ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

PROJET DE FIN D'ANNÉE

5ème Année en Ingénierie Informatique et Réseaux

Plateforme E-Commerce Centralisée
des Produits multi-sources avec un Système de Recommandation IA

Réalisé par :

Zineb Semane
El Assli Houssaine
Ayoub Mhandi

Encadrant Pédagogique :

Abderrahim Larhlimi

Présenté devant le jury :

M. CHIBA
M. Larhlimi

Année universitaire : 2025/2026

Table des matières

Introduction générale **1**

1 Présentation du cadre de projet **2**

1.1 Introduction 2

1.2 Contexte du projet 2

1.2.1 Description du projet 2

1.2.2 Informations générales 2

1.2.3 Objectifs du projet 3

1.3 Étude de l'existant 3

1.3.1 Description de l'existant 3

1.3.2 Critique de l'existant 4

1.3.3 Solution proposée 4

1.4 Architecture microservices 4

1.4.1 Services d'infrastructure (3 services) 4

1.4.2 Services métier core (4 services) 5

1.4.3 Services d'agrégation (1-2 services - En développement) 5

1.5 Choix du modèle de développement 5

1.6 Planning prévisionnel 5

1.7 Conclusion 6

2 Spécification des besoins **7**

2.1 Introduction 7

2.2 Spécification des besoins fonctionnels 7

2.2.1 Besoin fonctionnel global 1 : Gestion des utilisateurs 7

2.2.2 Besoin fonctionnel global 2 : Gestion des produits 8

2.2.3 Besoin fonctionnel global 3 : Gestion des commandes 8

2.2.4 Besoin fonctionnel global 4 : Gestion des paiements 8
2.2.5 Besoin fonctionnel global 5 : Agrégation de produits externes 9
2.2.6 Besoin fonctionnel global 6 : Intelligence Artificielle et Recommandations 9
2.3 Spécification des besoins non fonctionnels 9
2.4 Présentation des cas d'utilisation 10
2.4.1 Présentation des acteurs 10
2.4.2 Description des cas d'utilisation 10
2.4.3 Diagramme des cas d'utilisation global 11

3 Conception du système **12**
3.1 Introduction 12
3.2 Modélisation dynamique 12
3.2.1 Diagrammes de séquences 12
3.2.2 Diagrammes de collaboration 17
3.2.3 Diagrammes d'activité 17
3.2.4 Diagrammes d'états 19
3.2.5 Diagrammes d'état-transition 19
3.3 Modélisation statique 22
3.3.1 Diagramme de classes 22
3.3.2 Diagramme de packages (organisation du code) 23
3.4 Architecture de l'application 24
3.4.1 Architecture logicielle 24
3.4.2 Architecture matérielle 25
3.5 Modèle de données 25
3.5.1 Modèle relationnel 25
3.5.2 Dictionnaire de données 26
3.5.3 Diagramme de classe de la base de données (ERD) 27
3.6 Synthèse des diagrammes UML 28
3.7 Conclusion 28

4 Réalisation du système **29**
4.1 Introduction 29
4.2 Environnement de développement 29
4.2.1 Environnement matériel 29
4.2.2 Environnement logiciel 29

4.3 Architecture réalisée . . . 32
4.3.1 Structure du projet . . . 32
4.3.2 Communication inter-services . . . 33
4.3.3 Résilience et tolérance aux pannes . . . 33
4.4 Principales interfaces graphiques . . . 33
4.4.1 Page d'accueil (Dashboard) . . . 34
4.4.2 Catalogue de produits . . . 34
4.4.3 Gestion du panier et commandes . . . 34
4.4.4 Interface de paiement . . . 34
4.4.5 Tableau de bord administrateur . . . 34
4.4.6 Chatbot intelligent . . . 34
4.5 Fonctionnalités implémentées . . . 34
4.5.1 Services core complétés . . . 34
4.5.2 Services en développement . . . 35
4.6 Endpoints API principaux . . . 35
4.6.1 User Service . . . 35
4.6.2 Product Service . . . 35
4.6.3 Order Service . . . 35
4.6.4 Payment Service . . . 36
4.7 Intelligence Artificielle . . . 36
4.7.1 Intégration d'OpenAI GPT-4 . . . 36
4.7.2 Chatbot intelligent . . . 36
4.7.3 AI Recommendation Service (En développement) . . . 36
4.8 Conclusion . . . 37

Conclusion générale **38**

Bibliographie et Nétographie **39**

A Interfaces graphiques secondaires **41**

A.1 Interface principale de la plateforme MarketPlace . . . 41
A.2 Interface de gestion des produits . . . 42
A.3 Interface de gestion des utilisateurs . . . 42
A.4 Interface du chatbot . . . 43
A.5 Interface de paiement et validation de commande . . . 43

A.6 Interface du chatbot intelligentavec recommandations IA 44

Table des figures

2.1 Diagramme des cas d'utilisation global [20] 11

3.1 Diagramme de séquence : Authentification [9] 13
3.2 Diagramme de séquence : Paiement et validation de commande [11] 14
3.3 Diagramme de séquence : Création de produit par l'administrateur [9] 15
3.4 Diagramme de séquence : Recherche avec agrégation multi-sources [21] 16
3.5 Diagramme de séquence : Recommandation IA [15] 16
3.6 Diagramme de communication : Processus de paiement [11] 17
3.7 Diagramme d'activité : Processus de paiement [11] 18
3.8 Diagramme d'activité : Processus de recherche avec agrégation [21] 19
3.9 Diagramme d'états-transitions : Cycle de vie d'une commande [8] 20
3.10 Diagramme d'états-transitions : Cycle de vie d'un paiement [11] 21
3.11 Diagramme d'états-transitions : Cycle de vie d'un produit [8] 21
3.12 Diagramme de classes du système e-commerce [20] 22
3.13 Diagramme de packages : Organisation du code [20] 23
3.14 Diagramme de composants : Architecture modulaire 24
3.15 Diagramme de déploiement : Architecture matérielle [2] 25
3.16 Diagramme entité-association (ERD) : Modèle de données [3] 27

4.1 Java 17 (LTS) – Environnement de développement backend 30
4.2 Structure générale d'un microservice Spring Boot [5] 30
4.3 Architecture Spring Cloud de la plateforme e-commerce microservices [6] 31
4.4 Interface utilisateur développée avec React.js [16] 32
4.5 OpenAI GPT-4 pour l'intelligence artificielle et les recommandations [15] 36

A.1 Interface principale MarketPlace - Catalogue produits et chatbot IA intégrés 41
A.2 Interface principale de la plateforme MarketPlace 42

A.3 Interface principale de la plateforme MarketPlace 42
A.4 Interface principale de la plateforme MarketPlace 43
A.5 Interface principale de la plateforme MarketPlace 43
A.6 Interface du chatbot IA - Recommandations intelligentes et comparaison de produits 44

Liste des tableaux

1.1 Planning prévisionnel 5

2.1 Description du cas d'utilisation « Passer une commande » 10
2.2 Description du cas d'utilisation « Paiement PayPal » 10
2.3 Description du cas d'utilisation « Recherche de produits avec agrégation » 11

3.1 Dictionnaire de données – Tables principales 26
3.2 Récapitulatif des diagrammes UML produits 28

Introduction générale

Dans un contexte marqué par la transformation numérique et l'évolution constante du commerce électronique [1], le développement de plateformes e-commerce modernes et intelligentes constitue un enjeu majeur pour les entreprises. Ce projet de fin d'année porte sur la conception et la réalisation d'une plateforme e-commerce centralisée reposant sur une architecture microservices [6], permettant l'agrégation de produits de sources multiples et l'intégration de systèmes d'intelligence artificielle pour les recommandations [15].

La problématique principale est la suivante : comment concevoir une plateforme e-commerce moderne, sécurisée et évolutive, basée sur une architecture microservices distribuée, capable de répondre aux besoins fonctionnels et non fonctionnels des utilisateurs tout en garantissant performance, maintenabilité et intégration d'APIs externes pour l'agrégation de produits et les paiements en ligne [11].

Ce rapport est structuré en quatre chapitres principaux [4]. Le premier chapitre présente le cadre général du projet, l'étude de l'existant et la solution proposée. Le deuxième chapitre détaille la spécification des besoins fonctionnels et non fonctionnels. Le troisième chapitre est consacré à la conception du système avec une modélisation UML complète [20]. Enfin, le quatrième chapitre présente la réalisation de la solution développée, incluant l'environnement de développement et les principales interfaces graphiques [16].

Chapitre 1

Présentation du cadre de projet

1.1 Introduction

Ce chapitre présente le contexte général du projet [4], l'étude de l'existant ainsi que la solution proposée basée sur une architecture microservices moderne [2]. Il sert de transition avec le reste du document et définit les objectifs à atteindre.

1.2 Contexte du projet

1.2.1 Description du projet

Ce projet consiste en la conception et le développement d'une plateforme e-commerce centralisée permettant aux utilisateurs de rechercher et comparer des produits provenant de multiples sources (catalogue interne et sources externes comme Amazon) via une interface unifiée [1]. La monétisation s'effectue via des liens d'affiliation sur les produits externes.

La plateforme intègre également un système d'intelligence artificielle pour les recommandations de produits [15], avec un chatbot intelligent permettant aux utilisateurs d'interagir naturellement avec le système.

1.2.2 Informations générales

- **Type de Projet** : Projet académique (PFA - Projet de Fin d'Année)
- **Période** : Octobre - Décembre 2025
- **Durée** : 8 semaines (2 mois)
- **Architecture** : Microservices distribués (7-8 services) [2]
- **Paradigme** : Architecture microservices avec écosystème Spring Cloud [6]

1.2.3 Objectifs du projet

Objectifs fonctionnels

- Catalogue produits interne consultable avec gestion des catégories
- Système de recommandation pour recherche de produits externes [15]
- Agrégation multi-sources en temps réel (catalogue interne + Amazon) [21]
- Comparaison unifiée des produits
- Génération de revenus via affiliation
- Gestion complète des commandes et paiements en ligne [11]
- Chatbot intelligent pour assistance utilisateur

Objectifs techniques

- Architecture microservices distribuée avec Spring Cloud [6]
- Intégration APIs externes (PayPal [11], Amazon [21])
- Communication inter-services robuste via OpenFeign [13]
- Résilience et gestion d'erreurs (Circuit Breakers) [17]
- Frontend moderne avec React [16]
- Système d'intelligence artificielle pour recommandations [15]

1.3 Étude de l'existant

Cette partie comprend généralement trois parties : la description de l'existant, la critique de l'existant et la solution proposée [4].

1.3.1 Description de l'existant

Il est question d'expliquer comment le travail s'effectue actuellement dans le domaine des plateformes e-commerce [1]. Les plateformes e-commerce traditionnelles présentent généralement une architecture monolithique où tous les composants sont étroitement couplés. Cette approche présente plusieurs limitations [2] :

- Difficulté de scaling indépendant des composants,
- Risques de panne en cascade,
- Complexité de maintenance et d'évolution,
- Limitation de l'intégration avec des services externes,
- Absence de systèmes intelligents de recommandation [15].

1.3.2 Critique de l'existant

Cette partie permet d'évoquer les insuffisances de la solution actuelle. En effet, les points faibles de la solution actuelle doivent figurer dans cette section et aussi dans l'application qui sera développée. Les principales limites de l'existant sont [2] :

- Absence de centralisation des produits de multiples sources [21],
- Risques d'erreurs humaines dans la gestion manuelle,
- Manque de sécurité et de résilience [17],
- Difficulté de suivi et de monitoring,
- Absence de systèmes intelligents de recommandation [15],
- Limitation de l'expérience utilisateur [16].

1.3.3 Solution proposée

La solution proposée consiste à développer une plateforme e-commerce moderne basée sur une architecture microservices [2], séparant clairement le frontend (React [16]) et le backend (Spring Boot [5]). Cette solution permettra [1] :

- D'automatiser les processus de vente et de gestion,
- D'améliorer l'expérience utilisateur avec un chatbot intelligent [15],
- De faciliter la maintenance et l'évolution du système grâce à l'architecture modulaire,
- D'intégrer facilement des services externes (PayPal [11], Amazon [21]),
- De garantir la résilience grâce aux circuit breakers [17],
- De permettre le scaling horizontal indépendant de chaque service [6].

1.4 Architecture microservices

La plateforme repose sur une architecture microservices avec **7-8 services indépendants** [2] :

1.4.1 Services d'infrastructure (3 services)

0 **Config Server** (Port 8888) : Configuration centralisée pour tous les microservices avec support de profils d'environnement et rafraîchissement dynamique [6].

0 **Eureka Server** (Port 8761) : Service Discovery et registre de services avec dashboard web pour monitoring et découverte automatique des microservices [12].

0 **API Gateway** (Port 8080) : Point d'entrée unique pour toutes les requêtes avec routage intelligent, load balancing automatique, circuit breakers pour tolérance aux pannes et configuration CORS centralisée [6].

1.4.2 Services métier core (4 services)

0 **User Service** (Port 8083) : Gestion des utilisateurs (CRUD complet) avec gestion des rôles (CLIENT, ADMIN), authentification et profils utilisateur. Base de données H2 dédiée (user_db).

0 **Product Service** (Port 8081) : Catalogue produits avec catégories intégrées, gestion du stock et disponibilité, recherche par catégorie et mot-clé. Base de données H2 dédiée (product_db).

0 **Order Service** (Port 8085) : Création et gestion des commandes avec gestion des statuts (PENDING, CONFIRMED, SHIPPED, DELIVERED, CANCELLED), vérification disponibilité via OpenFeign [13], mise à jour automatique du stock et annulation avec restauration du stock. Base de données H2 dédiée (order_db).

0 **Payment Service** (Port 8084) : Intégration PayPal [11] (sandbox et production) avec workflow complet (Create → Approve → Execute), gestion des transactions (PENDING, COMPLETED, FAILED, CANCELLED, REFUNDED) et historique des paiements par commande. Base de données H2 dédiée (payment_db).

1.4.3 Services d'agrégation (1-2 services - En développement)

0 **External Aggregator Service** (Port 8087) : Agrégation de produits externes (Amazon [21]), normalisation des données produits, génération automatique de liens d'affiliation et gestion d'erreurs et fallback [17].

0 **AI Recommendation Service** (Port 8086) : Recommandations simplifiées (sans OpenAI GPT-4 dans la version actuelle [15]), analyse basique des requêtes utilisateur, agrégation des résultats internes et externes et ranking simple des produits.

1.5 Choix du modèle de développement

Nous avons adopté une méthodologie Agile itérative permettant de développer progressivement les fonctionnalités, d'effectuer des tests réguliers et d'intégrer les améliorations au fur et à mesure de l'avancement du projet. Cette approche permet de gérer efficacement la complexité de l'architecture microservices [2] et d'adapter le développement aux contraintes temporelles. Le choix de cette méthodologie est justifié par la nécessité de développer plusieurs microservices indépendants de manière itérative et incrémentale [4].

1.6 Planning prévisionnel

Le planning prévisionnel devra être représenté comme suit [4] :

TABLE 1.1 – Planning prévisionnel

Étape	**Octobre**				**Novembre**				**Décembre**							
Semaine	1	2	3	4	1	2	3	4	1	2	3	4				
Étude préalable	X	X														
Conception		X	X													
Réalisation Infrastructure			X	X	X											
Réalisation Services Core				X	X											
Services Agrégation						X	X									
Tests et Validation							X	X								
Documentation									X							

1.7 Conclusion

Ce chapitre a permis de présenter le contexte général du projet, d'analyser l'existant et de définir la solution proposée basée sur une architecture microservices moderne [2]. Les différents objectifs déjà cités précédemment ont été présentés [4]. Le chapitre suivant est consacré à la spécification détaillée des besoins.

Chapitre 2

Spécification des besoins

2.1 Introduction

Ce chapitre présente l'analyse des besoins permettant d'identifier les fonctionnalités attendues du système ainsi que les contraintes auxquelles il doit répondre afin d'assurer sa qualité et son efficacité [4].

2.2 Spécification des besoins fonctionnels

2.2.1 Besoin fonctionnel global 1 : Gestion des utilisateurs

Inscription/Connexion

Un utilisateur peut créer un compte et se connecter au système. Le système gère l'authentification et la session utilisateur.

Rôles

Le système différencie les rôles suivants [1] :

- **CLIENT** : Peut consulter les produits, passer des commandes, effectuer des paiements [11]
- **ADMIN** : Peut gérer les utilisateurs, les produits, consulter toutes les commandes

Gestion des profils

Les utilisateurs peuvent consulter et modifier leur profil, consulter leur historique de commandes.

2.2.2 Besoin fonctionnel global 2 : Gestion des produits

Catalogue produits

Le système permet de consulter un catalogue de produits avec catégories intégrées [3]. Les produits incluent des informations telles que nom, description, prix, stock, catégorie, image.

Recherche et filtrage

Les utilisateurs peuvent rechercher des produits par mot-clé, filtrer par catégorie, consulter les détails d'un produit [20].

Gestion du stock

Le système gère automatiquement le stock des produits, met à jour la disponibilité lors des commandes et restaure le stock lors des annulations.

2.2.3 Besoin fonctionnel global 3 : Gestion des commandes

Passer une commande

Le client peut ajouter des produits au panier puis valider une commande. Le système vérifie automatiquement la disponibilité du stock et valide l'utilisateur.

Suivi des commandes

Le client peut consulter l'état de sa commande. Les statuts possibles sont : PENDING, CONFIRMED, SHIPPED, DELIVERED, CANCELLED.

Annulation

Le client peut annuler une commande en statut PENDING, ce qui restaure automatiquement le stock des produits.

2.2.4 Besoin fonctionnel global 4 : Gestion des paiements

Paiement en ligne

Le client peut effectuer un paiement via PayPal [11] (en mode sandbox pour les tests). Le workflow inclut la création du paiement, l'approbation par l'utilisateur et l'exécution du paiement.

Confirmation

Le système confirme le paiement et met à jour l'état de la commande. Les statuts de paiement possibles sont : PENDING, COMPLETED, FAILED, CANCELLED, REFUNDED.

2.2.5 Besoin fonctionnel global 5 : Agrégation de produits externes

Recherche multi-sources

Le système permet de rechercher des produits dans le catalogue interne et dans des sources externes (Amazon [21]). Les résultats sont agrégés et présentés de manière unifiée [2].

Génération de liens d'affiliation

Pour les produits externes, le système génère automatiquement des liens d'affiliation permettant la monétisation.

2.2.6 Besoin fonctionnel global 6 : Intelligence Artificielle et Recommandations

Chatbot intelligent

Un chatbot permet aux utilisateurs d'interagir naturellement avec le système pour rechercher des produits, obtenir des recommandations et obtenir de l'aide [15].

Recommandations

Le système fournit des recommandations de produits basées sur les recherches de l'utilisateur et son historique (en développement) [15].

2.3 Spécification des besoins non fonctionnels

Ce sont les besoins qui permettraient d'améliorer la qualité des services de l'application comme la convivialité et l'ergonomie des interfaces, l'amélioration du temps de réponse, etc. Il est également possible de les présenter sous forme de puces [1].

- **Performance** : Temps de réponse acceptable pour les opérations courantes (< 2 secondes pour la plupart des requêtes), support de la charge simultanée de plusieurs utilisateurs [5],
- **Sécurité** : Protection des accès et des données sensibles, validation des entrées utilisateur, gestion sécurisée des paiements [11],
- **Disponibilité** : Accès continu aux services principaux grâce aux mécanismes de résilience (circuit breakers, fallback) [17],
- **Maintenabilité** : Code structuré et facilement évolutif grâce à l'architecture microservices [2], documentation complète [4],
- **Ergonomie** : Interface web claire, responsive et intuitive [16], chatbot accessible et facile à utiliser [15],
- **Scalabilité** : Architecture permettant le scaling horizontal indépendant de chaque service [6],
- **Interopérabilité** : Intégration avec APIs externes (PayPal [11], Amazon [21]) via des interfaces standardisées (REST).

2.4 Présentation des cas d'utilisation

2.4.1 Présentation des acteurs

Les acteurs principaux sont [4] :

- **Client** : Utilisateur final qui consulte les produits, passe des commandes et effectue des paiements [11],
- **Administrateur** : Gère les utilisateurs, les produits et consulte toutes les commandes.

2.4.2 Description des cas d'utilisation

TABLE 2.1 – Description du cas d'utilisation « Passer une commande »

Cas n°	UC-01
Acteur	Client
Objectif	Valider le panier et créer une commande
Pré-condition	Client authentifié, produits disponibles en stock
Post-condition	Commande créée avec statut PENDING, stock mis à jour
Scénario principal	1. Client sélectionne produits et les ajoute au panier 2. Client valide le panier 3. Système vérifie disponibilité stock 4. Système vérifie utilisateur 5. Système crée commande 6. Système réduit stock 7. Système calcule montant total

TABLE 2.2 – Description du cas d'utilisation « Paiement PayPal »

Cas n°	UC-02
Acteur	Client
Objectif	Effectuer le paiement d'une commande via PayPal
Pré-condition	Commande créée avec statut PENDING
Post-condition	Paiement effectué, commande confirmée
Scénario principal	1. Client initie paiement PayPal [11] 2. Système crée transaction PayPal 3. Système redirige vers PayPal 4. Client approuve paiement 5. Système exécute paiement 6. Système met à jour statut commande

TABLE 2.3 – Description du cas d'utilisation « Recherche de produits avec agrégation »

Cas n°	UC-03
Acteur	Client
Objectif	Rechercher des produits dans catalogue interne et externe
Pré-condition	Client connecté ou anonyme
Post-condition	Liste unifiée de produits affichée
Scénario principal	1. Client saisit requête de recherche 2. Système recherche dans catalogue interne 3. Système recherche dans sources externes (Amazon [21]) 4. Système normalise résultats 5. Système génère liens affiliation pour produits externes 6. Système agrège et classe résultats [2] 7. Système affiche liste unifiée

2.4.3 Diagramme des cas d'utilisation global

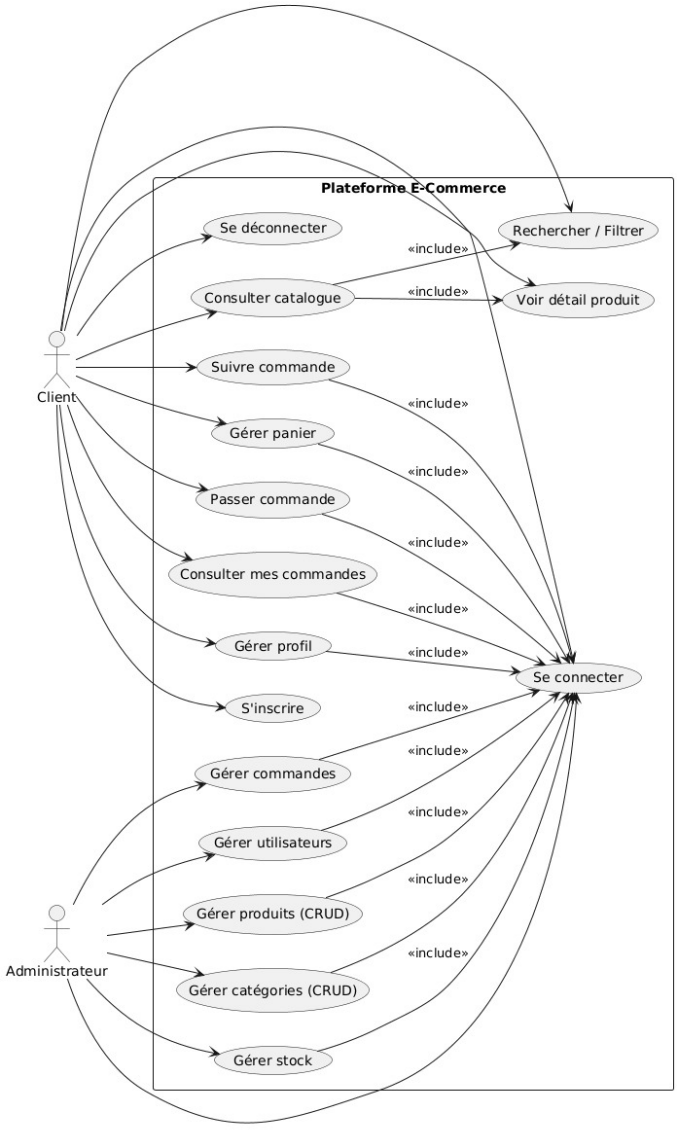

FIGURE 2.1 – Diagramme des cas d'utilisation global [20]

Chapitre 3

Conception du système

3.1 Introduction

La phase de conception constitue une étape essentielle dans le cycle de développement du système [4]. Elle permet de traduire les besoins fonctionnels et non fonctionnels exprimés précédemment en une solution technique claire et structurée [1]. Cette étape vise à définir l'architecture globale du système ainsi que les interactions entre ses différents composants [20].

3.2 Modélisation dynamique

La modélisation dynamique permet de décrire le comportement du système et les interactions entre les différents acteurs et composants au cours du temps [7, 8]. Elle facilite la compréhension du déroulement des traitements et des échanges d'informations entre les microservices [2].

3.2.1 Diagrammes de séquences

Un diagramme de séquence est un diagramme d'interaction dont le but est de décrire comment les objets collaborent au cours du temps et quelles responsabilités ils assument [9]. Il décrit un scénario d'un cas d'utilisation [4].

Avec les intéressants ajouts au diagramme de séquences apportés par UML 2, en particulier les cadres d'interactions (avec les opérateurs loop, opt et alt par exemple), ainsi que la possibilité de référencer une interaction décrite par ailleurs, le diagramme de séquence système nous semble constituer une excellente solution [20].

Les différents messages représentés par un diagramme de séquence sont [9] :

- **Message asynchrone** : Il n'attend pas de réponse et ne bloque pas l'émetteur qui ne sait pas si le message arrivera à destination,
- **Message synchrone** : L'émetteur reste alors blocqué le temps que dure l'invocation de l'opération,
- **Messages de création et destruction d'instance** : La création d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie et la destruction d'un objet est matérialisée par une croix qui marque la fin de la ligne de vie de l'objet.

Diagramme de séquence : Authentification

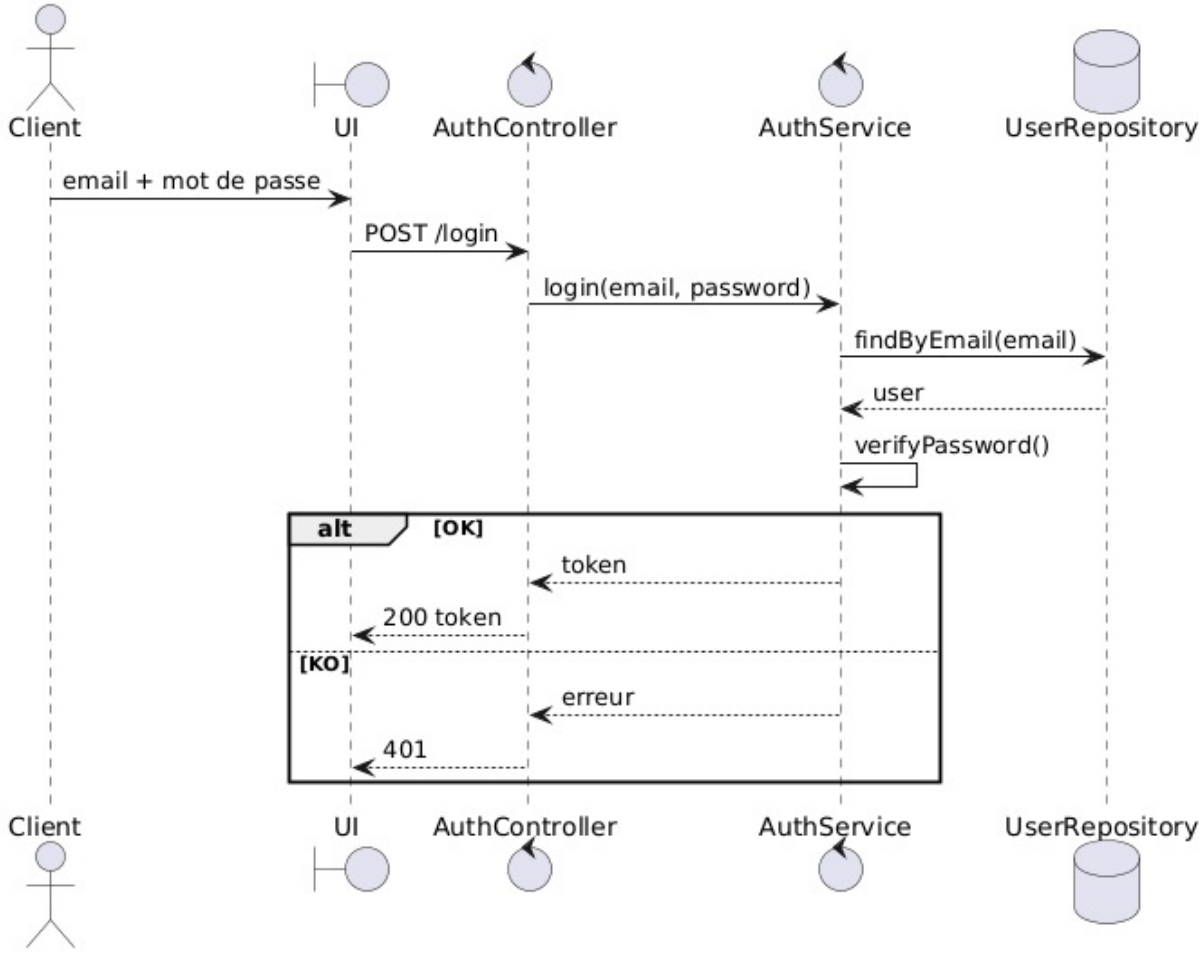

FIGURE 3.1 – Diagramme de séquence : Authentification [9]

Description : Ce diagramme illustre le processus d'authentification d'un utilisateur. L'utilisateur saisit ses identifiants via l'interface React [16], qui transmet la requête à l'API Gateway. Le Gateway route la requête vers le User Service qui valide les informations. Si les informations sont correctes, l'utilisateur est authentifié et redirigé vers le menu principal ; sinon, un message d'erreur est affiché.

Diagramme de séquence : Paiement et validation de commande

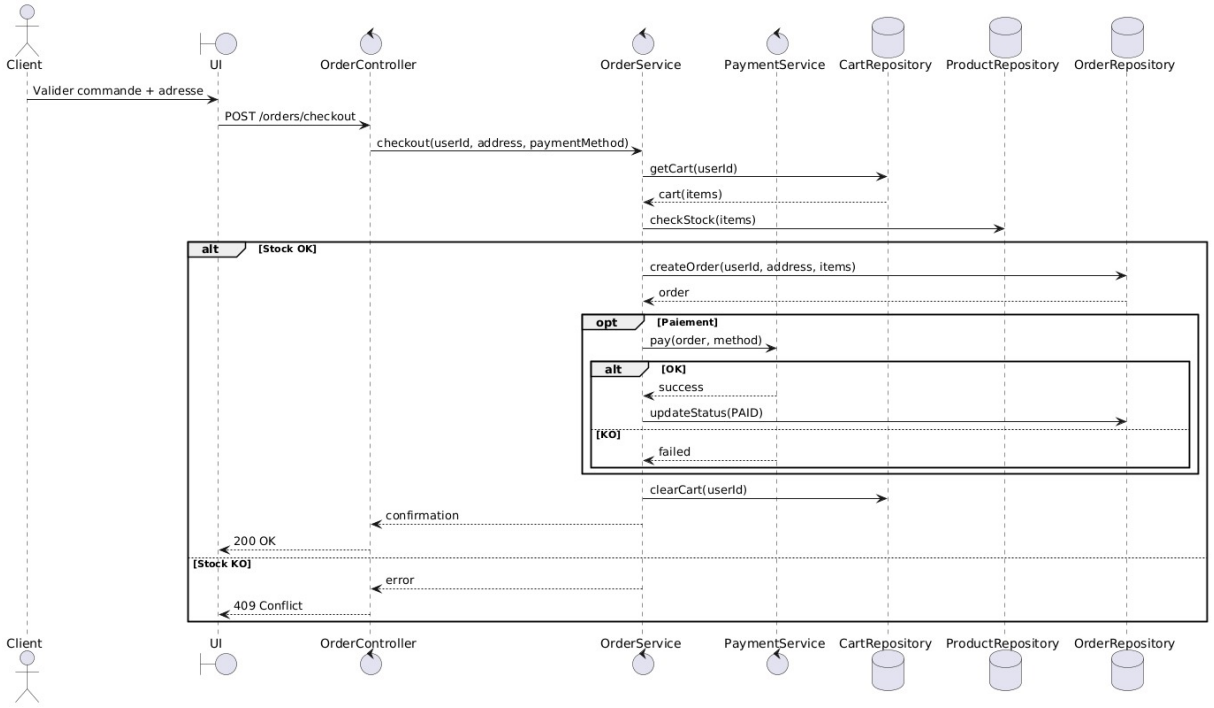

FIGURE 3.2 – Diagramme de séquence : Paiement et validation de commande [11]

Description : Ce diagramme présente le processus complet de paiement PayPal [11], depuis la validation du panier jusqu'à la confirmation de la commande. Il inclut l'interaction avec le Payment Service, la création de la transaction PayPal, la redirection vers PayPal, l'approbation utilisateur, l'exécution du paiement et la mise à jour du statut de la commande.

Diagramme de séquence : Création de produit (Admin)

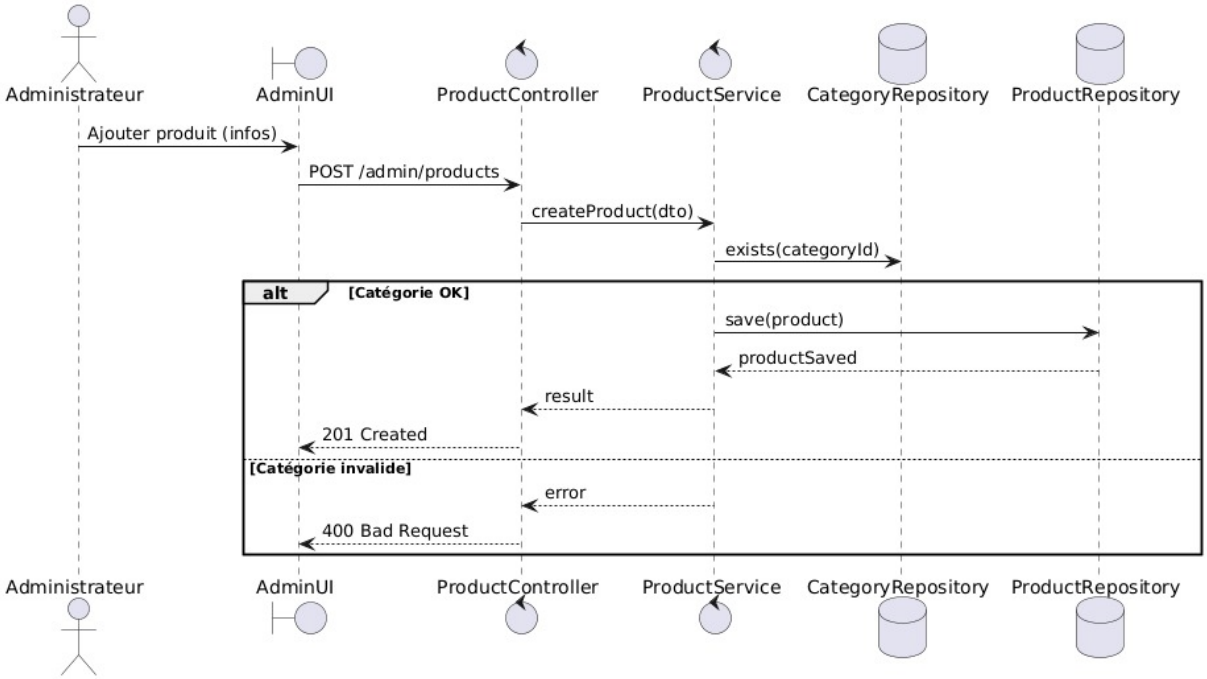

FIGURE 3.3 – Diagramme de séquence : Création de produit par l'administrateur [9]

Description : Ce diagramme montre comment un administrateur ajoute un nouveau produit au catalogue via l'interface d'administration. Le processus inclut la validation des données, la vérification des permissions administrateur, la création du produit dans le Product Service et la mise à jour de la base de données.

Diagramme de séquence : Recherche avec agrégation

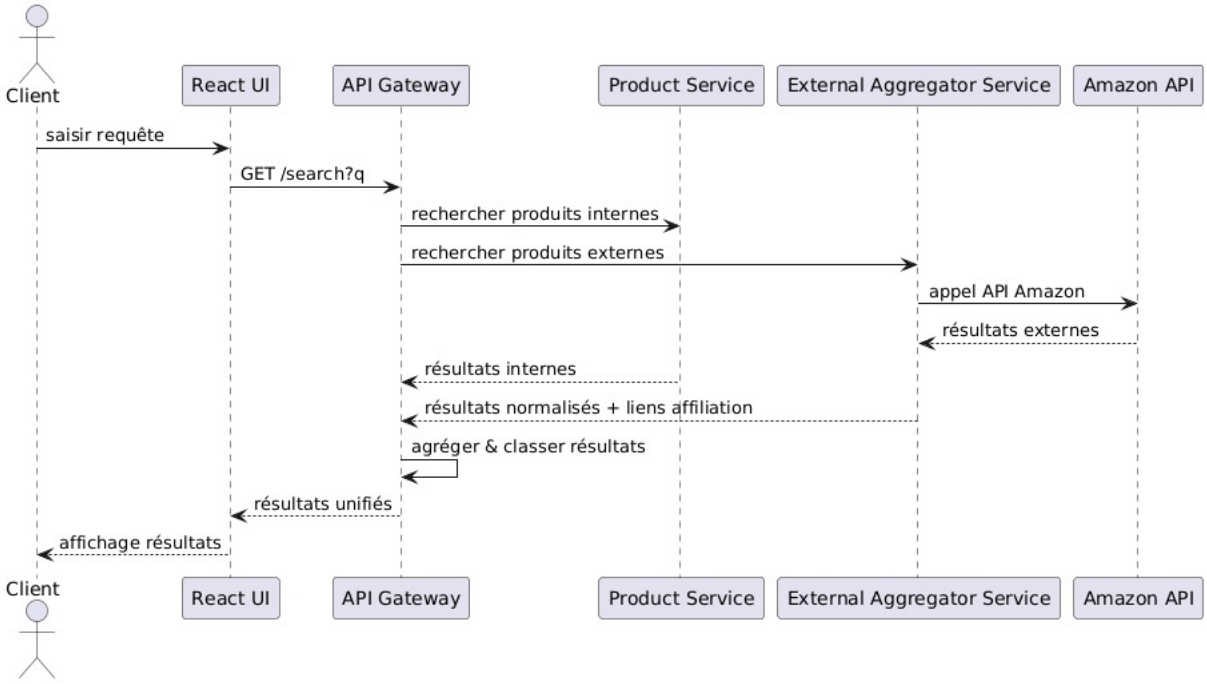

FIGURE 3.4 – Diagramme de séquence : Recherche avec agrégation multi-sources [21]

Description : Ce diagramme illustre le processus de recherche de produits avec agrégation. L'utilisateur saisit une requête, le système recherche dans le Product Service (produits internes), interroge l'External Aggregator Service qui appelle l'API Amazon [21], normalise les résultats, génère les liens d'affiliation et agrège les résultats pour affichage unifié.

Diagramme de séquence : Recommandation IA

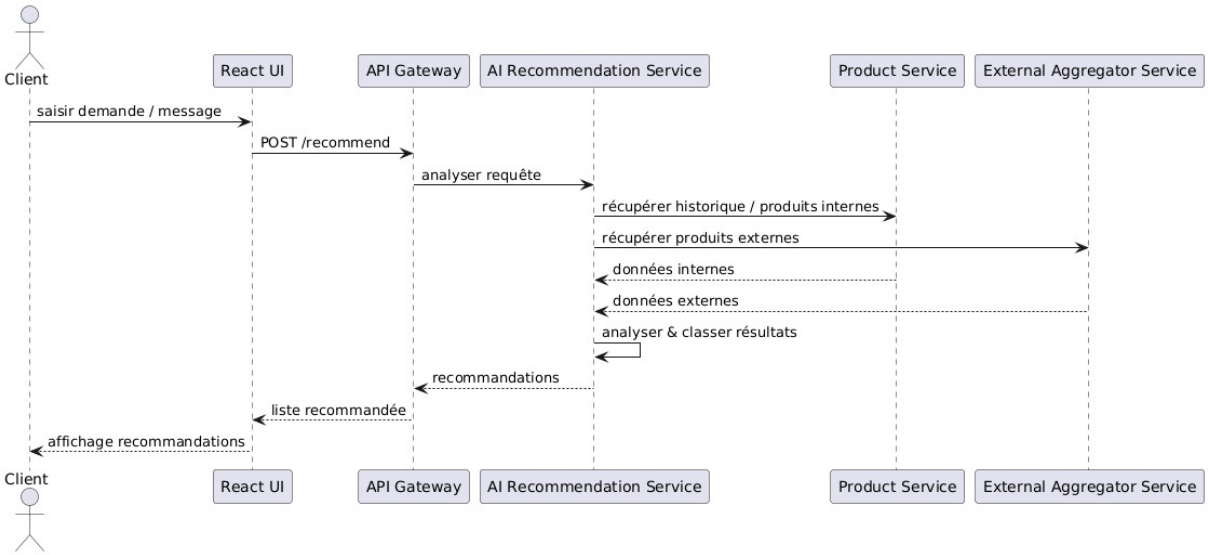

FIGURE 3.5 – Diagramme de séquence : Recommandation IA [15]

Description : Ce diagramme de séquence illustre le fonctionnement du service de recommandation basé sur l'intelligence artificielle [15]. La requête de l'utilisateur est transmise via l'interface utilisateur à l'API Gateway, qui la redirige vers le service de recommandation IA. Ce dernier analyse la demande, interroge le catalogue interne ainsi que les sources externes afin de récupérer les données nécessaires, puis génère une liste de recommandations personnalisées qui est renvoyée à l'utilisateur.

3.2.2 Diagrammes de collaboration

Les diagrammes de collaboration montrent des interactions entre objets (instances de classes et acteurs). En outre, ils permettent de représenter le contexte d'une interaction, car on peut y préciser les états des objets qui interagissent [20].

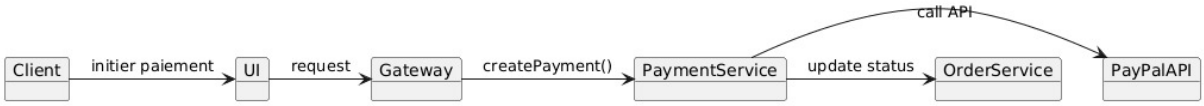

FIGURE 3.6 – Diagramme de communication : Processus de paiement [11]

Description : Ce diagramme de communication (anciennement diagramme de collaboration) montre les interactions entre les objets lors du processus de paiement, en mettant l'accent sur la structure des liens entre les instances des différents services (Order Service, Payment Service, PayPal API [11]).

3.2.3 Diagrammes d'activité

Les diagrammes d'activités permettent de mettre l'accent sur les traitements [7]. Ils sont donc particulièrement adaptés à la modélisation du cheminement de flots de contrôle et de flots de données. Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation [4].

Processus de paiement

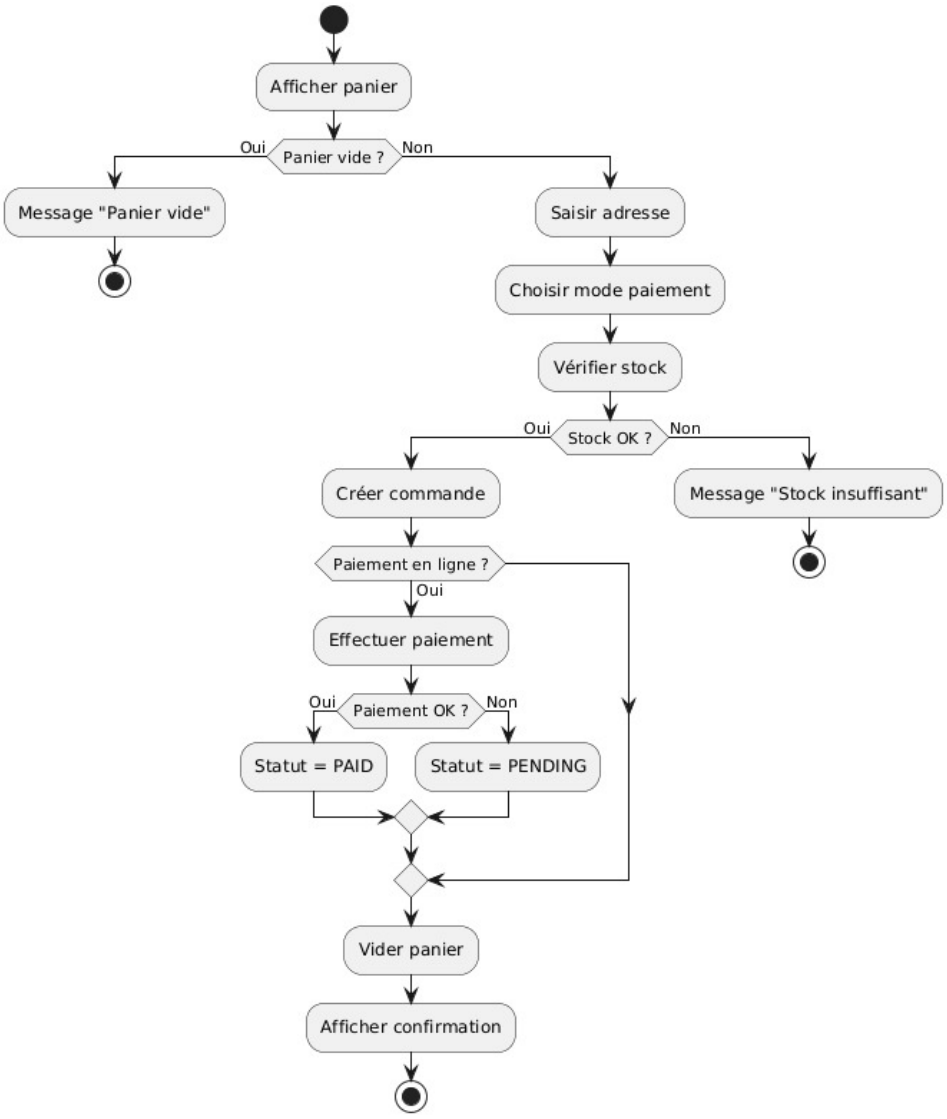

FIGURE 3.7 – Diagramme d'activité : Processus de paiement [11]

Description : Ce diagramme d'activité détaille les différentes étapes du processus de paiement PayPal [11], y compris les décisions (vérification stock, validation utilisateur), les branches parallèles (création transaction, mise à jour commande) et les conditions de succès/échec.

Processus de recherche avec agrégation

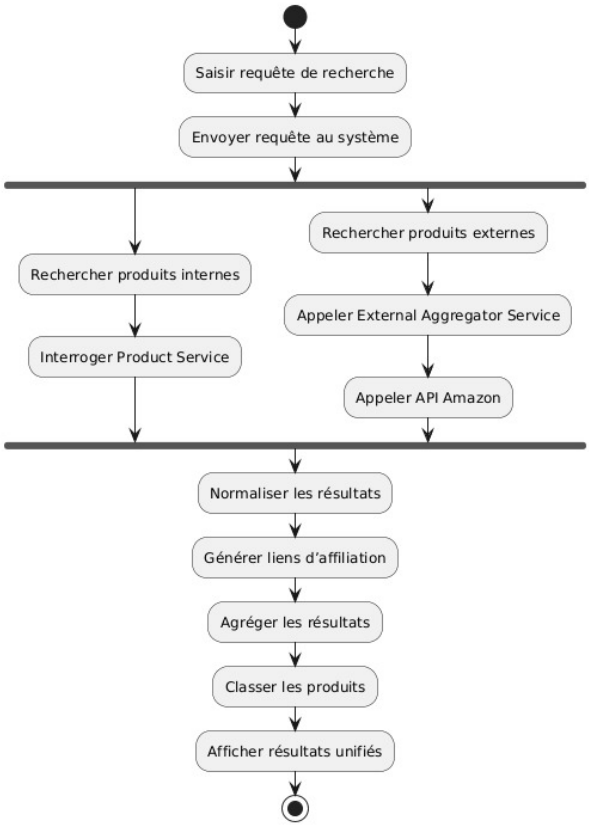

FIGURE 3.8 – Diagramme d'activité : Processus de recherche avec agrégation [21]

Description : Ce diagramme présente le flux d'activité pour la recherche de produits avec agrégation multi-sources [21], incluant la recherche parallèle dans le catalogue interne et les sources externes, la normalisation des résultats et l'agrégation finale.

3.2.4 Diagrammes d'états

3.2.5 Diagrammes d'état-transition

Les diagrammes d'états-transitions d'UML décrivent le comportement interne d'un objet à l'aide d'un automate à états finis [8]. Ils présentent les séquences possibles d'états et d'actions qu'une instance de classe peut traiter au cours de son cycle de vie en réaction à des événements discrets (de type signaux, invocations de méthode) [4].

État d'une commande

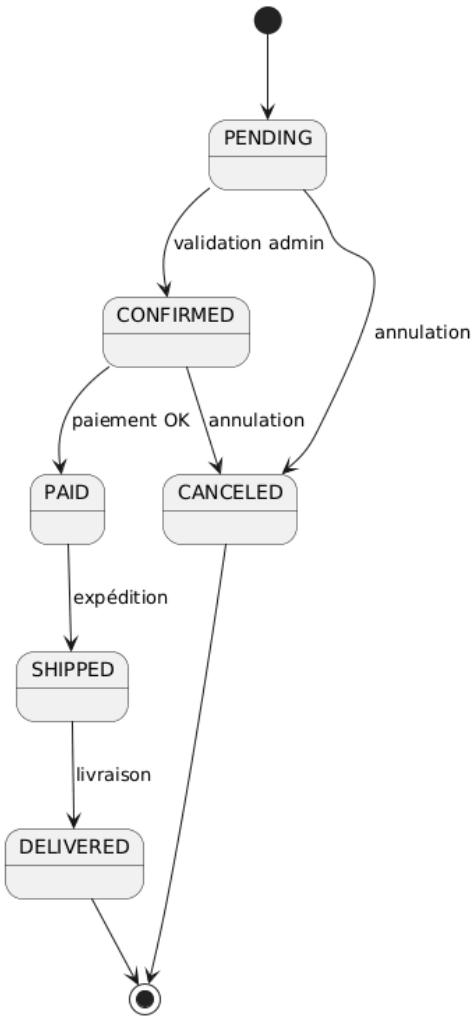

FIGURE 3.9 – Diagramme d'états-transitions : Cycle de vie d'une commande [8]

Description : Ce diagramme montre les différents états par lesquels passe une commande : PENDING (créée, en attente de paiement), CONFIRMED (paiement validé), SHIPPED (expédiée), DELIVERED (livrée), CANCELLED (annulée). Les transitions sont déclenchées par des événements spécifiques (paiement validé, expédition, livraison, annulation).

État du paiement

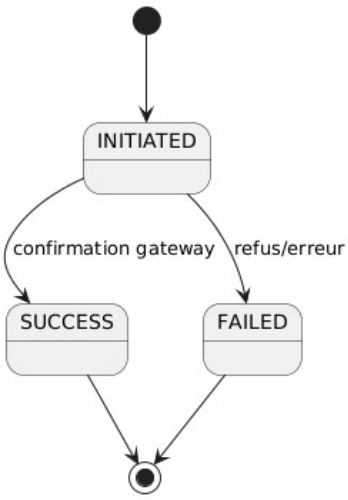

FIGURE 3.10 – Diagramme d'états-transitions : Cycle de vie d'un paiement [11]

Description : Ce diagramme illustre les transitions d'état d'un paiement : PENDING (transaction créée), COMPLETED (paiement validé), FAILED (échec du paiement), CANCELLED (annulé par l'utilisateur), REFUNDED (remboursé).

État d'un produit

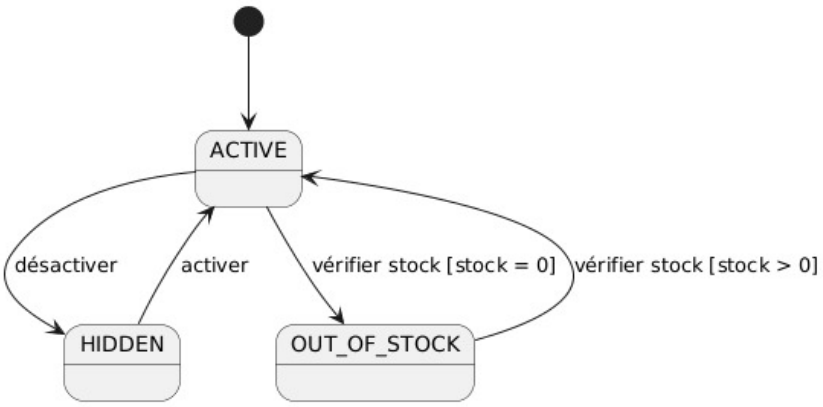

FIGURE 3.11 – Diagramme d'états-transitions : Cycle de vie d'un produit [8]

Description : Ce diagramme présente les états possibles d'un produit dans le catalogue : DISPONIBLE (en stock), RUPTURE_STOCK (stock épuisé), ARCHIVÉ (retiré du catalogue). Les transitions sont liées aux opérations de gestion du stock et de l'administration.

3.3 Modélisation statique

3.3.1 Diagramme de classes

Ici sera dressé le diagramme de classes de la future base de données [3, 1].

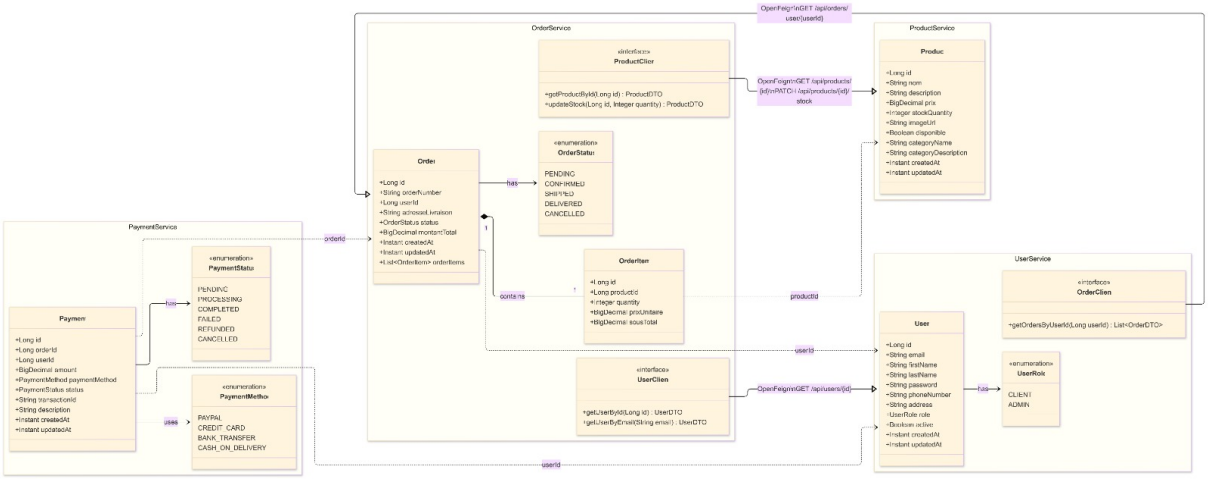

FIGURE 3.12 – Diagramme de classes du système e-commerce [20]

Description : Ce diagramme présente l'architecture microservices du projet e-commerce, montrant la répartition du système en services indépendants afin d'assurer une organisation claire et modulaire du code.

- **OrderService :** Gère les commandes des utilisateurs, y compris la création des commandes, le suivi de leur statut et la gestion des articles commandés.
- **ProductService :** Assure la gestion des produits et du stock. Il fournit les informations nécessaires sur les produits et met à jour les quantités disponibles.
- **UserService :** Gère les utilisateurs de la plateforme, leurs informations personnelles ainsi que leurs rôles (client ou administrateur).
- **PaymentService :** Prend en charge les paiements liés aux commandes et assure le suivi de l'état des transactions.

Relations principales : Le *OrderService* communique avec les services *ProductService*, *UserService* et *PaymentService* via des API REST (OpenFeign) afin de garantir le bon déroulement du processus de commande. Cette architecture favorise la modularité, la maintenabilité et l'évolutivité du système.

3.3.2 Diagramme de packages (organisation du code)

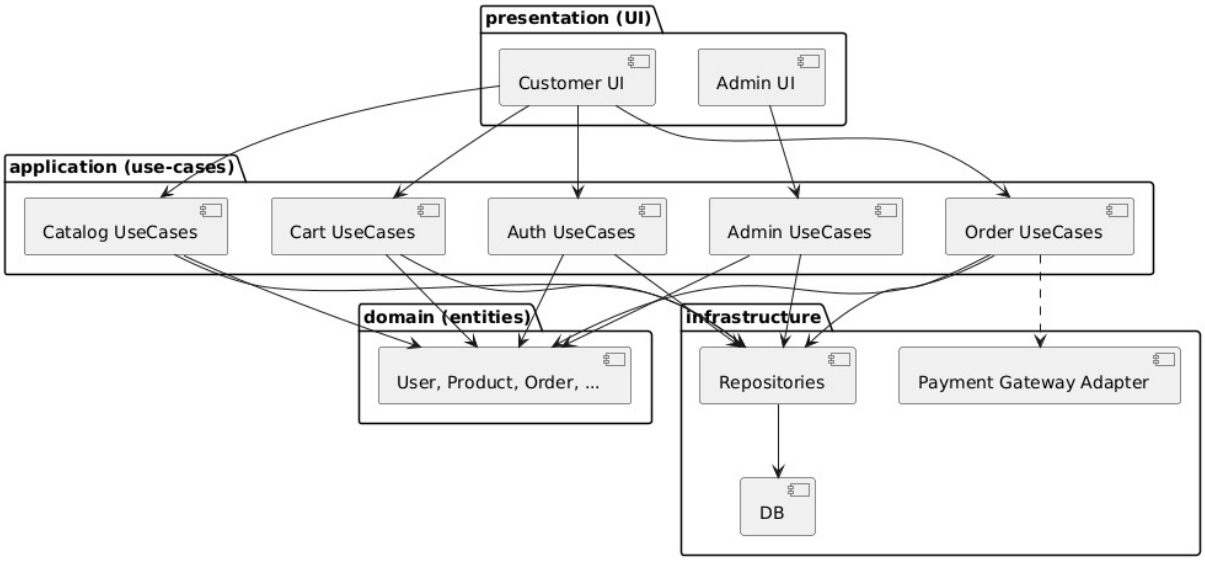

FIGURE 3.13 – Diagramme de packages : Organisation du code [20]

Description : Ce diagramme montre comment le projet est structuré en packages (modules) pour organiser le code de manière claire et modulaire [1] :

- **Présentation (UI) :** Contient les interfaces utilisateur, à savoir *Customer UI* (React [16]) pour le client et *Admin UI* (React [16]) pour l'administrateur.
- **Application (Use-Cases) :** Contient la logique métier spécifique à chaque fonctionnalité : authentification, gestion du catalogue, du panier, des commandes, des paiements et fonctions d'administration.
- **Domaine (Entities) :** Contient les objets métiers principaux comme *User*, *Product*, *Order*, *Payment*, etc.
- **Infrastructure :** Contient les composants techniques tels que *Repositories* pour l'accès aux données, *DB* pour la base de données H2, *Payment Gateway Adapter* pour la gestion des paiements PayPal [11], et les clients OpenFeign [13] pour la communication inter-services.

Relations principales : L'UI communique avec les cas d'utilisation, qui manipulent les entités et utilisent les repositories pour accéder à la base de données ou au service de paiement. Cette organisation en packages facilite la maintenance et l'évolution du code [4].

3.4 Architecture de l'application

3.4.1 Architecture logicielle

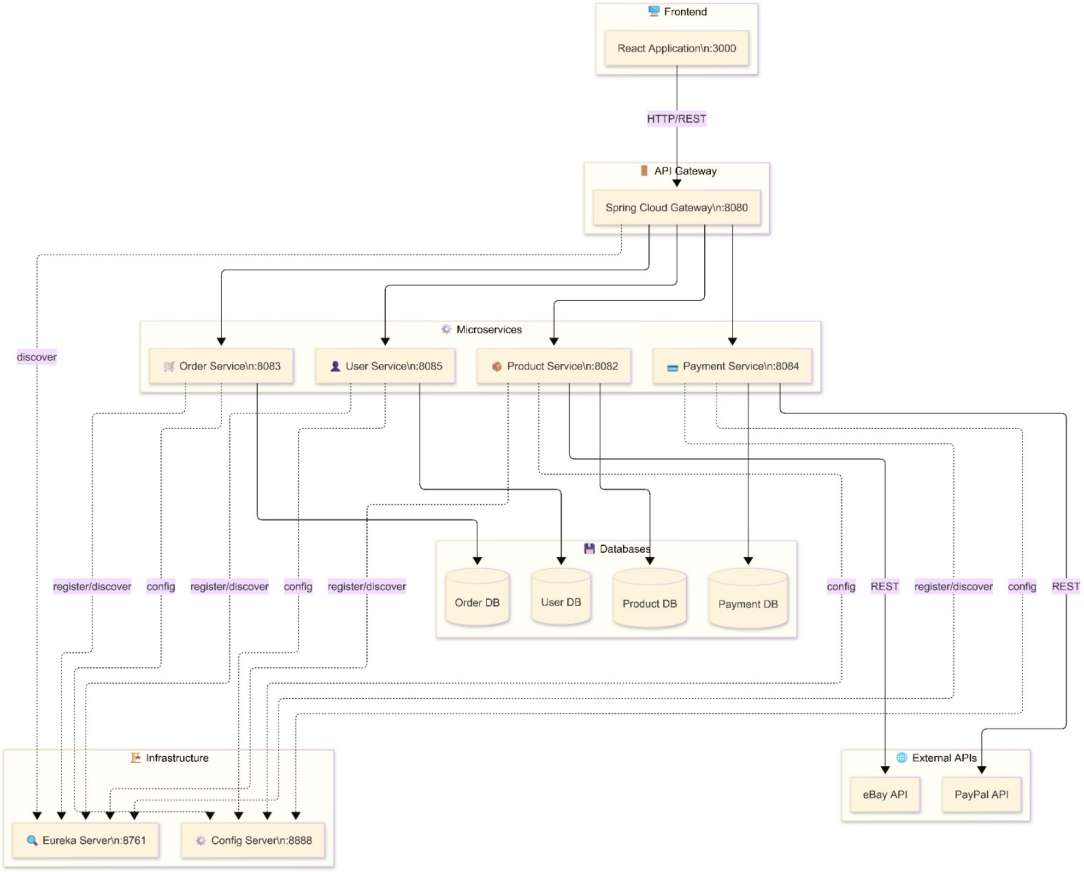

FIGURE 3.14 – Diagramme de composants : Architecture modulaire

Description : Ce diagramme de composants illustre l'architecture modulaire de la plateforme. Le **Frontend** (React) communique avec l'**API Gateway** (Spring Cloud Gateway) qui centralise toutes les requêtes. Les **Microservices** (Order Service, User Service, Product Service, Payment Service) sont indépendants avec leurs propres bases de données. L'**Infrastructure** inclut Eureka Server pour la découverte de services et Config Server pour la configuration centralisée. Les **APIs externes** (eBay, PayPal) sont intégrées via des appels REST. Cette architecture permet un scaling indépendant, une résilience améliorée et une maintenance facilitée.

3.4.2 Architecture matérielle

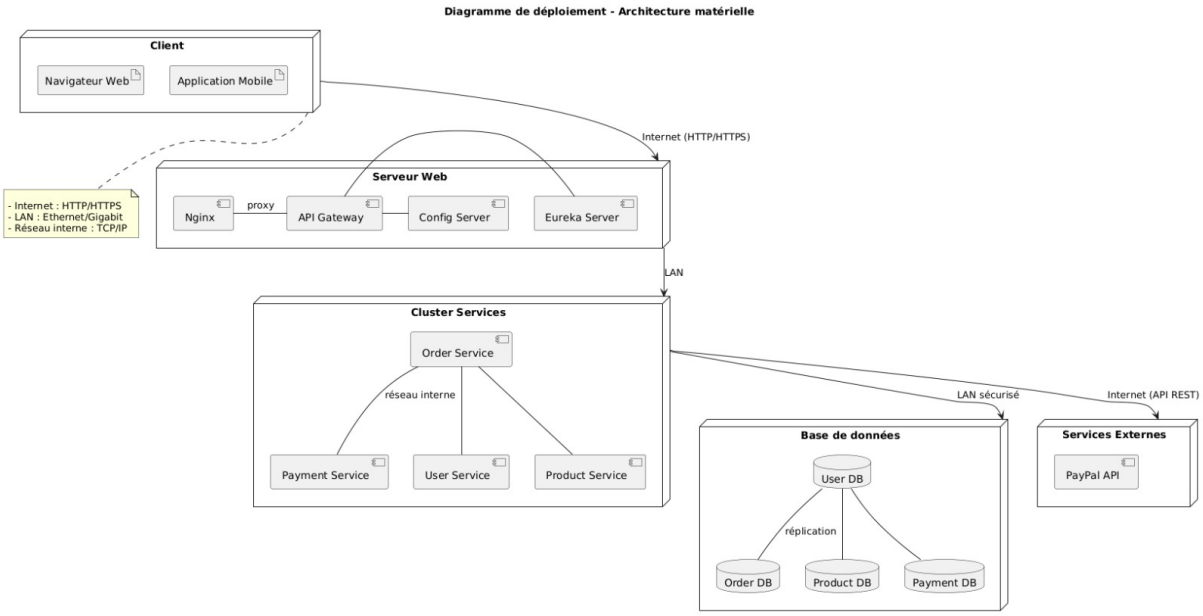

FIGURE 3.15 – Diagramme de déploiement : Architecture matérielle [2]

Description : Ce diagramme de déploiement montre la répartition physique des composants sur les serveurs [2]. Dans l'environnement de développement, tous les services peuvent tourner sur la même machine. En production, chaque microservice peut être déployé sur des serveurs distincts pour permettre le scaling horizontal indépendant [6].

3.5 Modèle de données

3.5.1 Modèle relationnel

Le modèle relationnel représente la traduction du diagramme de classes et du diagramme entité-association (ERD) en structures relationnelles exploitables par les bases de données [3]. Dans le cadre de cette plateforme e-commerce basée sur une architecture microservices, chaque microservice dispose de sa propre base de données, conformément au principe d'indépendance des données [2].

Les principales entités du système ont été transformées en tables relationnelles. Chaque table est définie par une clé primaire permettant d'identifier de manière unique chaque enregistrement. Les relations entre les entités sont assurées par des clés étrangères, garantissant l'intégrité référentielle [3].

Les tables principales du modèle relationnel sont les suivantes :

- **User** : stocke les informations des utilisateurs de la plateforme (clients et administrateurs), incluant les données d'authentification et les rôles.
- **Product** : contient les informations relatives aux produits du catalogue interne, telles que le nom, le prix, le stock et la catégorie.
- **Order** : représente les commandes passées par les clients, avec leur statut et le montant total.

- **OrderItem** : assure la relation entre les commandes et les produits, en précisant la quantité et le prix unitaire.
- **Payment** : enregistre les transactions de paiement associées aux commandes, notamment via PayPal [11].
- **ExternalProduct** : stocke temporairement les produits récupérés depuis des sources externes (Amazon [21]) dans le cadre de l'agrégation.

Les relations principales sont définies comme suit : un utilisateur peut effectuer plusieurs commandes (relation 1-N entre User et Order), une commande peut contenir plusieurs produits (relation 1-N entre Order et OrderItem), et chaque commande est associée à un paiement unique. Ce modèle relationnel permet une gestion cohérente et évolutive des données tout en respectant les contraintes de l'architecture microservices [2].

3.5.2 Dictionnaire de données

Les tableaux ci-dessous présentent le dictionnaire de données décrivant les principales tables du système [3]. Chaque colonne est définie par son type, ses contraintes et son rôle dans la base de données.

TABLE 3.1 – Dictionnaire de données – Tables principales

Nom de la colonne	**Type**	**Taille**	**Obligatoire**	**Valeur par défaut**	**Valeurs autorisées**	**PK**	**FK**
id	BIGINT	–	Oui	Auto	–	X	–
username	VARCHAR	100	Oui	–	Unique		
email	VARCHAR	255	Oui	–	Format email		
password	VARCHAR	255	Oui	–	–		
role	VARCHAR	20	Oui	CLIENT	CLIENT / ADMIN		
Table : User							
id	BIGINT	–	Oui	Auto	–	X	–
name	VARCHAR	150	Oui	–	–		
price	DECIMAL	10,2	Oui	0	0		
stock	INT	–	Oui	0	0		
category	VARCHAR	100	Oui	–	–		
Table : Product							
id	BIGINT	–	Oui	Auto	–	X	–
order_date	DATE	–	Oui	SYSDATE	–		
status	VARCHAR	30	Oui	PENDING	États commande		
total	DECIMAL	10,2	Oui	0	0		
user_id	BIGINT	–	Oui	–	–		X
Table : Order							
id	BIGINT	–	Oui	Auto	–	X	–
quantity	INT	–	Oui	1	1		
price	DECIMAL	10,2	Oui	–	0		
order_id	BIGINT	–	Oui	–	–		X
product_id	BIGINT	–	Oui	–	–		X
Table : OrderItem							
id	BIGINT	–	Oui	Auto	–	X	–

Nom de la colonne	Type	Taille	Obligatoire	Valeur par défaut	Valeurs autorisées	PK	FK
payment_date	DATE	–	Oui	SYSDATE	–		
status	VARCHAR	30	Oui	PENDING	États paiement		
amount	DECIMAL	10,2	Oui	0	0		
order_id	BIGINT	–	Oui	–	–		X
Table : Payment							

3.5.3 Diagramme de classe de la base de données (ERD)

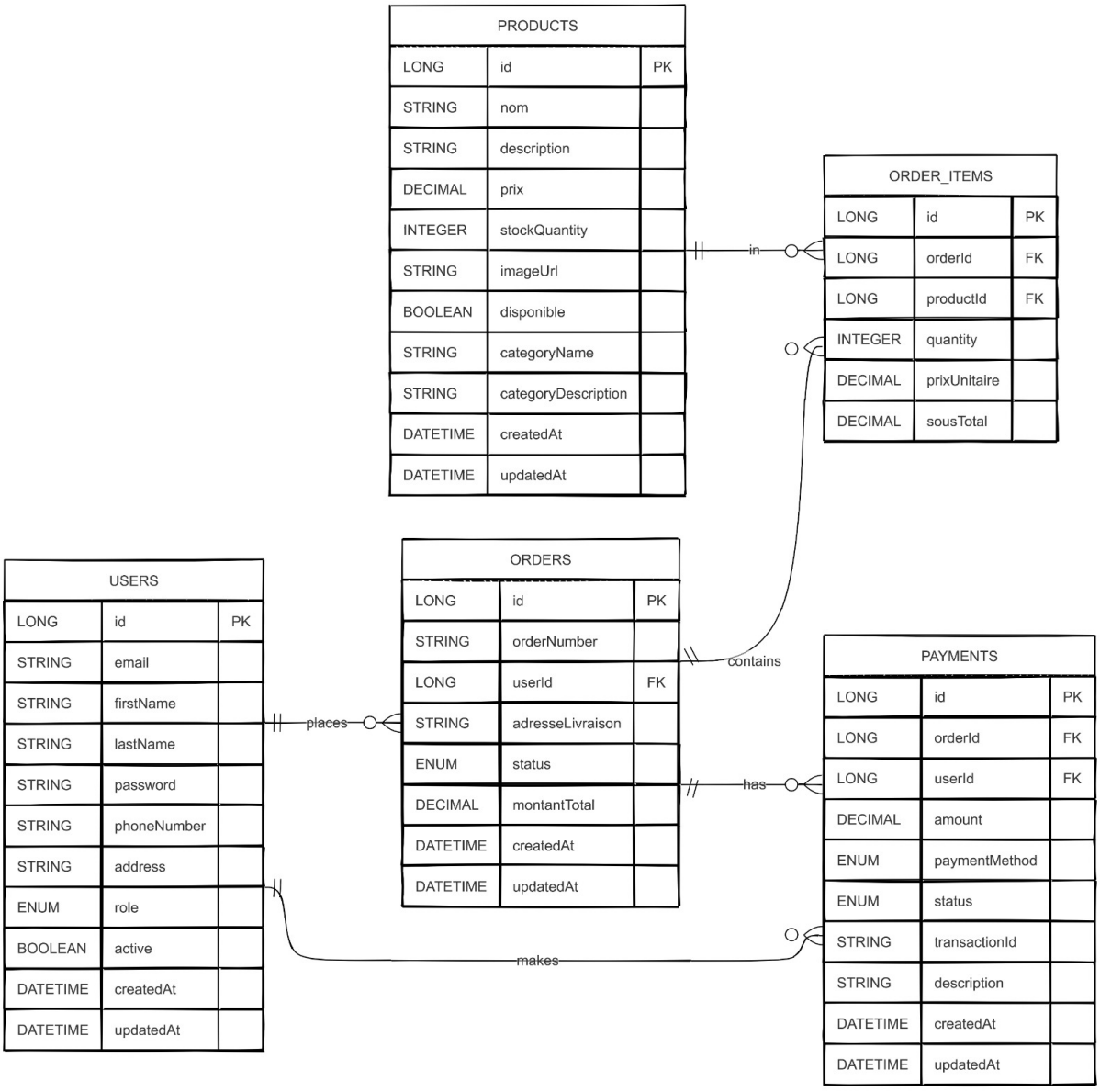

FIGURE 3.16 – Diagramme entité-association (ERD) : Modèle de données [3]

Description : Ce diagramme ERD présente le modèle conceptuel de données [3], avec les entités principales (User, Product, Order, OrderItem, Payment), leurs attributs et les relations entre elles (cardinalités, clés étran-

gères). Chaque microservice possède sa propre base de données H2, respectant ainsi le principe d'indépendance des données dans une architecture microservices [2].

3.6 Synthèse des diagrammes UML

Le tableau ci-dessous récapitule tous les diagrammes UML produits pour la conception du système [4] :

TABLE 3.2 – Récapitulatif des diagrammes UML produits

Type	**Nom**	**Fichier**	**Objectif principal**
Cas d'utilisation	Global	usecase-global.puml	Vue globale des fonctionnalités [20]
Statique	Classes simplifié	class-diagram-simplified.png	Structure des classes et contrôleurs [1]
Statique	Packages	package-diagram.puml	Organisation modulaire [20]
Dynamique	Communication	communication-checkout.puml	Interactions structurelles [11]
Dynamique	Activité	activity-*.puml	Flux de processus [7]
Dynamique	Flux de données	dataflow-order.png	Processus création commande [1]
Dynamique	États	state-*.puml	Cycles de vie des objets [8]
Architecture	Composants	component-diagram.png	Architecture modulaire [1]
Architecture	Couches	layered-architecture.png	Architecture en couches [2]
Architecture	Déploiement	deployment-architecture.jpg	Architecture matérielle [2]
Données	ERD	erd-database.puml	Modèle conceptuel de données [3]

3.7 Conclusion

Ce chapitre a permis de présenter la conception globale du système proposé à travers une modélisation complète et cohérente [4]. Nous avons commencé par la modélisation dynamique, en utilisant les diagrammes UML (diagrammes de séquences, d'activités et d'états) [20], afin de décrire les interactions entre les différents acteurs et composants du système ainsi que le déroulement des principaux scénarios fonctionnels. Cette approche a permis de mieux comprendre le comportement du système et d'anticiper les traitements à implémenter [1].

Nous avons ensuite abordé la modélisation statique à travers le diagramme de classes, le modèle relationnel et le dictionnaire de données [3], qui définissent la structure des données et les relations entre les différentes entités. Ces modèles assurent la cohérence, la pérennité et l'intégrité des données manipulées par l'application.

Enfin, l'architecture logicielle et matérielle a été présentée afin de justifier les choix techniques retenus, notamment l'adoption d'une architecture microservices favorisant la modularité, la scalabilité et la maintenabilité du système [2]. L'ensemble de ces éléments constitue une base solide pour la phase de réalisation, qui fera l'objet du chapitre suivant.

Chapitre 4

Réalisation du système

4.1 Introduction

Ce chapitre a pour objectif de présenter la solution logicielle et l'environnement de développement qui sont utilisés afin d'aboutir à développer l'application [5, 6].

4.2 Environnement de développement

4.2.1 Environnement matériel

Le développement de la plateforme a été réalisé sur des ordinateurs personnels disposant de ressources matérielles suffisantes pour supporter l'exécution simultanée de plusieurs microservices [2]. Les machines utilisées sont équipées de processeurs performants, d'une mémoire vive permettant l'exécution fluide des services Spring Boot [5] et d'un espace de stockage adéquat pour les outils de développement, les dépendances et la documentation du projet. Cette configuration a permis d'assurer un développement stable ainsi que des tests efficaces de l'architecture distribuée.

4.2.2 Environnement logiciel

Ce sont les outils logiciels utilisés pour le développement de l'application ou de la base de données, la modélisation des différents diagrammes de conception, etc. [4]

Backend

La réalisation de cette plateforme e-commerce repose sur un ensemble de technologies modernes adaptées aux architectures microservices [2] :

- **Java 17** (LTS) :

FIGURE 4.1 – Java 17 (LTS) – Environnement de développement backend

Langage de programmation principal avec typage fort, performance éprouvée et écosystème mature.

- **Spring Boot 3.4.1** :

FIGURE 4.2 – Structure générale d'un microservice Spring Boot [5]

Framework principal offrant auto-configuration, serveur embarqué (Tomcat) et fonctionnalités production-ready [5]. Starters utilisés : Web, Data JPA, Actuator.

- **Spring Cloud 2024.0.0** :

FIGURE 4.3 – Architecture Spring Cloud de la plateforme e-commerce microservices [6]

Écosystème pour microservices incluant [6] :

- Eureka Server pour la découverte des services [12]
- Spring Cloud Gateway pour le routage et l'API Gateway
- Spring Cloud Config pour la configuration centralisée
- OpenFeign pour la communication inter-services déclarative [13]

- **Persistance** : Chaque microservice dispose de sa propre base de données H2 (in-memory) avec Spring Data JPA et Hibernate pour l'accès aux données.
- **Communication et Résilience** :
  - OpenFeign pour les clients REST déclaratifs avec intégration automatique Eureka et load balancing client-side [13]
  - Resilience4j pour les circuit breakers, fallback methods automatiques, retry policies et rate limiting [17]
- **Intégrations Externes** :
  - PayPal REST API SDK 1.14.0 pour les paiements (sandbox et production) [11]
  - Amazon Product API pour l'agrégation de produits externes (en développement) [21]
- **Outils et Bibliothèques** :
  - Lombok pour la réduction du code boilerplate
  - MapStruct pour le mapping automatique Entity DTO
  - Bean Validation pour la validation déclarative des données
  - Spring Boot Actuator pour le monitoring (health checks, métriques) [5]
- **Build et Gestion** : Maven (projet multi-modules) avec structure parent-enfant pour la gestion centralisée des dépendances.

Frontend

- **React.js** :

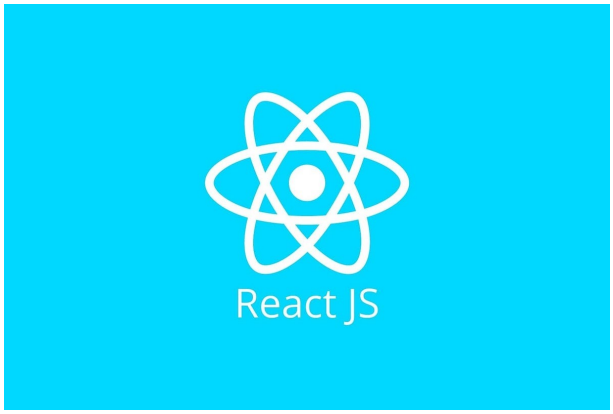

FIGURE 4.4 – Interface utilisateur développée avec React.js [16]

est une bibliothèque JavaScript open source développée par Facebook, utilisée pour construire des interfaces utilisateur interactives et réactives à partir de composants réutilisables [16].

- **React 19.2.3** : Framework JavaScript moderne pour les composants réutilisables et la gestion d'état efficace [16].
- **React Router DOM 7.10.1** : Navigation entre pages avec routes dynamiques et gestion de l'historique.
- **Axios 1.13.2** : Client HTTP pour les appels API avec configuration centralisée et gestion des erreurs.
- **React Scripts 5.0.1** : Configuration webpack intégrée avec hot reload en développement et build optimisé pour production.

Outils de développement

- IDE : IntelliJ IDEA / Eclipse pour le développement Java
- Visual Studio Code pour le développement React [16]
- Git pour la gestion de versions
- Postman pour les tests d'API
- Maven pour la gestion des dépendances et le build
- npm pour la gestion des dépendances frontend

4.3 Architecture réalisée

4.3.1 Structure du projet

Le projet est organisé en plusieurs modules principaux [1] :

- **Backend Microservices** (`App/catalogue-service/`) :
  - config-server/ : Configuration centralisée (Port 8888) [6]

  - eureka-server/ : Service Discovery (Port 8761) [12]
  - api-gateway/ : API Gateway avec circuit breakers (Port 8080) [6]
  - user-service/ : Gestion utilisateurs (Port 8083)
  - product-service/ : Catalogue produits (Port 8081)
  - order-service/ : Gestion commandes (Port 8085)
  - payment-service/ : Paiements PayPal (Port 8084) [11]
- **Frontend React** (`frontend/`) :
  - Composants React pour l'interface utilisateur [16]
  - Services API pour la communication avec le backend
  - Routing et navigation
- **Documentation** (`Documentation-Projet/`) :
  - Cahier des charges
  - Documentation technique [4]
  - Guides d'utilisation
  - Diagrammes UML [20]

4.3.2 Communication inter-services

La communication entre les microservices est réalisée via OpenFeign, permettant une communication déclarative et simplifiée [13] :

- Order Service User Service : Validation utilisateur
- Order Service Product Service : Vérification stock et prix
- External Aggregator Product Service : Produits internes
- External Aggregator Amazon API : Produits externes [21]

4.3.3 Résilience et tolérance aux pannes

Les circuit breakers sont implémentés au niveau de l'API Gateway avec Resilience4j, permettant [17] :

- Protection contre les cascades de pannes
- Fallback automatique avec messages d'erreur appropriés
- Health checks via Spring Boot Actuator [5]

4.4 Principales interfaces graphiques

La plateforme e-commerce propose plusieurs interfaces graphiques destinées aux utilisateurs finaux et aux administrateurs [16] :

4.4.1 Page d'accueil (Dashboard)

La page d'accueil permet aux utilisateurs d'accéder à une vue globale des produits disponibles, des statistiques et d'effectuer des recherches. Elle offre une navigation claire vers les différentes sections de la plateforme.

4.4.2 Catalogue de produits

Le catalogue de produits affiche les articles issus du catalogue interne avec leurs catégories, prix, disponibilité et images. En cours de développement, l'affichage inclura également les produits provenant de sources externes comme Amazon [21], offrant ainsi une vue unifiée.

4.4.3 Gestion du panier et commandes

L'interface de panier permet aux utilisateurs de gérer les produits sélectionnés avant la validation de la commande. L'interface de commandes permet de consulter l'historique et le statut des commandes passées.

4.4.4 Interface de paiement

L'interface de paiement assure le règlement sécurisé des commandes via PayPal [11]. Elle guide l'utilisateur à travers le processus de création, approbation et exécution du paiement.

4.4.5 Tableau de bord administrateur

Un tableau de bord administrateur est mis à disposition pour gérer les utilisateurs, les produits et les commandes. Il permet également de consulter les statistiques et de suivre l'activité de la plateforme.

4.4.6 Chatbot intelligent

Un chatbot intelligent (en développement) permet aux utilisateurs d'interagir naturellement avec le système pour rechercher des produits, obtenir des recommandations et obtenir de l'aide [15]. L'interface du chatbot est déjà implémentée et prête pour l'intégration avec le service de recommandation IA.

Ces interfaces ont été conçues pour offrir une navigation simple, claire et ergonomique, améliorant ainsi l'expérience utilisateur [16]. Le design est responsive et s'adapte aux différentes tailles d'écran.

4.5 Fonctionnalités implémentées

4.5.1 Services core complétés

- Configuration centralisée via Config Server [6]
- Service Discovery avec Eureka Server [12]
- API Gateway avec routage et circuit breakers [6]
- Gestion complète des utilisateurs avec rôles (CLIENT, ADMIN)

- Catalogue produits avec catégories et gestion du stock
- Gestion des commandes avec validation automatique et mise à jour du stock
- Intégration PayPal complète (Create → Approve → Execute) [11]
- Communication inter-services via OpenFeign [13]
- Résilience avec circuit breakers et fallback [17]

4.5.2 Services en développement

- External Aggregator Service pour l'agrégation de produits Amazon [21]
- AI Recommendation Service pour les recommandations intelligentes [15]
- Intégration complète du chatbot avec le service de recommandation [15]

4.6 Endpoints API principaux

4.6.1 User Service

- `GET /api/users` : Liste des utilisateurs
- `GET /api/users/{id}` : Détails utilisateur
- `POST /api/users` : Créer utilisateur
- `PUT /api/users/{id}` : Modifier utilisateur
- `DELETE /api/users/{id}` : Supprimer utilisateur

4.6.2 Product Service

- `GET /api/products` : Liste des produits
- `GET /api/products/{id}` : Détails produit
- `GET /api/products/category/{name}` : Produits par catégorie
- `POST /api/products` : Créer produit
- `PUT /api/products/{id}` : Modifier produit
- `PATCH /api/products/{id}/stock` : Mettre à jour stock

4.6.3 Order Service

- `GET /api/orders` : Liste des commandes
- `GET /api/orders/{id}` : Détails commande
- `GET /api/orders/user/{userId}` : Commandes par utilisateur
- `POST /api/orders` : Créer commande

- `PATCH /api/orders/{id}/cancel` : Annuler commande

4.6.4 Payment Service

- `POST /api/payments/paypal/create` : Créer paiement PayPal [11]
- `POST /api/payments/paypal/execute` : Exécuter paiement [11]
- `GET /api/payments/{id}` : Détails paiement

4.7 Intelligence Artificielle

4.7.1 Intégration d'OpenAI GPT-4

OpenAI GPT-4 est un modèle de langage avancé développé par OpenAI, capable de comprendre et générer du texte en langage naturel [15]. Il permet d'améliorer les recommandations intelligentes et d'enrichir l'expérience utilisateur via le chatbot de la plateforme.

FIGURE 4.5 – OpenAI GPT-4 pour l'intelligence artificielle et les recommandations [15]

4.7.2 Chatbot intelligent

Un chatbot intelligent a été développé dans le frontend React avec une interface utilisateur complète et moderne [16]. Le chatbot est prêt pour l'intégration avec le AI Recommendation Service qui fournira les réponses intelligentes basées sur les requêtes des utilisateurs .

4.7.3 AI Recommendation Service (En développement)

Le service de recommandation IA est en cours de développement et permettra :

- L'analyse intelligente des requêtes utilisateur en langage naturel
- La compréhension de l'intention de recherche
- La fourniture de recommandations personnalisées

• L'agrégation multi-sources avec ranking intelligent

Dans la version actuelle, une approche simplifiée est utilisée avec analyse basique par regex et keywords. L'intégration OpenAI GPT-4 est prévue pour une évolution future [15].

4.8 Conclusion

Ce chapitre a permis de présenter la phase de réalisation de la plateforme e-commerce, en mettant en évidence l'environnement de développement adopté [5, 6], l'architecture microservices réalisée [2] et les principales interfaces graphiques développées [16]. La solution obtenue repose sur une architecture microservices fonctionnelle et démontre la maîtrise des technologies utilisées. Elle constitue une base solide pour des évolutions futures, notamment l'extension du système d'agrégation [21], l'amélioration des fonctionnalités d'intelligence artificielle [15] et le déploiement du système dans un environnement de production réel.

Conclusion générale

Ce projet de fin d'année nous a permis de concevoir et de réaliser une plateforme e-commerce moderne reposant sur une architecture microservices distribuée [2, 1]. À travers ce travail, nous avons pu mettre en pratique les connaissances théoriques et techniques acquises au cours de notre formation en Ingénierie Informatique et Réseaux, notamment en matière de conception logicielle [4], de développement d'applications distribuées [6], d'intégration de services externes [11, 21] et d'intelligence artificielle [15]. L'étude des besoins et la phase de conception ont permis de définir une architecture claire, modulaire et évolutive basée sur 7-8 microservices indépendants [2]. La réalisation a abouti à une solution fonctionnelle intégrant plusieurs microservices indépendants, une communication inter-services efficace via OpenFeign [13], un mécanisme de tolérance aux pannes avec circuit breakers [17] ainsi que des services de paiement en ligne sécurisés via PayPal [11]. Le système développé répond aux exigences fonctionnelles et non fonctionnelles définies dans le cahier des charges. Les principaux résultats obtenus incluent une architecture microservices complète avec séparation claire des responsabilités [2], une communication inter-services robuste et déclarative via OpenFeign [13], une résilience garantie grâce aux circuit breakers et mécanismes de fallback [17], une intégration réussie avec PayPal pour les paiements en ligne [11], un frontend moderne et responsive développé avec React [16], et une base solide pour l'intégration de l'intelligence artificielle [15]. Au-delà des aspects techniques, ce projet nous a permis de développer des compétences en organisation, en travail d'équipe et en gestion de projet. Il constitue une base solide pouvant être enrichie par l'ajout de nouvelles fonctionnalités, telles que l'optimisation du système d'agrégation avec intégration complète d'Amazon [21], l'amélioration des mécanismes de recommandation avec OpenAI GPT-4 [15], le déploiement du système dans un environnement de production réel avec Docker et Kubernetes, ou encore l'ajout de fonctionnalités avancées d'intelligence artificielle comme le RAG (Retrieval-Augmented Generation) [15]. En conclusion, ce projet représente une expérience enrichissante et formatrice, illustrant l'importance des architectures modernes dans le développement des applications informatiques actuelles et ouvrant des perspectives intéressantes pour de futures évolutions. Il démontre notre maîtrise des technologies de pointe en développement logiciel et notre capacité à concevoir et réaliser des systèmes complexes et évolutifs.

Bibliographie et Nétographie

Bibliographie

[1] FOWLER, Martin. « Patterns of Enterprise Application Architecture », Addison-Wesley, 2002, 533 pages.

[2] NEWMAN, Sam. « Building Microservices : Designing Fine-Grained Systems », O'Reilly Media, 2015, 280 pages.

[3] REEVES, Hubert. « Bases de données relationnelles », Paris, Editions du seuil, 1988, p88.

[4] ROQUES, Pascal. « UML 2 par la pratique : Etudes de cas et exercices corrigés », 5ème Edition, 2006, p54.

[5] SPRING TEAM. « Spring Boot Reference Documentation », Version 3.4.1, 2024, 1 tome.

[6] SPRING TEAM. « Spring Cloud Documentation », Version 2024.0.0, 2024, 1 tome.

Nétographie

[7] `http://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-activites` : Diagrammes d'activités UML.

[8] `http://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-etats-transitions` : Diagrammes d'états-transitions en UML.

[9] `http://tvaira.free.fr/dev/uml/UML-DiagrammesDeSequence.pdf` : Fondements des diagrammes de séquence UML.

[10] `http://uml.free.fr/cours/p18.html` : Cours sur les diagrammes de paquetages UML.

[11] `https://developer.paypal.com` : Documentation développeur PayPal.

[12] `https://github.com/Netflix/eureka` : Documentation Eureka Service Discovery.

[13] `https://github.com/OpenFeign/feign` : Documentation OpenFeign.

[14] `https://martinfowler.com/articles/microservices.html` : Article sur les microservices.

[15] `https://platform.openai.com/docs` : Documentation OpenAI GPT-4.

[16] `https://react.dev` : Documentation officielle React.

[17] `https://resilience4j.readme.io` : Documentation Resilience4j.

[18] `https://spring.io/projects/spring-boot` : Documentation officielle Spring Boot.

[19] `https://spring.io/projects/spring-cloud` : Documentation officielle Spring Cloud.

[20] https://uml-diagrams.org : Référence complète sur UML.

[21] https://webservices.amazon.com/paapi5/documentation : Documentation Amazon Product Advertising API.

Annexe A

Interfaces graphiques secondaires

Cette annexe présente des captures d'écran supplémentaires des interfaces graphiques développées pour la plateforme e-commerce .

A.1 Interface principale de la plateforme MarketPlace

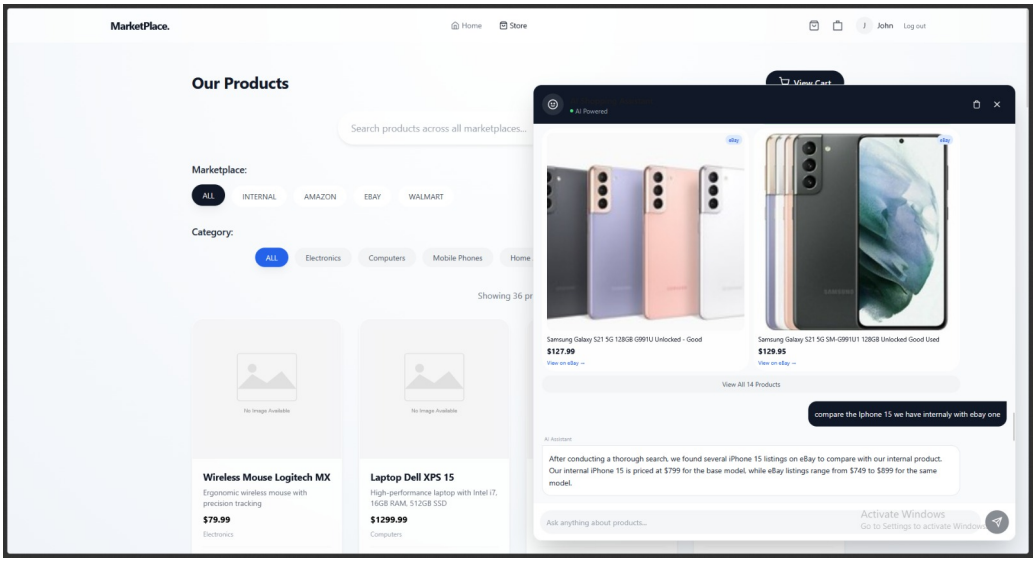

FIGURE A.1 – Interface principale MarketPlace - Catalogue produits et chatbot IA intégrés

Cette interface présente la page d'accueil complète de la plateforme e-commerce, intégrant à la fois le catalogue de produits et le chatbot intelligent dans une vue unifiée.

A.2 Interface de gestion des produits

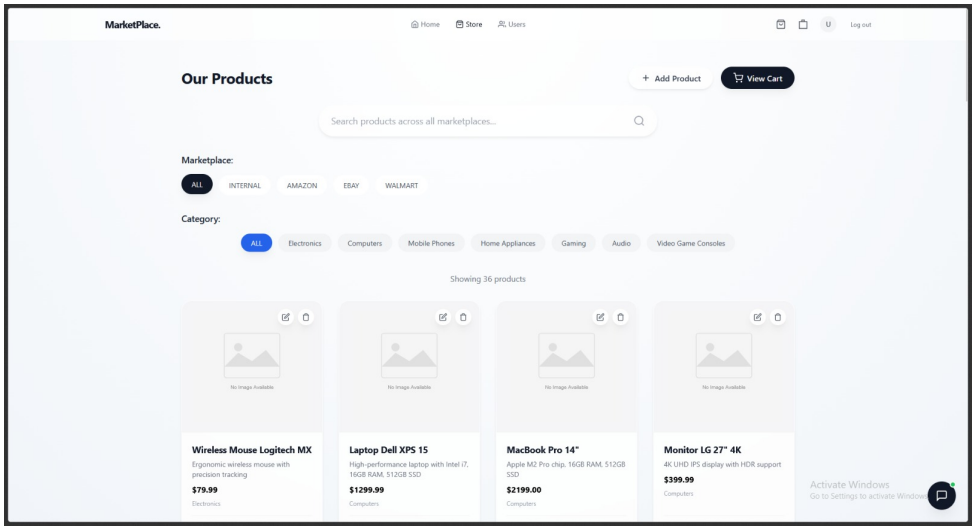

FIGURE A.2 – Interface principale de la plateforme MarketPlace

L'interface de gestion des produits permet aux administrateurs de créer, modifier et supprimer des produits du catalogue.

A.3 Interface de gestion des utilisateurs

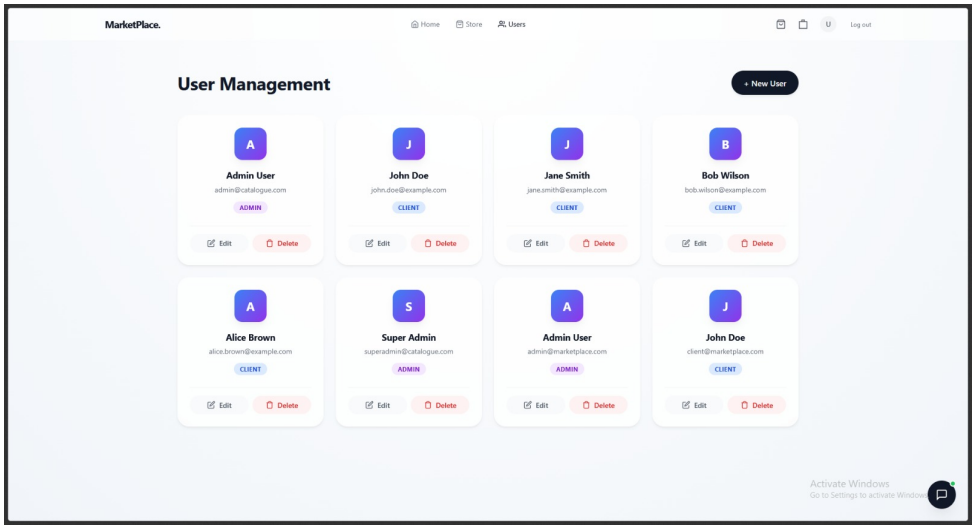

FIGURE A.3 – Interface principale de la plateforme MarketPlace

L'interface de gestion des utilisateurs permet aux administrateurs de consulter et gérer les comptes utilisateurs.

A.4 Interface du chatbot

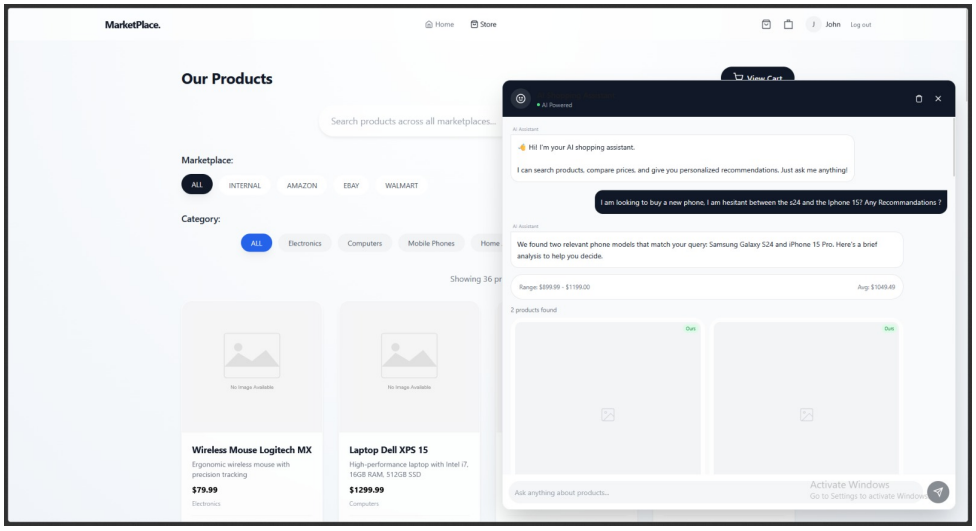

FIGURE A.4 – Interface principale de la plateforme MarketPlace

L'interface du chatbot intelligent permet aux utilisateurs d'interagir avec le système pour rechercher des produits et obtenir des recommandations .

A.5 Interface de paiement et validation de commande

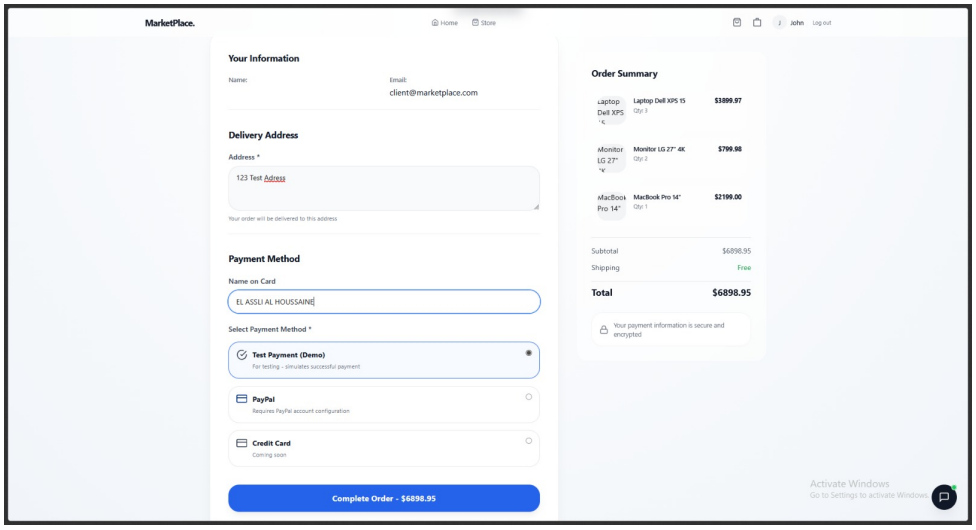

FIGURE A.5 – Interface principale de la plateforme MarketPlace

Cette interface présente le processus complet de finalisation d'une commande.

A.6 Interface du chatbot intelligentavec recommandations IA

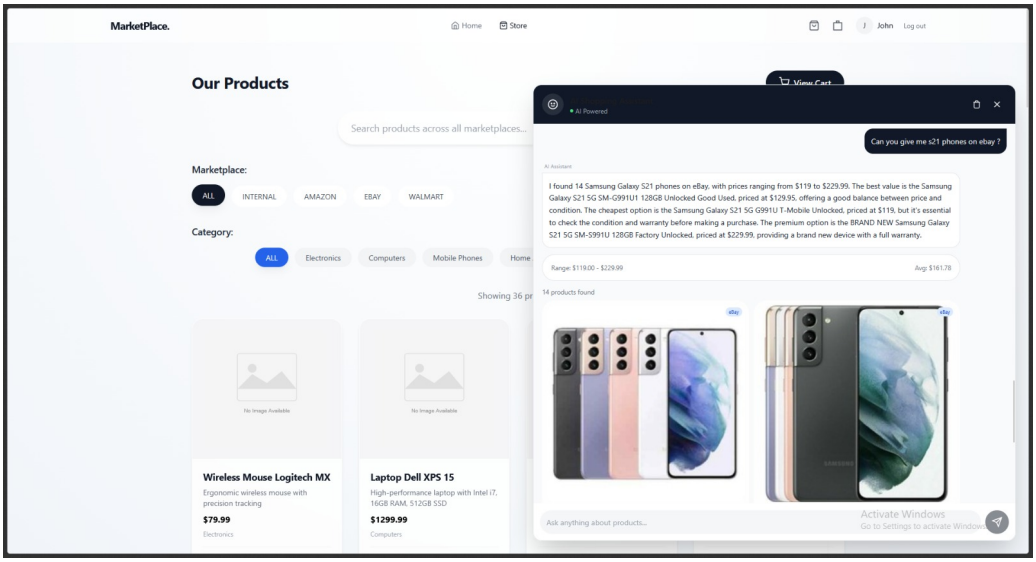

FIGURE A.6 – Interface du chatbot IA - Recommandations intelligentes et comparaison de produits

Cette interface démontre les capacités avancées du chatbot intelligent intégré à la plateforme. Le système utilise l'intelligence artificielle pour fournir des recommandations personnalisées et des analyses détaillées de produits.