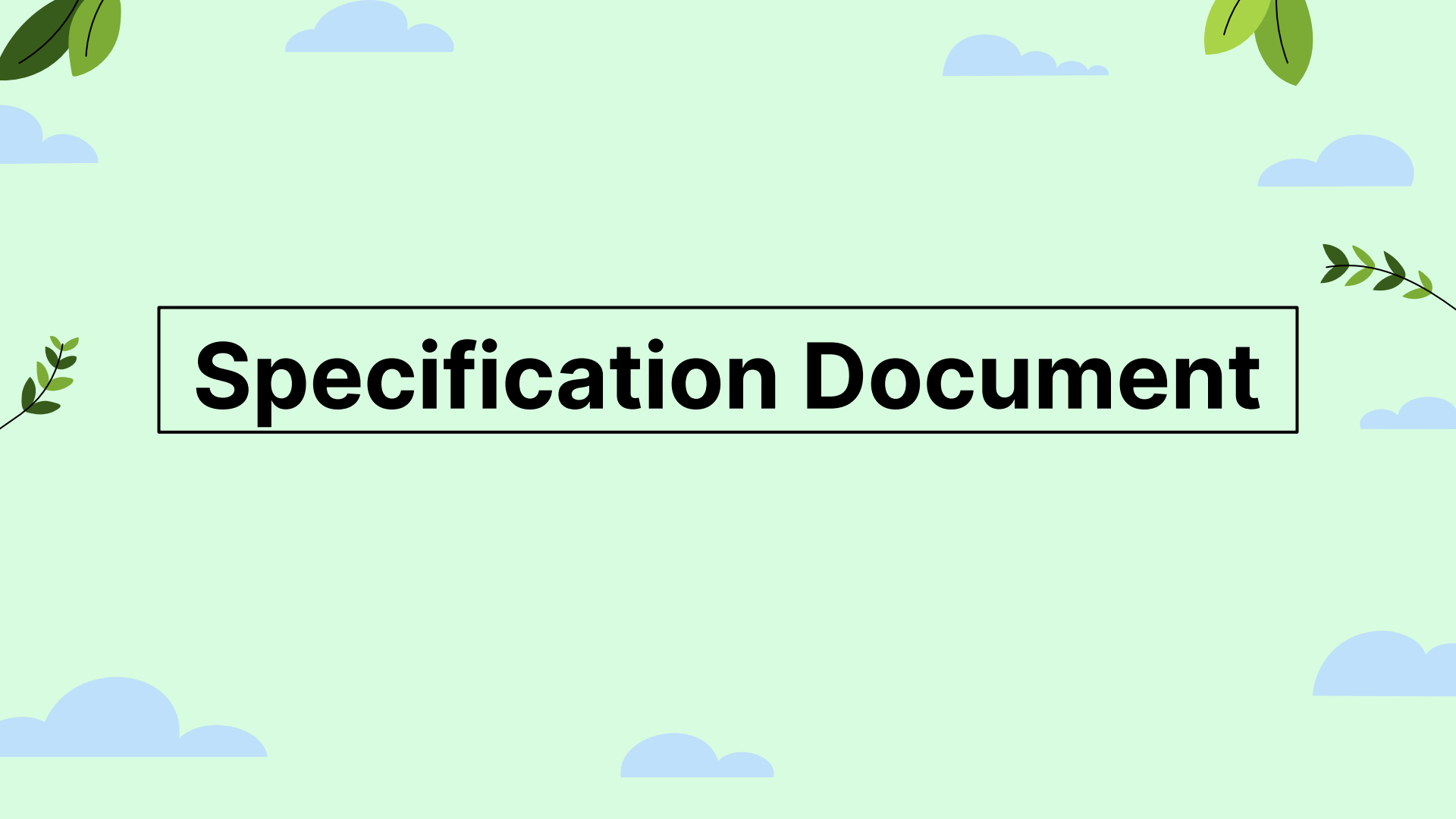
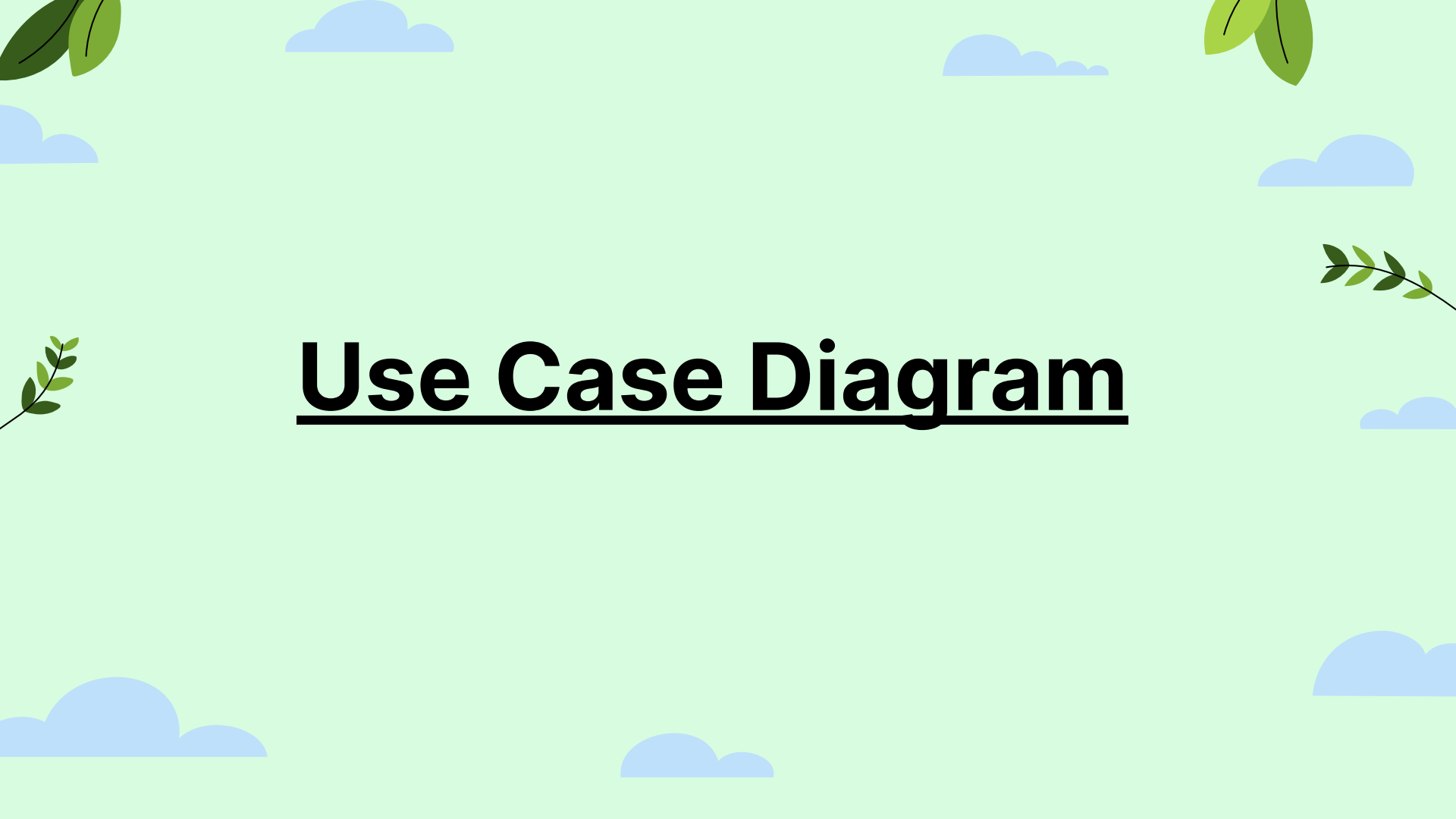




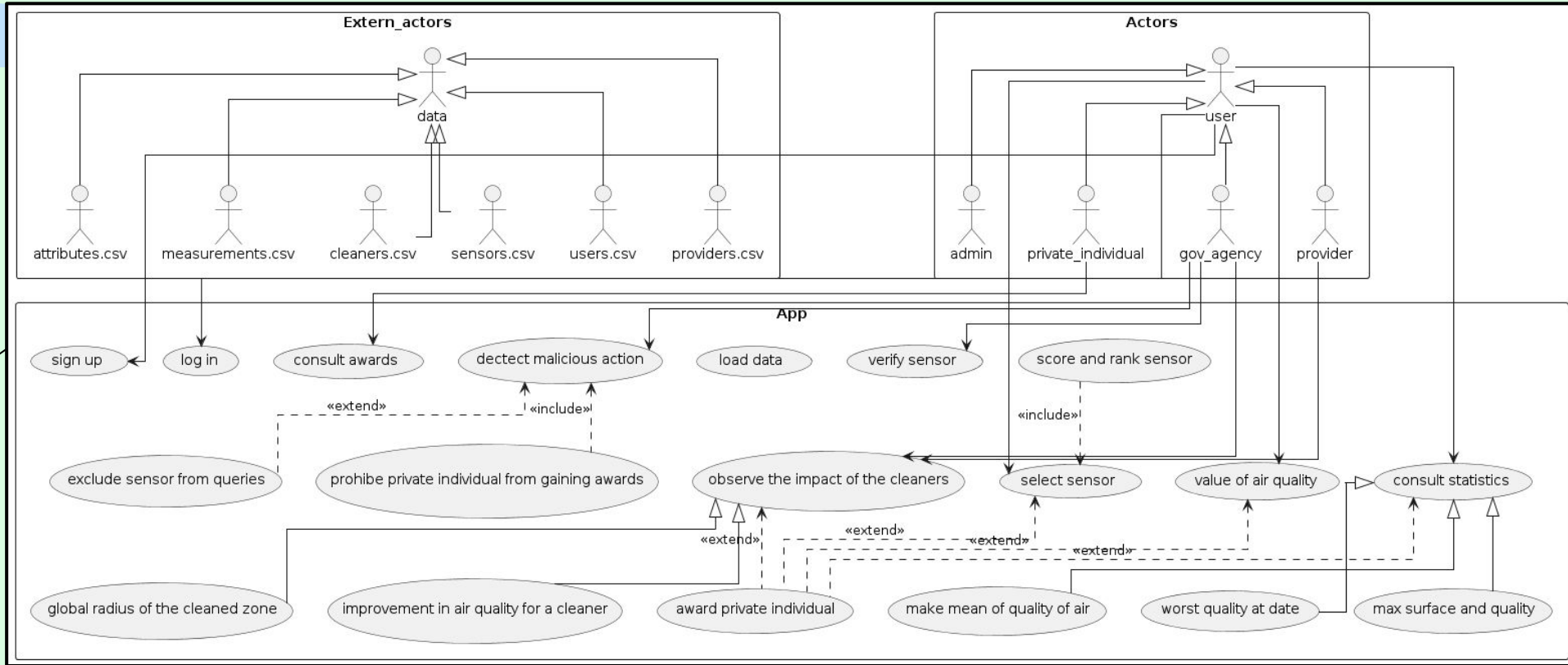
AIRWATCHER

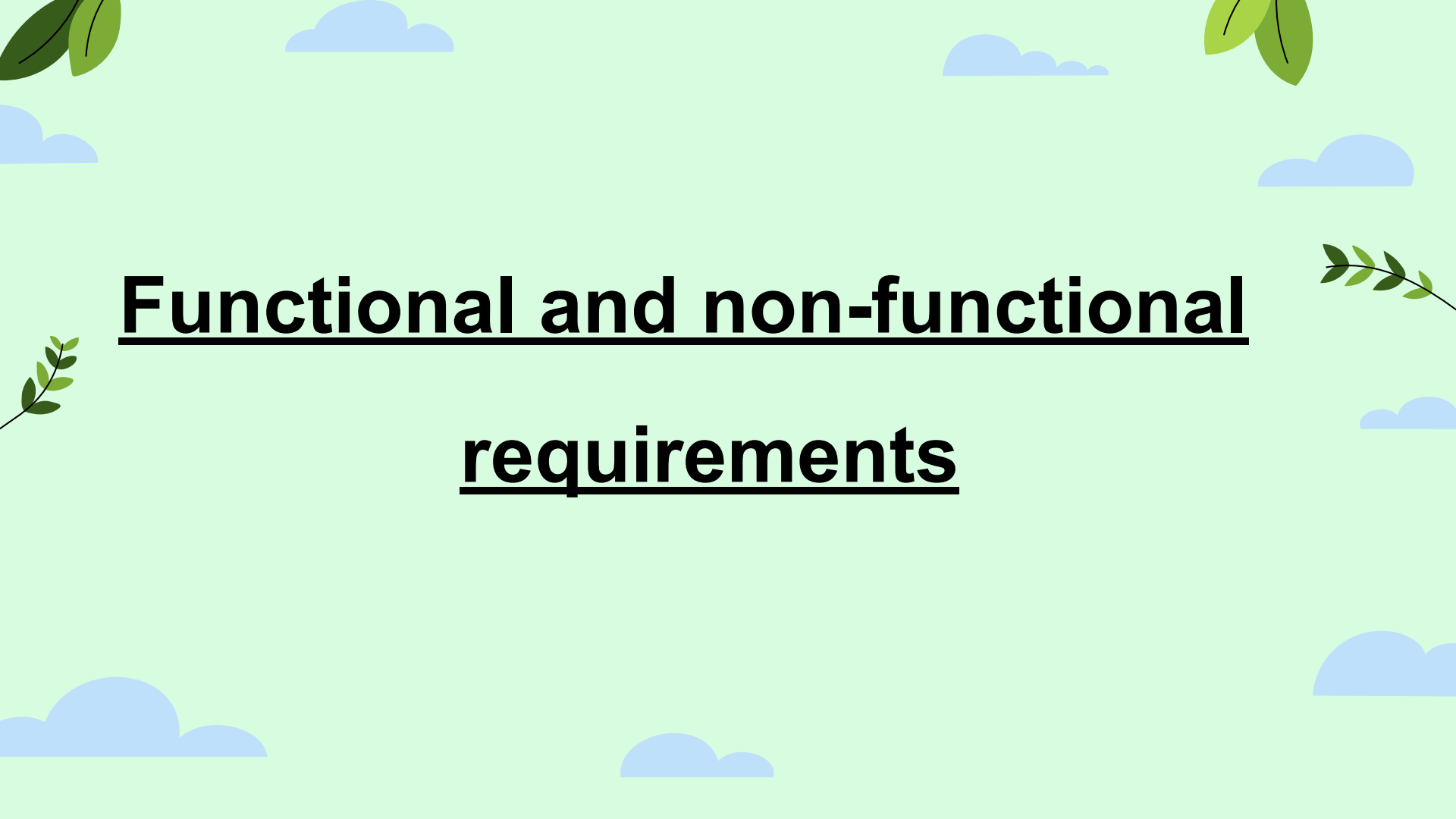


Specification Document




Use Case Diagram





Functional and non-functional **requirements**



Analysis of Security Risks

Atout	Vulnérabilité	Attaque	Risque (niveau impact)	Mesures de sécurité
Logiciel AirWatcher	<ul style="list-style-type: none"> - Des mots de passe faibles sont permis, par exemple, les mots de passe comme 123456, <u>motdepasse</u>, etc. 	<ul style="list-style-type: none"> - Deviner le mot de passe 	<ul style="list-style-type: none"> - Accès non autorisé (Élevé) - Violation de la confidentialité (Élevé) 	<ul style="list-style-type: none"> - Les mots de passe sont vérifiés et les mots de passe faibles ne sont pas acceptés - Un CAPTCHA est utilisé pour vérifier que les réponses sont données par un humain - Transmission et stockage de données avec chiffrement
Toutes les données (capteurs, purificateurs, utilisateurs,...)	<ul style="list-style-type: none"> - Transmission et stockage de données en local, sans chiffrement - Serveur central local non sécurisé qui contient toutes les données 	<ul style="list-style-type: none"> - Création de fausses données - Modification/suppression de données existantes - Interception des communications - Récupération des fichiers de données brutes - Utilisation de code malveillant 	<ul style="list-style-type: none"> - Accès non autorisé aux données (Élevé) - Manipulation des données donc Intégrité (Élevé) - Vol physique des capteurs ou purificateurs grâce aux coordonnées géographiques (Élevé) 	<ul style="list-style-type: none"> - Validation et vérification des données à travers un service d'analyse de capteurs - Permettre l'accès aux données seulement à travers des fonctionnalités fournies par l'application - Données brutes (et transmissions) chiffrées et stockées sur un serveur distant - Faire des sauvegardes régulières des données - Protection contre les logiciels malveillants
Disponibilité de l'application	<ul style="list-style-type: none"> - Point de défaillance unique car tout est disponible sur le même serveur local 	<ul style="list-style-type: none"> - Attaques DoS ou DDoS 	<ul style="list-style-type: none"> - Indisponibilité de l'application (Élevé) 	<ul style="list-style-type: none"> - Mécanismes de défense contre les attaques DoS et DDoS - Distribuer le système pour ne pas tout avoir sur le même serveur

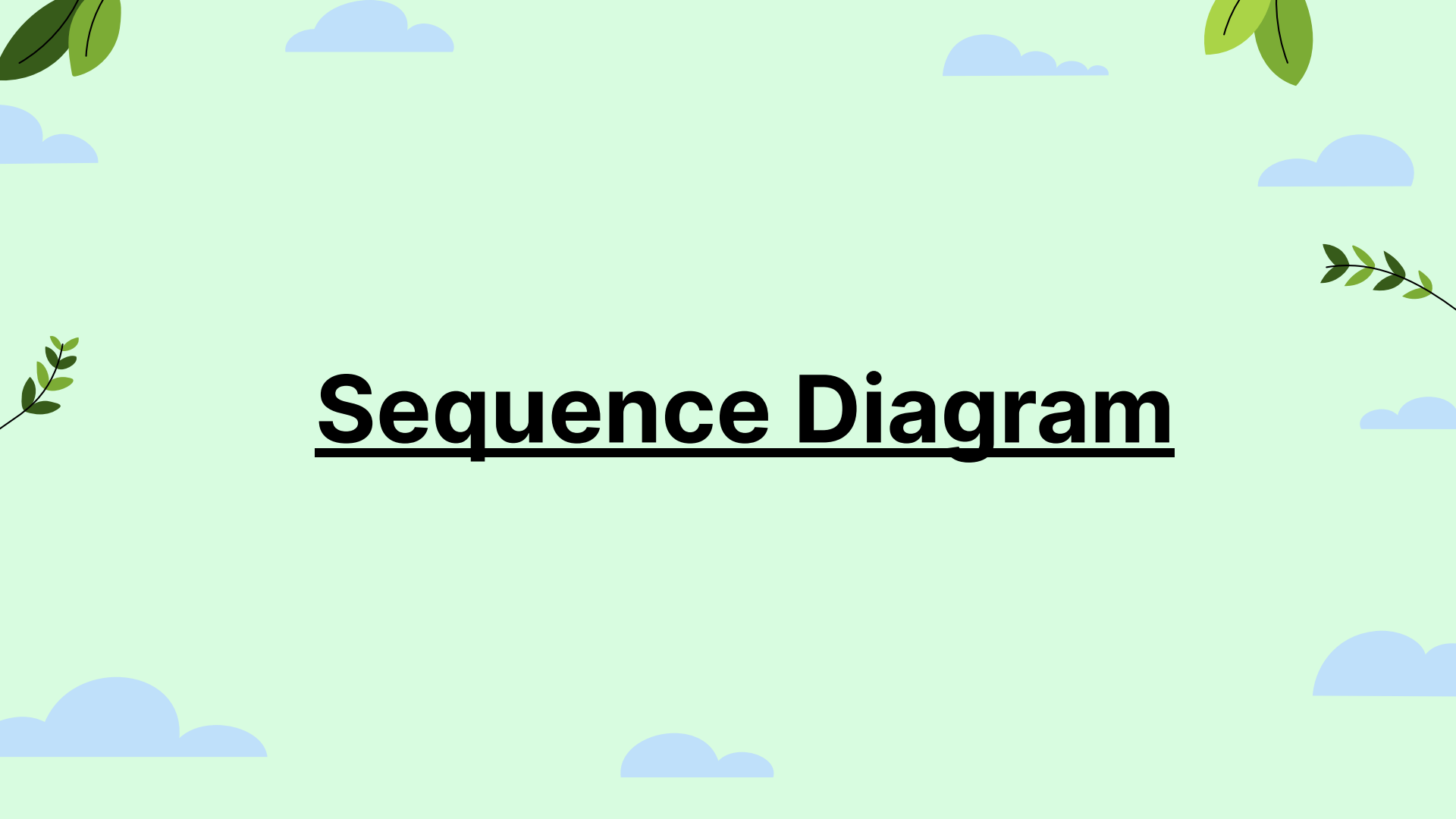


Validation Tests



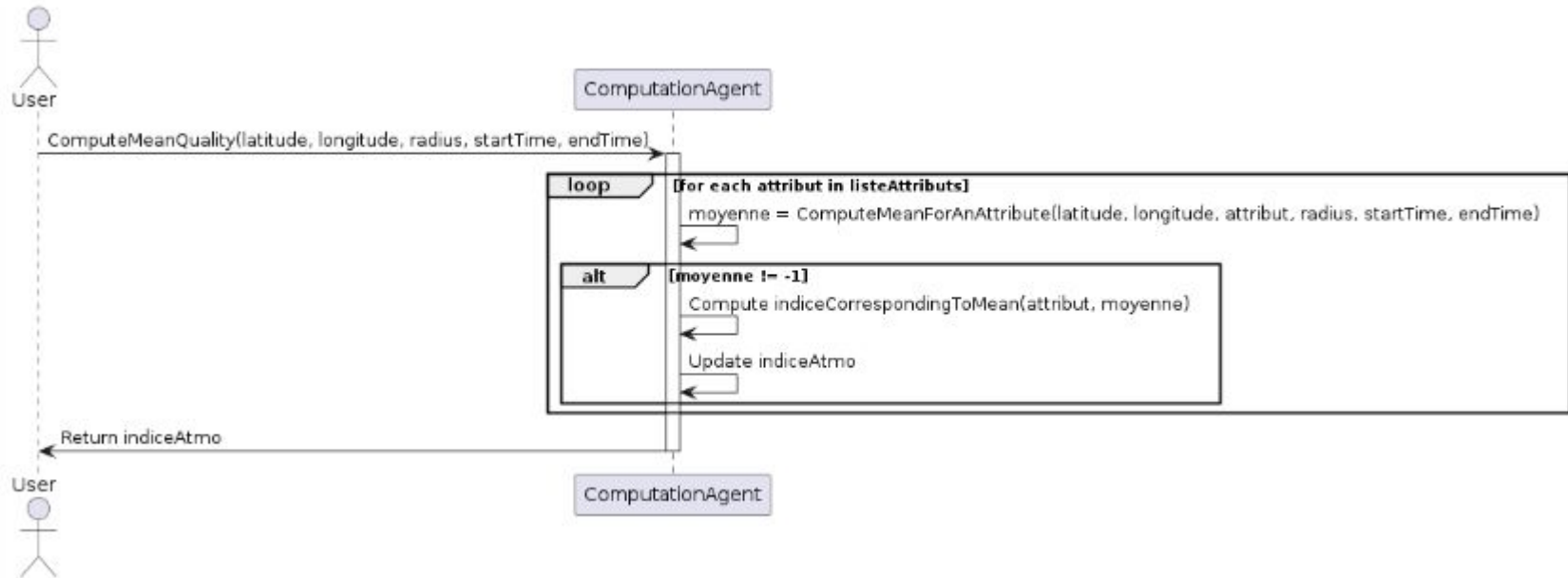
Design Document

Class Diagram

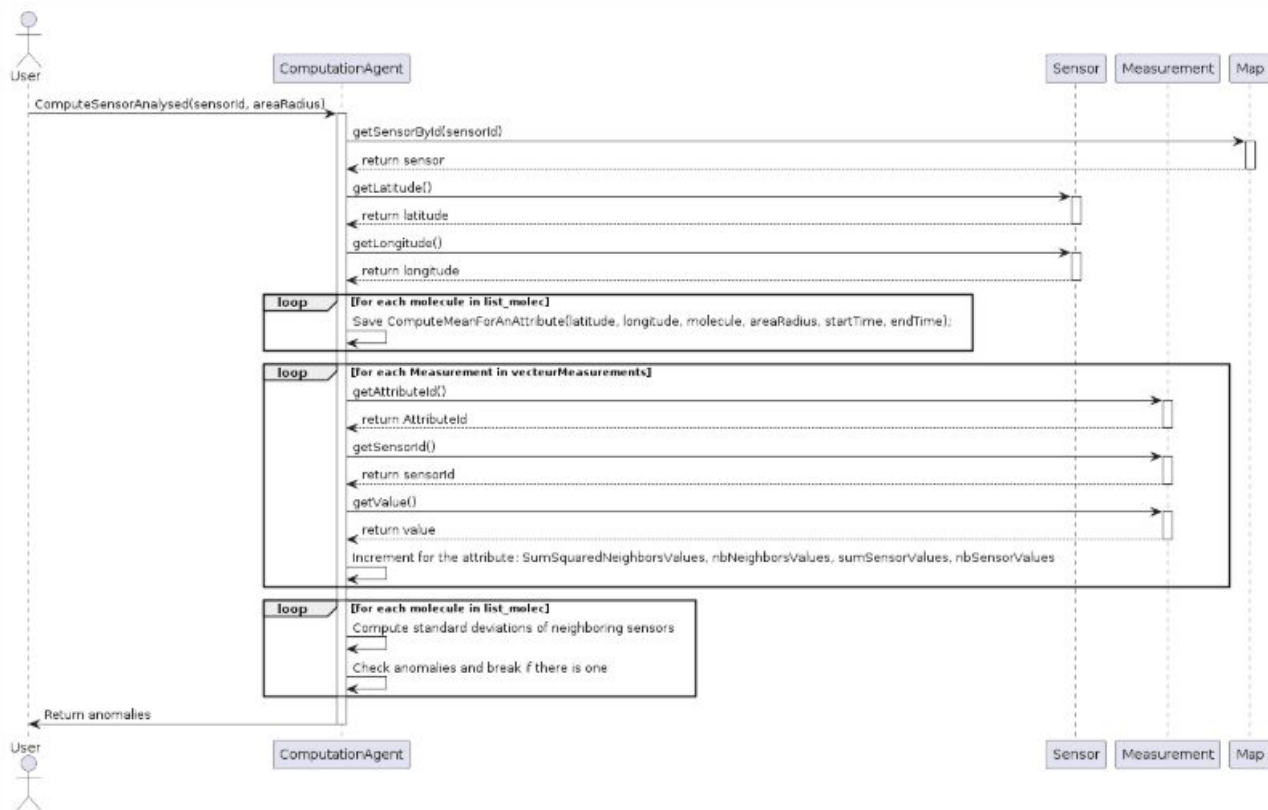


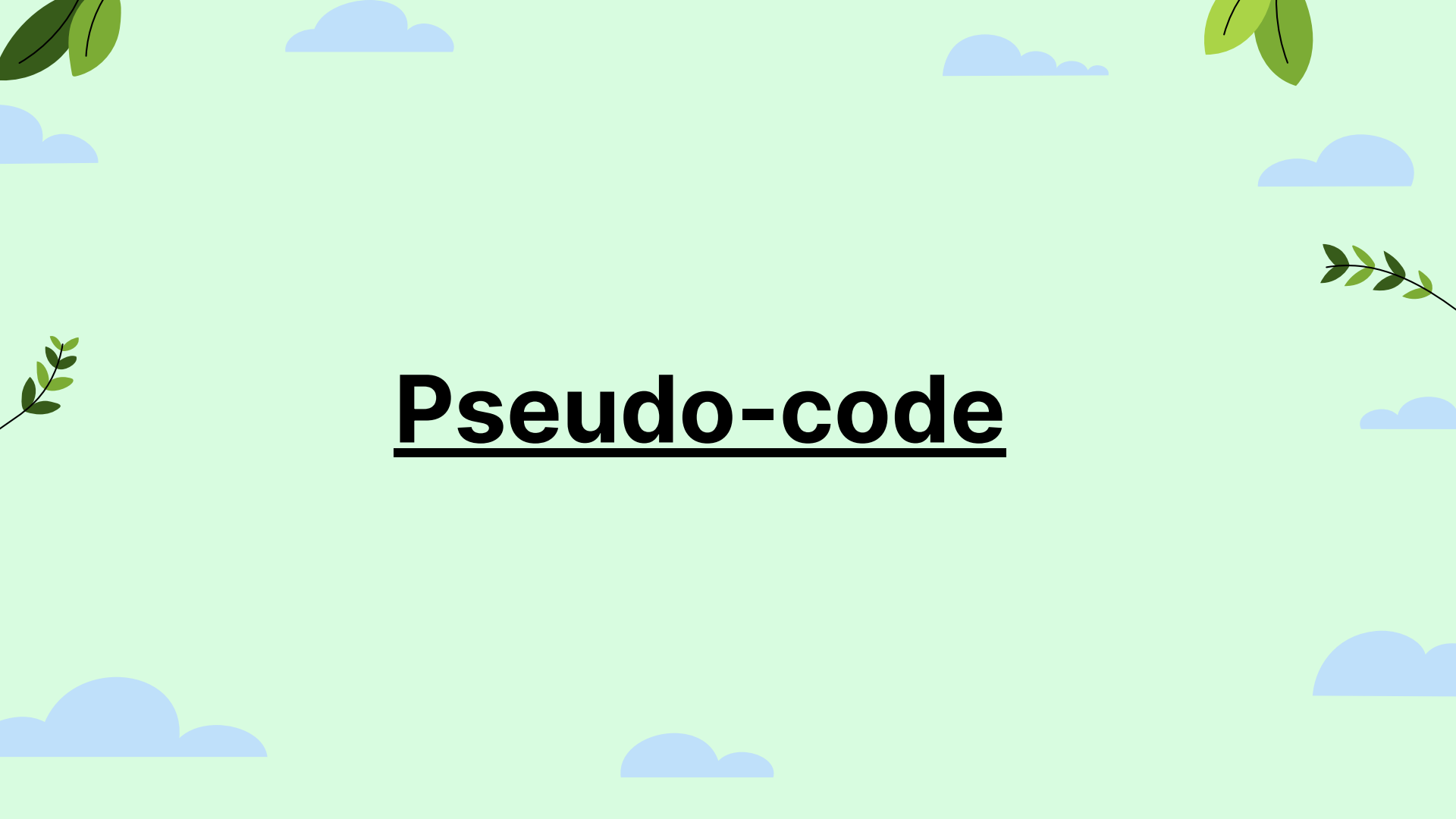
Sequence Diagram

Calcul de la moyenne de la qualité de l'air



Analyse d'un capteur





Pseudo-code

```

ComputeMeanQuality(latitude, longitude, radius, startTime, endTime):
    list_molec = ["O3", "NO2", "SO2", "PM10"]
    listeMoyennes = []
    indiceATMO = -1
    Pour chaque molecule dans list_molec faire :
        moyenne = ComputeMeanForAnAttribute(latitude, longitude, molecule, radius, startTime, endTime)

    Si (moyenne != 0) alors :
        ajouter(moyenne, listeMoyennes)
        indice = indiceCorrespondingToMean(molecule, moyenne)
    Si (indice > indiceATMO) alors : indiceATMO = indice
    Fin Si
    Fin Pour

    Si (indiceATMO == -1) alors :
        Traiter Erreur...
        Afficher("Calcul Impossible de l'indiceATMO")
    Fin Si

    Fin Pour

    retourner indiceATMO

```

```

ComputeMeanForAnAttribute(latitude, longitude, molecule, radius, startTime, endTime):
    moyenne = -1
    somme = 0
    nbValues = 0
    Pour chaque mesure dans measurements.csv faire :
        dateMesure = mesure.date
        sensor = mesure.sensor
        Si (mesure.attribute == molecule) alors :
            distance = calculateDistance(latitude, longitude, sensor.latitude, sensor.longitude)

            Si ((startTime < dateMesure et dateMesure < endTime) ou (startTime == null et endTime == null)) et
                (distance <= radius) alors :
                    somme += mesure.value
                    nbValues++
            Fin Si
        Fin Si
    Fin Pour

    moyenne = somme/nbValues
    retourner moyenne

```

ComputeSensorAnalysed(sensorID, areaRadius):

```

anomalies = false

sensor = rechercherSensorParID(sensorID)

dicoMeanCapteur = {"O3": 0, "NO2": 0, "SO2": 0, "PM10": 0}

Cr  er les dictionnaires dicoMeanAll, dicoSumOfSquaresAll, dicoSdAll, dicoNbValuesAll et dicoNbValuesCapteur

list_molec = ["O3", "NO2", "SO2", "PM10"]

// Initialiser les dictionnaires
Pour chaque molecule dans list_molec faire :
    dicoSumOfSquaresAll[molecule] = dicoNbValuesAll[molecule] = dicoNbValuesCapteur[molecule] = dicoSdAll[molecule] = dicoMeanAll[molecule] = 0
Fin Pour

// Calculer les moyennes globales pour chaque molecule
Pour chaque molecule dans list_molec faire :
    dicoMeanAll[molecule] = ComputeMeanForAnAttribute(sensor.latitude, sensor.longitude, molecule, areaRadius, 0, 0);
Fin Pour

// Lire les mesures et calculer les sommes et les carr  s
Pour chaque mesure dans measurements.csv faire :
    molecule = mesure.attribute

    Si mesure.sensorID == sensorID alors :
        dicoMeanCapteur[molecule] += mesure.value
        dicoNbValuesCapteur[molecule] += 1
    Fin Si

    dicoSumOfSquaresAll[molecule] += (mesure.value - dicoMeanAll[molecule]) * (mesure.value - dicoMeanAll[molecule])
    dicoNbValueAll[molecule] += 1

Fin Pour

Si le capteur n'a pas de mesure pour toutes les mol  cules alors :
    Afficher("Analyse impossible car le capteur n'a aucune mesure")
    retourner null
Fin Si

// Calculer les   carts-types et v  rifier les anomalies
Pour chaque molecule dans list_molec faire :
    Si (dicoNbValuesCapteur[molecule] == 0) faire : passer    la mol  cule suivante
    Fin Si

    Si (dicoNbValueAll[molecule] != dicoNbValuesCapteur[molecule]) faire :
        Afficher("Analyse impossible car il n'y a pas de mesure de capteurs voisins pour au moins une mol  cule")
        retourner null
    Fin Si


    dicoMeanCapteur[molecule] = dicoMeanCapteur[molecule] / dicoNbValuesCapteur[molecule];
    dicoSdAll[molecule] = sqrt(dicoSumOfSquaresAll[molecule] / dicoNbValueAll[molecule])

    Si ((dicoMeanCapteur[molecule] > dicoMeanAll[molecule] + dicoSdAll[molecule]) ou
        (dicoMeanCapteur[molecule] < dicoMeanAll[molecule] - dicoSdAll[molecule])) alors :
        anomalies = True
    Fin Si

Fin Pour

retourner anomalies

```



Unit-Tests