

**Nom complet : Zemmouche Nora**

**Mini-projet : Suivi et analyse de la  
consommation énergétique**

# 1. Introduction

Ce mini-projet vise à simuler un dispositif de suivi et d'analyse de la consommation énergétique.

Il repose sur la programmation orientée objet (POO), la simulation de capteurs IoT, la détection d'anomalies et l'enregistrement des données dans une base MongoDB.

## Objectifs :

- Mettre en place une architecture POO.
- Simuler des capteurs IoT.
- Stocker les mesures dans MongoDB.
- Déetecter les anomalies de consommation.
- Assurer la qualité logicielle via Pytest et flake8.

## Tâches réalisées :

1. Simulation des capteurs (valeurs aléatoires)
2. Stockage des données dans MongoDB
3. Analyse énergétique et détection d'anomalies
4. Tests unitaires et qualité du code

# 2. Code source

## 2.1 src/sensor.py

```
import random

class Sensor:
    def __init__(self, name="Sensor"):
        self.name = name

    def read(self):
        # Simule une lecture aléatoire
        return random.randint(0, 100)
```

## 2.2 src/mongodb\_client.py

```
import mongomock

class MongoDBClient:
    def __init__(self, db_name="test_db"):
        self.client = mongomock.MongoClient()
        self.db = self.client[db_name]

    def insert(self, collection_name, document):
        return self.db[collection_name].insert_one(document)

    def find(self, collection_name, query):
        return list(self.db[collection_name].find(query))
```

## 2.3 src/anomaly\_detector.py

```
import statistics

class AnomalyDetector:
    def __init__(self, threshold=3.0):
        self.threshold = threshold

    def is_anomaly(self, value, history):
        if len(history) < 2:
            return False
        mean = statistics.mean(history)
        std = statistics.pstdev(history)
        if std == 0:
            return False
        z = abs((value - mean) / std)
        return z >= self.threshold

    def detect(self, data):
        results = []
        for i, value in enumerate(data):
            history = data[:i] + data[i+1:]
            results.append(self.is_anomaly(value, history))
        return results
```

## 2.4 src/main.py

```
from anomaly_detector import AnomalyDetector
from sensor import Sensor
from mongodb_client import MongoDBClient

if __name__ == "__main__":
    # Exemple simple
    detector = AnomalyDetector(threshold=2.5)
    sensor = Sensor("TempSensor")
    db = MongoDBClient("my_db")

    data = [sensor.read() for _ in range(5)]
    print("Données capteur :", data)

    anomalies = detector.detect(data)
    print("Anomalies détectées :", anomalies)

    for i, value in enumerate(data):
        db.insert("sensor_data", {"value": value, "anomaly": anomalies[i]})

    print("Données insérées dans la base MongoDB (mongomock).")
```

## 2.5 teste\_anomaly.py :

```
from src.anomaly_detector import AnomalyDetector

def test_anomaly():
    detector = AnomalyDetector(threshold=2.5)
    data = [10, 12, 11, 20, 13]
    result = detector.detect(data)
    assert all(isinstance(r, bool) for r in result)
```

## 2.6 test\_mongobd.py :

```
from src.mongodb_client import MongoDBClient

def test_insert_and_find():
    db = MongoDBClient("test_db")
    doc = {"name": "test", "value": 123}
    result = db.insert("test_collection", doc)
    assert result.inserted_id is not None

    found = db.find("test_collection", {"name": "test"})
    assert len(found) == 1
    assert found[0]["value"] == 123
f
```

## **2.7 test\_sonor.py :**

```
from src.sonor import Sensor

def test_sensor_read():
    sensor = Sensor()
    value = sensor.read()
    assert isinstance(value, int)
```

### **Configuration / Dépendances**

Pour exécuter ce projet, les bibliothèques Python suivantes doivent être installées :

- pymongo
- mongomock
- pytest
- flake8

Ces dépendances sont listées dans le fichier `requirements.txt` du projet.

## **3. Conclusion**

Le mini-projet a permis de :

- Simuler des capteurs IoT générant des mesures aléatoires.
- Stocker ces mesures dans une base MongoDB.
- Détecter les anomalies de consommation grâce à un seuil défini.

Ce projet montre comment Python et MongoDB peuvent être utilisés pour le suivi énergétique et la détection d'anomalies.

Il offre également une base pour étendre le système vers une vraie collecte de données IoT.