Projet GL - AirWatcher
Dossier de Spécifications

I. Introduction

Un organisme gouvernemental est en charge de la surveillance de la qualité de l'air sur un territoire de grande échelle. Un nombre important de capteurs est installé sur ce territoire, et chaque capteur produit des données à une fréquence régulière.
Ce logiciel aura pour but de regrouper, analyser et présenter ces données d'une façon qui permettra de prendre des décisions pour améliorer la qualité de l'air.

II. Spécifications générales

1. Acteurs

Acteurs	Actions
Agence gouvernementale	Suivi de la qualité de l'air par le biais de données récoltés par des capteurs pour appuyer une prise de décision.
Entreprises de purificateurs géants	Installation des purificateurs, suivi de l'impact de leurs produits sur la qualité de l'air.
Particuliers	Enrichir la récolte de données à travers leurs téléphones portables.
Capteurs	Récolte des données.
Administrateur	Gérer les inscriptions et les comptes.

2. Matériel

L'application nécessite un ordinateur muni d'un environnement de compilation et d'exécution de code C++.

3. Données

Les données sont sauvegardés dans différents fichier .CSV organisés de la manière suivante:
Fichier mesures: Timestamp;SensorID;AttributeID;Value;

Fichier capteurs: SensorID;Latitude:Longitude;Description;
FIchier types de mesures: AttributeID,Unit;Description;

III. Spécifications fonctionnelles

Fonctionnalité	**1.0** Se connecter
Description	ermet à chaque acteur de s'authentifier auprès de l'application
Données Nécessaires	identifiant, mot de passe
Pré-conditions	Ne pas être connecté
Post-conditions	L'acteur est connecté
Données en sortie	(succès, échec) : (0,1)
Cas limite	Combinaison identifiant/mot de passe non reconnue

Fonctionnalité	**1.1** Se déconnecter
Description	Permet à l'acteur de fermer sa session en cours
Données Nécessaires	Aucune
Pré-conditions	Être connecté
Post-conditions	Aucune fonctionnalité de l'application est utilisable / Être déconnecté
Données en sortie	(succès,échec) : (0,1)
Cas limite	

Fonctionnalité	**1.2** Mesurer l'efficacité des installations
Description	Donne des informations sur la qualité de l'air entre deux instants donnés, et dans une zone centrée sur une installation dont la taille est à définir
Données Nécessaires	Données capteurs avant, Données capteurs après ("timespan")
Pré-conditions	sposer de données capteurs (.csv) exportées dans la BD interne à l'application
Post-conditions	Les données sont analysées et transmises
Données en sortie	Valeur positive ou négative (entier)
Cas limite	Données erronées / incohérence / non-existence ou manque de données

Fonctionnalité	**1.3** Envoyer des données concernant la qualité de l'air
Description	Envoyer données concernant l'air à un instant ou une période donnée, pour une zone géographique, à travers le fichier CSV uniquement
Données Nécessaires	Nom du fichier
Pré-conditions	Disposer de données capteurs (.csv)
Post-conditions	Les données sont bien reçue et sont stockées dans l'application
Données en sortie	Valeurs contenues dans le .csv / Taille de la zone assainie
Cas limite	instant t inexistant, période p ou instant t mal renseignés

Fonctionnalité	**1.4** Visualisation de la quantité de points (particulier)
Description	Le particulier a accès à son score.
Données Nécessaires	Le compte actif
Pré-conditions	Être connecté
Post-conditions	Visualisation de la quantité de points
Données en sortie	Un entier positif ou nul
Cas limite	

Fonctionnalité	**1.5** Identifier les capteurs “identiques”
Description	Sélectionne les capteurs qui présentent un comportement, une tendance similaire pour toutes Les mesures ainsi qu’une proximité géographique
Données Nécessaires	Des données capteurs
Pré-conditions	
Post-conditions	Une série de capteurs est transmise
Données en sortie	Une liste de capteurs
Cas limite	Aucun capteur en mémoire

Fonctionnalité	**1.6** Suivre la qualité de l'air / Agréger les données
Description	Analyse les données reçues et renvoie une description détaillée et synthétique de la qualité de l'air actuelle
Données Nécessaires	Les données des capteurs dans un périmètre donné (obtenu par la fonctionnalité **1.9** ou par des données saisis par clavier, e.g : cercle : coordonnée de centre, rayon)
Pré-conditions	
Post-conditions	Fournit un agrégat de données
Données en sortie	Un rapport synthétique de la qualité de l'air
Cas limite	Si aucun périmètre est fourni le système avertit

Fonctionnalité	**1.7** Identifier les données erronées
Description	Identifier et traiter (flag levé) les données erronées
Données Nécessaires	données envoyées par les particuliers
Pré-conditions	avoir une base de données recueillies
Post-conditions	il n'y a plus de fausses données dans la base de données
Données en sortie	fausses données
Cas limite	Erreurs faibles ou données erronées en proportions importantes

Fonctionnalité	**1.8** Attribution de points à un particulier
Description	Attribut à un utilisateur une quantité de points à un particulier en fonction de la qualité de sa contribution (nombre et cohérence de la/les donnée.s)
Données Nécessaires	Donné capteur du particulier
Pré-conditions	Avoir une donnée particulier

Post-conditions	Modification du compteur de points du particulier
Données en sortie	Valeur positive ou négative (entier)
Cas limite	Pas de particulier associé à la donnée particulier

Fonctionnalité	**1.9** Afficher et saisir le territoire
Description	IHM graphique qui affiche la carte et les capteurs et permet également aux utilisateurs de choisir un périmètre là-dessus : La carte peut être zoomée/dézoomée. (Servir un API extérieur pour son implémentation si possible)
Données Nécessaires	Données de cartes et capteurs
Pré-conditions	Cohérence entre les données de cartes et celles de capteurs présents
Post-conditions	Fournit des données géographique pour l'analyse à la suite
Données en sortie	Périmètre choisi
Cas limite	Format de carte trop grand/petit qui nécessite un redimensionnement

Fonctionnalité	**1.10** Connaître la zone impactée par un purificateur
Description	IHM graphique qui affiche la carte et les capteurs et permet également aux utilisateurs de choisir un périmètre là-dessus : La carte peut être zoomée/dézoomée. (Servir un API extérieur pour son implémentation si possible)
Données Nécessaires	Données de cartes et capteurs
Pré-conditions	Cohérence entre les données de cartes et celles de capteurs présents
Post-conditions	Fournit des données géographique pour l'analyse à la suite
Données en sortie	Périmètre choisi

Cas limite	Format de carte trop grand/petit qui nécessite un redimensionnement

IV. Spécifications non fonctionnelles

2.0 Qualité	Base de données bien formatée et exploitable. Code compréhensible et bien structuré.
2.1 Performance	Temps de réponse et d'actualisation convenable et complexité mesurable. (réponse < 5 secondes)
2.2 Robustesse	L'application doit assurer un bon fonctionnement et ne pas perdre sa stabilité face à des requêtes invalides ou toute utilisation ne respectant pas le manuel. (moins de 2 bug sur 100 exécutions de l'application)
2.3 Sécurité	La base de données doit être sécurisée et les droits d'accès bien maîtrisés. La base doit rester intègre en cas de bug.
2.4 Maintenabilité	L'application doit facilement permettre la mise à jour ainsi que la correction des données et des fonctionnalités
2.5 Capacité à évoluer	Implémentation de façon à ce que l'application puisse être enrichie par de nouvelles fonctionnalités et adaptable à un changement d'ordre de grandeur de la demande. (réutilisabilité, modules indépendants)
2.6 Fiabilité	Résultats obtenus corrects et précis.
2.7 Testabilité	Application fournie avec une suite de tests permettant, à tout moment, de vérifier son bon fonctionnement.

V. Analyse des risques liés à la sécurité

1. Injections

Les injections permettent de polluer les données, contourner certaines protections et d'accéder à des données censées être inaccessibles.
Une solution pour s'en affranchir est d'imposer des formats pour contrôler la saisie: type, longueur, veiller à ce que les caractères spéciaux soient échappés, de valider tous les champs de saisie.
Plus spécifiquement dans le contexte de cette application, il faudra surveiller les particuliers soumettant intentionnellement des données faussées. Pour ce faire, on pourrait associer un score de fiabilité à chaque particulier, calculé en se basant sur la cohérence des données qu'il fournit avec celles récoltés par des capteurs certifiés de qualité et éventuellement bannir les plus nuisibles.

2. Authentification faible et sessions vulnérables

Il s'agit ici du risque de se faire pirater son compte pour dans le but de polluer au nom de quelqu'un d'autre la base de données ou d'exploiter leurs points. Il existe pour cela de nombreuses pratiques de sécurisation comme l'utilisation de témoins de session sécurisés (cookies) ainsi que des jetons CSRF pour les formulaires. Ceci ne sera pas mis en place pour l'application car il ne s'agit pas d'un exercice de sécurité mais il demeure important d'en être sensibilisé.

3. Exposition de données sensibles

Une mauvaise encryption des données sensibles comme les mot-de-passe, les données personnelles ou celles liés aux programmes de points faciliterait les cas de fraudes. C'est pour ça qu'il est important de crypter les données et de ne pas stocker les plus sensibles.

4. Mauvaise configuration de la sécurité

Une configuration de sécurité obsolète ou mal configurée peut entraîner des accès, volontaires ou non, à des données ou à des fonctions de l'application. La meilleure façon de lutter contre ce problème est un

processus de consolidation répétables et testable et des processus de mise à jour et de correction réguliers.

Système	Atout	Vulnérabilité	Attaque	Risque
Application AirWatcher	Données récoltées Données personnelles	Données stockées et transmises sans cryptage Des mots de passes faibles sont autorisés par l'application Les données saisies ne sont pas validées	L'attaquant fausse les données stockées dans la base L'attaquant saisie des données inattendues L'attaquant utilise du code malveillant L'attaquant devine le mot de passe L'attaquant accède à des données qui ne sont pas censées lui être accessibles	L'attaquant obtient l'accès aux données personnelles d'un autre utilisateurs (son score par exemple) L'attaquant supprime la base de données ou nuit à son intégrité

VI. Plan de tests de validation

1. Approche Générale

On va tester les spécifications fonctionnelles et non fonctionnelles à l'aide des tests unitaires C++ (particulièrement pour les spécifications fonctionnelles). Beaucoup de spécifications non fonctionnelles seront vérifiées par les spécifications fonctionnelles ou en vérifiant si le programme entier fonctionne correctement. Les spécificités sont indiquées dans les tableaux ci-dessous.

2. Tests fonctionnels

Test No.	Fonctionnalité Testée	Scénario	Entrée	Critères de succès/échec
1.0	Se connecter	Un particulier, une entreprise ou un employé de l'agence veut accéder à son compte	1. Nom d'utilisateur 2. Mot de passe	L'utilisateur peut accéder à son compte si les données saisies sont bonnes. Le cas contraire, l'application continue de fonctionner et lui signale que la saisie est erronée.
1.1	Se déconnecter	L'utilisateur souhaite mettre fin à sa session.	--	La session en cours est achevée avec un code de retour 0. Dans le cas d'un problème, la valeur -1 est retournée.
1.2	Mesurer l'efficacité des installations	Une entreprise veut estimer l'impact de ses purificateurs.	1. Données captures avant 2. Données captures après	Retour de valeurs numériques indiquant la différence entre 2 différentes consultation.
1.3	Envoyer les données	Chaque capteur	1. Instant t	Les données

	concernant la qualité de l'air	envoie à une fréquence fixe les données qu'il a récoltées à la base	2. Période p	envoyées sont bien ajoutées au fichier CSV concerné. Dans le cas d'erreur, la valeur -1 est retournée.
1.4	Visualisation de la quantité de points (particulier)	Un particulier veut consulter les points qu'il a cumulés	1. Saisie sur la console à partir d'une session	Le score est affiché pour la session en cours. Dans le cas d'erreur, la valeur -1 est retournée.
1.5	Identifier les capteurs "identiques"	L'agence veut repérer les capteurs fournissant des résultats similaires.	1. Des données capteur	L'application retourne dans un fichier des "clusters" de capteurs regroupés selon les valeurs de leurs mesures.
1.6	Suivre la qualité de l'air / Agréger les données	Dans un certain périmètre, obtenir des information sur la qualité de l'air pendant un certain temps	1. Les données capteurs dans un périmètre donné	L'application retourne certaines informations sur la qualité de l'air, telles que la qualité moyenne de l'air
1.7	Identifier les "fausses données"	Vérifier si les données envoyées par des particuliers sont fausses.	1. Données envoyées par les particuliers	Lever un flag "données fausses" et identifier l'utilisateur qui en est la source.
1.8	Attribution de points à un particulier	Un particulier se voit attribuer des points lorsque des données qu'il a fournies sont exploitées.	1. Donnée capteur du particulier	Le solde du particulier est incrémenté.
1.9	Afficher et saisir le territoire	Un particulier choisit un périmètre sur la carte pour commencer la requête d'analyse.	1. Données de cartes et capteurs	L'affichage sera bon si la cohérence de données est vérifiée.

3. Tests non-fonctionnels

Test No.	Fonctionnalité Testée	Scénario	Entrée	Critères de succès/échec
2.0	Qualité	On vérifie qu'il n'y a pas d'erreurs dans le code	1. Programme complet	La réussite du programme
2.1	Performance	On vérifie si le programme n'est pas trop lent	1. Programme complet	Pas de time-out ni de longs délais d'attente (plus de 5 secondes)
2.2	Robustesse	On vérifie si le programme peut gérer des commandes anormales/invalides	1. Des cas extrêmes	La réussite du programme avec moins de 2 bug sur 100 exécutions de l'application
2.3	Sécurité	On vérifie si les accès à la base de données sont bien gérés	1. Une demande d'accès à la base de données	Accès à la base de données si l'utilisateur en a la permission. En cas de bug, avoir une version stockée des données.
2.4	Maintenabilité	On vérifie si les changements au système sont correctement mis en œuvre	1. Programme complet	La réussite du programme avec les modifications apportées aux données/fonctionnalités
2.5	Capacité à évoluer	On vérifie si le système peut traiter des bases de données/demandes plus importantes	1. Programme complet	La réussite du programme sans perte importante de qualité ou de performance
2.6	Fiabilité	On vérifie si nos fonctions et nos algorithmes	1. Des tests unitaires 2. Programme	La succès des tests unitaires pour toutes les

		fonctionnent correctement	complet	spécifications fonctionnelles
2.7	Testabilité	On vérifie si tous nos tests unitaires fonctionnent correctement	1. Des tests unitaires	L'exécution de tous les tests unitaires sans erreur

IX. Planning

Date	**Actions**
Semaine du 18 mars	Compréhension du sujet, début de la définition des spécifications
Cemaine du 15 avril	Diagramme de classes, finalisation de la liste des tests Réflexions sur les outils à utiliser et la répartition du travail au sein du groupe
Semaine du 3 mai	Mise en place du GitHub, début du développement
Mois de mai	Développement du code en continu, révisions de certains algorithmes ou fonctionnalités au vu des problèmes apparaissant
Fin mai	Finalisation du code, création des différents livrables à fournir avec le projet