



Suivi et analyse de l'affluence en agence bancaire



De l'API à la visualisation : un pipeline data complet

Introduction



Optimiser l'accueil en agence bancaire



Analyser l'affluence en temps réel



Mieux gérer les ressources et améliorer l'expérience client

Problématique

- ? Comment savoir quand et combien de clients entrent dans une agence ?
- ? Comment anticiper les pics d'affluence ?
- ? Comment ajuster les effectifs en conséquence ?

Solution : Une application de suivi de l'affluence

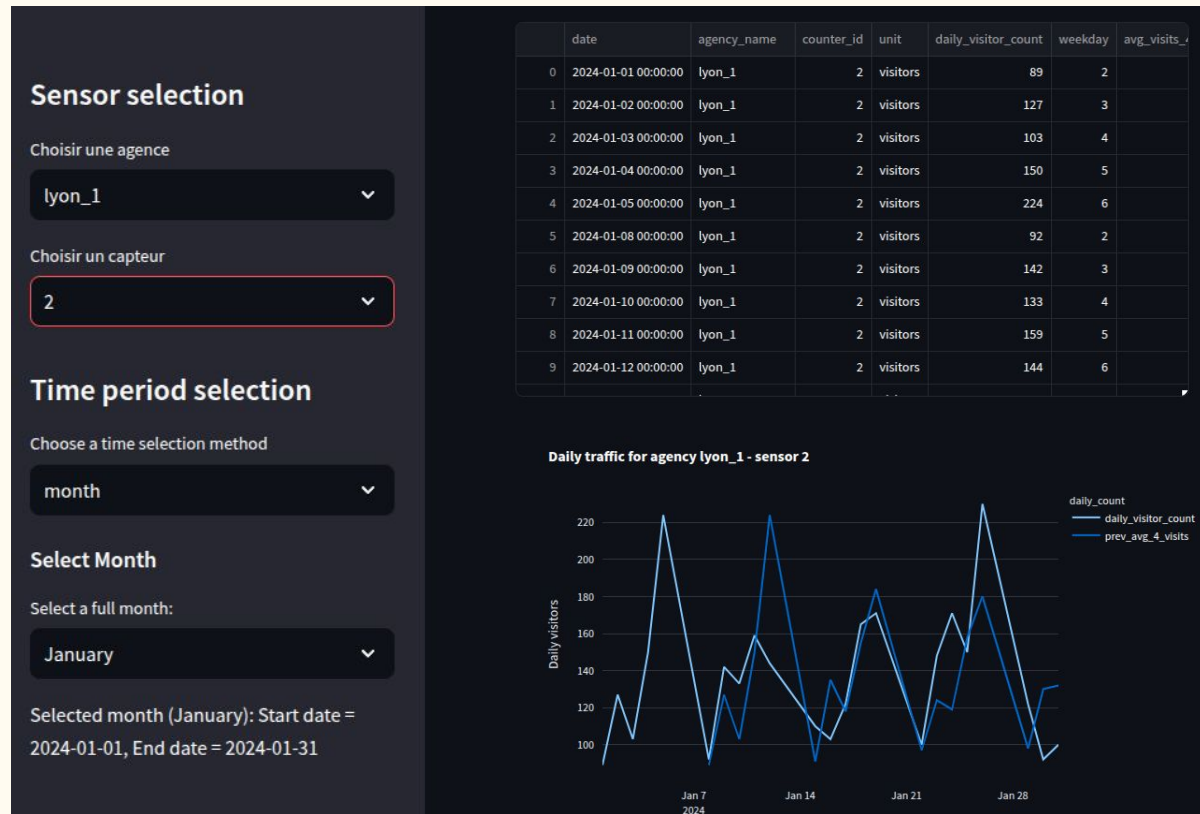
- ✓ Visualisation interactive des flux de visiteurs
- ✓ Données mises à jour quotidiennement
- ✓ Indicateurs clés pour optimiser la gestion

Démonstration de l'application



Tableau de bord Streamlit :

- Sélection d'une agence et d'une période
- Affichage de l'affluence sous forme de graphiques dynamiques
- Données mises à jour en continu



Collecte des données

API



API REST avec FastAPI



Données simulées en temps réel :

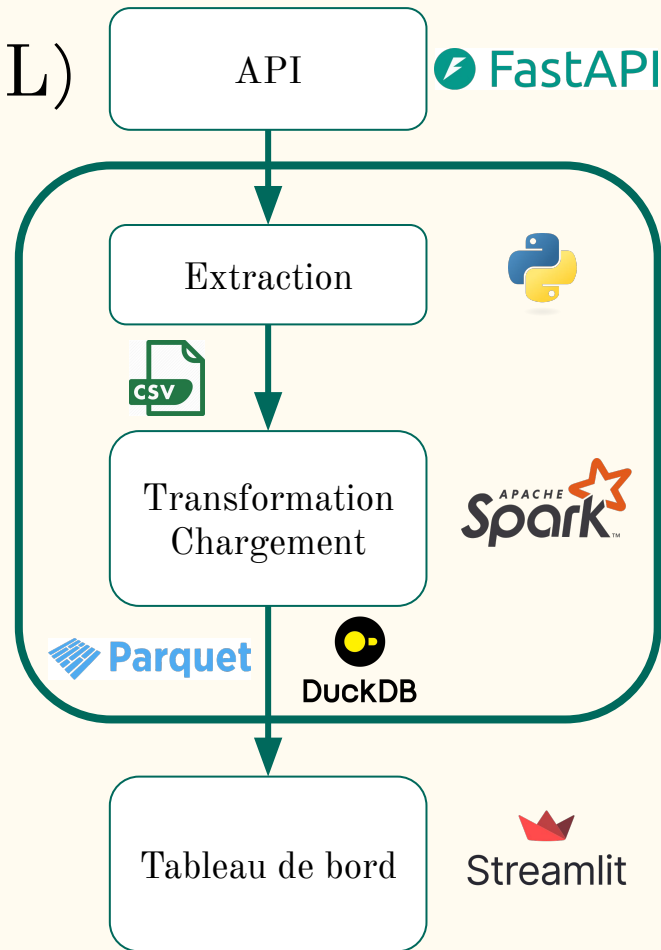
- Nombre de visiteurs par agence et par heure
- Défectuosité aléatoire des capteurs (mesure absente, réduite ou aberrante, capteur en panne)
- Erreurs aléatoires dans les données (nom d'agence avec faute de frappe, etc.)

Traitement des données (Pipeline ETL)



Étapes du pipeline :

- ① Extraction des données de l'API
- ② Nettoyage & Agrégation avec Spark
- ③ Calcul d'indicateurs :
 - Total visiteurs/jour
 - Moyennes glissantes
- ④ Stockage optimisé en Parquet



Orchestration avec Apache Airflow



🕒 Automatisation des tâches :

✅ DAG principal :

Extraction : chaque heure

Transformation : quotidienne

↺ DAG secondaire :

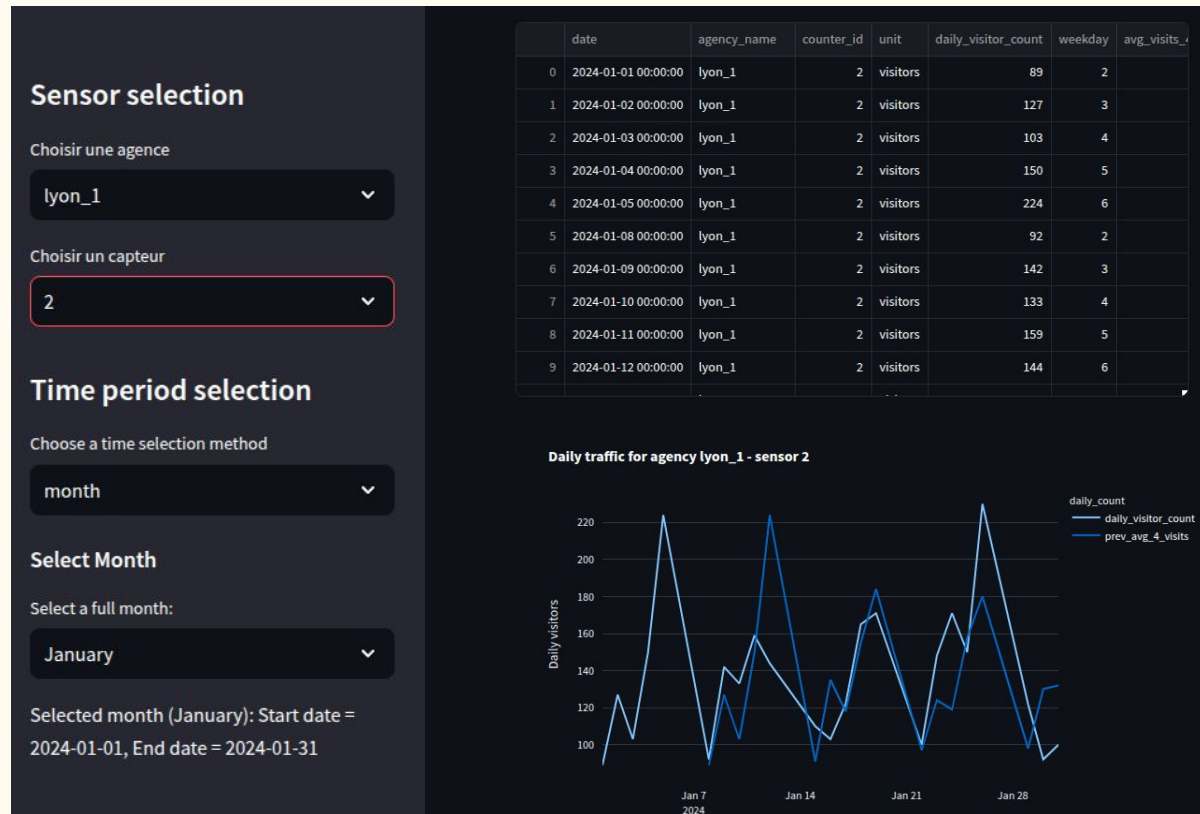
‘Backfill’ en cas de panne

Visualisation et analyse



Streamlit permet de :

- Lire les fichiers Parquet
- Générer des graphiques dynamiques
- Offrir une interface intuitive et évolutive pour explorer les données



Impact et bénéfices



Pour les agences bancaires :

- ✓ Meilleure gestion des effectifs
- ✓ Amélioration de l'expérience client
- ✓ Optimisation des ressources

Conclusion & Contact



Une solution complète, automatisée et efficace



Envie d'en savoir plus ? Échangeons !

Perspectives

- Hébergement sur un Amazon S3 (datalake) pour la scalabilité
- Cluster Kubernetes pour utiliser la parallélisation de Spark (actuellement : conteneurs docker orchestré par docker-compose)
- Implémentation des test de bon fonctionnement du pipeline
- DAG avec alerte en cas de détection de valeur élevée
- DAG avec alerte en cas de problème
- DAG pour gérer une panne en temps réel et lancer un backfill automatiquement
- Dashboard Grafana + Prometheus pour surveiller le bon fonctionnement du pipeline en temps réel
- Déploiement de Airflow sur un serveur (ici en standalone local)
- Ajout de nouvelles métriques :
- Ajout d'un bloc de Machine Learning pour prédire l'affluence future