

Pack Livrables – Modélisation, Architecture et Documentation

SmartShop 360

23 février 2026

Table des matières

1 Vue Métier

Livrable Urbaniste – Vue Métier

SmartShop 360

23 février 2026

Objectif

Cette vue représente les **acteurs** et les **processus métier** existants :

- Vente sur site
- Vente marketplace
- Analyse qualité

Acteurs

- Client Web
- Client Marketplace
- Analyste Data
- Responsable Qualité / Marketing
- Service Commercial
- Agent IA SmartShop 360

Processus

1. Le client passe commande via site ou marketplace.
2. Les ventes alimentent la source ERP et les avis alimentent la source JSON.
3. Le pipeline ETL unifie les référentiels produits (MDM) pour croiser ventes et avis.
4. Le responsable qualité/marketing exploite les alertes et KPI croisés.
5. Le service commercial et l'agent IA exploitent les résultats pour la décision.

Alignement avec le code

- Interface métier : `app.py` et modules `src/ui/*`
- Questions métiers en langage naturel : `src/ui/chat.py`
- Agent Text-to-SQL : `src/agent/graph.py`
- Alertes métiers (CRITIQUE / A_SURVEILLER) : vue `v_alerts`

Diagramme

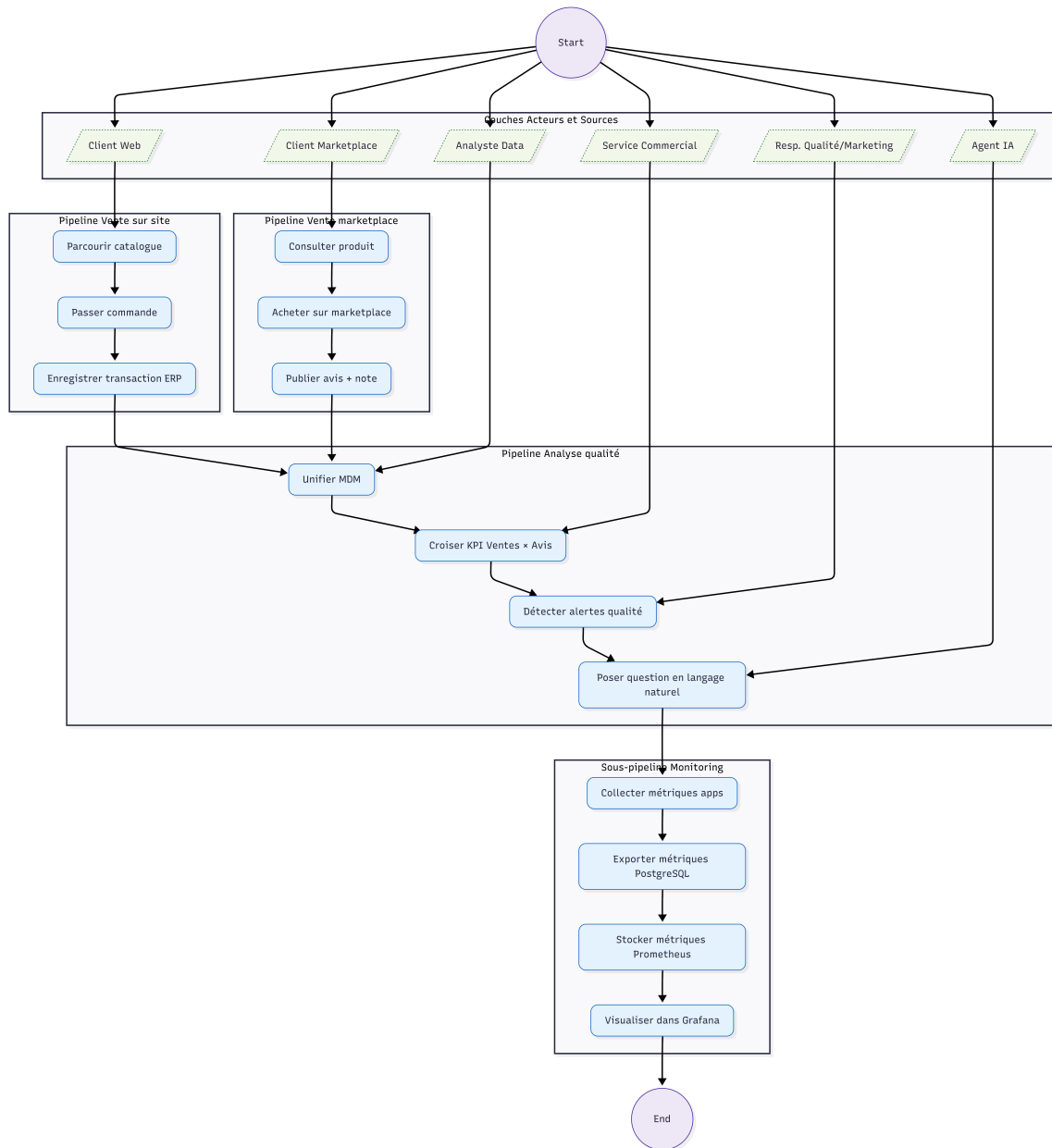


Diagramme Mermaid associé

Voir le fichier : 01_vue_metier.mmd

2 Vue Transformation

Livrable Urbaniste – Vue Transformation

SmartShop 360

23 février 2026

Objectif

Distinguer clairement :

- **SI Opérationnel** (sources de données externes)
- **SI Décisionnel** (entrepôt PostgreSQL + vues analytiques)

Mettre en évidence le **Référentiel Produit (MDM)** comme pivot entre les silos.

Description de la transformation

1. Les sources `online_retail_II.csv` (ERP) et `labeledReview.datasetFix.json` (avis) arrivent dans l'ETL.
2. `incremental.py` compare les hashes SHA-256 et évite les rechargements inutiles.
3. `cleaning.py` et `data_quality.py` nettoient/valident les données.
4. `mdm_mapping.py` produit le pivot `product_mapping` (TF-IDF + fallback rank).
5. `run_etl.py` charge les tables et publie les vues `v_product_kpi`, `v_customer_kpi`, `v_alerts`, `v_data_quality`.
6. Les couches UI, Agent IA, Analytics/ML et monitoring consomment le SI décisionnel.

Séparation SI Opérationnel / SI Décisionnel

- **Opérationnel** : fichiers sources (CSV/JSON), données hétérogènes non jointes.
- **Pivot MDM** : `product_mapping` relie les identifiants ERP et Avis.
- **Décisionnel** : PostgreSQL 16 (tables `products`, `customers`, `sales_facts`, `review_facts`) + vues KPI.

Diagramme

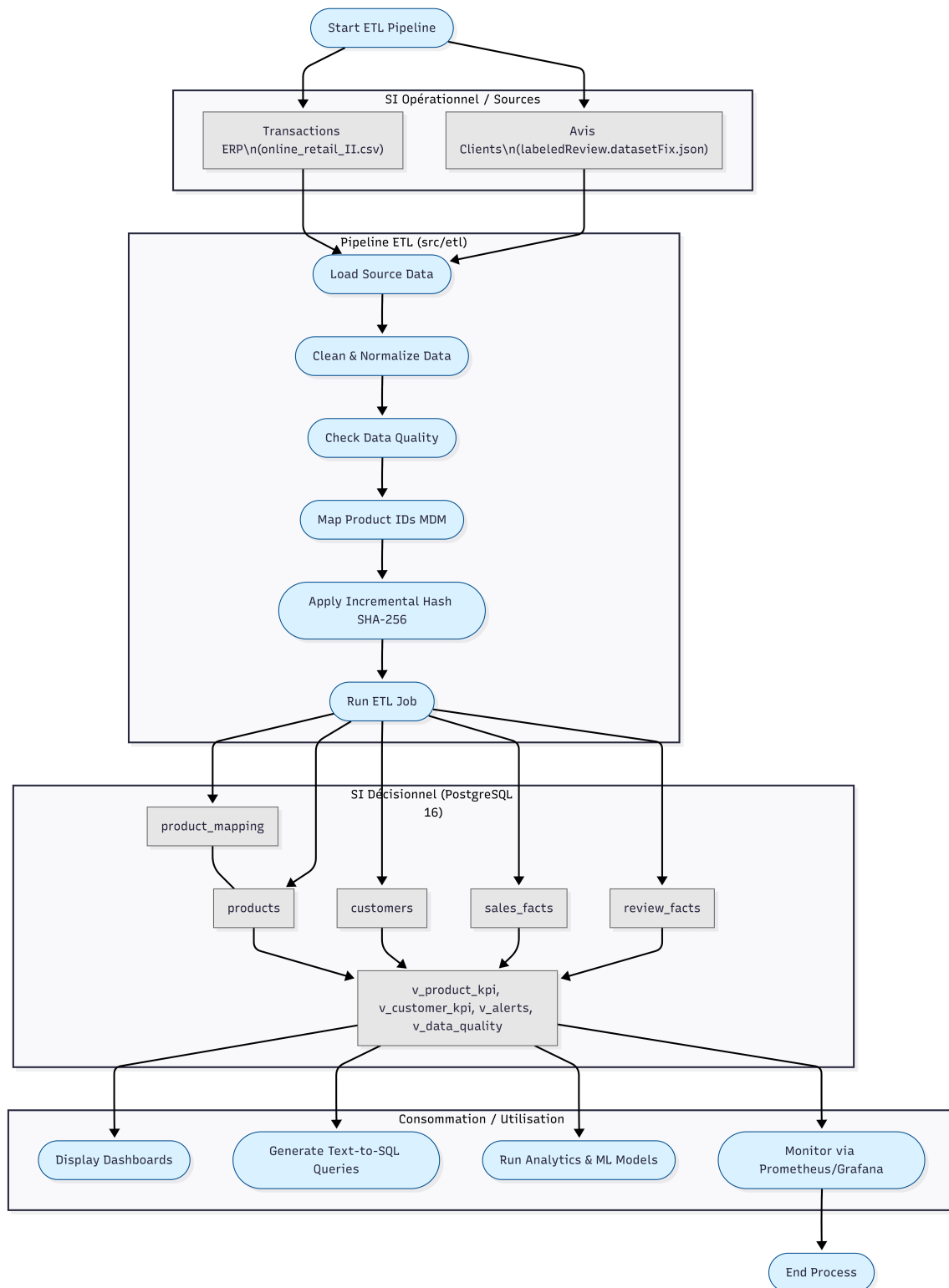


Diagramme Mermaid associé

Voir le fichier : 02_vue_transformation.mmd

3 DFD Pipeline ETL

Diagramme Technique – DFD Pipeline ETL

SmartShop 360

23 février 2026

Objectif

Détailler le flux de données de bout en bout :

Source → Nettoyage → Mapping/Unification → Base finale

Étapes du pipeline

1. Détection incrémentale : `should_run_etl()` compare les hashes des sources.
2. Ingestion/Nettoyage transactions : suppression annulations, quantités/prix invalides, calcul **Revenue** et **Margin**.
3. Ingestion avis : normalisation du sentiment et génération des notes.
4. Qualité : exécution des règles `run_all_validations()` sur transactions, avis et produits.
5. Mapping/Unification MDM : `build_product_mapping()` avec TF-IDF (seuil 0.35) et fallback rank.
6. Enrichissement avis : `attach_product_to_reviews()` + `ReviewDate`.
7. Chargement PostgreSQL : `products`, `customers`, `product_mapping`, `sales_facts`, `review_facts`.
8. Publication des vues analytiques : `v_product_kpi`, `v_customer_kpi`, `v_alerts`, `v_data_quality`.

Sorties principales

- Base finale relationnelle : PostgreSQL 16
- Vues KPI prêtes pour UI et Agent IA
- Historique incrémental des sources : `.etl_hashes.json`

Diagramme

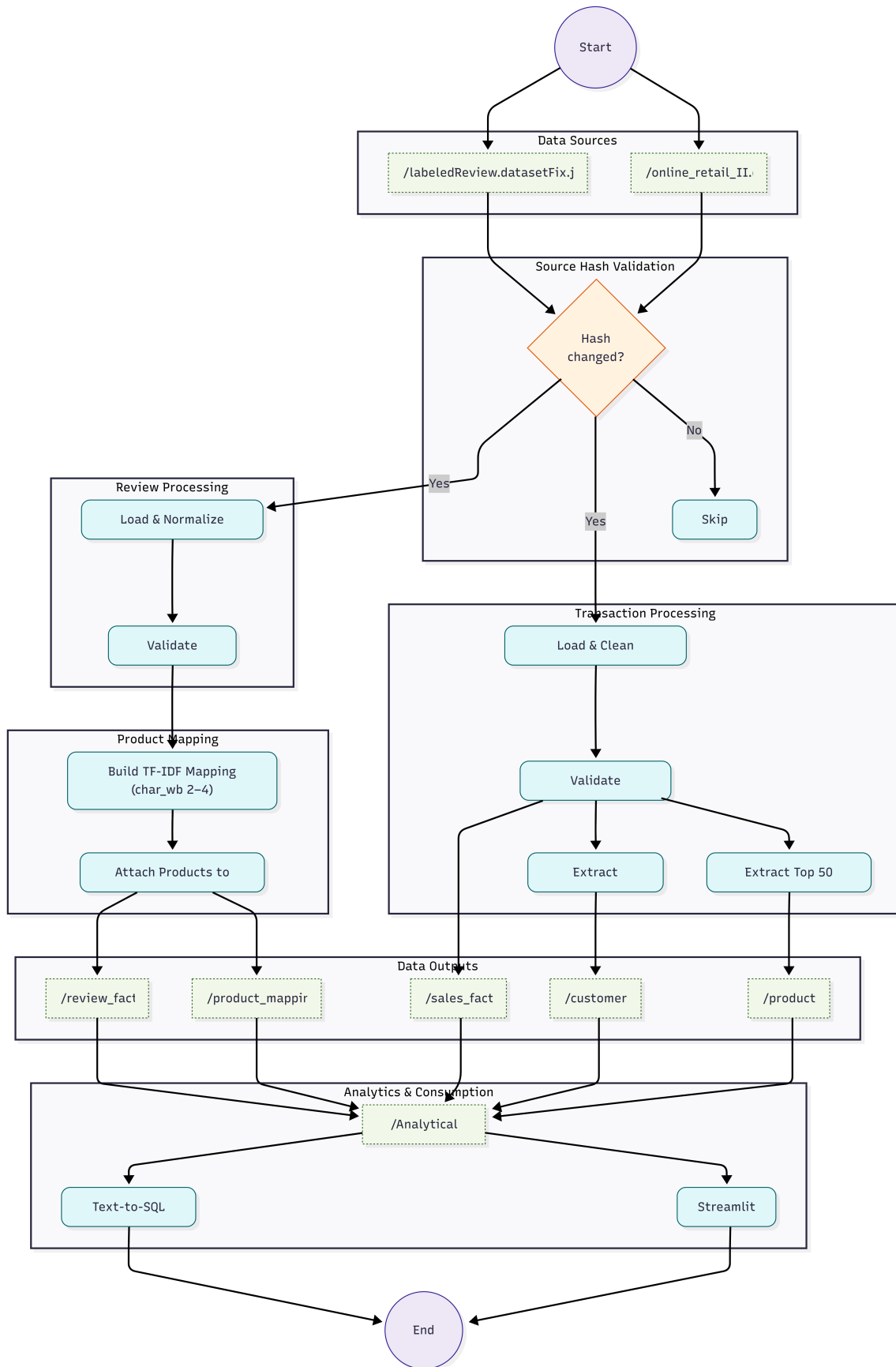


Diagramme Mermaid associé

Voir le fichier : 03_dfd_pipeline_etl.mmd

4 Architecture Physique

Diagramme Technique – Architecture Physique

SmartShop 360

23 février 2026

Objectif

Représenter les conteneurs applicatifs et les flux réseau :

- Conteneur PostgreSQL (stockage opérationnel et décisionnel)
- Conteneur App Python/Streamlit (UI + logique métier + ETL)
- Flux vers API LLM externe

Description

L'application Streamlit/Python interagit avec PostgreSQL pour l'ETL, le MDM et la consultation des KPIs. Les requêtes avancées de langage naturel sont transmises à une API LLM externe.

Diagramme Mermaid associé

Voir le fichier : `04_architecture_physique.mmd`

5 Documentation Urbanistique & Métier

Documentation Urbanistique & Métier

SmartShop 360

23 février 2026

Table des matières

1 Stratégie MDM (Master Data Management)

1.1 Approche implémentée dans le projet

Le projet met en place une table pivot `product_mapping` qui joue le rôle de *golden record* produit entre la source ERP et la source avis. Le mapping repose sur :

- normalisation des libellés dans le pipeline ETL,
- rapprochement fuzzy TF-IDF (`char_wb`, n-grammes 2 à 4),
- seuil de similarité cosinus (0.35),
- fallback `rank` quand aucun match n'atteint le seuil,
- conservation du `MatchScore` et de `MatchStrategy` (traçabilité).

1.2 Approche recommandée en contexte réel

En production, la stratégie cible est hiérarchisée :

1. **EAN/GTIN** comme clé de référence quand disponible (matching déterministe),
2. **fuzzy matching** (Levenshtein + TF-IDF) quand le code universel est absent,
3. **matching sémantique** par embeddings pour cas difficiles,
4. revue humaine des cas ambigus,
5. traçabilité systématique (règle, score, date, validateur) et boucle d'amélioration.

2 Dictionnaire de données (tables clés)

2.1 Product_Mapping (pivot MDM)

Table de référence de rapprochement :

- `ERP_StockCode` et `Review_ProductCode`,
- nom unifié produit,
- catégorie,
- score et stratégie de matching.

2.2 Products

Table de référence produits ERP (Top 50 dans le POC) :

- `ProductID`, `ProductName`, `Category`

2.3 Customers

Table clients ERP :

- `ClientID`, `Nom`, `Pays`

2.4 Sales_Facts

Table de faits des ventes :

- granularité transactionnelle (facture/ligne),
- quantités, chiffre d'affaires, marge,
- clés vers produits et clients.

2.5 Review_Facts

Table de faits des avis :

- identifiant produit côté avis,
- note, sentiment, texte d'avis,
- date d'avis.

2.6 Vues analytiques

- `v_product_kpi` (CA, note, avis, quantités)
- `v_customer_kpi` (CA client, panier moyen, commandes)
- `v_alerts` (statuts `CRITIQUE/A_SURVEILLER/OK`)
- `v_data_quality` (couverture et volumétrie)

3 Scénario de démonstration (vidéo 5 à 10 min)

3.1 Script recommandé

1. **Contexte (30s)** : présenter les deux silos et l'absence d'identifiant commun.
2. **Ingestion ETL (2-3 min)** : lancer `streamlit run app.py` (ETL auto si base vide).
3. **Validation (1 min)** : montrer `product_mapping` + `v_product_kpi`.
4. **Question complexe à l'agent (2-3 min)** :
“Quels produits vendus à plus de 50 exemplaires ont une note inférieure à 3 ?”
5. **Analyse de réponse (1-2 min)** : montrer le SQL généré, le résultat, et la décision métier.
6. **Robustesse (30s)** : démontrer le fallback hors-ligne (sans clé API) si nécessaire.

3.2 Critères de réussite

- cohérence entre les sources et les résultats,
- explicabilité des règles MDM,
- capacité de l'agent à répondre avec des données vérifiables,
- traçabilité : SQL généré + vues/colonnes utilisées.