



# Rapport du projet de fin de semestre Génie Informatique

**Conception et Développement d'une  
Application Web  
E-Commerce Ai Powered.**

**SMart.**

---

Realisé par :

MEFTAH Ahmed Reda  
OURAKH Ismail

Jury :  
ATLAS Abdelghafour  
BEKKARI Aissam

# Resumé

Dans le cadre de ma formation en Génie Informatique à l'*École Nationale des Sciences Appliquées*, j'ai eu l'opportunité de participer à un projet collectif réalisé sur une durée de 15 jours. L'objectif de ce projet était de concevoir et développer une application e-commerce innovante, baptisée **SMart**, qui intègre des fonctionnalités alimentées par l'intelligence artificielle pour améliorer l'expérience utilisateur.

**SMart** se distingue par un système de recommandations personnalisées, capable de proposer des produits en fonction des préférences des utilisateurs, ainsi qu'une gestion optimisée des produits et des commandes. Pour répondre aux besoins du projet, nous avons choisi des technologies modernes et performantes :

- React pour développer une interface utilisateur dynamique, interactive et responsive.
- Tailwind CSS pour concevoir un design moderne et adapté à tous les types d'appareils.
- Spring Boot pour assurer un backend robuste et évolutif.
- MongoDB pour gérer les données de manière efficace, notamment grâce à sa flexibilité pour les données structurées et non structurées.

Le projet a été réalisé en équipe de deux membres, avec une répartition claire des tâches pour garantir une progression fluide malgré le délai serré. Mon rôle a inclus la conception et l'implémentation de l'interface utilisateur, ainsi que l'intégration des fonctionnalités du backend avec le système de recommandations basé sur l'intelligence artificielle. Nous avons mis en œuvre un algorithme permettant d'analyser les données utilisateurs afin de personnaliser les suggestions de produits.

Cette expérience m'a permis de renforcer mes compétences en développement web full-stack et de maîtriser l'utilisation combinée de frameworks modernes. Elle m'a également appris à collaborer efficacement en équipe, à respecter des délais stricts et à proposer des solutions innovantes pour répondre à des besoins spécifiques.

# Abstract

As part of my training in Computer Engineering at the *École Nationale des Sciences Appliquées*, I had the opportunity to participate in a collective project carried out over a period of 15 days. The objective of this project was to design and develop an innovative e-commerce application, called **SMart**, which integrates features powered by artificial intelligence to improve the user experience.

**SMart** stands out for its personalized recommendation system, capable of offering products based on user preferences, as well as optimized product and order management. To meet the needs of the project, we chose modern and efficient technologies:

- React to develop a dynamic, interactive and responsive user interface.
- Tailwind CSS to design a modern design adapted to all types of devices.
- Spring Boot to ensure a robust and scalable backend.
- MongoDB to manage data efficiently, particularly thanks to its flexibility for structured and unstructured data.

The project was carried out as a team of two members, with a clear division of tasks to ensure smooth progress despite the tight deadline. My role included the design and implementation of the user interface, as well as the integration of backend features with the AI-based recommendation system. We implemented an algorithm to analyze user data in order to personalize product suggestions.

This experience allowed me to strengthen my skills in full-stack web development and master the combined use of modern frameworks. It also taught me how to collaborate effectively in a team, meet strict deadlines and propose innovative solutions to meet specific needs.

# Remerciements

Au terme de ce projet de fin d'année, nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à sa réussite.

Nous adressons nos remerciements les plus sincères à notre encadrant, **Dr. Aissam BEKKARI**, pour son accompagnement constant, sa disponibilité et ses précieux conseils. Son expertise et ses orientations ont été essentielles pour mener à bien ce projet.

Nous souhaitons également témoigner notre reconnaissance à l'ensemble du corps professoral de l'ENSA de Marrakech, dont les enseignements et le suivi attentif nous ont permis d'acquérir les connaissances et compétences nécessaires à la réalisation de ce travail.

Par ailleurs, nous exprimons notre gratitude à **Dr. Abdelghafour ATLAS** pour son expertise et ses conseils techniques éclairés, qui ont enrichi la qualité de notre projet.

Enfin, nous tenons à remercier nos familles respectives pour leur soutien moral et leur encouragement indéfectible tout au long de nos études. Leur confiance et leur appui ont été une source de motivation essentielle.

Ce projet est le fruit d'un véritable travail collectif et de la contribution de nombreuses personnes à différents niveaux. Nous leur en sommes profondément reconnaissants.

# Table of Contents

Resumé.....	2
Abstract.....	3
Remerciements.....	4
Introduction Generale.....	7
I. Presentation générale du projet.....	8
I.1. Introduction :.....	9
I.2. Étude de l'existant:.....	9
I.3. Objectifs du projet.....	9
I.4. Solutions proposées.....	10
I.5. Méthodologie de développement.....	10
I.6. Conclusion.....	10
II. Analyse de besoins et conception de l'application UrbanVoyage.....	11
II.1. Introduction.....	12
II.2. Analyse du besoin.....	12
a. Les besoins fonctionnels.....	12
b. Les besoins non fonctionnels.....	12
II.3. Conception UML.....	13
a. Modélisation du système.....	13
b. Présentation UML.....	13
c. Spécification des exigences pour SMart.....	13
II.4. Diagrammes UML : .....	14
a. Diagramme de cas d'utilisation.....	14
b. Diagramme de classes:.....	15
c. Diagramme de séquence.....	16
d. Diagramme d'activités.....	17
II.5. Méthodologies et gestion de projet.....	18
Cycle de développement adapté.....	18
Avantages de cette approche pour SMart.....	18
Priorisation des fonctionnalités.....	18
Planning de développement.....	18
Diagramme de GANTT.....	19
Le diagramme montre la chronologie complète du projet avec :.....	19
II.6. Conclusion.....	20
III. Etude technique.....	21
III.1. Introduction:.....	22
III.2. Choix technologiques.....	22
Front-end.....	22
a. React.....	22
b. Tailwind CSS.....	23
c. Librairies de gestion d'état et requêtes.....	23
d. Composants et animations.....	23
e. Paiement et intégration.....	24
Back-end.....	25
a. Spring Boot.....	25
b. Spring Security & JWT.....	25
c. Stripe.....	26
e. REST Template.....	26
Base de données.....	27
a. MongoDB.....	27
Entraînement du Modèle de Recommandation.....	28

Word2Vec avec Gensim.....	28
Pandas.....	29
NumPy.....	29
Architecture de systeme d'entrainement.....	30
Integration avec API Flask.....	30
Performance et Optimisation.....	31
III.3. Architecture logicielle.....	31
IV. Realisations & Tests.....	33
IV.1. Realisation.....	34
a. Authentification.....	34
b. Page d'Accueil et À Propos.....	35
c. Catalogue des Produits.....	36
d. Gestion du Panier.....	37
e. Administration des Produits.....	38
f. Navigation et Interface Utilisateur.....	39
IV.2. Tests.....	41
a. Tests Unitaires.....	41
b. Tests d'Intégration.....	41
c. Tests de Performance.....	42
d. Tests de Sécurité.....	42
e. Tests d'Acceptation.....	42
f. Intégration Continue.....	43
IV.3. Conclusion.....	43
V. Conclusion generale et Perspectives.....	44
V.1. Conclusion Générale.....	45
V.2. Perspectives.....	45

# Introduction Générale

Dans un monde de plus en plus connecté, les plateformes d'e-commerce jouent un rôle essentiel dans la simplification des transactions commerciales. C'est dans ce contexte que nous avons développé **SMart**, une application web innovante conçue pour offrir aux clients une expérience d'achat en ligne fluide, intuitive et personnalisée.

L'objectif principal de **SMart** est de fournir un espace digital moderne où les utilisateurs peuvent parcourir, rechercher et acheter des produits en toute simplicité. Ce qui distingue notre plateforme des autres solutions existantes est l'intégration de l'intelligence artificielle, qui constitue le cœur de son fonctionnement. Grâce à des algorithmes avancés, **SMart** est capable d'analyser les comportements et préférences des utilisateurs afin de leur proposer des recommandations personnalisées, adaptées à leurs besoins spécifiques.

En plus d'un moteur de suggestions intelligent, l'application offre une interface utilisateur conviviale et responsive, conçue pour s'adapter à tous les types d'appareils, qu'il s'agisse d'un smartphone, d'une tablette ou d'un ordinateur. L'ergonomie de la plateforme a été soigneusement pensée pour garantir une navigation intuitive, réduisant ainsi les obstacles à l'achat en ligne.

L'intelligence artificielle ne se limite pas aux recommandations : elle est également exploitée pour améliorer la recherche de produits, faciliter la gestion des stocks et optimiser l'interaction globale entre les clients et la plateforme.

En somme, **SMart** s'inscrit dans une vision moderne du commerce en ligne, où technologie et simplicité se rejoignent pour répondre aux attentes des consommateurs. Ce projet reflète notre engagement à exploiter des technologies innovantes pour proposer des solutions pratiques et performantes, contribuant ainsi à l'évolution de l'e-commerce.

# I. **Presentation générale du projet**

## I.1. Introduction :

Le commerce électronique a transformé les habitudes d'achat à l'échelle mondiale, offrant rapidité et commodité. Cependant, l'expérience utilisateur reste un défi majeur pour de nombreuses plateformes. Notre projet, **SMart**, a pour objectif de combiner les technologies modernes, notamment l'intelligence artificielle, pour offrir une plateforme qui soit à la fois facile à utiliser, efficace et personnalisée.

## I.2. Étude de l'existant:

Au Maroc, des plateformes telles que **Jumia** et **Avito** dominent le marché de l'e-commerce. Ces plateformes offrent un large éventail de produits et disposent d'un certain niveau de personnalisation. Toutefois, elles présentent certaines limitations, comme une navigation parfois complexe, un manque de recommandations intelligentes adaptées aux besoins des utilisateurs, ou encore des interfaces peu ergonomiques pour certains publics.

En comparaison, **SMart** se distingue par l'intégration de l'IA pour proposer :

- Des recommandations de produits basées sur les préférences et l'historique des utilisateurs.
- Une interface utilisateur simplifiée et responsive pour une navigation fluide sur tous types d'appareils.
- Une recherche intelligente permettant de trouver des produits plus rapidement.

## I.3. Objectifs du projet

Les objectifs principaux de **SMart** sont les suivants :

- Offrir une plateforme intuitive pour simplifier le processus d'achat en ligne.
- Utiliser l'intelligence artificielle pour personnaliser l'expérience utilisateur.
- Optimiser les performances globales de la plateforme, de la recherche de produits à la gestion des stocks.
- Proposer une solution innovante qui surpassé les limitations des plateformes existantes.

## I.4. Solutions proposées

Pour répondre aux objectifs fixés, **SMart** intègre les solutions suivantes :

- Une architecture basée sur React pour le frontend et Spring Boot pour le backend, assurant modularité et performance.
- Un moteur d'intelligence artificielle entraîné pour fournir des recommandations personnalisées.
- Une base de données MongoDB pour une gestion flexible et évolutive des données produits.
- Un design moderne et responsive, conçu avec Tailwind CSS, pour une expérience utilisateur optimale.

## I.5. Méthodologie de développement

La réalisation de **SMart** a été divisée en deux phases principales, sur une période de 15 jours :

1. **Développement de l'application** : Cette phase comprenait la conception et l'implémentation des différentes fonctionnalités de la plateforme, incluant la gestion des utilisateurs, la recherche de produits, et l'interface utilisateur.
2. **Entraînement du modèle IA** : Cette étape s'est concentrée sur la création et l'optimisation d'un modèle d'intelligence artificielle pour générer des recommandations précises et adaptées.

Ces deux phases ont été menées en parallèle, permettant une avancée structurée et efficace du projet.

## I.6. Conclusion

En conclusion, **SMart** représente une avancée significative dans le domaine du commerce électronique, en mettant l'accent sur l'innovation et la simplicité. Grâce à l'intégration de technologies modernes et d'un moteur IA puissant, notre plateforme offre une expérience d'achat inédite et personnalisée.

Ce projet reflète notre volonté d'apporter des solutions concrètes aux besoins des consommateurs et d'exploiter pleinement le potentiel des nouvelles technologies.

## **II. Analyse de besoins et conception de l'application UrbanVoyage**

## II.1. Introduction

**SMart** est une application e-commerce moderne propulsée par l'intelligence artificielle. Elle est conçue selon une architecture microservices pour assurer une meilleure scalabilité, maintenance et séparation des responsabilités.

L'application permet aux clients de parcourir et d'acheter des produits en ligne, tout en offrant aux administrateurs les outils nécessaires pour gérer l'inventaire et les opérations commerciales.

## II.2. Analyse du besoin

### a. Les besoins fonctionnels

- **Pour les clients :**
  - Authentification et gestion de compte
  - Navigation et recherche de produits
  - Gestion du panier d'achat
  - Gestion de la liste de souhaits
  - Processus de paiement sécurisé
- **Pour les administrateurs :**
  - Authentification sécurisée
  - Gestion complète du catalogue produits
  - Suivi des commandes et des transactions
  - Gestion des catégories et des marques

### b. Les besoins non fonctionnels

- Sécurité et protection des données utilisateurs
- Performance et temps de réponse optimaux
- Scalabilité pour gérer les pics de charge
- Interface utilisateur intuitive et responsive
- Fiabilité du système de paiement
- Maintenance et mise à jour facilitées

## II.3. Conception UML

### a. Modélisation du système

Le système **SMart** est divisé en trois microservices principaux :

1. **ProductsCrud** : Gestion du catalogue et des produits
2. **Authentication** : Gestion des utilisateurs et des rôles
3. **CheckoutOrder** : Gestion des commandes et des paiements

### b. Présentation UML

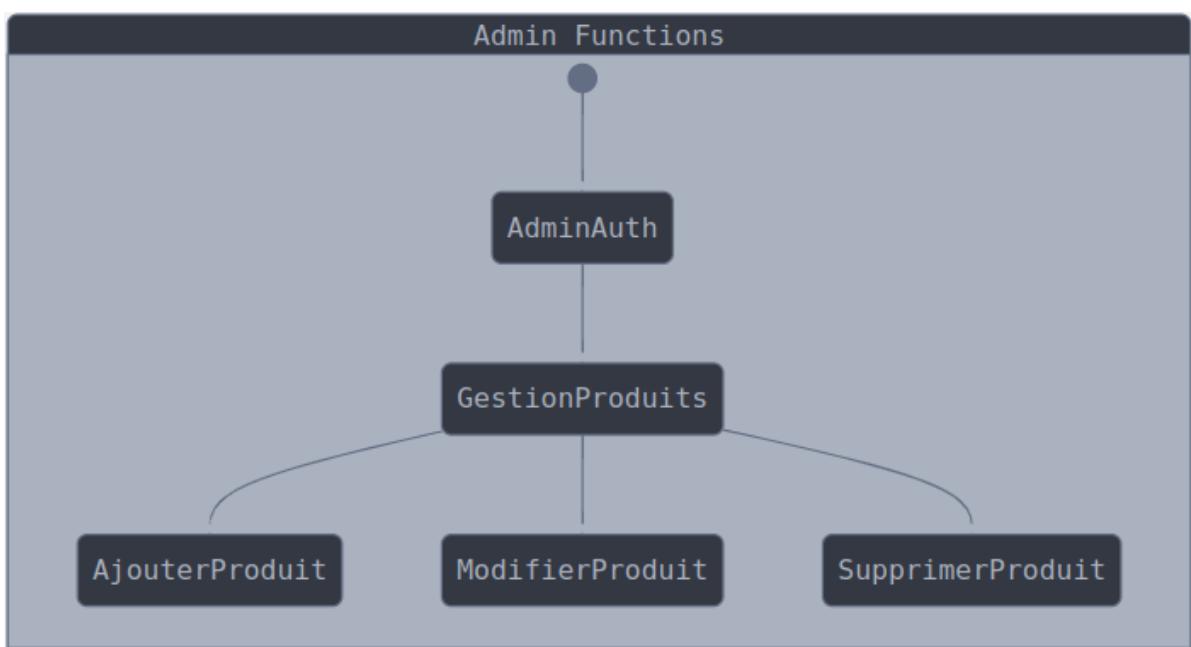
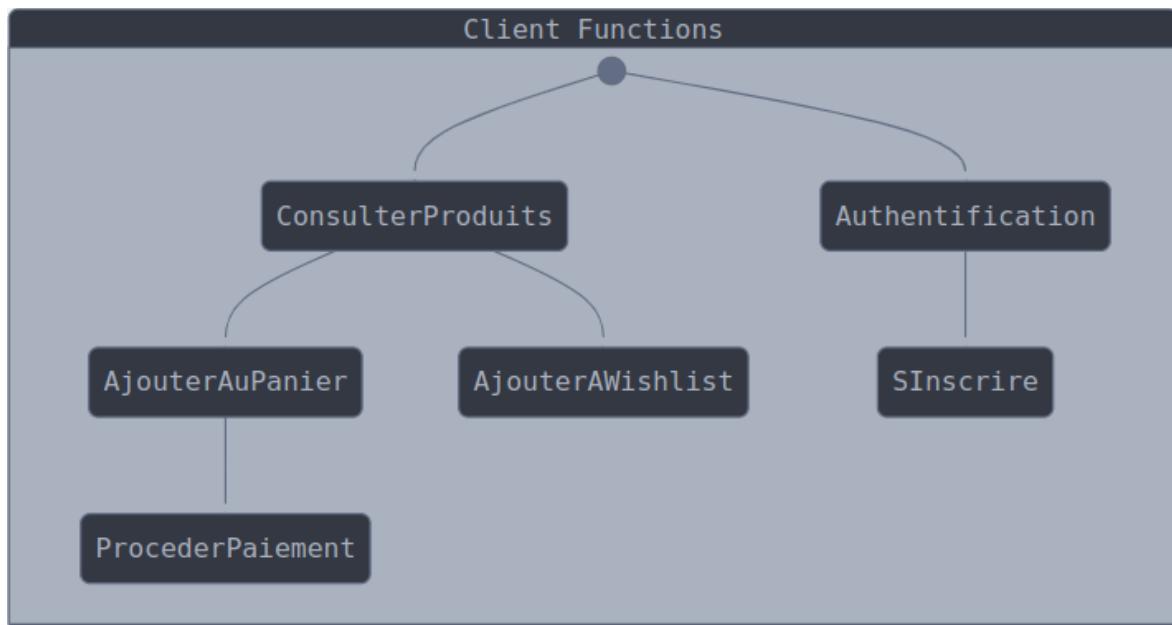
Cette section présente les différents diagrammes UML qui modélisent l'architecture et le comportement du système. Ces diagrammes incluent les cas d'utilisation, les classes, les séquences et les activités.

### c. Spécification des exigences pour **SMart**

- Architecture microservices
- Base de données MongoDB pour la persistance
- Intégration avec Stripe pour les paiements
- Système de gestion des stocks en temps réel
- Mécanisme de fidélisation (points de fidélité)

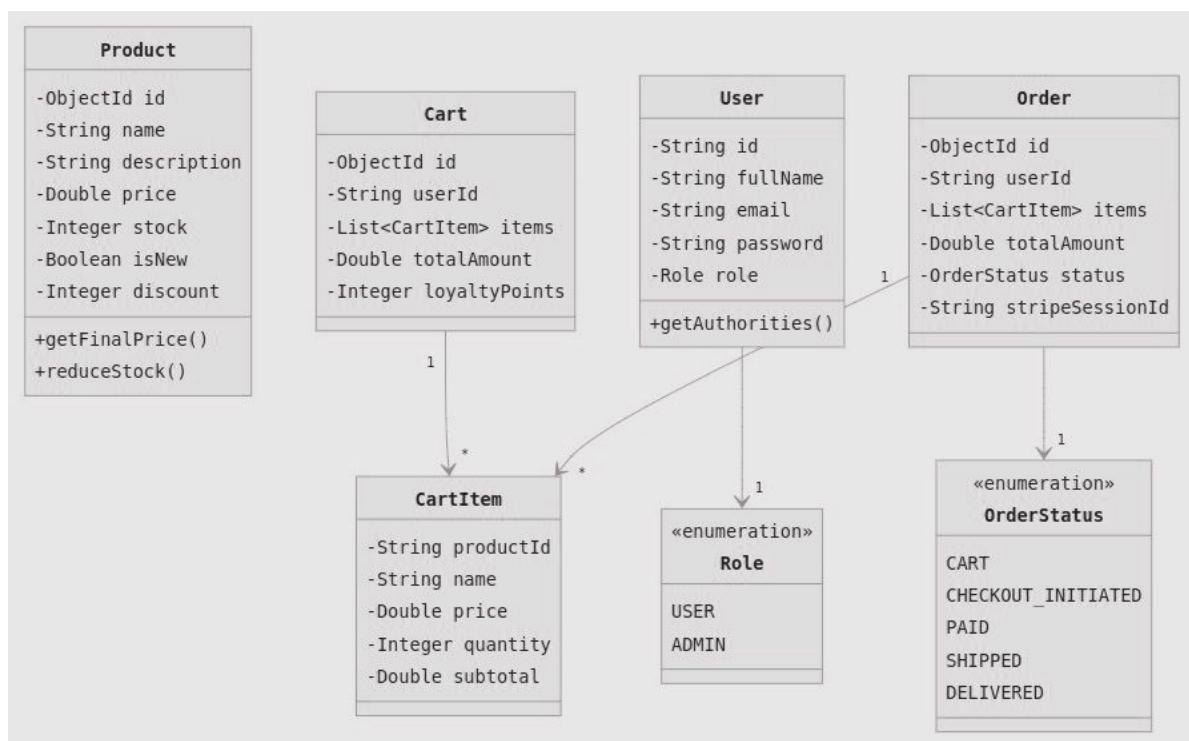
## II.4. Diagrammes UML :

### a. Diagramme de cas d'utilisation

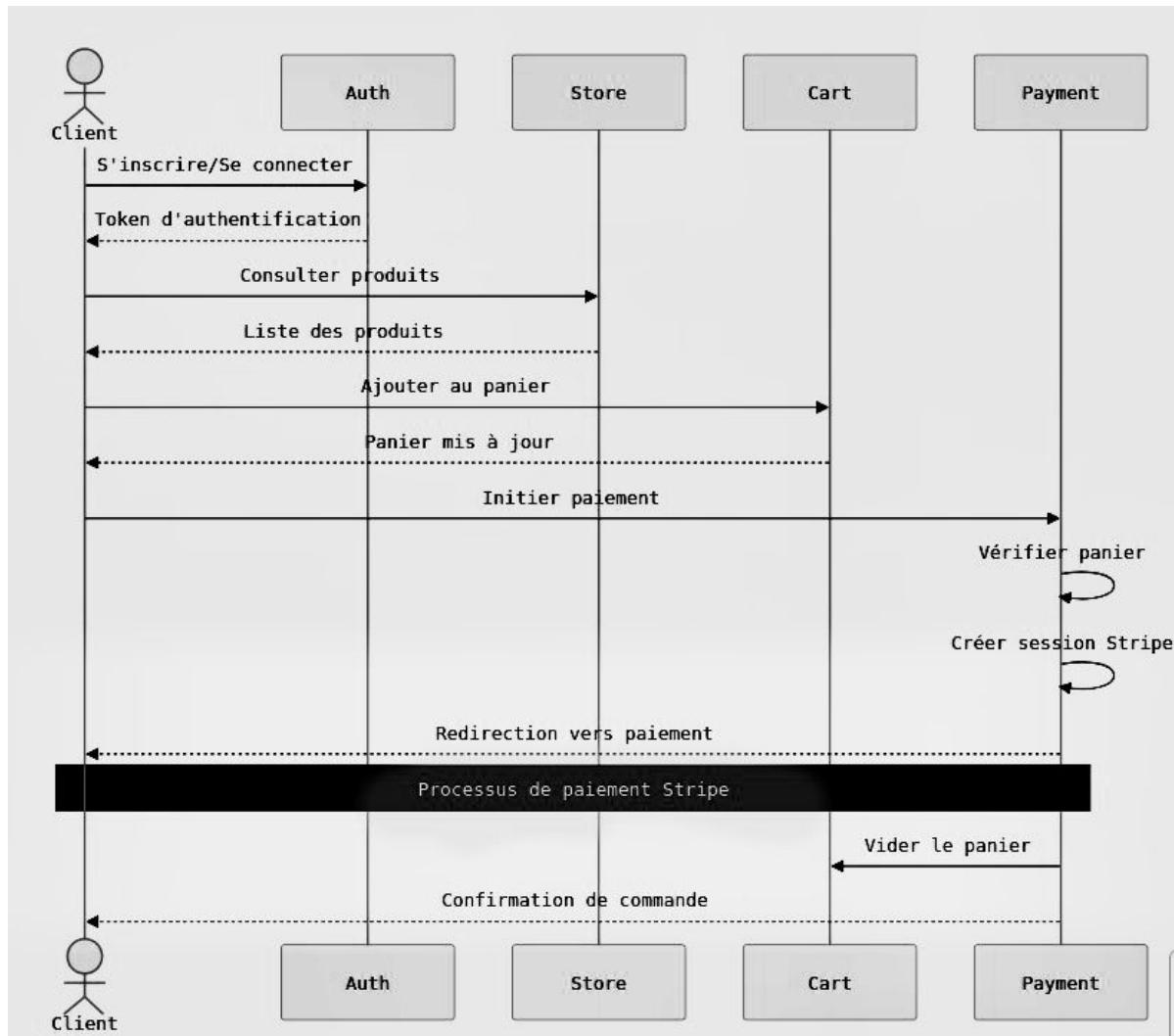


Le diagramme de cas d'utilisation ci-dessus illustre les principales interactions entre les acteurs (Client et Admin) et le système **SMart**. Il met en évidence deux types d'utilisateurs principaux avec leurs fonctionnalités respectives.

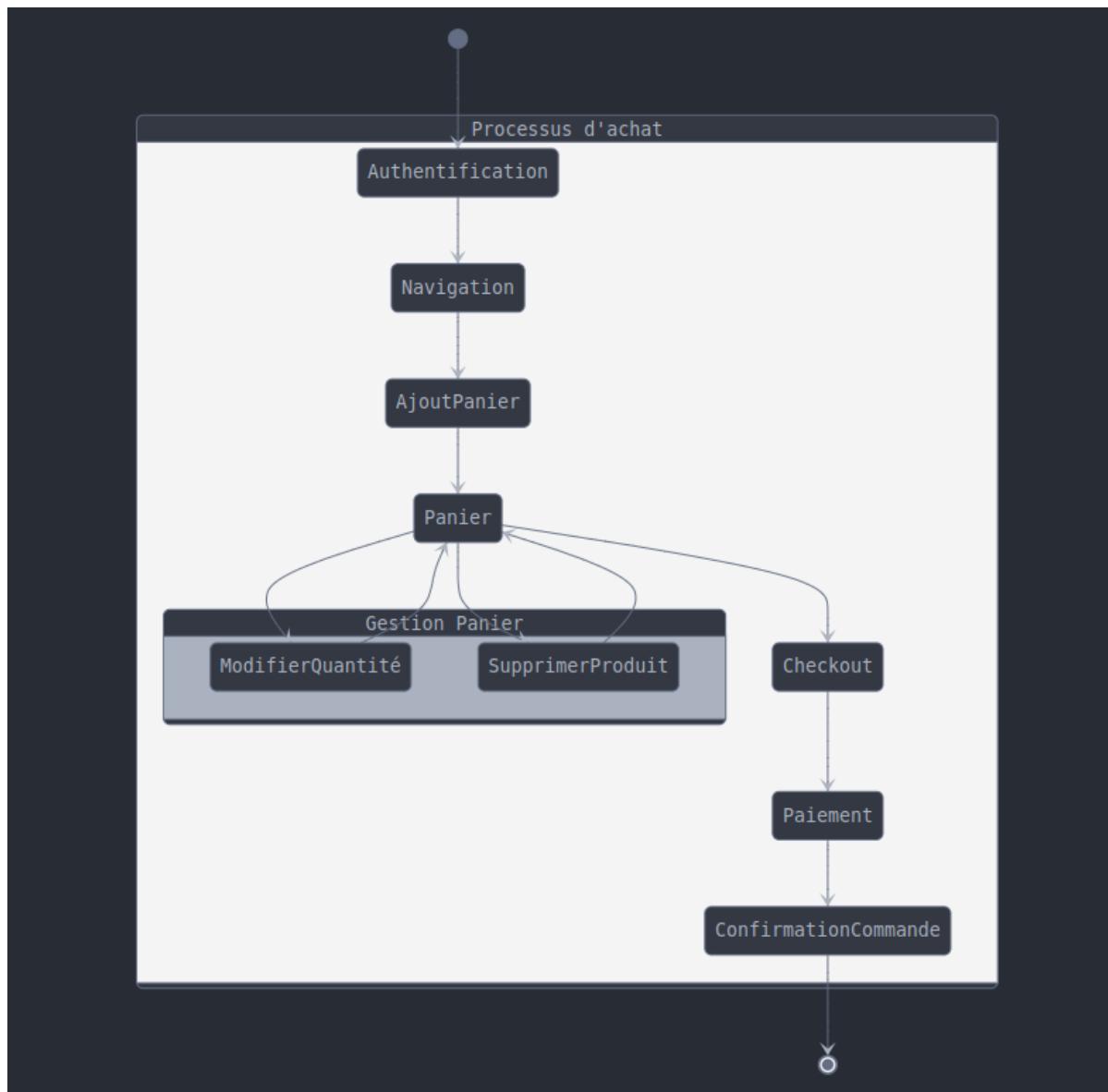
## b. Diagramme de classes:



### c. Diagramme de séquence



#### d. Diagramme d'activités



## II.5. Méthodologies et gestion de projet

### Cycle de développement adapté

Le projet **SMart** suit une approche Agile, permettant une adaptation rapide aux changements et une livraison continue de valeur.

### Avantages de cette approche pour **SMart**

- Développement itératif et incrémental
- Feedback continu des utilisateurs
- Adaptation rapide aux changements du marché
- Livraison régulière de nouvelles fonctionnalités

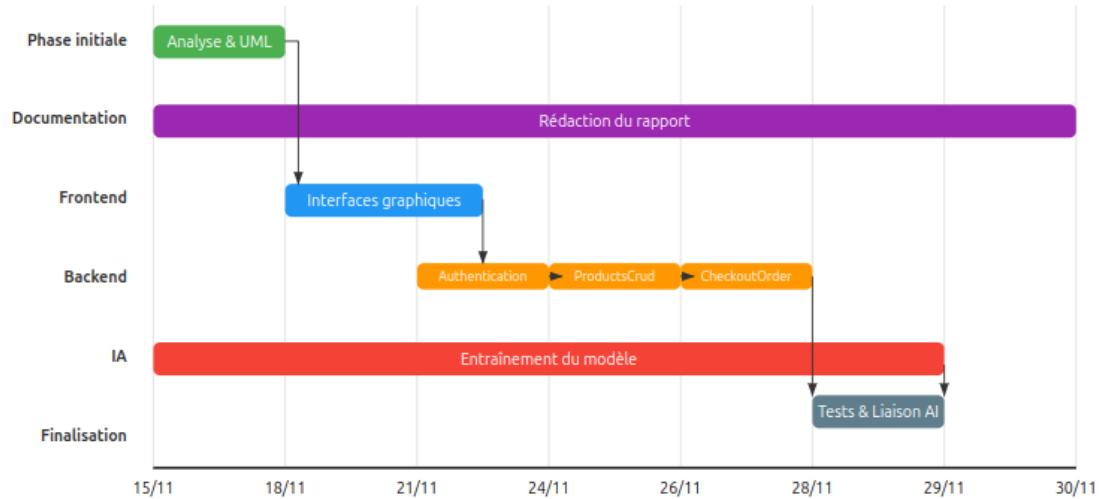
### Priorisation des fonctionnalités

- Fonctionnalités essentielles (MVP)
  - Authentification
  - Catalogue produits
  - Panier d'achat
  - Paiement sécurisé
- Fonctionnalités secondaires
  - Liste de souhaits
  - Système de points de fidélité
  - Recommandations IA

### Planning de développement

Le développement est divisé en sprints de deux semaines, avec des objectifs clairs pour chaque sprint.

## Diagramme de GANTT



Le diagramme montre la chronologie complète du projet avec :

- 1. Phase initiale** (15-18/11)
  - Analyse des besoins et conception UML
- 2. Documentation** (15-30/11)
  - Rédaction du rapport (en parallèle sur toute la durée)
- 3. Frontend** (18-21/11)
  - Création des interfaces graphiques
- 4. Backend** (21-28/11)
  - Authentication (21-24/11)
  - ProductsCrud (24-26/11)
  - CheckoutOrder (26-28/11)
- 5. Intelligence Artificielle**
  - Entraînement du modèle (15-28/11, en parallèle)
- 6. Finalisation** (28-30/11)
  - Tests & corrections
  - Liaison avec le modèle AI

Les caractéristiques principales :

- Code couleur distinct pour chaque phase majeure
- Dépendances clairement indiquées par des flèches
- Timeline précise avec dates
- Tâches parallèles visibles (documentation, IA)
- Progression séquentielle des microservices backend
- Phase finale de tests et d'intégration

## II.6. Conclusion

L'architecture microservices choisie pour **SMart** permet une grande flexibilité et scalabilité. La conception UML présentée ci-dessus démontre la robustesse et la modularité du système. Les diagrammes illustrent clairement les interactions entre les différents composants et acteurs du système, assurant une base solide pour le développement et la maintenance future de l'application.

### **III. Etude technique**

## **III.1. Introduction:**

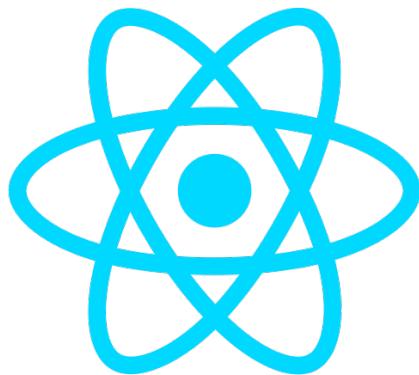
L'application **SMart** est développée en utilisant une architecture moderne basée sur le principe de microservices. Le frontend est construit avec React et diverses bibliothèques modernes, tandis que le backend utilise Spring Boot avec plusieurs microservices indépendants. Cette section détaille les choix technologiques et l'architecture utilisée.

## **III.2. Choix technologiques**

### **Front-end**

Notre application utilise un ensemble de technologies front-end modernes pour offrir une expérience utilisateur optimale :

#### **a. React**



React (v18.3.1) sert de bibliothèque principale pour la construction de l'interface utilisateur, offrant une approche basée sur les composants et une gestion efficace du DOM virtuel.

## b. Tailwind CSS



Tailwind CSS (v3.4.14) est utilisé comme framework CSS utilitaire, permettant un développement rapide et cohérent des interfaces avec une approche "utility-first".

## c. Librairies de gestion d'état et requêtes



- **Axios** (v1.7.8) pour les appels HTTP

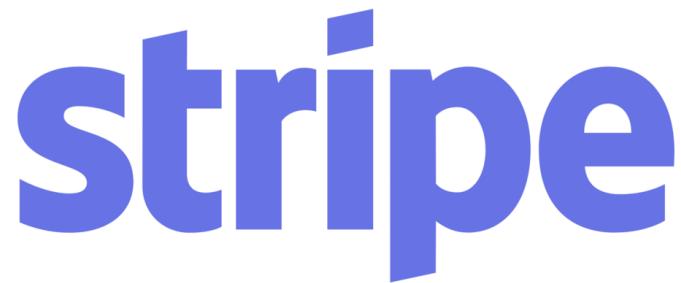


- **JWT-decode** (v4.0.0) pour la gestion des tokens d'authentification

## d. Composants et animations

- **Framer Motion** (v11.11.11) pour les animations fluides
- **Lucide React** (v0.454.0) pour les icônes
- **ShadCN UI** (v0.0.4) pour les composants UI réutilisables
- **Spline** pour introduire les objets 3D dans son application web

## e. Paiement et intégration



- **Stripe JS** (v5.2.0) et React Stripe JS (v3.0.0) pour l'intégration des paiements

## **Back-end**

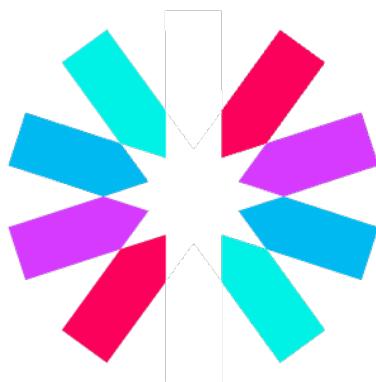
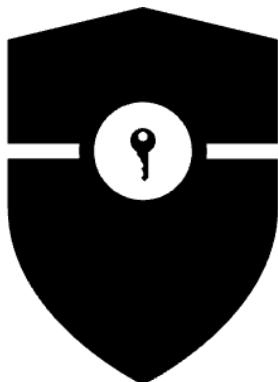
Notre backend est construit sur une architecture microservices robuste :

### **a. Spring Boot**



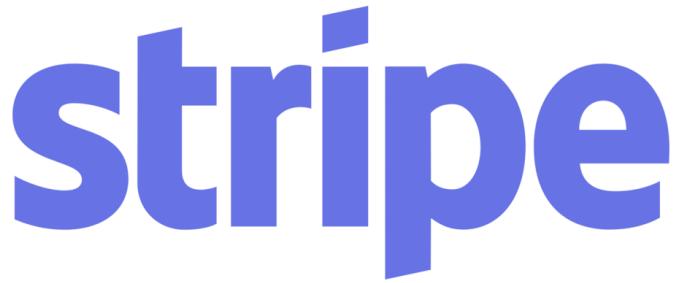
Framework principal pour le développement des microservices, offrant une configuration automatique et une grande productivité.

### **b. Spring Security & JWT**



Implémentation de la sécurité avec authentification basée sur les JSON Web Tokens (JWT).

### c. Stripe



Intégration de l'API Stripe pour la gestion des paiements sécurisés.

### e. REST Template

Dans notre architecture microservices, la communication entre les différents services est essentielle pour garantir une expérience fluide et cohérente pour l'utilisateur.

Le service **ProductsCrud** utilise **REST Template** pour communiquer avec le service **Authentication** lorsqu'il a besoin de l'ID de l'utilisateur actuel. Cette communication permet d'associer correctement chaque panier ou wishlist à un utilisateur spécifique, garantissant une gestion personnalisée des produits.

De son côté, le service **Checkout** communique avec **ProductsCrud** pour associer un panier à une transaction spécifique. Cela permet de s'assurer que chaque paiement correspond bien au panier de l'utilisateur, offrant ainsi une expérience de paiement fluide et cohérente.

Enfin, **Checkout** communique également avec **Authentication** pour valider que chaque utilisateur est bien identifié avant d'effectuer un paiement. Cette étape garantit que chaque utilisateur effectue un paiement pour sa propre transaction, renforçant la sécurité et la précision des informations liées à la commande.

## Base de données

### a. MongoDB



Dans notre architecture basée sur des microservices, chaque service dispose de sa propre base de données NoSQL, optimisée pour des besoins spécifiques.

Le service **Products** gère des collections liées aux informations sur les produits, garantissant flexibilité et évolutivité. Le service **Checkout** se concentre sur les collections relatives aux commandes et transactions, tandis que **Authentication** maintient ses propres collections pour la gestion des utilisateurs et des sessions.

Cette approche permet une gestion indépendante des données, assurant performance et scalabilité à chaque niveau du système.

## Entraînement du Modèle de Recommandation

Dans cette section, nous allons détailler les différents outils et technologies utilisés pour l'entraînement de notre modèle de recommandation de produits, ainsi que leur rôle spécifique dans le projet.

### Word2Vec avec Gensim



Gensim est une bibliothèque Python open-source spécialisée dans le traitement automatique du langage naturel et plus particulièrement dans la modélisation de thèmes et l'analyse sémantique. Dans notre projet, nous utilisons Gensim pour implémenter Word2Vec, un modèle d'apprentissage profond qui permet de créer des représentations vectorielles de mots.

Le modèle Word2Vec est entraîné sur notre jeu de données de commandes d'achats avec les paramètres suivants :

- **vector\_size=100** : Chaque produit est représenté par un vecteur de 100 dimensions
- **window=5** : Le contexte considéré est de 5 produits avant et après
- **min\_count=1** : Tous les produits sont pris en compte, même ceux n'apparaissant qu'une seule fois
- **workers=4** : Utilisation de 4 threads pour l'entraînement parallèle

## Pandas



Pandas est une bibliothèque Python essentielle pour la manipulation et l'analyse de données. Dans notre projet, elle est utilisée pour :

- Charger les données depuis le fichier CSV
- Nettoyer les données en supprimant les lignes vides
- Transformer les données en format approprié pour l'entraînement du modèle

## NumPy



NumPy est une bibliothèque fondamentale pour le calcul scientifique en Python. Dans notre système, elle est utilisée pour :

- Calculer le centre des vecteurs de produits
- Effectuer des calculs de distances euclidiennes
- Manipuler efficacement les vecteurs de grande dimension

## Architecture de système d'entraînement

L'entraînement du modèle suit une architecture modulaire composée de plusieurs fonctions clés :

1. **load\_data()** : Charge et nettoie les données d'entrée
2. **train\_word2vec\_model()** : Entraîne le modèle sur les données préparées
3. **calculate\_vector\_center()** : Calcule le point central de l'espace vectoriel
4. **suggest\_products\_near\_center()** : Identifie les produits génériques
5. **get\_similar\_products()** : Interface principale pour obtenir des recommandations

## Intégration avec API Flask



# Flask



Flask est un micro-framework web Python reconnu pour sa simplicité et sa flexibilité. Dans notre projet, nous l'utilisons pour créer une API REST qui sert d'interface entre notre modèle Word2Vec et l'application frontend.

Flask nous permet de charger le modèle en mémoire au démarrage du serveur et de gérer efficacement les requêtes de recommandation. Grâce à sa légèreté et sa facilité d'utilisation, nous avons pu rapidement mettre en place des endpoints permettant de récupérer des produits similaires ou des suggestions génériques lorsqu'un produit n'est pas trouvé dans le modèle.

L'intégration de Flask-CORS nous permet également de gérer de manière sécurisée les communications cross-origin avec notre frontend.

## Performance et Optimisation

Le système a été optimisé pour :

- Une utilisation efficace de la mémoire lors du chargement du modèle
- Un temps de réponse rapide pour les requêtes API
- Une gestion appropriée des cas où un produit n'existe pas dans le modèle

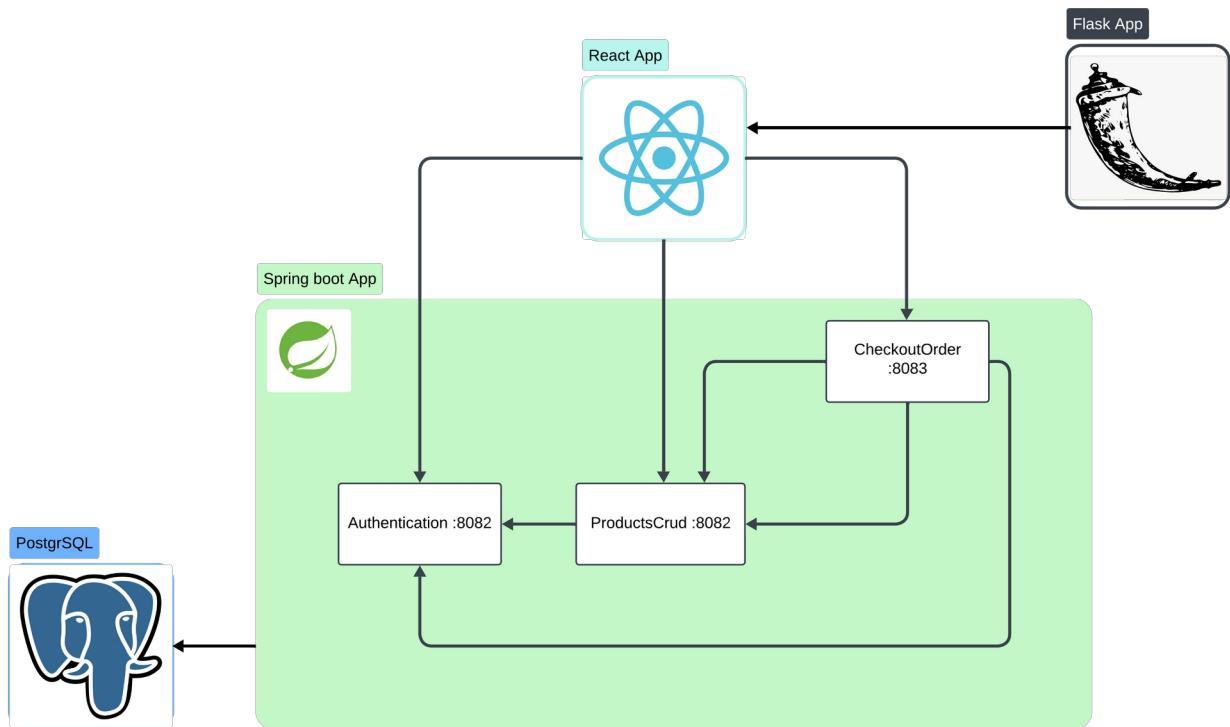
Cette architecture permet une séparation claire des responsabilités et une maintenance facilitée du système de recommandation.

## III.3. Architecture logicielle

Notre application adopte une architecture microservices moderne, privilégiant la séparation des responsabilités et la flexibilité du système. Le backend est structuré autour de trois microservices indépendants :

**Authentication**, **ProductsCrud** et **Checkout**, chacun exposant ses propres API REST sur des ports distincts. Cette modularité permet une maintenance facilitée et une évolution indépendante de chaque service.

Le frontend communique directement avec chaque microservice via des requêtes HTTP en ciblant les ports appropriés, évitant ainsi la complexité d'une gateway API. Les interactions inter-services sont gérées par REST Template, permettant une communication synchrone fiable. Par exemple, **ProductsCrud** interagit avec **Authentication** pour la gestion des paniers personnalisés, tandis que **Checkout** communique avec les deux autres services pour assurer des transactions sécurisées et cohérentes. Cette architecture distribuée offre une grande scalabilité tout en maintenant une forte cohésion fonctionnelle.

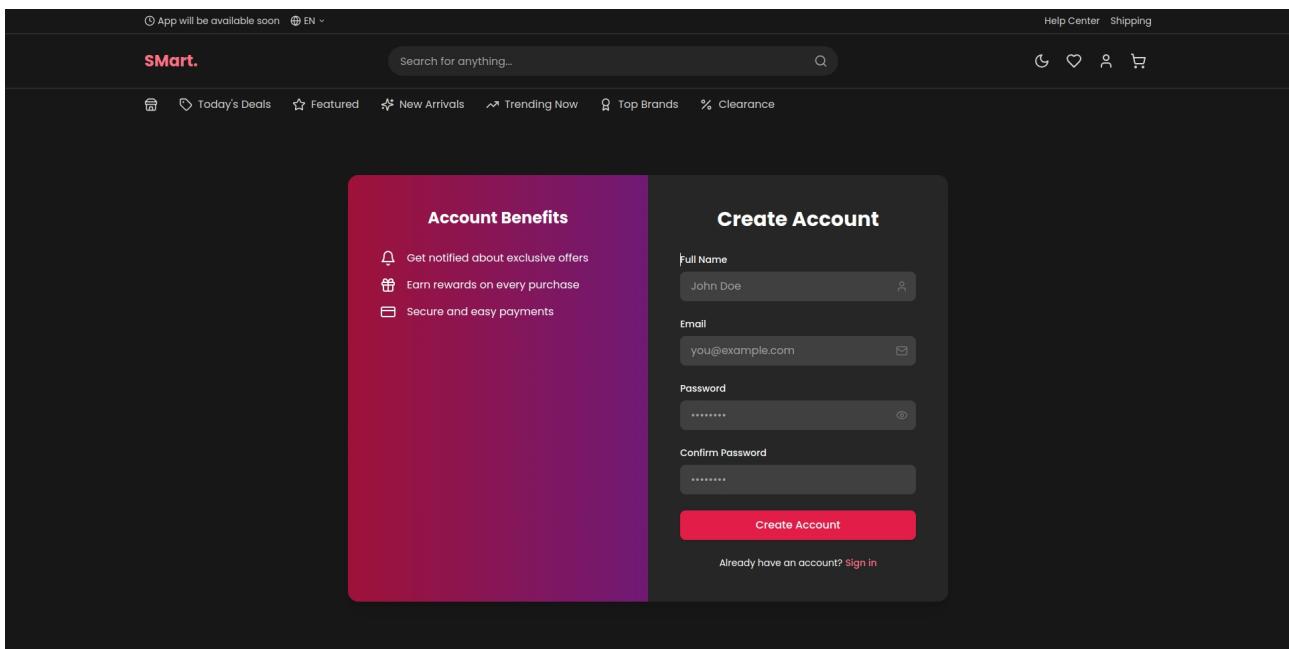
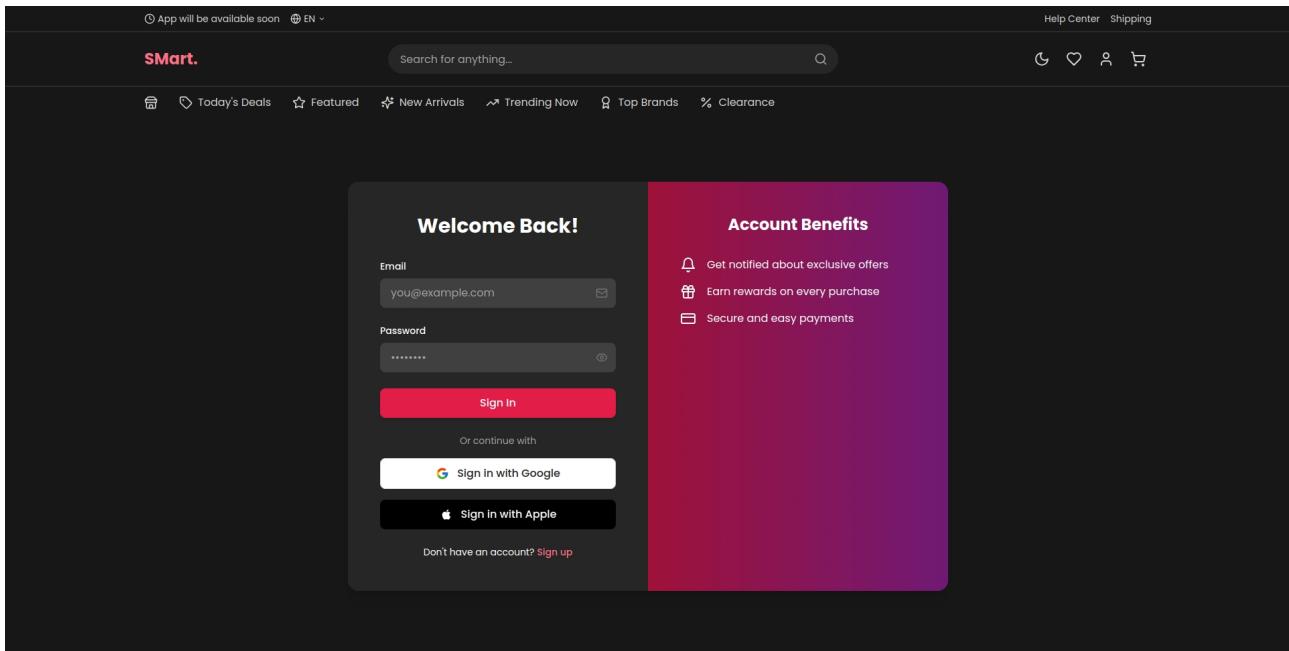


## **IV. Realisations & Tests**

# IV.1. Realisation

## a. Authentification

L'authentification est la première étape permettant aux utilisateurs d'accéder à leurs fonctionnalités personnalisées. Cette interface est gérée par le composant **Account.tsx**.



Description des fonctionnalités :

- Connexion avec email et mot de passe
- Inscription pour les nouveaux utilisateurs
- Gestion des erreurs de saisie
- Redirection vers la page d'accueil après connexion

## b. Page d'Accueil et À Propos

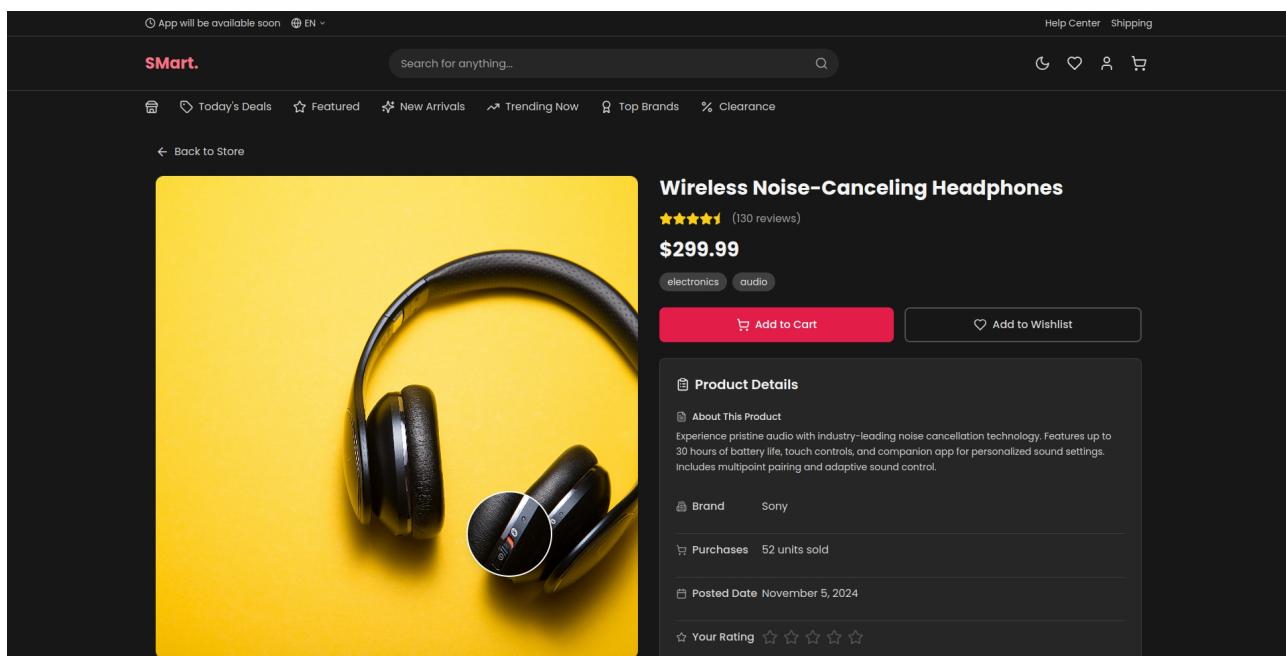
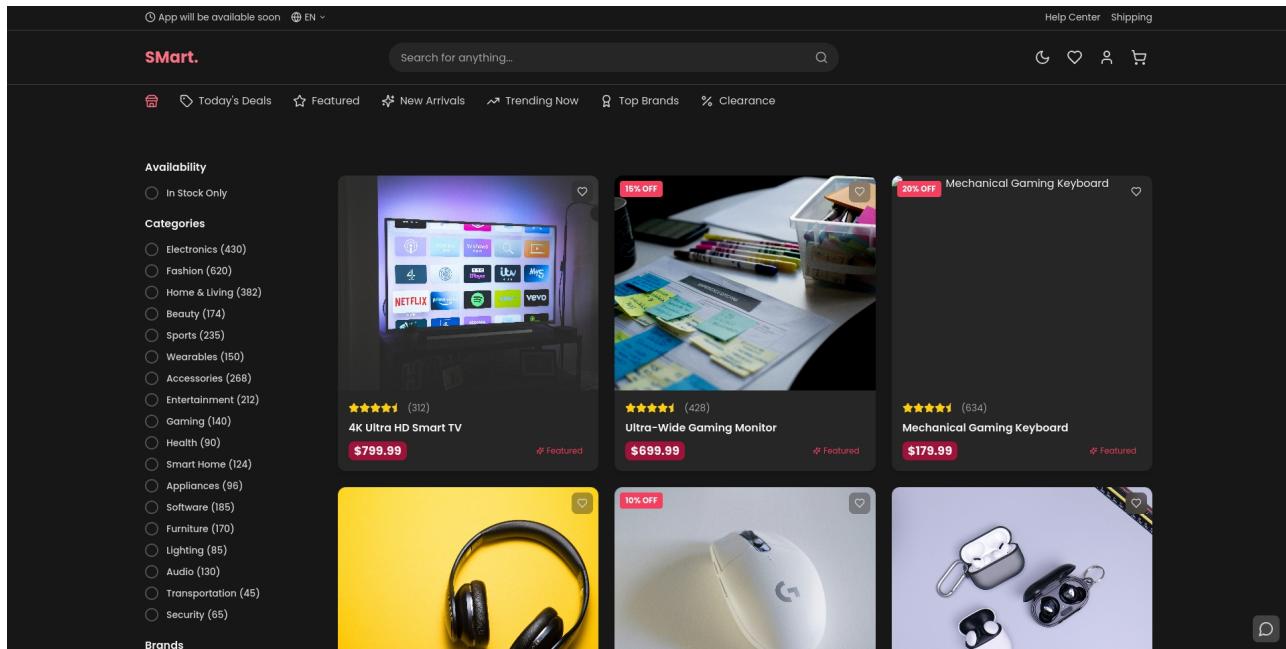
La page d'accueil (**Home.tsx**) sert de point d'entrée principal à l'application, tandis que la page À propos (**About.tsx**) présente les informations générales sur le service.

The screenshot shows the homepage of a shopping app called "SMart.". The top navigation bar includes links for "Help Center" and "Shipping". Below the header is a search bar with placeholder text "Search for anything...". A navigation menu features links for "Today's Deals", "Featured", "New Arrivals", "Trending Now", "Top Brands", and "Clearance". The main content area has a red background. It features a large text block: "Shop Smarter with AI-Powered Recommendations". Below this, a subtext reads: "Experience personalized shopping like never before. Our AI learns your preferences to show you products you'll love." Two buttons are present: a yellow "Start Shopping" button and a white "Learn More" button. To the right is a 3D rendering of a blue and silver AI robot head mounted on a cube base. A small black box labeled "AI Assistant" with the subtext "Always here to help" is positioned near the bottom left. At the very bottom of the screen, there is a dark bar with the text "Trending Now".

The screenshot shows the "About Us" page of the shopping app. The top navigation bar is identical to the homepage. The main content area has a red background. It features a large text block: "About Our AI-Powered Shopping Platform". Below this, a subtext reads: "Discover the story behind our innovative approach to online shopping and the team that makes it possible." At the bottom of the screen, there is a dark bar with the text "Our Mission". Below this, a paragraph of text states: "At AI Shopping Assistant, we're revolutionizing the online shopping experience by harnessing the power of artificial intelligence. Our platform learns from user preferences to provide personalized product recommendations, making shopping easier and more enjoyable than ever before." Another paragraph at the bottom states: "We believe that technology should simplify and enhance our lives. That's why we've developed an intuitive, AI-driven system that understands your unique tastes and needs, helping you discover products you'll love without the hassle of endless searching."

## c. Catalogue des Produits

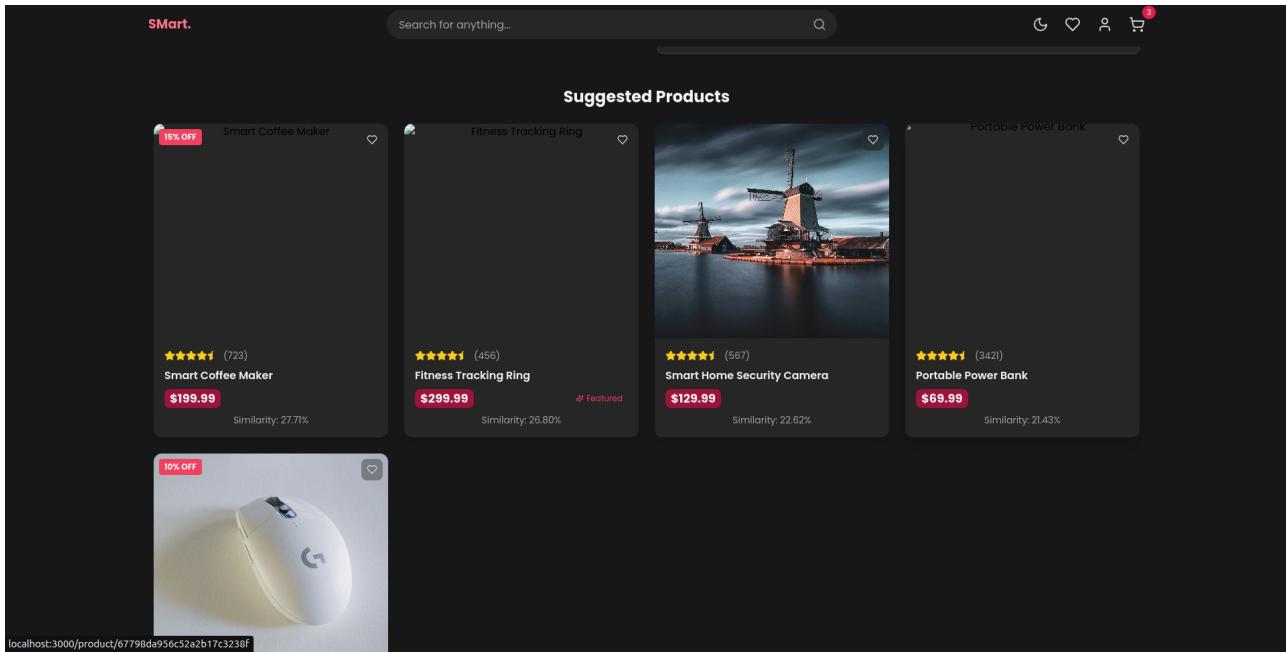
Le catalogue est géré par deux composants principaux : Store.tsx pour l'affichage général et ProductPage.tsx pour le détail des produits.



Fonctionnalités principales :

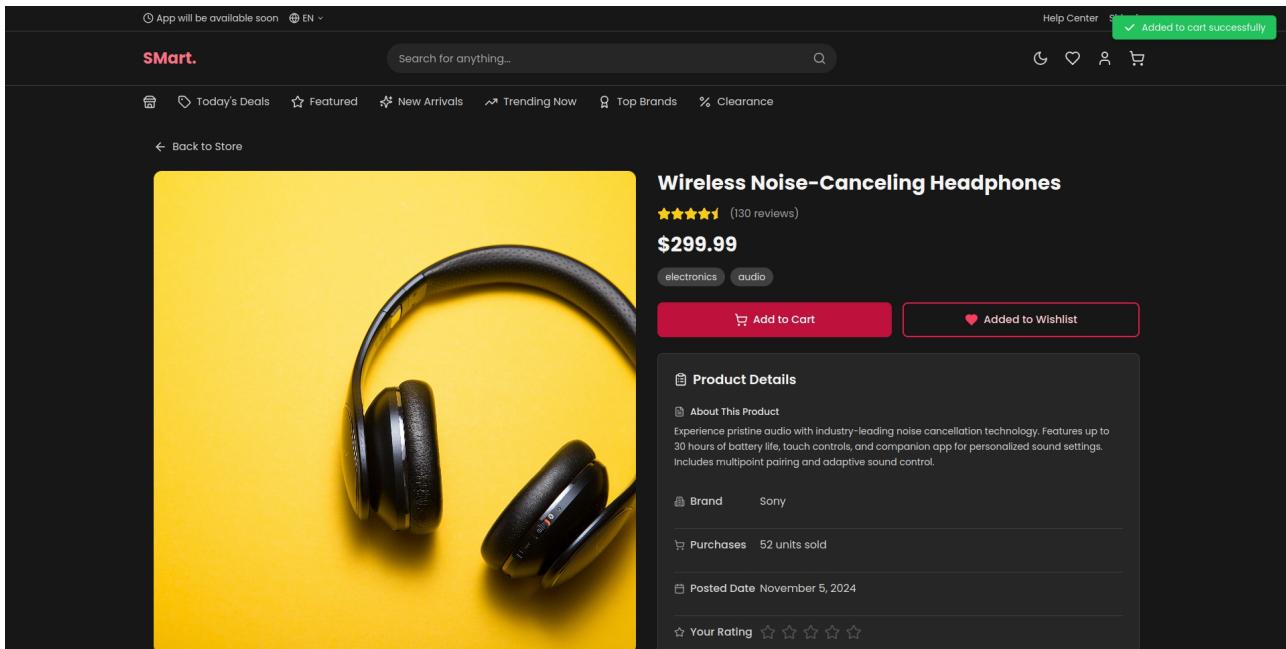
- Filtrage des produits
- Tri par prix et catégorie
- Vue détaillée des produits

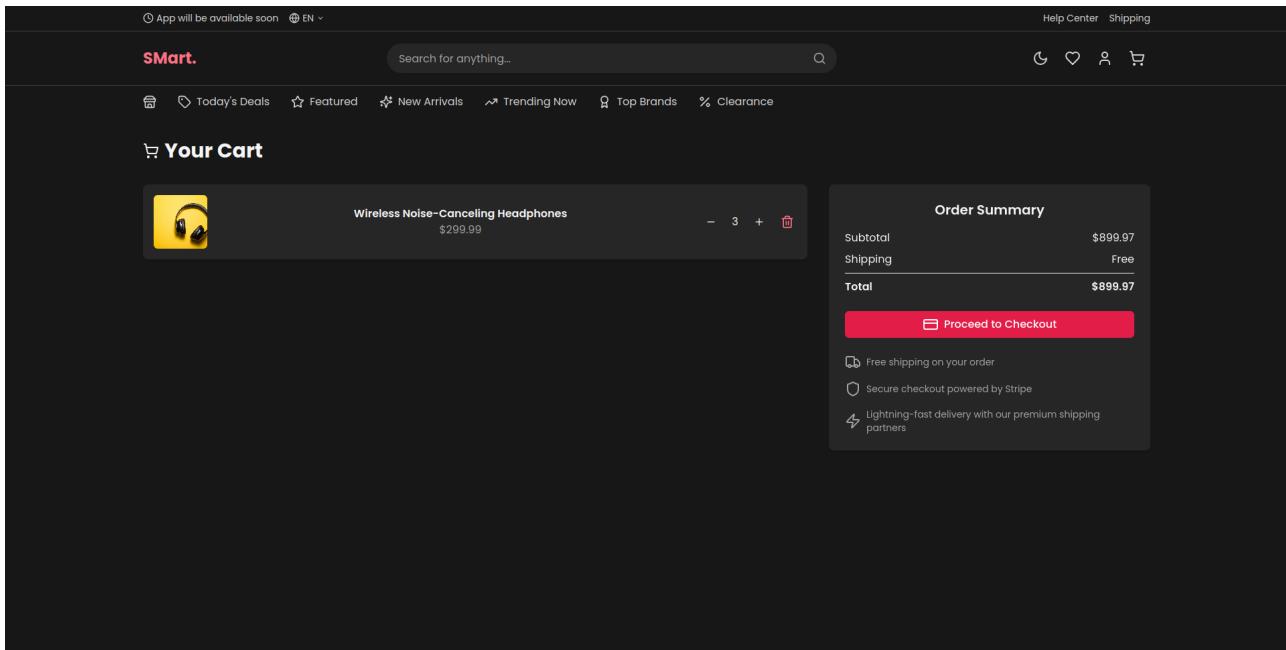
- Ajout au panier
- Avoir des suggestion des produits



## d. Gestion du Panier

Le panier est géré par le composant Cart.tsx, permettant aux utilisateurs de gérer leurs sélections avant l'achat.





## Fonctionnalités :

- Ajout/suppression de produits
- Modification des quantités
- Calcul automatique du total
- Passage à la commande

The payment page shows the following details:

- Contact information:** Ahmed Reda Meftah, TEST MODE, Email: meftahmedreda02@gmail.com
- Payment method:** Card selected. Card information: 4242 4242 4242 4242, Exp: 02/28, CVV: 123. Cardholder name: hmed. Country or region: Morocco.
- Buttons:** A blue 'Pay' button with a lock icon and a small 'stripe' logo below it.

## e. Administration des Produits

L'interface d'administration (ProductCRUD.tsx) permet la gestion complète du catalogue.

NAME	PRICE	RATING	CATEGORY	BRAND	ACTIONS
4K Ultra HD Smart TV	\$799.99	★ 4.8	electronics, entertainment, smart home	LG	
Smart Home Security Camera	\$129.99	★ 4.4	smart home, security	Nest	
Ultra-Wide Gaming Monitor	\$699.99	★ 4.6	electronics, gaming	Samsung	
Mechanical Gaming Keyboard	\$179.99	★ 4.7	gaming, accessories	Corsair	
Wireless Charging Pad	\$39.99	★ 4.4	electronics, accessories	Anker	
Portable Power Bank	\$69.99	★ 4.7	electronics, accessories	Anker	
Smart Coffee Maker	\$199.99	★ 4.4	smart home, appliances	Ninja	
Wireless Noise-Canceling Headphones	\$299.99	★ 4.5	electronics, audio	Sony	
Premium Smart Watch	\$199.99	★ 4.6	wearables, health	Apple	
Professional Gaming Mouse	\$149.99	★ 4.7	gaming, accessories	Razer	
Wireless Earbuds Pro	\$249.99	★ 4.8	electronics, audio	Apple	
Smart Robot Vacuum	\$399.99	★ 4.5	smart home, appliances	Roomba	

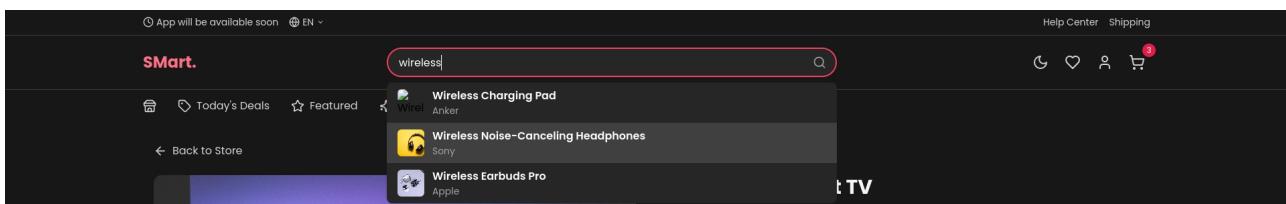
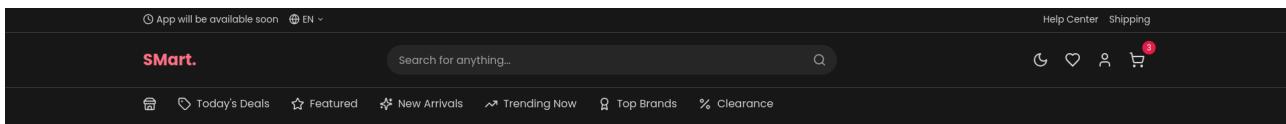
## Fonctionnalités administrateur :

- Ajout de nouveaux produits
- Modification des produits existants
- Suppression de produits
- Gestion des stocks

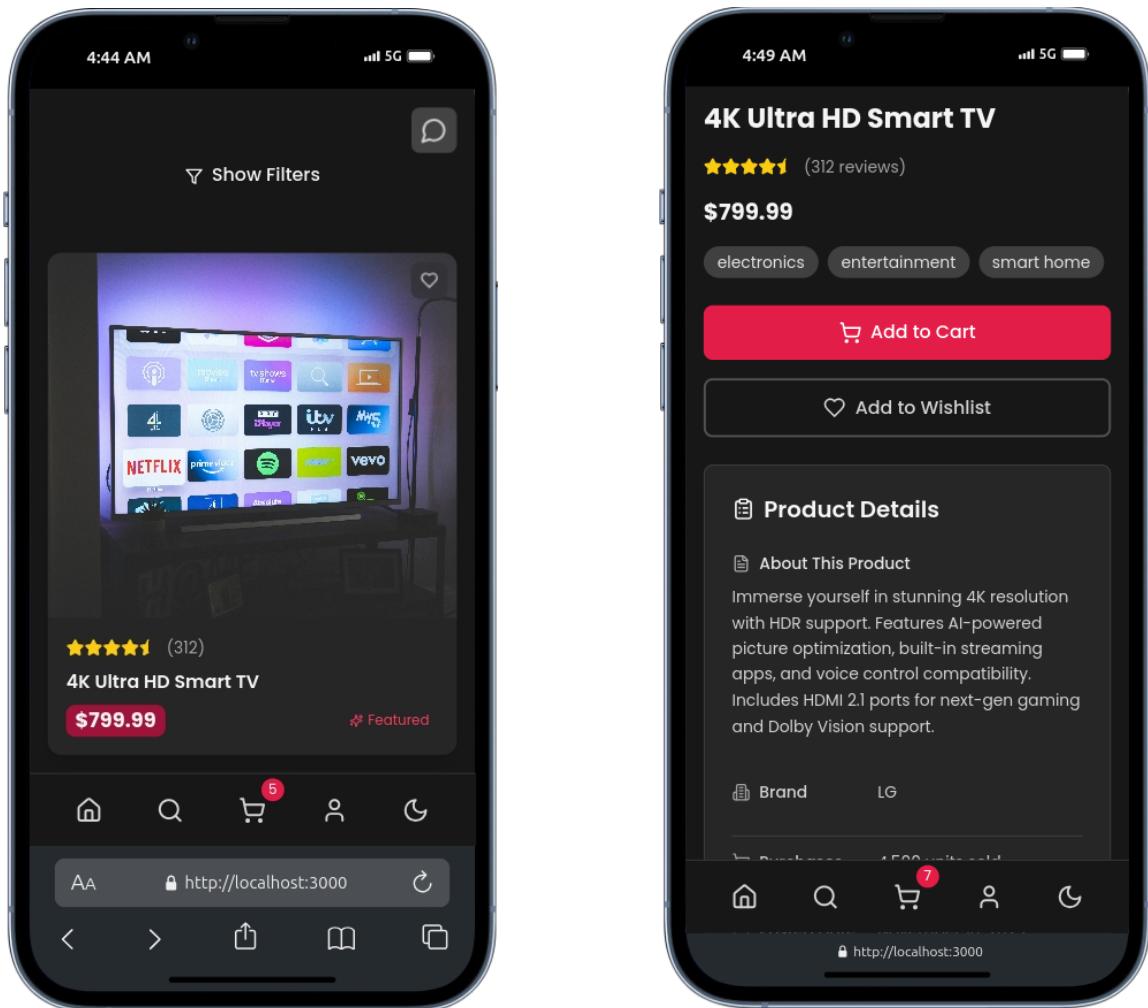
## f. Navigation et Interface Utilisateur

Points communs à toutes les pages :

- Navigation responsive
- Barre de recherche
- Menu utilisateur
- Panier accessible depuis toutes les pages
- Design cohérent et moderne



Chaque section ci-dessus représente une partie essentielle de l'interface utilisateur, avec ses composants React associés et ses fonctionnalités spécifiques. Les captures d'écran peuvent être insérées aux emplacements indiqués pour illustrer visuellement chaque fonctionnalité.



## **IV.2. Tests**

### **a. Tests d'API avec Postman**

Les tests d'API ont été réalisés pour vérifier le bon fonctionnement des endpoints de notre backend Spring Boot, assurant une communication efficace avec le frontend React.

- Objectif : Valider le fonctionnement des différents endpoints de l'API.
- Outils utilisés : Postman pour les tests d'API.
- Exemple : Test des requêtes CRUD pour les produits, validation des réponses API pour la gestion du panier et des commandes.

### **b. Tests Manuels Frontend**

Des tests manuels ont été effectués sur l'interface utilisateur pour s'assurer de la bonne expérience utilisateur et du bon fonctionnement des différentes fonctionnalités.

- Objectif : Vérifier le bon fonctionnement des composants React et des interactions utilisateur.
- Exemple : Test du formulaire d'inscription, de la navigation, de l'ajout au panier, et du processus de paiement.

### **c. Debugging et Corrections**

Pendant le développement, un processus continu de debugging a été mis en place pour identifier et corriger les problèmes rencontrés.

- Objectif : Résoudre les bugs et améliorer la stabilité de l'application.
- Exemple : Correction des problèmes d'affichage, optimisation des requêtes API, résolution des erreurs de console.

## **IV.3. Conclusion**

La phase de tests, bien que basique, a permis de valider les fonctionnalités essentielles de notre plateforme e-commerce. Les tests d'API avec Postman et les tests manuels ont été suffisants pour identifier et corriger les principaux problèmes, assurant un niveau acceptable de fonctionnalité pour cette première version de l'application.



## **V. Conclusion générale et Perspectives**

## V.1. Conclusion Générale

Le développement de **Smart**, a permis de créer une solution robuste et innovante qui combine les technologies modernes du commerce électronique avec l'intelligence artificielle. Ce projet, réalisé dans le cadre de notre formation d'ingénieur, représente une application concrète des concepts avancés en développement web, microservices et intelligence artificielle.

La réalisation de ce projet a nécessité une approche méthodique, débutant par une analyse approfondie des besoins, suivie d'une conception architecturale basée sur les microservices. L'implémentation a mis l'accent sur l'intégration des fonctionnalités e-commerce essentielles avec des capacités de recommandation AI, le tout supporté par une architecture moderne utilisant React, Spring Boot et Flask.

Ce projet nous a permis d'acquérir une expérience significative dans la gestion de projets complexes, l'utilisation des technologies modernes, et l'intégration de composants AI dans une application web professionnelle.

## V.2. Perspectives

Bien que notre plateforme soit fonctionnelle et réponde aux objectifs initiaux, plusieurs axes d'amélioration peuvent être envisagés pour enrichir l'expérience utilisateur et étendre les capacités du système :

- Amélioration de l'Authentification et de la Sécurité
- Intégration de systèmes d'authentification tiers (OAuth) pour simplifier la connexion des utilisateurs
- Support de l'authentification via iCloud pour les utilisateurs Apple
- Renforcement de la sécurité des transactions et de la protection des données utilisateurs

- Développement Mobile
  - Création d'une application mobile native complémentaire à la plateforme web
  - Synchronisation des paniers et des préférences entre les plateformes web et mobile
  - Implémentation de notifications push pour les promotions et le suivi des commandes
  - Optimisation de l'expérience d'achat sur mobile
- 
- Enrichissement de l'Intelligence Artificielle
  - Amélioration des capacités du chatbot pour une meilleure compréhension des requêtes clients
  - Intégration de fonctionnalités de traitement du langage naturel plus avancées
  - Personnalisation accrue des recommandations produits basée sur l'historique des interactions
  - Développement d'un système de support client automatisé plus sophistiqué

Ces perspectives d'évolution visent à renforcer la position de notre plateforme sur le marché, en offrant une expérience utilisateur plus riche et plus personnalisée, tout en tirant parti des dernières avancées technologiques en matière d'authentification et d'intelligence artificielle.

# Bibliographies

- **Spring Boot**

Documentation officielle de Spring Boot. Consulté sur  
<https://docs.spring.io/spring-boot/docs/current/reference/html/>

- **Spring Security**

Documentation de Spring Security. Consulté sur  
<https://docs.spring.io/spring-security/reference/index.html>

- **React**

Documentation officielle de React. Consulté sur <https://react.dev/learn>

- **Tailwind CSS**

Documentation de Tailwind CSS. Consulté sur <https://tailwindcss.com/docs>

- **Framer Motion**

Documentation de Framer Motion. Consulté sur  
<https://www.framer.com/motion/>

- **Spline**

Documentation de Spline. Consulté sur <https://docs.spline.design/>

- **Stripe**

Documentation de l'API Stripe. Consulté sur <https://stripe.com/docs/api>

- **MongoDB**

Documentation officielle de MongoDB. Consulté sur  
<https://www.mongodb.com/docs/>

- **JWT**

Documentation sur JSON Web Tokens. Consulté sur  
<https://jwt.io/introduction/>

- **Jenkins**

Documentation officielle de Jenkins. Consulté sur  
<https://www.jenkins.io/doc/>

- **SonarQube**

Documentation de SonarQube. Consulté sur <https://docs.sonarqube.org/>

- **Postman**

Documentation de Postman. Consulté sur  
<https://learning.postman.com/docs/>