Plan de tests

Informations

Auteurs : Fabien GELUS, Baptiste PAULETTO, Louis UNG, Mengxin ZHANG
Version courante : V0.3
Date de création : 01/04/2019
Dernière modification : 02/05/2019

Gestionnaire de version

Date	Version	Historique des modifications	Auteur
01/04/19	V0.1	Création du document et formulation du standard	Fabien GELUS, Baptiste PAULETTO, Louis UNG, Mengxin ZHANG
09/04/19	V0.2	Ajout au document d'informations pour prendre en compte les compléments fournis sur Moodle	Baptiste PAULETTO
02/05/19	V0.3	Reprise du document et enrichissement des différents scénarios possibles	Baptiste PAULETTO

Table des matières

1. Tests de validation

Scénarios de tests

Pour chacun des scénarios suivants, nous nous placerons dans le cadre d'une rencontre avec le client afin de lui démontrer que les différentes fonctionnalités proposées fonctionnent effectivement.
Les données utilisées pour la réalisation de ces tests et qui seront présents dans l'archive test sont les suivants : data_10sensors_1year.csv (fourni dans une archive sur moodle).
Un premier test d'installation puis lancement de l'application est facteur de tous les autres et nous n'allons donc pas le répéter au début de chaque scénario, vous le trouverez ci-dessous :

➔ Installation & lancement de l'application :
L'archive du projet est téléchargée puis décompressée.
Le client se place en suite dans le répertoire contenant le projet et effectue la commande approprié au lancement, soit : `./atmo`

L'application étant désormais lancée, nous pouvons passer au tests des différentes fonctionnalités

Scénario 1

Le client souhaite obtenir la qualité moyenne de l'air pour certaines coordonnées GPS, un rayon défini précisément ainsi qu'une date de début et de fin de l'analyse.
Il tapera alors la commande suivante :
Qm -d -19.4789835505555 -35.2425725968753 1

2017-01-01T00:01:20.609 2017-01-01T00:30:39.004

Suite à cela, il obtiendra les valeurs correspondantes à chacun des sous indices du capteur Sensor0 qui est celui lié à cette zone géographique avec cette durée de temps, plusieurs sorties sont donc possibles :

- Atmo Score: 2 (Très bon)
- O3;17.8902017543936
- NO2;42.4807462361763
- SO2;13.6449094925285
- PM10;1.55796479844986

Si l'on omet l'option –d, l'affichage sera de la forme suivante :

- Atmo Score: 2 (Très bon)

Si l'on sélectionne une zone où aucun capteur ne récupère de données :

- Pas de capteurs dans la zone sélectionnée.

Si la période de temps fournie est trop courte/aucun capteur ne capturait durant cette période :

- Période de temps trop courte, aucun capteur trouvé.

Les cas limites liés à l'utilisation de cette commande sont :

- Une période de temps trop courte pour permettre d'obtenir des informations de différents capteurs, empêchant alors l'analyse de la qualité dans la zone.
- Une période de temps qui dépasse les données disponibles sera alors tronquée pour correspondre aux informations dont on dispose.
- Une absence de capteurs dans la zone sélectionnée, rendant impossible l'analyse.
- Un rayon en kilomètre beaucoup trop grand pour un seul capteur fera appel à d'autres capteurs à proximité, engendrant alors des moyennes.

Les cas d'erreurs liées à l'utilisation de cette commande sont :

- Une erreur dans les paramètres passés : Coordonnées GPS inexistante, période de temps négative.
- Le format des paramètres non respectés : horodate incorrecte, virgule dans les coordonnées GPS.
- Un rayon en kilomètre non entier (pas de virgules sont attendues).
- L'omission d'un paramètre, s'il manque la longitude, la latitude ou une des deux horodates, le traitement sera alors impossible et une erreur se produira.

Scénario 2

Le client souhaite obtenir la liste des capteurs ayant un comportement similaire pour une période de temps donnée.
Il tapera alors la commande suivante :
sim *2017-01-01T00:00:00 2017-01-01T00:30:00*

Le système calculera le score Atmo de chacun des capteurs pour cette période et regroupera les résultats par valeur tout en affichant les résultats par ordre décroissant (du meilleur au moins bon score) :

- Atmo Score 1 (Très bon) :
- Sensor1
- Sensor9
- Atmo Score 2 (Très bon) :
- Sensor0
- Sensor2
- Sensor3
- Sensor4
- Sensor5
- Sensor6
- Sensor7
- Sensor8

Les cas limites liés à l'utilisation de cette commande sont :

- Une période de temps trop courte pour permettre d'obtenir des informations de différents capteurs, empêchant alors la comparaison de score.

Les cas d'erreurs liées à l'utilisation de cette commande sont :

- Une erreur dans les paramètres passés : période de temps négative.
- Le format des paramètres non respectés : horodate incorrecte.
- L'omission d'un paramètre, s'il n'y a qu'une seule horodate, une des deux bornes est donc manquante et le calcul impossible.

Scénario 3

Le client souhaite repérer potentiellement des capteurs en situation de dysfonctionnement sur une période de temps donnée.
Il tapera alors la commande suivante :
dysfonc *2017-01-01T00:00:00 2017-01-01T00:30:00*

Le système cherchera alors à trouver des capteurs qui ont un comportement incohérent et retournera une liste de ces derniers.
Pour déterminer cette incohérence, trois étapes sont exprimées, voir spécification fonctionnelle.
Plusieurs sorties seront alors possibles :

- Aucun capteur en dysfonctionnement dans cet intervalle.

Ou bien :

- Capteurs considérés comme en dysfonctionnement :
- Sensor0 (valeurs négatives)
- Sensor7 (aucune valeur retournées)

Les cas limites liés à l'utilisation de cette commande sont :

- Une période de temps trop courte pour permettre d'obtenir des informations de différents capteurs, ayant donc un retour vide simplement puisqu'aucun capteur n'était présent dans l'intervalle de temps.

Les cas d'erreurs liées à l'utilisation de cette commande sont :

- Une erreur dans les paramètres passés : période de temps négative.
- Le format des paramètres non respectés : horodate incorrecte.
- L'omission d'un paramètre, s'il n'y a qu'une seule horodate, une des deux bornes est donc manquante et la définition du dysfonctionnement impossible.

Scénario 4

Le client souhaite consulter les valeurs caractéristiques d'un point précis dont il connait les coordonnées GPS.
Il tapera alors la commande suivante :
carac *-19.4789835505555 -35.2425725968753 1*

2017-01-01T00:01:20.609 2017-01-01T00:30:39.004

Le système procèdera à un calcul de moyenne pondérée avec les capteurs alentours avant d'afficher le score Atmo ainsi que les valeurs qui ont permis à l'établissement de ce score.

- Atmo Score 2 (Très bon), valeurs moyennes utilisées :
- O3;17.8902017543936
- NO2;42.4807462361763
- SO2;13.6449094925285
- PM10;1.55796479844986

Si la période de temps fournie est trop courte/aucun capteur ne capturait durant cette période :

- Période de temps trop courte, aucun capteur trouvé.

Si les coordonnées GPS sont trop éloignées de capteurs :

- Aucun capteur à proximité, analyse impossible.

Les cas limites liés à l'utilisation de cette commande sont :

- Une période de temps trop courte pour permettre d'obtenir des informations de différents capteurs, empêchant alors l'analyse de la qualité dans la zone.
- Une période de temps qui dépasse les données disponibles sera alors tronquée pour correspondre aux informations dont on dispose.
- Des coordonnées GPS étant trop éloignées de capteurs (voir spécification fonctionnelle), seront considérées comme incorrecte rendant donc l'analyse incorrecte.

Les cas d'erreurs liées à l'utilisation de cette commande sont :

- Une erreur dans les paramètres passés : Coordonnées GPS inexistante, période de temps négative.
- Le format des paramètres non respectés : horodate incorrecte, virgule dans les coordonnées GPS.
- L'omission d'un paramètre, s'il manque la longitude, la latitude ou une des deux horodates, le traitement sera alors impossible et une erreur se produira.

2. Tests unitaires

Les tests unitaires que nous avons réalisés sont liées au projet et ont été développés à la suite de la validation de notre modélisation avec le client, vous pourrez les retrouver intégralement depuis le projet dans la partie UnitTest.
Vous trouverez ci-dessous un exemple de test rédigé au préalable du développement de l'application, lorsque nous avons essayé de nous conformer au TDD (Test-Driven Development) :

```
#include "stdafx.h"
#include "CppUnitTest.h"
#include "string"
#include "../ATMO/Capteur.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest
{
	TEST_CLASS(RecupereationDesDonnees)
	{
	public:
		TEST_METHOD(RecuperationDesDonnees)
		{
			TraitementDeDonnees
			Assert::AreEqual(toString, "Sensor1 Latitude:-
8.15758888291083 Longitude:-34.7692487876719");
		}
		TEST_METHOD(ConstructeurParDefaut)
		{
			Capteur c;
			std::string toString = c.toString();
			Assert::AreEqual(toString, "Sensor1 Latitude:-
8.15758888291083 Longitude:-34.7692487876719");
		}
		TEST8
	};
}
```

3. Tests de non-régression

Tout au long de notre développement, nous avons veillé qu'à chaque fin de session de production en équipe, nous n'avions pas empêché ce qui fonctionnait déjà, de continuer à fonctionner.
En effet, la création des classes « brique » du projet tel que Capteur, Mesure ou encore TypeMesure ont été réalisées dans un premier temps et par la suite, nous veillions qu'après chaque méthode et/ou classe les utilisant, elles continuaient de fonctionner convenablement.
Pour cela, nous avions mis en place des tests simples aves des valeurs basiques pouvant être vérifiées rapidement à l'exécution.