|  |  |
| --- | --- |
| **RAPPORT DESCRIPTIF DU PROJET** | **Auditeur : GAVI Holali David** |

[1. Contexte et objectif 1](#_Toc188296090)

[2. Données Open Data 1](#_Toc188296091)

[2.1. Source des données 1](#_Toc188296092)

[2.2. Nettoyage et préparation des données 2](#_Toc188296093)

[3. Service Producteur Kafka 2](#_Toc188296094)

[4. Service Consommateur Kafka 2](#_Toc188296095)

[4.1. Fonctionnement 2](#_Toc188296096)

[4.2. Gestion des bases de données 3](#_Toc188296097)

[4.3. API RESTful 3](#_Toc188296098)

[5. Service Kafka Streams 3](#_Toc188296099)

[5.1. Fonctionnalités 3](#_Toc188296100)

[5.2. Technologies utilisées 4](#_Toc188296101)

[6. Application Flutter 4](#_Toc188296102)

[6.1. Fonctionnalités 4](#_Toc188296103)

[6.2. Technologies utilisées 4](#_Toc188296104)

[7. Architecture Globale 4](#_Toc188296105)

[Conclusion 5](#_Toc188296106)

[Annexes 6](#annexe)

**Contexte et objectif**

Ce projet vise à mettre en place un pipeline complet pour la gestion, le traitement, la transformation et la visualisation de données issues de l'**Open Data**. Les données utilisées proviennent du site [data.gouv.fr](https://www.data.gouv.fr), et concernent les **stations de taxi**. Le projet repose sur une architecture distribuée utilisant Kafka, des services Spring Boot, un système de base de données, et une application Flutter pour la visualisation sur une carte Google Maps.

# **Données Open Data**

## **Source des données**

Les données brutes ont été récupérées depuis [data.gouv.fr](https://www.data.gouv.fr), une plateforme fournissant des jeux de données publiques en France. Ces données contiennent des informations sur les stations de taxi, notamment :

* Identifiant de la station.
* Nom de la station.
* Adresse.
* Code INSEE.
* Nombre d’emplacements.
* Latitude et longitude.
* Statut de la station (active ou inactive).

## **Nettoyage et préparation des données**

Avant leur ingestion dans le pipeline, les données ont été nettoyées et préparées à l'aide d'**OpenRefine** :

* **Correction des erreurs** dans les champs (orthographe, format des adresses, etc.).
* **Suppression des doublons** pour éviter les redondances.
* **Validation des coordonnées GPS** pour garantir une précision sur la carte.
* **Exportation** du fichier final au format **CSV**.

# **Service Producteur Kafka**

Le **service producteur Kafka** a pour rôle de lire les données nettoyées (au format CSV) et de les envoyer dans un **topic Kafka** nommé taxi-stations.raw. Voici les étapes principales :

1. **Lecture des Données** : Utilisation de la bibliothèque OpenCSV pour parser le fichier CSV.
2. **Transformation en JSON** : Conversion de chaque ligne du CSV en objet JSON.
3. **Envoi vers Kafka** : Les messages JSON sont envoyés au topic taxi-stations.raw via un producteur Kafka.

* **Technologies Utilisées**
* Spring Boot.
* Kafka Producer API.
* OpenCSV pour le parsing du fichier.

# **Service Consommateur Kafka**

Le **service consommateur Kafka** est un composant clé qui consomme les messages JSON envoyés par le producteur et les stocke dans une base de données pour une utilisation ultérieure.

## **Fonctionnement**

1. **Consommation Kafka** : Lecture des messages JSON depuis le topic taxi-stations.raw.
2. **Stockage en base de bonnées** :
   * **MySQL** est utilisé comme base de données principale.
   * En cas d'indisponibilité de MySQL, le service bascule automatiquement vers une base **H2** persistante grâce à une logique dans l'application.
3. **Exposition des Données** :
   * Une API RESTful expose les données stockées dans la base.

## **Gestion des bases de données**

1. **Configuration Multiple** :
   * Un fichier application.properties global est utilisé par défaut.
   * Des fichiers spécifiques (application-mysql.properties et application-h2.properties) configurent respectivement MySQL et H2.
2. **Détection Automatique** :
   * Le service teste la disponibilité de MySQL au démarrage.
   * Si MySQL est indisponible, il bascule automatiquement sur H2.

## **API RESTful**

L’API RESTful permet :

* **Récupération de toutes les stations**.
* **Recherche par identifiant**.
* **Pagination des résultats** pour optimiser les requêtes.

# **Service Kafka Streams**

Le **service Kafka Streams** est un composant autonome qui traite les données brutes issues du topic taxi-stations.raw. Son rôle est de transformer, regrouper et enrichir les données avant de les produire dans de nouveaux topics Kafka.

## **Fonctionnalités**

1. **Regroupement par Statut** :
   * Compte le nombre de stations en fonction de leur statut (active ou inactive).
   * Produit les résultats dans le topic stations.grouped.by.status.
2. **Filtrage pour Paris** :
   * Filtre les stations situées à Paris à partir des champs address et latitude/longitude.
   * Produit les résultats filtrés dans le topic paris.stations.
3. **Création automatique des topics** :
   * Les topics nécessaires (taxi-stations.raw, stations.grouped.by.status, paris.stations) sont créés dynamiquement au démarrage grâce à l’API Kafka Admin.

## **Technologies utilisées**

* Kafka Streams API.
* Jackson pour le parsing JSON.
* Kafka AdminClient pour la gestion des topics.

# **Application Flutter**

Une **application mobile Flutter** a été développée pour visualiser les données exposées par le consommateur via son API RESTful. Cette application affiche les stations de taxi sur une carte interactive.

## **Fonctionnalités**

1. **Récupération des Données** :
   * L’application consomme les endpoints de l’API pour récupérer les données des stations.
2. **Affichage sur Google Maps** :
   * Les stations sont affichées sur une carte Google Maps avec des marqueurs interactifs.
3. **Recherche** :
   * L’utilisateur peut rechercher une station spécifique par son nom ou son adresse.
4. **Navigation** :
   * Possibilité d’obtenir des itinéraires vers une station sélectionnée.

## **Technologies utilisées**

* Flutter pour le développement mobile.
* Google Maps API pour l’intégration cartographique.
* http pour consommer les endpoints REST.

# **Architecture Globale**

**Schéma :**

* **Producteur Kafka** : Topic d'entrée : taxi-stations.raw.
* **Kafka Streams** : Lit taxi-stations.raw et produit dans :
  + - stations.grouped.by.status (regroupement par statut).
    - paris.stations (stations filtrées pour Paris).
* **Consommateur Kafka** :
  + Lit stations.grouped.by.status et paris.stations.
  + Stocke dans MySQL (ou H2 si indisponible).
  + Expose une API RESTful.
* **Application Flutter** :
  + Consomme l’API pour afficher les données sur Google Maps.

# **Conclusion**

Ce projet illustre un pipeline complet, allant de l'ingestion et de la transformation des données brutes à leur visualisation interactive. Les principaux atouts du projet incluent :

* **Utilisation de données publiques Open Data**.
* **Architecture distribuée** basée sur Kafka.
* **Gestion de la persistance** avec basculement automatique entre MySQL et H2.
* **Flexibilité et scalabilité** grâce à Kafka Streams.
* **Accessibilité** via une application mobile intuitive.

Le projet démontre une intégration réussie de technologies modernes et répond efficacement aux besoins de gestion et de visualisation de données en temps réel.

**Une image contenant texte, capture d’écran, Police, noir et blanc

Description générée automatiquementAnnexe 1 : Console de démarrage du serveur Kafka, Zookeeper, consumer console**

**Une image contenant texte, capture d’écran, Police

Description générée automatiquementUne image contenant texte, capture d’écran, menu, noir et blanc

Description générée automatiquement**

**Une image contenant texte, logiciel, Icône d’ordinateur, Page web

Description générée automatiquementUne image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquementAnnexe 2 : OpenRefine**

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement**Annexe 3 : Requête API via Postman**

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement **Annexe 4 : Service Kafka Producer**

Une image contenant capture d’écran, logiciel, Logiciel multimédia, Appareils électroniques

Description générée automatiquement**Annexe 5 : Service Kafka consumer and** **API**

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement**Annexe 6 : Application flutter (extrait de codes)**

**Annexe 7: Interface UI de l’application mobile flutter**

