

1. Contexte et objectif .....	1
2. Données Open Data.....	1
2.1. Source des données .....	1
2.2. Nettoyage et préparation des données .....	2
3. Service Producteur Kafka.....	2
4. Service Consommateur Kafka .....	2
4.1. Fonctionnement.....	2
4.2. Gestion des bases de données .....	3
4.3. API RESTful .....	3
5. Service Kafka Streams .....	3
5.1. Fonctionnalités .....	3
5.2. Technologies utilisées .....	4
6. Application Flutter.....	4
6.1. Fonctionnalités .....	4
6.2. Technologies utilisées .....	4
7. Architecture Globale .....	4
Conclusion .....	5
Annexes.....	6

## 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://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.

## 1. Données Open Data

### 1.1. Source des données

Les données brutes ont été récupérées depuis [data.gouv.fr](https://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).

## 1.2. 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**.

## 2. 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.

## 3. 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.

### 3.1. Fonctionnement

1. **Consommation Kafka** : Lecture des messages JSON depuis le topic taxi-stations.raw.

## 2. **Stockage en base de données :**

- **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.

### 3.2. **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.

### 3.3. **API RESTful**

L'API RESTful permet :

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

## 4. **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.

### 4.1. **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.

### 4.2. Technologies utilisées

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

## 5. 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.

### 5.1. 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.

### 5.2. Technologies utilisées

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

## 6. 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.

## Annexe 1 : Console de démarrage du serveur Kafka, Zookeeper, consumer console

```
c:\kafka>.\bin\windows\kafka-server-start.bat .\config\server.properties
[2025-01-19 09:07:54,110] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2025-01-19 09:07:54,282] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2025-01-19 09:07:54,361] INFO starting (kafka.server.KafkaServer)
[2025-01-19 09:07:54,361] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2025-01-19 09:07:54,383] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2025-01-19 09:07:59,364] INFO Client environment:zookeeper.version=3.8.4-9316c2a7a97e1666d8f4593f34dd6fc36ecc436c, built on 2024-02-12 22:16 UTC (org.apache.zookeeper.ZooKeeper)
[2025-01-19 09:07:59,365] INFO Client environment:host.name=SAVED (org.apache.zookeeper.ZooKeeper)
[2025-01-19 09:07:59,365] INFO Client environment:java.version=23.0.1 (org.apache.zookeeper.ZooKeeper)
[2025-01-19 09:07:59,365] INFO Client environment:java.vendor=Oracle Corporation (org.apache.zookeeper.ZooKeeper)
[2025-01-19 09:07:59,365] INFO Client environment:java.home=C:\Program Files\Java\jdk-23 (org.apache.zookeeper.ZooKeeper)
[2025-01-19 09:07:59,365] INFO Client environment:java.class.path=c:\kafka\libs\activation-1.1.1.jar;c:\kafka\libs\aoaliance-repackaged-2.6.1.jar;c:\kafka\libs\argparse4j-0.7.0.jar;c:\kafka\libs\audience-annotations-0.12.0.jar;c:\kafka\libs\caffeine-2.9.3.jar;c:\kafka\libs\commons-beanutils-1.9.4.jar;c:\kafka\libs\commons-cli-1.4.jar;c:\kafka\libs\commons-collections-3.2.2.jar;c:\kafka\libs\commons-digester-2.1.jar;c:\kafka\libs\commons-io-2.14.0.jar;c:\kafka\libs\commons-lang3-3.12.0.jar;c:\kafka\libs\commons-logging-1.2.jar;c:\kafka\libs\commons-validator-1.7.jar;c:\kafka\libs\connect-api-3.9.0.jar;c:\kafka\libs\connect-basic-auth-extension-3.9.0.jar;c:\kafka\libs\connect-file-3.9.0.jar;c:\kafka\libs\connect-jackson-3.9.0.jar;c:\kafka\libs\connect-mirror-3.9.0.jar;c:\kafka\libs\connect-mirror-client-3.9.0.jar;c:\kafka\libs\connect-runtime-3.9.0.jar;c:\kafka\libs\connect-transforms-3.9.0.jar;c:\kafka\libs\error_prone_annotations-2.10.0.jar;c:\kafka\libs\hk2-api-2.6.1.jar;c:\kafka\libs\hk2-locator-2.6.1.jar;c:\kafka\libs\hk2-utils-2.6.1.jar;c:\kafka\libs\jackson-an
```

```
der)
[2025-01-19 09:07:35,417] INFO zookeeper.DigestAuthenticationProvider.enabled = true (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2025-01-19 09:07:35,419] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapshotLog)
[2025-01-19 09:07:35,427] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2025-01-19 09:07:35,427] INFO ----- (org.apache.zookeeper.server.ZooKeeperServer)
[2025-01-19 09:07:35,427] INFO |___ / | | (org.apache.zookeeper.server.ZooKeeperServer)
[2025-01-19 09:07:35,427] INFO // ___ | | __ ___ ___ ___ ___ ___ (org.apache.zookeeper.server.ZooKeeperServer)
[2025-01-19 09:07:35,428] INFO // / \ / \ | / / / \ / \ | ' \ / \ | '___ (org.apache.zookeeper.server.ZooKeeperServer)
[2025-01-19 09:07:35,428] INFO / __ | ( ) | | ( ) | < | __ | __ | | | | | | (org.apache.zookeeper.server.ZooKeeperServer)
[2025-01-19 09:07:35,428] INFO /____| \___/ \___/ | \ \ \___/ \___/ | | | | | | (org.apache.zookeeper.server.ZooKeeperServer)
[2025-01-19 09:07:35,428] INFO | | (org.apache.zookeeper.server.ZooKeeperServer)
[2025-01-19 09:07:35,428] INFO |_ | (org.apache.zookeeper.server.ZooKeeperServer)
[2025-01-19 09:07:35,428] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2025-01-19 09:07:40,377] INFO Server environment:zookeeper.version=3.8.4-9316c2a7a97e1666d8f4593f34dd6fc36ecc436c, built on 2024-02-12 22:16 UTC (org.apache.zookeeper.server.ZooKeeperServer)
[2025-01-19 09:07:40,377] INFO Server environment:host.name=SAVED (org.apache.zookeeper.server.ZooKeeperServer)
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Installez la dernière version de PowerShell pour de nouvelles fonctionnalités et améliorations ! https://aka.ms/PSWindows

PS C:\Users\DAVS-LAB> cd C:\kafka\
PS C:\kafka> .\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic taxi-stations.raw
{"id": "75119-T-484", "name": "Tanger", "insee": "75119", "address": "16T RUE DE TANGER", "emplacements": "6.0", "status": "en service", "latitude": 48.88618860688236, "longitude": 2.3689846538380746}
{"id": "75110-T-337", "name": "Haussmann", "insee": "75110", "address": "21 RUE D'ALSACE", "emplacements": "5.0", "status": "en service", "latitude": 48.87739249960617, "longitude": 2.357864242070205}
{"id": "75108-T-298", "name": "Champs Elysées / Bassano", "insee": "75108", "address": "103 AVENUE DES CHAMPS ELYSEES", "emplacements": "8.0", "status": "en service", "latitude": 48.87181698138564, "longitude": 2.309063551913162}
{"id": "75109-T-307", "name": "Haussmann / Taitbout", "insee": "75109", "address": "14 BOULEVARD HAUSSMANN", "emplacements": "4.0", "status": "en service", "latitude": 48.872540342976826, "longitude": 2.3372280997158885}
{"id": "75116-T-428", "name": "Poincaré / Lauriston", "insee": "75116", "address": "40 AVENUE RAYMOND POINCARÉ", "emplacements": "5.0", "status": "en service", "latitude": 48.866028098804327, "longitude": 2.28644253409334}
{"id": "75108-T-280", "name": "Roosevelt / Théâtre du Rond Point", "insee": "75108", "address": "43 AVENUE FRANKLIN D. ROOSEVELT", "emplacements": "6.0", "status": "en service", "latitude": 48.86822841945435, "longitude": 2.309953980037344}
{"id": "75111-T-348", "name": "Richard Lenoir / Amelot", "insee": "75111", "address": "1 BOULEVARD RICHARD LENOIR", "emplacements": "5.0", "status": "en service", "latitude": 48.85430347141947, "longitude": 2.3692108009535335}
{"id": "75112-T-901", "name": "Gare de Lyon SNCF", "insee": "75112", "address": "8 PLACE LOUIS ARMAND", "emplacements": "103.0", "status": "en service", "latitude": 48.8451106231118, "longitude": 2.373436105827891}
{"id": "75117-T-457", "name": "Rome / Legendre", "insee": "75117", "address": "135 RUE DE ROME", "emplacements": "2.0", "status": "en service", "latitude": 48.885707402091235, "longitude": 2.3166163174391654}
{"id": "75117-T-458", "name": "Mac-Mahon / Troyon", "insee": "75117", "address": "21 AVENUE MAC-MAHON", "emplacements": "6.0", "status": "en service", "latitude": 48.8769281012419, "longitude": 2.2942420413831746}
{"id": "75117-T-460", "name": "Pershing / Koenig", "insee": "75117", "address": "22 BOULEVARD PERSHING", "emplacements": "14.0", "status": "en service", "latitude": 48.88006722507015, "longitude": 2.2826396104296}
{"id": "75119-T-482", "name": "Petit-Meaux", "insee": "75119", "address": "121 RUE DE MEAUX", "emplacements": "8.0", "status": "en service", "latitude": 48.884585912691115, "longitude": 2.3788347261473257}
{"id": "75115-T-387", "name": "Quai de Grenelle / Novotel", "insee": "75115", "address": "61 QUAI DE GRENNELLE", "emplacements": "8.0", "status": "en service", "latitude": 48.84992294166981, "longitude": 2.2829801567496987}
{"id": "75101-T-211", "name": "Marengo", "insee": "75101", "address": "1 RUE DE MARENGO", "emplacements": "3.0", "status": "en service", "latitude": 48.86184837657944, "longitude": 2.339128927086149}
{"id": "75102-T-214", "name": "Capucines/Louis le Grand", "insee": "75102", "address": "1 BOULEVARD DES CAPUCINES", "emplacements": "4.0", "status": "en service", "latitude": 48.87076285656672, "longitude": 2.3339630975787142}
{"id": "75111-T-843", "name": "Saint Ambroise", "insee": "75111", "address": "57 BOULEVARD VOLTAIRE", "emplacements": "5.0", "status": "en service", "latitude": 48.862002676608256, "longitude": 2.373391871065638}
{"id": "75117-T-082", "name": "Etoile - Wagram", "insee": "75117", "address": "67 Boulevard Pereire", "emplacements": "10.0", "status": "en service", "latitude": 48.88717567894479, "longitude": 2.308386274370729}
{"id": "75115-T-070", "name": "Place Balard", "insee": "75115", "address": "3 PLACE BALARD", "emplacements": "3.0", "status": "en service", "latitude": 48.83691440100358, "longitude": 2.279066978305342}
{"id": "75111-T-038", "name": "Nation", "insee": "75111", "address": "3 AVENUE DU TRÉNE", "emplacements": "14.0", "status": "en service", "latitude": 48.84860852998622, "longitude": 2.3978274533514643}
{"id": "75117-T-090", "name": "Porte de Saint Ouen", "insee": "75117", "address": "3 AVENUE DE LA PORTE DE SAINT OUEN", "emplacements": "7.0", "status": "en service", "latitude": 48.89828666841568, "longitude": 2.3287853028781047}
{"id": "75109-T-320", "name": "Condorcet", "insee": "75109", "address": "69 RUE CONDORCET", "emplacements": "5.0", "status": "en service", "latitude": 48.88032485422117, "longitude": 2.340669811693258}
```

# Annexe 2 : OpenRefine

OpenRefine **bornes dappel taxi csv** Permalink

Open... Export ▾ Help

Facet / Filter Undo / Redo ↺ / ↻

419 rows

Show as: rows records Show: 5 10 25 50 100 500 1000 rows

Extensions Wikibase ▾

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

Not sure how to get started?  
[Watch these screencasts](#)

All	id	nom	insee	adresse	emplacements	statut	latitude	longitude
1	75119-T-484	Tanger	75119	167 RUE DE TANGER	6.0	en service	48.886188600688236	2.368964538300746
2	75110-T-337	Alsace	75110	21 RUE D'ALSACE	5.0	en service	48.87739249900617	2.357864242070205
3	75108-T-298	Champs Elysées / Bessano	75108	103 AVENUE DES CHAMPS ELYSEES	8.0	en service	48.87181688138564	2.300033551913162
4	75109-T-307	Hausmann / Talibout	75109	14 BOULEVARD HAUSSMANN	4.0	en service	48.872540342976826	2.3372200997150085
5	75116-T-429	Poincaré / Lauriston	75116	40 AVENUE RAYMOND POINCARE	5.0	en service	48.8600280804327	2.2864425340334
6	75108-T-280	Roosevelt / Théâtre du Rond Point	75108	43 AVENUE FRANKLIN D. ROOSEVELT	6.0	en service	48.86822841945435	2.309953980037344
7	75111-T-348	Richard Lenoir / Arnelot	75111	1 BOULEVARD RICHARD LENOIR	5.0	en service	48.85430347141947	2.3882108009535335
8	75112-T-901	Gare de Lyon SNCF	75112	8 PLACE LOUIS ARMAND	103.0	en service	48.8451106231118	2.373436105827891
9	75117-T-457	Rome / Legendre	75117	135 RUE DE ROME	2.0	en service	48.885707402091235	2.3166163174391654
10	75117-T-458	Mac-Mahon / Troyon	75117	21 AVENUE MAC-MAHON	6.0	en service	48.8769281012419	2.2842424143831746
11	75117-T-460	Pershing / Kosig	75117	22 BOULEVARD PERSHING	14.0	en service	48.88006722507015	2.28262950142426
12	75119-T-482	Petit-Moex	75119	121 RUE DE MARENGO	8.0	en service	48.884585912691115	2.3789347261473257
13	75115-T-387	Quai de Grenelle / Novotel	75115	61 QUAI DE GRENELLE	8.0	en service	48.84992294160681	2.2829001567496887
14	75111-T-211	Marengo	75111	1 RUE DE MARENGO	3.0	en service	48.86184837657944	2.338128927086149
15	75102-T-214	Capucines/Louis le Grand	75102	1 BOULEVARD DES CAPUCINES	4.0	en service	48.87076285656672	2.3339630975787142
16	75111-T-043	Saint-Ambrose	75111	57 BOULEVARD VOLTAIRE	5.0	en service	48.862002676008256	2.373391871005638
17	75117-T-082	Etoile - Wagram	75117	67 Boulevard Pereire	10.0	en service	48.88717567894479	2.305386274370729
18	75115-T-070	Place Balard	75115	3 PLACE BALARD	3.0	en service	48.83691440100358	2.279086918305342
19	75111-T-038	Nation	75111	3 AVENUE DU TRÔNE	14.0	en service	48.84860852998622	2.3978274533514643
20	75117-T-090	Porte de Saint Ouen	75117	3 AVENUE DE LA PORTE DE SAINT OUEN	7.0	en service	48.898286066841568	2.3287853028781047
21	75109-T-320	Condorcet	75109	69 RUE CONDORCET	5.0	en service	48.88032485422117	2.34069891693258
22	75119-T-225	Saint-Martin-République	75103	1 BOULEVARD SAINT-MARTIN	7.0	en service	48.86793297232912	2.3614072541382307
23	75120-T-489	Bagnolet / Pyrénées	75120	96 RUE DE BAGNOLET	3.0	en service	48.85024282040246	2.402474660388745
24	75119-T-104	Flandre - Stalingrad	75119	15 AVENUE DE FLANDRE	6.0	en service	48.85051311054795	2.3693014710783595
25	75104-T-119	Hôtel de Ville	75104	4 RUE DE LOBAU	5.0	en service	48.85646337472804	2.3537438183255324
26	75101-T-203	Rivoli/Rouget de l'Isle	75101	238 RUE DE RIVOLI	3.0	en service	48.86576104505841	2.282388878036847
27	75101-T-205	Juliet	75101	208 RUE DE RIVOLI	4.0	en service	48.86440055732531	2.3033592437040017
28	75115-T-408	Florian Delbarre / France TV	75115	28 RUE DU PROFESSEUR FLORIAN DELBARRE	4.0	en service	48.83834048160493	2.271871132095047
29	75116-T-409	Mina / Place Armand de Grasse	75116	33 AVENUE DIENA	6.0	en service	48.86513546830559	2.284949400386159
30	75116-T-411	Kleber / Dossière	75116	61 AVENUE KLEBER	12.0	en service	48.86662774435086	2.29013171815591
31	75101-T-208	Place du Palais Royal	75101	1 PLACE DU PALAIS ROYAL	6.0	en service	48.86273666697366	2.336346067792886
32	75116-T-425	Georges Lafont	75116	104 AVENUE GEORGES LAFONT	1.0	en service	48.83724841631142	2.25898848625146
33	75111-T-350	Charonne / Ledru-Rollin	75111	47 RUE DE CHARONNE	6.0	en service	48.85337878740128	2.317680896432455
34	75114-T-375	Saint-Jacques / Fernus	75114	17 BOULEVARD SAINT-JACQUES	7.0	en service	48.83189469198536	2.30904325340362
35	75114-T-382	Diderot	75114	113 RUE DIDOT	3.0	en service	48.82841623040249	2.315171456846368
36	75116-T-079	Porte d'Auteuil	75116	136 BOULEVARD EXELMANS	14.0	en service	48.847761482250135	2.25888784092498

OpenRefine **bornes dappel taxi csv** Permalink

Open... Export ▾ Help

Facet / Filter Undo / Redo ↺ / ↻

419 rows

Show as: rows records Show: 5 10 25 50 100 500 1000 rows

Extensions Wikibase ▾

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

Not sure how to get started?  
[Watch these screencasts](#)

Export custom tabular

Content Download Upload Option Code

Line-based text formats

☐ Tab-separated values (TSV)

☒ Comma-separated values (CSV)

☐ Custom separator

Line separator

Character encoding

Always quote text ☐

Other formats

☐ Excel (.xls)

☐ Excel in XML (.xlsx)

☐ HTML table

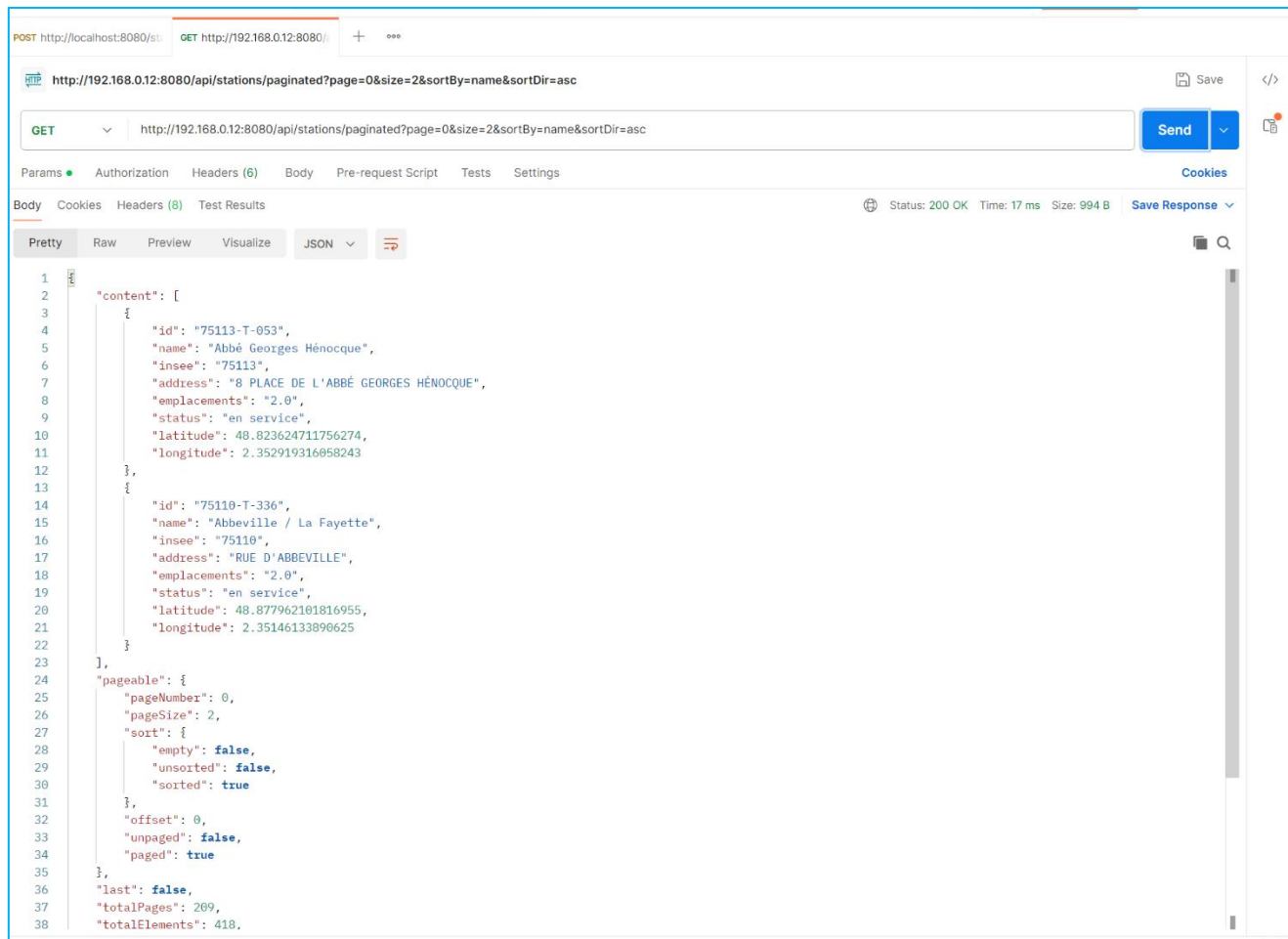
Preview

Download

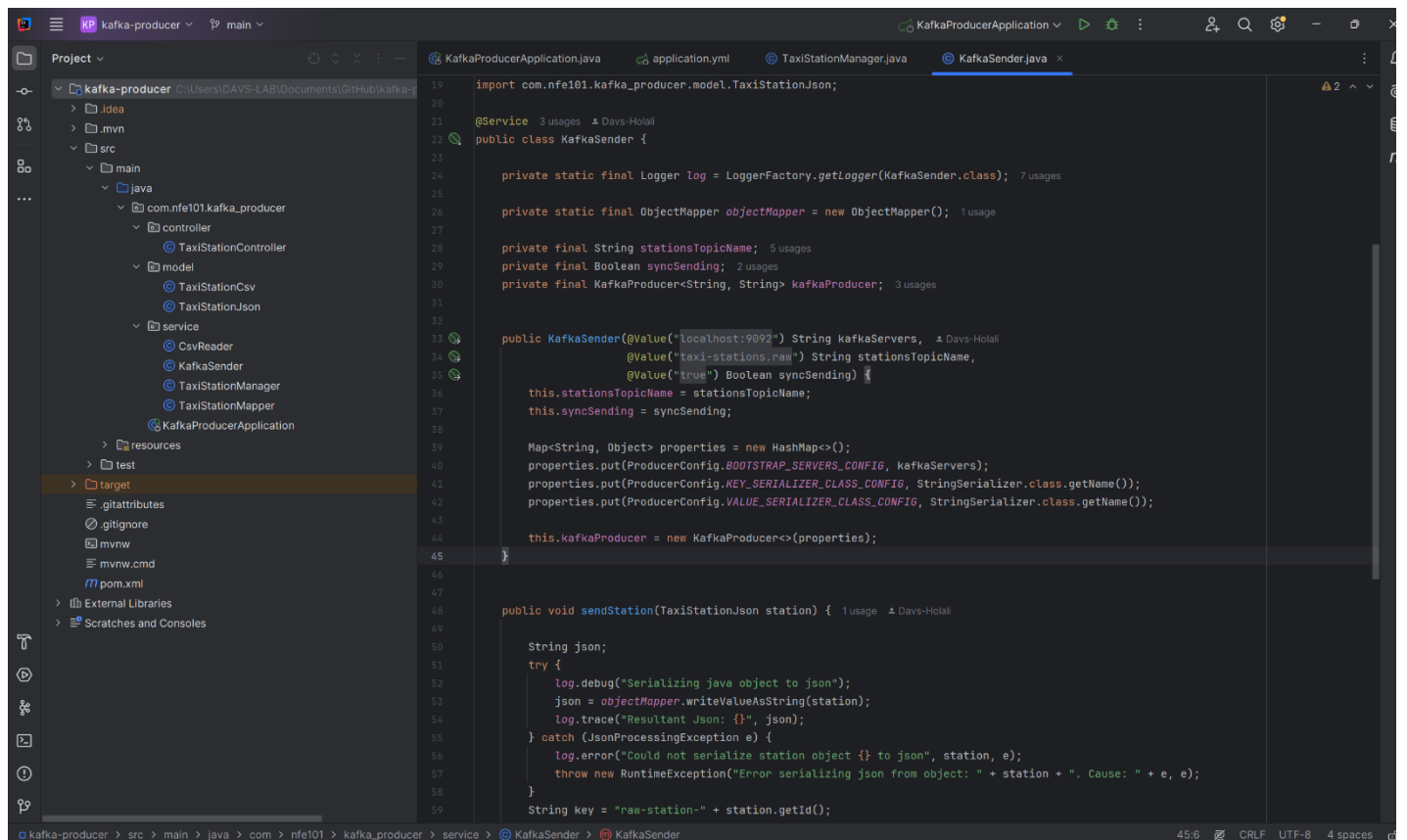
Cancel



## Annexe 3 : Requête API via Postman



## Annexe 4 : Service Kafka Producer





## Annexe 5 : Service Kafka consumer and API

The screenshot shows an IDE with the project 'kafka-consumer-api' open. The file 'KafkaConsumerApiApplication.java' is selected, showing the following code:

```
7
8 import java.sql.Connection;
9 import java.sql.DriverManager;
10
11 @SpringBootApplication
12 public class KafkaConsumerApiApplication {
13
14     public static void main(String[] args) {
15         SpringApplication app = new SpringApplication(KafkaConsumerApiApplication.class);
16         ConfigurableEnvironment env = new StandardEnvironment();
17
18         // Retrieve MySQL properties from environment variables or application properties
19         String mysqlUrl = env.getProperty(key: "spring.datasource.url", defaultValue: "jdbc:mysql://localhost:3306/");
20         String mysqlUser = env.getProperty("spring.datasource.username");
21         String mysqlPassword = env.getProperty("spring.datasource.password");
22
23         try (Connection conn = DriverManager.getConnection(mysqlUrl, mysqlUser, mysqlPassword)) {
24             // MySQL connection successful
25         }
26     }
27 }
```

The console output shows the application starting successfully:

```
2025-01-20T20:47:40.797+01:00 INFO 15312 --- [kafka-consumer-api] [ restartedMain] c.n.k.KafkaConsumerApiApplication : Starting KafkaConsumerApiApplication using Java 23.0.1 with PID :
2025-01-20T20:47:40.804+01:00 INFO 15312 --- [kafka-consumer-api] [ restartedMain] c.n.k.KafkaConsumerApiApplication : The following 2 profiles are active: "h2", "mysql"
2025-01-20T20:47:40.898+01:00 INFO 15312 --- [kafka-consumer-api] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-prop
2025-01-20T20:47:42.185+01:00 INFO 15312 --- [kafka-consumer-api] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-01-20T20:47:42.287+01:00 INFO 15312 --- [kafka-consumer-api] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 84 ms. Found 1 JPA re
2025-01-20T20:47:43.637+01:00 INFO 15312 --- [kafka-consumer-api] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2025-01-20T20:47:43.671+01:00 INFO 15312 --- [kafka-consumer-api] [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-01-20T20:47:43.672+01:00 INFO 15312 --- [kafka-consumer-api] [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.33]
2025-01-20T20:47:43.780+01:00 INFO 15312 --- [kafka-consumer-api] [ restartedMain] o.a.c.g.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-01-20T20:47:43.785+01:00 INFO 15312 --- [kafka-consumer-api] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2886 ms
```

## Annexe 6 : Application flutter (extrait de codes)

The screenshot shows an IDE with the project 'taxi\_bornes\_app' open. The file 'home\_screen.dart' is selected, showing the following code:

```
7 class HomeScreen extends StatefulWidget {
8     const HomeScreen({Key? key}) : super(key: key);
9
10    @override
11    State<HomeScreen> createState() => _HomeScreenState();
12 }
13
14 class _HomeScreenState extends State<HomeScreen> {
15     int _currentIndex = 0;
16     List<Station> stations = [];
17
18    @override
19    void initState() {
20        super.initState();
21        _fetchStations();
22    }
23
24    Future<void> _fetchStations() async {
25        try {
26            final fetchedStations = await ApiService.fetchStations(
27                page: 0,
28                size: 50,
29            );
30            setState() {
31                stations = fetchedStations;
32            };
33        } catch (e) {
34            ScaffoldMessenger.of(context)
35                .showSnackBar(SnackBar(content: Text('Erreur : $e')));
36        }
37    }
38
39    @override
40    Widget build(BuildContext context) {
41        // ...
42    }
43 }
```

Annexe 7: Interface UI de l'application mobile flutter

