**Plan de test UC Archiver les données : Projet Smart Territories**

**1- Identification du test**

Tests\_UCArchiverLesDonnées\_SmartTerritories

**2- Référence du module testé**

Ces tests seront effectués sur les méthodes de la classe CClientMQTT et CAnalyse, on travaillera sur la méthode de vérification des données puis envoi des données (de CAnalyse) ainsi que la réception des données du Client. Les données seront ensuite envoyées par l’API. On vérifiera donc le cycle complet du cas d’utilisation « Archiver les Données ».

**3- Objectif du test**

Ce test vérifiera le bon enregistrement des données dans la base, ainsi que les différents codes d’erreur lors de leur validation. Les différents scénarios seront validés.

**4- Procédure du test**

Test 1 : Envoi de données correctes par le broker

Données de test :

Requête broker :   
mosquitto\_pub -h localhost -t SmartTerritories -m '{"Capteur":"Roseraie","TypeDeDonnee":"PM10","Date":"2025-04-25 14","Valeur":"80"}'

Retour : Les données ont bien été enregistrées dans la base de données

Test 2 : Données incomplètes

*Test 2.1 🡺*

Données de test :

Requête broker :

mosquitto\_pub -h localhost -t SmartTerritories -m '{"Capteur":"Roseraie","Date":"2025-04-25 14","Valeur":"80"}'

Retour : « Erreur, champ manquant : TypeDeDonnee »

*Test 2.2 🡺*

Données de test :

Requête broker :

mosquitto\_pub -h localhost -t SmartTerritories -m '{"TypeDeDonnee":"PM10","Date":"2025-04-25 14","Valeur":"80"}'

Retour : « Erreur, champ manquant : Capteur »

*Test 2.3 🡺*

Données de test :

Requête broker :

mosquitto\_pub -h localhost -t SmartTerritories -m '{"Capteur":"Roseraie","TypeDeDonnee":"PM10","Valeur":"80"}'

Retour : « Erreur, champ manquant : Date »

*Test 2.4 🡺*

Données de test :

Requête broker :

mosquitto\_pub -h localhost -t SmartTerritories -m '{"Capteur":"Roseraie","TypeDeDonnee":"PM10","Date":"2025-04-25 14"}'

Retour : « Erreur, champ manquant : Valeur »

Test 3 : Date au mauvais format

Données de test :

Requête broker :

mosquitto\_pub -h localhost -t SmartTerritories -m '{"Capteur":"Roseraie","TypeDeDonnee":"PM10","Date":"2025-4-25 1","Valeur":"80"}'

Retour : « Format de date invalide. Format attendu : 'AAAA-MM-JJ HH' »

Test 4 : La valeur n’est pas un nombre ou convertible en nombre.

Données de test :

Requête broker :

mosquitto\_pub -h localhost -t SmartTerritories -m '{"Capteur":"Roseraie","TypeDeDonnee":"PM10","Date":"2025-04-25 14","Valeur":"abcd"}'

Retour : « Valeur non numérique »

**5- Résultats attendus**

Test 1 : Les données sont enregistrées dans la base

Test 2.1 : Message -> « Erreur, champ manquant : TypeDeDonnee »

Test 2.2 : Message -> « Erreur, champ manquant : Capteur »

Test 2.3 : Message -> « Erreur, champ manquant : Date »

Test 2.4 : Message -> « Erreur, champ manquant : Valeur »

Test 3 : Message -> « Format de date invalide. Format attendu : 'AAAA-MM-JJ HH' »

Test 4 : Message -> « Valeur non numérique »

**6- Moyens à mettre en œuvre**

Pour effectuer ces tests il faudra se servir de la machine virtuelle servant de test au code. Les différentes requêtes seront inscrites dans la procédure et l’accès à la base sera permis, ainsi on pourra vérifier l’enregistrement des données

Avant d’effectuer ces tests, il faudra démarrer l’API dans le terminal de la machine virtuelle. Il faudra se positionner dans le dossier : « Documents/API-ProgObjet » (cd Documents/API-ProgObjet) puis faire la commande : « node . » pour lancer l’API. Ensuite, on pourra se positionner dans un autre terminal et envoyer les données au client par le broker.

Dans le cas des tests, le broker est installé en localhost, ainsi on utilisera un broker directement positionné sur le serveur, cela nous permet de ne pas avoir besoin d’un accès à un réseau public.

J’ai donc installé un broker mosquitto au préalable de ces tests, directement sur la machine pour pouvoir effectuer les tests.

**Annexe : Données retournées/enregistrées par les tests**

**Test 1 : Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.**

**Test 2.1 : **

**Test 2.2 : **

**Test 2.3 : **

**Test 2.4 : **

**Test 3 : **

**Test 4 : **