Base de données E-commerce – shopdb

Sarah Mahmoudi
Version : v1.0
Date : 02/11/2025

2. Introduction et Objectifs

2.1 Contexte du projet

Ce projet s'inscrit dans le cadre de la conception d'une base de données relationnelle destinée à la gestion d'une plateforme **e-commerce**.
L'objectif principal est de permettre la mise en relation entre des vendeurs et des clients, de faciliter la commercialisation de produits en ligne, et d'assurer le suivi complet du processus d'achat, depuis la publication du produit jusqu'à sa livraison finale.

La base de données constitue le cœur du système d'information, garantissant :

- La **cohérence** des données,
- La **traçabilité** des échanges,
- Le **suivi des opérations métiers** (commandes, paiements, livraisons).

2.2 Objectifs fonctionnels

La base de données doit permettre :

- La **gestion des comptes utilisateurs** selon différents rôles : client, vendeur, livreur, gestionnaire de stock, fournisseur, analyste, administrateur.
- La **gestion du catalogue produits** : ajout, modification, consultation et désactivation automatique.
- La **gestion des stocks**, avec **seuil d'alerte** pour notifier les ruptures.
- La **gestion du panier**, de la commande et de la livraison des produits.
- La **prise en charge des paiements** ainsi que la **génération automatique des factures**.
- La possibilité pour les clients de **laisser des avis** sur les produits achetés.

2.3 Objectifs techniques

Sur le plan technique, la base vise à :

- **Assurer l'intégrité** via des clés primaires/étrangères clairement définies.
- **Optimiser les performances** grâce à des index adaptés aux requêtes fréquentes.
- Garantir la **modularité et l'évolutivité** du modèle de données.
- Centraliser certaines règles métier via :
    - **Procédure stockée** (sp_ajouter_ligne_commande)

- **Trigger de gestion du statut produit** (trg_produit_statut_by_stock)
- **Trigger de facturation automatique** (trg_facture_after_paiement)

- Faciliter le **reporting** à l'aide de vues métiers simples (v_stats_ventes, v_comportement_client, etc.).

2.4 Périmètre

Le projet comprend :

Gestion des comptes

Catalogue et stock

Commandes & paniers

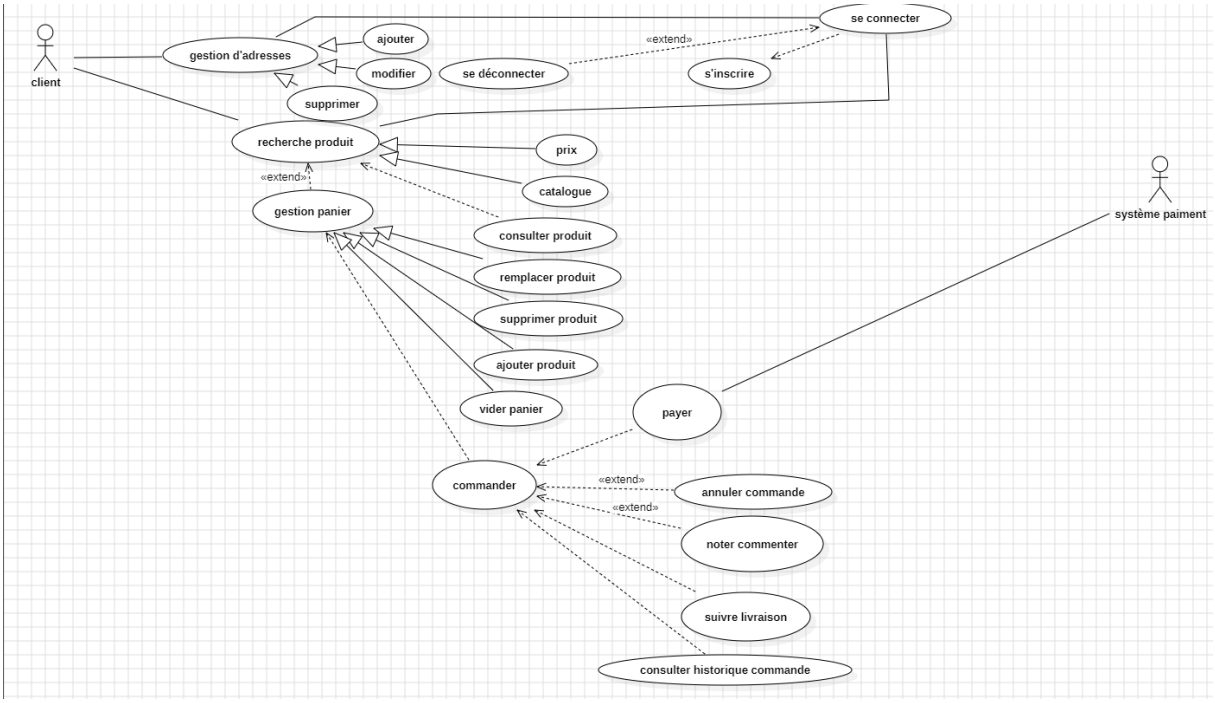
Paiements & factures

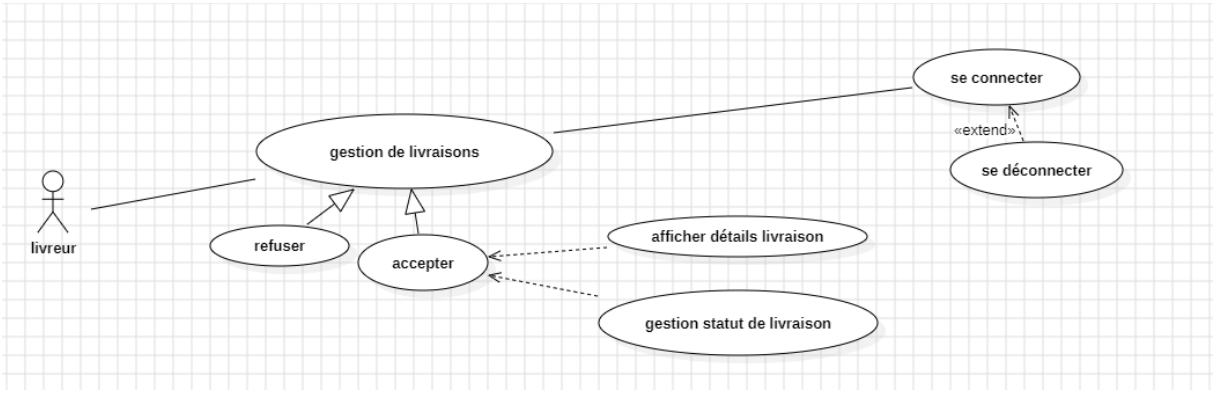
Livraisons

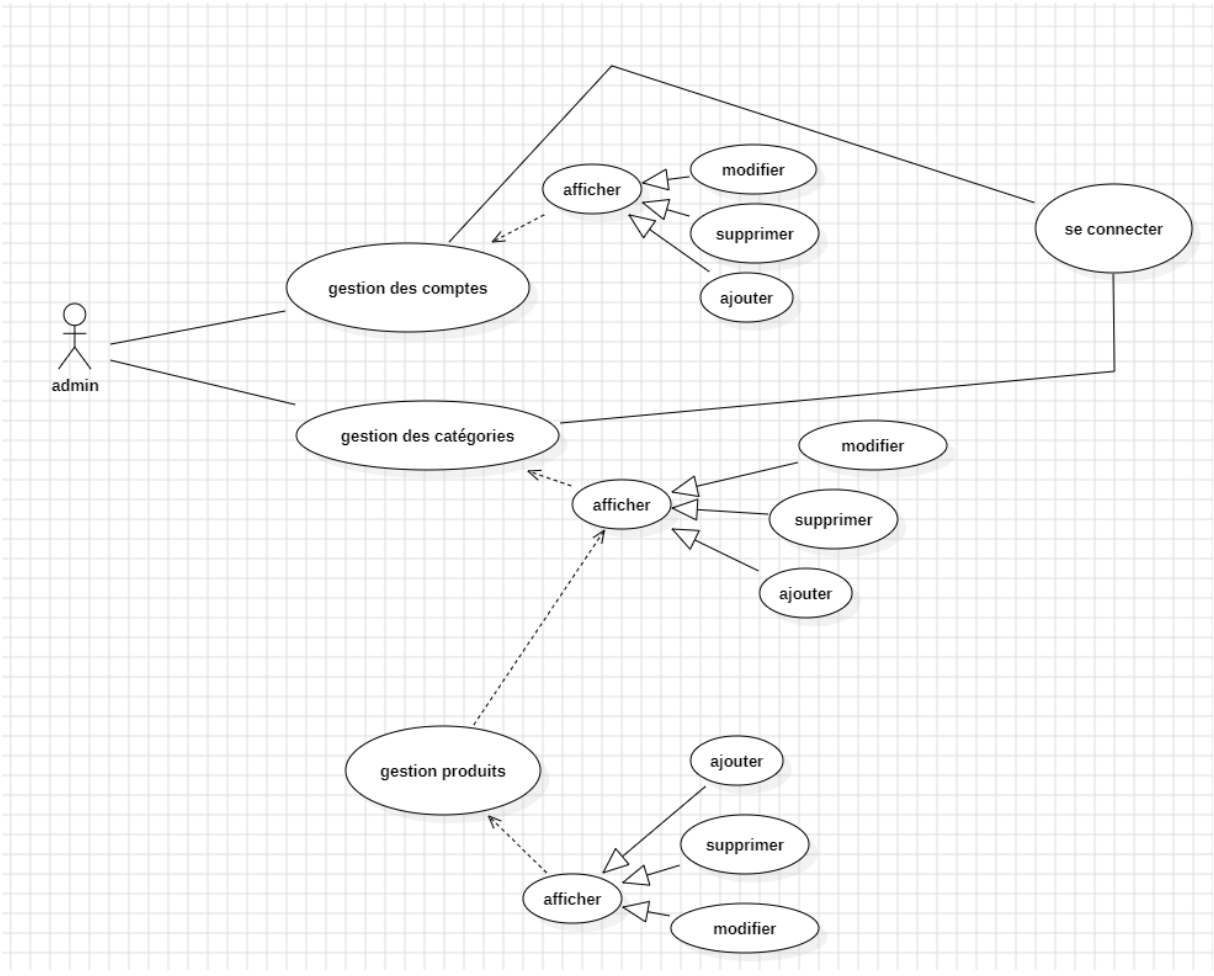
Avis & notation produit

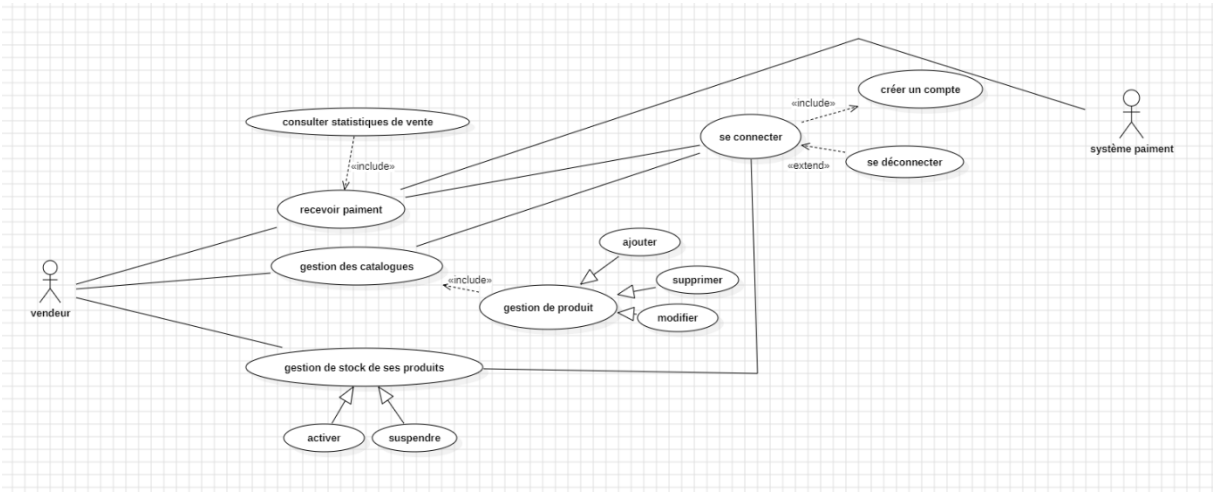
3. Conception du Système

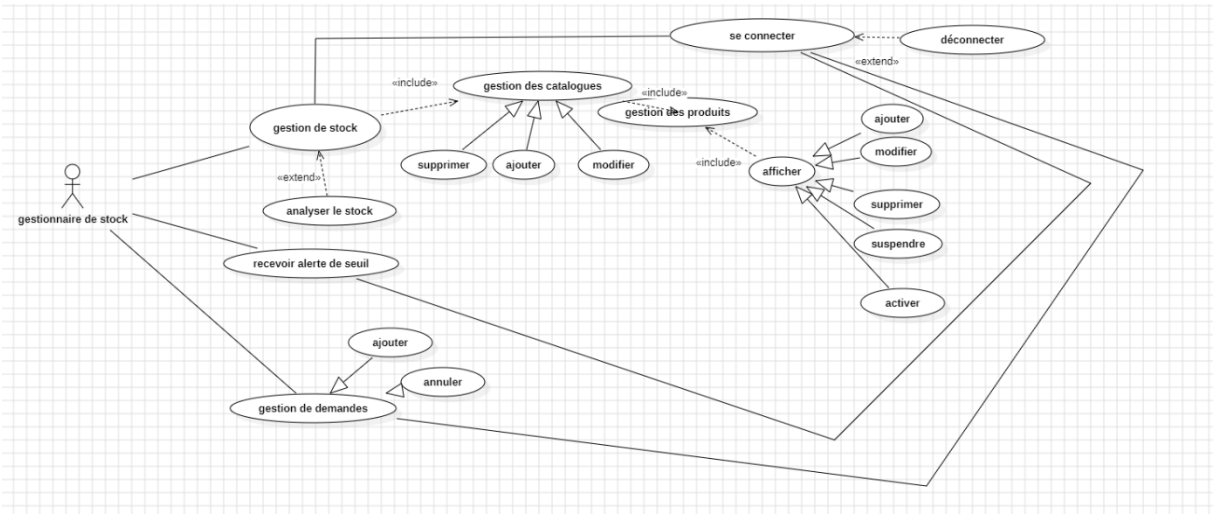
3.1 Analyse des besoins et identification des acteurs

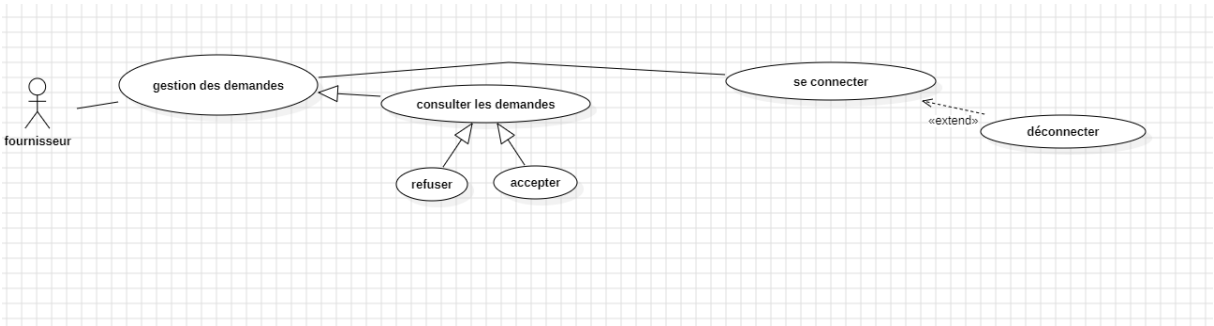


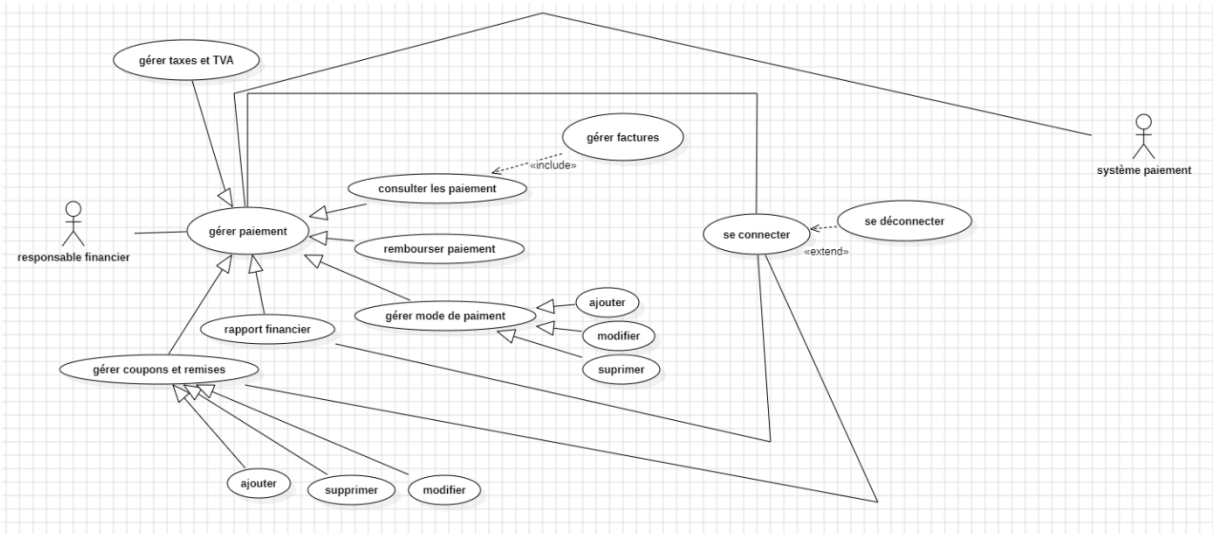

se connecter
«extend»
se déconnecter
gestion de livraisons
livreur
refuser
accepter
afficher détails livraison
gestion statut de livraison

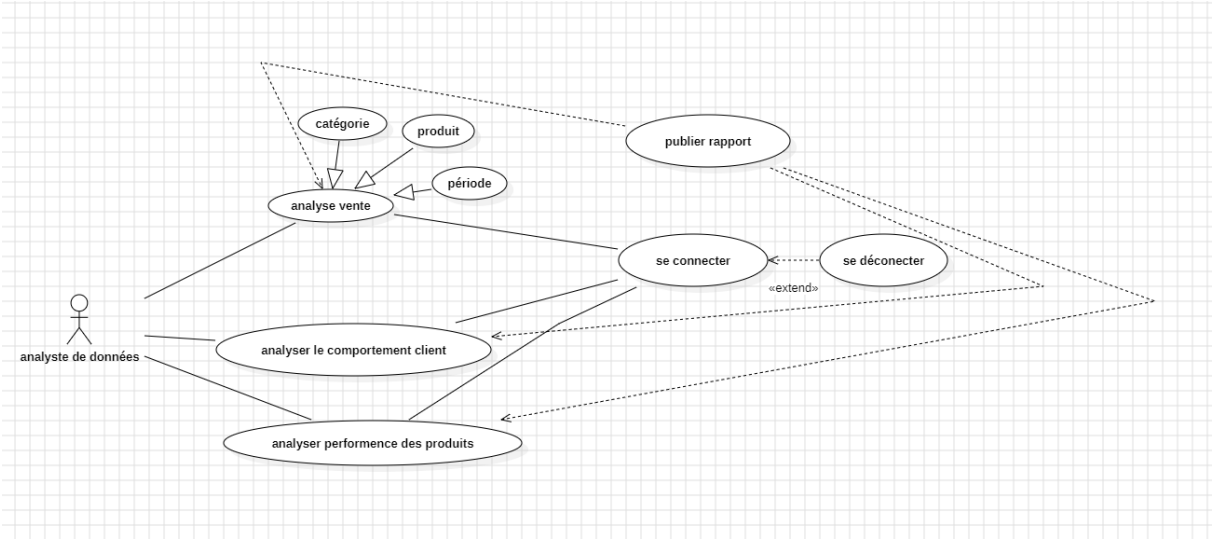

modifier
afficher
supprimer
se connecter
gestion des comptes
ajouter
admin
gestion des catégories
modifier
afficher
supprimer
ajouter
gestion produits
ajouter
supprimer
afficher
modifier


vendeur
consulter statistiques de vente
«include»
recevoir paiment
gestion des catalogues
«include»
gestion de produit
ajouter
supprimer
modifier
gestion de stock de ses produits
activer
suspendre
se connecter
«include»
créer un compte
«extend»
se déconnecter
système paiment


gestionnaire de stock
gestion de stock
«extend»
analyser le stock
recevoir alerte de seuil
«include»
gestion des catalogues
supprimer
ajouter
modifier
«include»
gestion des produits
«include»
afficher
ajouter
modifier
supprimer
suspendre
activer
ajouter
annuler
gestion de demandes
se connecter
déconnecter
«extend»


fournisseur
gestion des demandes
consulter les demandes
refuser
accepter
se connecter
«extend»
déconnecter

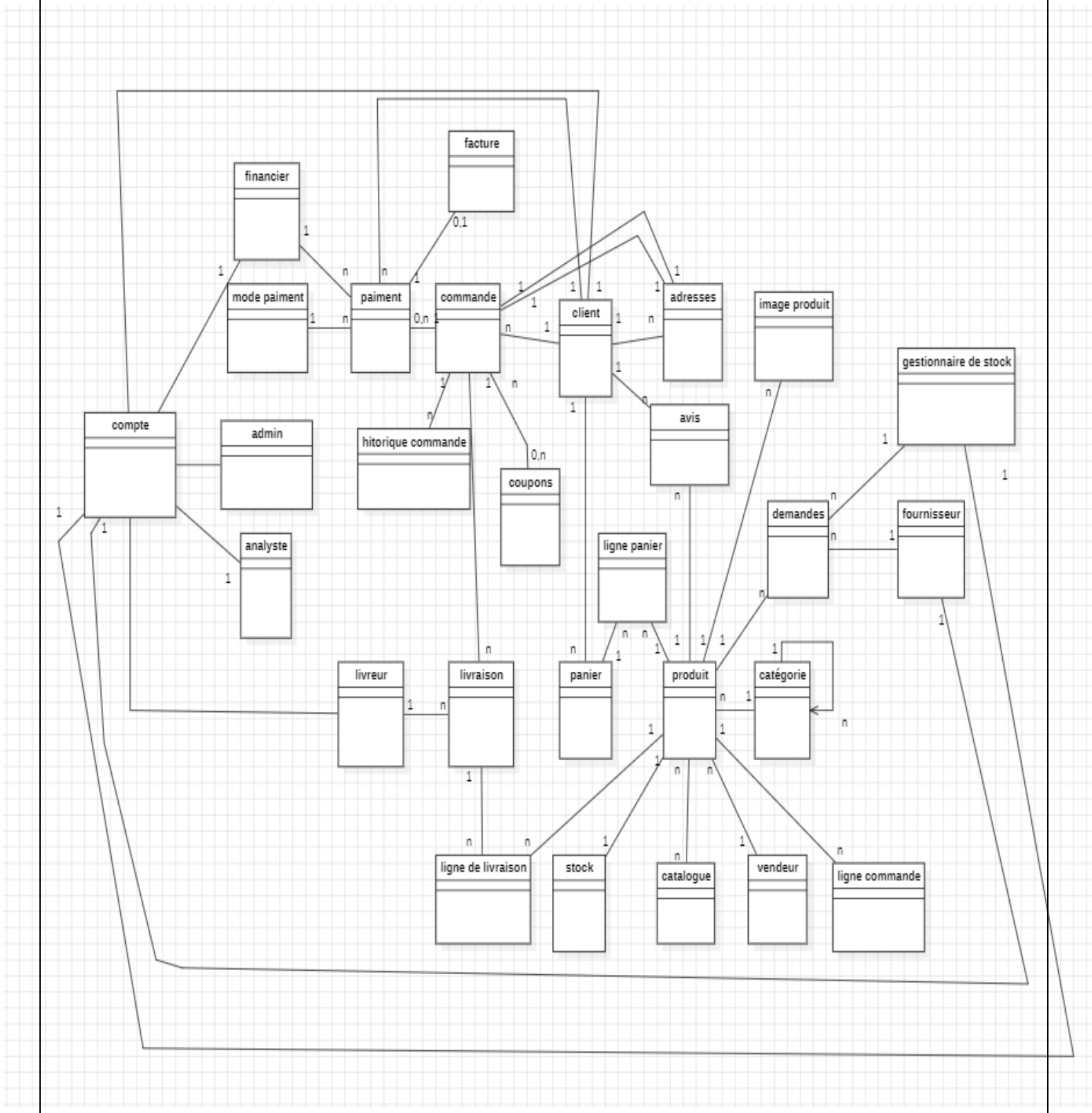


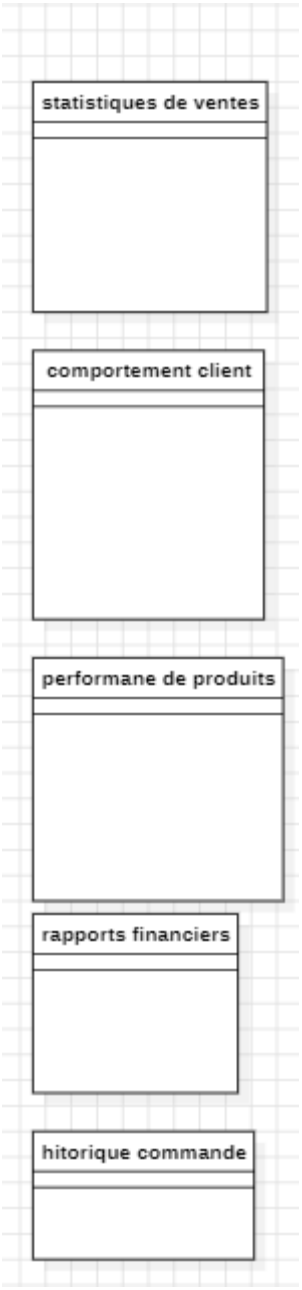


3.1 Analyse des besoins et identification des acteurs

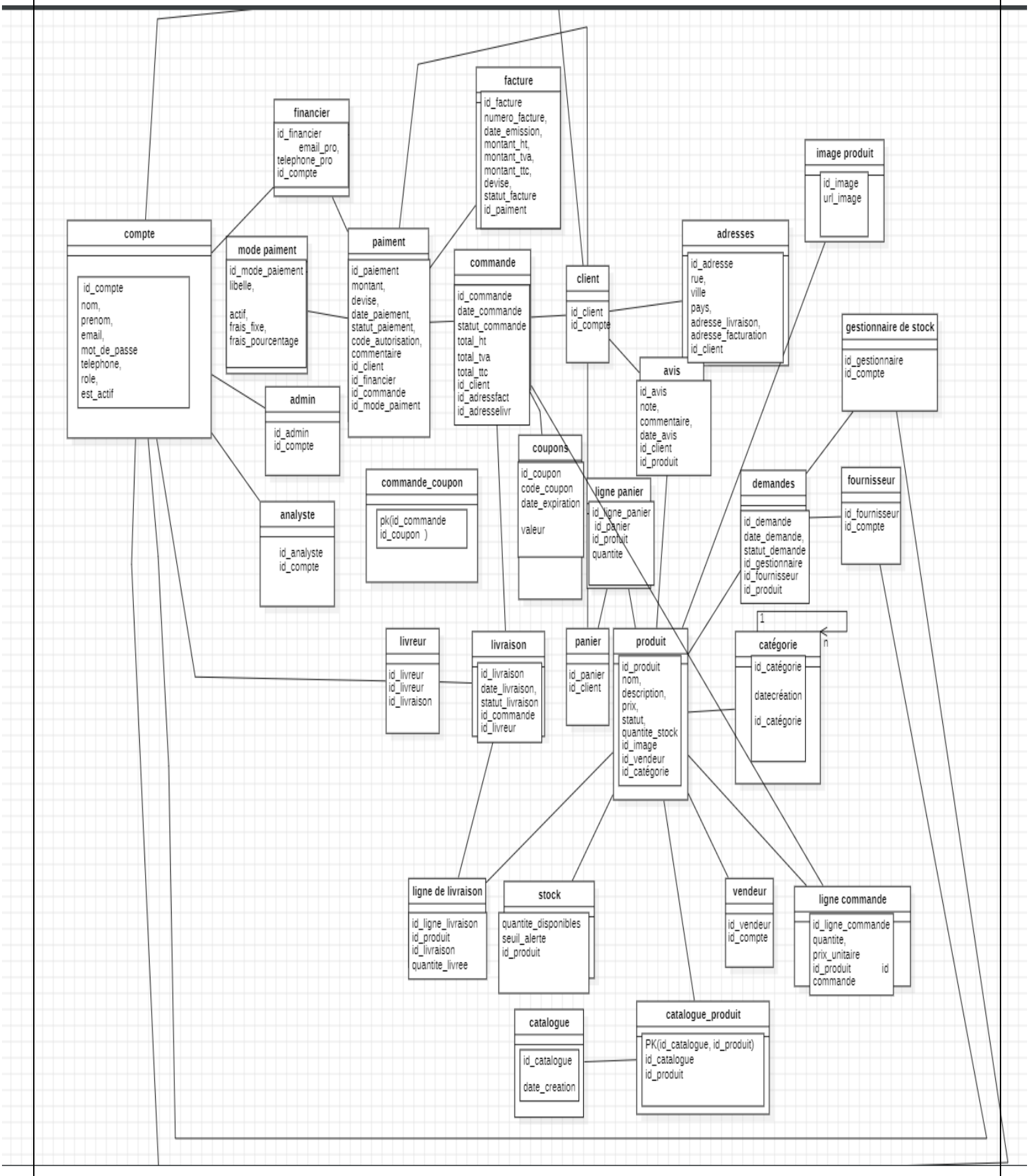
3.1 Analyse des besoins et identification des acteurs

3.2 MCL


facture
financier
mode paiment
paiment
commande
client
adresses
image produit
gestionnaire de stock
compte
admin
hitorique commande
coupons
avis
demandes
fournisseur
analyste
ligne panier
livreur
livraison
panier
produit
catégorie
ligne de livraison
stock
catalogue
vendeur
ligne commande


statistiques de ventes
comportement client
performane de produits
rapports financiers
hitorique commande

3.3 LMD



4.Extraits SQL

La table compte centralise les informations d’authentification et d’identité des utilisateurs.
Le champ role permet de distinguer les différents types d’utilisateurs du système.
Les tables admin, analyste, financier, gestionnaire_de_stock, vendeur, fournisseur, livreur et client sont reliées à compte par des clés étrangères.
Cette structure respecte le **principe de spécialisation**, permettant d’associer des autorisations spécifiques selon le rôle, **sans duplication d’informations personnelles**.

```sql
-- ========================
-- 10) INDEX
-- ========================
CREATE INDEX idx_compte_email        ON compte(email);
CREATE INDEX idx_prod_nom            ON produit(nom);
CREATE INDEX idx_cmd_client_date     ON commande(id_client, date_commande
CREATE INDEX idx_ligne_cmd_commande  ON ligne_commande(id_commande);
CREATE INDEX idx_pay_cmd             ON paiement(id_commande);
CREATE INDEX idx_liv_cmd             ON livraison(id_commande);

USE shopdb;

-- 1) Statistiques de ventes (par jour)
CREATE OR REPLACE VIEW v_stats_ventes AS
SELECT
  DATE(date_commande) AS jour,
  COUNT(*)            AS nb_commandes,
  SUM(total_ttc)      AS ca_ttc
FROM commande
GROUP BY DATE(date_commande);
```

Les index sont utilisés pour **améliorer les performances** des requêtes les plus fréquentes.

- idx_compte_email permet une recherche rapide lors de la connexion ou de la vérification d'un utilisateur.
- idx_prod_nom accélère la recherche de produits par nom dans le catalogue.
- Les index sur commande, ligne_commande, paiement et livraison optimisent l'historisation des ventes et la consultation des transactions.
  L'ajout d'index **réduit le temps d'exécution** en évitant les scans complets de tables (Full Scan).

Vue v_stats_ventes (Statistiques de ventes)

Cette vue permet d'obtenir une **synthèse quotidienne** du volume de ventes.
Elle calcule :

- Le **nombre de commandes passées**,
- Le **chiffre d'affaires journalier**.
  Elle est utile pour les tableaux de bord, rapports de performance et suivi commercial, **sans écrire de requêtes SQL complexes**.

```sql
-- 2) Comportement client (indicateurs basiques)
CREATE OR REPLACE VIEW v_comportement_client AS
SELECT
  c.id_client,
  COUNT(co.id_commande) AS nb_commandes,
  MAX(co.date_commande) AS derniere_commande,
  SUM(co.total_ttc)     AS total_ttc
FROM client c
LEFT JOIN commande co ON co.id_client = c.id_client
GROUP BY c.id_client;

-- 3) Performance produits (quantité vendue et CA HT)
CREATE OR REPLACE VIEW v_performance_produits AS
SELECT
  p.id_produit,
  p.nom,
  SUM(lc.quantite)                          AS qte_vendue,
  SUM(lc.quantite * lc.prix_unitaire)       AS ca_ht
FROM produit p
LEFT JOIN ligne_commande lc ON lc.id_produit = p.id_produit
GROUP BY p.id_produit, p.nom;

-- 4) Rapports financiers (par jour de paiement)
CREATE OR REPLACE VIEW v_rapports_financiers AS
SELECT
  DATE(p.date_paiement) AS jour,
  COUNT(*)              AS nb_paiements,
  SUM(p.montant)        AS montant_total
FROM paiement p
GROUP BY DATE(p.date_paiement);
```

Vue v_comportement_client (Analyse de fidélité)

Cette vue fournit une **analyse du comportement des clients** :

- nombre de commandes passées,
- date de la dernière commande,
- total dépensé.
  Elle permet d'identifier les clients fidèles et ceux inactifs, ce qui peut servir pour des **campagnes marketing ciblées** ou programmes de fidélité.

Vue v_performance_produits (Produits les plus vendus)

Cette vue permet de déterminer les **produits les plus performants** selon :

- la **quantité vendue**,
- le **chiffre d'affaires généré**.
  Elle aide le vendeur à **ajuster ses stocks**, ses stratégies de prix et ses priorités commerciales.

```
DELIMITER $$

DROP TRIGGER IF EXISTS trg_produit_statut_by_stock $$
CREATE TRIGGER trg_produit_statut_by_stock
AFTER UPDATE ON stock
FOR EACH ROW
BEGIN
  IF NEW.quantite_disponibles <= NEW.seuil_alerte THEN
    UPDATE produit
    SET statut = 0
    WHERE id_produit = NEW.id_produit;
  ELSE
    UPDATE produit
    SET statut = 1
    WHERE id_produit = NEW.id_produit;
  END IF;
END$$

DELIMITER ;
```

Trigger trg_produit_statut_by_stock (Désactivation automatique)

Ce trigger est exécuté après chaque mise à jour du stock.
S'il détecte que la quantité disponible est inférieure ou égale au seuil d'alerte, **le produit est**

automatiquement désactivé (statut = 0).
Cela empêche la vente de produits en rupture et améliore la **fiabilité du catalogue** pour les clients.
S'il y a du stock, le produit est réactivé (statut = 1).
Ce mécanisme **automatise une règle métier importante** sans intervention manuelle.

Optimisation de performance :