Cahier des charges technique

TECH-TONIQUE

MIEUX QUE LES AUTRES

Vision produit

Distributech est un grossiste en équipement électronique qui cherche à améliorer la gestion de ses données, qui sont selon l'existant dans un état anarchique qui n'est ni scalable ni évolutif, avec l'existence d'aucune structure permettant de centraliser les données pour les analyser.

Problématique

Le client a besoin d'une solution pour fluidifier sa logistique (aussi bien sa structuration des données que son analyse des données) en centralisant l'ensemble de ses données au même endroit.

Valeur pour l'utilisateur final

Une meilleure logistique permettra indéniablement au client de réduire le temps de travail nécessaire sur cet aspect, augmentant ainsi le temps de travail alloué à des activités productives et enrichissantes.

User Stories

Trois besoins ont été clairement identifiés, donnant lieu à trois US (User Stories) bien délimitées.

Système de suivi des stocks

Pour arriver à fluidifier la logistique du client, il faut considérer un système capable de capturer une photographie des stocks à une date donnée, et de retourner cette photographie par le biais d'un support lisible et accessible.

Historique des commandes

L'historique des commandes rassemble les données commerciales critiques du client, et doit être accessible dans le système final, qui centralisera l'ensemble des données dans une structure robuste et efficace.

Base de données centralisée

Pilier central du projet, la base de données doit être structurée de façon à pouvoir accueillir les données relatives aux commandes et aux stocks. Il s'agit du cœur du produit final, par lequel on passe systématiquement lorsque l'on souhaite intégrer périodiquement de nouvelles données, ou bien lorsque l'on souhaite extraire des statistiques précises relatives aux stocks ou au Chiffre d'Affaires (CA) généré par les diverses commandes du client.

Spécifications techniques

Architecture générale du système

Architecture logique

Entrées :

- Fichiers CSV de commandes
- Base SQLite locale contenant les données de stocks et de revendeurs

Traitement :

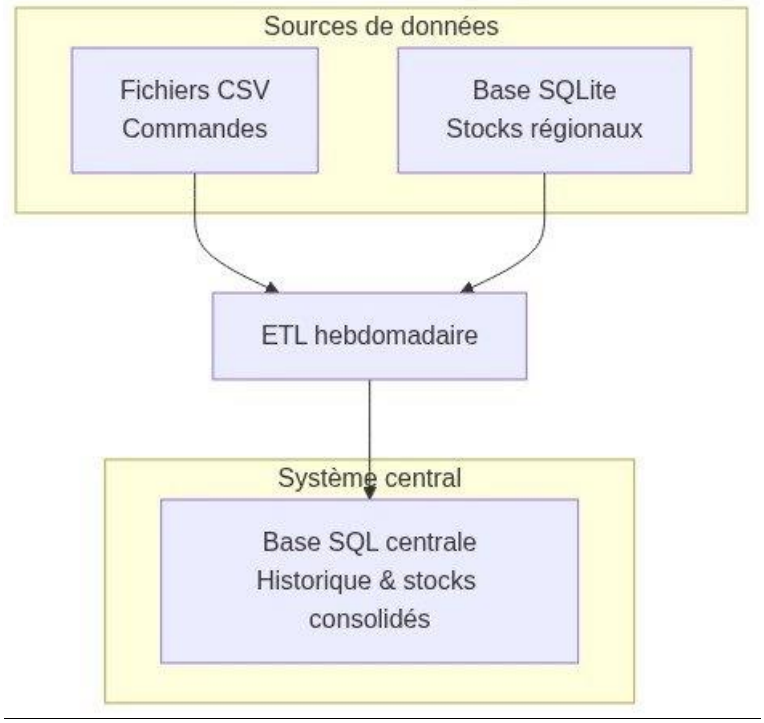
- Traitement des données via un Pipeline ETL (Extract -->Transform-->Load) en Python

Sorties :

- Base de données relationnelle centralisée SQL
- Fichier CSV généré automatiquement avec état du stock à jour

Shéma technique

Penser a ajouter légende



Environnement technique

Élément	Spécification technique
Langage de développement	Python 3.12.3, SQL
Librairies Python	Pandas 2.3.1, sqlite3, dotenv, pathlib, os, mysql.connector 9.3.0
Base source	SQLite, CSV
Base cible	MySQL (compatible SQL standard)
Outils supplémentaires	Docker 27.5.1, Git, Github, Notion
IDE	VsCode
Système d'exploitation	Windows & WSL, Linux (Ubuntu 24.04.2)

Modèle des données

Tables à créer :

Revendeurs (id_revendeur, nom_revendeur, region)

Produits (id_produit, nom_produit, cout_unitaire)

Production (id_production, quantite_production, date_production)

Regions (id_region, nom_region)

Commmandes (numero_commande, date_commande, quantitee_commande, prix_unitaire , total, region, revendeur, produit)

Structure du Pipeline ETL

Étape 1 – Extraction

- Lecture de fichier CSV de commandes
- Connexion à la base SQLite pour lecture des données actuelles de stock et des revendeurs

Étape 2 – Transformation

- Vérification des doublons
- Formatage des dates
- Validation des références (produits existants, revendeurs valides)
- Normalisation des noms/valeurs

Étape 3 – Chargement

- Création de la base MySQL
- Insertion dans la base MySQL
- Mise à jour des stocks (calcul basé sur les commandes et les réceptions)
- Génération du fichier csv pour le stock final (Procédure)

Contraintes techniques

Type	**Détail**
Matérielles	Poste avec Python et accès à distance à une base SQL
Logicielles	Outils open source (Python, SQLite, MySQL) Outils propriétaires : VsCode (Microsoft)
Sécurité des données	Données commerciales sensibles (revendeurs)
Sécurité des identifiants	Standard industrie, données sensibles rangées dans un fichier .env PAS UPLOAD sur git.
Performances	Capable de traiter plusieurs fichiers CSV hebdomadaires Scripts python efficaces

Robustesse	Gestion des erreurs lors de lecture de fichiers ou connexion DB et retour console.
Modularité	Code ETL structuré par fonctions/modules pour faciliter la maintenance
Stockage	BDD centralisée Mysql

Conformité RGPD

Respect des principes du RGPD

Principe RGPD	Application dans le projet
Minimisation	Aucune donnée personnelle superflue n'est collectée ni conservée.
Licéité et finalité	Le traitement est justifié par un intérêt légitime (logistique interne).
Exactitude	Les données sont validées et nettoyées automatiquement.
Limitation de durée	Les fichiers temporaires sont supprimés après traitement.
Intégrité et confidentialité	Le système utilise des accès limités, des logs et aucune exposition externe des données.

Cas d'évolution du projet

- Une analyse d'impact sur la vie privée
- Contrôle d'accès par rôle
- Chiffrement

Points de vigilance

Livrables techniques attendus

- `Fichier d'export SQL provenant de la base MySQL.`
- `etl_pipeline (extract.py, transform.py, load.py, main.py)` : scripts Python du processus ETL
- `.env` : fichier de configuration (connexions à Mysql)
- `stock_final.csv` : fichier généré après chargement
- `README.md` : instructions d'installation et d'exécution

Tests techniques

Test	Objectif	Résultat attendu
Insertion CSV	Charger un fichier CSV de commandes	Données présentes en base

Chargement SQLite	Lire revendeurs/stocks de SQLite	Tables correctement peuplées
Vérification doublons	Éliminer les doublons dans les lignes de commande	Aucun doublon dans la base cible
Génération stock final	Calculer le stock courant	Fichier `stock_final.csv` cohérent
Test d'erreur	Gérer les erreurs du code	Message d'erreur clair, pas de crash
Performance (fichiers multiples)	Traiter plusieurs fichiers d'un coup	Chargement des données complet