PROJET : Data-Profiler algorigramme de traitement de données IoT

Durée : 10 heures | Python IoT

Module : Développement avec les nouvelles technologies

CONTEXTE INDUSTRIEL

Dans les systèmes de décision automatisés, la performance est critique. L'objectif est de concevoir un outil d'**Observabilité Experte** capable d'ingérer 50 rapports d'exécution JSON bruts pour identifier les **hotspots** (goulots d'étranglement).
Vous devrez traiter des données réelles incluant des IDs de blocs, des liaisons et des mesures de latence pour fournir un diagnostic **visuel et chiffré**.

CHAINE DE VALEUR

`JSON Raw Logs` $\xrightarrow{Pandas}$ `Data Analytics` $\xrightarrow{Plotly}$ `Visualisation (Streamlit)`

1. INSTALLATION & SETUP (ENVIRONNEMENT DE TRAVAIL)

PRÉREQUIS & OUTILS

Objectif : standardiser l'environnement de développement afin de garantir que le projet est **reproductible** et **facile à corriger**.

- **Python (>3.10+)** via **Anaconda** .
- **IDE** : **PyCharm Community** (recommandé) ou VS Code.
- **Librairies** : `pandas`, `streamlit`, `plotly`.
- **Dataset** : un dossier contenant des runs JSON (`rule_engine_dataset/`.)

1.1 Étape 1 — Installer Conda (Anaconda / Miniconda)

À faire :

- Installer **Anaconda**.
- Vérifier l'installation dans un terminal.

```
conda --version
python --version
```

1.2 Étape 2 — Créer l'environnement Conda du projet

Créer un environnement dédié permet d'isoler les dépendances du projet.

```
# 1) Creer l'environnement
conda create -n graph-profiler python=3.11 -y

# 2) Activer l'environnement
conda activate graph-profiler
```

1.3 Étape 3 — Installer les packages Python

```
conda install -y pandas plotly
pip install streamlit
```

1.4 Étape 4 — Configurer PyCharm

PYCHARM : CONFIGURATION RAPIDE

1. En haut, à droite, cliquer sur "Current File".
2. Edit configurations.
3. Add new run configuration.
4. Python.
5. Cliquer sur "script" et choisir "module".
6. Dans le champ "module" saisir "streamlit".
7. Dans "script parameters" saisir "run <votrefichier>.py".

1.5 Étape 5 — Vérifications (sanity checks)

Test rapide : s'assurer que les imports fonctionnent.

```
python -c "import pandas as pd; import plotly; import streamlit; print('OK')
    "
```

Lancer Streamlit :

```
streamlit run <votrefichier>.py
```

DÉPANNAGE RAPIDE

— Si `streamlit` n'est pas trouvé : vérifier l'environnement actif (`conda activate graph-profiler`).
— Si conflit de versions : recréer l'environnement (`conda env remove -n graph-profiler`) puis recommencer.
— Si fichier JSON corrompu : votre code doit le gérer (try/except) sans arrêter l'analyse globale.

2. ARCHITECTURE DES DONNÉES (LE CONTRAT)

L'outil doit être capable de parser des fichiers respectant strictement la topologie suivante.

MODÈLE DE DONNÉES DE DEBUG (JSON)

Chaque run est identifié par un objet `run_info` (incluant le `graph_id` et le `timestamp`) et contient deux tableaux principaux.

1. execution_trace (Chronologie)
- `step` : Index séquentiel de l'exécution.
- `block_id` & `class` : Identification technique.
- `duration_ms` : Temps de calcul propre au bloc.
- `input / output` : Données (`raw` vs `processed`).
- `triggered_links` : Liste des liens activés.

2. link_data (Flux de données)
- `link_id` : Identifiant unique de la liaison.
- `traversed` : État d'activation (booléen).
- `value` : Donnée brute véhiculée sur le lien.

```
{
  "run_info": {
    "run_id": "RUN-100-f0eaa03c",
    "graph_id": "ENGINE_PROD_VOLUMES",
    "status": "SUCCESS",
    "total_ms": 5578,
    "timestamp": "2026-02-17T04:30:03.942383"
  },
  "execution_trace": [
    {
      "step": 1, "block_id": "B_START_01", "class": "ReadVar",
      "duration_ms": 17,
      "input": { "raw_value": 947.84 },
      "output": { "processed_value": 947.84 },
      "triggered_links": ["L_01_02"]
    }
  ],
  "link_data": [
    { "link_id": "L_01_02", "traversed": true, "value": 947.84 }
  ]
}
```

3. INSTRUCTIONS & PHASES DU PROJET

3.1 Phase 1 : Ingestion & Traitement (3h)

Objectif : Développer le moteur d'importation de données.

1. **Scan du dossier :** Parcourir automatiquement le répertoire contenant les 50 fichiers JSON.
2. **Data Wrangling :** Utiliser **Pandas** pour transformer ces fichiers en DataFrames.
3. **Nettoyage :** Gérer les valeurs aberrantes et les fichiers corrompus (gestion des exceptions).

3.2 Phase 2 : Analyse (3h)

Objectif : Extraire l'intelligence métier des données brutes.

1. **Profiling :** Calculer la durée moyenne et maximale par `block_id`.
2. **Détection d'anomalies :** Identifier les 3 blocs les plus lents statistiquement.

3.3 Phase 3 : Visualisation Interactive (4h)

Objectif : Créer l'interface de monitoring avec **Streamlit**.

1. **Dashboard :** Afficher les KPIs globaux (Taux de succès, Temps moyen total).
2. **Timeline (Gantt) :** Utiliser **Plotly** pour générer un graphique interactif de l'exécution.

Livrables attendus

— Dépôt Git avec code source Python structuré.
— Fichier `requirements.txt` (Streamlit, Pandas, Plotly).
— Dossier `rule_engine_dataset` contenant les 50 fichiers de tests.

4. GRILLE D'ÉVALUATION (BARÈME SUR 20)

Le projet sera évalué selon quatre axes majeurs lors d'une soutenance de 10 minutes.

Axe d'évaluation	Critères de performance	Points
Setup & Robustesse	• Présence du `requirements.txt` complet. • Ingestion sans crash des 50 fichiers (gestion des exceptions `try/except`).	/ 4
Analyse de Données	• Exactitude des calculs (Moyenne/Max) via les DataFrames **Pandas**. • Identification statistique précise du bloc "Bottleneck" (`B_AGGR_01`).	/ 6
Interface UX / UI	• Clarté du Dashboard **Streamlit** (Indicateurs clés et mise en page). • Interactivité : Filtrage par Run ID et mise à jour dynamique.	/ 5
Visualisation Experte	• Qualité de la Timeline **Plotly** (Lisibilité des étapes, couleurs). • Capacité à afficher le détail d'une liaison (`link_data`) au clic.	/ 5

5. MODALITÉS DE SOUTENANCE

La validation finale consiste en un "Stress Test" en direct :

1. **Démonstration du Dashboard :** Présentation des statistiques globales sur le dataset.
2. **Analyse d'un Run Critique :** Sélection d'un run avec le statut `ERROR` et explication de la défaillance via les données.
3. **Revue de Code :** Justification du choix des structures de données (dictionnaires vs DataFrames).

Livrables attendus

Rappel : Tout projet ne contenant pas de fichier `requirements.txt` ou ne parvenant pas à se lancer sur une machine vierge via `streamlit run` se verra appliquer une pénalité forfaitaire de **5 points**.