



Université Abdelmalek Essaadi  
**FACULTÉ POLYDISCIPLINAIRE  
À LARACHE**



**Département Informatique  
Master DevOps et Cloud Computing**

**Rapport de Projet  
Application e-commerce (mini)**

**Réalisé par**  
Hajar El Bachiri  
Fatima Zohra Boutefah Khatib  
Anass Eraoui  
Khalid Lazrag

**Encadré par**  
Pr. Mohamed EL Mahjouby

**Année universitaire 2025–2026**

# Dédicaces

Au nom d'Allah, le Clément, le Miséricordieux.

Nous dédions humblement ce travail à toutes les personnes qui nous ont soutenus et encouragés tout au long de la réalisation de ce mini-projet.

À nos chères familles, pour leur soutien constant, leurs encouragements et leur confiance. Leur appui moral a été une source de motivation essentielle dans notre parcours universitaire.

À notre encadrant, pour son accompagnement, ses conseils avisés et sa disponibilité durant la réalisation de ce projet. Son encadrement nous a permis de progresser et d'approfondir nos connaissances.

À nos camarades et amis, avec qui nous partageons cette expérience universitaire, pour leur esprit de collaboration et leur soutien.

Merci à tous.

# **Remerciements**

Nous tenons à exprimer nos sincères remerciements à notre encadrant, pour son accompagnement, sa disponibilité et ses conseils pertinents tout au long de la réalisation de ce mini-projet. Son encadrement nous a permis de mieux comprendre les concepts abordés dans le cadre de ce module et d'améliorer nos compétences techniques.

Nous remercions également l'ensemble des responsables du Master DevOps et Cloud Computing pour la qualité de la formation dispensée, ainsi que pour l'environnement pédagogique favorable à l'apprentissage et au développement de nos compétences.

Enfin, nous adressons nos remerciements à toutes les personnes ayant contribué, de près ou de loin, à la réalisation de ce travail.

# Résumé

Ce rapport présente le travail réalisé dans le cadre du mini-projet du module « Applications Distribuées » au sein de la Faculté Polydisciplinaire à Larache.

Dans un contexte où le commerce électronique connaît une croissance rapide, les plateformes en ligne permettent aux utilisateurs d'acheter des produits de manière simple, rapide et sécurisée. Ces applications reposent sur des architectures distribuées assurant la gestion efficace des données, la communication entre les différentes couches du système et la sécurisation des transactions.

L'objectif principal de ce projet est de concevoir et développer une application e-commerce permettant la gestion des produits, des utilisateurs et des commandes, tout en mettant en œuvre une architecture backend structurée et sécurisée.

Pour ce faire, nous avons utilisé Spring Boot et Spring MVC pour le développement du backend, Thymeleaf pour la couche de présentation, ainsi qu'une base de données MySQL pour la persistance des données. La sécurisation de l'application a été assurée par l'utilisation de JSON Web Token (JWT).

**Mots clés :** application web, e-commerce, applications distribuées, Spring Boot, Thymeleaf, MySQL, JWT.

# Liste des abréviations

---

Abréviation	Signification
API	Application Programming Interface (Interface de Programmation d'Application)
CSS	Cascading Style Sheets (Feuilles de Style en Cascade)
CRUD	Create, Read, Update, Delete (Créer, Lire, Mettre à jour, Supprimer)
HTML	HyperText Markup Language (Langage de Balises Hypertexte)
HTTP	HyperText Transfer Protocol (Protocole de Transfert Hypertexte)
JPA	Java Persistence API (API de Persistance Java)
JWT	JSON Web Token (Jeton Web JSON)
MVC	Model-View-Controller (Modèle-Vue-Contrôleur)
MySQL	Structured Query Language – Système de base de données relationnelle
ORM	Object-Relational Mapping (Mapping Objet-Relationnel)
REST	Representational State Transfer
SGBD	Système de Gestion de Base de Données
SQL	Structured Query Language (Langage de Requête Structuré)
UML	Unified Modeling Language (Langage de Modélisation Unifié)

---

# Table des matières

Dédicaces	i
Remerciements	ii
Résumé	iii
Liste des abréviations	iv
Introduction générale	1
Contexte général du projet	3
1.1 Introduction . . . . .	4
1.2 Présentation du projet . . . . .	4
1.3 Contexte du projet . . . . .	4
1.4 Problématique . . . . .	5
1.5 Objectifs du projet . . . . .	5
1.6 Exigences fonctionnelles . . . . .	5
1.7 Exigences non fonctionnelles . . . . .	6
1.8 Étude de l'existant . . . . .	7
1.9 Solution proposée . . . . .	7
1.10 Fonctionnalités principales . . . . .	7
1.11 Conclusion . . . . .	8
Déroulement du projet	9
2.1 Introduction . . . . .	10
2.2 Planification opérationnelle . . . . .	10
2.2.1 Diagramme de Gantt . . . . .	11
2.3 Phases du développement . . . . .	11
2.4 Conclusion . . . . .	13
Spécification et modélisation	14
3.1 Introduction . . . . .	15
3.2 Identification des acteurs . . . . .	15
3.3 Diagrammes de cas d'utilisation . . . . .	15

3.3.1	Authentification . . . . .	15
3.3.2	Navigation et Produits . . . . .	16
3.3.3	Panier et Commandes . . . . .	17
3.3.4	Administration . . . . .	17
3.4	Modélisation des données . . . . .	18
3.5	Diagrammes de séquence . . . . .	19
3.5.1	Validation JWT . . . . .	20
3.5.2	Traitement des commandes . . . . .	20
3.6	Conclusion . . . . .	21
<b>Implémentation et réalisation</b>		<b>23</b>
4.1	Introduction . . . . .	24
4.2	Environnement logiciel . . . . .	24
4.2.1	Java Development Kit (JDK) . . . . .	24
4.2.2	IntelliJ IDEA . . . . .	25
4.2.3	Git . . . . .	25
4.2.4	GitHub . . . . .	26
4.2.5	Railway (Plateforme Cloud) . . . . .	26
4.3	Technologies Backend . . . . .	28
4.3.1	Langage Java . . . . .	28
4.3.2	Spring Boot . . . . .	29
4.3.3	Spring MVC . . . . .	29
4.3.4	Spring Data JPA . . . . .	30
4.3.5	Hibernate . . . . .	31
4.3.6	Apache Maven . . . . .	31
4.3.7	Project Lombok . . . . .	32
4.4	Technologies Frontend . . . . .	32
4.4.1	HTML . . . . .	32
4.4.2	CSS . . . . .	33
4.4.3	JavaScript . . . . .	33
4.4.4	Bootstrap . . . . .	33
4.4.5	Thymeleaf . . . . .	34
4.4.6	Chart.js . . . . .	34
4.5	Base de données . . . . .	35
4.5.1	MySQL . . . . .	35

4.5.2	Configuration de la source de données . . . . .	35
4.5.3	Mapping Objet-Relationnel (ORM) . . . . .	36
4.6	Sécurité de l'application . . . . .	36
4.6.1	Concepts fondamentaux . . . . .	36
4.6.2	Spring Security . . . . .	37
4.6.3	JSON Web Token (JWT) . . . . .	37
4.6.4	BCrypt et protection des mots de passe . . . . .	38
4.6.5	Gestion des rôles et autorisations . . . . .	38
4.6.6	Gestion des rôles et autorisations . . . . .	39
4.7	Architecture logicielle du projet . . . . .	39
4.7.1	Architecture globale de l'application . . . . .	39
4.7.2	Organisation du projet dans l'environnement de développement	42
4.8	Présentation des fonctionnalités implémentées . . . . .	43
4.8.1	Interface d'accueil (Home) . . . . .	44
4.8.2	Interface d'inscription (Register) . . . . .	45
4.8.3	Interface d'authentification (Login) . . . . .	46
4.8.4	Consultation des produits . . . . .	46
4.8.5	Interface du panier(Cart) . . . . .	49
4.8.6	Interface de consultation des commandes . . . . .	50
4.8.7	Interface d'administration des produits . . . . .	51
4.8.8	Interface d'administration des commandes . . . . .	52
4.8.9	Interface d'administration des codes promotionnels . . . . .	53
4.8.10	Tableau de bord administrateur (Admin Dashboard) . . . . .	54
<b>Conclusion générale</b>		<b>55</b>

# Table des figures

2.1	Diagramme de Gantt du projet E-Commerce . . . . .	11
3.1	Diagramme de cas d'utilisation : Authentification . . . . .	16
3.2	Diagramme de cas d'utilisation : Navigation et Produits . . . . .	16
3.3	Diagramme de cas d'utilisation : Panier et Commandes . . . . .	17
3.4	Diagramme de cas d'utilisation : Administration . . . . .	18
3.5	Diagramme de classes . . . . .	19
3.6	Diagramme de séquence : Validation JWT . . . . .	20
3.7	Diagramme de séquence : Traitement des commandes . . . . .	21
4.1	logo de Java Development Kit (JDK) . . . . .	24
4.2	logo d'IntelliJ IDEA . . . . .	25
4.3	logo de Git . . . . .	25
4.4	logo de GitHub . . . . .	26
4.5	logo de Railway . . . . .	26
4.6	Tableau de bord des projets Railway . . . . .	27
4.7	Interface de gestion de la base MySQL sur Railway . . . . .	27
4.8	Interface de déploiement de l'application Spring Boot sur Railway . . . . .	28
4.9	logo de Java . . . . .	28
4.10	logo de Spring Boot . . . . .	29
4.11	logo de Spring MVC . . . . .	29
4.12	logo de Spring Data JPA . . . . .	30
4.13	logo de Hibernate . . . . .	31
4.14	logo d'Apache Maven . . . . .	31
4.15	logo de Project Lombok . . . . .	32
4.16	logo de HTML . . . . .	32
4.17	logo de CSS . . . . .	33
4.18	logo de JavaScript . . . . .	33
4.19	logo de Bootstrap . . . . .	33
4.20	logo de Thymeleaf . . . . .	34
4.21	logo de Chart.js . . . . .	34
4.22	logo de MySQL . . . . .	35
4.23	logo de Spring Security . . . . .	37
4.24	logo de JWT . . . . .	37

4.25 Architecture globale de l'application Spring Boot . . . . .	40
4.26 Architecture en couches du projet dans l'IDE . . . . .	43
4.27 Interface d'accueil (Home) . . . . .	44
4.28 Interface d'inscription (Register) . . . . .	45
4.29 Interface d'authentification (Login) . . . . .	46
4.30 Interface de consultation des produits . . . . .	47
4.31 Interface de consultation détaillée d'un produit . . . . .	48
4.32 Interface du panier (Cart) . . . . .	49
4.33 Interface de consultation des commandes . . . . .	50
4.34 Interface d'administration des produits . . . . .	51
4.35 Interface d'administration des commandes . . . . .	52
4.36 Interface d'administration des codes promotionnels . . . . .	53
4.37 Interface du tableau de bord administrateur . . . . .	54

# Introduction générale

L'évolution rapide des technologies numériques a profondément transformé les modes de consommation et les pratiques commerciales. Le commerce électronique, communément appelé e-commerce, s'est imposé comme un élément central de l'économie moderne. Il permet aux utilisateurs d'accéder à des services et à des produits à distance, tout en offrant aux entreprises de nouvelles opportunités de gestion, de visibilité et de croissance. Cette transformation repose en grande partie sur le développement d'applications informatiques distribuées capables de garantir performance, fiabilité et sécurité.

Les applications distribuées jouent aujourd'hui un rôle fondamental dans la conception des systèmes modernes. Elles permettent la séparation des responsabilités, la gestion efficace des données et la communication entre plusieurs composants logiciels. Dans le domaine du e-commerce, ces caractéristiques sont essentielles pour assurer la disponibilité des services, la protection des informations sensibles et une expérience utilisateur fluide.

Dans le cadre du module « Applications Distribuées », nous avons été amenés à concevoir et développer une mini-application e-commerce illustrant les principes étudiés durant le semestre. Ce projet vise à mettre en pratique les concepts théoriques liés aux architectures client-serveur, à la structuration d'une application web moderne, ainsi qu'à la sécurisation des échanges entre les différents composants du système.

L'application développée propose un ensemble de fonctionnalités essentielles à une plateforme de vente en ligne : gestion des produits, authentification des utilisateurs, manipulation du panier d'achat et traitement des commandes. Elle repose sur un backend construit avec Spring Boot et Spring MVC, une interface web utilisant Thymeleaf, ainsi qu'une base de données relationnelle MySQL. La sécurité est assurée par l'utilisation de JSON Web Tokens (JWT), garantissant un accès contrôlé aux ressources.

Dans le premier chapitre, nous présentons le cadre général du projet ainsi que les notions fondamentales liées aux applications distribuées et au commerce électronique. Nous y introduisons les concepts théoriques nécessaires à la compréhension de l'architecture adoptée, notamment le modèle client-serveur, la séparation des couches applicatives et les principes de communication entre les composants d'un système distribué.

Le deuxième chapitre est consacré à l'analyse et à la conception du système. Il décrit les besoins fonctionnels et non fonctionnels de l'application, les différents acteurs intervenant dans le système, ainsi que la modélisation UML comprenant les diagrammes de cas d'utilisation et le diagramme de classes. Cette phase permet de structurer l'application avant son implémentation et d'assurer une vision claire de son organisation

interne.

Le troisième chapitre détaille l'architecture technique et l'environnement de développement. Il présente les technologies utilisées, notamment Spring Boot, Spring MVC, Thymeleaf et MySQL, ainsi que le mécanisme d'authentification basé sur JWT. Ce chapitre met en évidence la structuration en couches (contrôleurs, services, accès aux données) et explique le rôle de chaque composant dans le fonctionnement global de l'application.

Enfin, le quatrième chapitre expose la phase d'implémentation et les principaux résultats obtenus. Il décrit les fonctionnalités réalisées, les interfaces développées et les interactions entre le frontend et le backend. Les difficultés rencontrées ainsi que les solutions adoptées y sont également présentées.

Ce rapport se termine par une conclusion générale qui synthétise les apports pédagogiques du projet et met en lumière les perspectives d'amélioration envisageables.

# CHAPITRE 1

---

---

Contexte général du projet

---

## 1.1 Introduction

Dans un contexte marqué par la digitalisation croissante des activités commerciales, les entreprises, qu'elles soient petites, moyennes ou grandes, ont désormais besoin de solutions informatiques leur permettant de gérer efficacement leurs opérations. La gestion des produits, des commandes, des utilisateurs et des transactions constitue aujourd'hui un élément central de la compétitivité.

Ce chapitre présente le cadre général de notre projet. Il expose le contexte, la problématique, les objectifs, ainsi qu'une étude de l'existant afin de justifier la solution proposée.

## 1.2 Présentation du projet

Le projet réalisé consiste en la conception et le développement d'un système de gestion e-commerce permettant d'administrer une plateforme de vente en ligne.

Un système e-commerce peut être défini comme : Une application web permettant la gestion des produits, des commandes, des utilisateurs et des transactions via Internet.

Selon International Business Machines (IBM), le commerce électronique désigne l'ensemble des transactions commerciales effectuées électroniquement via des réseaux numériques, principalement Internet [1].

Notre projet vise donc à développer une application web permettant :

- La gestion des produits
- La gestion des utilisateurs
- La gestion du panier
- Le traitement des commandes
- La gestion des codes promos

## 1.3 Contexte du projet

Le commerce électronique connaît une croissance mondiale continue. D'après Statista, le chiffre d'affaires mondial du e-commerce dépasse plusieurs milliers de milliards de dollars par an, avec une croissance soutenue.

Au Maroc également, le secteur connaît une expansion importante grâce à :

- La démocratisation d'Internet
- L'augmentation des paiements en ligne
- L'évolution des habitudes d'achat

Cependant, de nombreuses petites structures ne disposent pas de systèmes de gestion adaptés. Certaines utilisent :

- Des outils manuels
- Des solutions non intégrées
- Des plateformes complexes difficiles à personnaliser

Notre projet s'inscrit dans une démarche de conception d'un système e-commerce structuré, pédagogique et techniquement cohérent.

## 1.4 Problématique

Le développement d'un système de gestion e-commerce implique bien plus que la simple création d'une interface de vente en ligne. Il nécessite la mise en place d'une architecture cohérente permettant de gérer efficacement les produits, les catégories, les utilisateurs et les commandes, tout en garantissant la sécurité et la fiabilité des données.

De nombreuses plateformes existantes comme Shopify ou Magento proposent des solutions complètes, mais leur utilisation ne permet pas toujours de comprendre les mécanismes internes d'un système e-commerce. Dans un cadre académique, il est essentiel de concevoir et développer une solution personnalisée afin de maîtriser les concepts fondamentaux d'architecture logicielle et de gestion de base de données.

Ainsi, la problématique de ce projet consiste à concevoir un système de gestion e-commerce structuré, sécurisé et maintenable, capable d'administrer efficacement les opérations commerciales tout en respectant les bonnes pratiques du développement web.

## 1.5 Objectifs du projet

L'objectif principal de ce projet est de concevoir et développer un système de gestion e-commerce permettant d'administrer efficacement une plateforme de vente en ligne. Il s'agit de mettre en place une application web structurée, sécurisée et maintenable, capable de gérer les produits, les utilisateurs et les commandes. Ce projet vise également à appliquer les concepts étudiés durant la formation, notamment la conception orientée objet, la modélisation UML, la gestion des bases de données et l'architecture des applications web modernes.

## 1.6 Exigences fonctionnelles

Les exigences fonctionnelles définissent les services et les fonctionnalités que le système doit fournir aux utilisateurs. Elles décrivent précisément les tâches que l'application doit accomplir afin de répondre aux besoins identifiés.

**a. Authentification et gestion des utilisateurs**

- Les utilisateurs doivent pouvoir créer un compte et s'authentifier à l'aide d'identifiants sécurisés.
- Le système doit permettre la gestion des rôles (administrateur, utilisateur, etc.).
- L'administrateur doit pouvoir consulter, modifier ou supprimer les comptes utilisateurs.

**b. Gestion des produits**

- L'administrateur doit pouvoir ajouter un nouveau produit.
- Le système doit permettre la modification et la suppression des produits existants.
- Les produits doivent être affichés sous forme de liste organisée.

**c. Gestion des catégories**

- L'administrateur doit pouvoir créer, modifier et supprimer des catégories.
- Chaque produit doit être associé à une catégorie.
- Le système doit permettre l'affichage des produits par catégorie.

**d. Gestion des commandes**

- Les utilisateurs doivent pouvoir passer une commande.
- Le système doit enregistrer automatiquement les informations relatives à chaque commande.
- L'administrateur doit pouvoir consulter l'historique des commandes.

**e. Consultation et suivi des opérations**

- Le système doit permettre l'affichage des détails d'une commande.
- Les informations doivent être présentées de manière claire.
- Le système doit conserver un historique des opérations.

## 1.7 Exigences non fonctionnelles

Les exigences non fonctionnelles décrivent les qualités attendues du système. Elles ne concernent pas directement les fonctionnalités, mais plutôt les critères de performance, de sécurité, de fiabilité et de maintenabilité que l'application doit respecter.

**a. Sécurité**

Le système doit garantir la protection des données et sécuriser l'authentification des utilisateurs.

**b. Performance**

L'application doit répondre rapidement aux requêtes et gérer efficacement les interactions avec la base de données.

**c. Fiabilité**

Les données doivent être correctement enregistrées et rester cohérentes.

**d. Maintenabilité**

Le code doit être structuré, organisé et documenté afin de faciliter les évolutions futures.

**e. Évolutivité**

L'architecture doit permettre l'ajout de nouvelles fonctionnalités sans modification majeure du système existant.

**f. Utilisabilité**

L'interface doit être claire, intuitive et facile à utiliser.

## 1.8 Étude de l'existant

De nombreuses plateformes e-commerce existent aujourd'hui, allant des solutions complètes destinées aux grandes entreprises aux applications simplifiées pour les petites structures. Ces solutions offrent souvent des fonctionnalités avancées mais peuvent présenter une complexité élevée ou des contraintes techniques.

Dans un cadre pédagogique, développer une application dédiée permet de mieux comprendre les mécanismes internes des systèmes distribués sans dépendre d'outils préconfigurés.

## 1.9 Solution proposée

La solution développée repose sur une architecture web moderne intégrant un backend basé sur Spring Boot et Spring MVC, une interface utilisateur dynamique avec Thymeleaf, et une base de données MySQL.

L'application met en œuvre un système d'authentification sécurisé utilisant JWT, garantissant un contrôle d'accès aux différentes fonctionnalités.

## 1.10 Fonctionnalités principales

L'application propose les fonctionnalités suivantes :

- Gestion des produits
- Gestion des utilisateurs

- Gestion du panier
- Traitement des commandes
- Authentification sécurisée
- Consultation des données en temps réel

## 1.11 Conclusion

Ce premier chapitre a permis de présenter le cadre général du projet, son contexte, sa problématique et ses objectifs. L'étude de l'existant a montré l'importance des systèmes e-commerce modernes et a justifié le développement d'une solution personnalisée dans un cadre académique.

Le chapitre suivant décrira le déroulement du projet, la planification des tâches et le processus de développement adopté.

## **CHAPITRE 2**

---

---

Déroulement du projet

---

## 2.1 Introduction

Ce chapitre présente le déroulement du projet depuis sa planification initiale jusqu'à sa mise en œuvre finale. Il décrit l'organisation des tâches, la méthodologie adoptée et les différentes phases du développement. L'objectif est de montrer comment le projet a été structuré afin d'assurer une progression logique, maîtrisée et conforme aux objectifs pédagogiques du module « Applications Distribuées ».

## 2.2 Planification opérationnelle

Afin d'assurer une gestion rigoureuse du projet, une planification détaillée a été réalisée sous forme de diagramme de Gantt.

Ce diagramme permet de visualiser :

- Les différentes phases du projet
- La durée de chaque tâche
- L'enchaînement chronologique des activités
- La répartition des responsabilités entre les membres de l'équipe

Le projet s'est déroulé sur une période de trois semaines (Janvier – Février 2026), selon une organisation progressive et structurée.

## 2.2.1 Diagramme de Gantt

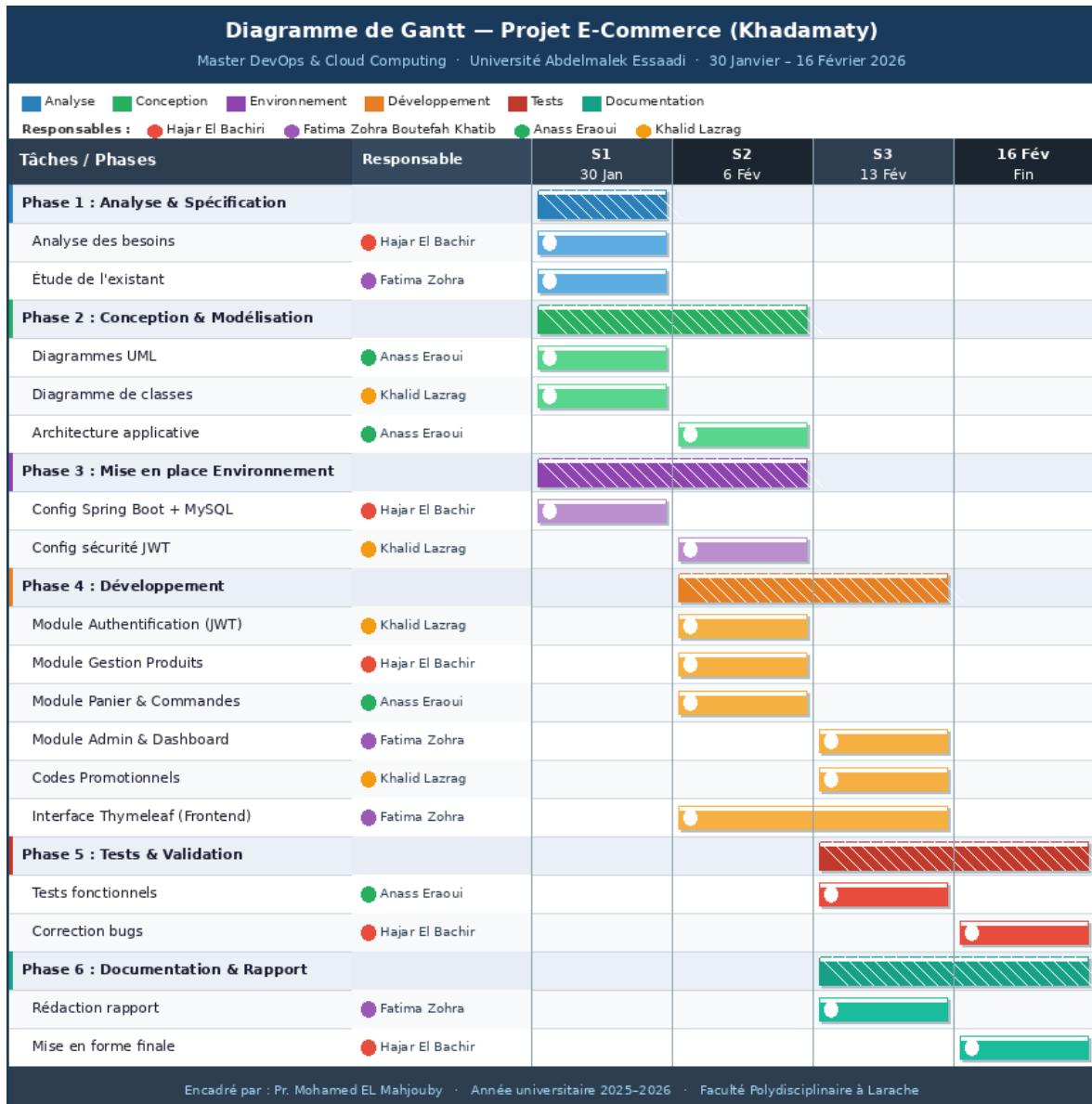


Figure 2.1: Diagramme de Gantt du projet E-Commerce

## 2.3 Phases du développement

Le projet a été structuré en six phases principales, conformément au diagramme de Gantt présenté précédemment.

### Phase 1 : Analyse et Spécification

Cette phase initiale a été consacrée à :

- L'analyse des besoins fonctionnels

- L'étude de l'existant
- La rédaction des spécifications

Elle a permis d'établir une base claire pour la conception du système.

## Phase 2 : Conception et Modélisation

Durant cette phase, les éléments suivants ont été réalisés :

- Diagrammes de cas d'utilisation
- Diagramme de classes
- Diagrammes de séquence
- Définition de l'architecture applicative

Cette étape a permis de structurer le système avant son implémentation.

## Phase 3 : Mise en place de l'environnement

Cette phase a consisté à :

- Configurer Spring Boot et MySQL
- Mettre en place la configuration de sécurité JWT
- Préparer l'environnement de développement

Elle a assuré une base technique stable pour le développement.

## Phase 4 : Développement

Le développement a été organisé en modules :

- Module d'authentification (JWT)
- Module de gestion des produits
- Module panier et commandes
- Module administration et tableau de bord
- Gestion des codes promotionnels
- Interface frontend avec Thymeleaf

Chaque module a été développé et testé progressivement.

## Phase 5 : Tests et Validation

Cette phase a inclus :

- Tests fonctionnels
- Correction des anomalies détectées

Elle a permis d'assurer la stabilité et la fiabilité de l'application.

## Phase 6 : Documentation et Rapport

La dernière phase a été consacrée à :

- La rédaction du rapport final
- La mise en forme et validation du document

Cette étape clôture officiellement le projet.

## 2.4 Conclusion

Ce chapitre a présenté le déroulement méthodologique du projet à travers une planification structurée et une organisation progressive des tâches.

Le diagramme de Gantt met en évidence une gestion rigoureuse du temps et une répartition claire des responsabilités.

L'approche incrémentale adoptée a permis de développer, tester et intégrer les différentes fonctionnalités de manière maîtrisée, tout en garantissant la cohérence architecturale du système.

Cette organisation méthodique constitue une base solide pour la phase de spécification et de modélisation détaillée dans le chapitre suivant.

# CHAPITRE 3

---

Spécification et modélisation

---

### 3.1 Introduction

Après l'analyse des besoins fonctionnels et non fonctionnels présentée dans le chapitre précédent, il est nécessaire de formaliser ces exigences à travers une étape de spécification et de modélisation.

La modélisation constitue une phase essentielle du cycle de développement logiciel. Elle permet de représenter le système sous forme de modèles abstraits facilitant la compréhension, la communication entre les parties prenantes et la préparation de l'implémentation.

### 3.2 Identification des acteurs

En ingénierie logicielle, un acteur est défini comme une entité externe interagissant avec le système afin d'atteindre un objectif spécifique . Un acteur peut être une personne, un système externe ou une organisation.

Dans notre application e-commerce, les acteurs identifiés sont :

- **Administrateur** : responsable de la gestion globale du système (utilisateurs, produits, commandes, configurations).
- **Gestionnaire** : supervise les ventes, le stock et le traitement des commandes.
- **Client** : consulte le catalogue, gère son panier et effectue des commandes.

Cette séparation garantit une gestion sécurisée et conforme aux responsabilités de chaque profil utilisateur.

### 3.3 Diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation est un diagramme UML permettant de représenter les interactions entre les acteurs et le système.

Selon l'OMG (Object Management Group), il décrit les fonctionnalités offertes par le système et les relations entre ces fonctionnalités et les acteurs .

Ces diagrammes permettent de clarifier les exigences fonctionnelles du système.

#### 3.3.1 Authentification

La figure suivante présente le cas d'utilisation pour l'authentification. Tous les acteurs doivent s'authentifier pour accéder à leurs fonctionnalités respectives.

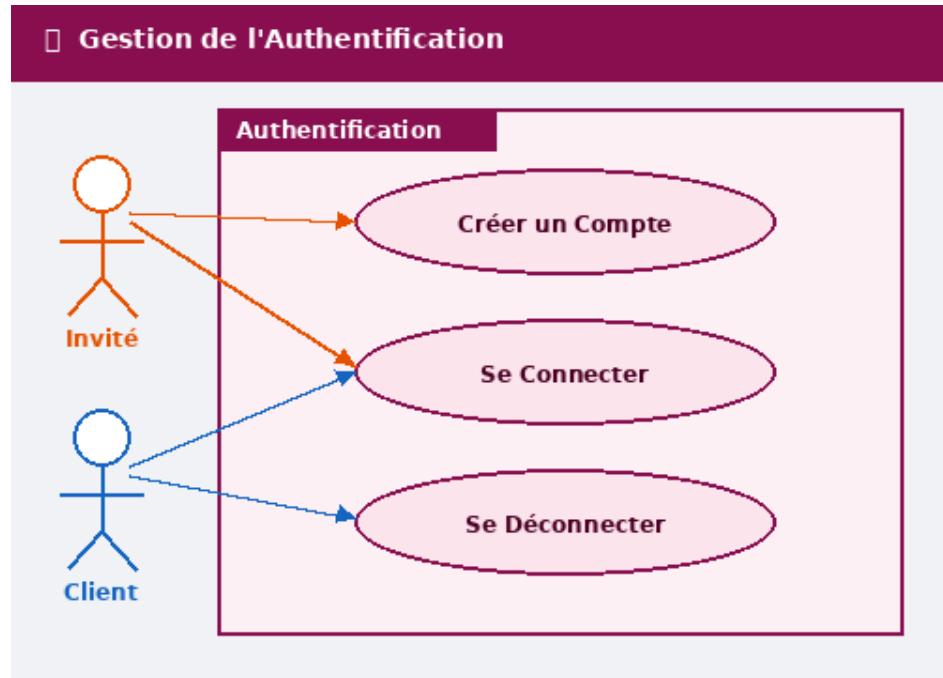


Figure 3.1: Diagramme de cas d'utilisation : Authentification

### 3.3.2 Navigation et Produits

Ce diagramme détaille les actions liées à la navigation dans le catalogue et la consultation des produits, principalement effectuées par le client.

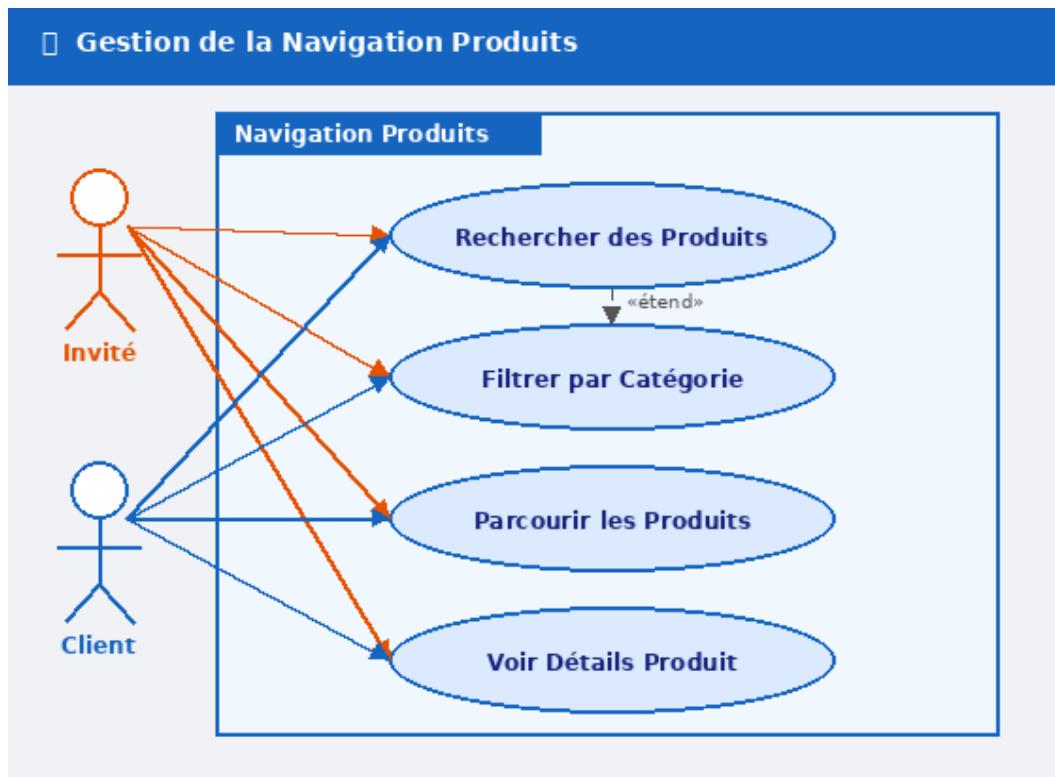


Figure 3.2: Diagramme de cas d'utilisation : Navigation et Produits

### 3.3.3 Panier et Commandes

La gestion du panier et le processus de commande sont au cœur de l'expérience client. Ce diagramme illustre les étapes de la sélection des articles jusqu'à la validation de la commande.

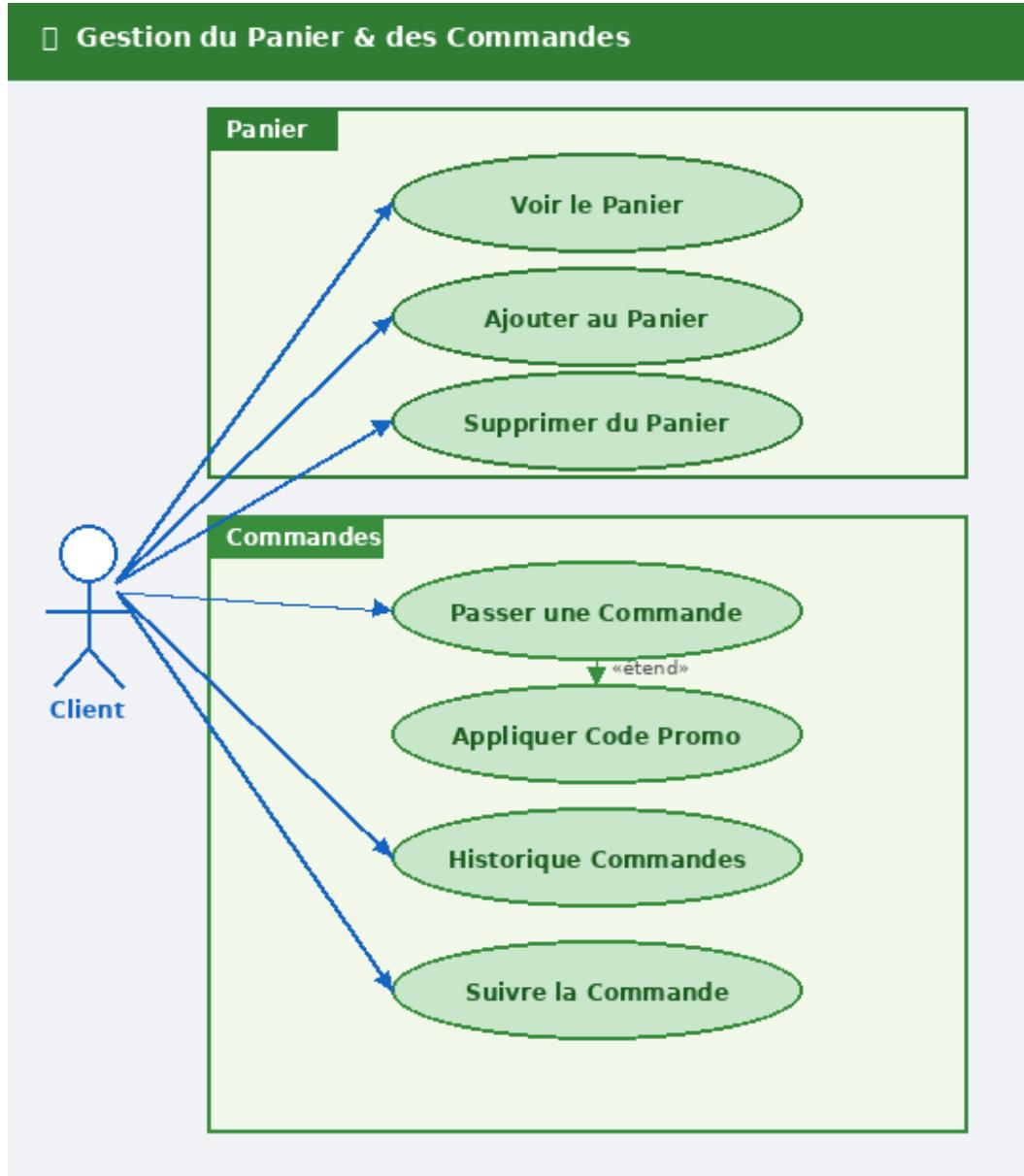


Figure 3.3: Diagramme de cas d'utilisation : Panier et Commandes

### 3.3.4 Administration

L'administration du système couvre la gestion globale, incluant la gestion des utilisateurs, des produits et des configurations système.

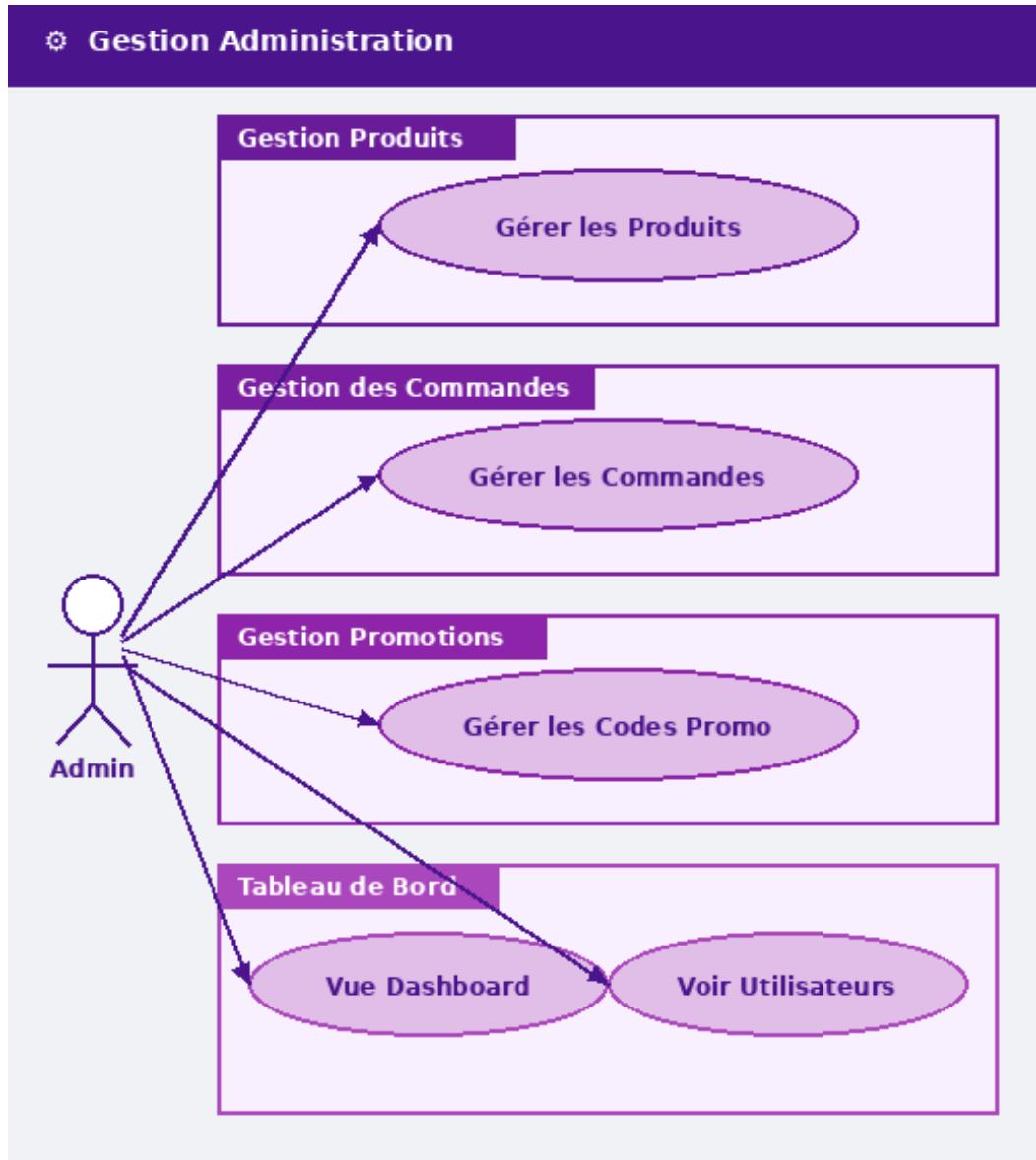


Figure 3.4: Diagramme de cas d'utilisation : Administration

### 3.4 Modélisation des données

Le diagramme de classes suivant définit la structure statique de notre système, montrant les entités principales (Utilisateur, Produit, Commande, etc.) et leurs relations. Il sert de base pour la création de la base de données.

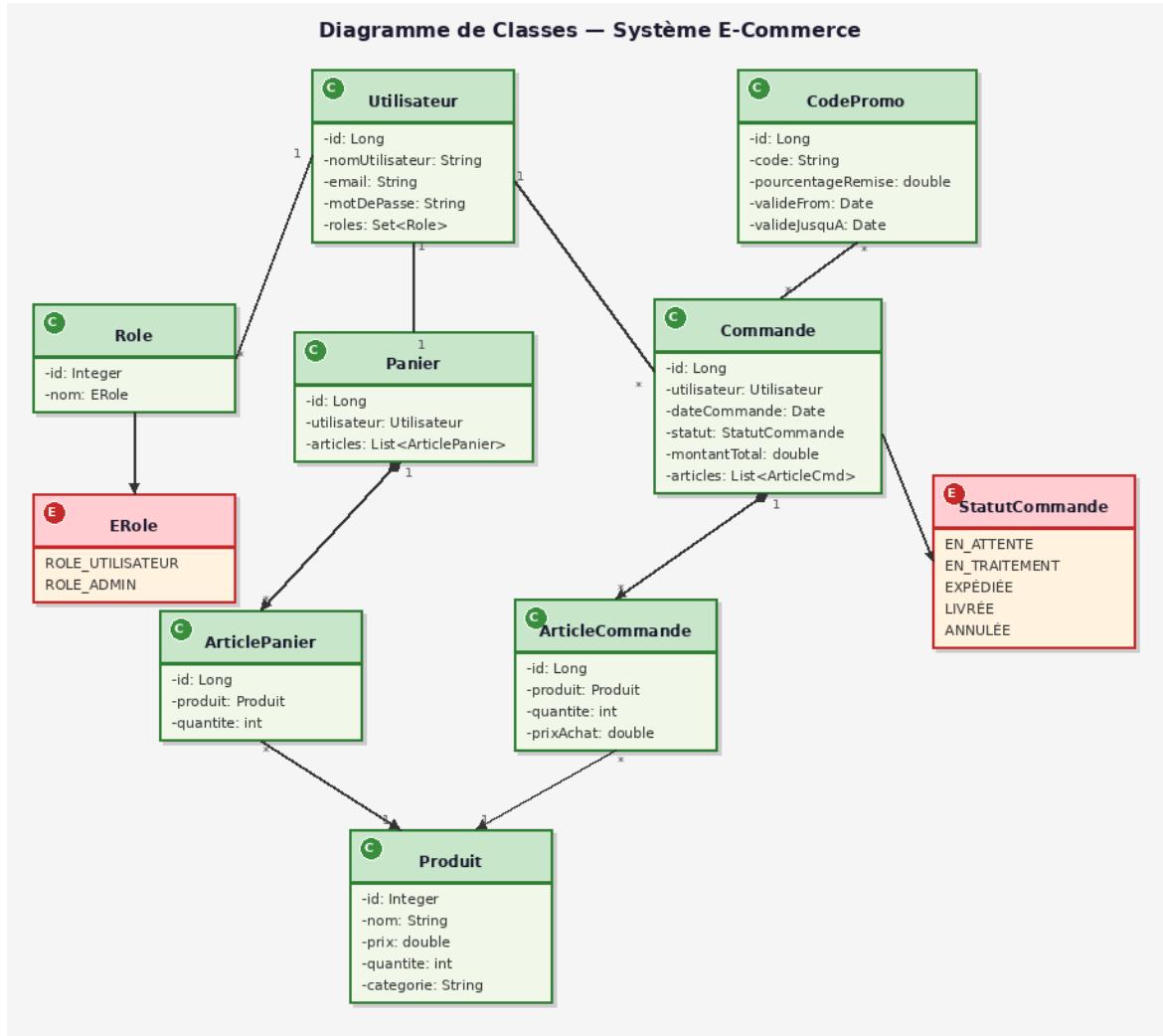


Figure 3.5: Diagramme de classes

Chaque entité contient des attributs spécifiques permettant une gestion cohérente des informations commerciales.

### 3.5 Diagrammes de séquence

Le diagramme de séquence est un diagramme comportemental UML représentant les interactions entre objets dans un ordre chronologique.

Il met en évidence :

- Les objets impliqués
- Les messages échangés
- L'ordre temporel des interactions

Ces diagrammes permettent de comprendre le déroulement dynamique des scénarios critiques du système.

### 3.5.1 Validation JWT

Ce diagramme illustre le processus de sécurisation des échanges via la validation des jetons JWT (JSON Web Token) lors des requêtes API.

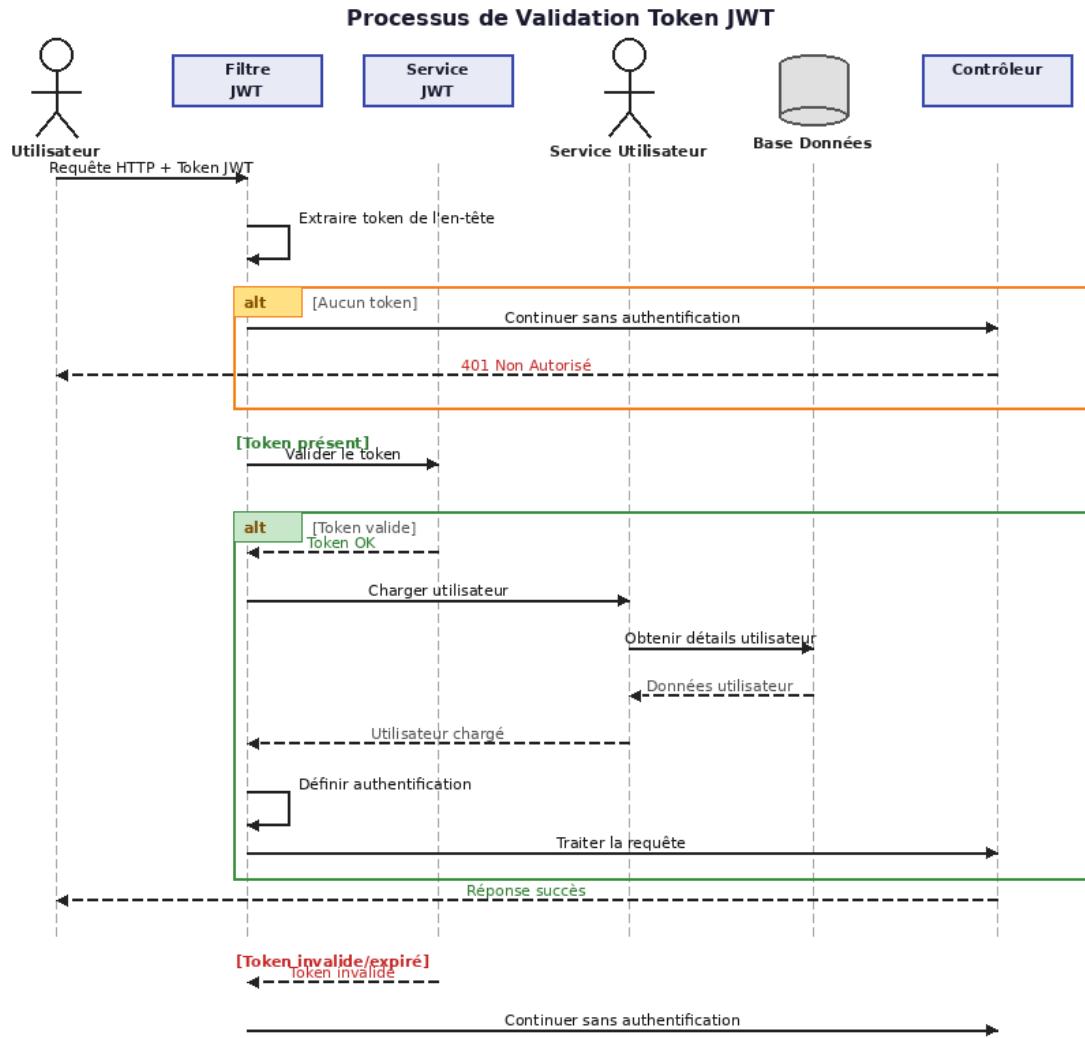


Figure 3.6: Diagramme de séquence : Validation JWT

### 3.5.2 Traitement des commandes

Le diagramme ci-dessous détaille le flux de traitement d'une commande, de sa validation par le client jusqu'à son enregistrement et la mise à jour du stock.

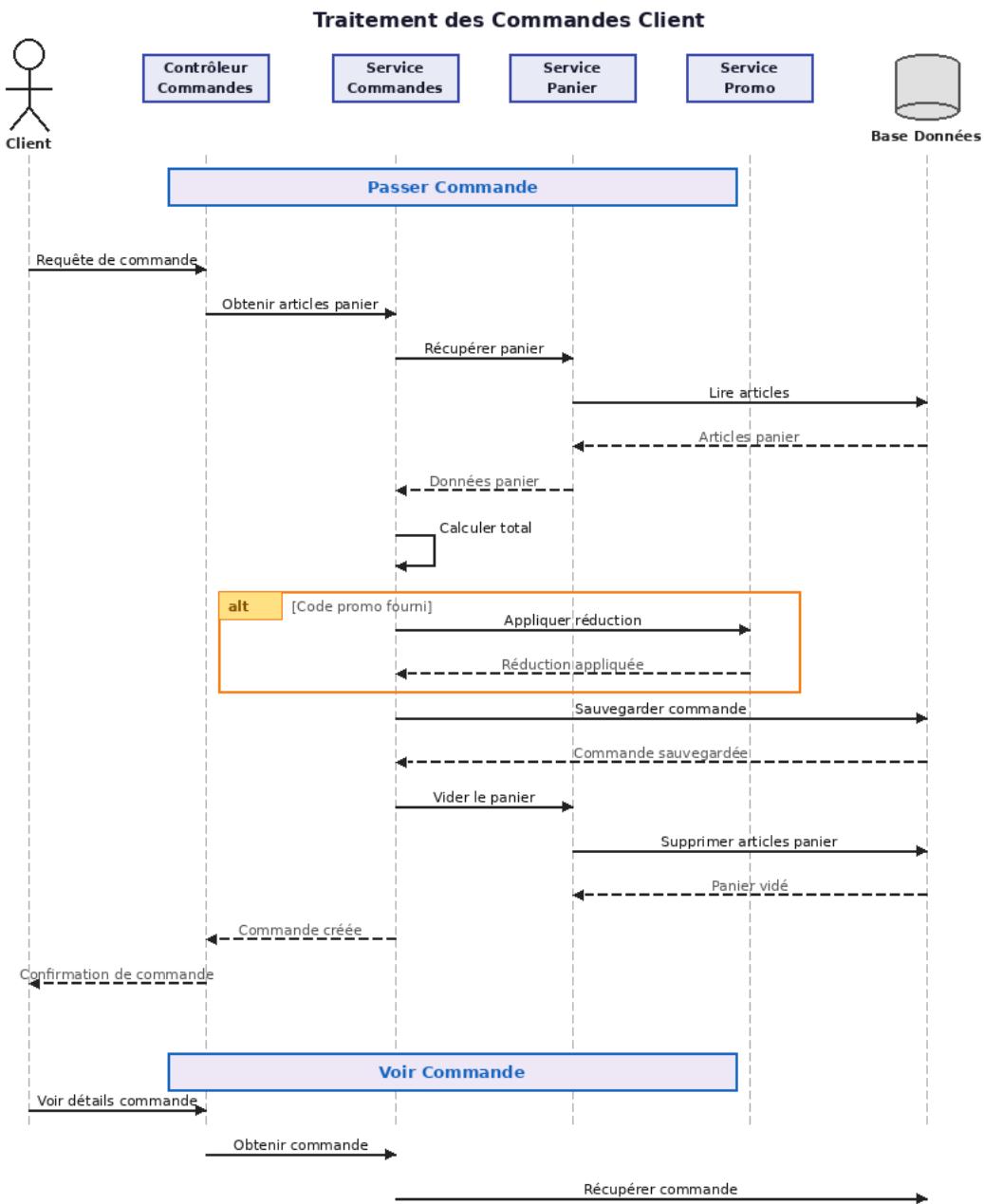


Figure 3.7: Diagramme de séquence : Traitement des commandes

### 3.6 Conclusion

Ce chapitre a permis de formaliser les exigences du système à travers une modélisation structurée basée sur le langage UML.

L'identification des acteurs, la définition des cas d'utilisation, la modélisation des données ainsi que l'analyse des scénarios dynamiques offrent une vision claire et cohérente du fonctionnement de l'application.

Ces modèles constituent une base essentielle pour la phase d'implémentation présentée dans le chapitre suivant. Ils garantissent une meilleure organisation du dévelop-

pement, une cohérence architecturale et une évolutivité maîtrisée du système.

# **CHAPITRE 4**

---

Implémentation et réalisation

---

## 4.1 Introduction

Ce chapitre présente la phase d'implémentation du système de gestion e-commerce. Après avoir étudié les besoins et conçu l'architecture du projet, nous avons procédé au développement concret de l'application en utilisant un ensemble de technologies modernes adaptées aux applications web professionnelles.

Nous détaillerons dans un premier temps l'environnement de développement et les outils utilisés, puis nous présenterons les différentes fonctionnalités implémentées à travers des captures d'écran commentées.

## 4.2 Environnement logiciel

L'environnement logiciel constitue la base technique sur laquelle repose le développement de l'application e-commerce. Il regroupe l'ensemble des outils, plateformes et technologies ayant permis la conception, le développement, les tests ainsi que le déploiement du système.

### 4.2.1 Java Development Kit (JDK)



Figure 4.1: logo de Java Development Kit (JDK)

Le Java Development Kit (JDK) est un ensemble d'outils permettant le développement d'applications en langage Java. Il comprend le compilateur Java (javac), la machine virtuelle Java (JVM) ainsi que les bibliothèques standards nécessaires à l'exécution des programmes.

Dans ce projet, le JDK a été utilisé pour compiler et exécuter l'application Spring Boot. Le langage Java constitue le socle du développement backend, offrant robustesse, portabilité et sécurité [2].

#### 4.2.2 IntelliJ IDEA

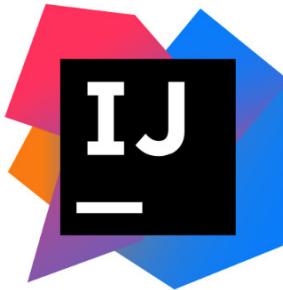


Figure 4.2: logo d'IntelliJ IDEA

IntelliJ IDEA est un environnement de développement intégré (IDE) conçu principalement pour le développement Java. Il offre des fonctionnalités avancées telles que l'autocomplétion intelligente, la détection d'erreurs en temps réel, le débogage interactif ainsi que l'intégration avec les outils de gestion de projet comme Maven et Git.

Dans le cadre de ce projet, IntelliJ IDEA a été utilisé pour structurer le code selon l'architecture en couches, gérer les dépendances Maven et faciliter les phases de test et de correction [3].

#### 4.2.3 Git

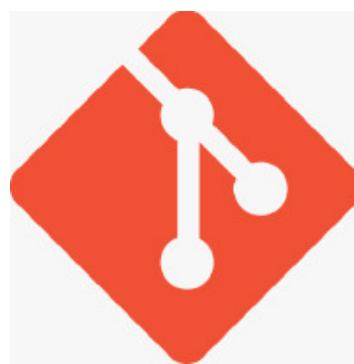


Figure 4.3: logo de Git

Git est un système de gestion de version distribué permettant de suivre les modifications du code source, gérer différentes branches de développement et collaborer efficacement sur un projet logiciel.

Il a été utilisé pour versionner le code et assurer la traçabilité des modifications effectuées tout au long du développement [4].

#### 4.2.4 GitHub



Figure 4.4: logo de GitHub

GitHub est une plateforme d'hébergement de code basée sur Git. Elle permet le stockage distant du projet, la gestion des versions ainsi que la collaboration entre développeurs.

Dans ce projet, GitHub a servi de dépôt centralisé pour le code source et a facilité le déploiement continu vers la plateforme cloud [5].

#### 4.2.5 Railway (Plateforme Cloud)



Figure 4.5: logo de Railway

Railway est une plateforme cloud de type Platform as a Service (PaaS) permettant de déployer rapidement des applications web sans configuration complexe d'infrastructure [6].

Dans ce projet, Railway a été utilisé pour :

- Déployer l'application Spring Boot
- Héberger la base de données distante
- Gérer les variables d'environnement

**Interface des projets Railway** La figure suivante montre le tableau de bord principal de Railway. Cette interface permet de visualiser les différents projets déployés (application Spring Boot et base de données MySQL), ainsi que leur état (online, production, etc.).

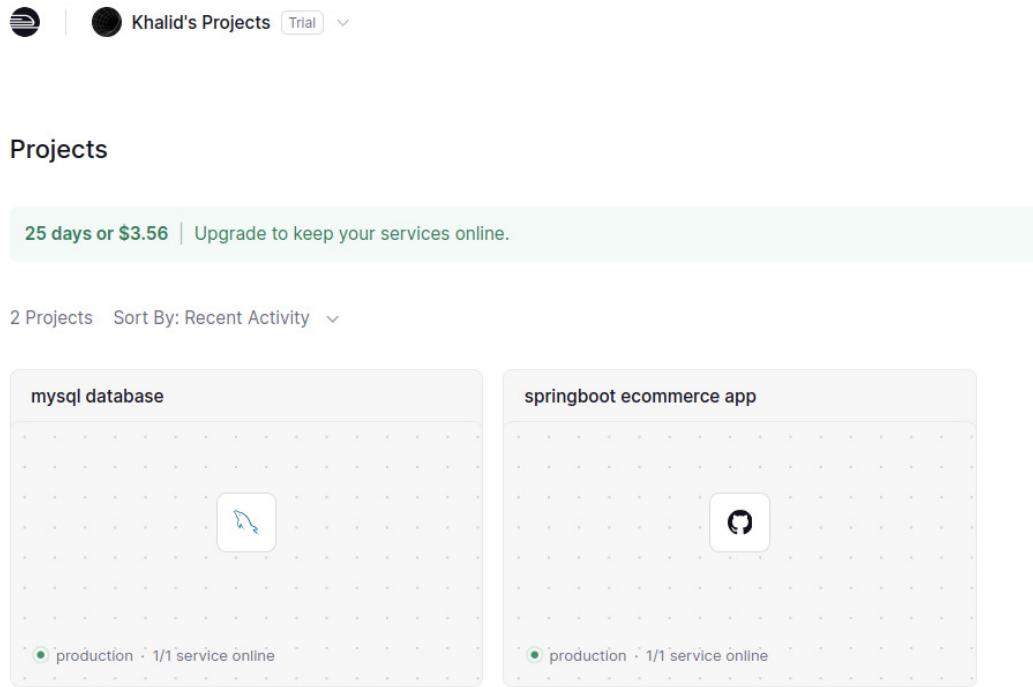


Figure 4.6: Tableau de bord des projets Railway

**Interface de la base de données MySQL** Railway permet l'hébergement d'une base de données MySQL managée. La figure suivante montre l'interface de gestion de la base de données, incluant les tables.

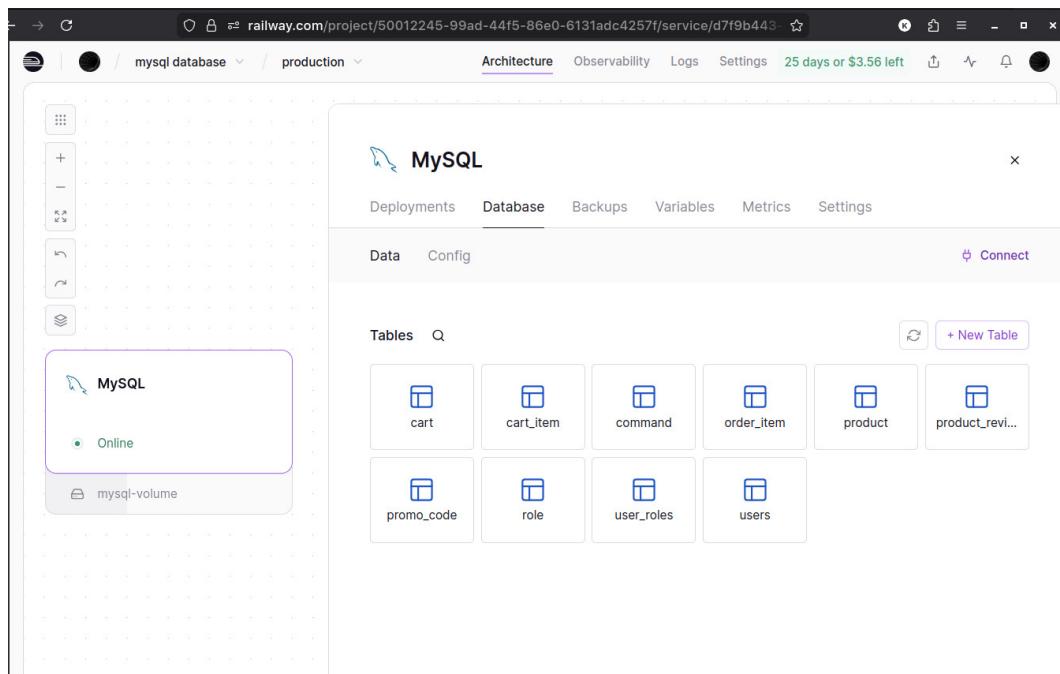


Figure 4.7: Interface de gestion de la base MySQL sur Railway

**Interface de déploiement de l'application** La figure suivante illustre l'interface de déploiement de l'application Spring Boot connectée au dépôt GitHub. Chaque mise à jour du code déclenche automatiquement un nouveau déploiement (CI/CD).

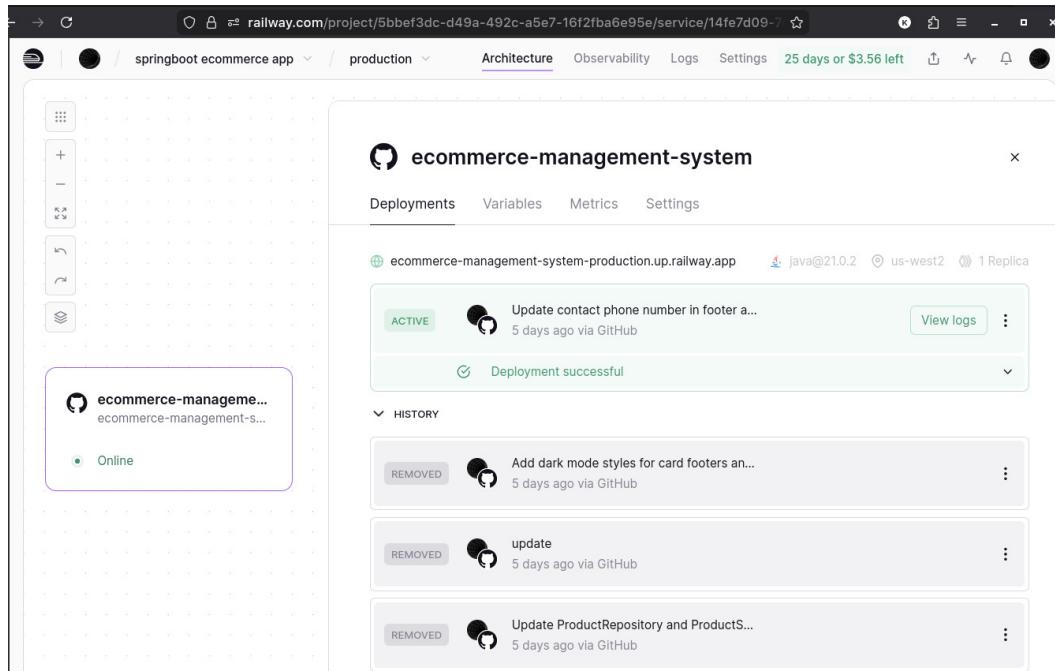


Figure 4.8: Interface de déploiement de l'application Spring Boot sur Railway

## 4.3 Technologies Backend

Le backend représente la partie serveur de l'application. Il assure le traitement des requêtes, l'implémentation de la logique métier, la gestion des accès aux données ainsi que la sécurisation des ressources.

Dans ce projet, le backend a été développé en Java en s'appuyant principalement sur l'écosystème Spring.

### 4.3.1 Langage Java

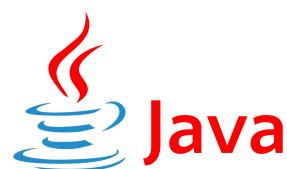


Figure 4.9: logo de Java

Java est un langage de programmation orienté objet, robuste et portable grâce au principe « Write Once, Run Anywhere » rendu possible par la machine virtuelle Java (JVM).

Il est particulièrement adapté au développement d'applications web d'entreprise en raison de sa stabilité, de sa gestion avancée de la mémoire et de son vaste écosystème de bibliothèques.

Dans ce projet, Java constitue le langage principal pour l'implémentation des contrôleurs, services, entités et configurations de sécurité [7].

### 4.3.2 Spring Boot



Figure 4.10: logo de Spring Boot

Spring Boot est un framework open source basé sur l'écosystème Spring, destiné à simplifier le développement d'applications Java. Il permet de créer rapidement des applications autonomes, prêtes à être déployées, en réduisant la configuration manuelle.

Il repose sur le principe de la configuration automatique (Auto-Configuration) et intègre un serveur embarqué (Tomcat par défaut), ce qui permet d'exécuter l'application sans configuration complexe.

Dans notre projet, Spring Boot constitue le framework principal du backend. Il assure la gestion des requêtes HTTP, l'intégration avec la base de données et la structuration du projet selon une architecture en couches [8].

### 4.3.3 Spring MVC



Figure 4.11: logo de Spring MVC

Spring MVC (Model-View-Controller) est un module du framework Spring permettant la gestion des requêtes web selon le modèle architectural MVC.

Il sépare l'application en trois composants :

- Modèle (Model) : gestion des données
- Vue (View) : affichage des informations
- Contrôleur (Controller) : traitement des requêtes et coordination entre le modèle et la vue

Dans notre application, Spring MVC permet de gérer les endpoints (URLs), traiter les requêtes des utilisateurs et retourner les réponses appropriées [9].

#### 4.3.4 Spring Data JPA



Figure 4.12: logo de Spring Data JPA

Spring Data JPA est un module facilitant l'accès aux bases de données relationnelles en utilisant le concept d'ORM (Object Relational Mapping).

Il permet de manipuler les données via des interfaces Repository sans écrire manuellement les requêtes SQL complexes [10].

Dans notre projet, Spring Data JPA est utilisé pour :

- L'insertion des produits
- La récupération des catégories
- la gestion des commandes
- l'authentification des utilisateurs

#### 4.3.5 Hibernate



Figure 4.13: logo de Hibernate

Hibernate est une implémentation de la spécification JPA (Java Persistence API). Il assure le mapping objet-relationnel (ORM), permettant de convertir les objets Java en tables relationnelles [11].

Hibernate gère :

- Le mapping des entités via annotations (@Entity, @OneToMany, etc.)
- La gestion des transactions
- La génération automatique des requêtes SQL

Dans ce projet, Hibernate fonctionne en arrière-plan via Spring Data JPA [11].

#### 4.3.6 Apache Maven



Figure 4.14: logo d'Apache Maven

Apache Maven est un outil de gestion et d'automatisation de projet basé sur le concept du Project Object Model (POM). Il permet de gérer les dépendances, compiler le code, exécuter les tests unitaires et générer les artefacts nécessaires au déploiement.

Dans cette application, Maven assure la gestion des bibliothèques telles que Spring Boot, Spring Security, JWT et MySQL Connector. Il facilite également la portabilité du projet entre différents environnements [12].

### 4.3.7 Project Lombok



Figure 4.15: logo de Project Lombok

Project Lombok est une bibliothèque Java qui réduit considérablement le code répétitif (boilerplate code) en générant automatiquement des méthodes comme les getters, setters, constructeurs et méthodes `toString` via des annotations.

Dans ce projet, Lombok a permis d'améliorer la lisibilité des classes entités et des modèles, tout en réduisant la complexité du code source [13].

## 4.4 Technologies Frontend

Le frontend représente la partie visible de l'application avec laquelle l'utilisateur interagit directement. Il assure l'affichage des données, la mise en forme des interfaces ainsi que l'interactivité du système.

Dans ce projet, le frontend repose sur des technologies web standards combinées avec un moteur de templates côté serveur.

### 4.4.1 HTML



Figure 4.16: logo de HTML

HTML (HyperText Markup Language) est le langage standard de structuration des pages web. Il permet de définir les éléments de base d'une interface, tels que les formulaires, tableaux, boutons et sections.

Dans notre application, HTML est utilisé pour concevoir les différentes interfaces : page d'accueil, formulaire d'inscription, page produit, panier et espace administrateur [14].

#### 4.4.2 CSS



Figure 4.17: logo de CSS

CSS (Cascading Style Sheets) est un langage de mise en forme utilisé pour styliser les pages HTML. Il permet de définir les couleurs, les polices, les espacements, ainsi que la disposition des éléments.

Dans ce projet, CSS est utilisé pour assurer une interface moderne, responsive et ergonomique, améliorant ainsi l'expérience utilisateur [15].

#### 4.4.3 JavaScript



Figure 4.18: logo de JavaScript

JavaScript est un langage de programmation exécuté côté client dans le navigateur. Il permet d'ajouter de l'interactivité aux pages web [16].

Dans notre plateforme e-commerce, JavaScript est utilisé pour :

- La gestion dynamique du panier
- Les filtres de recherche
- L'affichage interactif des graphiques statistiques

#### 4.4.4 Bootstrap



Figure 4.19: logo de Bootstrap

Bootstrap est un framework frontend open-source permettant de concevoir des interfaces web modernes, responsives et cohérentes. Il repose sur une grille (Grid System) flexible et propose un ensemble de composants prêts à l'emploi tels que les boutons, formulaires, cartes, barres de navigation et modales.

L'un des principaux avantages de Bootstrap est sa capacité à adapter automatiquement l'interface aux différentes tailles d'écran (ordinateurs, tablettes, smartphones), ce qui facilite la conception d'applications responsives.

Dans le cadre de ce projet e-commerce, Bootstrap a été utilisé pour :

- Structurer la mise en page des interfaces
- Styliser les formulaires d'inscription et de connexion
- Concevoir le tableau de bord administrateur
- Améliorer l'expérience utilisateur grâce à des composants visuels harmonisés

L'intégration de Bootstrap a permis d'accélérer le développement de l'interface tout en garantissant une présentation professionnelle et ergonomique [17].

#### 4.4.5 Thymeleaf



Figure 4.20: logo de Thymeleaf

Thymeleaf est un moteur de templates Java utilisé pour générer des pages HTML dynamiques côté serveur.

Il permet d'intégrer des données provenant du backend directement dans les pages HTML, tout en gardant un code propre et lisible [18].

#### 4.4.6 Chart.js



Chart.js

Figure 4.21: logo de Chart.js

Chart.js est une bibliothèque JavaScript permettant la création de graphiques interactifs (courbes, barres, diagrammes circulaires).

Dans le tableau de bord administrateur, Chart.js est utilisé pour visualiser :

- L'évolution du chiffre d'affaires
- La répartition des commandes selon leur statut

Cette visualisation facilite l'analyse des performances commerciales [19].

## 4.5 Base de données

La base de données constitue le composant central de persistance du système. Elle assure le stockage structuré et durable des informations relatives aux utilisateurs, produits, commandes et catégories.

### 4.5.1 MySQL



Figure 4.22: logo de MySQL

MySQL est un système de gestion de base de données relationnelle (SGBDR) open source. Il permet de stocker, organiser et gérer les données de manière structurée [20].

Dans notre application, MySQL est utilisé pour :

- Stocker les informations des utilisateurs
- Enregistrer les produits
- Gérer les catégories
- Conserver les commandes

### 4.5.2 Configuration de la source de données

La connexion entre l'application Spring Boot et la base MySQL est configurée dans le fichier `application.properties`.

Les paramètres principaux comprennent :

- L'URL de la base de données
- Le nom d'utilisateur
- Le mot de passe
- Le driver JDBC

Spring Boot configure automatiquement la source de données grâce à son mécanisme d'auto-configuration.

Cette approche simplifie la connexion à la base locale ainsi qu'à la base distante déployée sur Railway.

### 4.5.3 Mapping Objet-Relationnel (ORM)

Le Mapping Objet-Relationnel (ORM) est une technique permettant de convertir les objets Java en tables relationnelles.

Dans ce projet, Hibernate (via JPA) assure ce mapping à l'aide d'annotations telles que :

- @Entity
- @Id
- @OneToMany
- @ManyToOne

Cette abstraction permet de manipuler les données sous forme d'objets Java sans écrire directement des requêtes SQL complexes.

## 4.6 Sécurité de l'application

La sécurité constitue un élément fondamental dans toute application e-commerce. Elle vise à protéger les données sensibles (mots de passe, informations utilisateurs, commandes) et à contrôler l'accès aux différentes fonctionnalités du système.

Dans ce projet, la sécurité repose principalement sur l'intégration de Spring Security, l'utilisation des JSON Web Tokens (JWT) ainsi que le hachage sécurisé des mots de passe avec BCrypt.

### 4.6.1 Concepts fondamentaux

La sécurité d'une application web repose principalement sur deux notions essentielles :

- **Authentification** : processus permettant de vérifier l'identité d'un utilisateur (login + mot de passe).

- **Autorisation** : mécanisme déterminant les droits d'accès d'un utilisateur authentifié (rôle USER ou ADMIN).

Dans notre système, un utilisateur doit d'abord s'authentifier pour obtenir un token JWT. Ensuite, son rôle détermine les ressources accessibles.

#### 4.6.2 Spring Security



Figure 4.23: logo de Spring Security

Spring Security est un framework puissant dédié à la sécurisation des applications basées sur Spring. Il fournit un ensemble de mécanismes pour :

- Gérer l'authentification
- Contrôler les autorisations
- Protéger contre les attaques CSRF
- Sécuriser les endpoints REST

Dans notre application e-commerce, Spring Security a été configuré pour [21] :

- Restreindre l'accès aux pages administrateur
- Protéger les routes sensibles
- Intégrer un filtre JWT personnalisé

#### 4.6.3 JSON Web Token (JWT)



Figure 4.24: logo de JWT

JSON Web Token (JWT) est un standard ouvert (RFC 7519) permettant l'échange sécurisé d'informations entre deux parties sous forme d'objet JSON signé.

Un token JWT est composé de trois parties :

- Header
- Payload
- Signature

Dans notre système :

1. L'utilisateur s'authentifie avec ses identifiants.
2. Le serveur génère un JWT signé.
3. Le client envoie ce token dans l'en-tête HTTP `Authorization`.
4. Le serveur valide le token avant d'autoriser l'accès.

L'approche JWT permet une architecture **stateless**, ce qui signifie que le serveur ne conserve pas de session active. Cela améliore la scalabilité et la performance de l'application [22].

#### 4.6.4 BCrypt et protection des mots de passe

Le stockage des mots de passe en clair représente une faille de sécurité critique. Pour éviter cela, nous utilisons l'algorithme BCrypt.

BCrypt est un algorithme de hachage adaptatif qui :

- Ajoute un **salt** automatique
- Rend le hachage lent (résistance aux attaques par force brute)
- Permet d'ajuster le facteur de complexité

Spring Security fournit la classe `BCryptPasswordEncoder` pour encoder les mots de passe avant leur enregistrement en base de données.

Ainsi, même en cas de fuite de données, les mots de passe restent protégés.

#### 4.6.5 Gestion des rôles et autorisations

Le système implémente un contrôle d'accès basé sur les rôles :

- **ROLE\_USER** : accès aux fonctionnalités classiques (consultation produits, panier, commandes).

- **ROLE\_ADMIN** : accès au tableau de bord et aux fonctionnalités de gestion (CRUD produits, gestion commandes, codes promo).

Les restrictions sont configurées dans la classe de configuration Spring Security via des règles d'accès sur les endpoints.

Cette séparation garantit que seules les personnes autorisées peuvent modifier les données sensibles.

#### 4.6.6 Gestion des rôles et autorisations

Le système implémente un contrôle d'accès basé sur les rôles :

- **ROLE\_USER** : accès aux fonctionnalités classiques (consultation produits, panier, commandes).
- **ROLE\_ADMIN** : accès au tableau de bord et aux fonctionnalités de gestion (CRUD produits, gestion commandes, codes promo).

Les restrictions sont configurées dans la classe de configuration Spring Security via des règles d'accès sur les endpoints.

Cette séparation garantit que seule les personnes autorisées peuvent modifier les données sensibles.

### 4.7 Architecture logicielle du projet

L'application e-commerce développée repose sur une architecture en couches (Layered Architecture), un modèle d'organisation logicielle permettant de séparer clairement les responsabilités de chaque composant du système.

Cette approche favorise :

- Une meilleure lisibilité du code
- Une maintenance facilitée
- Une évolutivité maîtrisée
- Une séparation claire entre logique métier et accès aux données

#### 4.7.1 Architecture globale de l'application

La figure suivante illustre l'architecture logicielle globale de l'application développée avec Spring Boot.

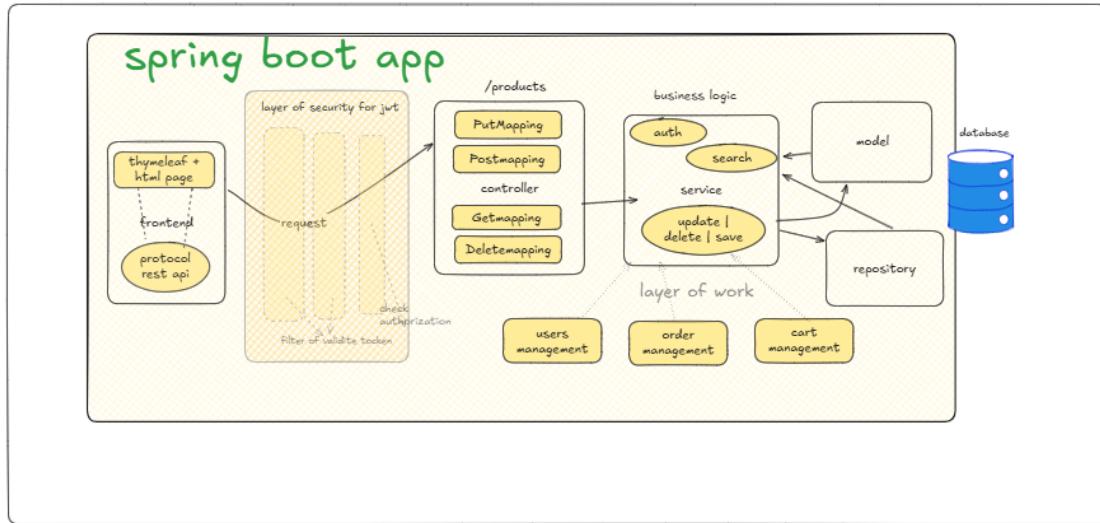


Figure 4.25: Architecture globale de l'application Spring Boot

Cette architecture met en évidence les différentes couches du système ainsi que leurs interactions.

### 1. Couche Frontend

La couche Frontend correspond à l'interface utilisateur. Elle est composée de pages HTML générées dynamiquement à l'aide du moteur de templates Thymeleaf.

Les requêtes utilisateur sont envoyées vers le backend via le protocole HTTP en utilisant des appels REST.

### 2. Couche de sécurité

Avant d'atteindre les contrôleurs, chaque requête passe par une couche de sécurité implémentée avec Spring Security et JWT.

Cette couche :

- Intercepte les requêtes HTTP
- Vérifie la présence du token JWT
- Valide la signature et l'intégrité du token
- Contrôle les autorisations selon le rôle (USER / ADMIN)

Cette architecture de sécurité garantit que seules les requêtes authentifiées et autorisées accèdent aux ressources protégées.

### 3. Couche Controller

Les contrôleurs représentent le point d'entrée des requêtes validées.

Ils utilisent les annotations suivantes :

- `@GetMapping`
- `@PostMapping`
- `@PutMapping`
- `@DeleteMapping`

Chaque contrôleur reçoit la requête HTTP, traite les paramètres et délègue la logique métier à la couche Service.

#### 4. Couche Service (Logique métier)

La couche Service contient la logique métier de l'application.

Elle gère notamment :

- L'authentification et la gestion des utilisateurs
- La recherche et la gestion des produits
- La gestion des commandes
- La gestion du panier
- L'application des règles métier

Cette couche assure la cohérence des traitements avant interaction avec la base de données.

#### 5. Couche Repository

La couche Repository assure l'accès aux données via Spring Data JPA.

Elle permet :

- L'insertion des données
- La récupération des enregistrements
- La mise à jour
- La suppression

Elle abstrait les détails techniques liés à la persistance.

#### 6. Couche Model (Entités)

La couche Model représente les entités métier (User, Product, Order, Cart).

Ces entités sont mappées aux tables relationnelles à l'aide de l'ORM Hibernate.

## 7. Base de données

La base de données MySQL constitue la couche de persistance finale.

Toutes les opérations CRUD transitent par les différentes couches avant d'atteindre cette couche.

## 8. Modules métier

Le système est structuré en modules fonctionnels :

- Gestion des utilisateurs
- Gestion des produits
- Gestion des commandes
- Gestion du panier

Chaque module respecte la même architecture en couches, garantissant une cohérence globale du système.

### 4.7.2 Organisation du projet dans l'environnement de développement

La figure suivante illustre la structure du projet dans l'environnement de développement (IDE). Elle montre l'organisation des packages selon l'architecture en couches.

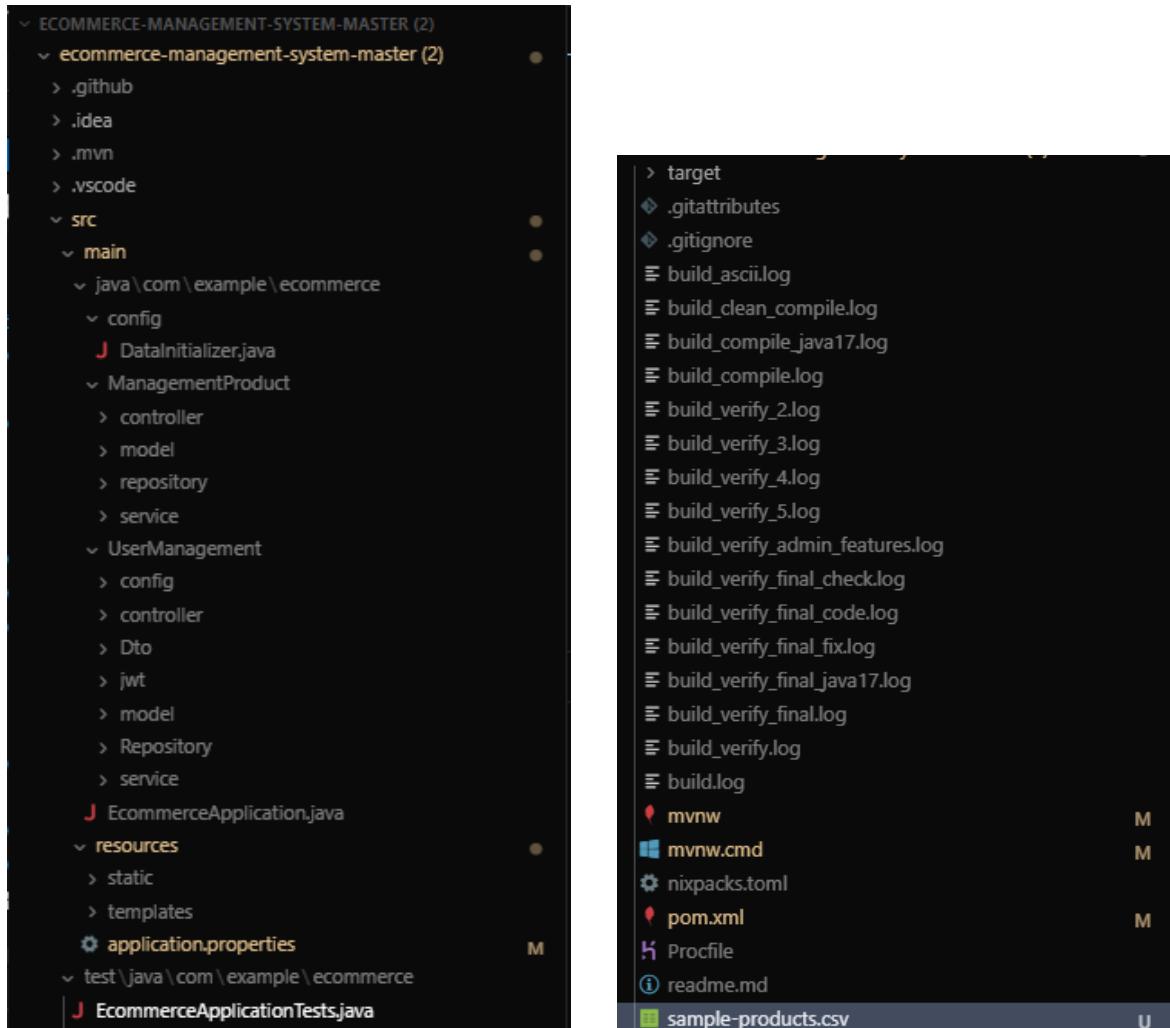


Figure 4.26: Architecture en couches du projet dans l'IDE

L'organisation en packages reflète directement la séparation des responsabilités :

- **controller** : gestion des requêtes HTTP
- **service** : logique métier
- **repository** : accès aux données
- **model** : entités
- **config** : configuration sécurité et JWT

Cette structuration garantit une architecture claire, modulaire et évolutive.

## 4.8 Présentation des fonctionnalités implémentées

L'application développée offre plusieurs fonctionnalités clés permettant de gérer efficacement une plateforme e-commerce. Ces fonctionnalités sont accessibles via une interface utilisateur intuitive et sont soutenues par une logique métier robuste.

#### 4.8.1 Interface d'accueil (Home)

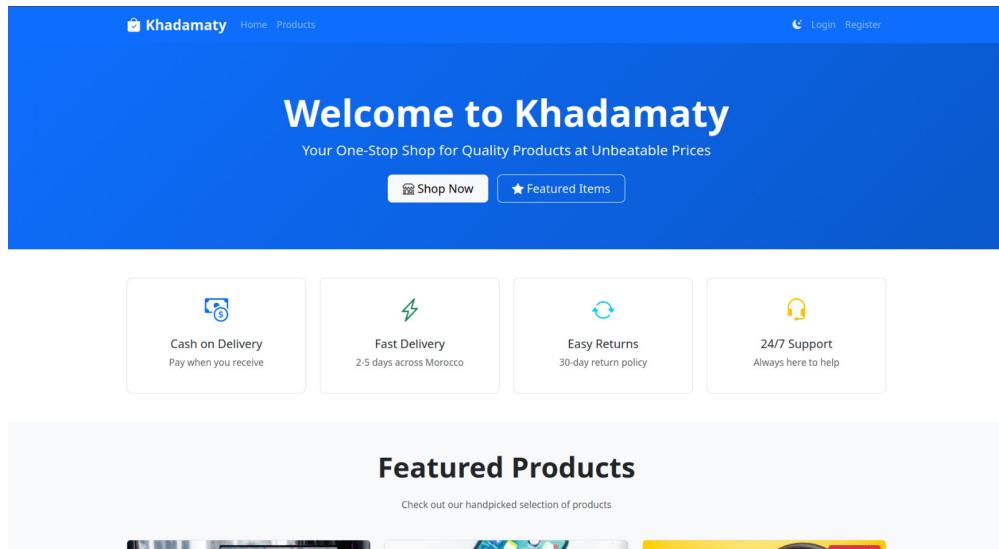


Figure 4.27: Interface d'accueil (Home)

L'interface d'accueil constitue la page principale de la plateforme. Elle représente le premier point de contact entre l'utilisateur et le système. Cette page affiche généralement les produits disponibles, les catégories principales ou les informations promotionnelles.

Elle permet à l'utilisateur de naviguer facilement vers les différentes sections du site, telles que les produits, le panier ou la connexion. L'objectif principal de cette interface est d'offrir une vue claire et structurée du contenu disponible, tout en garantissant une expérience utilisateur intuitive.

D'un point de vue technique, cette page récupère les données depuis la base de données via la couche Service, qui interagit avec la couche Repository. Les informations sont ensuite transmises à la vue pour être affichées dynamiquement à l'aide du moteur de templates.

Cette interface joue un rôle fondamental dans l'ergonomie générale du système, car elle influence directement l'expérience et la navigation de l'utilisateur.

#### 4.8.2 Interface d'inscription (Register)

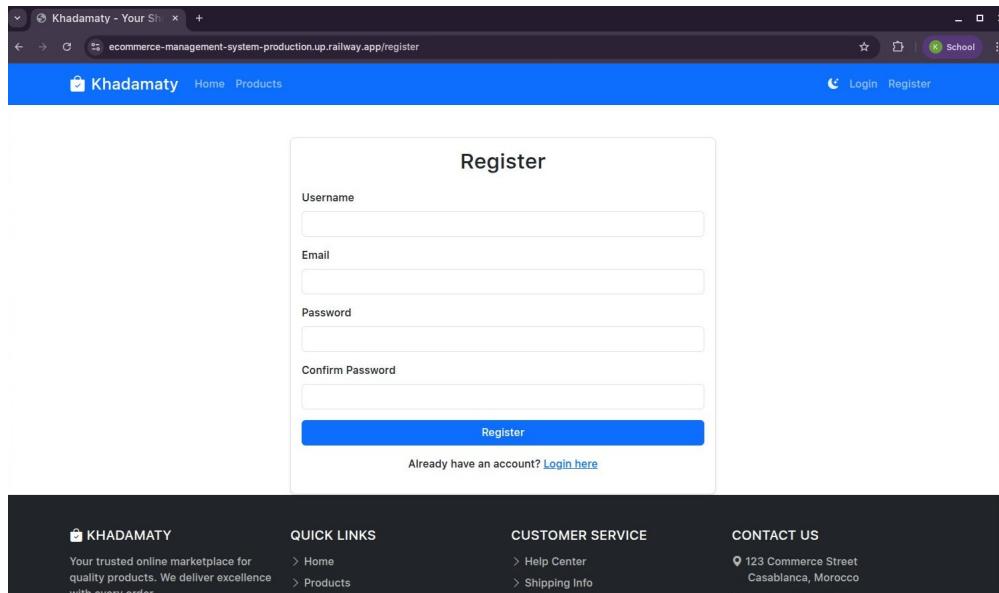


Figure 4.28: Interface d'inscription (Register)

L'interface d'inscription permet aux nouveaux utilisateurs de créer un compte afin d'accéder aux fonctionnalités de la plateforme. L'utilisateur doit renseigner les informations nécessaires telles que le nom, l'adresse e-mail, le mot de passe et éventuellement son rôle (administrateur ou client).

Lors de la soumission du formulaire, les données sont validées côté serveur afin de garantir leur cohérence et leur conformité (format de l'email, champs obligatoires, etc.). Les informations sont ensuite enregistrées dans la base de données via la couche Repository.

Cette fonctionnalité constitue un élément essentiel du système, car elle assure la gestion des accès et la traçabilité des utilisateurs.

### 4.8.3 Interface d'authentification (Login)

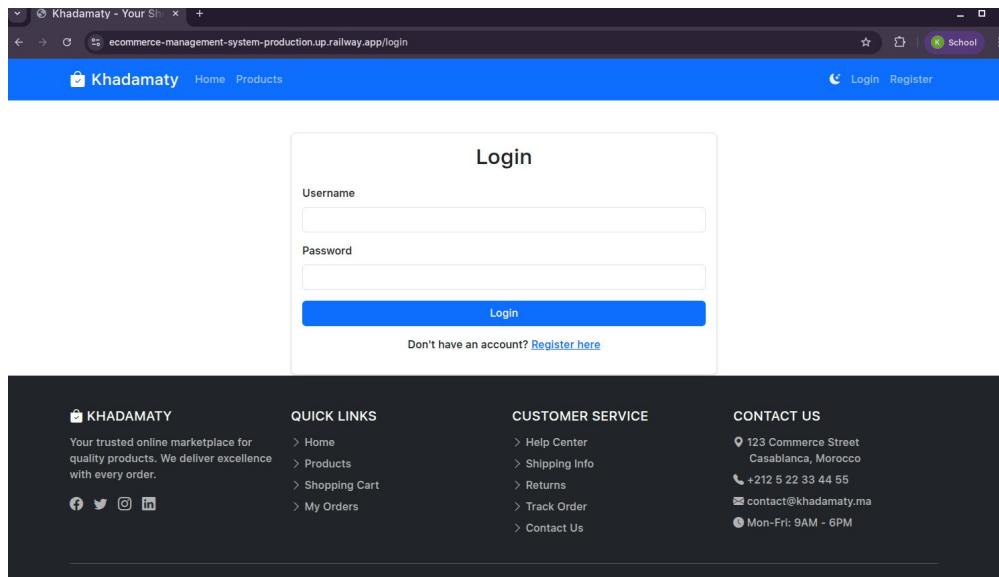


Figure 4.29: Interface d'authentification (Login)

L'interface de connexion permet aux utilisateurs enregistrés d'accéder à leur espace personnel. L'utilisateur saisit son adresse e-mail et son mot de passe, qui sont ensuite vérifiés par le système.

Le mécanisme d'authentification contrôle la validité des informations saisies et redirige l'utilisateur vers le tableau de bord correspondant à son rôle. Cette étape garantit la sécurité du système en limitant l'accès aux ressources protégées.

### 4.8.4 Consultation des produits

- Affichage de la liste des produits avec filtres

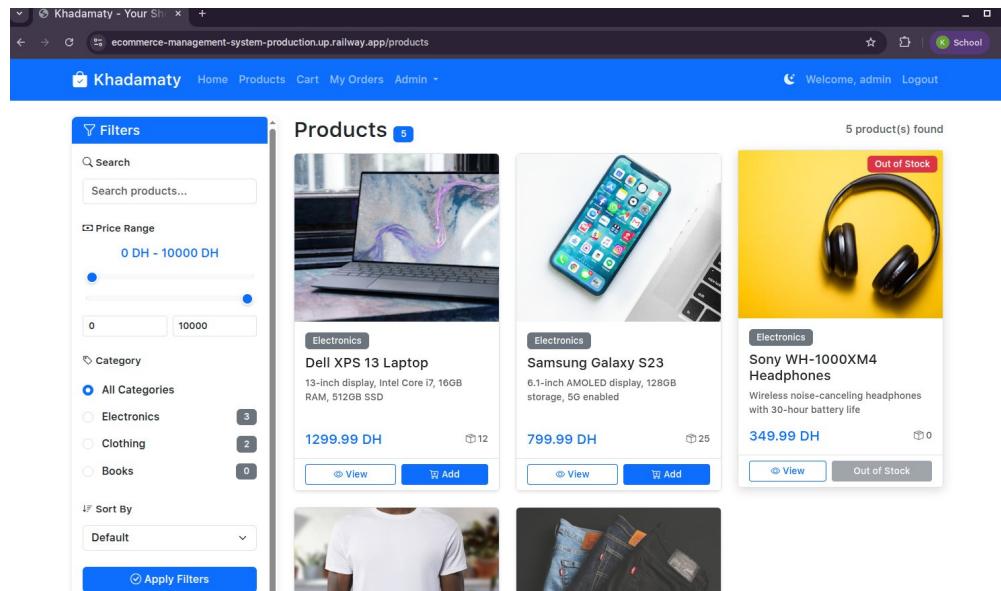


Figure 4.30: Interface de consultation des produits

Cette interface permet aux utilisateurs de consulter l'ensemble des produits disponibles sur la plateforme. Les produits sont affichés sous forme structurée, facilitant la navigation et la comparaison.

Un système de filtrage est intégré afin d'améliorer l'expérience utilisateur. L'utilisateur peut affiner sa recherche selon :

- La catégorie du produit
- Une plage de prix spécifique

Ce mécanisme permet de réduire le nombre de résultats affichés et d'accéder plus rapidement aux produits correspondant aux critères recherchés.

D'un point de vue technique, les filtres déclenchent des requêtes dynamiques traitées par le contrôleur. Ces paramètres sont transmis à la couche Service, qui interagit avec la couche Repository pour récupérer uniquement les produits correspondant aux critères sélectionnés.

Cette fonctionnalité améliore significativement l'ergonomie du système et simule le fonctionnement réel des plateformes e-commerce professionnelles.

- Affichage détaillé d'un produit

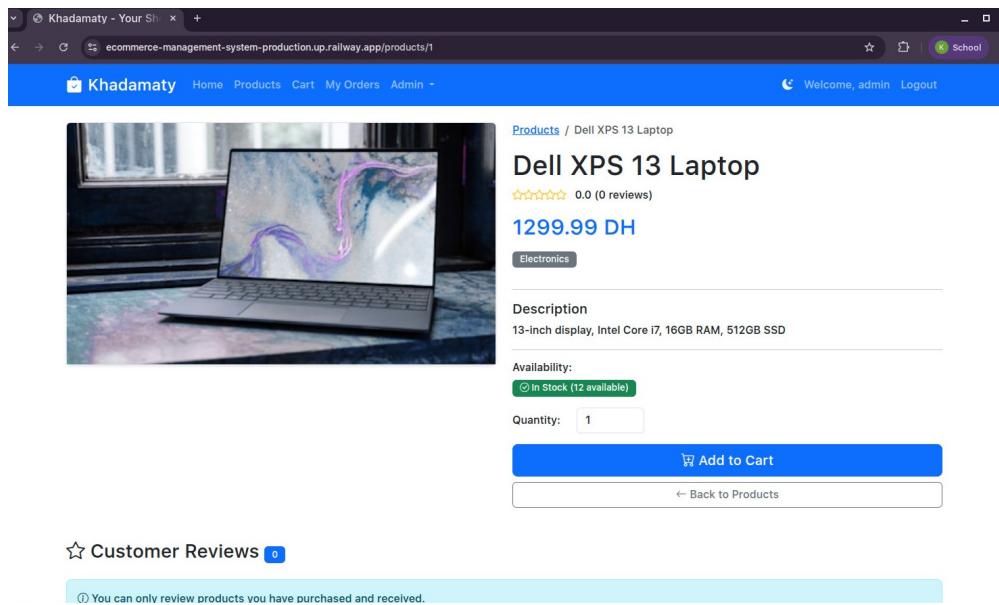


Figure 4.31: Interface de consultation détaillée d'un produit

Cette interface présente les informations détaillées d'un produit sélectionné. Elle affiche généralement :

- Le nom du produit
- La description complète
- Le prix
- La catégorie d'appartenance
- Les options disponibles (taille, couleur, etc.)

L'objectif de cette page est de fournir à l'utilisateur toutes les informations nécessaires avant l'ajout au panier.

Techniquement, l'identifiant du produit est transmis via l'URL au contrôleur, qui récupère les informations correspondantes depuis la base de données. Les données sont ensuite envoyées à la vue pour affichage dynamique.

Cette étape constitue un élément clé du parcours utilisateur, car elle précède l'acte d'achat.

#### 4.8.5 Interface du panier(Cart)

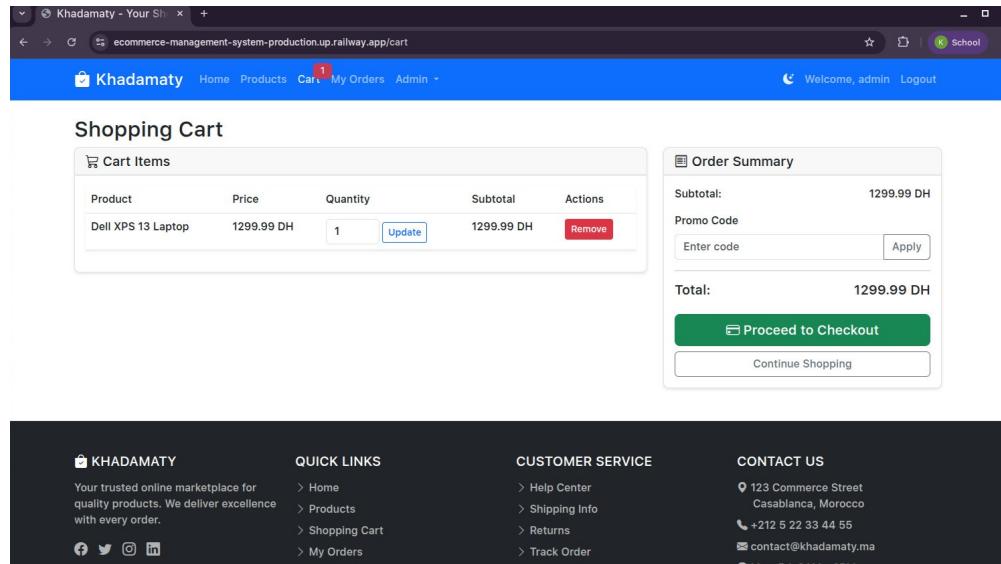


Figure 4.32: Interface du panier (Cart)

L’interface du panier permet à l’utilisateur de consulter les produits qu’il a sélectionnés avant de finaliser sa commande. Elle affiche les informations essentielles telles que le nom du produit, la quantité choisie, le prix unitaire ainsi que le total global de la commande.

L’utilisateur peut modifier la quantité d’un produit ou le supprimer du panier. Il a également la possibilité d’appliquer un code promotionnel afin de bénéficier d’une réduction, de continuer ses achats en retournant vers le catalogue, ou de lancer le processus de commande pour passer à l’étape de validation. Ces fonctionnalités garantissent une grande flexibilité dans le processus d’achat et contribuent à améliorer l’expérience utilisateur.

Sur le plan technique, le panier est généralement géré soit par la session utilisateur, soit par une entité persistée en base de données. Les opérations d’ajout, de suppression, de mise à jour et d’application de promotions sont traitées par la couche Controller, tandis que la logique métier correspondante est assurée par la couche Service.

Le panier constitue une étape intermédiaire essentielle dans le processus de commande, car il prépare la validation finale et l’enregistrement de la transaction dans le système, tout en offrant à l’utilisateur un contrôle complet sur ses choix avant confirmation.

#### 4.8.6 Interface de consultation des commandes

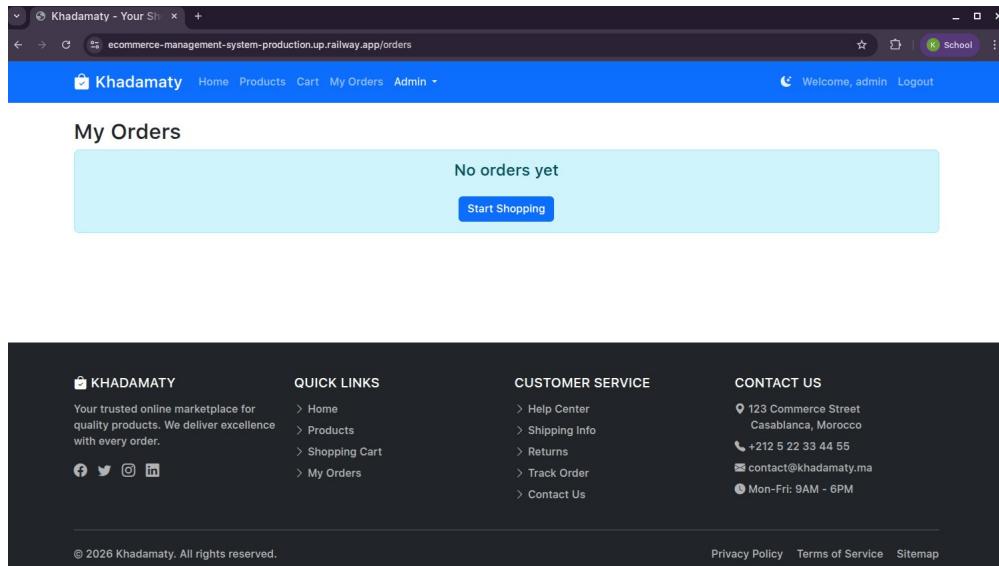


Figure 4.33: Interface de consultation des commandes

Cette interface affiche les informations essentielles liées aux commandes, telles que leur état, leur date ou leur contenu, offrant ainsi une vue claire sur les transactions réalisées. Lorsqu'aucune commande n'a encore été enregistrée, un message informatif est présenté pour indiquer la situation, accompagné d'une option permettant de rediriger l'utilisateur vers le catalogue afin de commencer ses achats.

Sur le plan fonctionnel, cette page facilite le suivi des opérations effectuées et renforce la transparence du processus d'achat. Elle contribue à améliorer l'expérience utilisateur en lui donnant un contrôle et une visibilité sur ses interactions avec le système.

D'un point de vue technique, les données des commandes sont récupérées depuis la base de données via la couche Repository, traitées par la couche Service, puis affichées par la couche Controller. Cette organisation garantit une gestion structurée, fiable et maintenable des informations liées aux commandes.

Cette interface représente ainsi une étape importante du parcours utilisateur, en assurant le suivi post-achat et en favorisant une interaction continue avec la plateforme.

#### 4.8.7 Interface d'administration des produits

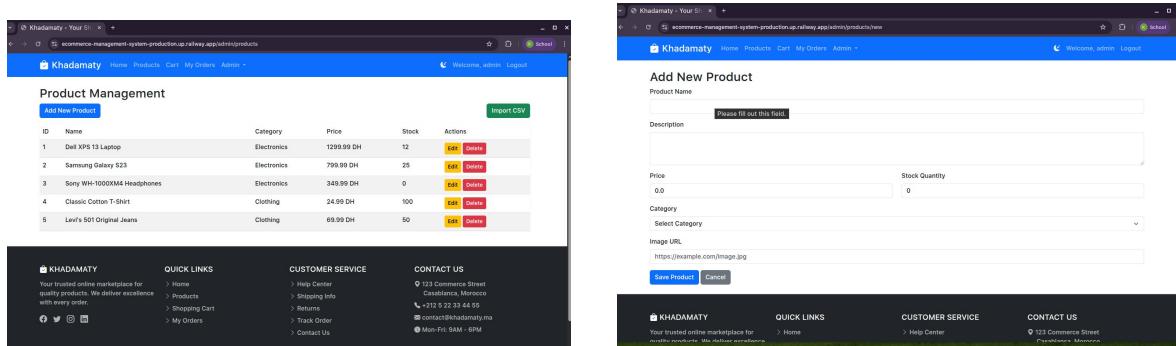


Figure 4.34: Interface d'administration des produits

La gestion des produits dans l'application repose sur deux interfaces complémentaires destinées à l'administrateur : une interface de gestion des produits et une interface dédiée à l'ajout de nouveaux produits. Cette séparation permet une organisation claire des fonctionnalités et facilite l'utilisation du système.

L'interface de gestion des produits offre une vue d'ensemble du catalogue. Elle permet à l'administrateur de consulter la liste complète des produits, de modifier leurs informations ou de les supprimer. Elle intègre également une fonctionnalité d'importation CSV, permettant l'ajout massif de produits de manière rapide et efficace, ce qui optimise le processus de gestion du catalogue.

L'interface d'ajout de produit est spécifiquement conçue pour la création d'un nouvel article. Elle propose un formulaire structuré permettant de saisir les informations essentielles telles que le nom, la description, le prix, la catégorie, le stock et l'image. Cette interface favorise une saisie claire et contrôlée des données.

Ensemble, ces deux interfaces implémentent les opérations fondamentales du modèle CRUD (Create, Read, Update, Delete), essentielles aux applications de gestion. Sur le plan architectural, chaque action est traitée par le contrôleur d'administration, qui sollicite la couche Service pour exécuter la logique métier avant interaction avec la base de données. Cette organisation garantit une séparation des responsabilités, une meilleure maintenabilité et une évolutivité du système.

#### 4.8.8 Interface d'administration des commandes

Order ID	Customer	Phone	Products	Date	Total	Status	Actions
#8	AnassW3re	0628792253	Dell XPS 13 Laptop (x7) Samsung Galaxy S23 (x1)	2026-02-14 22:05	9899.92 DH	PENDING	<button>Confirm</button> <button>Cancel</button>
#7	AnassW3re	0628792253	Dell XPS 13 Laptop (x11)	2026-02-13 23:43	14199.89 DH	CANCELLED	
#6	AnassW3re	0628792253	Sony WH-1000XM4 Headphones (x40)	2026-02-13 23:39	13999.6 DH	DELIVERED	
#5	AnassW3re	0628792253	Dell XPS 13 Laptop (x1)	2026-02-13 23:38	1299.99 DH	DELIVERED	
#4	AnassW3re	0628792253	Dell XPS 13 Laptop (x1) Classic Cotton T-Shirt (x1) Levi's 501 Original Jeans (x1)	2026-02-13 23:37	1394.97 DH	CANCELLED	
#3	hajar	+212618908770	Dell XPS 13 Laptop (x1)	2026-02-13 23:24	1299.99 DH	CANCELLED	
#2	boutefah	0688569946	Dell XPS 13 Laptop (x1)	2026-02-13 23:23	1299.99 DH	DELIVERED	
#1	boutefah		Dell XPS 13 Laptop (x1)	2026-02-13 17:40	1299.99 DH	DELIVERED	

**KHADAMATY**  
Your trusted online marketplace for quality products. We deliver excellence with every order.

**QUICK LINKS**

- > Home
- > Products
- > Shopping Cart

**CUSTOMER SERVICE**

- > Help Center
- > Shipping Info
- > Returns

**CONTACT US**

- 📍 123 Commerce Street  
Casablanca, Morocco
- 📞 +212 5 22 33 44 55

Figure 4.35: Interface d'administration des commandes

Cette interface permet à l'administrateur de superviser et gérer l'ensemble des commandes effectuées sur la plateforme. Elle présente une vue structurée sous forme de tableau contenant les informations essentielles de chaque commande, notamment l'identifiant, le client, les produits commandés, la date, le montant total ainsi que l'état de la commande.

Le système offre plusieurs fonctionnalités opérationnelles importantes. L'administrateur peut consulter les détails de chaque commande afin d'assurer le suivi des transactions. Des actions directes sont également disponibles pour modifier le statut d'une commande (en attente, confirmée, livrée ou annulée), permettant une gestion dynamique du cycle de traitement.

L'interface intègre également une fonctionnalité d'exportation des données au format CSV, facilitant l'analyse ou l'archivage des commandes en dehors du système. Cette option est particulièrement utile pour la gestion administrative et le reporting.

Sur le plan technique, cette fonctionnalité repose sur une interaction entre la couche Controller et la couche Service, garantissant la récupération, la mise à jour et la persistance des données dans la base de données. Cette organisation assure la cohérence des informations tout en respectant l'architecture en couches adoptée dans le projet.

Ainsi, la gestion des commandes constitue un élément central du système e-commerce, permettant un contrôle efficace des opérations commerciales et assurant la traçabilité complète des transactions.

#### 4.8.9 Interface d'administration des codes promotionnels

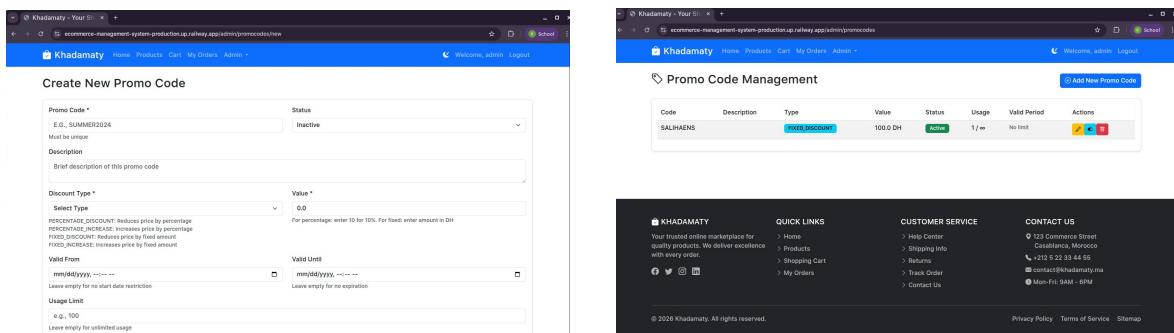


Figure 4.36: Interface d'administration des codes promotionnels

Cette section présente le module de gestion des codes promotionnels, destiné exclusivement à l'administrateur. Ce module permet de créer, configurer et administrer des offres promotionnelles afin d'influencer dynamiquement le prix des produits et d'encourager les achats.

La première interface permet la création d'un nouveau code promotionnel. L'administrateur peut définir plusieurs paramètres essentiels, tels que le nom du code, sa description, le type de remise (pourcentage ou montant fixe), la valeur appliquée, la période de validité ainsi que la limite d'utilisation. Ces informations permettent un contrôle précis du comportement du code promotionnel et garantissent sa cohérence avec les règles commerciales définies.

La seconde interface est dédiée à la gestion des codes existants. Elle offre des fonctionnalités administratives complètes permettant de modifier les paramètres d'un code, de l'activer ou de le désactiver selon les besoins, ou encore de le supprimer. Cette gestion dynamique permet d'adapter rapidement les stratégies promotionnelles en fonction du contexte commercial.

Sur le plan technique, ces opérations reposent sur l'interaction entre les couches Controller et Service, assurant la validation des données et leur enregistrement dans la base de données. Ce module illustre la capacité du système à intégrer des mécanismes commerciaux avancés tout en respectant l'architecture logicielle adoptée.

Ainsi, la gestion des codes promotionnels constitue un outil stratégique au sein de la plateforme e-commerce, offrant flexibilité et contrôle dans la mise en œuvre des politiques de tarification.

#### 4.8.10 Tableau de bord administrateur (Admin Dashboard)

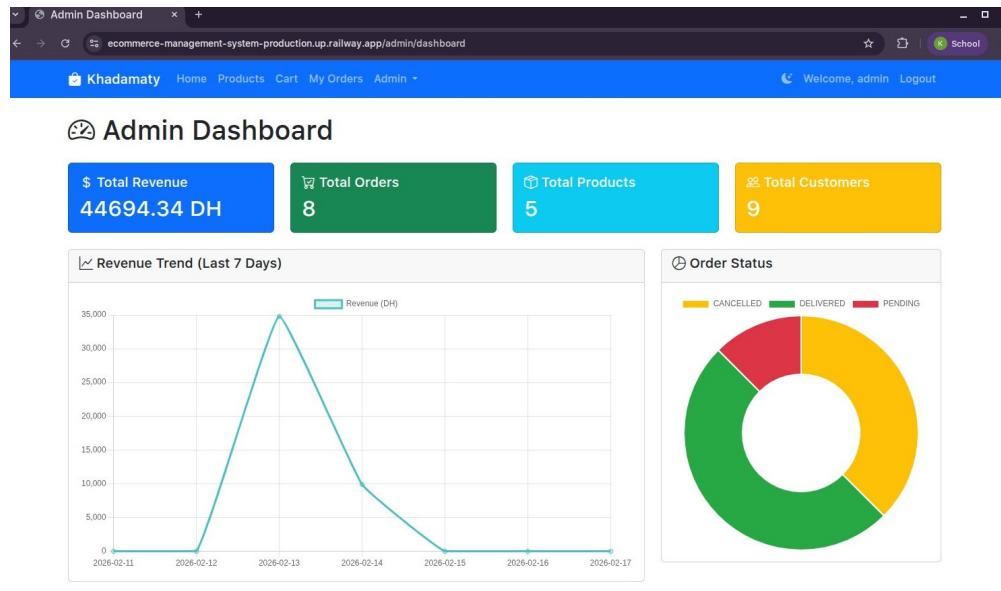


Figure 4.37: Interface du tableau de bord administrateur

Le tableau de bord administrateur constitue l'interface centrale de supervision du système e-commerce. Il offre une vue synthétique et analytique des principales données relatives à l'activité de la plateforme.

Dans la partie supérieure, des indicateurs statistiques globaux sont affichés sous forme de cartes récapitulatives. Ces indicateurs comprennent notamment :

- Le nombre total de produits disponibles
- Le nombre d'utilisateurs inscrits
- Le nombre de commandes passées
- Le chiffre d'affaires généré

Ces métriques permettent à l'administrateur d'avoir une vision rapide et claire de la performance globale du système.

La section suivante présente un graphique illustrant l'évolution du chiffre d'affaires sur les sept derniers jours. Cette représentation dynamique permet d'analyser les tendances de vente et d'identifier d'éventuelles variations d'activité. Les données sont récupérées depuis la base de données, agrégées côté serveur via la couche Service, puis affichées sous forme graphique à l'aide d'une bibliothèque de visualisation chart.js.

Enfin, un diagramme circulaire présente la répartition des commandes selon leur statut (en attente, livrée, annulée). Cette visualisation facilite le suivi opérationnel et permet d'identifier rapidement la proportion des commandes traitées ou en cours.

D'un point de vue architectural, le tableau de bord centralise des données provenant de plusieurs entités (Commandes, Produits, Utilisateurs) et démontre l'intégration cohérente des différentes couches du système. Il constitue ainsi un outil stratégique d'aide à la décision pour l'administrateur.

# Conclusion générale

Notre projet de fin de module, réalisé au sein du département d’Informatique de la Faculté Polydisciplinaire de Larache, a porté sur la conception et le développement d’une application e-commerce distribuée intégrant des mécanismes de sécurité avancés et un déploiement en environnement Cloud.

Cette expérience a constitué une opportunité concrète de mettre en pratique les connaissances théoriques acquises durant notre formation, notamment dans les domaines des applications distribuées, de l’architecture logicielle et de la sécurité informatique.

À travers les différentes phases du projet — analyse des besoins, modélisation UML, conception architecturale, développement, sécurisation et déploiement — nous avons pu nous confronter aux exigences réelles de la conception d’un système web complet. L’intégration de technologies telles que Spring Boot, Spring Security avec JWT et MySQL nous a permis d’approfondir nos compétences techniques et de mieux comprendre les enjeux liés à la performance, à la sécurité et à la scalabilité d’une application moderne.

Au-delà de l’aspect technique, ce projet nous a également permis de renforcer nos compétences en gestion du temps, en organisation méthodique du travail et en collaboration au sein d’une équipe. La planification à travers un diagramme de Gantt et la répartition claire des responsabilités ont contribué à assurer une progression cohérente et structurée.

Bien que l’application développée réponde aux objectifs fixés, plusieurs perspectives d’amélioration peuvent être envisagées, telles que l’intégration d’un système de paiement en ligne, la migration vers une architecture microservices, l’automatisation complète des tests ou encore la containerisation via Docker. Ces évolutions permettraient d’enrichir davantage la solution et de la rapprocher des standards industriels actuels.

Nous exprimons enfin notre gratitude envers la Faculté Polydisciplinaire de Larache et l’équipe pédagogique pour l’encadrement et les connaissances transmises tout au long de ce module. Cette expérience représente une étape importante dans notre parcours académique et constitue une base solide pour notre future insertion professionnelle dans le domaine des systèmes distribués et du développement web avancé.

# Webographie

[1] Qu'est-ce que le e-commerce ?

<https://www.ibm.com/fr-fr/think/topics/ecommerce>

[2] Java Development Kit Documentation

<https://www.oracle.com/java/technologies/javase-downloads.html>

[3] IntelliJ IDEA Documentation

<https://www.jetbrains.com/idea/documentation/>

[4] Git Documentation

<https://git-scm.com/doc>

[5] GitHub Documentation

<https://docs.github.com/en/get-started>

[6] Railway Documentation

<https://docs.railway.app/>

[7] Java Documentation

<https://docs.oracle.com/en/java/>

[8] Spring Boot Documentation

<https://spring.io/projects/spring-boot>

[9] Spring MVC Documentation

<https://docs.spring.io/spring-framework/reference/web/webmvc.html>

[10] Spring Data JPA Documentation

<https://spring.io/projects/spring-data-jpa>

[11] Hibernate Documentation

<https://hibernate.org/orm/documentation/>

- 
- [12] Apache Maven Documentation  
<https://maven.apache.org/documentation.html>
  - [13] Project Lombok Documentation  
<https://projectlombok.org/features/all>
  - [14] HTML Documentation  
<https://developer.mozilla.org/en-US/docs/Web/HTML>
  - [15] CSS Documentation  
<https://developer.mozilla.org/en-US/docs/Web/CSS>
  - [16] JavaScript Documentation  
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>
  - [17] Bootstrap Documentation  
<https://getbootstrap.com/docs/5.3/getting-started/introduction/>
  - [18] Thymeleaf Documentation  
<https://www.thymeleaf.org/documentation.html>
  - [19] Chart.js Documentation  
<https://www.chartjs.org/docs/latest/>
  - [20] MySQL Documentation  
<https://dev.mysql.com/doc/>
  - [21] Spring Security Documentation  
<https://docs.spring.io/spring-security/reference/index.html>
  - [22] JWT Documentation  
<https://jwt.io/>
  - [23] BCrypt Documentation  
<https://docs.spring.io/spring-security/reference/features/authentication/password-storage.html>