**Plan de test CAnalyse : Projet Smart Territories**

**1- Identification du test**

Tests\_CAnalyse\_SmartTerritories

**2- Référence du module testé**

Ces tests seront effectués sur la classe CAnalyse, on travaillera sur le constructeur de la classe, la méthode de vérification des données puis l’envoi des données jusqu’à l’API qui sera ici non disponible. Cela nous permettra donc de vérifier l’entièreté de la classe CAnalyse.

**3- Objectif du test**

Ce test vérifiera la validation des données reçues, ainsi que leur envoi à l’API. Les différents scénarios seront validés.

**4- Procédure du test**

Test 1 : Instanciation d'un objet CAnalyse

Ce test correspond au constructeur de la classe, on instanciera un nouvel objet de la classe CAnalyse, comprenant le lien à notre API (ici fictive).

En prenant en compte la mise en œuvre de la partie 6 de ce plan de test « Moyens à mettre en œuvre ainsi que l’annexe correspondant à la fiche de test, on obtient le résultat suivant :

« URL d'API choisie : http://localhost:8080 /SmartTerritories/addData »

Ainsi, on nous retourne les données correspondant à l’API, le chemin de connexion à celle-ci

Test 2 : Réception de données pour envoi

Ce test correspond à la méthode « envoiMQTT » de la classe. Elle nous permet, après vérification, d’envoyer les données à l’API pour enregistrement. Ici notre API n’existe pas. Les lignes de codes correspondant à l’envoi ont donc été masquées. On retourne simplement ce qui devrait être envoyé pour la sauvegarde :

« Paquet MQTT validé avec succès.  
Donnée envoyée à l’API : {  
 Capteur: 'Roseraie',  
 TypeDeDonnee: 'Temperature',  
 Date: '2025-05-19 15',  
 Valeur: 23.4  
} »

On remarque aussi la ligne permettant d’identifier la validation de vérification, qui nous permets de savoir que celle-ci est terminée.

Cette suite de test correspond à la méthode de vérification des données, ainsi on vérifiera, critères par critères, que le retour est le bon si un d’entre eux n’est pas contenu dans les données. En se servant de la partie 6 de la fiche de test ainsi que de l’annexe, on obtient les résultats suivants :

*Test 3.1 : Champ manquant : Capteur*

« Champ manquant : Capteur  
Paquet MQTT rejeté après validation. »

*Test 3.2 : Champ manquant : TypeDeDonnee*

« Champ manquant : TypeDeDonnee  
Paquet MQTT rejeté après validation. »

*Test 3.3 : Champ manquant : Date*

« Champ manquant : Date  
Paquet MQTT rejeté après validation. »

*Test 3.4 : Champ manquant : Valeur*

« Champ manquant : Valeur  
Paquet MQTT rejeté après validation. »

Test 4 : Date au mauvais format

Ce test tiendra compte du format de la date. En effet même si les différents critères sont présents, le format est important, principalement celui de la date car il viendrait poser de multiples problèmes dans la sauvegarde et la recherche des données.   
Ainsi, en suivant les moyens à mettre en œuvre ainsi que la syntaxe de la fiche de test, on obtient le résultat suivant :

« Format de date invalide. Format attendu : 'AAAA-MM-JJ HH'  
Paquet MQTT rejeté après validation. »

Test 5 : Valeur non numérique

Ce test tiendra compte d’une autre erreur possible dans la réception des données. En effet, si la valeur envoyée par les capteurs n’est pas convertible en nombre (entier ou non), cela posera un problème dans l’enregistrement dans la base de données. C’est pour cela qu’on vérifie directement son format et sa compatibilité.  
Ainsi, le retour est le suivant :

« Valeur non numérique  
Paquet MQTT rejeté après validation. »

**5- Résultats attendus**

Test 1 : Instanciation d'un objet CAnalyse :   
« http://localhost:8080/SmartTerritories/addData »  
Ainsi on vérifie notre lien jusqu’à l’API.

Test 2 : Réception de données pour envoi :

Données reçues :   
« {  
 Capteur: 'Roseraie',  
 TypeDeDonnee: 'Temperature',  
 Date: '2025-05-19 15',  
 Valeur: 23.4  
} »

Test 3.1 : Champ manquant : Capteur :

« Champ manquant : Capteur   
Paquet rejeté après validation »

Test 3.2 : Champ manquant : TypeDeDonnee :

« Champ manquant : TypeDeDonnee  
Paquet rejeté après validation »

Test 3.3 : Champ manquant : Date :

« Champ manquant : Date  
Paquet rejeté après validation »

Test 3.4 : Champ manquant : Valeur :

« Champ manquant : Valeur  
Paquet rejeté après validation »

Test 4 : Date au mauvais format :

« Date au mauvais format. Utilisez : AAAA-MM-DD hh   
Paquet rejeté après validation »

Test 5 : Valeur non numérique :

« Valeur non numérique  
Paquet rejeté après validation »

**6- Moyens à mettre en œuvre**

Pour effectuer ces tests il faudra se servir de la machine virtuelle servant de test au code.

Il faudra se positionner dans le code test-CAnalyse.js (Dossier Documents/API-ProgObjet/Tests) via un terminal (commande « cd Documents/API-ProgObjet/Tests » pour changer de dossier et « nano test-CAnalyse.js » pour ouvrir le fichier) ou alors l’ouvrir par un explorateur de fichiers.

Pour le bon déroulement de ces tests, il faudra modifier le code en fonction de la fiche de test présente à la fin du fichier.

Il faudra mettre en commentaire ( « // ») les parties qui ne correspondent pas au code du test (voir les captures d’écran en annexe)

Pour démarrer le code, il faudra entrer, dans le terminal (connecté au répertoire de test) : « node test-CAnalyse.js »

Annexe : Fiche de test et programme modifié

Test 1 :



Test 2 :

Une image contenant texte, capture d’écran, document, Police

Le contenu généré par l’IA peut être incorrect.

Test 3.1 :

Une image contenant texte, capture d’écran, document, Police

Le contenu généré par l’IA peut être incorrect.

Test 3.2 :

Une image contenant texte, capture d’écran, Police, document

Le contenu généré par l’IA peut être incorrect.

Test 3.3 :

Une image contenant texte, capture d’écran, document, Police

Le contenu généré par l’IA peut être incorrect.

Test 3.4 :

Une image contenant texte, capture d’écran, document, Police

Le contenu généré par l’IA peut être incorrect.

Test 4 :

Une image contenant texte, capture d’écran, Police, document

Le contenu généré par l’IA peut être incorrect.

Test 5 :

Une image contenant texte, capture d’écran, Police, document

Le contenu généré par l’IA peut être incorrect.