres <<[nclude>> Gouvernement <<Extend>> /\ Signaler capteur Mesurer défaillant l'algorithme <<include>> Mesurer la zone l'amélioration de la d'amélioration qualité de l'air Extend>> Mesurer performance de l'algorithme Ajouter un point <<include>> aux particuliers Fabriquant User Calculer Moyenne AQI dans un rayon <<include>> performance de <=Extend>> Ajouter un point l'algorithme aux particuliers période produire une valeur de qualité d'air Se déconnecter <sinclude>> <<Extend>> Mesurer Ajouter un point performance de Se connecter au particulier l'algorithme

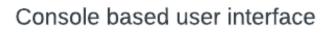
Diagramme de cas d'utilisation

2.

CONCEPTION

Architecture

 L'architecture "Layered architecture pattern" sera utilisée ici, elle offre une structure claire et une organisation logique qui aide à gérer la complexité des systèmes logiciels.



Application management with Business Objects and Application services : AQI computing, Sensors comparision and analyse etc...

Application functionalities with csv file interaction services

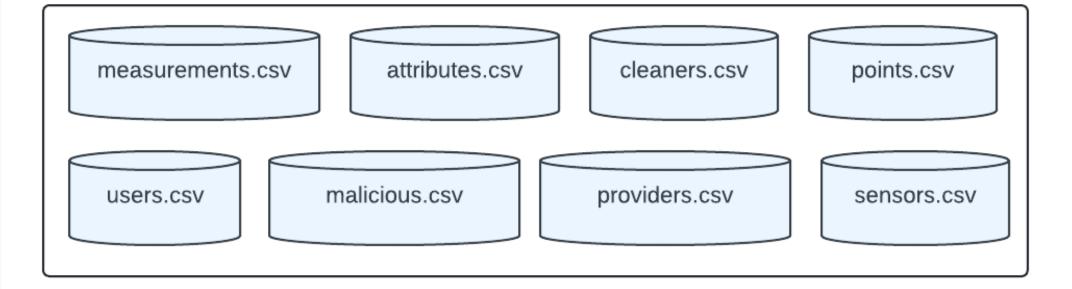


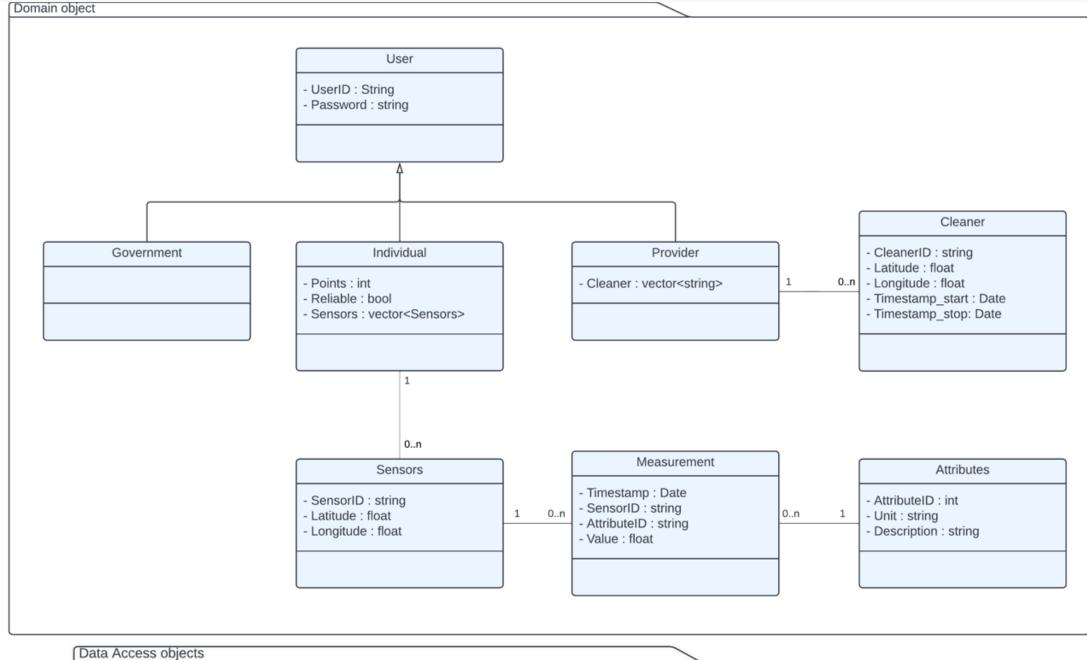
Diagramme de classe

Interface + afficherResultatAnalyse(): void + afficherRanking(): void + afficherAQIMoment (): void + afficherAQIPeriode (): void + afficherAmélioration (): void + afficherZoneEffective (): void + afficherAQIPoint (): void

Service object

Services

- + analyzeSensor(string sensorID) : bool
- + compareSensor(string sensorID, Date debut, Date fin) : int
- + computeAQIMoment (float latitude, float longitude, float radius, Date moment) : int
- + computeAQIPeriod (float latitude, float longitude, float radius, Date start, Date end) : float
- + measureImprovement (Cleaner cleaner, Date start, Date fin): int
- + measureAreaEffect (string cleanerID) : float
- + computeAQIPoint (float latitude, float longitude, Date moment) : int



SensorDAO IndividualDAO sensors : vector<Sensors> + findByID(string sensorID) : Sensors + markMalicious(string userID): + allSensors() : vector<Sensors> + sensorsCircle (float latitude, float longitude, + addPoint(string userID) : void float radius) : vector<Sensors> + findUserBySensor(string + rankedSensors (float latitude, float longitude) : sensorID): string vector<Sensors> CleanerDAO AttributeDAO cleaners : vector<Cleaner> + loadCleanersFromCSV(string filepath) : void + allAttributes() : Vector<Attributes> + allCleaners() : vector<Cleaner> + findByID (String CleanerID): Cleaner

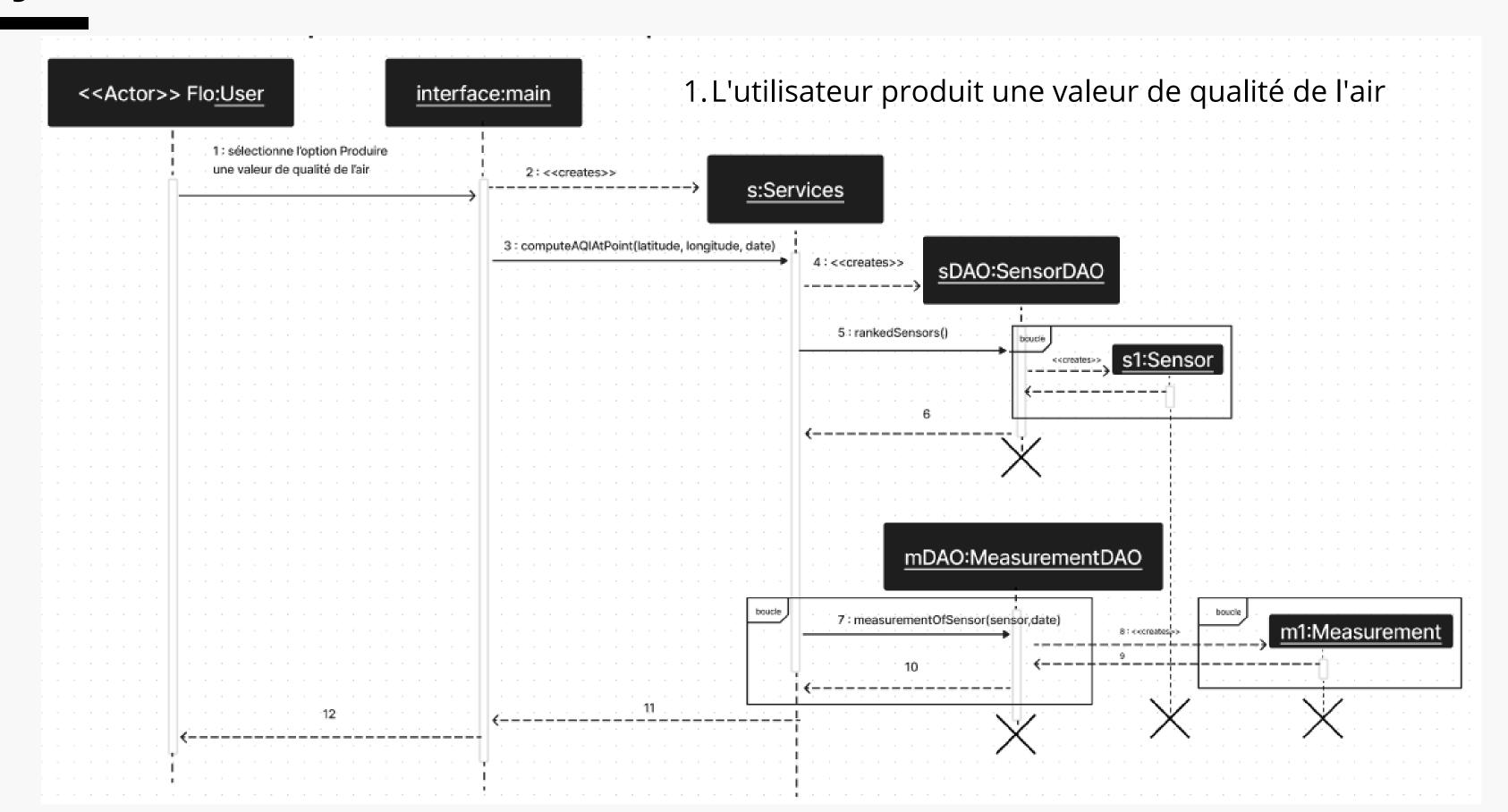
MeasurementsDAO

- measurements : vector<Measurement>
- measurementsFilePath: string
- sensorsFilePath : string
- isWithinPeriod(Date date, Date start, Date end) :
- + loadMeasurementsFromCSV(string filePath) : void
- + isWithinRadius(float, float, float, float, int) : bool
- + measurementsInCircleAtMoment(float, float, int,

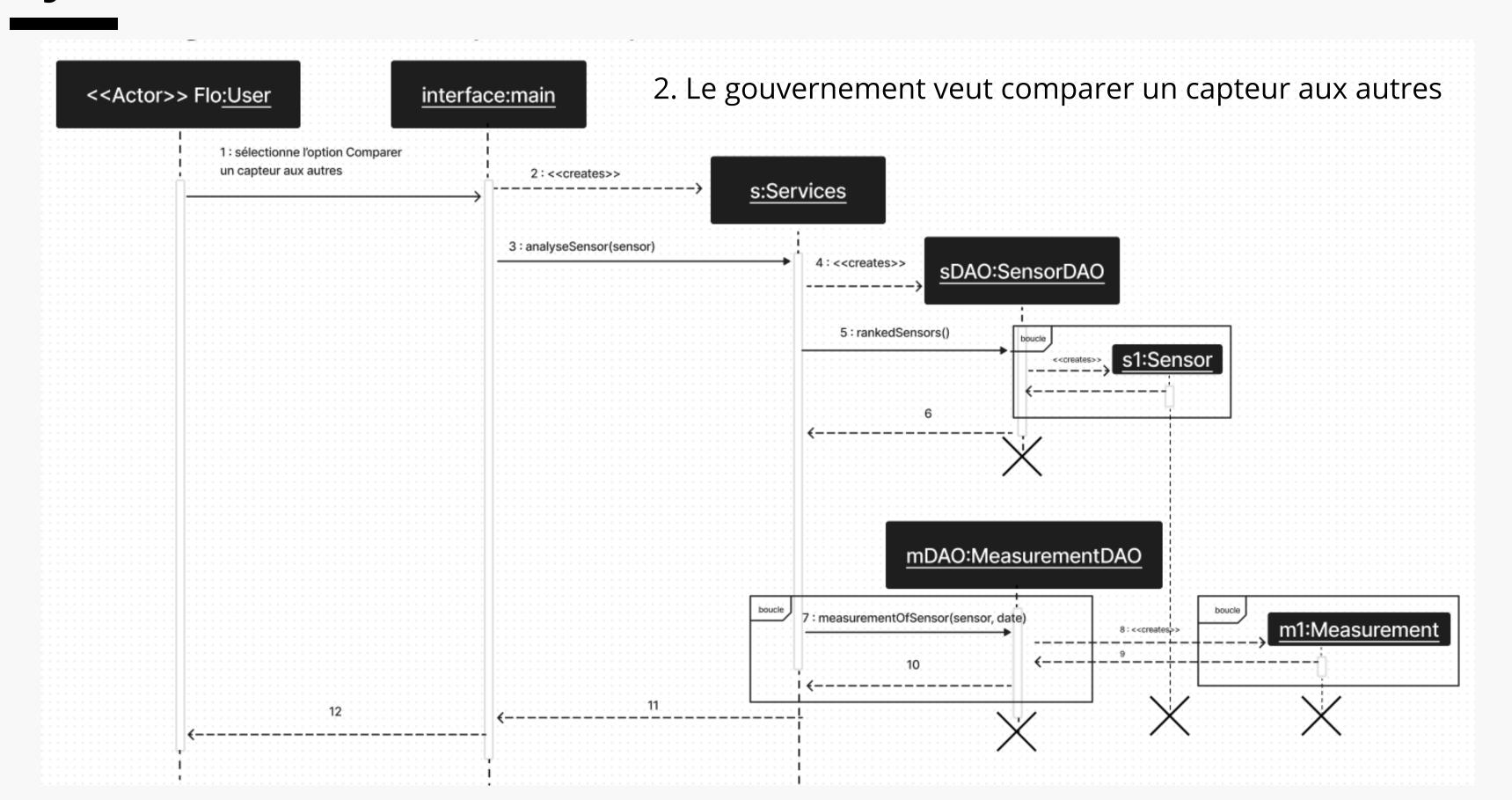
Date): vector<Measurement>

- + measurementsOfSensorInPeriod(string sensorID,
- Date start, Date end) : vector<Measurement>
- + AQIfromMeasurement(vector<Measurement>): int + MeasureImprovement(Sensors sensor, Date start,
- Date stop): int

Diagrammes de séquence des trois scénarios majeurs



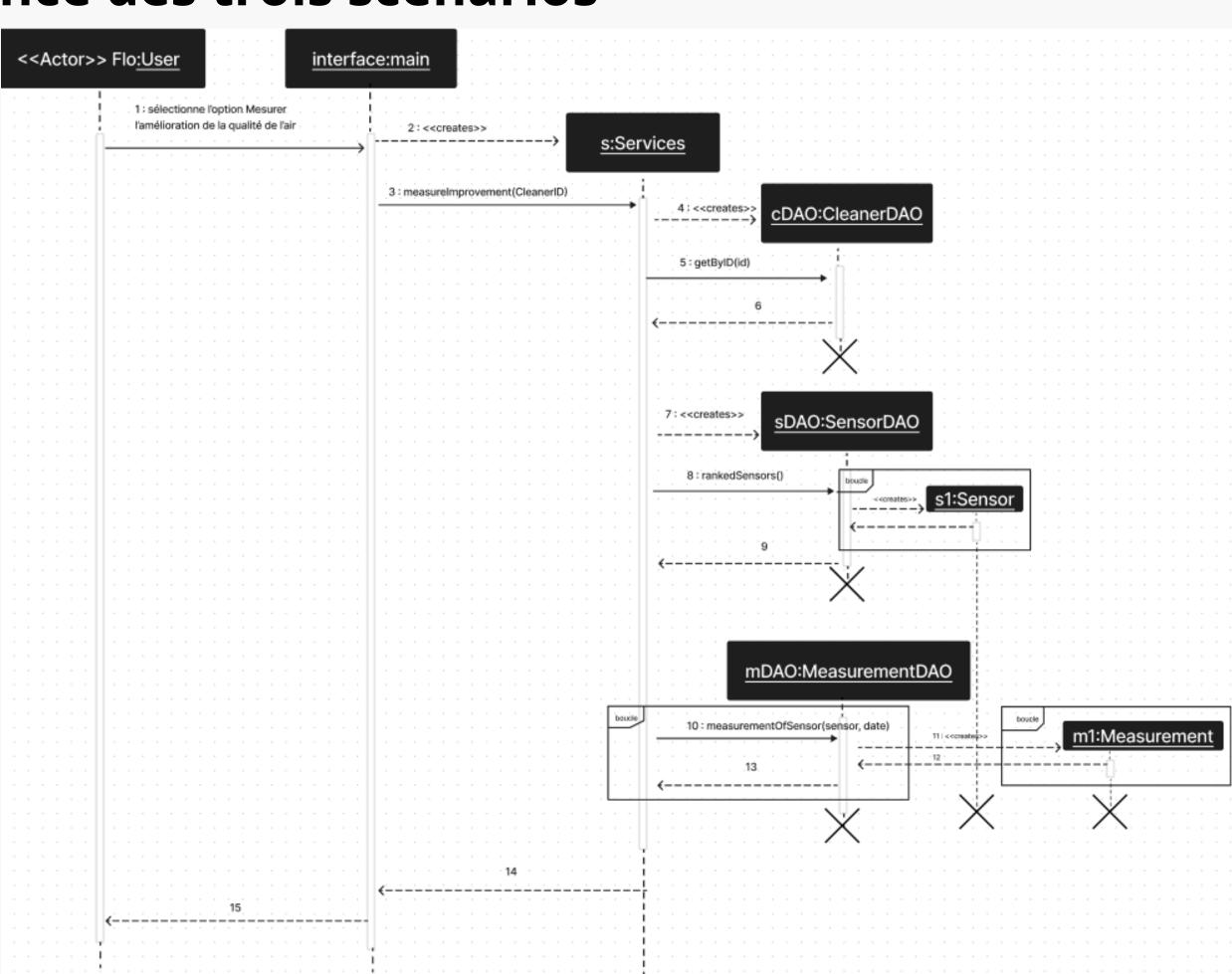
Diagrammes de séquence des trois scénarios majeurs



Diagrammes de séquence des trois scénarios

majeurs

3. Le fabricant veut mesurer l'amélioration de la qualité de l'air



4.

DÉMONSTRATION

MERCI!