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

**1- Identification du test**

Tests\_CAPI\_SmartTerritories

**2- Référence du module testé**

Ces tests seront effectués sur les méthodes de la classe CAPI. Ces méthodes permettent d’obtenir les données de la base en fonction des éléments présents dans la requête. Ces mêmes éléments seront vérifiés avant de recevoir les données. On peut aussi insérer de nouvelles données dans la base.

**3- Objectif du test**

Ce test vérifiera les différentes possibilités de requête de l’API pour vérifier du code d’erreur ainsi que le message qui lui est associé.

**4- Procédure du test**

*Test 1.1 : Requête renvoi de toutes les données*Ce test correspond à l’utilisation de la requête « allData » qui permets donc de retourner toutes les données de la base. Ainsi, après avoir utilisé la requête curl présente dans la fiche de test, en suivant la mise en œuvre, on obtient les données suivantes :

{  
 Capteur: "Roseraie",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-24 15",  
 Valeur: 18  
},  
{  
 Capteur: "Saint-Serge",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-28 08",  
 Valeur: 12  
},  
{  
 Capteur: "Saint-Serge",  
 TypeDeDonnee: "Humidite",  
 Date: "2025-05-26 09",  
 Valeur: 70  
},  
{  
 Capteur: "Monplaisir",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-26 09",  
 Valeur: 13  
}

*Test 1.2 : Requête de filtre vide*Ce test correspond à l’utilisation de la requête « dataFilter » ne comprenant aucun filtre. En suivant la mise en œuvre, on obtient le résultat suivant :

« Requête reçue : GET /SmartTerritories/dataFilter avec paramètres : {}  
Tentative de requête sans filtre. Utilisation de /allData recommandée. »

*Test 2.1 : Filtre par Capteur*La suite de test suivante correspond à un filtre par critère simple : ainsi on va chercher les données en entrant un seul critère à la fois. Ici on nous retourne les données suivantes pour le capteur Saint-Serge

{  
 Capteur: "Saint-Serge",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-28 08",  
 Valeur: 12  
},  
{  
 Capteur: "Saint-Serge",  
 TypeDeDonnee: "Humidite",  
 Date: "2025-05-26 09",  
 Valeur: 70  
}

*Test 2.2 : Filtre par TypeDeDonnee*Données retournées :

{  
 Capteur: "Roseraie",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-24 15",  
 Valeur: 18  
},  
{  
 Capteur: "Saint-Serge",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-28 08",  
 Valeur: 12  
},  
{  
 Capteur: "Monplaisir",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-26 09",  
 Valeur: 13  
}

*Test 2.3 : Filtre par Date*Données retournées :

{  
 Capteur: "Saint-Serge",  
 TypeDeDonnee: "Humidite",  
 Date: "2025-05-26 09",  
 Valeur: 70  
},  
{  
 Capteur: "Monplaisir",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-26 09",  
 Valeur: 13  
}

*Test 2.4 : Filtre avec plage de dates*Données retournées :

{  
 Capteur: "Saint-Serge",  
 TypeDeDonnee: "Humidite",  
 Date: "2025-05-26 09",  
 Valeur: 70  
},  
{  
 Capteur: "Monplaisir",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-26 09",  
 Valeur: 13  
}

*Test 2.5 : Filtre avec limite*

Données retournées :

{  
 Capteur: "Roseraie",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-24 15",  
 Valeur: 18  
},  
{  
 Capteur: "Saint-Serge",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-28 08",  
 Valeur: 12  
},  
{  
 Capteur: "Saint-Serge",  
 TypeDeDonnee: "Humidite",  
 Date: "2025-05-26 09",  
 Valeur: 70  
}

Test 3 : Valeur inconnue  
Ce test correspond à un retour de données dit « valide » mais qui renvoie à un filtre non connu par la base de données : on filtre par une valeur qui n’existe pas.   
Le message qui nous est retourné est donc :   
« Aucune donnée trouvée avec les critères fournis »

Test 4 : Mauvais format de date  
Le format de la date étant très stricte pour l’insertion dans la base, il faut être sur que l’utilisateur ne se trompe pas. Dans le cas d’une erreur on renvoit donc le format exact à suivre :

{"message":"Format de date invalide. Utiliser YYYY-MM-DD hh."}

Test 5 : Mauvais critère

Dans le cas où l’utilisateur se trompe de critère, que ce soit une faute de syntaxe ou une simple erreur, il faut renvoyer un avertissement. On retrouve donc le message suivant :

{"message":"Paramètres invalides détectés."}

Ainsi que du coté de l’API le message suivant :

« Paramètre(s) invalide(s) reçu(s) : NouveauCritère »

*Test 6.1 : Plusieurs dates de début*On laisse le choix à l’utilisateur d’afficher les données en fonction d’une plage de dates. Cependant on ne peut pas afficher de plage comprenant deux dates de début :

« Il ne peut pas y avoir plusieurs dates de début. »

*Test 6.2 : Plusieurs dates de fin*On vérifie maintenant le même scénario de test mais avec plusieurs dates de fin :

« Il ne peut pas y avoir plusieurs dates de fin. »

*Test 6.3 : Date de début/fin et date simple*Malgré le fait qu’on puisse entrer une plage de date ou une date seule, les deux combinés entraine une erreur. En effet la date seule serait ignorée et on retournerait seulement la plage de dates. Il nous faut donc un avertissement pour l’utilisateur :

"La date (simple) sera ignorée car vous avez entré une date de début et/ou de fin"

Test 7 : Insertion des données  
Ce test correspond à une insertion des données via le broker & client MQTT. En supposant que ces données seraient validées par la classe CAnalyse qui contient son propre plan de test. Ainsi on nous retourne bien la validation de l’insertion (les données ne sont pas inscrites dans la base car aucune connexion n’est mise en place).

"Insertion simulée réussie"

**5- Résultats attendus**

Test 1.1 : Requête renvoi de toutes les données  
On attend le code retour correspondant aux données en échantillon.

Test 1.2 : Requête de filtre vide  
On attend le code retour statut http 200 (success) avec les données correspondant à la requête

Test 2.1 : Filtre par Capteur  
On attend le retour des données qui comportent le capteur donné

Test 2.2 : Filtre par TypeDeDonnee  
On attend le retour des données qui comportent le TypeDeDonnee donné.

Test 2.3 : Filtre par Date  
On attend le retour des données qui comportent la date donnée.

Test 2.4 : Filtre avec plage de dates  
On attend le retour des données qui sont présentes dans la plage donnée.

Test 2.5 : Filtre avec Limite  
On attend le retour des données maximum correspondant à la limite. Si limite = 3 alors 3 données.

Test 3 : Valeur inconnue  
Message : « Aucune donnée trouvée avec ces critères »

Test 4 : Mauvais format de date  
Message « Format de date invalide. Utiliser YYYY-MM-DD hh. »

Test 5 : Mauvais critère  
Message « Paramètres invalides détectés. »

Test 6.1 : Plusieurs dates de début  
Message : « Il ne peut pas y avoir plusieurs dates de début ».

Test 6.2 : Plusieurs dates de fin  
Message : « Il ne peut pas y avoir plusieurs dates de fin ».

Test 6.3 : Date de début/fin et date simple  
Message : « La date (simple) sera ignorée car vous avez entré une date de début et/ou de fin »

Test 7 : Insertion des données  
Message : « Insertion simulée réussie » en renvoyant les données insérées.

**6- Moyens à mettre en œuvre**

Pour effectuer ces tests il faudra se servir de la machine virtuelle servant de test au code. Un échantillon de données sera inscrit sur une base de données de test, ainsi on pourra retourner les données voulues.

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/Tests » (cd Documents/API-ProgObjet) puis faire la commande : « node test-CAPI.js » pour lancer l’API. Ensuite on effectuera des requêtes http via curl (dans un autre terminal) pour récupérer les données renvoyées ainsi que les potentielles erreurs. On pourra rester dans le terminal où on a lancé le code (node) pour vérifier toutes les étapes.

A savoir : l’échantillon de données suivant sera utilisé pour simuler les retours :

{  
 Capteur: "Roseraie",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-24 15",  
 Valeur: 18  
},  
{  
 Capteur: "Saint-Serge",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-28 08",  
 Valeur: 12  
},  
{  
 Capteur: "Saint-Serge",  
 TypeDeDonnee: "Humidite",  
 Date: "2025-05-26 09",  
 Valeur: 70  
},  
{  
 Capteur: "Monplaisir",  
 TypeDeDonnee: "Temperature",  
 Date: "2025-05-26 09",  
 Valeur: 13  
}