**3éme Année Cycle d’Ingénieur**

École Marocaine des Sciences de l’Ingénieur de Tanger

**Projet De Fin D’année**

**Filière**

5IIR Ingénierie Informatique et Réseaux

**Intitulé du Projet**

Développement d’une plateforme e-commerce basée sur une architecture micro services

**Membres de jury :**

**Réalisé par :**

OUMAIMA EL-MANSOURI

LOUBNA HABIL

ABDESAMIE

M.

**Encadré par :**

Mme. Nora Mouhib

**Année Universitaire 2024-2025**

**5IIR**

***Remerciements***

Nous souhaitons exprimer notre sincère gratitude à toutes les personnes qui nous ont accompagnés et soutenus tout au long de ce projet de fin d'année.

Nous tenons tout particulièrement à remercier notre encadrante, **Mme Nora Mouhib**, pour sa patience, sa bienveillance et ses précieux conseils. Elle a su nous guider et nous orienter à chaque étape de notre travail avec professionnalisme, tout en nous apportant des retours constructifs qui ont grandement contribué à l'avancement de notre projet. Son encadrement a été une source d'inspiration et de motivation pour notre groupe.

Nous remercions également nos camarades de groupe, qui ont fait preuve d'une excellente collaboration et d'un esprit d'équipe irréprochable. Leur implication, leurs idées et leur soutien mutuel ont permis de mener à bien ce projet.

Nous souhaitons également remercier l'ensemble des professeurs et de l'équipe pédagogique de l'EMSI, pour leurs enseignements et leur soutien tout au long de notre formation. Leur expertise et leurs conseils nous ont permis de développer les compétences nécessaires pour réaliser ce projet.

Enfin, nous adressons nos plus sincères remerciements à nos familles et à nos amis, pour leur patience, leur soutien moral et leur encouragement constant, qui ont été d'une grande aide durant les moments de stress et de doute.

# *Résumé*

Dans un contexte où les solutions de commerce en ligne connaissent une forte demande et nécessitent une gestion efficace et évolutive, le développement d’une plateforme e-commerce repose sur l’architecture microservices. Ce projet vise à créer une plateforme performante et évolutive permettant aux utilisateurs de naviguer, acheter des produits et gérer leurs commandes de manière optimale.

L'objectif de ce projet est de concevoir et de développer une plateforme e-commerce modulable, utilisant Spring Cloud pour une architecture microservices qui assure une scalabilité, une gestion sécurisée des utilisateurs, des paiements et des produits.

La solution développée permettra de :

* Gérer les utilisateurs : création de comptes, gestion des rôles et authentification sécurisée.
* Gérer les produits : affichage détaillé des produits, gestion du catalogue, recherche par catégories.
* Gérer les commandes : ajout au panier, validation de commande et suivi des statuts.
* Gérer les paiements : intégration de systèmes de paiement sécurisés et génération de factures.
* Fournir des statistiques : tableau de bord pour l’administrateur avec des rapports sur les ventes et l’activité de la plateforme.

Le projet sera réalisé par un groupe de trois personnes sur une durée de trois mois, avec pour but de livrer une application fonctionnelle et facile à maintenir grâce à l’utilisation de microservices.

Mots-clés : E-commerce, Spring Boot, architecture microservices, gestion des utilisateurs, gestion des produits, gestion des commandes, paiement en ligne, Spring Cloud.

# *Abstract*

In a context where e-commerce solutions are experiencing high demand and require efficient and scalable management, the development of a platform based on a microservices architecture is essential. This project aims to create a high-performance and scalable e-commerce platform that enables users to browse, purchase products, and manage their orders optimally.

The objective of this project is to design and develop a modular e-commerce platform using Spring Cloud for microservices architecture, ensuring scalability, secure management of users, payments, and products.

The developed solution will allow:

* User management : account creation, role management, and secure authentication.
* Product management : detailed product display, catalog management, category-based search.
* Order management : adding to cart, order validation, and status tracking.
* Payment management : integration of secure payment systems and invoice generation.
* Providing statistics : an administrator dashboard with sales and activity reports.

The project will be carried out by a team of three people over a duration of three months, with the goal of delivering a functional and maintainable application using a microservices-based architecture.

Keywords: E-commerce, Spring Boot, microservices architecture, user management, product management, order management, online payments, Spring Cloud.

***Table de matière***

[*Résumé* 3](#_Toc190792352)

[*Abstract* 4](#_Toc190792353)

[Liste des figures 7](#_Toc190792354)

[Introduction Générale 8](#_Toc190792355)

[1. Contexte du projet 8](#_Toc190792356)

[2. Objectifs 8](#_Toc190792357)

[4. Structure du rapport 10](#_Toc190792358)

[Chapitre 1 : *État de l'art* 11](#_Toc190792359)

[1. Besoins/Problématiques 12](#_Toc190792360)

[2. Systèmes Similaires et Thématiques 12](#_Toc190792361)

[2.1 Systèmes Similaires 12](#_Toc190792362)

[2.2 Thématiques Liées 13](#_Toc190792363)

[3. Méthodes et Technologies Utilisées 13](#_Toc190792364)

[3.1 Méthodes Utilisées 13](#_Toc190792365)

[3.2 Technologies Utilisées 13](#_Toc190792366)

[4. Comparaison des Technologies 14](#_Toc190792367)

[4.1 Spring Boot vs Node.js 14](#_Toc190792368)

[4.2 React vs Angular 14](#_Toc190792369)

[5. Synthèse 15](#_Toc190792370)

[Chapitre 2 : *Description fonctionnelle détaillée* 16](#_Toc190792371)

[1.Objectifs Approfondis : 17](#_Toc190792372)

[2.Utilisateur : 17](#_Toc190792373)

[3.Gestion de l'équipe : 18](#_Toc190792374)

[4.Outils de gestion 18](#_Toc190792375)

[Chapitre 3 : Conception du Système 22](#_Toc190792376)

[1. Architecture générale : 23](#_Toc190792377)

[2. Modèle de données : 23](#_Toc190792378)

[3. Diagrammes UML 23](#_Toc190792379)

[Chapitre 4: Développement de l'Application 25](#_Toc190792380)

[1. Outils Utilisés 26](#_Toc190792381)

[2. Développement du Back-End 27](#_Toc190792382)

[2.1. Implémentation des fonctionnalités 27](#_Toc190792383)

[2.2 Tests API avec Postman 27](#_Toc190792384)

[3. Développement du Front-End 27](#_Toc190792385)

[3.1. Implémentation de l'interface utilisateur 27](#_Toc190792386)

[4. Gestion du Code Source 28](#_Toc190792387)

[4.1 GitHub pour le Backend 28](#_Toc190792388)

[4.2 GitHub pour le Frontend 28](#_Toc190792389)

[Chapitre 5: Démonstration de l'interface utilisateur. 30](#_Toc190792390)

[Conclusion générale 32](#_Toc190792391)

[Références Bibliographiques 33](#_Toc190792392)

**Liste des figures**

Figure 1 : diagramme de Gantt

Figure 2 :Diagramme de classes

Figure 3 : Diagramme de cas d’utilisation globale

# Introduction Générale

## 1. Contexte du projet

L'e-commerce a connu une croissance significative au cours des dernières années, transformant le commerce traditionnel en une activité accessible à toute heure et en tout lieu. Cette évolution est soutenue par la nécessité pour les entreprises de s'adapter aux attentes des consommateurs, qui exigent des plateformes rapides, sécurisées et personnalisées. Cependant, la gestion de ces plateformes complexes nécessite une infrastructure capable de s'adapter rapidement aux changements du marché, tout en offrant une haute disponibilité et une performance optimale.

Dans ce contexte, l'architecture microservices est apparue comme une solution idéale pour décomposer des applications monolithiques complexes en services indépendants, chacun responsable d'une fonctionnalité spécifique. L’utilisation de cette approche permet de déployer, maintenir et mettre à jour chaque composant sans affecter le reste de l’application, offrant ainsi flexibilité et scalabilité.

Ce projet consiste en la création d’une **plateforme e-commerce basée sur microservices**. L'objectif est de concevoir une application modulaire et extensible qui pourra répondre aux besoins des commerçants en ligne tout en garantissant une expérience fluide et sécurisée pour les utilisateurs.

## 2. Objectifs

Les objectifs du projet sont les suivants :

* **Conception d'une plateforme e-commerce moderne :** Développer une solution permettant aux utilisateurs de rechercher, acheter, et gérer leurs commandes de manière fluide, tout en permettant aux administrateurs de gérer les produits et les utilisateurs.
* **Architecture basée sur des microservices :** Mettre en place une architecture de type microservices permettant de découper le système en plusieurs services indépendants, chacun ayant une responsabilité bien définie (gestion des produits, gestion des utilisateurs, gestion des paiements, etc.).
* **Scalabilité et performances :** Assurer que la plateforme puisse supporter une forte charge de trafic et de transactions, avec une architecture flexible et scalable, permettant une mise à jour et une extension aisée des fonctionnalités.
* **Sécurité :** Garantir la sécurité des données sensibles, comme les informations des utilisateurs et les transactions, à travers des mécanismes de cryptage et une gestion rigoureuse des accès.
* **Intégration continue et tests :** Mettre en place des pratiques de développement modernes telles que l’intégration continue (CI), les tests automatisés et la validation des fonctionnalités tout au long du cycle de développement.
* **3. Méthodologie**

Le développement de la plateforme suivra une **méthodologie agile** avec une approche itérative et incrémentale, ce qui permet de s’adapter rapidement aux retours des utilisateurs et aux évolutions des besoins du projet. Les principales étapes du processus sont les suivantes :

* **Phase de planification et conception :** Identification des besoins des utilisateurs, définition des fonctionnalités de la plateforme, choix des technologies, et création de l’architecture générale du système.
* **Phase de développement :** Développement des microservices et du frontend avec intégration continue des fonctionnalités à chaque étape. Chaque microservice sera testé indépendamment et intégré dans un environnement de test.
* **Phase de validation et tests :** Tests unitaires, tests d’intégration, et validation par les utilisateurs pour s’assurer que la plateforme répond aux attentes et aux exigences de sécurité.
* **Phase de déploiement :** Mise en production de la plateforme, avec une surveillance continue des performances et de la sécurité.

La gestion de projet sera facilitée par des outils de gestion de projet comme **Jira** pour le suivi des tâches et des sprints, **GitHub** pour la gestion du code source, et des outils de communication comme **Slack** pour la coordination de l’équipe.

## 4. Structure du rapport

Le rapport est structuré de la manière suivante :

* **Chapitre 1 : Introduction Générale**
  + Contexte du projet
  + Objectifs
  + Méthodologie
  + Structure du rapport
* **Chapitre 2 : Analyse des besoins**
  + Identification des besoins des utilisateurs finaux et des administrateurs
  + Analyse des fonctionnalités de la plateforme
* **Chapitre 3 : Conception du système**
  + Architecture générale du système
  + Choix technologiques
  + Modèle de données et diagrammes UML
* **Chapitre 4 : Développement et mise en œuvre**
  + Développement des microservices
  + Développement de l’interface utilisateur
  + Tests et validation
* **Chapitre 5 : Résultats et évaluation**
  + Performances de la plateforme
  + Sécurité et gestion des erreurs
  + Retour des utilisateurs et recommandations
* **Chapitre 6 : Conclusion**
  + Bilan du projet
  + Perspectives d’évolution de la plateforme

# Chapitre 1 : *État de l'art*

**1**

***État de l'art***

## 1. **Besoins**/Problématiques

Dans le domaine du e-commerce, plusieurs défis doivent être relevés pour garantir une plateforme performante et évolutive :

* **Scalabilité et performance** : Un site e-commerce doit pouvoir gérer un grand nombre de produits, d'utilisateurs et de transactions. L'architecture microservices, qui divise le système en services indépendants, permet d'assurer une montée en charge fluide. Chaque microservice peut être développé, déployé et mis à jour indépendamment, ce qui garantit une flexibilité optimale.
* **Gestion des données et disponibilité** : Les sites de e-commerce traitent des volumes massifs de données sensibles (comme les informations des clients et des paiements). Une architecture microservices avec des outils comme **Spring Cloud** (incluant **Eureka** et **Spring Cloud Gateway**) permet de garantir une gestion optimale de ces données tout en assurant une haute disponibilité du système.
* **Sécurité des transactions** : L'une des préoccupations majeures dans le domaine du e-commerce est la sécurité des informations sensibles des utilisateurs, notamment pour les paiements en ligne. L'intégration de **Spring Security** et de solutions comme **OAuth2** ou **JWT** est nécessaire pour sécuriser l'accès et protéger les données personnelles.

## 2. Systèmes Similaires et Thématiques

### 2.1 Systèmes Similaires

De grandes plateformes comme **Amazon**, **Ebay** ou **Rakuten** ont adopté des architectures microservices pour gérer leurs sites e-commerce à grande échelle. Ces entreprises utilisent des systèmes distribués, modulaire et résilient pour gérer des milliers de produits, des millions d'utilisateurs, et des transactions financières quotidiennes. Elles sont un modèle d'efficacité et de robustesse dans l'implémentation des architectures microservices dans le domaine du commerce en ligne.

### 2.2 Thématiques Liées

Les thématiques suivantes sont pertinentes dans le contexte d'un projet e-commerce en microservices :

* **Sécurisation des données** : Il est crucial d'assurer la protection des données personnelles des utilisateurs, ainsi que des informations relatives aux paiements. L'usage de technologies comme **Spring Security**, **OAuth2** et **JWT** sont indispensables pour garantir cette sécurité.
* **Optimisation de l'expérience utilisateur (UX)** : Pour offrir une navigation fluide et intuitive, l'utilisation de **React** comme framework frontend permettra de créer une interface dynamique et réactive, adaptée aux besoins des utilisateurs tout en garantissant une expérience cohérente sur le site.
* **Omnicanal** : La plateforme doit être accessible non seulement via un site web, mais aussi via des applications mobiles, tout en maintenant une cohérence dans les données et les transactions entre ces différents canaux. L'architecture microservices permet cette flexibilité avec des API REST qui peuvent être utilisées aussi bien par un site web qu'une application mobile.

## 3. Méthodes et Technologies Utilisées

### 3.1 Méthodes Utilisées

Le projet s'appuie sur des méthodes éprouvées pour garantir un développement rapide et efficace :

* **Méthode agile** : En utilisant des approches comme **Scrum**, le projet peut évoluer de manière itérative et incrémentale, permettant de répondre rapidement aux besoins des utilisateurs et d'adapter le développement au fil du temps.

### 3.2 Technologies Utilisées

* **Backend** : **Spring Boot** et **Spring Cloud** : Pour créer des microservices robustes et modulaires. **Spring Cloud** permet de gérer la communication entre les services via des outils comme **Eureka** (pour la découverte des services) et **Spring Cloud Gateway** (pour le routage des requêtes).
* **Frontend** : **React** : Framework JavaScript utilisé pour le développement du frontend du site e-commerce. React permet de créer des interfaces utilisateurs dynamiques, réactives et modulaires. Il sera utilisé pour gérer les composants, les états et la navigation du site.
* **Base de données : MongoDB :** Pour gérer les données transactionnelles, telles que les informations sur les utilisateurs et les commandes. MongoDB est une base de données NoSQL qui permet une gestion flexible et évolutive des données, idéale pour les structures de données non relationnelles.
* **API Gateway** : Un **API Gateway** tel que **Spring Cloud Gateway** permet de centraliser l'accès à l'ensemble des microservices et de gérer la répartition des requêtes.
* **Sécurité** : **Spring Security** est utilisé pour la gestion de l'authentification et des autorisations, tandis que **OAuth2** et **JWT** sont utilisés pour la gestion sécurisée des sessions et des tokens.

## 4. Comparaison des Technologies

### 4.1 Spring Boot vs Node.js

* **Spring Boot** est une technologie mature, particulièrement bien adaptée aux environnements d'entreprise et aux applications nécessitant une forte sécurité et une gestion des transactions complexes, ce qui est essentiel pour un site e-commerce.
* **Node.js**, bien qu’il offre de bonnes performances pour les applications légères et les traitements en temps réel, n'est pas aussi adapté pour des applications complexes en raison de ses limitations dans le traitement des transactions critiques.

### 4.2 React vs Angular

* **React** développé par Facebook, est une bibliothèque plus légère et flexible, centrée sur la création d’interfaces utilisateur. Bien que plus facile à prendre en main, elle permet une grande liberté dans la gestion du projet. React est particulièrement adapté pour les sites e-commerce grâce à sa capacité à offrir des interfaces interactives et réactives. En utilisant des bibliothèques supplémentaires comme Redux pour la gestion d'état et React Router pour le routage, React peut parfaitement répondre aux besoins d'un site e-commerce nécessitant des performances élevées et une expérience utilisateur fluide et dynamique.
* **Angular** est un framework complet développé par Google, idéal pour les projets de grande envergure nécessitant une structure bien définie. Il offre une intégration native de fonctionnalités comme la gestion des formulaires, le routage et l’injection de dépendances. Sa courbe d’apprentissage est plus longue, mais il est plus adapté pour des applications complexes avec une architecture stricte.

## 5. ****Synthèse****

L'architecture microservices, combinée avec des technologies comme Spring Boot, React, Spring Security et des outils de gestion de conteneurs comme Docker et Kubernetes, permet de créer une plateforme e-commerce performante, évolutive et sécurisée. L'utilisation de Spring Cloud pour la gestion des microservices et des API, ainsi que de MySQL et MongoDB pour les bases de données, garantit une solution robuste et flexible, capable de répondre aux besoins actuels et futurs des utilisateurs.

Le choix de React pour le frontend permettra de créer une interface utilisateur moderne, réactive et facile à maintenir, contribuant ainsi à une expérience d'achat fluide et agréable pour les utilisateurs.

# Chapitre 2 : *Description fonctionnelle détaillée*

**2**

***Description fonctionnelle détaillée***

## 1.Objectifs Approfondis :

L’objectif principal de notre projet est de fournir une plateforme e-commerce fonctionnelle et intuitive, permettant aux utilisateurs de rechercher, acheter et gérer leurs commandes de manière fluide. Voici les objectifs approfondis :

**Simplifier l’expérience utilisateur :**

Offrir une interface conviviale et accessible qui permet aux clients de naviguer facilement dans le catalogue de produits.

Intégrer une fonctionnalité de recherche avancée pour trouver rapidement des produits.

**Améliorer l’efficacité de la gestion des commandes :**

Mettre en place un module robuste pour le suivi des commandes en temps réel.

Fournir des notifications automatiques (e-mails) pour informer les clients des statuts de leurs commandes (confirmation, expédition, livraison).

**Renforcer la sécurité et la fiabilité :**

Assurer une authentification sécurisée des utilisateurs grâce à JWT (JSON Web Tokens).

Intégrer des solutions de paiement fiables et sécurisées (PayPal, Stripe) pour protéger les transactions.

**Optimiser l’administration :**

Offrir aux administrateurs des outils pour gérer efficacement les produits, utilisateurs et commandes via une interface dédiée.

Générer des rapports de suivi des ventes pour aider à la prise de décision stratégique.

## 2.Utilisateur :

Dans notre projet, nous avons défini deux principales catégories d’utilisateurs avec des besoins spécifiques :

**Clients :**

Les clients sont les principaux utilisateurs de la plateforme. Ils peuvent :

* Créer un compte pour accéder à leurs commandes et historiques.
* Parcourir le catalogue de produits et effectuer des recherches.
* Ajouter des produits à leur panier, passer des commandes et effectuer des paiements.
* Recevoir des notifications par e-mail sur leurs commandes.

**Administrateurs :**

Les administrateurs ont accès à un tableau de bord pour gérer les aspects suivants :

* Gestion des utilisateurs (création, modification, suppression).
* Gestion des produits (ajout de nouveaux articles, mise à jour des descriptions et des prix).
* Suivi des commandes passées par les clients et résolution des problèmes éventuels.

## 3.Gestion de l'équipe :

La réalisation de ce projet a été le fruit d’une collaboration étroite entre les trois membres de l’équipe, à savoir Loubna Habil, Oumaima El-Mansouri et ABDesamie Mejjadi. Nous avons adopté une approche collective, où chacun a contribué activement et de manière équivalente au développement de l’application, en travaillant aussi bien sur le backend avec Spring Boot que sur le frontend avec React.

Dans le cadre du backend, nous avons développé les microservices nécessaires pour gérer les utilisateurs, les produits et les commandes, tout en assurant une intégration fluide avec la base de données MongoDB. Parallèlement, sur le frontend, nous avons créé des interfaces utilisateur dynamiques et intuitives, permettant une expérience utilisateur fluide et efficace.

Cette collaboration uniforme a permis de garantir une cohérence dans le développement, de surmonter les éventuels défis techniques, et de livrer une application e-commerce fonctionnelle et adaptée aux besoins identifiés. Nous avons veillé à partager nos idées et à tester ensemble les fonctionnalités afin de maintenir un haut niveau de qualité tout au long du projet.

## 4.Outils de gestion

Pour mener à bien ce projet, nous avons utilisé différents outils afin de structurer, suivre et organiser efficacement les étapes de développement. Ces outils ont permis une coordination optimale et une gestion fluide de l’ensemble des tâches.

Diagramme de Gantt :

Pour planifier et suivre les différentes phases du projet, nous avons élaboré un diagramme de Gantt en utilisant un outil en ligne spécialisé. Cet outil nous a permis de :

* + Définir les différentes étapes et les jalons du projet.
  + Estimer la durée des tâches et leurs dépendances.
  + Visualiser l’avancement global du projet à travers un calendrier interactif.

Le diagramme de Gantt a été un outil essentiel pour structurer le projet, en identifiant les étapes clés et en respectant les délais. **La Figure 1** illustre le diagramme de Gantt que nous avons élaboré, où chaque phase est clairement représentée avec sa durée prévue et ses interdépendances.

**Git :**

Pour la gestion du code source et la collaboration, nous avons utilisé **Git** via une plateforme en ligne (GitHub). Cet outil a joué un rôle crucial dans :

* Le suivi des modifications apportées au code.
* La gestion des versions et des branches, permettant un développement simultané sur différentes parties du projet.
* La collaboration efficace entre les membres de l’équipe, grâce à des fonctionnalités comme les pull requests et les commits.

La **Figure 2** présente l’organisation de notre dépôt GitHub, avec les différentes branches et l’historique des contributions. Grâce à cet outil, nous avons pu centraliser le code et éviter les conflits lors des modifications simultanées.

**5.Gantt :**

Pour assurer une gestion efficace du projet, nous avons élaboré un diagramme de Gantt

détaillé. Ce diagramme présente les différentes phases du projet, ainsi que les tâches associées

à chacune de ces phases, leur durée estimée et les dépendances entre elles. Il permet de

visualiser clairement le calendrier de réalisation du projet et d'identifier les étapes clés, les

jalons et les éventuels chevauchements de tâches. Le diagramme de Gantt constitue un outil

essentiel pour planifier, suivre et coordonner les activités du projet tout au long de son

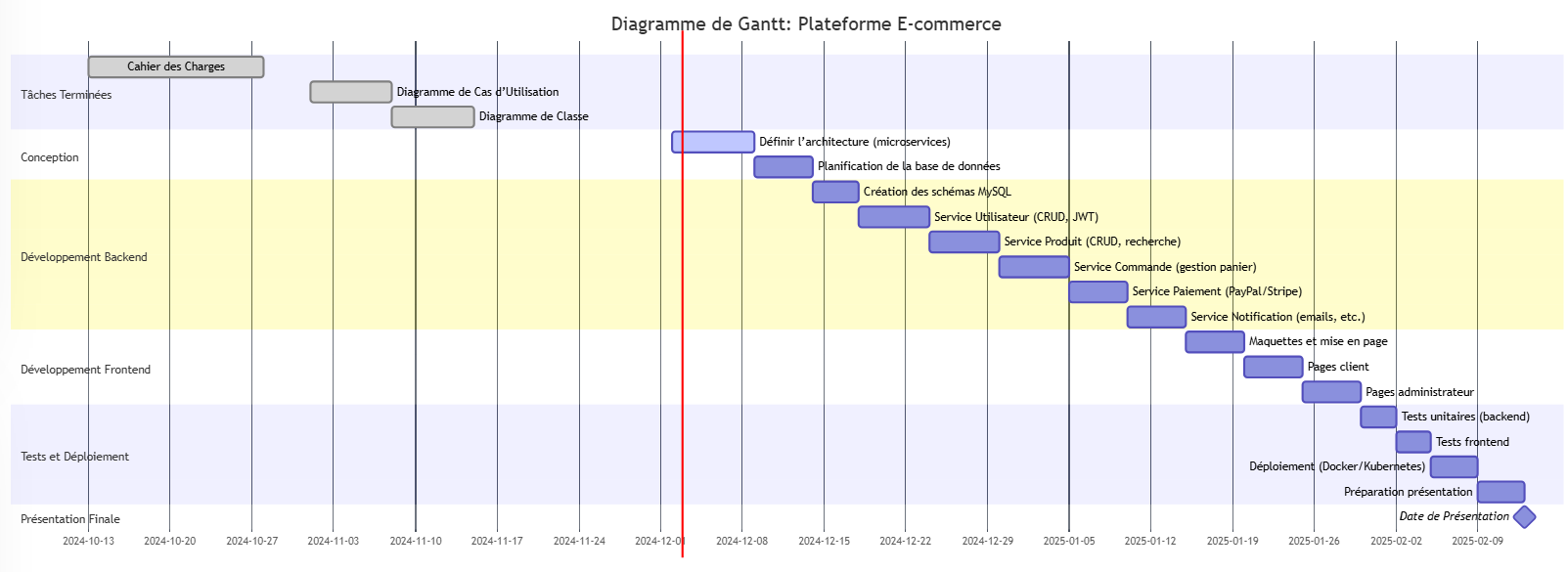
déroulement.

Figure 1 : diagramme de Gantt

Le diagramme de Gantt présenté ci-dessus illustre les différentes étapes et phases du projet de développement de notre plateforme e-commerce. Chaque phase est représentée sous forme de barre horizontale, permettant de visualiser clairement la progression du projet, les jalons clés et les tâches déjà réalisées. Voici une description détaillée des sections principales :

**Tâches terminées :**

Cette section couvre les tâches initiales, notamment :

**Cahier des charges :** Définition des besoins fonctionnels et non fonctionnels du projet.

**Diagramme de cas d'utilisation :** Identification des interactions entre les utilisateurs et le système.

**Diagramme de classes :** Modélisation des entités principales du projet et leurs relations.

**Conception :**

Dans cette phase, nous avons planifié les aspects techniques du projet, incluant :

Définir l'architecture (microservices) : Spécification de l'architecture logicielle pour diviser le projet en modules indépendants.

Planification de la base de données : Élaboration des schémas MongoDB pour la persistance des données.

**Développement Backend :**

Cette phase marque le cœur de l’implémentation du backend, avec :

Création des schémas MongoDB : Conception et implémentation des bases de données relationnelles.

Service Utilisateur (CRUD, JWT) : Gestion des utilisateurs avec des opérations CRUD (Créer, Lire, Mettre à jour, Supprimer) et l'intégration de l'authentification via JWT.

Service Produit (CRUD, recherche) : Implémentation des fonctionnalités pour gérer les produits et la recherche.

Service Commande (gestion panier) : Développement du module permettant aux utilisateurs de gérer leurs commandes et paniers.

Service Paiement (PayPal/Stripe) : Intégration de solutions de paiement sécurisées.

Service Notification (emails, etc.) : Configuration d’un service d’envoi d’e-mails pour les confirmations et notifications.

**Développement Frontend :**

Cette phase concerne l’interface utilisateur :

Maquettes et mise en page : Création de designs pour offrir une expérience utilisateur intuitive.

Pages client et administrateur : Développement des interfaces spécifiques pour les clients et les administrateurs.

**Tests et Déploiement :**

Dans cette étape, nous avons effectué des tests pour garantir la qualité et la stabilité du système :

Tests unitaires (backend) : Validation des services backend pour assurer leur bon fonctionnement.

Tests frontend : Vérification des fonctionnalités côté client.

Déploiement (Docker/Kubernetes) : Préparation du déploiement en environnement de production.

**Présentation finale :**

La dernière étape est consacrée à la préparation pour la démonstration finale et la remise du projet.

Ce diagramme de Gantt provisoire a été un outil essentiel pour structurer les phases du projet et coordonner efficacement les tâches. Il nous a permis de suivre l'évolution du projet tout en identifiant les éventuelles dépendances ou retards.

# Chapitre 3 : Conception du Système

**3**

**Conception du Système**

## 1. Architecture générale :

L'architecture de l'application e-commerce est fondée sur un modèle client-serveur et se décompose en plusieurs couches :

* **Couche de présentation** : Responsables de l'interface utilisateur, les composants de cette couche incluent les pages web et les fonctionnalités interactives accessibles via un navigateur ou une application mobile.
* **Couche logique** : Cette couche gère les processus métiers, le traitement des données et la logique applicative. Elle comprend des services web et des API qui connectent la présentation à la base de données.
* **Couche de données** : Représente la base de données où sont stockées toutes les informations de l'application, telles que les produits, les utilisateurs, les commandes et les paiements.

## 2. Modèle de données :

Le modèle de données repose sur une base relationnelle optimisée pour gérer les informations e-commerce. Les principales entités et leurs relations sont décrites comme suit :

* **Utilisateur** : Contient les informations des clients et des administrateurs (nom, email, mot de passe, adresse, etc.).
* **Produit** : Définit les détails des produits (nom, description, prix, stock, etc.).
* **Commande** : Gère les transactions passées par les utilisateurs (id commande, utilisateur, produits, total, statut, etc.).
* **Paiement** : Enregistre les détails des paiements (montant, date, mode de paiement, statut).
* **Catégorie** : Classe les produits dans des groupes logiques.

Les relations entre ces entités sont modélisées sous forme de diagramme entité-relation (ERD).

## 3. Diagrammes UML

**Diagramme de classes :**

**A diagram of a computer

Description automatically generated with medium confidence**

Figure 2:Diagramme de classe

Le diagramme de classes illustre la structure conceptuelle de l'application e-commerce. Il comprend des entités principales telles que Utilisateur, Produit, Commande, Paiement, et leurs relations. Les utilisateurs sont divisés en Client et Administrateur, chacun ayant des rôles spécifiques dans le système. Les classes comme Panier, Commande, et Produit assurent une gestion cohérente des processus commerciaux. Les concepts d'autorisation sont intégrés via les classes Rôle et Permission, garantissant une gestion sécurisée des accès.

**Diagramme de cas d'utilisation :**

**A diagram of a person with text

Description automatically generated**

Figure 3 : Diagramme de cas d’utilisation globale

Le diagramme de cas d'utilisation présente les interactions entre les utilisateurs et le système e-commerce. Les utilisateurs principaux incluent Visiteur, Client, et Administrateur. Le Visiteur peut consulter les produits, se connecter ou créer un compte, ce qui lui permet de devenir Client. Les Clients peuvent ajouter des produits au panier, passer une commande et voir l'historique des commandes. L'Administrateur, quant à lui, a des privilèges étendus, comme gérer les produits, les commandes et les utilisateurs. Ce diagramme montre également des relations comme « include » et « extend », qui reflètent des dépendances entre différentes actions, offrant ainsi une vue complète des fonctionnalités disponibles et des rôles associés.

# Chapitre 4: Développement de l'Application

**4**

**Développement de l'Application**

Introduction

Dans ce chapitre, nous détaillons les étapes de développement de la plateforme e-commerce, en mettant l’accent sur les choix techniques, l’implémentation des microservices, le développement de l’interface utilisateur et la gestion du code source. L’objectif principal est de démontrer comment les outils et technologies présentés dans les chapitres précédents ont été concrètement mis en œuvre pour réaliser une application fonctionnelle, robuste et scalable.

## 1. Outils Utilisés

Pour la réalisation du projet, plusieurs outils ont été employés pour faciliter le développement, la gestion de projet, et les tests :

* **GanttProject** : Un outil de gestion de projet utilisé pour planifier et suivre les différentes étapes du développement. Il permet de visualiser les délais, les jalons, et la répartition des tâches.
* **Postman** : Un outil de test d'API utilisé pour simuler et vérifier les requêtes et réponses HTTP, assurant le bon fonctionnement des services RESTful développés dans l'application.
* **MongoDB** : Base de données relationnelle utilisée pour stocker les informations clés du projet, telles que les utilisateurs, les réservations, et les places de parking. Elle permet une gestion efficace et sécurisée des données.
* **IntelliJ IDEA** : Un environnement de développement intégré (IDE) pour le développement du backend en Java avec Spring Boot. IntelliJ fournit des outils avancés pour la gestion du code, la navigation, et le débogage.
* **Visual Studio Code** : Utilisé pour le développement du front-end avec React. Cet éditeur léger et flexible offre de nombreuses extensions pour faciliter le développement en TypeScript, HTML et CSS.
* **GitHub** : Outil de gestion de version pour suivre les modifications du code et collaborer avec d'autres développeurs. Il a été utilisé à la fois pour le backend et le frontend, avec des branches dédiées pour le développement, les tests et la production.

## 2. Développement du Back-End

### 2.1. Implémentation des fonctionnalités

Nous avons découpé le backend en plusieurs microservices, chacun dédié à une fonctionnalité précise :

* **Service Utilisateur** : Implémentation des opérations CRUD pour la gestion des comptes, intégration de l’authentification sécurisée via JWT et gestion des rôles.
* **Service Produit** : Gestion du catalogue (ajout, modification, suppression et recherche de produits) en utilisant Spring Boot et en exposant des endpoints REST pour le frontend.
* **Service Commande** : Mise en place du processus de commande, incluant la gestion du panier, la validation des commandes et le suivi des statuts.
* **Service Paiement** : Intégration de solutions de paiement sécurisées (PayPal, Stripe) permettant le traitement des transactions et la génération automatique des factures.
* **Service Notification** : Configuration d’un module d’envoi d’e-mails pour les confirmations de commande et autres notifications importantes.

### 2.2 Tests API avec Postman

Chaque micro-service a été rigoureusement testé à l’aide de Postman. Nous avons créé des collections de requêtes pour vérifier le bon fonctionnement des endpoints, tester les cas de succès et d’erreur, et s’assurer que les échanges en JSON respectaient les standards définis. Ces tests ont permis d’identifier rapidement des dysfonctionnements et d’améliorer la robustesse de l’application.

## 3. Développement du Front-End

### 3.1. Implémentation de l'interface utilisateur

Le développement du frontend a été réalisé en React afin de créer une interface dynamique et réactive. Les principales fonctionnalités développées sont :

* Accueil et Catalogue : Une page d’accueil présentant les produits phares et un catalogue détaillé accessible via une navigation intuitive.
* Détail du produit : Une interface permettant d’afficher toutes les informations d’un produit, avec des options pour ajouter au panier et consulter des avis.
* Panier et Commande : Un module de gestion du panier intégré avec le système de commande, offrant une expérience d’achat fluide et sécurisée.
* Interface administrateur : Un tableau de bord dédié aux administrateurs pour la gestion des utilisateurs, des produits et le suivi des commandes.

Les services React communiquent avec le backend via des requêtes HTTP, utilisant des bibliothèques comme Axios ou Fetch, garantissant une synchronisation en temps réel des données et une expérience utilisateur homogène.

## 4. Gestion du Code Source

## 4.1 GitHub pour le Backend

Le dépôt GitHub dédié au backend contient l’ensemble du code développé avec Spring Boot et Spring Cloud. Voici quelques points clés de son organisation :

* Structure des Branches :
  + La branche principale (main/master) héberge la version stable et validée.
  + La branche de développement (develop) intègre les nouvelles fonctionnalités avant leur fusion dans la branche principale.
  + Des branches de fonctionnalités (feature branches) sont créées pour travailler sur des modules spécifiques sans perturber la stabilité globale.
* Processus de Pull Request et Revue de Code :
  + Chaque contribution est soumise via un pull request.
  + Les revues de code par les membres de l’équipe permettent de garantir la qualité et la cohérence du code.
* Intégration Continue (CI) :
  + Des outils comme GitHub Actions sont utilisés pour lancer automatiquement des tests unitaires et d’intégration lors de la fusion des branches, minimisant ainsi les risques de régression.

### 4.2 GitHub pour le Frontend

Le dépôt GitHub dédié au frontend contient le code source de l’interface utilisateur développée en Angular. L’organisation suit des principes similaires, adaptés aux spécificités du développement web :

* Organisation des Branches :
  + La branche principale (main/master) contient la version stable destinée au déploiement.
  + La branche de développement (develop) regroupe les modifications en cours d’intégration.
  + Des branches de fonctionnalités (feature branches) permettent de travailler sur des composants individuels (pages, widgets, etc.) de manière isolée.
* Pull Requests et Revues de Code :
  + Chaque modification est soumise via un pull request pour être validée par d’autres développeurs, garantissant ainsi le respect des bonnes pratiques et la qualité du code.
* Automatisation et Tests :
  + L’utilisation d’outils d’intégration continue assure la compilation correcte, l’exécution des tests unitaires et la génération de builds optimisés pour le déploiement en production.

#### Conclusion

La phase de développement a permis de concrétiser l’architecture microservices et de créer une application e-commerce robuste et évolutive. Grâce à l’organisation rigoureuse du code source sur GitHub, une collaboration efficace a été assurée entre les membres de l’équipe, tant pour le backend que pour le frontend. L’intégration continue et les tests automatisés garantissent la qualité du produit final, répondant ainsi aux exigences de performance et de sécurité du projet.

# Chapitre 5: Démonstration de l'interface utilisateur.

**5**

**Démonstration de l'interface utilisateur**

Introduction

**Conclusion**

# Conclusion générale

Ce projet de développement d’une plateforme e-commerce basée sur une architecture microservices a permis de répondre à des besoins actuels en matière de scalabilité, de sécurité et de performance. Les principaux objectifs ont été atteints :

• La conception d’une solution modulaire et extensible, permettant une gestion optimisée des utilisateurs, des produits, des commandes et des paiements.  
• La mise en œuvre d’une architecture microservices facilitant le déploiement, la maintenance et l’évolution de l’application.  
• L’intégration d’une interface utilisateur moderne et intuitive, développée avec Angular, offrant une expérience d’achat fluide.  
• La réalisation d’une solution sécurisée, grâce à l’utilisation de Spring Security, JWT et d’API sécurisées, garantissant la protection des données sensibles.

Le développement a été réalisé en suivant une méthodologie agile, favorisant la collaboration, les tests réguliers et l’amélioration continue. Malgré certains défis techniques, notamment dans l’intégration des différents services et la gestion des communications entre eux, l’équipe a su mettre en place des solutions efficaces et évolutives.

En perspective, plusieurs axes d’amélioration pourront être envisagés : – L’optimisation des performances pour gérer un trafic encore plus important.  
– Le déploiement d’une application mobile native pour étendre l’accessibilité de la plateforme.  
– L’intégration d’outils d’analyse avancée pour mieux comprendre le comportement des utilisateurs et affiner l’offre.

Enfin, ce projet a été l’occasion de mettre en pratique des compétences techniques et de gestion de projet, tout en répondant aux exigences d’un environnement e-commerce moderne et compétitif.

# Références Bibliographiques

|  |  |  |
| --- | --- | --- |
| [1] |  | **Spring Boot Reference Documentation** – Documentation officielle de Spring Boot, disponible sur [spring.io](https://spring.io/projects/spring-boot). |
| [2] |  | **React Documentation** – Documentation complète de React, consultable sur [reactjs.org](https://reactjs.org" \t "_new). |
| [3] |  | **Microservices Patterns: With examples in Java** – Livre de Chris Richardson, présentant des schémas et bonnes pratiques pour les architectures microservices. |
| [4] |  | **Spring Cloud Documentation** – Ressources et guides sur la mise en œuvre des microservices avec Spring Cloud, disponibles sur [spring.io](https://spring.io/projects/spring-cloud). |
| [5] |  | **RESTful Web Services** – Manuel et références pour la conception d’API REST, par Leonard Richardson et Sam Ruby |