Wassim FAYALA
Adam HANANE
Martin PREVOT

Challenge DataLake

Présentation et objectifs du projet :

- Scrape des données emploi et compétences depuis plusieurs sources (Glassdoor, Adzuna, GitHub, Google Trends, etc.)

- Stocke ces données dans PostgreSQL via un pipeline raw_to_psql

- Expose une API REST (Django + DRF) pour consommer ces indicateurs

- **USE CASE :** Comparer la demande sur le marché des langages de programmation avec les tendances actuelles en Europe

Architecture du projet :

jobtech/

```
├── data/raw            # Données brutes
├── src/
│   ├── scrapers/      # Tous les scrapers (Glassdoor, Adzuna, etc.)
│   ├── cleaning/      # Nettoyage par source
│   └── raw_to_psql.py #Transfert des données de la raw vers la base PSQL en nettoyant les données
├── api/               # API Django REST Framework
│   ├── job_api/       # Projet Django principal
│   ├── api_core/      # App DRF centralisant les endpoints
│   └── manage.py
```

Workflow complet d'exécution

- Scrapers → data/raw/<source>/

- Cleaning → enrichissement et normalisation (exécution des fonctions dans raw_to_psql.py)

- raw_to_psql.py → insertion en base PostgreSQL

- API REST → exposition des données

Explication des fichiers principaux :

Fonctionnement global :

Chaque scraper est un script Python indépendant situé dans :

`src/scrapers/`

Lorsqu'un scraper est exécuté :
Il appelle une API publique (ou fait du scraping web).
Il télécharge les données pertinentes (offres d'emploi, salaires, tendances tech…).

Il enregistre les résultats en local :
JSON ou CSV
Dans un dossier spécifique à la source :

data/raw/<source_name>/

Pipeline raw_to_psql.py :

Ce fichier agit comme un **ETL (Extract, Transform, Load)** :

1. Lit toutes les données brutes dans `data/raw/`
2. Applique les scripts de nettoyage (dans `src/cleaning/`)
3. Charge le résultat dans PostgreSQL.

Étapes détaillées

1. Nettoyage spécial pour certaines sources

Pour certaines sources, il appelle directement les scripts de cleaning dans le dossier cleaning qui contient des fonctions de nettoyage/traitement pour les sources de données

2. Chargement dans PostgreSQL

Pour chaque source on :
Normalise le nom en table (ex: google trends → google_trends)
Parcourt tous les fichiers .csv et .json
Charge les données dans la table correspondante :

Explication générale de l'API

Le projet JobTech expose une API RESTful (read-only) construite avec Django 5 et Django REST Framework (DRF).
Cette API permet aux équipes externes de :

Consommer les indicateurs emploi et compétences.
Sans avoir besoin d'accéder directement à la base PostgreSQL.

Fonctionnalités principales :

- Exposer les données stockées dans PostgreSQL via des endpoints REST.
- Sécuriser l'accès grâce à une authentification par token.
- Pagination automatique (50 résultats par défaut, configurable).
- API read-only : aucune modification des données n'est autorisée.

Endpoints principaux :

Endpoint	Méthode	Description
/api/adzuna/median-salary/	GET	Médiane des salaires par pays
/api/glassdoor/competences/	GET	Liste des compétences et leur fréquence de mentions dans les offres
/api/google-trends/	GET	Données Google Trends (trends des langages par pays)
/api/google-trends/?geoName={pays}	GET	Données Google Trends filtrées par pays