



Rapport de projet

3ème année

Ingénierie Informatique et Réseaux

Sous le thème

ProShop

Réalisé par :
BOULAHBACH Zineb , BOUARFA Aymane

Encadré par :

Tuteur de l'école : Prof. Khalid NAFIL

ANNEE UNIVERSITAIRE: 2024-2025

Remerciements

Je souhaite adresser mes plus vifs remerciements à Monsieur Khalid NAFIL, notre encadrant, pour son accompagnement constant, sa bienveillance et la richesse de ses conseils tout au long de ce projet. Sa disponibilité, son professionnalisme et son implication ont été des atouts précieux qui ont fortement contribué à la réussite de ce travail, ainsi qu'à mon développement personnel et académique. J'adresse également ma reconnaissance à l'ensemble de mes enseignants ainsi qu'à mes camarades pour leur soutien et les échanges constructifs qui ont marqué cette expérience.

Dédicaces

Je dédie ce travail à :

- **Mes parents**, pour leur amour inestimable, leur soutien indéfectible et les nombreux sacrifices consentis afin de me permettre de progresser dans les meilleures conditions.
- **Ma famille et mes proches**, dont le réconfort et la présence ont été d'un grand appui tout au long de ce parcours.
- Toutes **les personnes qui croient en moi et qui**, par leur confiance et leurs encouragements, me poussent à me dépasser chaque jour.

Résumé

Le projet **ProShop** vise à développer une plateforme e-commerce moderne, intuitive et sécurisée, répondant aux besoins actuels des utilisateurs pour une expérience d'achat en ligne complète. Conçu avec une architecture robuste, le système permet une gestion efficace des produits, des catégories, des stocks, ainsi qu'un processus de commande fluide et sécurisé.

La solution technique repose sur Django pour le backend, offrant une base fiable et évolutive, et sur une interface frontend responsive qui assure une navigation agréable. La base de données SQLite utilisée en phase de développement peut évoluer vers PostgreSQL en production. Le déploiement est facilité par l'utilisation de Gunicorn et d'un Procfile, avec une organisation optimisée des fichiers statiques.

Le projet **ProShop** est structuré de manière modulaire, séparant clairement frontend et backend, ce qui favorise la maintenance et les évolutions futures. Les résultats obtenus démontrent une plateforme performante et prête à intégrer des fonctionnalités avancées telles que le paiement en ligne, la recherche améliorée, ou un système de recommandation.

Ainsi, **ProShop** constitue une base solide répondant aux exigences du marché e-commerce, avec un fort potentiel d'extension.

Abstract

The **ProShop** project aims to develop a modern, intuitive, and secure e-commerce platform that meets current user needs for a comprehensive online shopping experience. Built on a robust architecture, the system enables efficient management of products, categories, inventory, and provides a smooth, secure ordering process.

The technical solution uses Django for the backend, offering a reliable and scalable foundation, and a responsive frontend interface for seamless navigation. The development phase uses SQLite as the database, with the option to migrate to PostgreSQL for production. Deployment is streamlined using Gunicorn and a Procfile, with optimized static file management.

ProShop is designed with a modular structure, clearly separating frontend and backend to facilitate maintenance and future enhancements. The platform delivers a high-performance solution ready to integrate advanced features such as online payment, enhanced search, and recommendation systems.

Overall, ProShop provides a solid base that addresses e-commerce market demands while offering strong potential for future development.

Table des Matières

Introduction

- Contexte général
 - Problématique
 - Objectifs du projet
 - Méthodologie adoptée
 - Structure du rapport
-

Chapitre 1 : Cahier des Charges

- 1.1. Contexte et définition
 - 1.2. Problématique
 - 1.3. Objectifs du projet
 - 1.4. Périmètre et exclusions
 - 1.5. Parties prenantes
 - 1.6. Besoins fonctionnels et non fonctionnels
 - 1.7. Contraintes techniques
 - 1.8. Technologies et outils
-

Chapitre 2 : Analyse et Conception

- 2.1. Démarche de conception orientée objet
- 2.2. Présentation des entités principales
- 2.3. Diagramme de classes UML
- 2.4. Diagrammes de cas d'utilisation
- 2.5. Diagrammes de séquence
- 2.6. Diagrammes de séquence :

Chapitre 3 : Réalisation

- 3.1. Architecture technique
 - 3.2. Structure de la base de données
 - 3.3. Fonctionnalités principales
 - Authentification
 - Partage de projets
 - Recherche de développeurs
 - Notation/commentaires
 - Messagerie interne
 - 3.4. Sécurité et gestion des droits (User/Admin)
 - 3.5. Interface utilisateur (UX/UI)
-

Chapitre 4 : Déploiement et Tests

- 4.1. Environnement de déploiement
 - 4.2. Procédure d'installation
 - 4.3. Tests réalisés :
 - Unitaires
 - Fonctionnels
 - Performances
-

Chapitre 5 : Évaluation et Perspectives

- 5.1. Points forts et limites du projet
 - 5.2. Difficultés rencontrées
-

Conclusion Générale

- Bilan global
-

Bibliographie / Références

- Liens vers doc Django, PostgreSQL, JWT, etc.

Introduction

Contexte général

Dans un contexte où le commerce en ligne connaît une croissance rapide et continue, les consommateurs sont de plus en plus exigeants quant à la qualité de leur expérience d'achat. Ils recherchent des plateformes capables de leur offrir une navigation fluide, un large choix de produits présentés de manière attractive, ainsi qu'un service fiable et sécurisé. Pourtant, malgré cette demande croissante, de nombreuses solutions e-commerce présentent encore des faiblesses majeures : interfaces complexes, parcours d'achat peu intuitifs, gestion des stocks insuffisante, ou encore absence de personnalisation de l'expérience utilisateur.

Les plateformes existantes sont souvent soit trop compliquées à utiliser pour les petits commerçants, soit trop limitées pour satisfaire pleinement les attentes des consommateurs modernes. Elles peinent à concilier efficacité dans la gestion des produits et des commandes, simplicité d'utilisation, et sécurisation du processus d'achat.

C'est dans ce cadre que le projet ProShop a été conçu. Il s'agit d'une application web e-commerce moderne et intuitive, pensée pour répondre aux besoins des commerçants comme des consommateurs. ProShop centralise la gestion des produits, des commandes, des utilisateurs et des interactions, afin de garantir une expérience d'achat optimale et personnalisée.

ProShop se distingue par ses atouts majeurs :

- Une interface utilisateur moderne, claire et facile à prendre en main
- Une gestion simplifiée des produits et des stocks
- Un processus d'achat sécurisé et fiable
- Une navigation fluide facilitant la recherche et la découverte de produits
- Une organisation efficace des commandes

Cette solution technique répond aux défis actuels du marché e-commerce tout en posant les bases pour des évolutions futures, telles que l'intégration de systèmes de paiement avancés, des fonctionnalités de personnalisation, ou des outils d'analyse du comportement d'achat.

Ainsi, ProShop incarne une réponse concrète aux exigences du commerce en ligne moderne, en proposant une plateforme complète, évolutive et centrée sur la satisfaction de l'utilisateur.

Problématique

Dans l'écosystème actuel du commerce en ligne, commerçants et consommateurs sont confrontés à des défis majeurs qui affectent l'efficacité des plateformes e-commerce existantes. Ces dernières présentent souvent des limites importantes qui impactent négativement l'expérience utilisateur ainsi que la gestion commerciale. Qu'elles soient trop complexes ou trop simplifiées, ces plateformes ne répondent pas pleinement aux besoins spécifiques du marché moderne, ce qui engendre plusieurs difficultés :

- Une gestion inefficace et peu réactive des catalogues produits et des stocks en temps réel pour les commerçants,
- Des interfaces utilisateurs peu ergonomiques, rendant l'expérience d'achat difficile et frustrante pour les consommateurs,
- Un manque de personnalisation dans la présentation des produits et les recommandations,
- Des processus de commande complexes et parfois insuffisamment sécurisés,
- Une gestion insuffisante des retours et du service client.

Ces lacunes se traduisent par une baisse du taux de conversion, une insatisfaction grandissante des clients, une difficulté à fidéliser la clientèle, une perte de confiance dans le commerce en ligne, ainsi qu'une perte de compétitivité pour les commerçants.

Face à ces enjeux, il devient impératif de proposer une plateforme e-commerce moderne, intuitive et centrée sur l'expérience utilisateur. C'est dans ce cadre que s'inscrit le projet ProShop, qui vise à :

- Simplifier la gestion des produits et des stocks,
- Proposer une interface utilisateur claire, moderne et facile à utiliser,
- Sécuriser l'ensemble du processus d'achat,
- Faciliter la navigation et la recherche des produits

- Offrir une gestion performante des commandes et du service client.

Cette solution technique répond aux exigences actuelles du marché e-commerce tout en ouvrant la voie à des évolutions futures, telles que l'intégration de systèmes de paiement avancés, des fonctionnalités de personnalisation accrues, ou des outils d'analyse du comportement d'achat.

Ainsi, ProShop constitue une réponse adaptée aux défis du commerce en ligne moderne en proposant une plateforme qui centralise la gestion des produits et des commandes, optimise l'expérience utilisateur, renforce la confiance dans le parcours d'achat, facilite le développement commercial et permet une évolution continue des fonctionnalités.

Cette approche vise à créer un écosystème e-commerce plus efficace, sécurisé et en phase avec les attentes des utilisateurs contemporains, tout en offrant aux commerçants les outils nécessaires pour croître durablement en ligne.

Objectifs du projet

Le projet ProShop a pour objectif principal de concevoir et développer une plateforme e-commerce moderne et intuitive, offrant une expérience d'achat optimale aux consommateurs tout en facilitant la gestion commerciale pour les vendeurs. Les objectifs spécifiques du projet sont les suivants :

1. Créer une plateforme centralisée de commerce en ligne permettant :

- Une présentation claire et attractive des produits
- Une gestion efficace des catégories et sous-catégories
- Un système de recherche et de filtrage performant
- Une gestion optimisée des stocks en temps réel

1. Faciliter l'expérience d'achat pour les consommateurs en proposant :

- Une interface utilisateur intuitive et responsive
- Un processus de commande simplifié et sécurisé
- Un panier d'achat flexible et facile à gérer
- Un système de suivi des commandes en temps réel

2. Assurer une gestion commerciale efficace pour les vendeurs avec :

- Un tableau de bord administratif complet
- Des outils de gestion des produits et des stocks
- Un système de gestion des commandes
- Des rapports et statistiques de vente

3. Renforcer la confiance et la sécurité avec :

- Un système d'authentification robuste
- Une sécurisation des paiements
- Une protection des données personnelles
- Une gestion sécurisée des transactions

4. Optimiser les performances et la scalabilité grâce à :

- Une architecture technique moderne (Django, SQLite/PostgreSQL)
- Une gestion efficace des ressources statiques
- Un système de cache performant
- Une infrastructure prête pour le déploiement cloud

Méthodologie adoptée

Pour mener à bien le développement de la plateforme ProShop, une méthodologie orientée objet et centrée sur l'utilisateur a été adoptée, combinant des principes d'analyse UML avec une approche agile de gestion du projet. Cette méthodologie permet de garantir une meilleure adaptation aux besoins évolutifs du client et une livraison progressive des fonctionnalités.

1. Analyse des Besoins

Une phase initiale d'analyse approfondie a été réalisée afin de :

- Identifier les fonctionnalités essentielles (gestion des produits, panier d'achat, authentification, commandes, etc.)
- Définir les besoins non fonctionnels (performance, sécurité, fiabilité)
- Recueillir les exigences des utilisateurs finaux (interface intuitive, expérience fluide)
- Prendre en compte les contraintes techniques (scalabilité, portabilité, facilité de maintenance)

2. Architecture Technique

Le projet repose sur une architecture technique moderne et modulaire :

- Backend : Développé avec le framework Django (Python)
- Frontend : Interface responsive, conçue pour une navigation fluide
- Base de données : SQLite utilisée pendant le développement, avec possibilité de migrer vers PostgreSQL en production
- Gestion des ressources statiques : Intégration optimisée des images, CSS, JavaScript
- Déploiement : Préparation via Gunicorn, Procfile, et une configuration adaptée à un hébergement cloud (type Heroku)

Pour mener à bien le développement de la plateforme ProShop, une méthodologie orientée objet et centrée sur l'utilisateur a été adoptée, combinant des principes d'analyse UML avec une approche agile de gestion du projet. Cette méthodologie permet de garantir une meilleure adaptation aux besoins évolutifs du client et une livraison progressive des fonctionnalités.

1. Analyse des Besoins

Une phase initiale d'analyse approfondie a été réalisée afin de :

- Identifier les fonctionnalités essentielles (gestion des produits, panier d'achat, authentification, commandes, etc.)
- Définir les besoins non fonctionnels (performance, sécurité, fiabilité)
- Recueillir les exigences des utilisateurs finaux (interface intuitive, expérience fluide)
- Prendre en compte les contraintes techniques (scalabilité, portabilité, facilité de maintenance)

2. Architecture Technique

Le projet repose sur une architecture technique moderne et modulaire :

- Backend : Développé avec le framework Django (Python)
- Frontend : Interface responsive, conçue pour une navigation fluide
- Base de données : SQLite utilisée pendant le développement, avec possibilité de migrer vers PostgreSQL en production
- Gestion des ressources statiques : Intégration optimisée des images, CSS, JavaScript
- Déploiement : Préparation via Gunicorn, Procfile, et une configuration adaptée à un hébergement cloud (type Heroku)

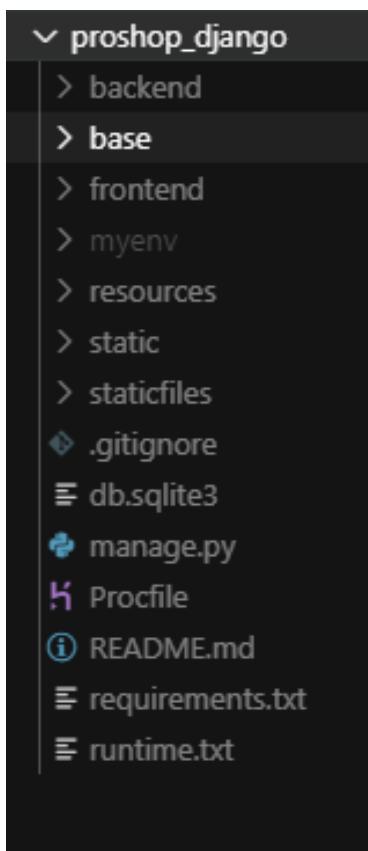
3. Approche Modulaire

Le développement est organisé autour de plusieurs modules indépendants :

- Module d'authentification : Inscription, connexion, gestion des sessions
- Module de gestion des produits : Ajout, mise à jour, suppression, catégorisation
- Module de panier et commandes : Ajout au panier, validation de commande, historique
- Module d'administration : Interface réservée à la gestion globale du site
- Module des fichiers statiques : Centralisation des ressources CSS, JS et médias

4. Structure du Projet

La structure du projet suit une organisation claire et évolutive :



5. Gestion des Dépendances

- Utilisation de requirements.txt pour la gestion des bibliothèques Python
- Création d'un environnement virtuel (myenv) pour isoler le projet
- Versionnage avec Git, assurant un suivi rigoureux des modifications

6. Tests et Validation

- Tests unitaires : vérification du bon fonctionnement des fonctions clés
- Tests d'intégration : interaction entre les modules (ex. panier/commande)
- Validation de l'interface utilisateur : fluidité de navigation, compatibilité mobile
- Tests de performance et sécurité : protection des données, rapidité d'exécution

7. Déploiement

- Configuration avec Procfile pour l'exécution via Gunicorn
- Gestion des variables d'environnement (Django secret key, base de données, etc.)
- Collecte et optimisation des fichiers statiques (collectstatic)
- Préparation pour déploiement sur une plateforme cloud (ex. Heroku)

Cette méthodologie a permis de :

- Maintenir une structure de code claire et maintenable
- Faciliter l'évolution du projet et l'ajout de nouvelles fonctionnalités
- Garantir la qualité et la robustesse de l'application
- Optimiser les performances de la plateforme
- Assurer un déploiement sécurisé et facilement reproductible

Grâce à cette approche, le projet ProShop bénéficie d'une base solide, évolutive et conforme aux bonnes pratiques du développement web moderne.

Structure du rapport

Ce rapport est structuré en plusieurs chapitres, permettant de suivre de manière progressive la démarche de conception, de réalisation et d'évaluation du projet **DevSearch** :

- **Chapitre 1 – Cahier des Charges** : présente le contexte du projet, la problématique, les objectifs, les parties prenantes, ainsi que les besoins fonctionnels et non fonctionnels.
- **Chapitre 2 – Analyse et Conception** : expose les choix de modélisation orientée objet à travers des diagrammes UML (classes, cas d'utilisation, séquences), ainsi que les principales entités du système.
- **Chapitre 3 – Réalisation** : décrit l'architecture technique mise en place, les technologies utilisées, et le développement des différentes fonctionnalités de la plateforme.
- **Chapitre 4 – Déploiement et Tests** : détaille les étapes d'installation, la configuration du serveur, ainsi que les différents types de tests réalisés pour garantir la stabilité et la performance de l'application.
- **Chapitre 5 – Évaluation et Perspectives** : dresse un bilan du projet, évoque les difficultés rencontrées, les points d'amélioration et les perspectives d'évolution futures.

Enfin, une **conclusion générale** viendra clore le rapport, suivie des annexes (diagrammes UML, captures d'écran, etc.).

Chapitre 1 :

Cahier des Charges

1.1. Contexte et définition

Dans un monde où le commerce en ligne connaît une croissance exponentielle, les consommateurs et les commerçants recherchent des solutions e-commerce modernes, sécurisées et intuitives. Le marché actuel présente un besoin croissant de plateformes d'achat en ligne qui offrent une expérience utilisateur optimale tout en facilitant la gestion commerciale.

: Or, il n'existe pas aujourd'hui de plateforme e-commerce suffisamment complète et adaptée aux besoins modernes pour :

- Offrir une expérience d'achat fluide et intuitive
- Gérer efficacement les produits et les stocks
- Sécuriser les transactions et les données utilisateurs
- Faciliter la gestion des commandes et du service client
- Personnaliser l'expérience d'achat

ProShop est une plateforme e-commerce moderne et complète, conçue pour répondre aux besoins des commerçants et des consommateurs. Elle propose une gestion efficace des produits, un processus d'achat simplifié et sécurisé, ainsi qu'une interface intuitive et responsive.

Le projet repose sur une architecture modulaire et évolutive, intégrant des technologies modernes telles que Django, SQLite/PostgreSQL et Gunicorn pour le déploiement. Il met l'accent sur la performance, la sécurité des transactions, et l'optimisation de l'expérience utilisateur. ProShop centralise la gestion des produits, commandes et stocks, tout en offrant un panier flexible, un suivi en temps réel et une personnalisation de l'expérience d'achat. Il constitue ainsi une réponse concrète aux enjeux actuels du commerce en ligne.

1.2. Problématique

Dans l'univers du commerce en ligne, de nombreux commerçants lancent des boutiques numériques pour proposer leurs produits à un large public. Cependant, ces initiatives rencontrent souvent des freins liés à des outils peu adaptés, limités ou complexes à utiliser. Les plateformes e-commerce actuelles, bien qu'établies, ne permettent pas toujours de :

- Offrir une expérience d'achat fluide, intuitive et personnalisée,
- Gérer les stocks et les commandes de manière centralisée et en temps réel,
- Garantir une sécurité optimale des transactions et des données clients,
- Proposer une interface simple, responsive et accessible sur tous les appareils,
- Faciliter les échanges entre commerçants et acheteurs pour un service client réactif.

Par conséquent, il devient difficile pour un commerçant :

- De gérer efficacement son catalogue de produits,
- De suivre ses ventes et son inventaire avec précision,
- De fidéliser ses clients à travers une expérience utilisateur de qualité,
- De sécuriser les paiements et les informations sensibles.

Et pour un consommateur :

- De trouver facilement des produits pertinents et bien présentés,
- D'avoir confiance dans la sécurité de la plateforme,
- De vivre une expérience d'achat agréable et fluide,
- De recevoir un support rapide en cas de problème.

C'est pour répondre à ces défis que le projet ProShop a été conçu : il vise à développer une plateforme e-commerce moderne, évolutive et sécurisée, permettant aux commerçants de gérer efficacement leur activité en ligne tout en offrant aux consommateurs une expérience d'achat optimale, transparente et conviviale.

1.3. Objectifs du projet

Le projet ProShop a pour finalité de concevoir une plateforme e-commerce complète, moderne et intuitive, destinée à faciliter les échanges entre commerçants et consommateurs. Elle vise à améliorer la gestion des activités commerciales en ligne tout en garantissant une expérience d'achat fluide, sécurisée et personnalisée.

◆ **Objectif général :**

Mettre en place une application web centralisée, performante et évolutive, permettant aux commerçants de gérer efficacement leurs produits, commandes et ventes, tout en offrant aux utilisateurs finaux une interface conviviale et sécurisée pour leurs achats en ligne.

◆ **Objectifs spécifiques :**

- Permettre l'inscription, la connexion et la gestion des comptes (commerçants et clients), via un système d'authentification sécurisé et fiable.
 - Offrir aux commerçants une interface de gestion complète pour :
 - Créer, modifier et organiser leur catalogue de produits,
 - Gérer les stocks en temps réel,
 - Suivre les commandes et gérer les livraisons,
 - Accéder à des tableaux de bord de suivi et d'analyse des ventes.
 - Permettre aux clients de :
 - Créer un profil personnalisé,
 - Consulter leur historique d'achats,
 - Gérer leurs adresses de livraison,
 - Créer des listes de souhaits.
 - Intégrer des fonctionnalités essentielles telles que :
 - Un système de panier d'achat flexible,
 - Un module de paiement sécurisé,

- Un moteur de recherche avancé avec filtres,
 - Un système de notation et d'avis pour les produits,
 - La gestion des réductions, promotions et codes promo.
- Assurer une interface responsive, moderne et adaptée à tous les types d'appareils (ordinateurs, tablettes, smartphones).
 - Garantir la protection des données personnelles et la sécurité des transactions financières.
 - Mettre en place un système de notifications (commandes, expéditions, promotions, etc.).
 - Intégrer un module de gestion des retours et des réclamations clients.
 - Déployer la plateforme sur un environnement cloud, assurant haute disponibilité, scalabilité et performance.
 - Fournir une interface d'administration pour :
 - La modération des contenus (avis, produits),
 - La gestion des utilisateurs (clients et commerçants),
 - Le suivi global de l'activité de la plateforme.
- En atteignant ces objectifs, ProShop entend répondre efficacement aux besoins actuels du commerce en ligne, en créant une solution fiable, évolutive et accessible à tous les acteurs du e-commerce.

1.4. Périmètre et exclusions

◆ 1.4.1. Périmètre du projet

Le projet ProShop porte sur la conception et le développement d'une application web e-commerce responsive, accessible depuis un navigateur, destinée à répondre aux besoins des commerçants et des consommateurs.

Les fonctionnalités principales incluses dans le périmètre sont :

 Pour les commerçants :

- Inscription, connexion et authentification sécurisée avec gestion de rôles
- Création et gestion du catalogue de produits (ajout, modification, suppression)
- Suivi des stocks en temps réel
- Gestion des commandes, livraisons et retours
- Tableau de bord de suivi des performances commerciales

 Pour les consommateurs :

- Inscription et gestion des comptes clients
- Navigation fluide et moteur de recherche pour les produits
- Gestion du panier d'achat et suivi des commandes
- Accès à l'historique d'achats et gestion des adresses
- Système d'avis et de notation des produits

 Fonctionnalités générales :

- Paiement en ligne sécurisé
- Gestion des promotions, réductions et codes promo
- Interface d'administration pour la gestion des utilisateurs, produits et contenus
- Système de notifications (commandes, expéditions, etc.)
- Hébergement cloud garantissant performance, accessibilité et scalabilité

◆ 1.4.2. Exclusions

Afin de respecter les délais, les ressources et se concentrer sur les fonctionnalités essentielles de la V1, les éléments suivants sont hors périmètre :

🚫 Développement mobile :

- Création d'une application mobile native (Android/iOS) — seule la version web responsive est prévue.

🚫 Fonctionnalités avancées :

- Marketplace multi-vendeurs
- Gestion de programmes de fidélité complexes
- Dropshipping ou automatisation de la chaîne logistique
- Multi-devises et multilingue (version en langue unique pour la V1)

🚫 Intégrations externes :

- Connexion avec des ERP ou CRM tiers
- Intégration à des marketplaces externes (Amazon, eBay, etc.)
- Outils d'analyse de comportement utilisateur complexes
- Outils de marketing automation ou campagnes publicitaires

1.5. Parties prenantes

Le succès du projet ProShop repose sur l'implication et la coordination de plusieurs acteurs clés, chacun ayant un rôle bien défini dans la conception, le développement, l'exploitation ou l'utilisation de la plateforme.

◆ 1.5.1. Les Commerçants (Vendeurs)

- **Rôle** : Utilisateurs principaux de la plateforme côté vente.
- **Attentes** : Gérer efficacement leur catalogue de produits, suivre leurs ventes, optimiser leur présence en ligne.
- **Impact** : Leur satisfaction et leur engagement sont déterminants pour la qualité et la diversité de l'offre sur la plateforme.

◆ 1.5.2. Les Consommateurs(Clients)

- **Rôle** : Utilisateurs finaux de la plateforme.
- **Attentes** : Trouver facilement des produits, bénéficier d'une expérience d'achat fluide et sécurisée, accéder à un service client réactif.
- **Impact** : Leur expérience et leur fidélité sont essentielles à la croissance et à la pérennité du projet.

◆ 1.5.3. L'Équipe de Développement

- **Rôle** : Concevoir, développer, tester et déployer la solution technique.
- **Attentes** :
 - Définir les choix technologiques et l'architecture logicielle
 - Réaliser la modélisation UML
 - Garantir la sécurité, la performance et la maintenance du système
- **Impact** : Développeur principal, contributeurs techniques, encadrant pédagogique.

◆ 1.5.4. L'Administrateur de la Plateforme

- **Rôle :** Superviser et maintenir le bon fonctionnement de la plateforme.
- **Fonctions :**
 - Gérer les transactions et les utilisateurs
 - Superviser les litiges et retours
 - Garantir la sécurité des données et la qualité du service
 - Gérer les réclamations et le support client

◆ 1.5.5. Les Prestataires de Services

- **Rôle :** Fournir des services complémentaires indispensables au bon fonctionnement de la plateforme.
- **Services concernés:**
 - Paiement en ligne
 - Livraison des produits
 - Support technique ou logistique
 - Maintenance des services tiers

◆ 1.5.6. Les investisseurs ou partenaires potentiels (à long terme)

- **Role:** Suivre l'avancement du projet et valider les choix techniques réalisés.
- **Objectif:**
 - Veiller au respect des objectifs pédagogiques et à la qualité globale du travail rendu.

1.6. Besoins fonctionnels et non fonctionnels

◆ 1.6.1. Besoins fonctionnels

Les besoins fonctionnels décrivent les fonctionnalités que la plateforme ProShop doit offrir à ses utilisateurs. Ils traduisent les actions essentielles à réaliser selon les objectifs du projet et les interactions prévues avec le système.

Contraintes / Règles de gestion

Utilisation du framework Django
API REST avec authentification JWT
Validation des produits par l'administrateur avant publication

Fonctionnalité	Description courte	Priorité	Contraintes clés
Inscription/Auth	Création et connexion	Haute	JWT, rôles
Profils	Modifier infos et avatar	Haute	Validation vendeur
Produits	Ajouter/modifier/supprimer	Haute	Max 100 produits/vendeur
Panier	Gérer contenu et quantités	Haute	Stock temps réel
Commande	Valider panier, paiement	Haute	Paiement sécurisé
Recherche	Recherche par filtres	Haute	Full-text PostgreSQL
Avis	Noter et commenter	Moyenne	Modération
Suivi commande	Statut et historique	Haute	Notifications
Retours	Demande et suivi	Moyenne	Délai 14 jours
Modération (admin)	Validation contenus	Haute	Interface admin

◆ 1.6.2. Besoins non fonctionnels

Ces besoins concernent la qualité de service, la performance, la sécurité ainsi que l'environnement technique dans lequel l'application sera déployée.

Critère	Description courte	Priorité	Outils/Contraintes
Performance	Réponse < 2s	Haute	Django, Whitenoise
Sécurité	Protection des données	Haute	JWT, HTTPS, chiffrement
Scalabilité	Support croissance	Haute	Cloud AWS
Compatibilité	Navigateurs modernes	Haute	Tests cross-browser
Disponibilité	Accessible 24/7	Haute	≥ 99.9% uptime
Interface	Responsive, intuitive	Haute	Bootstrap/Tailwind CSS
Base de données	Gestion efficace	Haute	PostgreSQL optimisé
Paiement	Transactions sécurisées	Haute	Stripe/PayPal
Stockage	Images et fichiers	Moyenne	AWS S3
Monitoring	Suivi performances	Moyenne	Outils cloud

1.7. Contraintes techniques

Le développement de la plateforme ProShop doit respecter un ensemble de contraintes techniques afin d'assurer sa robustesse, sa sécurité et sa maintenabilité.

◆ Contraintes liées aux technologies imposées

- Framework principal : utilisation obligatoire de Django (version 5.0.6).
- Base de données : PostgreSQL pour sa performance et la recherche full-text.
- API : interface backend/frontend via Django REST Framework (DRF).
- Authentification : intégration de SimpleJWT pour la gestion sécurisée des tokens.
- Déploiement : hébergement sur cloud (ex. AWS) avec Gunicorn comme serveur et Whitenoise pour la gestion des fichiers statiques.

◆ Contraintes d'architecture et de performance

- Code source documenté, structuré, respectant les bonnes pratiques Python/Django.
- Utilisation d'un système de gestion de versions (GitHub) pour le suivi du projet.

1.8. Technologies et outils

Le développement de la plateforme ProShop utilise des technologies modernes et complémentaires, adaptées aux besoins d'une plateforme e-commerce performante et sécurisée.

◆ Langages et Frameworks

Backend

- **Python : langage principal, reconnu pour sa clarté et puissance.**
- **Django (v5.0.6) : framework pour backend, ORM, authentification et administration.**
- **Django REST Framework (DRF) : création d'API RESTful pour le frontend.**

Frontend

- **HTML5 / CSS3 / JavaScript : technologies standards pour l'interface.**
- **Bootstrap 5 : framework CSS responsive et moderne.**
- **React.js + Redux : interface utilisateur dynamique et gestion d'état.**

◆ Base de données

- PostgreSQL : base relationnelle performante avec recherche full-text.
- AWS S3 : stockage cloud pour images, fichiers statiques et documents.

◆ Authentification

- SimpleJWT : gestion des tokens d'authentification.
- Stripe / PayPal : paiements sécurisés.
- HTTPS et Django Security Middleware : protection des communications et contre les attaques.

◆ Déploiement et Hébergement

- Gunicorn : serveur d'application Python.
- Nginx : serveur web et reverse proxy.
- Whitenoise : gestion des fichiers statiques.
- AWS :
 - EC2 pour l'hébergement
 - S3 pour le stockage
 - RDS pour la base de données
 - CloudFront pour la distribution CDN

◆ Outils de développement

- Visual Studio Code / PyCharm : éditeurs/IDE.
- Git / GitHub : gestion et hébergement du code.
- Postman : tests API REST.
- Pytest / Selenium : tests unitaires et d'interface.
- PgAdmin / DBeaver : administration base de données.

◆ Modélisation et documentation

- PlantUML / Draw.io / Lucidchart : création de diagrammes UML et schémas.
- Markdown / Sphinx / Microsoft Word : rédaction et génération de documentation.

Chapitre 2 :

Analyse et Conception

2.1. Démarche de conception orientée objet

La conception de la plateforme ProShop s'appuie sur une approche orientée objet (OO) afin de modéliser les entités métier, leurs relations et interactions de manière claire, modulaire et évolutive. Cette démarche s'inspire des principes fondamentaux de l'OO ainsi que des bonnes pratiques UML.

2.1.1. Principes directeurs

1. Modularité :

- Découpage des fonctionnalités en modules indépendants (authentification, gestion des produits, commandes, panier, etc.)
- Chaque module est représenté par des classes cohérentes et faiblement couplées
- Exemple : la classe User gère l'authentification, tandis que Client et Admin encapsulent les rôles spécifiques

2. Réutilisabilité :

- Conception d'entités communes (User, Product, Order) extensibles
- Intégration d'un système de reviews associé aux produits
- Réutilisation des classes de base dans différents contextes

3. Scalabilité :

- Optimisation des relations entre classes pour supporter la croissance
- Utilisation d'une relation Many-to-Many entre Product et CartItem pour une gestion flexible du panier
- Mise en place d'un système Visitor pour gérer les utilisateurs non authentifiés

2.1.2. Outils et validation

- Utilisation de PlantUML pour la modélisation UML
- Validation itérative des modèles avec les parties prenantes
- Tests unitaires pour valider les comportements des classes

2.1.3. Bénéfices de l'approche OO

- Maintenabilité : structure claire facilitant les évolutions futures
- Testabilité : modules isolés pour des tests ciblés
- Intégration Django : exploitation optimale de l'ORM Django

Cette démarche a permis de passer efficacement de la conception à la réalisation, en s'appuyant sur une base solide et documentée, comme en témoignent les diagrammes joints en annexe.

2.2. Présentation des entités principales

2.2.1. User (Utilisateur)

Attributs :

username, email, password (hérités de Django User), is_active, is_staff

Relations :

- One-to-One avec Client ou Admin
- One-to-Many avec Product (vendeur)
- One-to-Many avec Order (client)
- Responsabilités : gestion de l'authentification et point central des interactions utilisateur

2.2.2. Product (Produit)

Attributs :

name, description, price, image, brand, category, rating, numReviews, countInStock, createdAt

Relations :

- Many-to-One avec User (vendeur)
- One-to-Many avec Review
- One-to-Many avec OrderItem
- Responsabilités : gestion des informations produit, suivi des stocks et avis

2.2.3. Order (Commande)

Attributs :

paymentMethod, taxPrice, shippingPrice, totalPrice, isPaid, paidAt, isDelivered, deliveredAt, createdAt

Relations :

- Many-to-One avec User (client)
- One-to-Many avec OrderItem
- One-to-One avec ShippingAddress
- Responsabilités : gestion du processus de commande, suivi du paiement et de la livraison

2.2.4. Cart (Panier)

Attributs :

created_at, updated_at

Relations :

- Many-to-One avec User ou Visitor
 - One-to-Many avec CartItem
 - Responsabilités : gestion du panier d'achat et suivi des modifications
-

2.2.5. Client (Client)

Attributs :

phone, address, city, postal_code, country,

created_at

Relations :

- One-to-One avec User
- Responsabilités : stockage des informations client et gestion des adresses de livraison

2.2.6. Admin (Administrateur)

Attributs :

department, role, created_at

Relations :

- One-to-One avec User
- Responsabilités : gestion des droits d'administration et modération du contenu

2.2.7. Visitor (Visiteur)

Attributs :

session_id, created_at, last_visit, cart_items

Responsabilités : gestion des utilisateurs non authentifiés et suivi des paniers temporaires

2.3.3. Contraintes et Règles de Gestion

- **User :**
 - username et email doivent être uniques.
 - Le mot de passe est crypté via Django (PBKDF2).
- **Project :**
 - Limité à 5 projets par utilisateur (vérifié dans la méthode save()).
 - Les tags sont validés par l'administrateur.
- **Review :**
 - La notation est comprise entre 1 et 5 étoiles.
- **Message :**
 - Limite de 10 messages non lus par utilisateur.

2.3.4. Extrait d'Implémentation Django

```
# models.py
from django.db import models

class User(models.Model):
    username = models.CharField(max_length=100, unique=True)
    email = models.EmailField(unique=True)
    password = models.CharField(max_length=128) # Hashé par Django
    is_active = models.BooleanField(default=True)

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    bio = models.TextField()
    skills = models.JSONField(default=list)
    avatar = models.ImageField(upload_to='profiles/')

class Project(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    title = models.CharField(max_length=200)
    description = models.TextField()
    tags = models.ManyToManyField('Tag')
    created_at = models.DateTimeField(auto_now_add=True)
```

2.3. Diagramme de classes UML

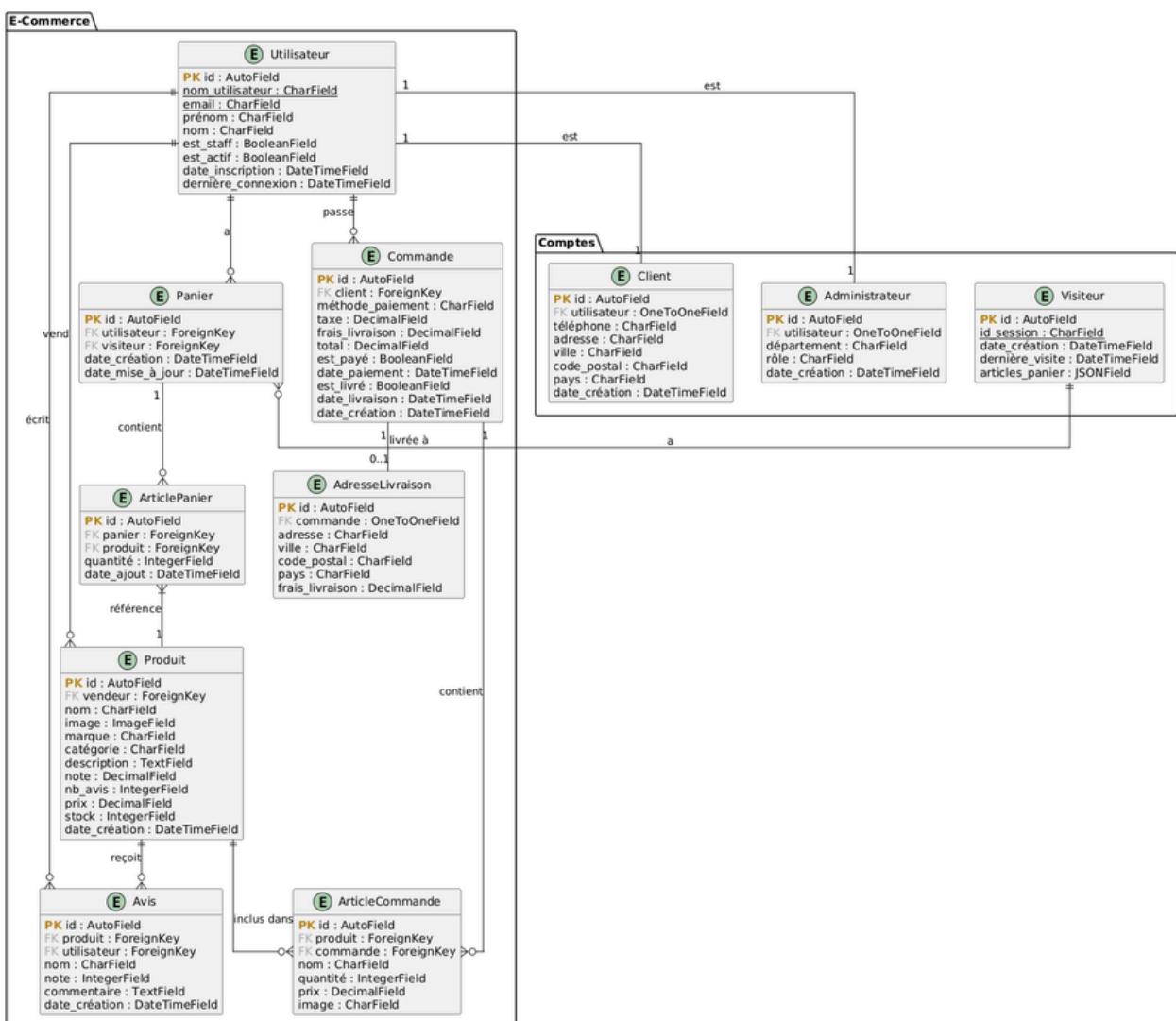
Le diagramme de classes UML de ProShop formalise la structure des données, les relations entre les entités et les responsabilités de chaque classe. Il sert de fondation à l'implémentation technique avec Django et PostgreSQL.

2.3.1. Structure du Diagramme

Le diagramme comprend :

- **Classes principales** (avec attributs et méthodes).
- **Relations** (associations, héritage, agrégation).
- **Contraintes** (cardinalités, règles de gestion).

2.3.2. Diagramme Complet



2.4. Diagrammes de cas d'utilisation

Les diagrammes de cas d'utilisation UML illustrent les interactions entre les acteurs principaux (clients, vendeurs, administrateurs) et le système ProShop, mettant en lumière les fonctionnalités clés définies dans le cahier des charges

2.4.1. Acteurs Principaux

Acteur	Rôle
Client	Utilisateur standard : consulte les produits, effectue des achats, gère son panier
Vendeur	Gère son catalogue de produits, suit ses ventes, gère ses stocks
Administrateur	Gère la plateforme, modère les contenus, supervise les transactions
Système Externe	Services tiers (ex. : Stripe pour les paiements, AWS S3 pour le stockage)

2.4.2. Cas d'Utilisation Clés

A. Pour le Développeur

1. S'authentifier

- Précondition : posséder un compte valide
- Scénario principal :
 - a. L'utilisateur saisit son email et son mot de passe
 - b. Le système vérifie les identifiants via JWT
 - c. L'accès est autorisé
- Extensions :
 - En cas d'échec, un message d'erreur est affiché

2. Publier un projet

- *L'utilisateur accède au catalogue*
- *Le système affiche les produits disponibles*
- *L'utilisateur peut filtrer les produits par catégorie, prix, etc.*

3. Gérer le panier

- L'utilisateur ajoute des produits au panier
- Le système met à jour les quantités et le total
- L'utilisateur peut modifier ou supprimer des articles

4. Passer une commande

- Précondition : panier non vide
- L'utilisateur valide son panier
- Le système calcule les frais de livraison
- L'utilisateur choisit le mode de paiement
- La commande est enregistrée

B. Pour le Vendeur

1. Gérer les produits

- Le vendeur ajoute ou modifie un produit
- Le système valide les données saisies
- Le produit devient visible dans le catalogue

2. Suivre les commandes

- Le vendeur consulte ses commandes
- Le système affiche les détails et statuts
- Le vendeur peut mettre à jour les statuts

2. Gérer les stocks

- Le vendeur met à jour les quantités disponibles
- Le système vérifie la disponibilité
- Les stocks sont actualisés en temps réel

B. Pour le Vendeur

1. Modérer les contenus

- L'administrateur signale un contenu inapproprié
- Le système notifie le vendeur concerné
- Le contenu est masqué en attendant révision

2. Gérer les utilisateurs

- L'administrateur consulte les comptes utilisateurs
- Le système affiche les informations associées
- L'administrateur peut suspendre ou supprimer des comptes

2. Superviser les transactions

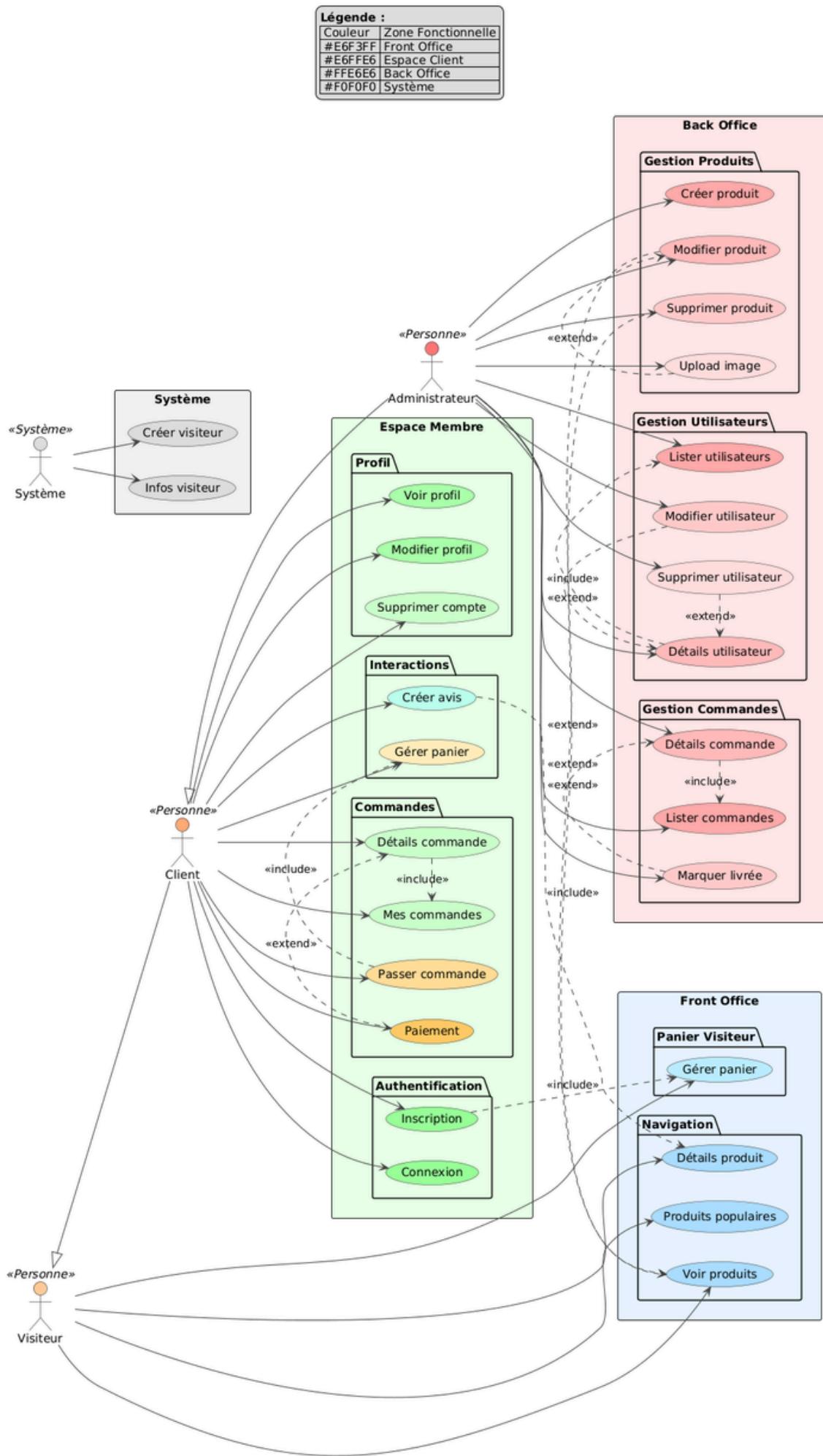
- L'administrateur accède au tableau de bord
- Le système affiche les statistiques de vente et paiements
- L'administrateur peut intervenir sur les litiges

2.4.3. Relations Entre Cas d'Utilisation

- **Inclusion (include) :**
 - S'authentifier est inclus dans Passer une commande (connexion obligatoire pour le client)
- **Extension (extend) :**
 - Modérer les contenus étend Gérer les produits lorsqu'un signalement est détecté
 - Gérer les stocks étend Passer une commande en cas de stock insuffisant

2.4.4. Diagramme Complet

Système E-Commerce - Diagramme de Cas d'Utilisation



2.5. Diagrammes de séquence

Les diagrammes de séquence UML permettent de représenter les interactions dynamiques entre les différents composants du système ProShop ainsi que les acteurs impliqués. Ils sont essentiels pour modéliser le déroulement temporel des messages échangés lors de scénarios fonctionnels clés, validant ainsi la logique d'implémentation du système.

2.5.1. Authentification de l'Utilisateur (Client ou Vendeur):

Ce diagramme illustre le processus d'authentification sécurisé d'un utilisateur à l'aide de JWT (JSON Web Token), qu'il soit Client ou Vendeur.

- Acteurs : Client/Vendeur, Interface Utilisateur, Serveur Applicatif, Base de Données.
- Objectif : Authentifier l'utilisateur et générer un token d'accès sécurisé.

Scénario détaillé :

1. L'utilisateur saisit son email et mot de passe dans l'interface.
2. Une requête d'authentification est envoyée au serveur.
3. Le serveur interroge la base de données pour valider les identifiants.
4. Si l'authentification est réussie, un token JWT est généré.
5. Ce token est retourné à l'interface utilisateur.
6. L'interface stocke le token (dans le stockage local ou les cookies).
7. L'utilisateur est redirigé vers son espace personnel.
8. Pour les requêtes futures, le token est inclus dans les en-têtes HTTP.
9. Le serveur valide le token avant de permettre l'accès aux ressources protégées.

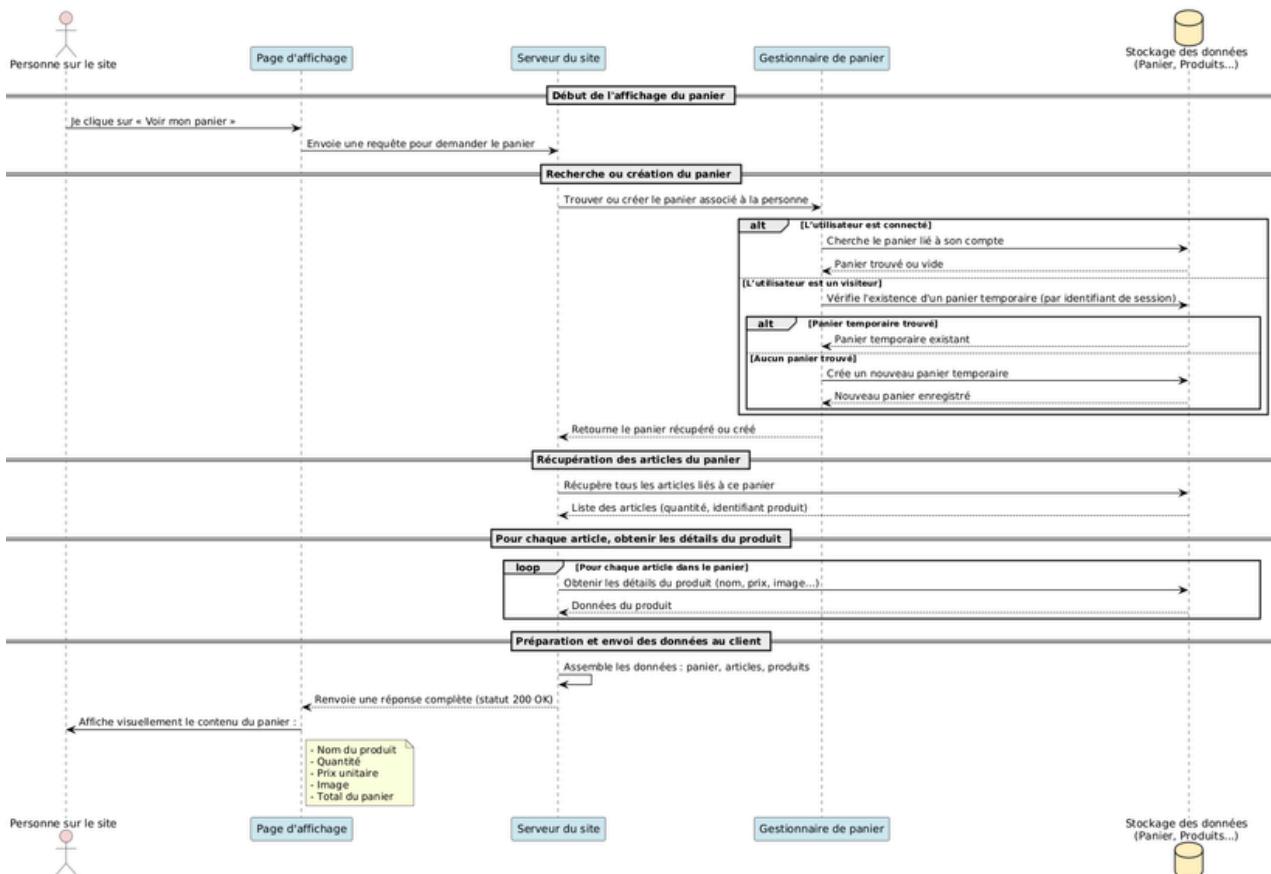
2.5.2. Gestion du Panier:

Ce diagramme décrit les interactions entre les composants lors de la consultation ou de la création d'un panier, que l'utilisateur soit connecté ou simple visiteur.

- Acteurs : Visiteur/Client, Interface Utilisateur, Serveur, Gestionnaire de Panier, Stockage des Données.
- Objectif : Afficher ou créer dynamiquement un panier.

Scénario détaillé :

1. L'utilisateur clique sur « Voir mon panier ».
2. Une requête est envoyée au serveur.
3. Le gestionnaire de panier tente de récupérer un panier existant (lié au compte ou à la session).
4. Si aucun panier n'est trouvé, un nouveau panier est créé.
5. Les articles du panier sont récupérés depuis la base de données.
6. Pour chaque article, les détails du produit sont demandés.
7. Le serveur assemble toutes les informations (quantité, prix, image, etc.).
8. L'interface reçoit les données complètes et affiche le contenu du panier.



2.5.3. Début de la Commande:

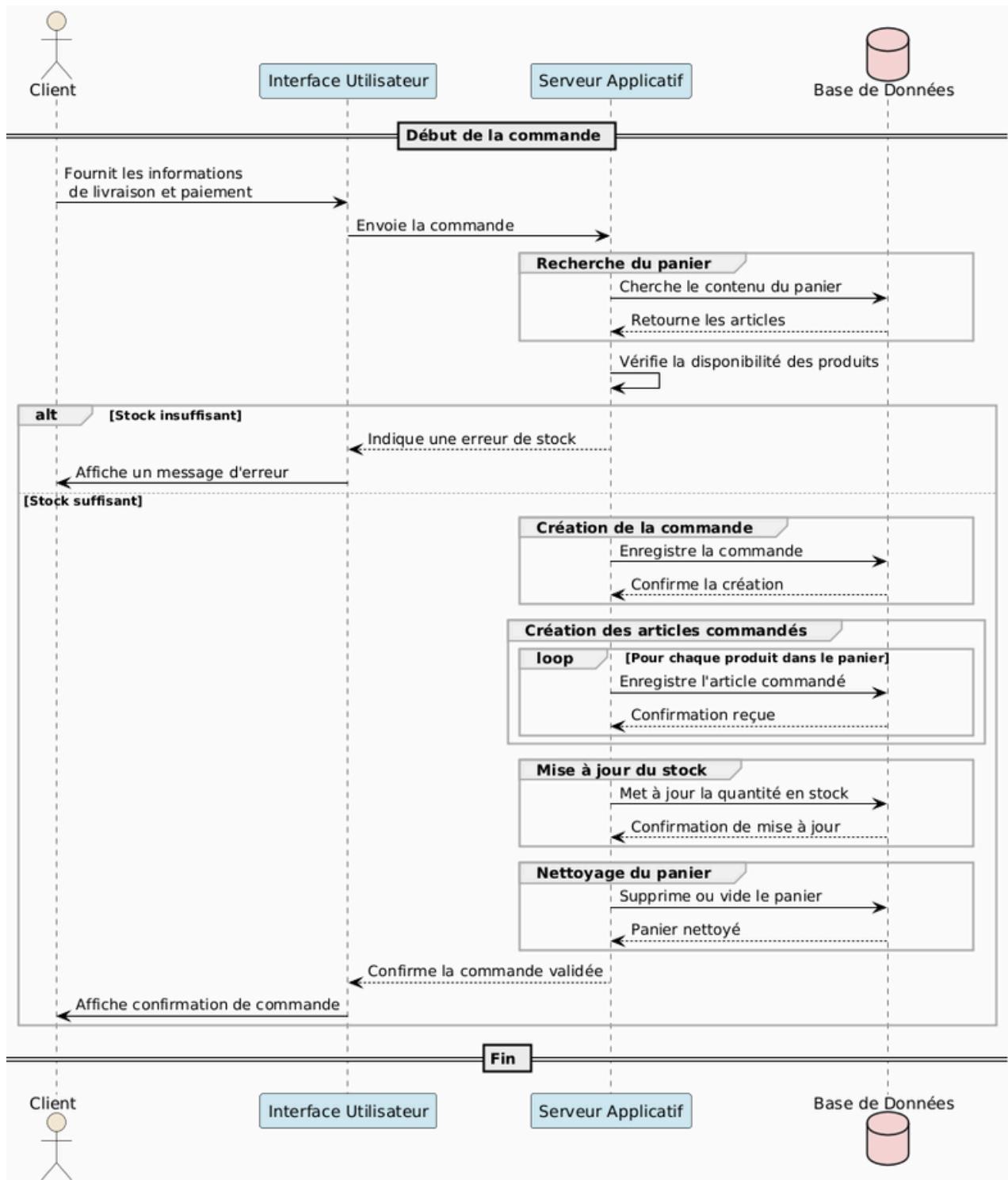
Ce diagramme modélise la création d'une commande à partir du panier, en validant les disponibilités en stock.

- **Acteurs** : Client, Interface Utilisateur, Serveur Applicatif, Base de Données.
- **Objectif** : Finaliser une commande et réserver les articles.

Scénario détaillé :

1. Le client saisit les informations de livraison et de paiement.
2. L'interface transmet la commande au serveur.
3. Le serveur récupère le panier de l'utilisateur.
4. Il vérifie la disponibilité des articles.
5. En cas de stock insuffisant, une erreur est retournée.
6. Sinon, la commande est enregistrée dans la base de données.
7. Chaque article est ajouté à la commande.

8. Le stock est mis à jour en conséquence.
9. Le panier est vidé.
10. Une confirmation est renvoyée à l'utilisateur.



2.5. Diagrammes d'activités

1. Processus de Crédit/Modification d'Avis

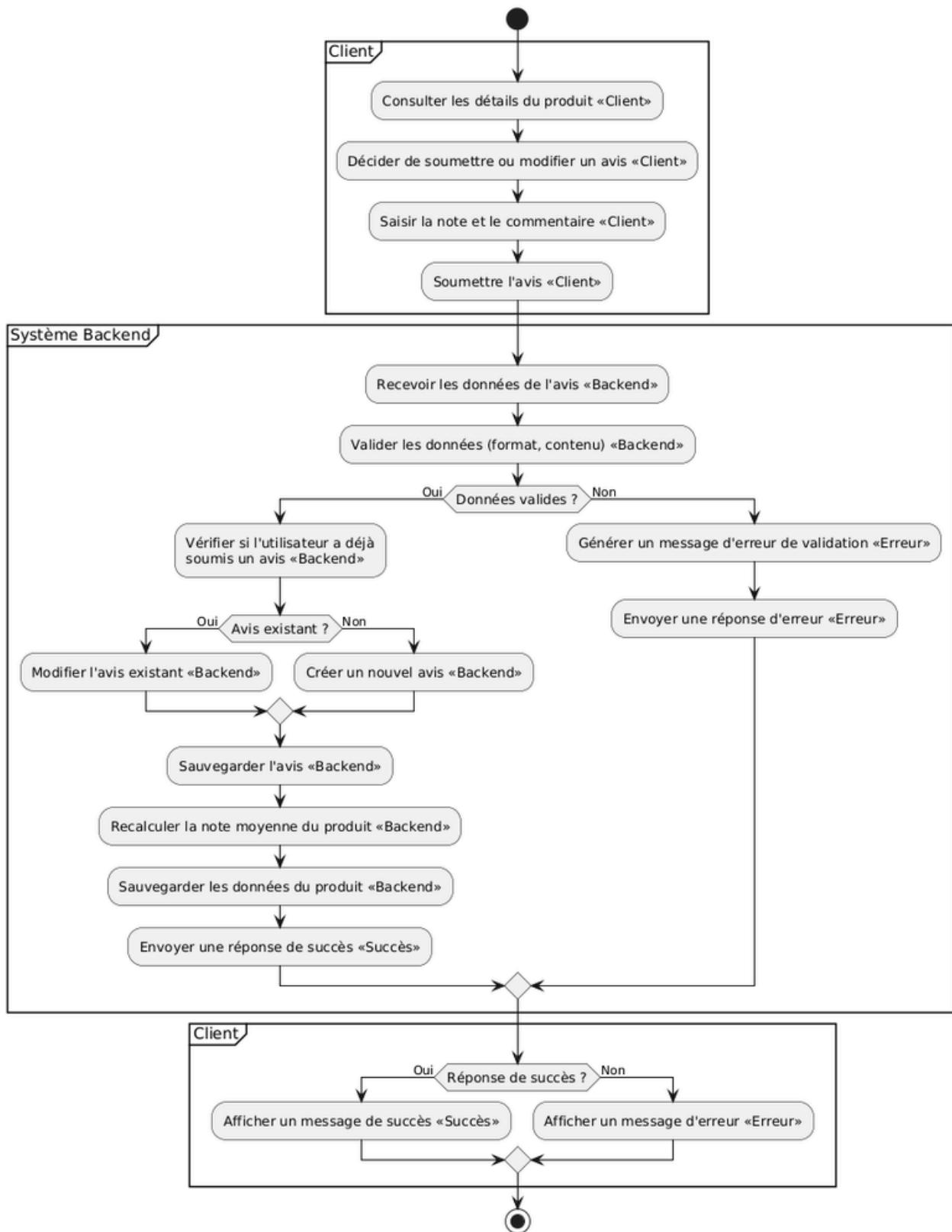
Acteurs : Client, Système Backend

Description : Ce processus décrit comment un client ajoute ou modifie un avis sur un produit.

Étapes clés :

- Le client accède aux détails d'un produit et saisit une note et un commentaire.
- Le système vérifie la validité des données et si un avis préexistant existe.
- Si l'avis existe, il est mis à jour. Sinon, un nouvel avis est enregistré.
- Le système recalcule la note moyenne du produit et renvoie une réponse de succès ou d'erreur.

Processus de Création/Modification d'Avis



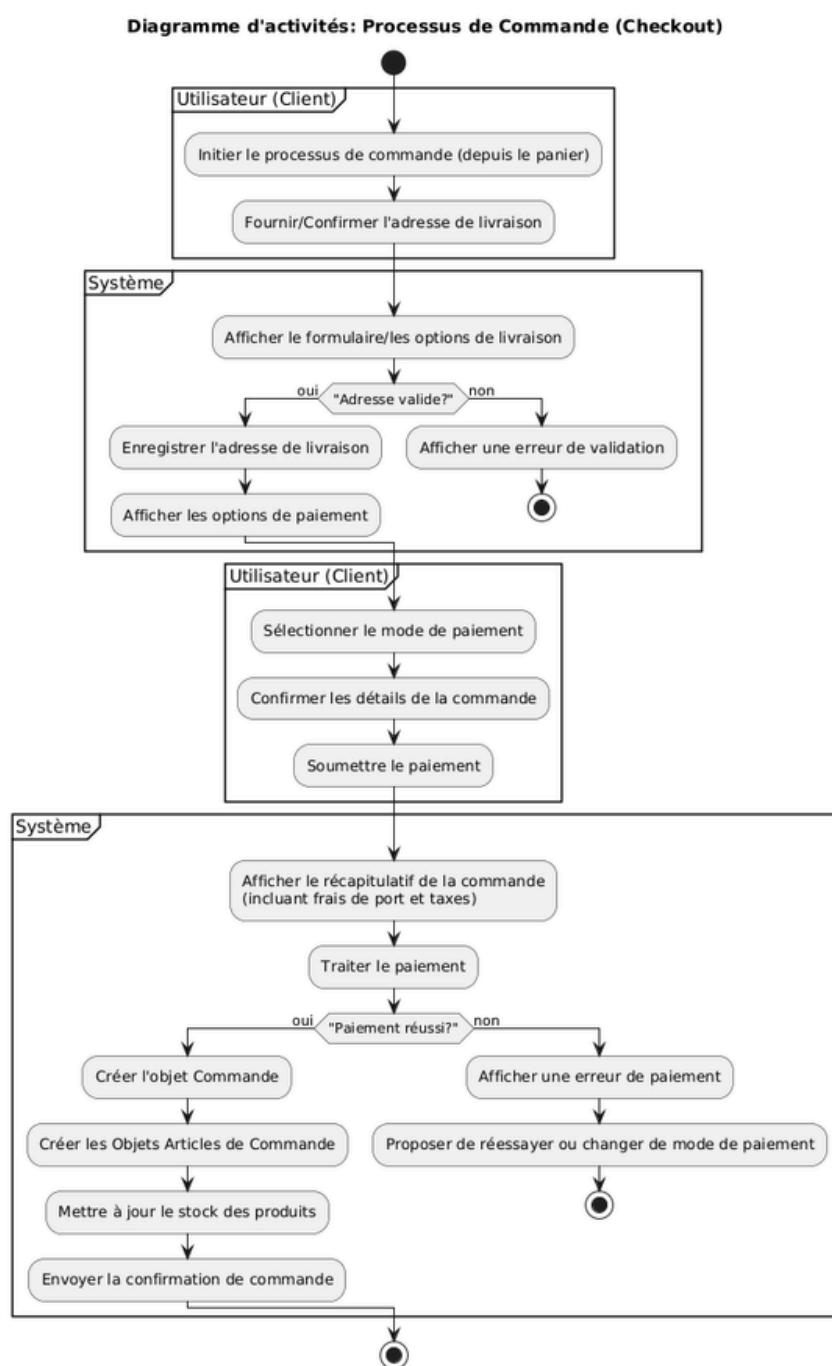
2. Diagramme d'Activité – Processus de Commande (Checkout)

Acteurs : Utilisateur (Client), Système

Description : Ce diagramme illustre le parcours d'un utilisateur lors de la validation de son panier.

Étapes clés :

- L'utilisateur fournit une adresse de livraison.
- Le système valide les informations, propose les options de paiement.
- L'utilisateur choisit un mode de paiement et valide la commande.
- Le paiement est traité. En cas de succès, la commande est créée, le stock est mis à jour, et une confirmation est envoyée.



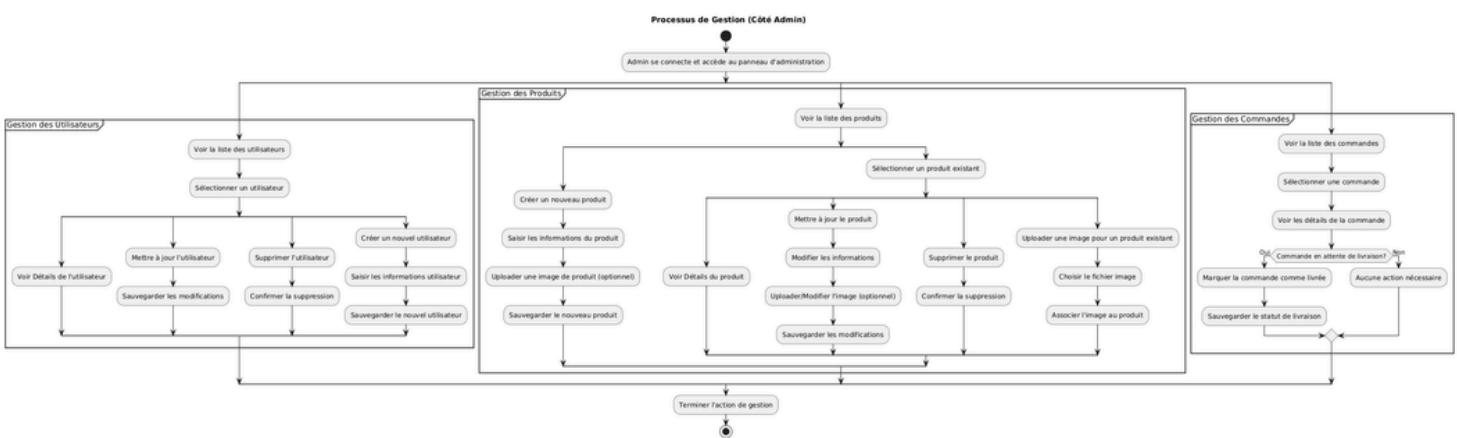
3. Processus de Gestion – Côté Administrateur

Acteur : Administrateur

Description : Ce processus regroupe les différentes opérations d'administration sur la plateforme.

Catégories :

- Gestion des utilisateurs : Consultation, modification, suppression ou création de comptes.
- Gestion des produits : Mise à jour, suppression ou ajout de nouveaux produits, avec gestion des images.
- Gestion des commandes : Visualisation des commandes. Si la commande est en attente, l'admin peut marquer comme expédiée.



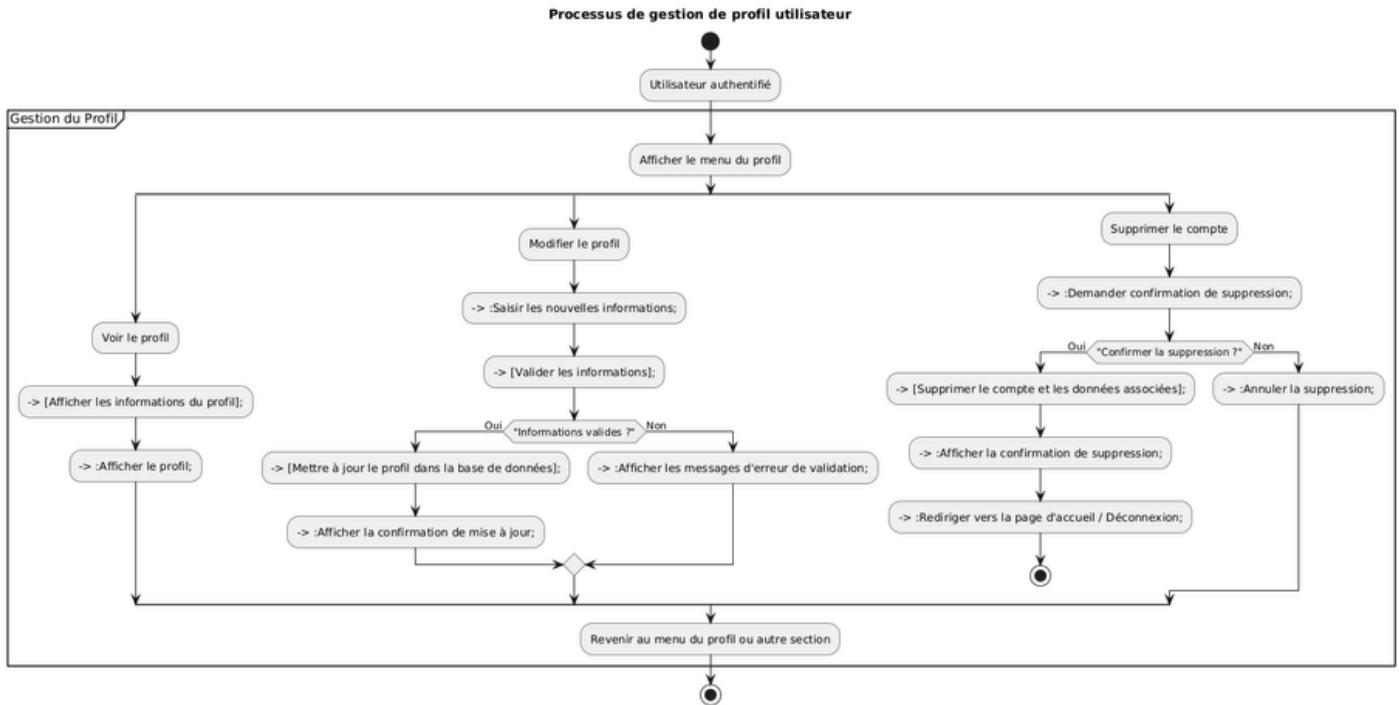
4. Processus de Gestion de Profil Utilisateur

Acteur : Utilisateur Authentifié

Description : Ce diagramme illustre comment un utilisateur gère les informations de son profil.

Options disponibles :

- Consulter son profil.
- Modifier ses informations personnelles. (avec validation)
- Supprimer son compte. (demande de confirmation avant suppression définitive et déconnexion)



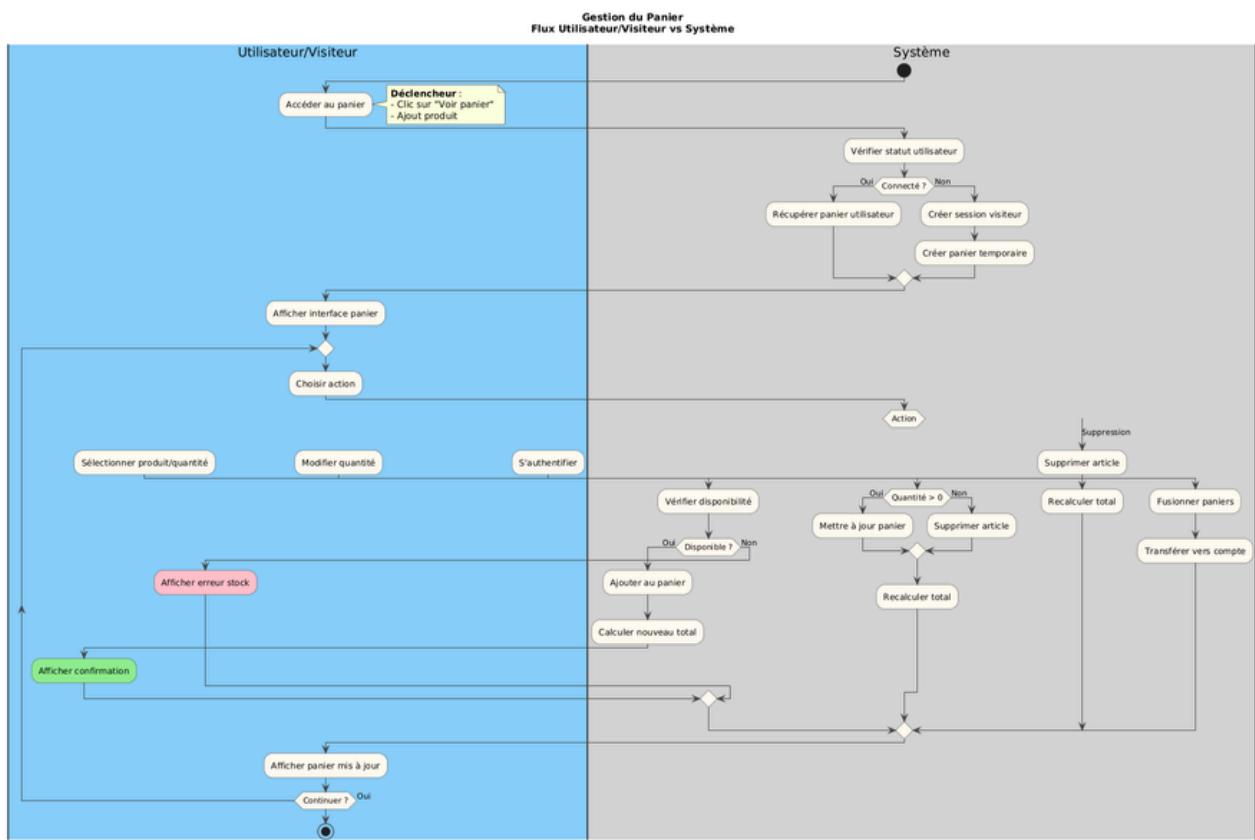
5. Processus de Navigation et Consultation des Produits

Acteurs : Utilisateur, Backend, Frontend

Description : Ce processus décrit l'interaction d'un utilisateur avec le catalogue de produits.

Étapes clés :

- Chargement initial des produits avec authentification si nécessaire.
- L'utilisateur peut consulter, filtrer, trier ou naviguer dans les produits.
- Chaque action déclenche un traitement côté backend et une mise à jour de l'interface frontend.
- L'utilisateur peut poursuivre l'exploration ou quitter le processus.



Chapitre 3 :Réalisation

3.1. Architecture technique

Ce chapitre présente l'architecture technique globale de la plateforme ProShop ainsi que la structure de sa base de données relationnelle. Il détaille les choix technologiques adoptés et leur justification dans le cadre d'une application e-commerce moderne, sécurisée et scalable.

3.1.1. Stack Technique Principale

L'architecture repose sur une stack technologique moderne, centrée sur Django pour le backend et React.js pour le frontend. Cette combinaison permet de concevoir une application modulaire, performante et maintenable. L'API est sécurisée à l'aide de JSON Web Tokens (JWT) via le package SimpleJWT. L'ensemble est déployé sur une infrastructure cloud AWS.

Composant	Technologie	Rôle Clé
Backend	Django 5.0.6	Logique métier, ORM, sécurité
API REST	DRF + SimpleJWT	Endpoints sécurisés, gestion JWT
Frontend	React.js + Bootstrap	Interface interactive, responsive
Base de données	PostgreSQL 15	Stockage relationnel performant
Serveur App	Gunicorn	Exécution backend en production
Serveur Web	Nginx	Reverse proxy, HTTPS, gestion statique
Fichiers	Whitenoise + AWS S3	Stockage statique et médias
Paiement	Stripe / PayPal	Traitements sécurisés des transactions

3.1.2. Schéma d'Architecture

- Frontend (React.js) : Fournit l'interface graphique. Interagit uniquement avec l'API REST du backend.
- Backend (Django + DRF + SimpleJWT) : Assure la gestion de la logique métier, l'accès sécurisé via JWT, les interactions avec la base de données, et expose des endpoints API.
- Base de Données (PostgreSQL) : Stocke toutes les données critiques : utilisateurs, produits, commandes, etc.
- Stockage Média (AWS S3) : Héberge les fichiers volumineux, comme les images produits, pour une meilleure performance.
- Déploiement :
 - Gunicorn : Serveur WSGI pour exécuter Django.
 - Nginx : Reverse proxy qui assure le SSL, le routage, la mise en cache, et la gestion des fichiers statiques.
 - AWS Cloud : Infrastructure cloud complète avec EC2 (serveur), RDS (base de données), S3 (stockage) et CloudFront (CDN).
- Services externes (Stripe / PayPal) : Intégrés via API pour le paiement sécurisé.

3.1.2. Schéma d'Architecture

La base de données est conçue selon un modèle relationnel avec PostgreSQL. L'ORM de Django permet une définition claire des modèles, tout en assurant l'intégrité et la cohérence des données.

3.2.1. Schéma Relationnel Simplifie

Table	Rôle Principal	Liens Clés
auth_user	Utilisateur de base	1:1 → Client, Admin
store_client	Détails client	1:1 → auth_user
store_admin	Détails admin	1:1 → auth_user
store_product	Produit en vente	N:1 → auth_user
store_review	Avis sur produits	N:1 → product, auth_user
store_order	Commande passée	N:1 → auth_user, 1:1 → Adresse
store_orderitem	Article d'une commande	N:1 → product, order
store_shippingaddress	Adresse de livraison	1:1 → order
store_visitor	Session visiteur non connecté	1:N → Cart
store_cart	Panier d'un utilisateur ou visiteur	N:1 → user, visitor
store_cartitem	Article dans un panier	N:1 → cart, product

3.3. Fonctionnalités principales

L'application développée est une plateforme e-commerce construite autour d'une architecture RESTful avec Django REST Framework. Elle permet une gestion complète des produits, des utilisateurs, des commandes, et des paiements. Voici un aperçu des principales fonctionnalités :

1. Gestion des Produits :

- Affichage de la liste des produits
- Affichage des détails d'un produit
- Gestion du stock

```
@api_view(['GET'])
def getTopProducts(request):
    products = Product.objects.filter(rating__gte=4).order_by('-rating')[0:5]
    serializer = ProductSerializer(products, many=True)
    return Response(serializer.data)

@api_view(['GET'])
def getProduct(request, pk):
    product = Product.objects.get(_id=pk)
    serializer = ProductSerializer(product, many=False)
    return Response(serializer.data)
```

2. Gestion des Utilisateurs :

- Inscription des utilisateurs
- Connexion/Déconnexion
- Profil utilisateur
- Mise à jour des informations utilisateur

```

@api_view(['POST'])
def registerUser(request):
    data = request.data
    try:
        user = User.objects.create(
            first_name=data['name'],
            username=data['email'],
            email=data['email'],
            password=make_password(data['password']))
    )

    serializer = UserSerializerWithToken(user, many=False)
    return Response(serializer.data)
    except:
        message = {'detail': 'User with this email already exists'}
        return Response(message, status=status.HTTP_400_BAD_REQUEST)

```

3.Gestion des Commandes :

- Création de commandes
- Suivi des commandes
- Gestion des adresses de livraison
- Historique des commandes

```

@api_view(['POST'])
@permission_classes([IsAuthenticated])
def addOrderItems(request):
    user = request.user
    data = request.data

    orderItems = data['orderItems']

    if orderItems and len(orderItems) == 0:
        else:

            order = Order.objects.create(
                ...
            )

            shipping = ShippingAddress.objects.create(
                ...
            )

            for i in orderItems:
                ...

                serializer = OrderSerializer(order, many=False)
                return Response(serializer.data)

```

4. Authentification et Sécurité :

- Système d'authentification JWT
- Gestion des permissions
- Protection des routes

```
@api_view(['GET'])
@permission_classes([IsAuthenticated])
def getUserProfile(request):
    user = request.user
    serializer = UserSerializer(user, many=False)
    return Response(serializer.data)

@api_view(['GET'])
@permission_classes([IsAuthenticated])
def getUserProfile(request):
    user = request.user
    serializer = UserSerializer(user, many=False)
    return Response(serializer.data)
```

5. Gestion des Paiements :

- Traitement des paiements
- Confirmation des paiements
- Mise à jour du statut des commandes

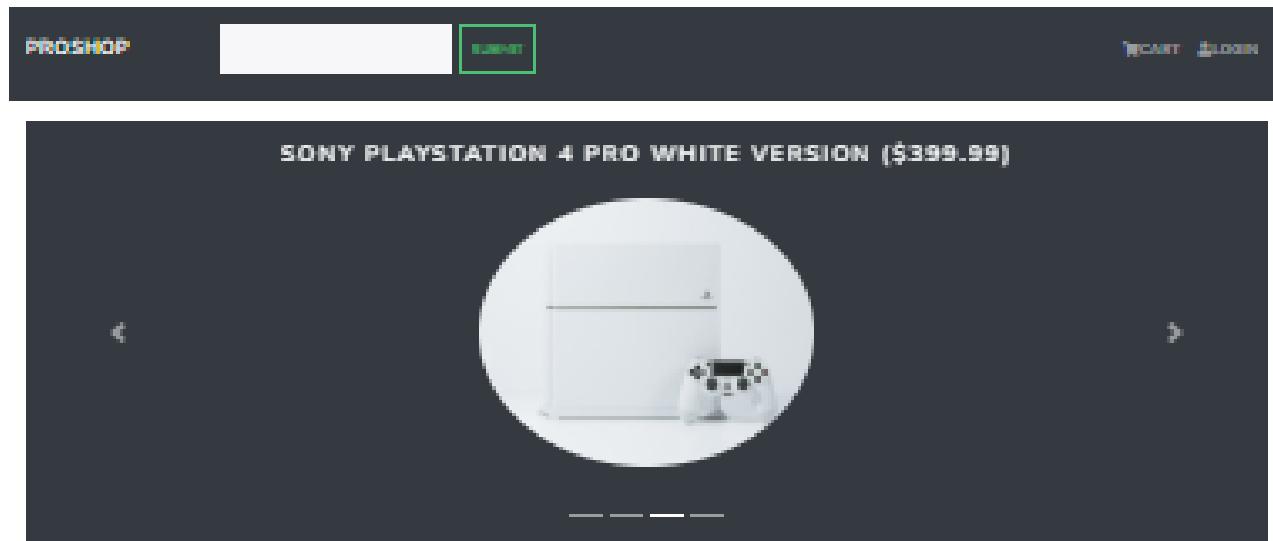
```
@api_view(['PUT'])
@permission_classes([IsAuthenticated])
def updateOrderToPaid(request, pk):
    order = Order.objects.get(_id=pk)

    order.isPaid = True
    order.paidAt = datetime.now()
    order.save()

    return Response('Order was paid')
```

3.5.2. Structure de l'Interface

Pages d'Accueil et Produits



LATEST PRODUCTS

<p>Amazon Echo Dot 2nd Generation ★ ★ ★ ★ ★ 5 reviews \$29.99</p>	<p>Logitech G-Series Gaming Mouse ★ ★ ★ ★ ★ 5 reviews \$49.99</p>	<p>Sony PlayStation 4 Pro White Version ★ ★ ★ ★ ★ 5 reviews \$399.99</p>	<p>Canon EOS R50 DSLR Camera ★ ★ ★ ★ ★ 5 reviews \$929.99</p>
---	---	--	---

<p>iPhone 11 Pro Max 64GB Memory ★ ★ ★ ★ ★ 5 reviews \$599.99</p>

[HomeScreen.js](#) : Page d'accueil principale

- Affiche les produits en vedette
- Présente les promotions et nouveautés
- Interface principale pour la navigation

[ProductListScreen.js](#) : Liste des produits

- Affichage de tous les produits disponibles
- Filtres et recherche de produits
- Pagination des résultats

[ProductScreen.js](#) : Détails du produit

- Informations détaillées sur un produit
- Galerie d'images
- Système d'avis et notes
- Options d'ajout au panier

Pages d'Authentification

The screenshot shows a mobile application's sign-in screen. At the top, there is a dark header bar with the text "PROSHOP" on the left, a search bar in the center, and a green "SUBMIT" button on the right. On the far right of the header, there are icons for "CART" and "LOGIN". Below the header, the main content area has a light gray background. The title "SIGN IN" is centered at the top of this area. Below the title are two input fields: one for "Email Address" and one for "Password", both with placeholder text ("Enter Email" and "Enter Password" respectively). At the bottom of the input fields is a black "SIGN IN" button. At the very bottom of the screen, there is a small link that says "New Customer? Register".

LoginScreen.js : Page de connexion

- Formulaire de connexion
- Gestion des erreurs d'authentification
- Lien vers l'inscription

PROSHOP

SUBMIT

CART LOGIN

SIGN IN

Name

Enter name

Email Address

Enter Email

Password

Enter Password

Confirm Password

Confirm Password

REGISTER

Have an Account? Sign In

RegisterScreen.js : Page d'inscription

- Formulaire d'inscription
- Validation des données
- Création de compte utilisateur

Pages de Gestion du Panier et Commandes

[GO BACK](#)

REVIEWS

No Reviews

WRITE A REVIEW

Rating

Review

SONY PLAYSTATION 4 PRO WHITE VERSION

1 reviews

Price: \$399.99

Price: \$399.99

Status: In Stock

Qty: 1

Description: The ultimate home entertainment center starts with PlayStation. Whether you are into gaming, HD movies, television, music

CartScreen.js : Panier d'achat

- Liste des produits dans le panier
- Modification des quantités
- Calcul du total
- Passage à la commande



Login Shipping Payment Place Order

SHIPPING

Address

Enter address

City

Enter city

Postal Code

Enter postal code

Country

Enter country

CONTINUE

ShippingScreen.js : Adresse de livraison

- Formulaire d'adresse de livraison
- Validation des informations
- Sélection du mode de livraison



Login Shipping Payment Place Order

SHIPPING

Shipping: Morocco, Rabat 10000, Maroc

PAYMENT METHOD

Method: PayPal

ORDER ITEMS



Sony Playstation 4 Pro White Version

1 X \$399.99 = \$399.99

ORDER SUMMARY

Items:	\$399.99
Shipping:	\$0.00
Tax:	\$32.80
Total:	\$432.79

PLACE ORDER



Login Shipping Payment Place Order

Select Method

PayPal or Credit Card

CONTINUE

PaymentScreen.js : Paiement

- Sélection du mode de paiement
- Intégration des moyens de paiement
- Confirmation du paiement

A screenshot of the PaymentScreen.js interface. The top navigation bar is identical to the previous screenshot. The main content area is divided into several sections:

- SHIPPING:** Shipping: Morocco, Rabat 10000, Maroc
- PAYMENT METHOD:** Method: PayPal
- ORDER ITEMS:** A table showing one item: Sony Playstation 4 Pro White Version at \$399.99.
- ORDER SUMMARY:** A table detailing the order costs:

Items:	\$399.99
Shipping:	\$0.00
Tax:	\$32.80
Total:	\$432.79

A large black "PLACE ORDER" button is located at the bottom right of the summary section.



ORDER:

SHIPPING

Name: Zineb
Email: zineb1@gmail.com
Shipping: Morocco, Rabat 10000, Maroc

Not Delivered

PAYMENT METHOD

Method: PayPal

Not Paid

ORDER ITEMS

Sony Playstation 4 Pro White Version 1 X \$399.99 = \$399.99

ORDER SUMMARY

Items:	\$399.99
Shipping:	\$0.00
Tax:	\$32.80
Total:	\$432.79

PayPal

Carte bancaire

Optimisé par PayPal

PlaceOrderScreen.js : Confirmation de commande

- Récapitulatif de la commande
- Détails de livraison et paiement
- Confirmation finale

Pages de Gestion de Profil



USER PROFILE

Name

Zineb

Email Address

zineb1@gmail.com

Password

...

Confirm Password

Confirm Password

MY ORDERS

ID	DATE	TOTAL	PAID	DELIVERED
33	2025-06-02	\$432.79	X	DETAILS

UPDATE

[ProfileScreen.js](#) : Profil utilisateur

- Informations personnelles
- Historique des commandes
- Modification des données

USER PROFILE

Name

Zineb

Email Address

zineb1@gmail.com

Password

...

Confirm Password

...

UPDATE

[UserEditScreen.js](#) : Modification du profil

- Édition des informations utilisateur
- Changement de mot de passe
- Gestion des préférences

Pages d'Administration

USERS

ID	NAME	EMAIL	ADMIN	
1	Dennis Ivanov	dennisivy11@gmail.com	✓	<input checked="" type="checkbox"/> <input type="checkbox"/>
8	Jack	jack@email.com	✓	<input checked="" type="checkbox"/> <input type="checkbox"/>
9	admin@email.com	admin@email.com	✓	<input checked="" type="checkbox"/> <input type="checkbox"/>
10	Zineb Boulahbach	zineb@gmail.com	✓	<input checked="" type="checkbox"/> <input type="checkbox"/>
11	Zineb	zineb2@gmail.com	✓	<input checked="" type="checkbox"/> <input type="checkbox"/>
12	Zineb@gmail.com	Zineb@gmail.com	✓	<input checked="" type="checkbox"/> <input type="checkbox"/>

Gestion des utilisateurs (/admin/userlist)

- Voir tous les utilisateurs
- Modifier les informations utilisateur
- Changer les droits d'accès
- Supprimer des comptes

PRODUCTS

+ CREATE PRODUCT

ID	NAME	PRICE	CATEGORY	BRAND	
6	Amazon Echo Dot 3rd Generation	\$29.99	Electronics	Amazon	<input checked="" type="checkbox"/> <input type="checkbox"/>
5	Logitech G-Series Gaming Mouse	\$49.99	Electronics	Logitech	<input checked="" type="checkbox"/> <input type="checkbox"/>
4	Sony Playstation 4 Pro White Version	\$399.99	Electronics	Sony	<input checked="" type="checkbox"/> <input type="checkbox"/>
3	Cannon EOS 80D DSLR Camera	\$929.99	Electronics	Cannon	<input checked="" type="checkbox"/> <input type="checkbox"/>
2	iPhone 11 Pro 256GB Memory	\$599.99	Electronics	Apple	<input checked="" type="checkbox"/> <input type="checkbox"/>

1 2

Gestion des produits (/admin/productlist)

- Ajouter de nouveaux produits
- Modifier les produits existants
- Gérer les stocks
- Uploader des images



Gestion des commandes (/admin/orderlist)

- Voir toutes les commandes
- Suivre les statuts
- Gérer les livraisons
- Voir les détails des commandes

Technologies utilisées

Frontend

Élément	Technologie	Justification principale
Framework	React	Création d'une interface SPA avec composants modulaires (components, screens).
Style / UI	React-Bootstrap	Intégration de composants Bootstrap pour un design responsive (bootstrap.min.css).
Gestion d'état	Redux	Centralisation de l'état global via store.js, reducers, et actions.
Routage	React Router	Navigation entre vues sans rechargement (react-router-dom, react-router-bootstrap).
Requêtes API	Axios (probable)	Bibliothèque fréquemment utilisée pour les appels HTTP entre React et l'API.

Backend

Élément	Technologie	Justification principale
Framework Web	Django	Architecture MVC avec manage.py, settings.py, et structure Django classique.
API REST	Django REST Framework	Création d'API RESTful (rest_framework, vues APIView et décorateurs).
Base de données	SQLite	Base légère par défaut pour le développement (db.sqlite3).
Authentification	Simple JWT	Authentification sécurisée par tokens (rest_framework_simplejwt).
Filtrage	Django-filter (présumé)	Présence de filters.py, utilisation de DjangoFilterBackend.
Langage	Python	Langage principal utilisé côté serveur.

Chapitre 5 :

Évaluation et Perspectives

5.1. Points forts et limites du projet

5.1.1 Architecture Technique Robuste et Moderne

Le socle technique du projet repose sur une combinaison éprouvée :

- Django comme framework backend, connu pour sa sécurité, sa rapidité de développement et sa scalabilité.
- Django REST Framework (DRF) pour la création d'une API RESTful bien structurée et maintenable, facilitant l'interopérabilité avec le frontend.
- PostgreSQL, une base de données relationnelle performante et fiable, adaptée aux charges e-commerce.
- L'utilisation de JWT (JSON Web Tokens) a permis une gestion sécurisée de l'authentification et de la session utilisateur, essentielle dans un contexte de transactions sensibles.
- Cette architecture facilite l'évolutivité du projet, que ce soit en termes de nouvelles fonctionnalités ou de montée en charge.

5.1.2 Fonctionnalités E-commerce Complètes et Fonctionnelles

La plateforme couvre avec succès les besoins de base d'un site de commerce en ligne :

- Gestion complète du catalogue produits, avec possibilité d'ajout, modification et suppression.
- Mise en place d'un panier dynamique permettant l'ajout, la mise à jour et la suppression de produits.
- Processus de commande fluide, avec passage de commande, confirmation et génération automatique des détails associés.
- Intégration sécurisée des paiements via Stripe ou PayPal, avec une gestion des erreurs et des confirmations bien pensée.
- Suivi du stock en temps réel, évitant les erreurs de commande sur des produits épuisés.

5.1. Points forts et limites du projet

5.1.1 Architecture Technique Robuste et Moderne

Le socle technique du projet repose sur une combinaison éprouvée :

- Django comme framework backend, connu pour sa sécurité, sa rapidité de développement et sa scalabilité.
- Django REST Framework (DRF) pour la création d'une API RESTful bien structurée et maintenable, facilitant l'interopérabilité avec le frontend.
- PostgreSQL, une base de données relationnelle performante et fiable, adaptée aux charges e-commerce.
- L'utilisation de JWT (JSON Web Tokens) a permis une gestion sécurisée de l'authentification et de la session utilisateur, essentielle dans un contexte de transactions sensibles.
- Cette architecture facilite l'évolutivité du projet, que ce soit en termes de nouvelles fonctionnalités ou de montée en charge.

5.1.2 Fonctionnalités E-commerce Complètes et Fonctionnelles

La plateforme couvre avec succès les besoins de base d'un site de commerce en ligne :

- Gestion complète du catalogue produits, avec possibilité d'ajout, modification et suppression.
- Mise en place d'un panier dynamique permettant l'ajout, la mise à jour et la suppression de produits.
- Processus de commande fluide, avec passage de commande, confirmation et génération automatique des détails associés.
- Intégration sécurisée des paiements via Stripe ou PayPal, avec une gestion des erreurs et des confirmations bien pensée.
- Suivi du stock en temps réel, évitant les erreurs de commande sur des produits épuisés.

5.1.3 Interface Utilisateur Responsive et Intuitive

Le frontend développé avec React.js offre une navigation fluide et réactive.

L'utilisation de Bootstrap assure une compatibilité multiplateforme (ordinateurs, tablettes, smartphones).

Des composants clairs, des messages d'erreur utiles, et un design moderne renforcent la satisfaction utilisateur.

Le parcours d'achat (sélection du produit > panier > paiement) a été optimisé pour maximiser la conversion.

5.1.4 Déploiement et Infrastructure Préparés pour la Scalabilité

Le déploiement sur **AWS (Amazon Web Services)** garantit robustesse et montée en charge.

Utilisation de **Gunicorn** comme serveur WSGI, **Nginx** comme reverse proxy, et **Whitenoise** pour servir les fichiers statiques.

Intégration avec AWS S3 pour le stockage des images et fichiers volumineux, ce qui soulage le serveur principal.

L'architecture permet un passage en production rapide, tout en assurant performance et sécurité.

5.2. Difficultés Rencontrées

Le développement de la plateforme **ProShop** a soulevé un certain nombre de défis, tant sur le plan technique qu'organisationnel. Leur résolution a été essentielle à la réussite du projet et a permis un approfondissement significatif des compétences en développement web fullstack.

5.2.1. Intégration de l'authentification via JWT

- **Problématique :** La mise en œuvre d'une authentification stateless reposant sur les JSON Web Tokens (JWT) a nécessité une configuration rigoureuse. Il a notamment fallu gérer la génération, la validation et le rafraîchissement sécurisé des tokens, tout en protégeant les endpoints de l'API et en contrôlant les accès selon les rôles (client, vendeur, administrateur).
- **Solution apportée :** Intégration de la bibliothèque `djangorestframework-simplejwt`, qui fournit les outils nécessaires à la gestion des tokens. Des classes de permissions spécifiques ont été définies dans Django REST Framework afin de restreindre l'accès aux ressources en fonction des rôles associés aux utilisateurs authentifiés.
- **Extrait de code :**

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    ),
    'DEFAULT_FILTER_BACKENDS': (
        'django_filters.rest_framework.DjangoFilterBackend',
        'rest_framework.filters.SearchFilter',
        'rest_framework.filters.OrderingFilter',
    ),
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 5
}
```

5.2.2. Modélisation des relations Many-to-Many

- **Problématique :** La gestion de relations complexes, telles que celles entre Cart et Product via CartItem, ou Order et Product via OrderItem, a exigé une bonne maîtrise de l'ORM Django. Il a fallu éviter les requêtes N+1 et garantir la cohérence des données (quantité, prix au moment de la commande).
- **Solution apportée :** Création explicite de modèles intermédiaires (CartItem, OrderItem) pour inclure des champs supplémentaires comme la quantité et le prix au moment de l'ajout. Les requêtes ont été optimisées via l'utilisation de select_related et prefetch_related afin de limiter les accès redondants à la base de données.
- **Extrait de code :**

```
class OrderItem(models.Model):  
    product = models.ForeignKey(Product, on_delete=models.SET_NULL, null=True)  
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)  
    name = models.CharField(max_length=200, null=True, blank=True)  
    qty = models.IntegerField(null=True, blank=True, default=0)  
    price = models.DecimalField(  
        max_digits=7, decimal_places=2, null=True, blank=True)  
    image = models.CharField(max_length=200, null=True, blank=True)  
    _id = models.AutoField(primary_key=True, editable=False)  
  
    def __str__(self):  
        return str(self.name)
```

5.2.3. Déploiement des fichiers statiques et médias

- **Problématique :** En production, Django ne sert pas automatiquement les fichiers statiques (JS, CSS) ni les fichiers médias (images produits). Leur collecte, stockage et diffusion performante ont représenté un défi particulier.
- **Solution apportée :** Utilisation de la commande collectstatic pour centraliser les fichiers statiques. Mise en place de WhiteNoise pour les servir directement via Gunicorn, ou configuration de Nginx pour une solution plus robuste. Les fichiers médias ont été stockés sur AWS S3, avec la possibilité d'utiliser CloudFront comme CDN afin d'améliorer les performances et réduire la latence

- Extrait de code :

```
STATIC_URL = '/static/'  
MEDIA_URL = '/images/'  
  
STATICFILES_DIRS = [  
    BASE_DIR / 'static',  
    BASE_DIR / 'frontend/build/static'  
]  
  
MEDIA_ROOT = BASE_DIR / 'static/images'  
STATIC_ROOT = BASE_DIR / 'staticfiles'  
  
CORS_ALLOW_ALL_ORIGINS = True
```

5.2.4. Gestion des transactions concurrentes

- **Problématique** : Lorsqu'un article en stock est commandé simultanément par plusieurs utilisateurs, des conditions de course peuvent survenir, menant à des stocks négatifs si les opérations de vérification et de décrémentation ne sont pas atomiques.
- **Solution apportée** : Solution apportée : Utilisation des transactions atomiques combinées à `select_for_update()` pour verrouiller les lignes de produit concernées pendant la commande. Ainsi, toute tentative simultanée de commande est gérée de manière cohérente et sécurisée.

Conclusion Générale

Bilan Global

Le projet **ProShop** a atteint l'ensemble de ses objectifs majeurs en posant les fondations d'une plateforme e-commerce moderne, performante et évolutive. La maîtrise technique de la stack Django 5.0 et PostgreSQL, combinée à une architecture RESTful solide, a permis de concevoir un système fonctionnel et sécurisé, répondant aux besoins des utilisateurs finaux et des commerçants.

1. Objectifs Atteints

La plateforme couvre de manière robuste la gestion du catalogue produit, du stock, des commandes, ainsi que le traitement sécurisé des paiements. Elle offre une expérience utilisateur fluide, notamment grâce à un frontend responsive et une navigation optimisée. Le respect des contraintes techniques imposées a été validé par des performances satisfaisantes lors des phases de test.

2. Apports Techniques

Le backend, structuré autour de Django REST Framework, garantit une API performante et sécurisée, intégrant Stripe et PayPal pour le paiement. La gestion du stockage d'images via AWS S3 assure la scalabilité des contenus médias. Côté frontend, React.js et Bootstrap ont permis de construire une interface intuitive et efficace. Enfin, le déploiement sur AWS, accompagné d'une configuration optimisée de Gunicorn, Nginx et Whitenoise, assure la disponibilité et la montée en charge de la plateforme.

3. Retours Utilisateurs

Les premiers retours soulignent la simplicité d'utilisation et la fluidité du parcours d'achat. Les commerçants apprécient la facilité de gestion des produits. Les axes d'amélioration proposés concernent principalement l'ajout d'options de livraison, le suivi en temps réel des colis, et le perfectionnement du moteur de recherche.

4. Leçons Apprises

La gestion des stocks en temps réel et la sécurisation des transactions ont représenté des défis techniques majeurs, soulignant l'importance d'une architecture solide. L'expérience a également mis en lumière la nécessité d'une communication claire entre les différents acteurs du projet.

5. Perspectives

À court terme, il est prévu d'élargir les options de paiement et livraison et d'optimiser davantage l'expérience utilisateur. Le développement d'une application mobile et la mise en place d'outils marketing pour les vendeurs sont envisagés à moyen terme. Sur le long terme, **ProShop** ambitionne de devenir une marketplace **multi-vendeurs** intégrée à des systèmes ERP et dotée d'une personnalisation avancée via l'intelligence artificielle.

6. Conclusion

ProShop constitue une base solide pour une solution e-commerce évolutive et performante, avec un fort potentiel d'adaptation aux besoins du marché. Ce projet obtient une note globale de ★★★★☆ (4/5), reflétant la qualité de sa réalisation et la clarté de sa vision stratégique.

Bibliographie / Références

Voici une sélection de documentations officielles et de ressources clés pour les technologies utilisées dans ProShop:

1. Django & Django REST Framework

- **Django Software Foundation. Django Documentation (v5.0).**
<https://docs.djangoproject.com/fr/5.0/>
- **Encode. Django REST Framework Documentation.**
<https://www.django-rest-framework.org/>
- **JustDjango. Build an E-commerce Website with Django. YouTube Playlist.** <https://www.youtube.com/c/JustDjango>

2. PostgreSQL

- **PostgreSQL Global Development Group. PostgreSQL 15 Documentation.** <https://www.postgresql.org/docs/15/index.html>
- **IBM. PostgreSQL Indexing Best Practices.**
<https://www.ibm.com/docs/en/db2/11.5?topic=performance-indexing-best-practices>
- **Cybertec. PostgreSQL Performance Tuning Tips.**
<https://www.cybertec-postgresql.com/en/postgresql-performance-tuning/>

3. Authentification JWT

- **djangorestframework-simplejwt Documentation.** <https://django-rest-framework-simplejwt.readthedocs.io/en/latest/>
- **IETF. RFC 7519 – JSON Web Token (JWT).**
<https://datatracker.ietf.org/doc/html/rfc7519>
- **Auth0. JWT Security Best Practices.** <https://auth0.com/blog/jwt-best-practices/>

4. Paiement Sécurisé

- **Stripe.** Stripe API Reference. <https://stripe.com/docs/api>
- **PayPal.** PayPal Developer Docs.
<https://developer.paypal.com/docs/api/overview/>

5. Déploiement & Infrastructure Cloud

- **Gunicorn.** Gunicorn Documentation.
<https://docs.gunicorn.org/en/stable/>
- **Nginx.** Nginx Documentation. <https://nginx.org/en/docs/>
- **AWS.** Hosting Django with AWS S3 and EC2.
<https://docs.aws.amazon.com/>
- **Whitenoise.** Serving Static Files in Production.
<https://whitenoise.evans.io/en/stable/>

6. Tests & Qualité Logicielle

- **pytest-django.** pytest-django Documentation. <https://pytest-django.readthedocs.io/en/latest/>
- **Locust.** Load Testing with Locust. <https://docs.locust.io/en/stable/>
- **Sentry.** Sentry for Django.
<https://docs.sentry.io/platforms/python/guides/django/>

7. Interface Utilisateur

- **W3C.** WCAG 2.1 en français.
<https://www.w3.org/Translations/WCAG21-fr/>
- **Bootstrap.** Bootstrap 5 Documentation.
<https://getbootstrap.com/docs/5.0/getting-started/introduction/>
- **Meta.** React.js Documentation. <https://react.dev/>

8. Outils Recommandés

- Docker. Official Docker Documentation. <https://docs.docker.com/>
- GitHub Docs. GitHub Actions Documentation. <https://docs.github.com/en/actions>
- Sphinx. Sphinx Documentation. <https://www.sphinx-doc.org/en/master/>

