PROJET FINAL NFE 101 2024 - 2025

Réalisé par : FONKOUO SANOU TEMENA LOIC

Introduction

Dans le cadre de l'Unité d'Enseignement **NFE 101**, portant sur l'intégration et l'évolution des systèmes d'information, nous avons été amenés à concevoir et développer un projet personnel basé sur les exemples étudiés en cours. Ce projet vise à mettre en pratique les concepts clés abordés, en explorant les capacités d'intégration et de traitement des flux de données en temps réel à l'aide de **Kafka**.

Pour répondre aux exigences du projet, nous avons développé plusieurs mini-applications Kafka, chacune jouant un rôle spécifique dans le traitement, la transformation ou l'analyse des données.

Afin de mieux situer le contexte, cette introduction s'articulera autour des éléments suivants :

1. **Description du jeu de données utilisé** : Nous présenterons en détail les données traitées, leur origine, ainsi que leur structure.
2. **Travail de prétraitement des données** : Nous expliquerons les étapes et les techniques employées pour préparer les données avant leur ingestion dans Kafka.
3. **Architecture de notre application :** Nous présenterons à l'aide d'un schéma l'architecture de nos applications, pour illustrer leur rôle.
4. **Description des mini-applications Kafka** : Nous décrirons le rôle et le fonctionnement de chaque application, en les illustrant par des captures d'écran et des explications techniques pertinentes.

Description du jeu de données

Le jeu de données intitulé **"Carte des pharmacies de Paris"** fournit une liste détaillée des pharmacies situées à Paris, incluant leurs coordonnées, adresses et informations de géolocalisation. Ces données proviennent du répertoire FINESS, avec une mise à jour datant de septembre 2013.

Le fichier contient 987 enregistrements, chacun représentant une pharmacie, avec les champs suivants :

Champ	Description
nofinesset	Numéro FINESS de l'établissement.
nofinessej	Numéro FINESS de l'entité juridique.
rs	Raison sociale de la pharmacie.
rslongue	Raison sociale longue.
compldistrib	Complément de distribution.
numvoie	Numéro de la voie.
typvoie	Type de voie (rue, avenue, boulevard, etc.).
voie	Libellé de la voie.
compvoie	Complément de voie.
lieuditbp	Lieu-dit ou Boîte Postale.
departement	Code du département.
libdepartement	Libellé du département.
cp	Code postal.
commune	Nom de la commune.
telephone	Numéro de téléphone.
telecopie	Numéro de télécopie.
dateouv	Date d'ouverture de la pharmacie.
dateautor	Date d'autorisation.
datemaj	Date de la dernière mise à jour des informations.
wgs84	Coordonnées géographiques au format WGS84 (latitude, longitude).
lat	Latitude.
lng	Longitude.

Figure 1: Description des colonnes du jeu de données.

Ce jeu de données est disponible aux formats CSV et JSON, facilitant ainsi son intégration et son exploitation dans divers systèmes d'information. Il est distribué sous la **Licence Ouverte / Open Licence version 2.0**, permettant une utilisation libre sous réserve de mentionner la source.

Travail de prétraitement des données

Pour garantir la qualité des données et leur intégration dans notre application, nous avons effectué un prétraitement minutieux en utilisant plusieurs outils adaptés. Voici les étapes réalisées :

- Correction des espaces et des colonnes mal renseignées

Nous avons utilisé OpenRefine pour détecter et corriger les espaces mal renseignés dans certaines colonnes, notamment celles contenant des données de type date. Cela a permis d'homogénéiser les formats de ces colonnes et de garantir une meilleure cohérence.

- 2. Gestion des caractères spéciaux liés à l'encodage

Certains caractères spéciaux provenant d'un problème d'encodage dans les données brutes ont été identifiés. OpenRefine a également été utilisé pour les corriger, en remplaçant ces caractères par des alternatives correctes.

- 3. Problèmes de décalage dans les colonnes

Le jeu de données contenait certaines lignes problématiques avec des virgules en surplus, ce qui entraînait un désalignement entre les colonnes et leurs en-têtes. Pour résoudre ce problème, nous avons utilisé Visual Studio Code pour rechercher rapidement les lignes contenant un nombre d'occurrences de virgules supérieur à celui attendu. Cela nous a permis de corriger ces lignes manuellement.

- Changement du séparateur de colonnes

Initialement, le fichier CSV utilisait le point-virgule (;) comme séparateur, ce qui a posé des problèmes lors de l'importation des données dans notre classe CsvReader. Nous avons tenté de changer le séparateur de

point-virgule à virgule (,), à l'aide d'Excel, mais cela n'a pas permis de résoudre le problème. Finalement, nous avons conservé le point-virgule comme séparateur et avons ajusté notre code en précisant explicitement ce séparateur lors de la lecture des données.

En résumé, bien que le jeu de données ne présentât pas de problèmes majeurs nécessitant un prétraitement exhaustif, ces étapes ont permis de garantir la qualité et la compatibilité des données pour leur exploitation dans notre application.

Architecture de notre application

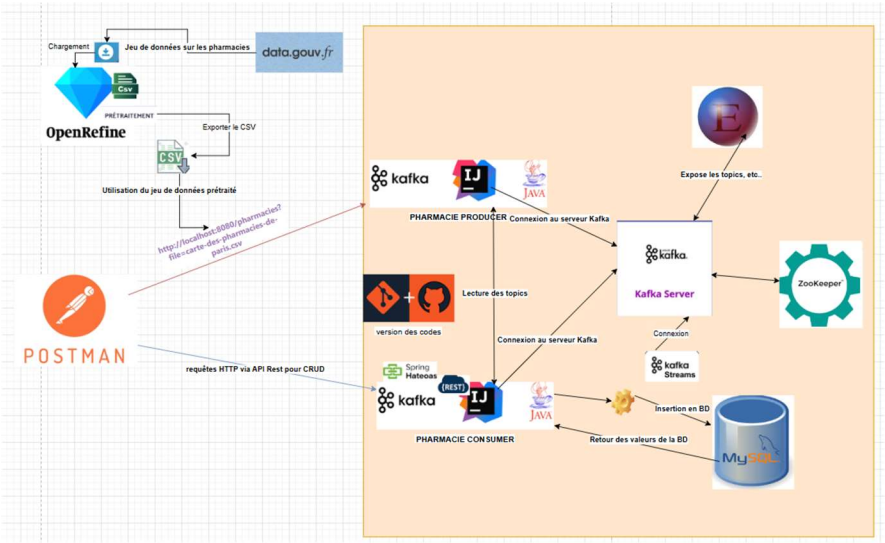

Figure 2: Architecture de notre application.

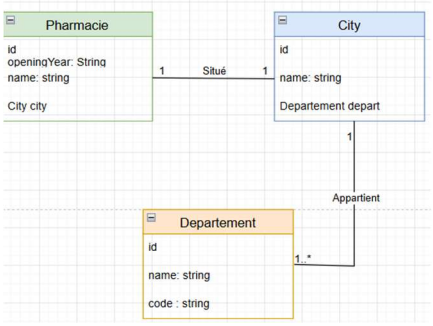

Figure 3: Diagramme de classe.

1. Obtention des Données

Les données brutes ont été obtenues à partir des topics Kafka où elles sont publiées.

2. Prétraitement et Structuration

Une fois les données récupérées, un processus de prétraitement est appliqué pour corriger, valider et structurer les informations (comme mentionné dans la section précédente). Cela garantit que seules des données propres et conformes aux exigences du système sont intégrées.

3. Mapping Objet-Relationnel (ORM)

Après le prétraitement, les données sont transformées en objets métier grâce à un processus de Mapping Objet-Relationnel (ORM). Ces objets sont ensuite reliés aux tables correspondantes de la base de données, suivant la structure définie dans le diagramme de classe présenté plus haut.

4. Stockage en Base de Données

Les objets mappés sont finalement persistés dans la base de données. Cette opération permet de stocker efficacement les informations pour une utilisation ultérieure dans d'autres parties de l'application (Ces données sont exposées sous forme d'API Rest par le Consumer).

Description des mini-applications Kafka

Pharmacies-Producer

Cette application joue le rôle de Producer dans notre architecture Kafka. Elle a pour mission de :

- Lire le fichier contenant les données nettoyées.
- Injecter chaque enregistrement de manière unitaire dans un topic Kafka spécifique.

Ce module constitue l'entrée principale du pipeline de traitement de données.

Pharmacies-Consumer

Cette application représente le Consumer principal et contient une grande partie de la logique métier. Ses principales responsabilités sont :

- Consommer les messages Kafka : Récupérer les enregistrements depuis le topic Kafka.
- Traitement des données : Effectuer les traitements nécessaires pour transformer les données en objets prêts à être persistés.
- Insertion dans la base de données : Stocker les données traitées dans une base MySQL en respectant le modèle relationnel défini.
- Exposition des données via API REST : Grâce à Spring HATEOAS, l'application fournit des endpoints CRUD (Create, Read, Update, Delete) pour interagir avec les données via des services RESTful.

Pharmacies-Streams

Initialement, les fonctionnalités de cette application étaient intégrées dans Pharmacies-Consumer. Cependant, pour réduire les couplages et améliorer la modularité, une application distincte a été mise en place.

- Cette application est dédiée à l'agrégation et à l'analyse des données en temps réel.
- Elle applique des opérations spécifiques (telles que le regroupement et le comptage) sur les messages provenant du topic Kafka.

Client Postman

Cet outil est utilisé pour tester et valider les endpoints REST exposés par l'application Pharmacies-Consumer. Il permet de simuler des appels aux API (GET, POST, PUT, DELETE) afin de s'assurer que les services RESTful fonctionnent correctement et répondent aux attentes.

Quelques captures d'écrans

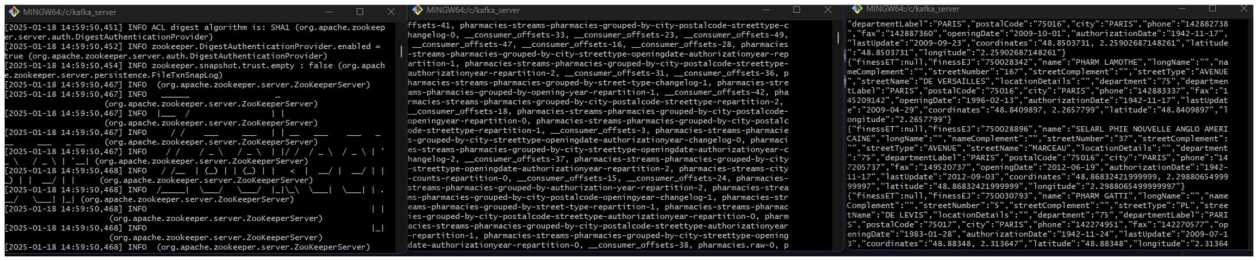

Figure 4: Server Kafka, Zookeeper, contenu des messages envoyés.

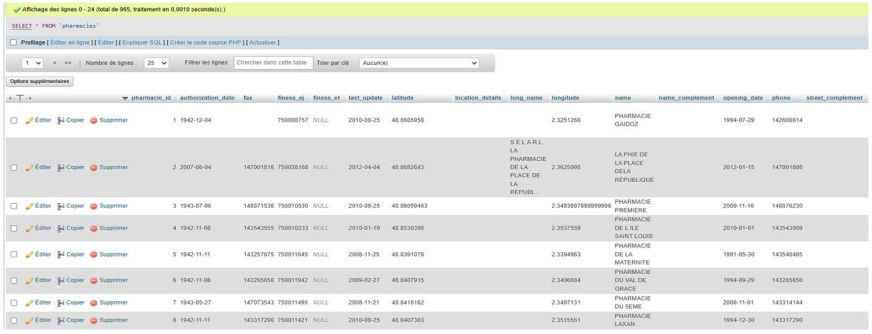

Figure 5: Données insérées en DB.

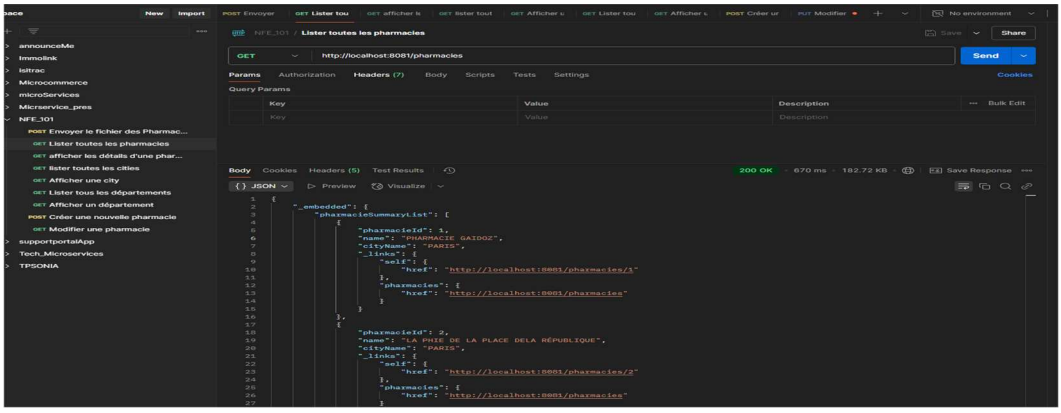

Figure 6: Liste non exhaustive des différents endpoints testés.

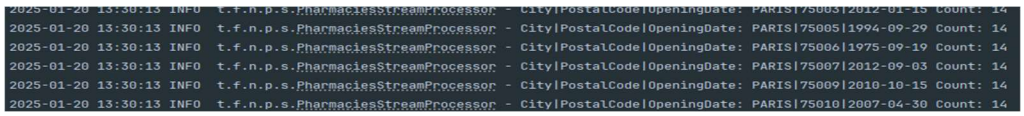

Figure 7: Exemple de groupement par City|Codepostal|OpeningDate.

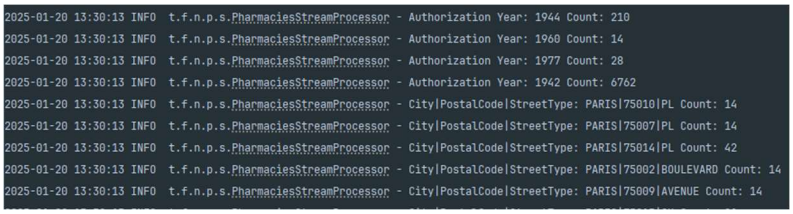

Figure 8: Groupement par Année d'autorisation, City|Codepostal|OpeningYear

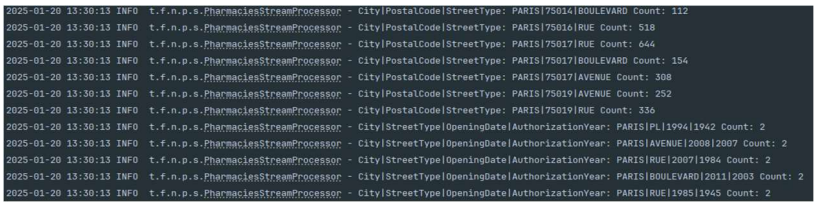

Figure 9; Groupement par City|PostalCode|StreetType et Groupement par City|StreetType|OpeningDate|AuthorizationYear

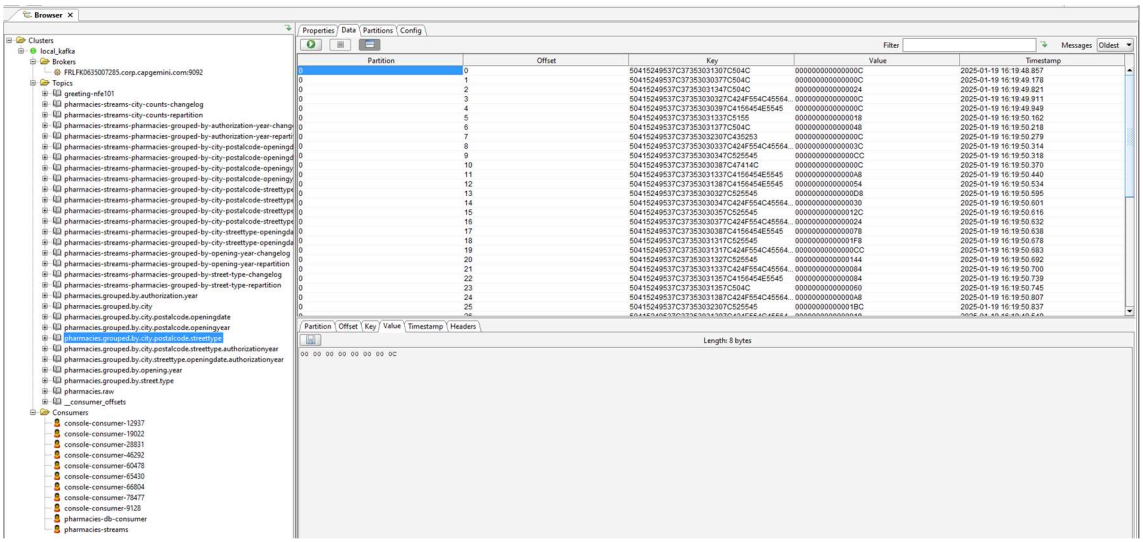

Figure 10: Affichage des Topics dans OffsetExplorer

Conclusion et Bibliographie

Ce projet illustre l'utilisation d'une architecture Kafka pour traiter, analyser et exposer des données. Grâce à une séparation des responsabilités entre les différentes applications (Producer, Consumer et Streams), chaque composant joue un rôle précis dans le pipeline. Le prétraitement des données, leur stockage dans une base MySQL et l'exposition via une API REST sont réalisés dans ce TP. Enfin, l'intégration d'outils comme Postman permet de tester les services développés.

https://www.data.gouv.fr/fr/datasets/carte-des-pharmacies-de-paris/

https://github.com/loicosquare/TP-NFE-101.git

https://github.com/features/copilot