Fonctionnement du projet et étapes de conception

Ce projet a été pensé comme une solution complète pour analyser le marché de l'emploi technologique en Europe. Il repose sur un pipeline de données automatisé, allant de la collecte brute jusqu'à l'exposition des résultats via une API. Cette architecture modulaire permet de croiser les données issues de différentes sources, d'en garantir la qualité, et de les rendre accessibles pour des analyses avancées.

Collecte multi-sources

La première étape du projet consiste à agréger des données provenant de plusieurs plateformes complémentaires. Des scripts Python ont été développés pour extraire des offres d'emploi depuis Adzuna, Indeed ou LinkedIn, mais aussi des tendances technologiques depuis GitHub et Google Trends, ainsi que des résultats d'enquêtes de développeurs comme le Stack Overflow Survey. Chaque source nécessite un traitement adapté : certains flux sont accessibles via une API, d'autres par scraping, et d'autres encore via le téléchargement de fichiers structurés.

Stockage brut dans un data lake local

Une fois collectées, les données sont stockées dans un dossier raw, sans transformation. Les fichiers sont enregistrés en format JSON ou CSV, selon leur origine. Ce choix permet de conserver une trace fidèle des données originales, garantissant une meilleure traçabilité et facilitant les étapes ultérieures de nettoyage ou de réexploitation.

Ingestion dans MongoDB

Les fichiers bruts sont ensuite injectés dans une base de données NoSQL, MongoDB. Grâce à sa structure flexible, cette base facilite l'exploration de jeux de données hétérogènes. Elle constitue également une zone tampon entre les données brutes et les étapes de transformation, rendant les requêtes exploratoires plus souples à ce stade.

Nettoyage et normalisation

Nous avons mis beaucoup d'importance sur la qualité des données. À partir de MongoDB, un script extrait les documents et applique différentes règles de nettoyage. Cela comprend l'unification des intitulés de compétences, la normalisation des salaires (conversion des devises, fréquence, plages de valeurs), le nettoyage des champs texte (offres d'emploi, descriptions), ou encore le formatage des dates. Les données ainsi transformées sont ensuite stockées dans un dossier intermédiaire datasets_clean.

Construction du Data Warehouse

Les données nettoyées sont importées dans un entrepôt de données relationnel construit avec SQLite. Ce data warehouse est organisé selon une structure en étoile, avec des tables de dimensions (pays, compétences, entreprises, sources, dates) et des

tables de faits (offres, tendances GitHub, tendances Google, réponses Stack Overflow). Ce modèle relationnel permet d'effectuer des analyses croisées efficaces et d'optimiser les performances des requêtes.

Exposition via une API Django

Afin de rendre les données accessibles et interrogeables, nous avons développé une API REST basée sur Django REST Framework. Cette API expose différents endpoints sécurisés par token, permettant d'accéder aux référentiels, aux offres d'emploi, aux tendances, ou encore à des statistiques agrégées.

Orchestration du pipeline

L'ensemble du pipeline a été conçu pour être facilement automatisable. Chaque étape — de la collecte au nettoyage, puis à l'insertion dans MongoDB ou SQLite — est orchestrée par des scripts shell. Ces scripts peuvent être planifiés via crontab (sous Linux) et services avec powershell afin d'assurer une exécution régulière (donc des mises a jour à jour des données) sans intervention manuelle. La modularité du pipeline permet également de relancer une étape sans affecter les autres.

Points clés de la conception

Ce projet s'appuie sur une architecture modulaire et scalable. Chaque composant du pipeline est indépendant, ce qui facilite la maintenance, les tests et les évolutions futures. On a apporté beaucoup d'importance à la qualité des données, avec l'application de règles de nettoyage précises et des contrôles rigoureux. l'architecture a été pensé pour pouvoir accueillir de nouvelles sources de données ou de nouveaux modules d'analyse. Enfin, une documentation complète, comprenant un fichier README et une interface Swagger, facilite la prise en main et l'utilisation de l'API.