

LES LIVRABLES

Projet PGI – CosméBeauté

Mise en place d'un Progiciel de Gestion Intégré pour une PME de vente de cosmétiques en ligne

Sommaire :

1. Phase initiale

- 1.1 Cahier des charges
- 1.2 Étude de faisabilité
- 1.3 Plan de projet

2. Analyse

- 2.1 SFG
- 2.2 SFD
- 2.3 Spécifications techniques
- 2.4 Modèles UML

3. Conception

- 3.1 Conception architecturale
- 3.2 Maquettes / Prototypes

4. Développement & Test

- 4.1 Documentation technique
- 4.2 Plan de tests
- 4.3 Rapports de tests

Mise en place d'un Progiciel de Gestion Intégré pour une PME de vente de cosmétiques en ligne

Modules : Ventes – Achats – Stocks – RH

1. Phase initiale

1.1 Cahier des charges

Phrase clé : Identifier les besoins et les contraintes du système.

Livrable : Cahier des charges fonctionnel.

Explication : Ce document recueille les attentes des utilisateurs, les fonctionnalités souhaitées et les contraintes techniques, organisationnelles ou budgétaires.

Contenu adapté au projet :

Objectif :

Mettre en place un PGI centralisé pour gérer toutes les activités de CosméBeauté (ventes en ligne, gestion des stocks, approvisionnement et ressources humaines).

Besoins exprimés :

- Suivi en temps réel des ventes sur le site e-commerce.
- Mise à jour automatique des stocks après chaque commande.
- Centralisation et automatisation des commandes fournisseurs.
- Gestion simplifiée des employés (contrats, paie, congés).
- Génération automatique de factures et de tableaux de bord.

Contraintes :

- Budget limité → nécessité d'utiliser un PGI open source ou low-cost.
- Intégration obligatoire avec le site e-commerce existant (Prestashop).
- Respect strict du RGPD pour les données clients et salariés.

1.2 Étude de faisabilité

Phrase clé : Évaluer la faisabilité technique, économique et organisationnelle du projet.

Livrable : Rapport de faisabilité.

Explication : Analyse si le projet peut être réalisé compte tenu des moyens, compétences, coûts et contraintes de la PME.

Contenu adapté :

Faisabilité technique :

- ❖ Utilisation d'un PGI open source (Odoo ou Dolibarr).
- ❖ Connecteurs compatibles avec Prestashop pour synchroniser les commandes et les stocks.
- ❖ Hébergement sur un serveur Linux simple (Ubuntu Server).

Faisabilité économique :

- ❖ Coût logiciel et paramétrage ≈ 2 000 €.
- ❖ Formation réalisée en interne pour limiter les coûts.
- ❖ Retour sur investissement estimé en 1 an (réduction d'erreurs, gain de temps).

Faisabilité organisationnelle :

- ❖ 2 employés administratifs impliqués dans le paramétrage et les tests.
- ❖ Formation en interne sur l'utilisation du PGI.
- ❖ Légère adaptation des processus actuels (ventes, achats, stocks, RH).

Conclusion :

Le projet est techniquement réalisable, économiquement rentable et organisationnellement acceptable pour CosméBeauté.

1.3 Plan de projet

Phrase clé : Planifier les étapes, les ressources, les coûts et les risques du projet.

Livrable : Plan de gestion de projet.

Explication : Document de pilotage qui fixe le calendrier, les tâches, le budget et les responsables.

Contenu adapté :

Contenu adapté :

Étape	Durée	Responsable	Livrable
Analyse des besoins	2 semaines	Chef de projet	Cahier des charges
Étude de faisabilité	1 semaine	Équipe projet	Rapport de faisabilité
Conception & choix du PGI	3 semaines	Technicien SI	Schémas & prototypes
Développement / paramétrage	4 semaines	Développeur	Modules PGI configurés
Tests et validation	2 semaines	Utilisateurs clés	Rapport de validation
Mise en production	1 semaine	Chef de projet	Manuel / guide d'installation

2. Analyse

2.1 Spécifications fonctionnelles générales (SFG)

Phrase clé : Décrire globalement les fonctionnalités du système.

Livrable : Document de SFG.

Explication : Vue d'ensemble des fonctionnalités principales du PGI, sans entrer dans les détails techniques.

Contenu adapté :

Le PGI CosméBeauté doit permettre :

- ✓ De gérer les ventes en ligne (site e-commerce) et éventuellement en boutique.
- ✓ De suivre les stocks avec les entrées, sorties, inventaires et alertes de seuil.
- ✓ De gérer les achats fournisseurs : commandes, réceptions, factures.
- ✓ De suivre les ressources humaines : dossiers salariés, paie simplifiée, congés.

- ✓ De fournir un tableau de bord global (ventes, stocks, achats, RH).

2.2 Spécifications fonctionnelles détaillées (SFD)

Phrase clé : Détailler chaque fonctionnalité avec les règles de gestion et scénarios.

Livrable : Document de SFD.

Explication : Sert de référence aux développeurs, avec cas d'utilisation et règles métier.

Contenu adapté :

Module Ventes :

- ✓ Règle : la commande n'est validée qu'après confirmation du paiement.
- ✓ Cas d'usage : un client commande un coffret de soins → le stock est mis à jour → une facture est générée automatiquement.

Module Achats :

- ✓ Règle : une commande fournisseur est déclenchée dès que le stock passe sous le seuil minimum.
- ✓ Cas d'usage : le stock d'un sérum passe sous le seuil → le système propose une commande fournisseur.

Module Stocks :

- ✓ Règle : tout retour client doit être réintégré manuellement après contrôle qualité.
- ✓ Cas d'usage : un produit retourné abîmé n'est pas remis en stock mais enregistré comme perte.

Module RH :

- ✓ Règle : chaque salarié a un identifiant unique ; les absences doivent être validées par le responsable hiérarchique.
- ✓ Cas d'usage : un salarié pose des congés → demande envoyée → validation par responsable → mise à jour du solde de congés.

2.3 Spécifications techniques

Phrase clé : Définir l'architecture, les technologies et les choix techniques.

Livrable : Dossier technique.

Explication : Contient les décisions sur les langages, la base de données, la sécurité et les performances.

Contenu adapté :

- PGI choisi : Odoo (version communautaire).
- Langage : Python.
- Base de données : SQL.
- Serveur : Ubuntu Server hébergé chez OVH.
- Sécurité : HTTPS, pare-feu, authentification par mot de passe fort, sauvegarde automatique quotidienne des données.

2.4 Modèles UML

Phrase clé : Représenter graphiquement les éléments du système et leurs interactions.

Livrable : Diagrammes UML (cas d'utilisation, classes, séquence).

Explication : Aident à comprendre le fonctionnement fonctionnel et technique avant le développement.

Contenu adapté :

Diagramme de cas d'utilisation :

Acteurs : Client, Ventes, Magasinier, Acheteur, RH, Comptable, Admin.

UC majeurs : Gérer catalogue, Passer commande, Payer, Traiter commande, Réception achat, Gérer stocks, Gérer retours, Export paie, Consulter KPI.

Diagramme de cas d'utilisation

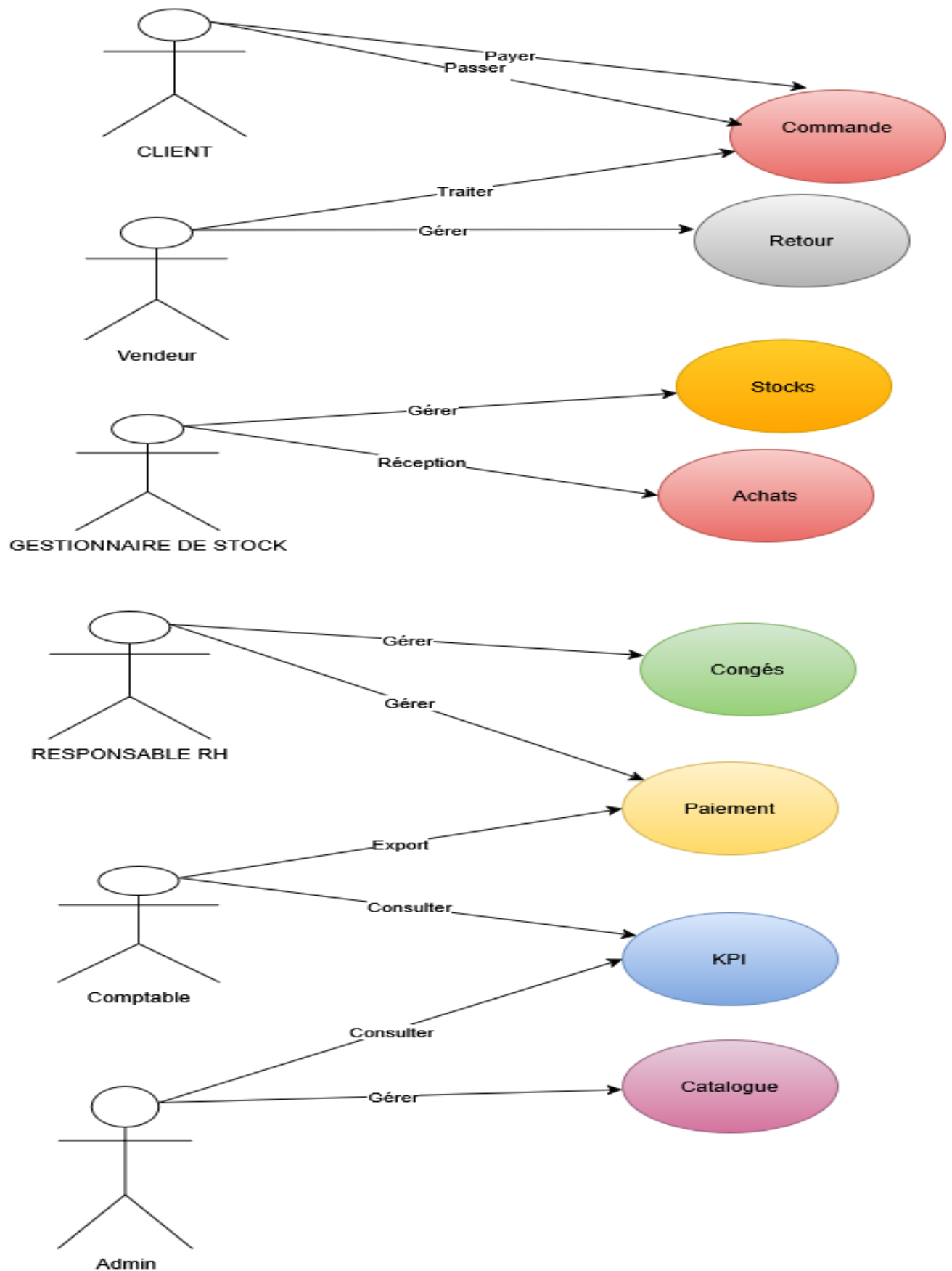
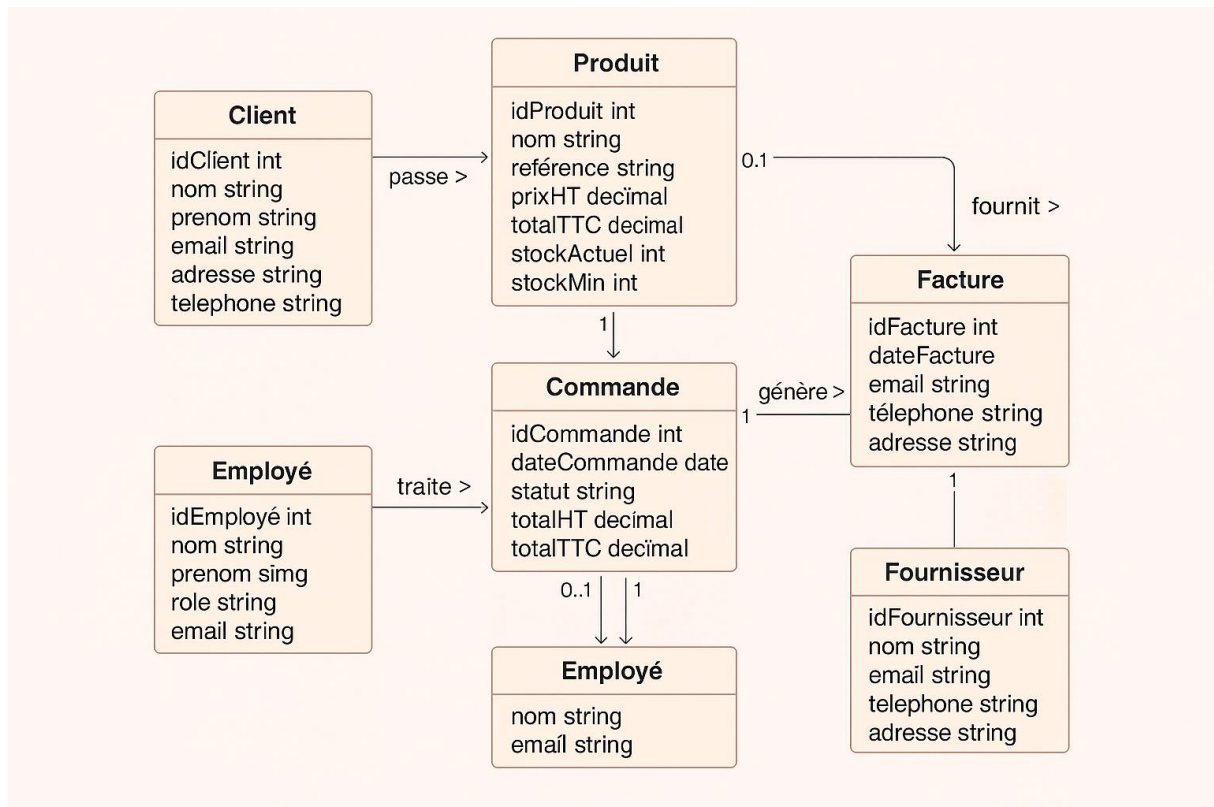


Diagramme de classes :

Classes principales : Produit, Commande, Client, Employé, Fournisseur, Facture.

DIAGRAMME DE CLASSE



3. Conception

3.1 Conception architecturale

Phrase clé : Définir l'architecture globale du système.

Livrable : Schéma d'architecture logique et physique.

Explication : Décrit la répartition des composants et la manière dont ils communiquent.

Contenu adapté :

Architecture logique :

Front-end : site e-commerce Prestashop.

Back-end : PGI Odoo.

Base de données PostgreSQL.

Communication : API REST entre le site Prestashop et Odoo.

Sécurité : pare-feu, accès restreints à l'interface d'administration, sauvegardes quotidiennes.

3.2 Maquettes / Prototypes

Phrase clé : Préparer visuellement l'interface utilisateur avant le développement.

Livrable : Maquettes d'écrans, prototypes (par exemple sur Figma).

Explication : Permettent de valider l'ergonomie et le parcours utilisateur.

Contenu adapté :

Page d'accueil avec catalogue de produits (cosmétiques, soins, etc.).

Interface employé pour la saisie des ventes et la gestion des commandes.

Tableau de bord RH (vue des salariés, absences, paie).

Écran de gestion de stock : ajout / retrait de produits, inventaire, alertes de seuil.

4. Développement & Test

4.1 Documentation technique

Phrase clé : Fournir un guide de développement et d'intégration.

Livrable : Documentation développeur (API, classes, services).

Explication : Sert à comprendre la structure du code et à faciliter la maintenance.

Contenu adapté :

Structure des modules Odoo.

Description des modèles : product.py, sale.py, employee.py, etc.

Routes API principales : /api/orders, /api/stocks, /api/employees.

Procédure d'installation et de mise à jour des modules personnalisés.

4.2 Plan de tests

Phrase clé : Définir les stratégies de tests du système.

Livrable : Plan de test (tests unitaires, intégration, performance, sécurité).

Explication : Détaille les types de tests à réaliser et les critères d'acceptation.

Contenu adapté :

Tests unitaires : vérifier chaque module (ventes, achats, stocks, RH).

Tests d'intégration : vérifier les flux complets (ex : commande client → mise à jour stock → écriture dans PGI).

Tests de performance : temps de réponse < 3 secondes avec 50 requêtes simultanées.

Tests de sécurité : contrôle des droits par rôle, gestion des mots de passe, accès restreints à l'administration.

4.3 Rapports de tests

Phrase clé : Présenter les résultats des tests et les anomalies corrigées.

Livrable : Rapport de validation.

Explication : Fournit un bilan sur la qualité du système testé avant mise en production.

Contenu adapté :

Module	Résultat	Anomalies détectées	Correction
Ventes	✓ Fonctionnel	Erreur sur calcul de remise	Script corrigé
Achats	✓ Fonctionnel	Lenteur sur la requête fournisseur	Optimisation SQL
Stocks	✓ Fonctionnel	Mauvaise mise à jour sur retour produit	Ajout d'une règle de gestion
RH	✓ Fonctionnel	Aucune	—