

Tests automatisés avec FastAPI

C'est quoi ?

Les tests automatisés permettent de vérifier que ton API fonctionne comme prévu en simulant des requêtes HTTP sans avoir besoin de la lancer dans un navigateur ou via `curl`.

FastAPI intègre `TestClient`, un outil de test basé sur `requests`, utilisable avec `pytest`.

Pourquoi on l'utilise ?

- Pour vérifier que les routes répondent correctement (200, 401, 404...)
 - Pour tester les résultats retournés
 - Pour protéger contre les régressions (un changement qui casse un truc ailleurs)
 - Pour gagner du temps lors des mises à jour du code
-

Comment ça fonctionne ?

1. On crée un `TestClient` à partir de l'app FastAPI
 2. On simule des requêtes HTTP (GET, POST, etc.)
 3. On vérifie que la réponse a le bon statut, le bon contenu, etc.
-

Vocabulaire essentiel

- `` : framework de test Python
 - ` : client HTTP de test fourni par FastAPI (wrapper sur `requests`)
 - `` : mot-clé pour vérifier qu'une condition est vraie
 - `` : code HTTP (200, 404, etc.)
 - `` : fonction pour lire le contenu JSON d'une réponse
-

Syntaxe & construction

```
from fastapi.testclient import TestClient
from myApi import app # l'app FastAPI que tu as déjà codée

client = TestClient(app)

def test_get_machines():
    response = client.get("/machines")
    assert response.status_code == 200
    assert isinstance(response.json(), list)
```



Prérequis / Installation

```
pip install pytest
```



Présentation visuelle

Un fichier `test_api.py` dans ton projet. Chaque fonction commence par `test_`, et tu peux en créer autant que tu veux pour chaque route.

```
machineMonitor/  
├─ test_api.py  ← ton fichier de tests  
└─ main.py     ← ton API principale
```



Résultat attendu

Quand tu lances `pytest`, il exécute automatiquement tous les fichiers et fonctions de test et te dit ce qui passe  ou ce qui plante .

Tu es prêt à coder ton premier test !