Auteurs : BIAUD, GIRAUDON, MAILLARD, WARIN

AirWatcher

I. Spécifications

I.1. Introduction

Nous avons développé une application qui permet de traiter les données de capteur de qualité de l'air. A partir de ces données, l'application pourra fournir des statistiques sur une zone géographique, détecter des capteurs défectueux…

Cette application est destinée au gouvernement, mais tout utilisateur peut accéder à l'application ainsi qu'à des statistiques de qualité de l'air.

L'objectif de ce TP est de réaliser toutes les étapes de conception précédant le développement de l'application. Puis de développer l'application en vérifiant son bon fonctionnement grâce à une liste de tests.

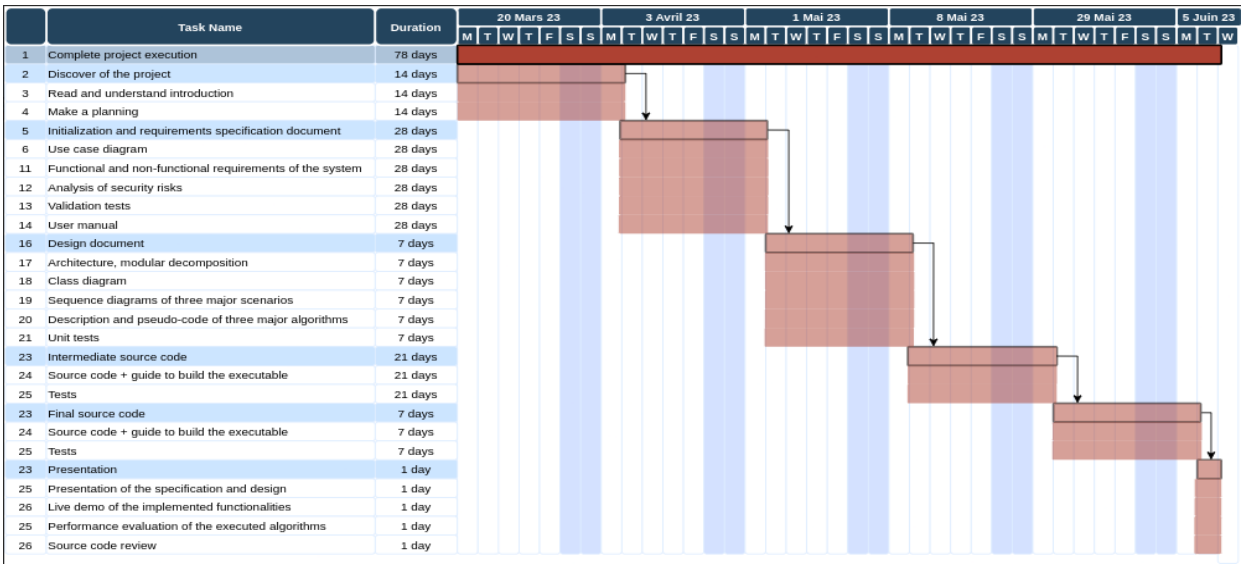
I.2. Organisation

Afin de gérer ce projet de la meilleure manière, nous nous sommes attribué les rôles suivants :

- Hugo Warin sera notre chef de projet, il s'occupera de distribuer les tâches.
- Alexandre Biaud est rédacteur, il s'occupe de mettre en forme les rapports à rendre.
- Clément Giraudon est notre contrôleur qualité, il supervisera les tests.
- Swan Maillard est responsable technique, il s'occupe de mettre en place les outils techniques utiles.

Nous avons tous une part égale dans la rédaction des spécifications, la conception, le développement et les tests de l'application.

Nous avons aussi construit le diagramme de Grantt suivant :



	Task Name	Duration
1	Complete project execution	78 days
2	Discover of the project	14 days
3	Read and understand introduction	14 days
4	Make a planning	14 days
5	Initialization and requirements specification document	28 days
6	Use case diagram	28 days
11	Functional and non-functional requirements of the system	28 days
12	Analysis of security risks	28 days
13	Validation tests	28 days
14	User manual	28 days
16	Design document	7 days
17	Architecture, modular decomposition	7 days
18	Class diagram	7 days
19	Sequence diagrams of three major scenarios	7 days
20	Description and pseudo-code of three major algorithms	7 days
21	Unit tests	7 days
23	Intermediate source code	21 days
24	Source code + guide to build the executable	21 days
25	Tests	21 days
23	Final source code	7 days
24	Source code + guide to build the executable	7 days
25	Tests	7 days
23	Presentation	1 day
25	Presentation of the specification and design	1 day
26	Live demo of the implemented functionalities	1 day
25	Performance evaluation of the executed algorithms	1 day
26	Source code review	1 day

Figure 1 : Diagramme de GRANTT

I.3. Diagramme de cas d'utilisation

Nous avons commencé par effectuer le diagramme de cas d'utilisation suivant :

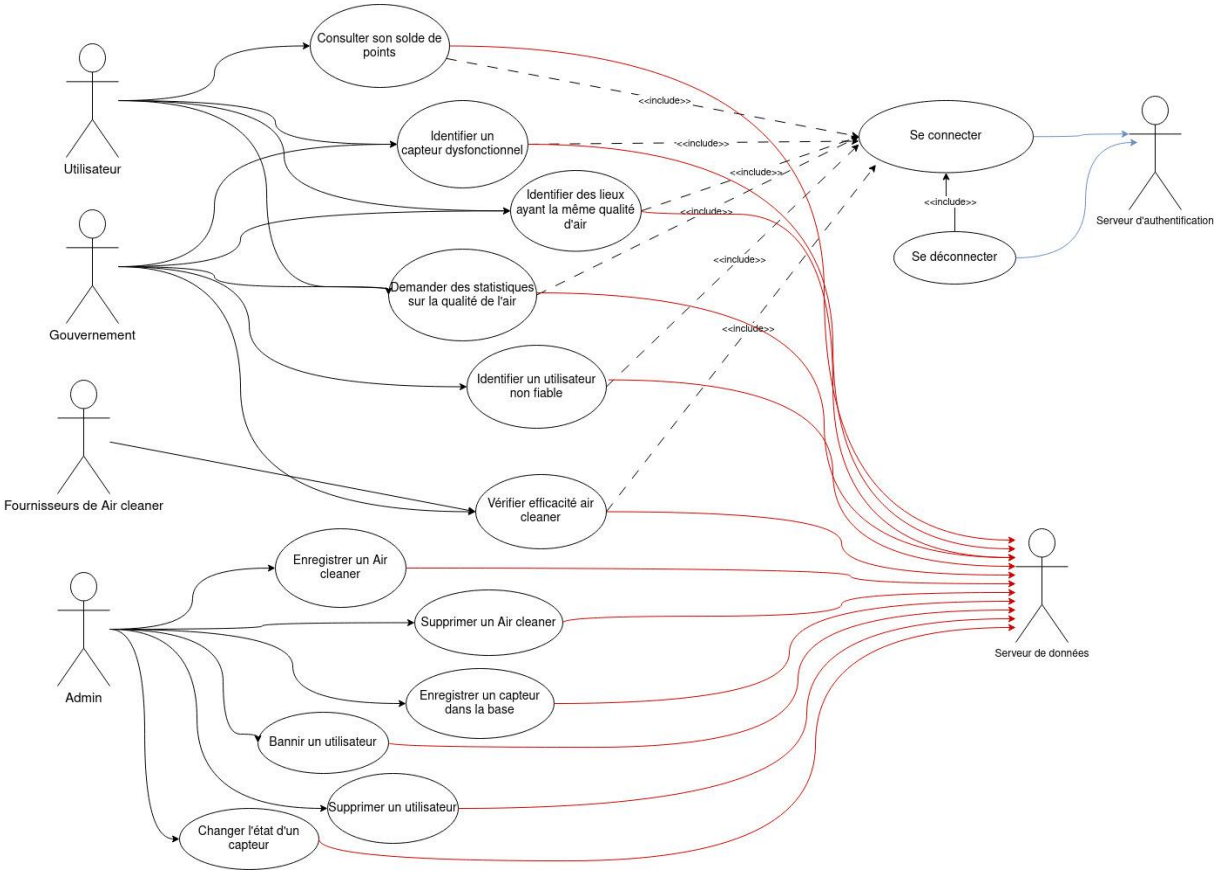


Figure 2: Diagramme de cas d'utilisation

I.4. Liste des besoins fonctionnels

Nous avons ensuite énuméré les fonctionnalités que doivent réaliser l'application et clarifier leur signification :

I.4.a. Authentifier un utilisateur :

Accès : Tous les utilisateurs

Description : Pour utiliser l'application, il faut se connecter. L'utilisateur pourra donc entrer un login et un mot de passe et accéder au reste des fonctionnalités s'il est connu dans la base de données.

I.4.b. Identifier les capteurs dysfonctionnels et alerter l'agence

Accès : Gouvernement uniquement

Description : le logiciel va étudier tous les capteurs présents dans la base de données et chercher des anomalies pour identifier un capteur défectueux. Nous utiliserons la méthode suivante :

Pour un capteur donné, on regarde les valeurs des 5 capteurs les plus proches (calcul de la moyenne, et de l'écart-type pour chacune des 4 catégories de polluants) et on vérifie que pour chaque catégorie, la valeur du capteur est comprise dans moyenne +/- écart-type.

Il faut faire attention à ne pas compter les capteurs dysfonctionnels parmi les capteurs environnants.

I.4.c. Générer des statistiques

Accès : Tous les utilisateurs

Description : Il est possible de générer des statistiques sur un capteur ou un groupe de capteur dans une zone circulaire. On peut étudier un moment précis ou une période. Voici les statistiques : Score de l'air (Très bon/Bon/Moyen/Médiocre/Mauvais/Très Mauvais), indice ATMO global (maximum), indices ATMO pour les 4 catégories (O3, SO2, NO2 et PM10) ainsi que leurs concentrations respectives.

Le logiciel va calculer la valeur moyenne de chaque polluant et son écart-type. Enfin, il calcule le score ATMO moyen. Le calcul du score ATMO est expliqué ici : https://fr.wikipedia.org/wiki/Indice_de_qualit%C3%A9_de_l%27air

I.4.d. Connaître la qualité de l'air d'un point particulier

Accès : Tous les utilisateurs

Description : L'utilisateur précise les coordonnées d'un point précis. Ensuite, le logiciel trouve les 3 capteurs les plus proches de ce point. Il fait alors une moyenne des densités des polluants à partir des trois données pondérées par la distance séparant le point du capteur et indique l'indice ATMO.

I.4.e. Trouver les capteurs dans des zones où la qualité de l'air est similaire

Accès : Gouvernement uniquement

Description : Le client souhaiterait savoir si deux capteurs différents ont les mêmes caractéristiques en termes de qualité d'air. Pour cela, l'utilisateur choisit un capteur et une plage de temps. On calcule la différence de score pour chaque catégorie (O3, SO3, NO2 et PM10) entre les 2 capteurs puis on fait la moyenne de ces différences. Plus cette moyenne est petite, plus la qualité de l'air est similaire entre les deux capteurs. On affiche les 10 capteurs les plus similaires dans l'ordre décroissant de similarité.

I.4.f. Vérifier qu'un capteur est fiable

Accès : gouvernement uniquement

Description : Des particuliers peuvent obtenir un capteur et collecter des données autour de chez eux. Il faut alors s'assurer que les données qu'ils envoient ne sont pas erronées. Pour un capteur donné, on regarde les valeurs des 5 capteurs les plus proches (calcul de la moyenne, et de l'écart-type pour chacune des 4 catégories de polluants) et on vérifie que pour chaque catégorie de notre capteur est compris dans moyenne +/- écart-type. Il faut faire attention à ne pas compter les capteurs dysfonctionnels parmi les capteurs environnants.

I.4.g. Donner des points aux utilisateurs

Description : Lorsqu'une demande de statistique ou d'information d'une zone géographique utilise un capteur d'un particulier, celui-ci gagne un point. Au bout d'un certain nombre de points, cet utilisateur gagnera un bon de réduction pour l'achat d'un purificateur d'air !

I.4.h. Vérifier l'efficacité des purificateurs d'air

Accès : Tous les utilisateurs

Description : Des purificateurs d'air sont installés à certains points. Le logiciel va comparer les données des capteurs alentour avant et après l'utilisation des purificateurs d'air jusqu'à ce que la différence soit inférieure à 25%. Il pourra alors afficher le taux d'efficacité en son centre et le rayon d'efficacité.

I.4.i. Gérer les utilisateurs

Accès : Le gouvernement

Description : Il s'agit de créer, supprimer ou modifier le statut des utilisateurs.

I.4.j. Gérer les capteurs

Accès : Le gouvernement

Description : Il s'agit de créer, supprimer des capteurs.

I.4.k. Calculer le temps de calcul des fonctionnalités

Accès : Tous les utilisateurs

Description : Quand l'utilisateur effectue une requête, il souhaiterait connaître le temps de calcul de cette dernière.

I.5. Liste des besoins non-fonctionnels

Nous avons ensuite pu définir les besoins non-fonctionnels suivant :

I.5.a. Il doit être impossible d'altérer les données

Description : Le gouvernement peut modifier des données concernant les capteurs et les utilisateurs, mais personnes ne peut modifier les données des capteurs.

I.5.b. L'application doit pouvoir accéder aux données des capteurs en temps réel

Description : L'application aura accès à une base locale qui sera une copie de la base de données. La copie est réalisée toutes les 15 minutes pour que les données soient pertinentes.

I.5.c. Le programme doit être efficace

Description : Les algorithmes utilisés seront les plus performants possibles (exemple : tri fusion pour un tri).

I.6. Analyse de la sécurité

Notre application possède des données confidentielles qui ne doivent pas être divulguées.

Elle peut donc être la cible de différentes attaques.

Nous pouvons être confrontés à différents attaquants :

- Des groupes pétroliers ou des industriels qui voudraient trafiquer les données des capteurs présents à proximité de leurs usines.
- Des autres gouvernements.
- Groupes de hackers divers.

Nous devons prendre en compte ces informations pour protéger notre application et plus précisément nos données qui sont ici nos atouts.

Il faut protéger notre base de données de capteurs, les points des utilisateurs, et les mesures en elles-mêmes.

Nous devons donc faire un point sur les différentes vulnérabilités de notre application :

- Les mots de passe faibles : Il faut qu'il y ait un politique de mot de passe assez strict, éviter les mots de passe présents sur des bases de données qui ont fuitées (ex : rockyou.txt / secLists).
- Protéger les entrées utilisateurs : Il faut absolument vérifier les informations saisies par les utilisateurs pour éviter des comportements inattendus de notre application, ou différents types d'injections.
- Transmission de données non chiffrées : Afin de conserver la confidentialité et l'intégrité des données, il faut chiffrer la transmission de données.
- Mots de passe stockés en clair : Les mots passe stockés en clair peut devenir une vulnérabilité très importante en cas d'intrusion dans le système. Ces attaques n'auront peut-être pas beaucoup d'impact sur notre application, mais peuvent avoir un plus grand impact sur les utilisateurs, notamment ceux qui utilisent le même mot de passe partout. Il faut donc chiffrer les mots de passe avec des algorithmes puissants (ex sha256 ou sha512crypt).

Les différentes failles listées ci-dessus peuvent compromettre notre application. Voici une liste non-exhaustive d'attaques possibles :

- SQL Injection.
- Interception de transmission de données (BURP).
- Accès à des zones mémoires non autorisées.
- Code Injection.

Il faut relativiser, les probabilités d'attaques restent assez faibles. Pour autant, puisque notre application implique des membres du gouvernement, nous devons être particulièrement attentifs.

La liste des différentes menaces est la suivante :

- Introduction dans les serveurs du gouvernement
- Suppression des données
- Altération des données
- Mots de passe compromis
- Ransomware

Les risques les plus importants ici sont la perte des données ou la fuite des mots de passe.

Ces risques peuvent avoir un impact désastreux sur le gouvernement, entraînant des systèmes compromis, une perte de crédibilité ou encore la divulgation de fausses informations.

Pour éviter cela, voici une liste de contre-mesures applicables :

- Politique de mots de passe strict.

- Chiffrement le transfert de données.
- Chiffrement les mots de passe.
- Nettoyage les données saisies par un utilisateur.

I.7. Manuel d'utilisation et interface

Voici une description succincte de l'interface de notre application. À la suite de cette description, vous trouverez une maquette détaillée de notre interface console avec les différents menus et un exemple de chaque fonctionnalité.

À l'exécution, on doit renseigner un login et un mot de passe. On peut ainsi se connecter à un compte utilisateur, compte propriétaire de capteur, compte propriétaire d'un air Cleaner, compte gouvernement et compte admin.

On peut ensuite choisir une action à effectuer (obtenir la liste des capteurs, obtenir les infos d'un capteur…) en tapant le numéro correspondant à l'action. Certaines actions ne sont disponibles qu'à certains types de comptes.

Certaines actions nous emmènent vers un nouveau menu demandant l'identifiant d'un capteur, une zone géographique (périmètre ou point précis) ou l'identifiant d'un air Cleaner.

Il peut nous être proposé de préciser une ou deux dates. Il faut écrire les dates sous la forme JOUR/MOIS/ANNÉE. On peut alors obtenir des statistiques en lien avec les paramètres renseignés.

Voici maintenant la maquette de l'interface console. Les chemins **/nom_chemin** sont purement indicatifs et n'apparaîtront pas dans l'application finale.

/ [pas connecté]

Veuillez-vous connecter. Vous pouvez vous connecter en tant qu'utilisateur lambda, utilisateur possédant un capteur, admin, provider.
Login : [INPUT]
Mot de passe : [INPUT]

/ [connecté]

Vous pouvez :
1 - Obtenir la liste des capteurs (**/capteurs_liste**)
2 - Obtenir les infos d'un capteur (**/capteur_info**)
3 - Obtenir les statistiques dans un périmètre géographique (**/zone_stats**)
4 - Obtenir les statistiques à une position géographique (**/position_stats**)
5 - Scanner les capteurs (**/scan**) [ADMIN]
6 - Obtenir la liste des air cleaner (**/cleaners_liste**) [ADMIN/PROVIDER]
7 - Obtenir les infos d'un air cleaner (**/cleaner_info**) [ADMIN/PROVIDER]
8 - Déconnexion

/capteurs_liste

Capteurs publics :
- sensor_id_1 (latitude_1 ; longitude_1)

- sensor_id_2 (latitude_2 ; longitude_2)
…

Capteurs privés :
- user_id_1 : sensor_id_1 (latitude_1 ; longitude_1)
…

[REDIRECTION VERS /]

/capteur_info

Identifiant du capteur : [INPUT]
Le capteur est situé aux coordonnées (latitude ; longitude)

Que souhaitez-vous visualiser ?

1 - Qualité de l'air à un instant donnée (**/capteur_stats_instant**)
2 - Qualité de l'air sur une période donnée (**/capteur_stats_periode**)
3 - Liste des capteurs en zone de qualité d'air similaire (**/capteurs_similaires**)
4 - Revenir au menu (**/**)

/capteur_stats_instant

Date (EX : 2019-12-06 12:30) : [INPUT]
[AFFICHAGE /stats]
[REDIRECTION VERS /capteur_info]

/capteur_stats_periode

Date Début (EX : 2019-12-06 12:30) : [INPUT]
Date Fin (EX : 2019-12-06 12:30) : [INPUT]

[AFFICHAGE /stats]
Tendance : Amélioration / Stabilité / Dégradation
[REDIRECTION VERS / capteur_info]

/capteurs_similaires

Date Début (EX : 2019-12-06 12:30) : [INPUT]
Date Fin (EX : 2019-12-06 12:30) : [INPUT]

Liste des capteurs dans des zones dont la qualité d'air est similaire (dont l'indice ATMO moyen de chaque catégorie sont +- 1 les mêmes) :
- sensor_id_1 (latitude_1 ; longitude_1)
- sensor_id_2 (latitude_2 ; longitude_2)
…

[REDIRECTION VERS /capteur_info]

/zone_stats

Statistiques d'une zone géographique circulaire.
Latitude: [INPUT]
Longitude: [INPUT]

Rayon: [INPUT]

Que souhaitez-vous visualiser ?

1 - Qualité de l'air à un instant donnée (**/zone_stats_instant**)
2 - Qualité de l'air sur une période donnée (**/zone_stats_periode**)
4 - Revenir au menu (**/**)

/zone_stats_instant

Date (EX : 2019-12-06 12:30) : [INPUT]
[AFFICHAGE /stats]
[REDIRECTION VERS / zone_stats]

/zone_stats_periode

Date Début (EX : 2019-12-06 12:30) : [INPUT]
Date Fin (EX : 2019-12-06 12:30) : [INPUT]

[AFFICHAGE /stats]
Tendance : Amélioration / Stabilité / Dégradation
[REDIRECTION VERS / zone_stats]

/position_stats

Obtient les statistiques de la position par triangulation avec des capteurs proches.
Latitude : [INPUT]
Longitude : [INPUT]
Date (EX : 2019-12-06 12:30) : [INPUT]
[AFFICHAGE /stats]

[REDIRECTION VERS /]

/stats

SCORE : Très bon / Bon / Moyen / Médiocre / Mauvais / Très Mauvais
Indice ATMO GLOBAL : …
Indice ATMO en O3 : … (Concentration)
Indice ATMO en SO2 : … (Concentration)
Indice ATMO en NO2 : … (Concentration)
Indice ATMO en PM10 : … (Concentration)

/scan

Liste des capteurs dysfonctionnels :
- sensor_id_1 (latitude_1 ; longitude_1)
- sensor_id_2 (latitude_2 ; longitude_2)

…

Liste des capteurs privés non-fiables :

- user_id_1 : sensor_id_1 (latitude_1 ; longitude_1)

…

[REDIRECTION VERS /]

/cleaners_liste

Liste des Air Cleaners :

- cleaner_id_1 (provider_id_1)
- cleaner_id_2 (provider_id_2)

…

[REDIRECTION VERS /]

/cleaner_info

Id de l'Air Cleaner : [INPUT]

Informations de nettoyage
Coordonnées : (latitude, longitude)
Heure de début : …
Heure de fin : …
Rayon de nettoyage : …
Niveau d'amélioration de la zone : …

[REDIRECTION VERS /]

I.8. Tests de validation

Enfin, nous avons lister les scénarios à tester pour vérifier le bon fonctionnement de l'application :

Function	Type	Test	Error n°	Retour
Connexion avec mail/pseudo et mdp	U/G/P/A	Login invalide	101	Le login est invalide.
		Login incorrect et mdp incorrect	102	Le mot de passe incorrect.
		Login et mdp corrects		
Obtenir la liste des capteurs	U/G/P/A	Pas de liste des capteurs	201	Il n'y a pas de capteurs enregistrés.
		Liste des capteurs présente		

Obtenir les informations sur un capteur	U/G/P/A	Identifiant incorrect	202	L'identifiant est incorrect.
		Capteur non fiable	203	Ce capteur n'est pas fiable.
		Capteur dysfonctionnel	204	Ce capteur est dysfonctionnel.
		Identifiant correct		
Qualité de l'air à un instant donnée	U/G/P/A	Date hors de l'intervalle du capteur	301	La date doit être comprise entre [date_min] et [date_max].
		Date dans l'intervalle du capteur		
Qualité de l'air sur une période donnée	U/G/P/A	Date début hors de l'intervalle du capteur	302	La date doit être comprise entre [date_min] et [date_max].
		Date de fin hors de l'intervalle du capteur	303	La date doit être comprise entre [date_min] et [date_max].
		Date valide		
Liste des capteurs en zone de qualité d'air similaire	U/G/P/A	Date début hors de l'intervalle du capteur	304	La date doit être comprise entre [date_min] et [date_max].
		Date de fin hors de l'intervalle du capteur	305	La date doit être comprise entre [date_min] et [date_max].
		Date début valide		
Obtenir les statistiques dans un périmètre géographique	U/G/P/A	Latitude invalide	205	La latitude doit être comprise entre [latitude_min] et [longitude_max].
		Longitude invalide	206	La longitude doit être comprise entre [longitude_min] et [longitude_max].
		Rayon invalide	207	Le rayon doit être compris entre [rayon_min] et [rayon_max].
		Zone géographique présente		
Qualité de l'air à un	U/G/P/A	Date invalide	306	La date doit être comprise entre

instant donnée				[date_min] et [date_max].
		Date valide		
Qualité de l'air sur une période donnée	U/G/P/A	Date début hors de l'intervalle des capteurs	307	La date doit être comprise entre [date_min] et [date_max].
		Date de fin hors de l'intervalle des capteurs	308	La date doit être comprise entre [date_min] et [date_max].
		Date valide		
Revenir au menu	U/G/P/A	Menu présent		
Obtenir les statistiques à une position géographique	U/G/P/A	Date hors de l'intervalle des capteurs	208	La date doit être comprise entre [date_min] et [date_max].
		Latitude invalide	209	La latitude doit être comprise entre [latitude_min] et [longitude_max].
		Longitude invalide	210	La longitude doit être comprise entre [longitude_min] et [longitude_max].
		Date valide		
Scanner les capteurs	G	Pas de liste de capteur	211	Il n'y a pas de capteurs enregistrés.
	G	Capteurs particuliers non-fiables		
	G	Capteurs dysfonctionnels		
Obtenir la liste des air cleaner	G/P	Pas de liste des air cleaner	212	Il n'y a pas d'air cleaners enregistrés.
	P	Liste des air cleaner possédés		
	G	Liste des air cleaner		
Obtenir les informations sur un air cleaner	G/P	Identifiant invalide	213	L'identifiant est invalide.
	P	Air cleaner non possédé	214	Vous ne possédez pas cet air cleaner.

	P	Air cleaner possédé		
Déconnexion	U/G/P/A	Connecté		
	U = Utilisateur		1xx = interface de 1er niveau	
	G = Gouvernement		2xx = interface de 2e niveau	
	P = Provider possédant un air cleaner		3xx = interface de 3e niveau	
	A = Admin			

Figure 3: tableau des tests de validation

II. Spécifications

II.1. Diagramme de classe

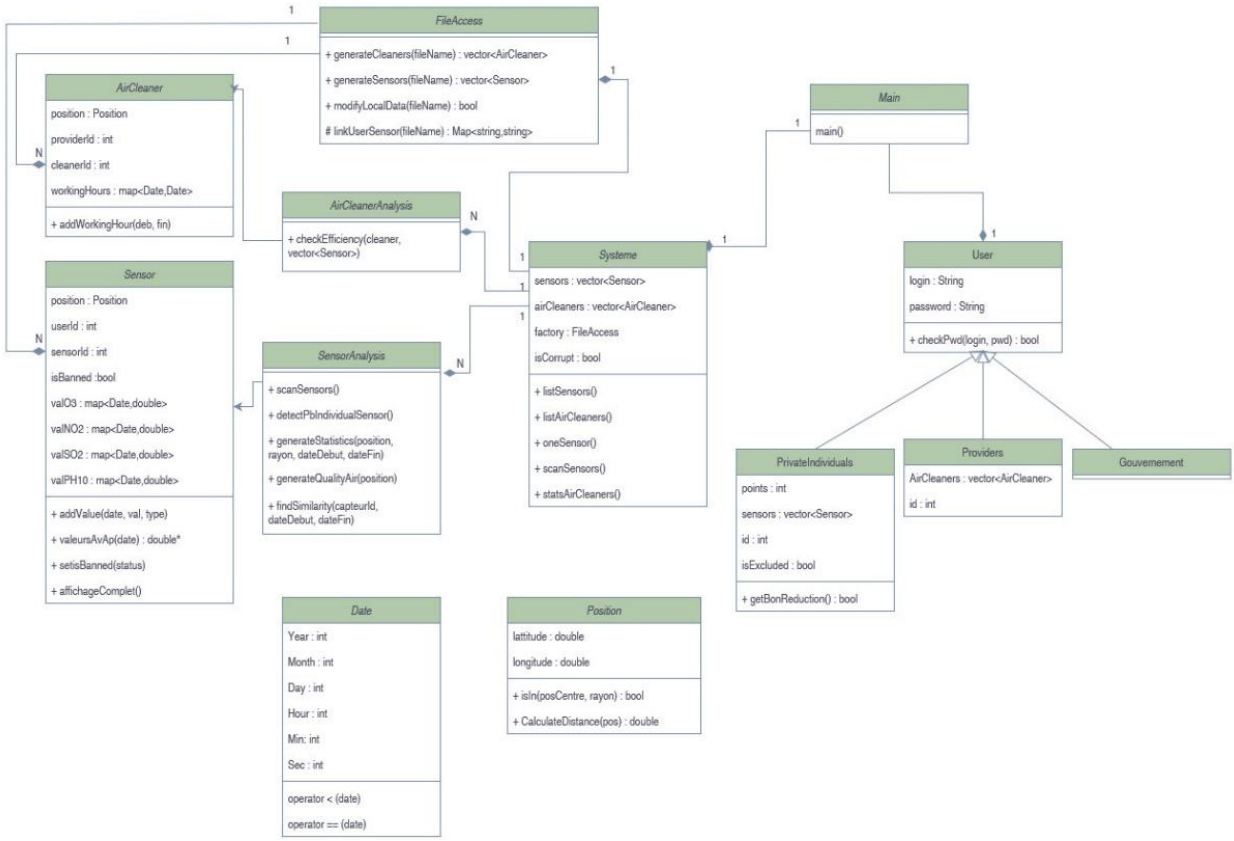


Figure 4: diagramme de classe de AirWatcher

La classe **Main** contient uniquement le menu.

Il commencera par créer un **System**. Le **System** créé la **Fabrique** qui construit tous les éléments à partir des fichiers (**Sensor**, **AirWatcher** et **User**). Ils sont ensuite récupérés par le **System**.

Les **SensorAnalysis** et **CleanerAnalysis** sont créés lors de l'appel d'une méthode et détruits à la fin de la méthode. Cela permettrait de faire du multithreading lors du lancement d'une fonctionnalité.

II.2. Décomposition modulaire

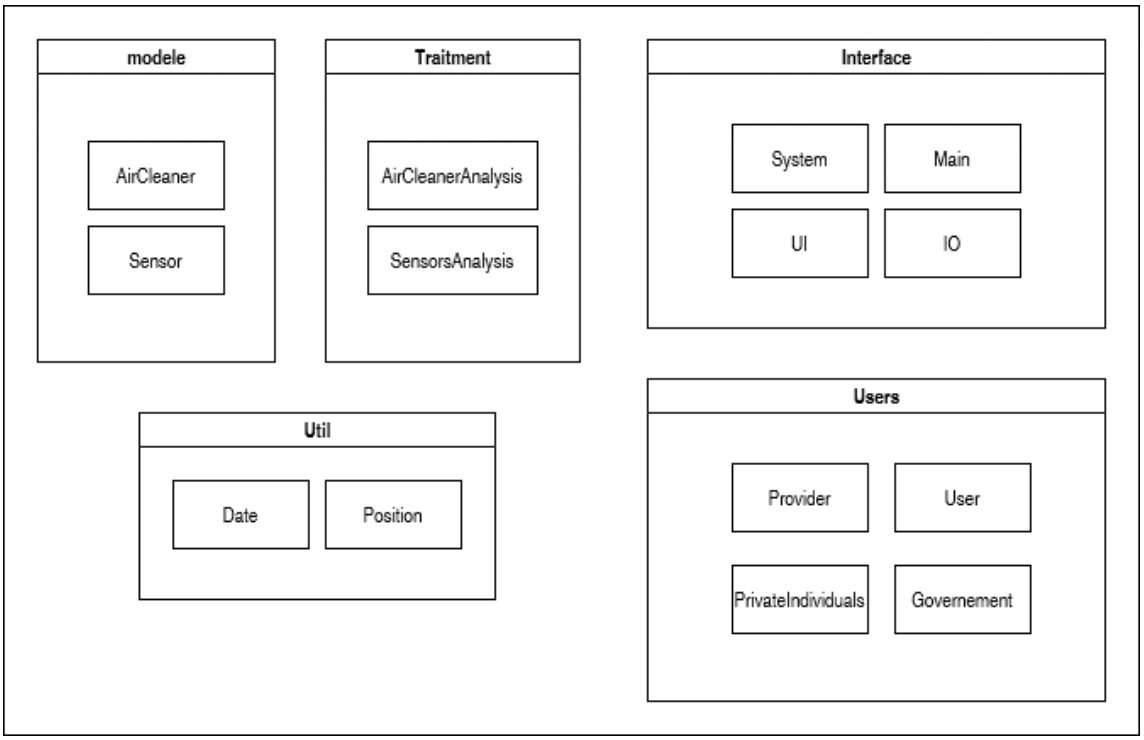


Figure 5: Diagramme de décomposition des classes

II.3. Diagrammes de séquences

II.4. Principaux algorithmes

La méthode scanSensors() permet de détecter les capteurs dysfonctionnels et de les afficher.

```
void scanSensors(listeDesCapteurs) {
        pour chaque capteur dans listeDesCapteurs:
                valeurO3 <- dernière valeur de O3 enregistrée par le capteur
                // Idem pour NO2, SO2 et PM10

                mapDistanceCapteurs <- multimap

                pour chaque autreCapteur dans listeDesCapteurs:
                        si autreCapteur != capteur:
                                mapDistanceCapteurs[distance(capteur, autreCapteur)] = autreCapteur
                        fin si
                fin pour

                moyenneO3 <- 0
                varianceO3 <- 0
                // Idem pour NO2, SO2 et PM10
```

```
                pour les 5 premiers autreCapteur dans mapDistanceCapteurs:
                        O3 <- dernière valeur de O3 enregistrée par autreCapteur
                        moyenneO3 <- (moyenneO3 + O3)
                        moyenneO3 <- moyenneO3 / 5
                        // Idem pour NO2, SO2 et PM10
                fin pour
                pour les 5 premiers autreCapteur dans mapDistanceCapteurs:
                        O3 <- dernière valeur de O3 enregistrée par autreCapteur
                        varianceO3 <- (O3 - moyenneO3)^2 / 4
                        // Idem pour NO2, SO2 et PM10
                fin pour

                si |valeurO3 - moyenneO3| > varianceO3:
                        afficher “Le capteur “ + capteur + “ semble défectueux. Voici ses informations :”
                        affichageInformationsCapteur(capteur)
                fin si
                // Idem pour NO2, SO2 et PM10
}
```

La méthode checkEfficiency(airCleaner) permet de calculer l’efficacité d’un airCleaner sur sa période de fonctionnement

```
checkEfficiency(airCleaner, listeDesCapteurs) {
        mapDistanceCapteurs <- multimap

        pour chaque capteur non défectueux dans listeDesCapteurs:
                mapDistanceCapteurs[distance(airCleaner, capteur)] = capteur
        fin pour

        debut <- date de début de fonctionnement de airCleaner
        fin <- date de fin de fonctionnement de airCleaner
        efficacite <- tableau[4]      // Contiendra l’efficacité pour O3, NO2, SO2, PM10

        capteurPlusProche <- premier capteur de mapDistanceCapteurs
        mesures <- mesures de capteurPlusProche avant debut et apres fin airCleaner

        pour les valeurs de O3, NO3, SO3 et PM10 de mesures (0 <= i < 4) :
                efficacite[i] = (mesure debut - mesure fin) / mesure debut
        fin pour

        si efficacite O3 < 5% ou efficacite NO2 < 5% ou efficacite SO2 < 5% ou efficacite PM10 < 5% :
                afficher “L’air cleaner n’a eu aucun effet sur l’air environnant”
        sinon
                afficher l’efficacite pour O3, NO2, SO2 et PM10

                dernierCapteurEfficace <- capteurPlusProche
                pour chaque capteur de mapDistanceCapteurs:
                        mesures <- mesures de capteur avant debut et apres fin airCleaner
                        efficacite[i] = (mesure debut - mesure fin) / mesure debut
                        si efficacite O3 < 5% ou efficacite NO2 < 5%  … :
                                sortir de la boucle
                        sinon
```

```
                                dernierCapteurEfficace <- capteur
                        fin si
                fin pour

                afficher “Rayon d’action : ” + distance(airCleaner, dernierCapteurEfficace)
        fin si
}
```

La méthode findSimilar() permet de trouver les capteurs similaires à un capteur sur une période donnée

```
findSimilar(capteur, debut, fin, listeDesCapteurs) {

        moyenneO3 <- moyenne de O3 enregistrées par le capteur entre debut et fin
        // Idem pour NO2, SO2 et PM10

        afficher “Capteurs similaires à ” + capteur
        pour chaque autreCapteur dans listeDesCapteurs:
                moyenneAutreO3 <- moyenne de O3 enregistrées par autreCapteur entre debut et fin
                varianceAutreO3 <- variance de O3 enregistrées par autreCapteur entre debut et fin
                // Idem pour NO2, SO2 et PM10

                si |moyenneO3  - moyenneAutreO3 | <= varianceAutreO3 et idem pour NO2, SO2 et PM10 :
                        afficher “Le capteur “ + autreCapteur + “est similaire. Voici ses informations :”
                        affichageInformationsCapteur(autreCapteur)
                fin si
        fin pour
}
```

II.5. Fichier de sauvegarde

Nous devons créer des fichiers de sauvegarde afin de se remémorer les login et mot de passe des utilisateurs. Ces derniers ne seront pas cryptés.

Il s’agit du fichier data.csv. Il contient les données dans le format suivant:

LOGIN ; MDP ; TYPE ; [NB de point] ; [NB de capteurs/airCleaners] ; [ID du capteur/airCleaner 1] ; [ID du capteur/airCleaner 2]…

Le type est 1 si c’est un utilisateur disposant d’un capteur ou non, 2 si c’est un fournisseur d’airCleaner ou 3 si c’est un membre du gouvernement.

Les valeurs suivantes sont présentes pour les utilisateurs et les fournisseurs uniquement.

Pour un utilisateur, il s’agit du nombre de points qu’il a accumulé, puis du nombre de capteurs qu’il possède et enfin liste des id de ces capteurs.

Pour les fournisseurs, il s’agit uniquement du nombre de airCleaner déployé ainsi que la liste des id de ces derniers.

II.6. Tests unitaires

User checkpasswd(String login, String password)

Doit renvoyer user si l’utilisateur est dans le fichier data.csv

nullptr sinon

données : checkpasswd(user1, mdp) ⇒ user

checkpasswd(user1, mdp2) ⇒ NULLPTR

String getBonReduction()

Si le user.getPoints() est suffisant : Renvoyer vrai

Sinon : Renvoyer faux

données : if user.getPoints() > 150 ⇒ True

user.getPoints() < 150 ⇒ False

bool scanSensors()

Si le capteur possède une anomalie (par exemple un écart de valeur > 10% par heure) on renvoie True et on appelle aSensor.setIsValid(false)

Sinon on renvoie False

données : pour O2 : 12h 21.1, 13h 22.0, 14h 21.5 ⇒ False

: 12h 21.1, 13h 35.0, 14h 15.5 ⇒ True

String generateStatistics(pos, startTime, endTime)

Renvoie les statistiques (voir Partie 1) sous forme d'une chaîne de caractères

Si pas de données viables, renvoyer "Pas de données"

generateStatistics(0.5:1, 01/01/2023:13h, 01/01/2023:13h)

if unCapteur.pos == 0.5:1 && unCapteur. ⇒ mesure

if tousLesCapteur.pos != 0.5:1 ⇒ pas de mesure

String generateStatistics(pos, r, startTime, endTime)

Renvoie les statistiques (voir Partie 1, pas de capteur à moins de 50km) d'une zone sous forme d'une chaîne de caractères

Si pas de données viables, renvoyer "Pas de données"

vector<Sensor> detectSimilarZones(Sensor aSensor, startTime, endTime)

Renvoie un vecteur de capteurs étant similaire (même moyenne et écart-type) triés par ordre décroissant de similarité.

Renvoie un vecteur vide s'il n'existe pas de capteurs similaires.

(moyenne des données sur le temps passé en paramètre)

Sensor 1 : O3 = 50 NO2 = 45

Sensor 2 : O3 = 49 NO2 = 46 ⇒ les deux capteurs sont similaires

Sensor 1 : O3 = 50 NO2 = 65

Sensor 2 : O3 = 49 NO2 = 70 ⇒ aucun capteur n'est similaire

double checkEfficiency(AirCleaner)

Renvoie la zone d'éfficacité en mètre du airCleaner

données : Nous avons 5 capteurs disposés, chacun à une distance différente du aircleaner. Le cinquième capteur est le plus loin. Pour les quatre premiers capteurs, les mesures sont réduites de 25 %. Or, pour le cinquième, elles ne sont pas réduites. Le rayon d'efficacité doit donc se baser sur le capteur 4 avec une efficacité de 25 %

Bool IsIn(position, r)

Renvoie vrai si la position est dans le cercle donné en paramètre

Sinon renvoie faux

données : this.lat : 45 this.long : 0.5

lat = 45 long = 0.7 r = 2000km ⇒ true

lat = 45 long = 0.7 r = 50m ⇒ false