



## Institut National des Postes et Télécommunications (INPT)

Rabat, Maroc

Filière : Ingénierie des Données (INE 3)

### MODULE

#### Sécurité des Applications Web Modernes

## Rapport de Projet

### Application E-Commerce Full Stack

Gestion de Produits avec Authentification Sécurisée

Réalisé par :  
Reda El Hadfi

Encadré par :  
Pr. Abdelhay HAQIQ

Année Universitaire 2025-2026

# Résumé

## Français

Ce rapport présente la conception et la réalisation d'une application web Full Stack de gestion de produits e-commerce. Le projet met en œuvre les meilleures pratiques de sécurité des applications web modernes, incluant l'authentification sécurisée par JWT, le hashage des mots de passe avec bcrypt, la gestion des rôles (RBAC), et la protection contre les vulnérabilités OWASP Top 10 (injection NoSQL, XSS, CSRF).

L'application est développée avec une architecture client-serveur en trois tiers : un frontend React.js moderne et responsive, un backend Node.js/Express robuste et sécurisé, et une base de données MongoDB Atlas hébergée dans le cloud. Elle offre des fonctionnalités complètes de CRUD, recherche avancée, filtrage multi-critères, pagination, upload de fichiers et un tableau de bord administrateur avec statistiques en temps réel.

Le projet a été déployé en production sur des services cloud (Render pour le backend, Vercel pour le frontend), démontrant ainsi une maîtrise complète du cycle de développement d'une application web moderne et sécurisée.

**Mots-clés :** Full Stack, React, Node.js, MongoDB, Sécurité Web, JWT, REST API, CRUD, E-Commerce

## English

This report presents the design and implementation of a Full Stack e-commerce product management web application. The project implements best practices in modern web application security, including secure JWT authentication, bcrypt password hashing, role-based access control (RBAC), and protection against OWASP Top 10 vulnerabilities (NoSQL injection, XSS, CSRF).

The application is developed with a three-tier client-server architecture : a modern and responsive React.js frontend, a robust and secure Node.js/Express backend, and a MongoDB Atlas database hosted in the cloud. It offers complete CRUD functionalities, advanced search, multi-criteria filtering, pagination, file upload, and an admin dashboard with real-time statistics.

The project has been deployed to production on cloud services (Render for backend, Vercel for frontend), demonstrating complete mastery of the development cycle of a modern and secure web application.

**Keywords :** Full Stack, React, Node.js, MongoDB, Web Security, JWT, REST API, CRUD, E-Commerce

# Table des matières

|  |           |
|--|-----------|
| <b>Résumé</b>  | <b>1</b>  |
| <b>1 Introduction et Contexte</b>                        | <b>6</b>  |
| 1.1 Présentation du Projet . . . . .                     | 6         |
| 1.2 Problématique . . . . .                              | 6         |
| 1.3 Objectifs du Projet . . . . .                        | 6         |
| 1.4 Cahier des Charges Résumé . . . . .                  | 7         |
| 1.4.1 Exigences Fonctionnelles . . . . .                 | 7         |
| 1.4.2 Exigences Non-Fonctionnelles . . . . .             | 7         |
| 1.5 Méthodologie de Travail . . . . .                    | 7         |
| <b>2 Analyse et Conception</b>                           | <b>8</b>  |
| 2.1 Analyse des Besoins . . . . .                        | 8         |
| 2.1.1 Besoins Fonctionnels . . . . .                     | 8         |
| 2.1.2 Identification des Acteurs . . . . .               | 8         |
| 2.2 Diagramme de Cas d'Utilisation . . . . .             | 8         |
| 2.3 Modélisation de la Base de Données . . . . .         | 10        |
| 2.3.1 Schéma Relationnel . . . . .                       | 10        |
| 2.3.2 Description des Collections . . . . .              | 11        |
| 2.4 Contraintes de Sécurité . . . . .                    | 11        |
| <b>3 Architecture et Technologies</b>                    | <b>12</b> |
| 3.1 Architecture Globale . . . . .                       | 12        |
| 3.1.1 Tier 1 : Couche Présentation (Frontend) . . . . .  | 12        |
| 3.1.2 Tier 2 : Couche Logique Métier (Backend) . . . . . | 13        |
| 3.1.3 Tier 3 : Couche Données (Database) . . . . .       | 13        |
| 3.2 Stack Technologique . . . . .                        | 13        |
| 3.2.1 Justification des Choix . . . . .                  | 13        |
| 3.2.2 Avantages de l'Architecture Choisie . . . . .      | 14        |
| 3.3 Structure du Code . . . . .                          | 14        |
| 3.3.1 Structure Backend . . . . .                        | 14        |
| 3.3.2 Structure Frontend . . . . .                       | 14        |
| <b>4 Fonctionnalités Détaillées</b>                      | <b>16</b> |
| 4.1 Authentification et Autorisation . . . . .           | 16        |
| 4.1.1 Inscription des Utilisateurs . . . . .             | 16        |
| 4.1.2 Connexion Sécurisée . . . . .                      | 16        |
| 4.1.3 Gestion des Rôles (RBAC) . . . . .                 | 17        |
| 4.2 Gestion des Produits (CRUD) . . . . .                | 17        |
| 4.2.1 Création de Produit (Admin) . . . . .              | 17        |
| 4.2.2 Liste des Produits . . . . .                       | 19        |
| 4.2.3 Modification de Produit (Admin) . . . . .          | 19        |
| 4.2.4 Suppression de Produit (Admin) . . . . .           | 20        |
| 4.2.5 Détails d'un Produit . . . . .                     | 20        |
| 4.3 Recherche et Filtrage Avancés . . . . .              | 21        |

|          |  |           |
|----------|--|-----------|
| 4.3.1    | Recherche Textuelle . . . . .                          | 21        |
| 4.3.2    | Filtrage Multi-Critères . . . . .                      | 21        |
| 4.3.3    | Tri des Résultats . . . . .                            | 22        |
| 4.3.4    | Pagination . . . . .                                   | 22        |
| 4.4      | Upload de Fichiers . . . . .                           | 23        |
| 4.4.1    | Upload d'Images Produits . . . . .                     | 23        |
| 4.5      | Dashboard Administrateur . . . . .                     | 24        |
| 4.5.1    | Statistiques Globales . . . . .                        | 24        |
| 4.5.2    | Distribution par Catégorie . . . . .                   | 25        |
| 4.5.3    | Alertes Stock . . . . .                                | 25        |
| <b>5</b> | <b>Interface Utilisateur et Expérience Utilisateur</b> | <b>26</b> |
| 5.1      | Design System . . . . .                                | 26        |
| 5.1.1    | Palette de Couleurs . . . . .                          | 26        |
| 5.1.2    | Typographie . . . . .                                  | 26        |
| 5.2      | Responsive Design . . . . .                            | 26        |
| 5.3      | Navigation . . . . .                                   | 26        |
| 5.3.1    | Barre de Navigation . . . . .                          | 26        |
| 5.3.2    | Page d'Accueil . . . . .                               | 27        |
| 5.4      | Animations et Transitions . . . . .                    | 27        |
| 5.5      | Feedback Visuel . . . . .                              | 28        |
| 5.5.1    | États de Chargement . . . . .                          | 28        |
| 5.5.2    | Messages de Succès et d'Erreur . . . . .               | 28        |
| <b>6</b> | <b>Backend et API REST</b>                             | <b>29</b> |
| 6.1      | Architecture Backend . . . . .                         | 29        |
| 6.2      | Endpoints Principaux . . . . .                         | 29        |
| 6.2.1    | Routes d'Authentification . . . . .                    | 29        |
| 6.2.2    | Routes des Produits . . . . .                          | 29        |
| 6.3      | Tests avec Postman . . . . .                           | 29        |
| 6.4      | Sécurité de l'API . . . . .                            | 30        |
| 6.4.1    | Authentification JWT . . . . .                         | 30        |
| 6.4.2    | Validation des Données . . . . .                       | 30        |
| 6.4.3    | Protection contre les Vulnérabilités . . . . .         | 30        |
| <b>7</b> | <b>Déploiement et Tests</b>                            | <b>31</b> |
| 7.1      | Architecture de Déploiement . . . . .                  | 31        |
| 7.2      | Configuration du Déploiement . . . . .                 | 31        |
| 7.2.1    | Backend sur Render . . . . .                           | 31        |
| 7.2.2    | Frontend sur Vercel . . . . .                          | 31        |
| 7.3      | Application Déployée . . . . .                         | 32        |
| 7.4      | Tests Réalisés . . . . .                               | 32        |
| 7.4.1    | Tests Fonctionnels . . . . .                           | 32        |
| 7.4.2    | Tests de Sécurité . . . . .                            | 32        |
| 7.4.3    | Tests de Performance . . . . .                         | 32        |
| 7.5      | Monitoring . . . . .                                   | 33        |
| 7.5.1    | Outils de Monitoring . . . . .                         | 33        |

|   |           |
|---|-----------|
| <b>8 Conclusion et Perspectives</b>                 | <b>34</b> |
| 8.1 Bilan du Projet . . . . .                       | 34        |
| 8.1.1 Objectifs Atteints . . . . .                  | 34        |
| 8.1.2 Compétences Acquises . . . . .                | 34        |
| 8.2 Défis Rencontrés . . . . .                      | 35        |
| 8.2.1 Défis Techniques . . . . .                    | 35        |
| 8.2.2 Solutions Apportées . . . . .                 | 35        |
| 8.3 Perspectives et Améliorations Futures . . . . . | 35        |
| 8.3.1 Fonctionnalités Utilisateur . . . . .         | 35        |
| 8.3.2 Fonctionnalités Administrateur . . . . .      | 35        |
| 8.3.3 Améliorations Techniques . . . . .            | 36        |
| 8.3.4 Améliorations Sécurité . . . . .              | 36        |
| 8.4 Conclusion Finale . . . . .                     | 36        |
| <b>A Code Source Important</b>                      | <b>38</b> |
| A.1 Schéma Mongoose User . . . . .                  | 38        |
| A.2 Middleware d'Authentification JWT . . . . .     | 39        |
| <b>B Commandes de Déploiement</b>                   | <b>41</b> |
| B.1 Commandes Git . . . . .                         | 41        |
| B.2 Déploiement Backend (Render) . . . . .          | 41        |
| B.3 Déploiement Frontend (Vercel) . . . . .         | 41        |

# Table des figures

|      |  |    |
|------|--|----|
| 2.1  | Diagramme de cas d'utilisation du système . . . . .  | 9  |
| 2.2  | Schéma de la base de données MongoDB . . . . .       | 10 |
| 3.1  | Architecture client-serveur en trois tiers . . . . . | 12 |
| 4.1  | Page d'inscription avec validation . . . . .         | 16 |
| 4.2  | Page de connexion . . . . .                          | 17 |
| 4.3  | Formulaire de création de produit . . . . .          | 18 |
| 4.4  | Liste des produits avec filtres . . . . .            | 19 |
| 4.5  | Formulaire de modification de produit . . . . .      | 20 |
| 4.6  | Page de détails d'un produit . . . . .               | 21 |
| 4.7  | Résultats de recherche . . . . .                     | 21 |
| 4.8  | Panneau de filtres . . . . .                         | 22 |
| 4.9  | Système de pagination . . . . .                      | 23 |
| 4.10 | Upload d'image avec aperçu . . . . .                 | 24 |
| 4.11 | Dashboard administrateur . . . . .                   | 25 |
| 5.1  | Barre de navigation . . . . .                        | 27 |
| 5.2  | Page d'accueil . . . . .                             | 27 |

# Chapitre 1

## Introduction et Contexte

### 1.1 Présentation du Projet

Dans le cadre du module **Sécurité des Applications Web Modernes**, ce projet vise à concevoir et développer une application web Full Stack complète de gestion de produits e-commerce. L'objectif principal est de mettre en pratique les concepts de sécurité avancés enseignés durant la formation, tout en démontrant une maîtrise des technologies web modernes.

L'application permet aux utilisateurs de consulter un catalogue de produits avec des fonctionnalités avancées de recherche et de filtrage, tandis que les administrateurs disposent d'un système complet de gestion (CRUD) des produits et d'un tableau de bord statistique.

### 1.2 Problématique

Les applications web e-commerce sont constamment ciblées par des attaques malveillantes (injection SQL/NoSQL, XSS, CSRF, vol de sessions). Il est donc crucial d'implémenter des mécanismes de sécurité robustes dès la conception pour protéger les données des utilisateurs et garantir l'intégrité du système.

La problématique centrale de ce projet est : *Comment développer une application web moderne, performante et sécurisée qui respecte les standards de sécurité OWASP et offre une expérience utilisateur optimale ?*

### 1.3 Objectifs du Projet

Les objectifs de ce projet sont multiples :

1. **Sécurité** : Implémenter une authentification sécurisée (JWT), un hashage des mots de passe (bcrypt), une gestion des rôles, et protéger contre les vulnérabilités OWASP.
2. **Fonctionnalités** : Développer un système CRUD complet avec recherche, filtrage, tri, pagination et upload de fichiers.
3. **Architecture** : Concevoir une architecture client-serveur en trois tiers avec séparation claire des responsabilités.
4. **Interface** : Créer une interface utilisateur moderne, responsive et intuitive avec TailwindCSS.
5. **Déploiement** : Déployer l'application en production sur des services cloud (Render, Vercel, MongoDB Atlas).
6. **Documentation** : Rédiger une documentation technique complète et professionnelle.

## 1.4 Cahier des Charges Résumé

Le cahier des charges du projet définit les exigences fonctionnelles et non-fonctionnelles suivantes :

### 1.4.1 Exigences Fonctionnelles

- **Authentification** : Inscription, connexion, déconnexion avec JWT
- **Gestion des utilisateurs** : Deux rôles (admin, user) avec permissions différentes
- **Gestion des produits** : CRUD complet (Créer, Lire, Modifier, Supprimer)
- **Recherche avancée** : Par nom, description, catégorie, prix
- **Filtrage** : Par catégorie, plage de prix, disponibilité en stock
- **Tri** : Par prix, nom, date de création
- **Pagination** : Navigation par pages (12 produits/page)
- **Upload** : Téléchargement d'images pour les produits
- **Dashboard** : Statistiques pour les administrateurs

### 1.4.2 Exigences Non-Fonctionnelles

- **Sécurité** : Protection contre OWASP Top 10, chiffrement des données sensibles
- **Performance** : Temps de réponse < 2 secondes
- **Scalabilité** : Architecture permettant la montée en charge
- **Maintenabilité** : Code propre, modulaire et documenté
- **Responsive** : Compatible mobile, tablette et desktop
- **Disponibilité** : Application accessible 24/7 en production

## 1.5 Méthodologie de Travail

Le développement du projet a suivi une approche **itérative et incrémentale** avec les étapes suivantes :

1. **Analyse et conception** : Définition des besoins, modélisation de la base de données, conception de l'architecture
2. **Développement backend** : Création des modèles, controllers, middlewares et routes
3. **Développement frontend** : Création des composants React, pages et gestion d'état
4. **Tests** : Tests manuels avec Postman, tests d'intégration
5. **Déploiement** : Mise en production sur les plateformes cloud
6. **Documentation** : Rédaction de la documentation technique

Cette méthodologie a permis de livrer un produit fonctionnel à chaque itération, tout en maintenant un code de qualité et sécurisé.

# Chapitre 2

## Analyse et Conception

### 2.1 Analyse des Besoins

#### 2.1.1 Besoins Fonctionnels

L'analyse des besoins a permis d'identifier les fonctionnalités essentielles à implémenter :

| ID  | Fonctionnalité                                   | Priorité |
|-----|--|----------|
| F1  | Inscription et authentification des utilisateurs | Haute    |
| F2  | Gestion des rôles (admin/user)                   | Haute    |
| F3  | CRUD complet des produits (admin)                | Haute    |
| F4  | Consultation des produits (public)               | Haute    |
| F5  | Recherche textuelle dans produits                | Moyenne  |
| F6  | Filtrage par catégorie et prix                   | Moyenne  |
| F7  | Tri des résultats                                | Moyenne  |
| F8  | Pagination des résultats                         | Moyenne  |
| F9  | Upload d'images produits                         | Moyenne  |
| F10 | Dashboard avec statistiques (admin)              | Basse    |

TABLE 2.1 – Liste des besoins fonctionnels

#### 2.1.2 Identification des Acteurs

Le système interagit avec deux types d'acteurs principaux :

- **Utilisateur simple (user)** : Peut consulter les produits, effectuer des recherches et des filtrages, mais ne peut pas modifier le catalogue.
- **Administrateur (admin)** : Dispose de tous les droits de l'utilisateur simple, plus la possibilité de créer, modifier et supprimer des produits, ainsi que d'accéder au tableau de bord statistique.

### 2.2 Diagramme de Cas d'Utilisation

Le diagramme de cas d'utilisation ci-dessous illustre les interactions entre les acteurs et le système :

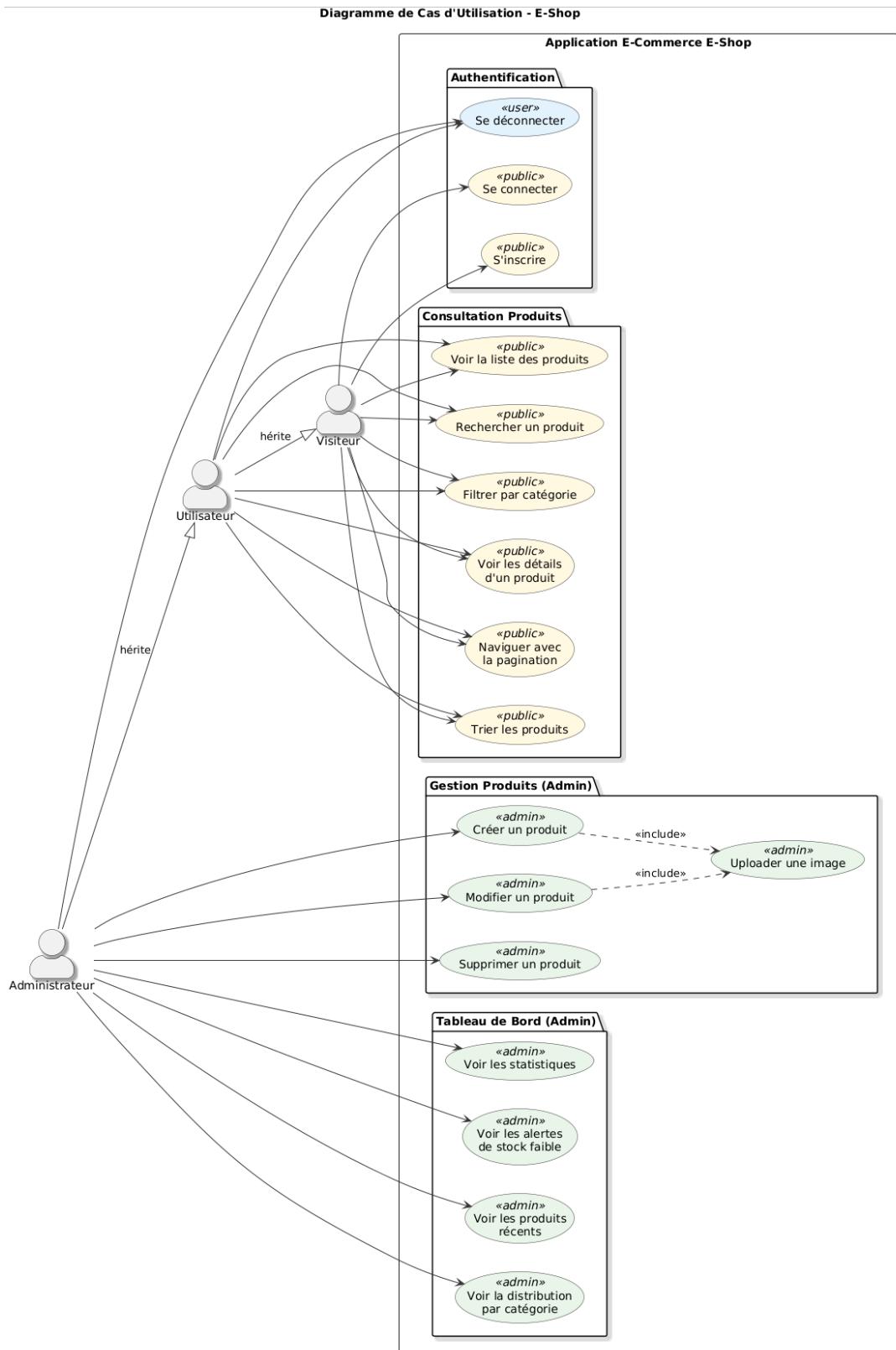


FIGURE 2.1 – Diagramme de cas d'utilisation du système

## 2.3 Modélisation de la Base de Données

### 2.3.1 Schéma Relationnel

La base de données MongoDB est conçue avec deux collections principales :

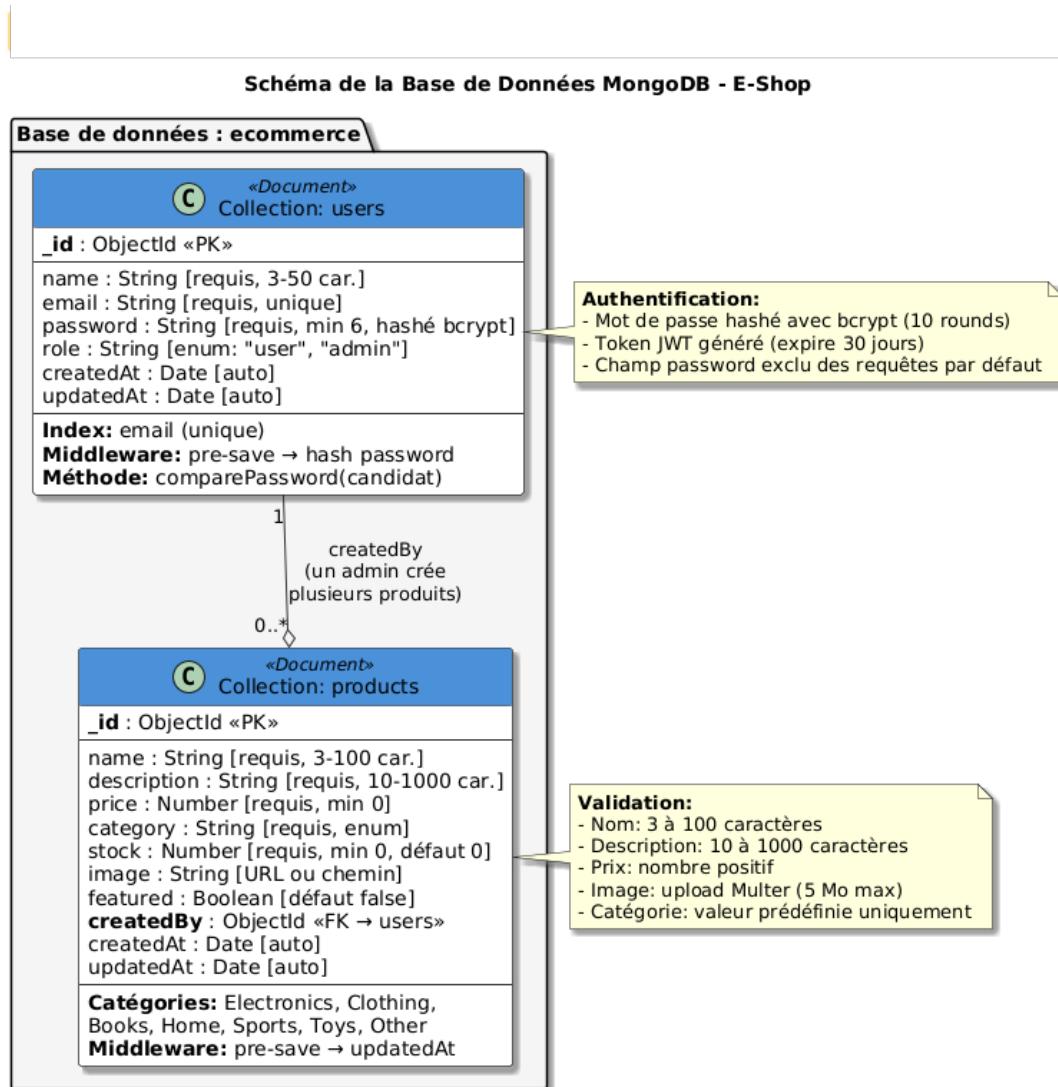


FIGURE 2.2 – Schéma de la base de données MongoDB

### 2.3.2 Description des Collections

#### Collection Users

| Champ     | Type     | Description                                       |
|-----------|----------|---|
| _id       | ObjectId | Identifiant unique (auto-généré)                  |
| name      | String   | Nom complet de l'utilisateur (2-50 caractères)    |
| email     | String   | Email unique, validé par regex                    |
| password  | String   | Mot de passe hashé avec bcrypt (salt rounds : 10) |
| role      | String   | Rôle (enum : "user", "admin"), défaut : "user"    |
| createdAt | Date     | Date de création (auto-généré)                    |
| updatedAt | Date     | Date de dernière modification (auto-généré)       |

TABLE 2.2 – Structure de la collection Users

#### Collection Products

| Champ       | Type     | Description                                 |
|-------------|----------|---|
| _id         | ObjectId | Identifiant unique (auto-généré)            |
| name        | String   | Nom du produit (3-100 caractères)           |
| description | String   | Description détaillée (10-1000 caractères)  |
| price       | Number   | Prix en euros (minimum 0.01)                |
| category    | String   | Catégorie (enum : 7 valeurs prédéfinies)    |
| stock       | Number   | Quantité en stock (entier $\geq 0$ )        |
| image       | String   | Chemin de l'image uploadée                  |
| createdBy   | ObjectId | Référence vers l'utilisateur créateur       |
| createdAt   | Date     | Date de création (auto-généré)              |
| updatedAt   | Date     | Date de dernière modification (auto-généré) |

TABLE 2.3 – Structure de la collection Products

## 2.4 Contraintes de Sécurité

Plusieurs contraintes de sécurité ont été intégrées au niveau de la conception :

- **Validation des données** : Validation stricte à deux niveaux (client et serveur)
- **Hashage des mots de passe** : Utilisation de bcrypt avec 10 salt rounds
- **Authentification stateless** : JWT avec expiration de 30 jours
- **Contrôle d'accès** : Middleware d'autorisation basé sur les rôles
- **Protection injection** : Utilisation de Mongoose avec validation de schémas
- **Protection XSS** : Sanitization avec Express Validator et échappement React
- **Upload sécurisé** : Validation du type MIME, limite de taille (5MB)

# Chapitre 3

## Architecture et Technologies

### 3.1 Architecture Globale

L'application suit une architecture **client-serveur en trois tiers** qui sépare clairement les responsabilités :

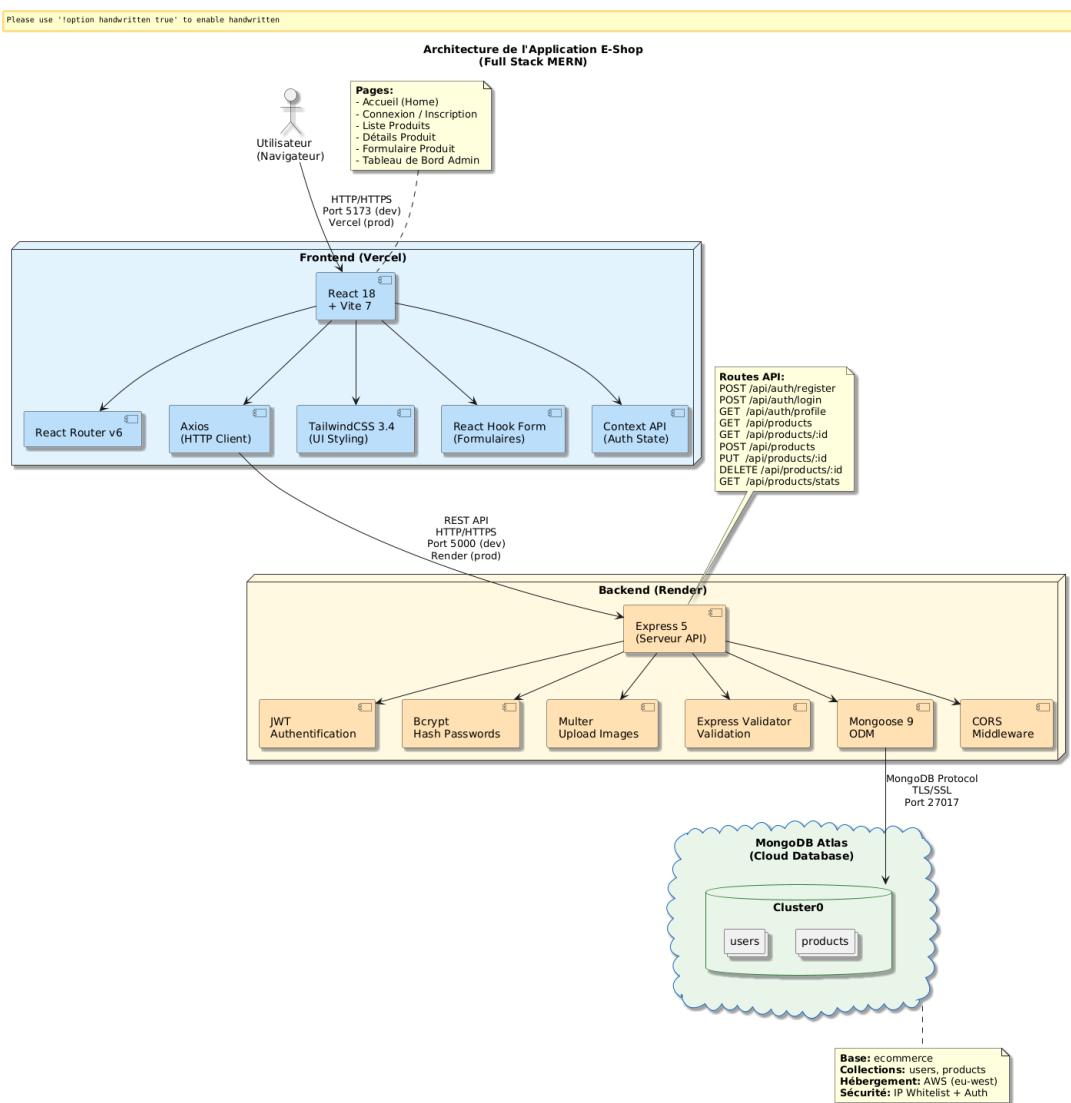


FIGURE 3.1 – Architecture client-serveur en trois tiers

#### 3.1.1 Tier 1 : Couche Présentation (Frontend)

**Technologies :** React.js 18.3.1, Vite 7.3.1, TailwindCSS 3.4.1

**Responsabilités :**

- Interface utilisateur interactive et responsive

- Gestion de l'état de l'application (Context API)
- Validation côté client (React Hook Form)
- Communication avec le backend via Axios
- Routage côté client (React Router v6)

### 3.1.2 Tier 2 : Couche Logique Métier (Backend)

**Technologies :** Node.js, Express.js 5.2.1, Mongoose 9.1.5

**Responsabilités :**

- API RESTful avec endpoints sécurisés
- Authentification JWT et autorisation RBAC
- Validation des données (Express Validator)
- Logique métier et règles de gestion
- Gestion des fichiers uploadés (Multer)
- Interaction avec la base de données

### 3.1.3 Tier 3 : Couche Données (Database)

**Technologies :** MongoDB Atlas (Cloud)

**Responsabilités :**

- Stockage persistant des données
- Indexation pour performance
- Agrégations pour statistiques
- Backup automatique
- Chiffrement au repos et en transit

## 3.2 Stack Technologique

### 3.2.1 Justification des Choix

Le choix des technologies a été guidé par plusieurs critères :

| Technologie | Rôle             | Justification   |
|-------------|------------------|---|
| React 18    | Frontend Library | Composants réutilisables, Virtual DOM, Large écosystème |
| Vite        | Build Tool       | HMR rapide, Build optimisé, Configuration minimale      |
| TailwindCSS | Styling          | Utility-first, Responsive, Purge CSS automatique        |
| Node.js     | Backend Runtime  | JavaScript full-stack, Asynchrone, NPM                  |
| Express     | Web Framework    | Minimaliste, Middleware, Large communauté               |
| MongoDB     | Database NoSQL   | Flexible, Scalable, Cloud-ready                         |
| JWT         | Authentication   | Stateless, Self-contained, Standard                     |
| Bcrypt      | Password Hashing | Salt automatique, Résistant brute force                 |

TABLE 3.1 – Technologies utilisées et justifications

### 3.2.2 Avantages de l'Architecture Choisie

#### Séparation des préoccupations :

- Développement parallèle frontend/backend possible
- Facilite la maintenance et les tests
- Permet le remplacement d'une couche sans impacter les autres

#### Scalabilité :

- Frontend et backend peuvent être scalés indépendamment
- MongoDB Atlas offre le sharding horizontal
- Possibilité d'ajouter un load balancer

#### Sécurité :

- Séparation physique des couches
- Backend non exposé directement (API only)
- Chiffrement des communications (HTTPS)

## 3.3 Structure du Code

### 3.3.1 Structure Backend

```

1 backend/
2 |-- src/
3 |   |-- config/
4 |   |   |-- database.js      # Connexion MongoDB
5 |   |   +-- seed.js        # Peuplement BD
6 |   |-- models/
7 |   |   |-- User.js         # Schema utilisateur
8 |   |   +-- Product.js      # Schema produit
9 |   |-- controllers/
10 |   |   |-- authController.js
11 |   |   +-- productController.js
12 |   |-- middleware/
13 |   |   |-- auth.js          # JWT verification
14 |   |   |-- validator.js     # Validation
15 |   |   +-- upload.js        # Multer config
16 |   +-- routes/
17 |   |   |-- authRoutes.js
18 |   |   +-- productRoutes.js
19 |-- uploads/                      # Fichiers uploadés
20 |-- server.js                     # Point d'entrée
21 |-- package.json
22 +-- .env                          # Variables d'environnement

```

Listing 3.1 – Arborescence du backend

### 3.3.2 Structure Frontend

```
1 frontend/
2 |-- src/
3 |   |-- components/
4 |   |   |-- Navbar.jsx
5 |   |   |-- Footer.jsx
6 |   |   |-- ProductCard.jsx
7 |   |   +-- ProtectedRoute.jsx
8 |   |-- pages/
9 |   |   |-- Home.jsx
10 |   |   |-- Login.jsx
11 |   |   |-- Register.jsx
12 |   |   |-- Products.jsx
13 |   |   |-- ProductDetail.jsx
14 |   |   |-- ProductForm.jsx
15 |   |   +-- Dashboard.jsx
16 |   |-- context/
17 |   |   +-- AuthContext.jsx
18 |   |-- utils/
19 |   |   +-- axios.js
20 |   |-- App.jsx
21 |   |-- main.jsx
22 |   +-- index.css
23 |-- package.json
24 +-- .env
```

Listing 3.2 – Arborescence du frontend

# Chapitre 4

## Fonctionnalités Détaillées

### 4.1 Authentification et Autorisation

#### 4.1.1 Inscription des Utilisateurs

L'application permet aux nouveaux utilisateurs de créer un compte avec validation en temps réel.

##### Processus d'inscription :

1. L'utilisateur remplit le formulaire (nom, email, mot de passe, rôle)
2. Validation côté client (React Hook Form)
3. Envoi des données au backend
4. Validation côté serveur (Express Validator)
5. Hashage du mot de passe avec bcrypt
6. Création de l'utilisateur dans MongoDB
7. Génération d'un token JWT
8. Redirection vers le dashboard

The screenshot shows the 'Inscription' (Registration) page of an E-Shop application. At the top, there is a navigation bar with links for 'Accueil', 'Produits', 'Connexion', and a blue 'Inscription' button. The main content area has a light gray background and features a white registration form. The form is titled 'Inscription'. It contains four input fields: 'Nom' (Name), 'E-mail' (Email), 'Mot de passe' (Password), and 'Confirmer le Mot de Passe' (Confirm Password). Below the password fields, a red error message 'Please fill out this field.' is displayed. A large blue 'S'inscrire' (Sign Up) button is located at the bottom of the form. At the very bottom of the page, there is a link 'Vous avez déjà un compte ? Connectez-vous ici' (Already have an account? Log in here).

FIGURE 4.1 – Page d'inscription avec validation

#### 4.1.2 Connexion Sécurisée

La connexion utilise JWT pour une authentification stateless et sécurisée.

##### Mécanisme de connexion :

- Saisie email et mot de passe

- Recherche de l'utilisateur par email
- Comparaison du mot de passe avec bcrypt.compare()
- Génération d'un token JWT (expiration 30 jours)
- Stockage du token dans localStorage
- Ajout automatique du token aux requêtes (Axios interceptor)

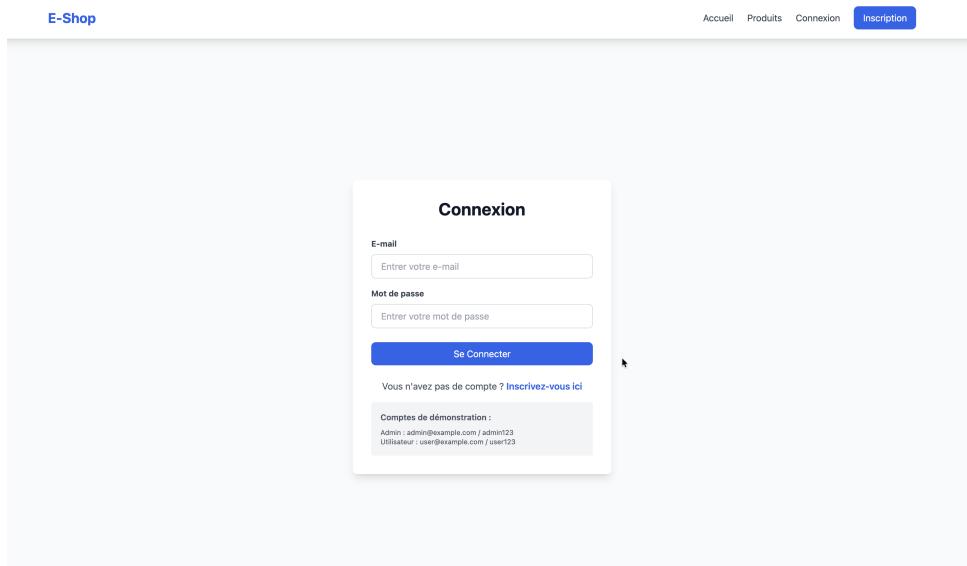


FIGURE 4.2 – Page de connexion

### 4.1.3 Gestion des Rôles (RBAC)

L'application implémente un système de contrôle d'accès basé sur les rôles :

| Rôle         | Permissions  | Restrictions  |
|--------------|--|---|
| <b>user</b>  | <ul style="list-style-type: none"> <li>— Consulter les produits</li> <li>— Rechercher et filtrer</li> <li>— Voir les détails</li> </ul>  | <ul style="list-style-type: none"> <li>— Ne peut pas créer de produits</li> <li>— Pas d'accès au dashboard</li> </ul> |
| <b>admin</b> | <ul style="list-style-type: none"> <li>— Toutes permissions "user"</li> <li>— Créer des produits</li> <li>— Modifier des produits</li> <li>— Supprimer des produits</li> <li>— Accès au dashboard</li> </ul> | <ul style="list-style-type: none"> <li>— Responsabilité complète sur le catalogue</li> </ul>                          |

TABLE 4.1 – Permissions par rôle

## 4.2 Gestion des Produits (CRUD)

### 4.2.1 Crédit de Produit (Admin)

Les administrateurs peuvent ajouter de nouveaux produits avec upload d'image.

## Créer un Nouveau Produit

**Nom du Produit \***

**Description \***

**Prix (€) \***

**Stock \***

**Catégorie \***

**Image du Produit**

No file chosen

Marquer comme En Vedette

**Technologies** **Réalisé**

FIGURE 4.3 – Formulaire de création de produit

**Validation implémentée :**

- Nom : 3-100 caractères, obligatoire
- Description : 10-1000 caractères, obligatoire
- Prix : Nombre  $> 0$ , obligatoire
- Catégorie : 7 valeurs prédéfinies, obligatoire
- Stock : Entier  $\geq 0$ , obligatoire
- Image : JPEG/PNG/GIF/WEBP, max 5MB, optionnel

## 4.2.2 Liste des Produits

La page principale affiche tous les produits dans une grille responsive.

The screenshot shows the 'Produits' (Products) section of the E-Shop application. At the top, there is a navigation bar with links for Accueil (Home), Produits (Products), Tableau de Bord (Dashboard), Bienvenue, Admin User (Admin), Admin (Admin), and Déconnexion (Logout). Below the navigation is a search bar labeled 'Recherche' (Search) with placeholder text 'Rechercher des produits...', a category dropdown set to 'Tout' (All), a sorting dropdown set to 'Date' (Date) with 'Décroissant' (Descending) selected, and a button '+ Ajouter un Produit' (Add a Product).

The main area displays a grid of products:

- Lampe de Bureau LED**: A black desk lamp. Description: Lampe de bureau LED moderne avec luminosité et température de couleur réglables. Price: 45.99 €. Stock: 60. Category: Maison. Actions: Voir Détails, Modifier, Supprimer.
- Introduction à la...**: A stack of books. Description: Manuel complet sur les fondamentaux de la cybersécurité. Menaces, défenses et bonnes... Price: 49.99 €. Stock: 35. Category: Livres. Actions: Voir Détails, Modifier, Supprimer.
- Fauteuil de Bureau...**: An ergonomic office chair. Description: Fauteuil de bureau ergonomique avec support lombaire et hauteur réglable. Confortable pour d... Price: 249.99 €. Stock: 30. Category: Maison. Actions: Voir Détails, Modifier, Supprimer.
- Cafetière Programmable**: A coffee maker. Description: Cafetière programmable avec carafe thermique. Prépare 12 tasses de café parfait. Price: 129.99 €. Stock: 35. Category: Maison. Actions: Voir Détails, Modifier, Supprimer.
- Bookshelf**: A bookshelf filled with books. Description: Not visible in the image.
- Couple in Kitchen**: A couple in a kitchen. Description: Not visible in the image.
- Books on Shelf**: A stack of books. Description: Not visible in the image.
- Yellow Headboard**: A yellow upholstered headboard. Description: Not visible in the image.

FIGURE 4.4 – Liste des produits avec filtres

## 4.2.3 Modification de Produit (Admin)

Les administrateurs peuvent modifier les informations d'un produit existant.

## Créer un Nouveau Produit

**Nom du Produit \***

**Description \***

**Prix (€) \***

**Stock \***

**Catégorie \***

**Image du Produit**

Choose File b4561bf230b22bcdffeb538bca90fe77.jpg



Marquer comme En Vedette

**Créer le Produit** **Annuler**

FIGURE 4.5 – Formulaire de modification de produit

### 4.2.4 Suppression de Produit (Admin)

La suppression nécessite une confirmation pour éviter les erreurs. Un dialogue de confirmation s'affiche avec les boutons Annuler et Confirmer avant toute suppression définitive.

### 4.2.5 Détails d'un Produit

Une page dédiée affiche toutes les informations d'un produit.

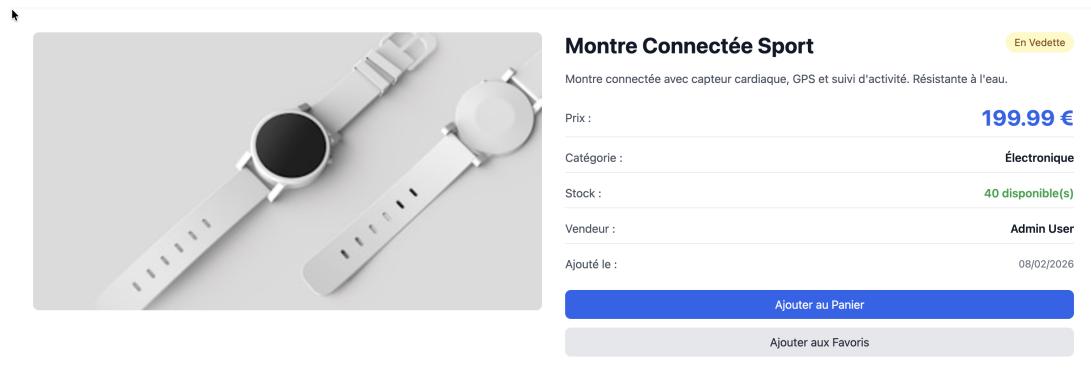


FIGURE 4.6 – Page de détails d'un produit

## 4.3 Recherche et Filtrage Avancés

### 4.3.1 Recherche Textuelle

La recherche s'effectue simultanément dans le nom et la description des produits.  
**Implémentation :**

- Recherche insensible à la casse
- Utilisation de regex MongoDB
- Recherche en temps réel (debounce 300ms)
- Mise en évidence des résultats

FIGURE 4.7 – Résultats de recherche

### 4.3.2 Filtrage Multi-Critères

Les utilisateurs peuvent combiner plusieurs filtres :

- **Par catégorie** : Dropdown avec 7 catégories
- **Par plage de prix** : Sliders min/max
- **Par disponibilité** : Checkbox "En stock uniquement"

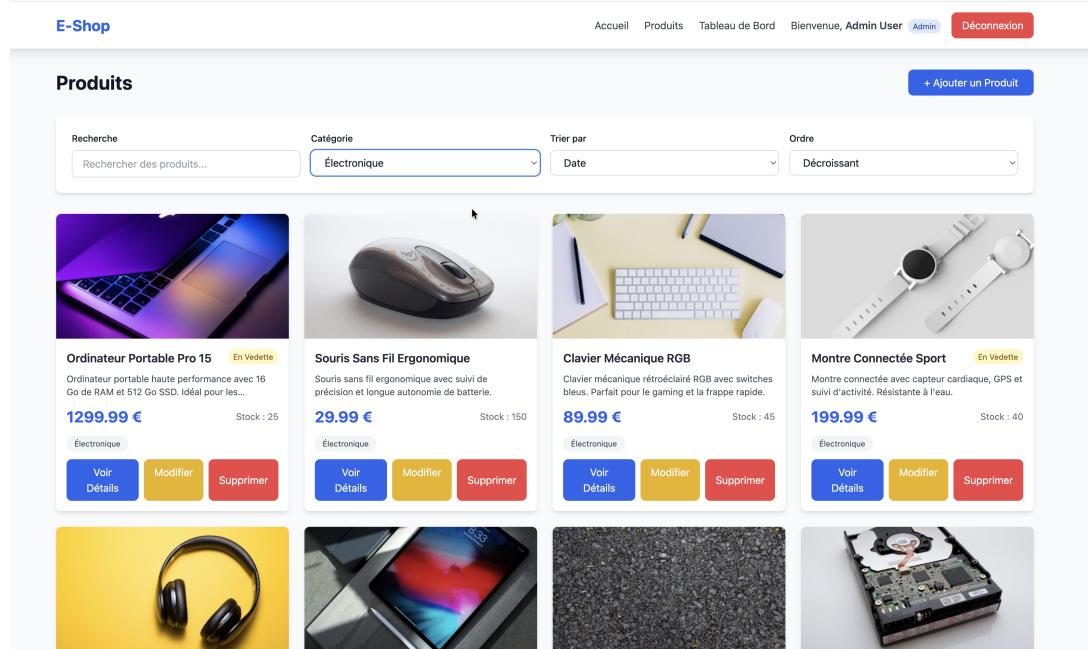


FIGURE 4.8 – Panneau de filtres

### 4.3.3 Tri des Résultats

Plusieurs options de tri sont disponibles :

| Option           | Description                   |
|------------------|-------------------------------|
| Prix croissant   | Du moins cher au plus cher    |
| Prix décroissant | Du plus cher au moins cher    |
| Nom A-Z          | Ordre alphabétique            |
| Nom Z-A          | Ordre alphabétique inverse    |
| Plus récents     | Date de création décroissante |
| Plus anciens     | Date de création croissante   |

TABLE 4.2 – Options de tri disponibles

### 4.3.4 Pagination

La pagination permet de naviguer efficacement dans un grand catalogue.  
**Caractéristiques :**

- 12 produits par page (configurable)
- Numéros de pages cliquables
- Boutons Précédent/Suivant
- Affichage "Page X sur Y"
- Compteur total de résultats

- Conservation des filtres entre les pages

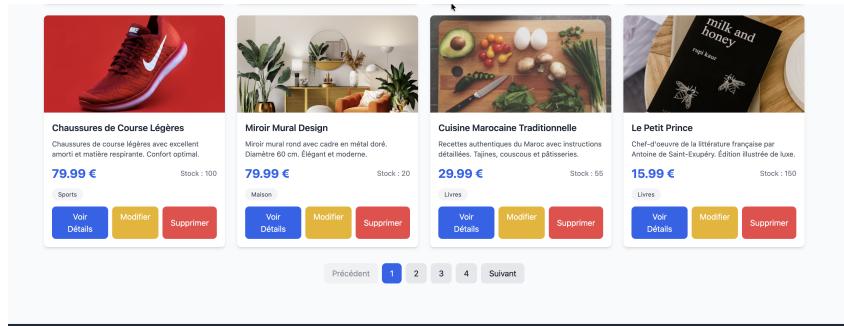


FIGURE 4.9 – Système de pagination

## 4.4 Upload de Fichiers

### 4.4.1 Upload d’Images Produits

L’application permet l’upload sécurisé d’images pour les produits.

**Sécurité implémentée :**

- Vérification du type MIME (pas juste l’extension)
- Limite de taille : 5 MB
- Types autorisés : JPEG, JPG, PNG, GIF, WEBP
- Renommage automatique (timestamp + random)
- Stockage dans dossier dédié (/uploads)

### Créer un Nouveau Produit

Nom du Produit \*

Description \*

Prix (€) \*

Stock \*

Catégorie \*

Image du Produit

Choose File b4561bf230b22bcdffeb538bca90fe77.jpg

Marquer comme En Vedette

**Créer le Produit**   **Annuler**

FIGURE 4.10 – Upload d'image avec aperçu

## 4.5 Dashboard Administrateur

### 4.5.1 Statistiques Globales

Le dashboard affiche des statistiques en temps réel sur le catalogue.

#### Métriques affichées :

- Nombre total de produits
- Valeur totale du stock (prix × quantité)
- Prix moyen des produits
- Stock total (somme des quantités)
- Nombre de produits en rupture de stock
- Nombre de produits à stock faible (< 10 unités)

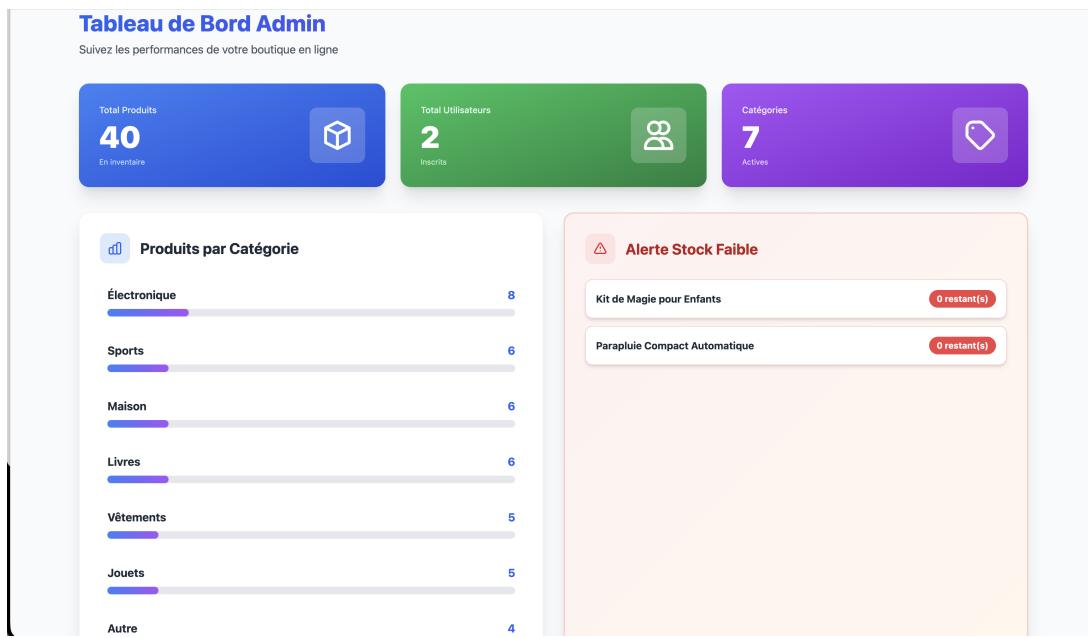


FIGURE 4.11 – Dashboard administrateur

#### 4.5.2 Distribution par Catégorie

Un tableau affiche la répartition des produits par catégorie avec le nombre de produits et la valeur totale pour chaque catégorie.

#### 4.5.3 Alertes Stock

Le dashboard signale les produits nécessitant un réapprovisionnement.

- **Rupture de stock :** Badge rouge pour stock = 0
- **Stock faible :** Badge orange pour stock < 10
- Liste déroulante des produits concernés
- Action rapide vers la page de modification

# Chapitre 5

## Interface Utilisateur et Expérience Utilisateur

### 5.1 Design System

#### 5.1.1 Palette de Couleurs

L'application utilise une palette de couleurs moderne et cohérente :

| Couleur           | Code Hex | Utilisation                         |
|-------------------|----------|-------------------------------------|
| Bleu primaire     | #3B82F6  | Boutons principaux, liens           |
| Violet secondaire | #8B5CF6  | Accents, gradients                  |
| Vert succès       | #10B981  | Messages de succès, badges positifs |
| Rouge danger      | #EF4444  | Erreurs, suppressions               |
| Gris neutre       | #6B7280  | Texte secondaire, bordures          |

TABLE 5.1 – Palette de couleurs de l'application

#### 5.1.2 Typographie

- **Police principale** : Inter, system-ui, sans-serif
- **Tailles** : Adaptatives de 0.875rem (14px) à 2.25rem (36px)
- **Poids** : Normal (400), Medium (500), Bold (700)

### 5.2 Responsive Design

L'application est entièrement responsive avec trois breakpoints principaux :

| Appareil | Largeur        | Disposition                |
|----------|----------------|----------------------------|
| Mobile   | < 640px        | 1 colonne, menu burger     |
| Tablette | 640px - 1024px | 2 colonnes, menu condensé  |
| Desktop  | > 1024px       | 3-4 colonnes, menu complet |

TABLE 5.2 – Breakpoints responsive

### 5.3 Navigation

#### 5.3.1 Barre de Navigation

La navbar est présente sur toutes les pages et s'adapte selon l'état de connexion.

Éléments affichés :

- Logo de l'application (lien vers accueil)
- Lien "Produits"
- Si connecté : Lien "Dashboard" (admin uniquement)
- Si connecté : Menu utilisateur avec nom et déconnexion
- Si non connecté : Boutons "Connexion" et "Inscription"



FIGURE 5.1 – Barre de navigation

### 5.3.2 Page d'Accueil

La page d'accueil présente l'application avec un design moderne.

#### Sections :

- Hero section avec gradient animé
- Description de l'application
- Bouton CTA "Voir les Produits"
- Liste des fonctionnalités principales

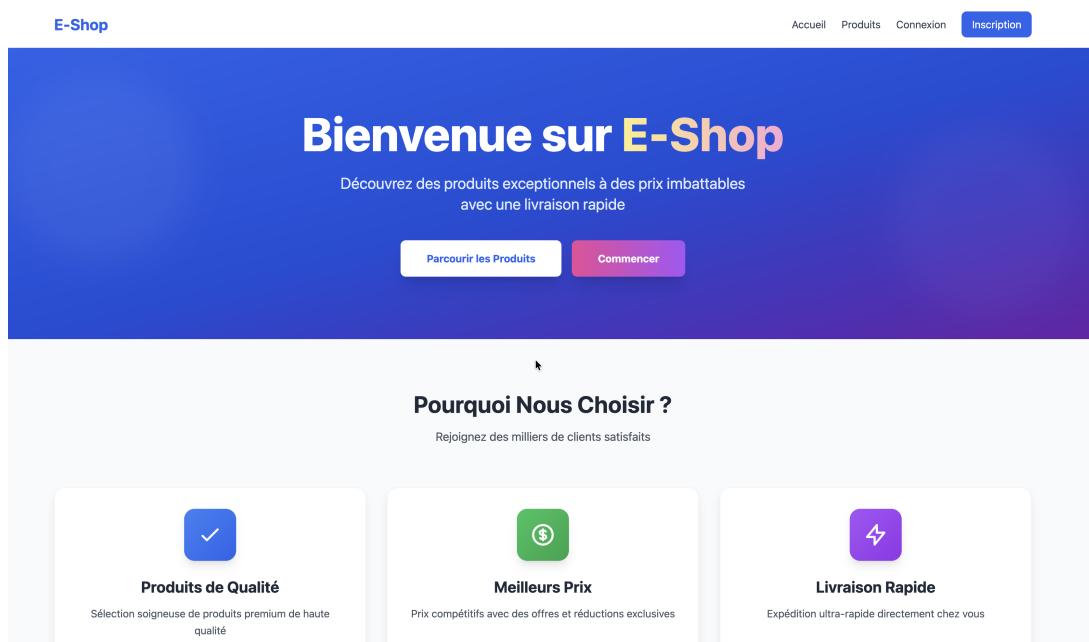


FIGURE 5.2 – Page d'accueil

## 5.4 Animations et Transitions

L'application utilise des animations subtiles pour améliorer l'UX :

- **Hover effects** : Scale et shadow sur les cartes produits
- **Fade-in** : Apparition progressive des éléments au chargement
- **Slide-in** : Modales qui glissent depuis le haut
- **Transitions fluides** : 300ms avec easing naturel

## 5.5 Feedback Visuel

### 5.5.1 États de Chargement

Plusieurs indicateurs informent l'utilisateur durant les opérations asynchrones :

- Spinners animés lors du chargement des données
- Skeleton loaders pour les cartes produits
- Texte "Chargement..." explicite
- Désactivation des boutons durant les requêtes

### 5.5.2 Messages de Succès et d'Erreur

**Messages de succès :**

- Toast notifications (coin supérieur droit)
- Icône verte avec checkmark
- Auto-dismiss après 3 secondes

**Messages d'erreur :**

- Affichage sous les champs concernés
- Couleur rouge avec icône d'alerte
- Bordure rouge sur les inputs invalides

# Chapitre 6

## Backend et API REST

### 6.1 Architecture Backend

Le backend suit une architecture MVC (Model-View-Controller) adaptée aux API REST :

- **Models** : Définition des schémas de données (User, Product)
- **Controllers** : Logique métier et traitement des requêtes
- **Routes** : Définition des endpoints et liaison aux controllers
- **Middleware** : Authentification, autorisation, validation

### 6.2 Endpoints Principaux

#### 6.2.1 Routes d'Authentification

| Méthode | Endpoint           | Description                         |
|---------|--------------------|-------------------------------------|
| POST    | /api/auth/register | Inscription d'un nouvel utilisateur |
| POST    | /api/auth/login    | Connexion et génération du JWT      |
| GET     | /api/auth/me       | Récupération du profil utilisateur  |

TABLE 6.1 – Routes d'authentification

#### 6.2.2 Routes des Produits

| Méthode | Endpoint                      | Description                      | Accès  |
|---------|-------------------------------|----------------------------------|--------|
| GET     | /api/products                 | Liste avec filtres et pagination | Public |
| GET     | /api/products/ :id            | Détails d'un produit             | Public |
| POST    | /api/products                 | Création d'un produit            | Admin  |
| PUT     | /api/products/ :id            | Modification d'un produit        | Admin  |
| DELETE  | /api/products/ :id            | Suppression d'un produit         | Admin  |
| GET     | /api/products/stats/dashboard | Statistiques                     | Admin  |

TABLE 6.2 – Routes des produits

### 6.3 Tests avec Postman

L'API a été testée de manière exhaustive avec Postman. L'ensemble des 23 tests passés couvrent l'authentification, le CRUD produits, la pagination, le filtrage et la gestion des erreurs.

## 6.4 Sécurité de l'API

### 6.4.1 Authentification JWT

Toutes les routes protégées nécessitent un token JWT valide dans le header :

<sup>1</sup> Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

### 6.4.2 Validation des Données

Express Validator assure la validation stricte des entrées :

- Validation des types de données
- Vérification des formats (email, prix, etc.)
- Sanitization (trim, escape)
- Messages d'erreur explicites

### 6.4.3 Protection contre les Vulnérabilités

Mesures implémentées :

- **Injection NoSQL** : Validation avec Mongoose, pas de `$where`
- **XSS** : Sanitization avec Express Validator
- **CSRF** : Tokens JWT dans headers (pas de cookies)
- **Rate Limiting** : À implémenter pour production
- **CORS** : Configuration stricte des origines autorisées

# Chapitre 7

## Déploiement et Tests

### 7.1 Architecture de Déploiement

L'application est déployée sur trois services cloud distincts :

| Composant | Service       | URL   |
|-----------|---------------|---|
| Frontend  | Vercel        | <a href="https://[votre-app].vercel.app">https://[votre-app].vercel.app</a> |
| Backend   | Render        | <a href="https://[backend].onrender.com">https://[backend].onrender.com</a> |
| Database  | MongoDB Atlas | Cloud (Europe)  |

TABLE 7.1 – Services de déploiement

### 7.2 Configuration du Déploiement

#### 7.2.1 Backend sur Render

**Configuration :**

- **Build Command** : `npm install`
- **Start Command** : `npm start`
- **Environment** : Node
- **Instance Type** : Free (avec limitations)

**Variables d'environnement :**

- `NODE_ENV=production`
- `MONGODB_URI=mongodb+srv://...@cluster0.mongodb.net/ecommerce`
- `JWT_SECRET=[secret sécurisé]`
- `CLIENT_URL=https://websec-bice.vercel.app`

#### 7.2.2 Frontend sur Vercel

**Configuration :**

- **Framework** : Vite
- **Build Command** : `npm run build`
- **Output Directory** : dist

**Variables d'environnement :**

- `VITE_API_URL=https://redaelhadfii.onrender.com`

## 7.3 Application Déployée

L'application est accessible en production aux adresses suivantes :

- **Frontend (Vercel)** : <https://websec-bice.vercel.app/>
- **Backend API (Render)** : <https://redaelhadfii.onrender.com/>
- **Code source (GitHub)** : <https://github.com/redaelhadfii/websec/tree/main/fullstack-app>

## 7.4 Tests Réalisés

### 7.4.1 Tests Fonctionnels

Tous les scénarios utilisateurs ont été testés :

| Test | Description                         | Résultat     |
|------|-------------------------------------|--------------|
| T1   | Inscription avec email valide       | ✓ Réussi     |
| T2   | Inscription avec email existant     | ✓ Erreur 409 |
| T3   | Connexion avec credentials valides  | ✓ Réussi     |
| T4   | Connexion avec mauvais mot de passe | ✓ Erreur 401 |
| T5   | Création produit par admin          | ✓ Réussi     |
| T6   | Création produit par user           | ✓ Erreur 403 |
| T7   | Recherche de produits               | ✓ Réussi     |
| T8   | Filtrage par catégorie              | ✓ Réussi     |
| T9   | Upload image > 5MB                  | ✓ Erreur 413 |
| T10  | Pagination                          | ✓ Réussi     |

TABLE 7.2 – Résultats des tests fonctionnels

### 7.4.2 Tests de Sécurité

| Test | Description                     | Résultat     |
|------|---------------------------------|--------------|
| S1   | Accès route protégée sans token | ✓ Erreur 401 |
| S2   | Accès route admin par user      | ✓ Erreur 403 |
| S3   | Token JWT expiré                | ✓ Erreur 401 |
| S4   | Injection NoSQL dans login      | ✓ Bloqué     |
| S5   | Upload fichier .exe             | ✓ Bloqué     |
| S6   | XSS dans champ nom              | ✓ Échappé    |

TABLE 7.3 – Résultats des tests de sécurité

### 7.4.3 Tests de Performance

- **Temps de réponse API** : < 500ms (moyenne)
- **Temps de chargement page** : < 2 secondes
- **Lighthouse Score** : 90+ (Performance)

## 7.5 Monitoring

### 7.5.1 Outils de Monitoring

- **Render Dashboard** : Logs en temps réel, utilisation CPU/RAM
- **Vercel Analytics** : Nombre de visiteurs, temps de chargement
- **MongoDB Atlas** : Métriques de la base de données

# Chapitre 8

## Conclusion et Perspectives

### 8.1 Bilan du Projet

Ce projet de développement d'une application web Full Stack de gestion de produits e-commerce a permis de mettre en pratique l'ensemble des concepts de sécurité des applications web modernes enseignés dans le module.

#### 8.1.1 Objectifs Atteints

Tous les objectifs fixés en début de projet ont été atteints :

- ✓ **Authentification sécurisée** : JWT, bcrypt, expiration des tokens
- ✓ **Autorisation granulaire** : RBAC avec deux rôles distincts
- ✓ **CRUD complet** : Toutes les opérations implémentées
- ✓ **Recherche et filtrage** : Fonctionnalités avancées opérationnelles
- ✓ **Interface responsive** : Compatible mobile, tablette et desktop
- ✓ **Protection OWASP** : Injection, XSS, CSRF traités
- ✓ **Déploiement cloud** : Application accessible en production
- ✓ **Documentation complète** : 3768 lignes de documentation technique

#### 8.1.2 Compétences Acquises

Ce projet a permis de développer et consolider de nombreuses compétences :

##### Compétences Techniques :

- Maîtrise de React.js et de l'écosystème frontend moderne
- Développement d'API REST sécurisées avec Node.js/Express
- Conception et modélisation de bases de données NoSQL
- Implémentation des standards de sécurité OWASP
- Déploiement et configuration de services cloud
- Utilisation de Git et GitHub pour le versioning

##### Compétences Transversales :

- Analyse et conception d'applications complexes
- Gestion de projet et planification
- Résolution de problèmes techniques
- Rédaction de documentation technique
- Veille technologique

## 8.2 Défis Rencontrés

### 8.2.1 Défis Techniques

Plusieurs défis techniques ont été rencontrés et surmontés :

1. **Compatibilité Mongoose 9.x** : Adaptation des hooks pre-save (suppression du callback `next()`)
2. **TailwindCSS v4** : Incompatibilité initiale, résolution par downgrade vers v3.4.1
3. **Fast Refresh React** : Warnings ESLint, solution par séparation des hooks
4. **CORS en production** : Configuration stricte des origines autorisées
5. **Render Free Tier** : Gestion du sleep après 15 minutes d'inactivité

### 8.2.2 Solutions Apportées

Chaque défi a été résolu de manière méthodique :

- Lecture approfondie de la documentation officielle
- Recherche sur GitHub Issues et Stack Overflow
- Tests itératifs et débogage systématique
- Refactoring du code quand nécessaire

## 8.3 Perspectives et Améliorations Futures

Bien que l'application soit fonctionnelle et sécurisée, plusieurs améliorations peuvent être envisagées :

### 8.3.1 Fonctionnalités Utilisateur

- **Panier d'achat** : Ajout de produits, gestion des quantités, calcul du total
- **Système de commandes** : Historique, statuts (en cours, livrée), suivi
- **Paiement en ligne** : Intégration Stripe ou PayPal
- **Profil utilisateur** : Modification des informations, avatar
- **Favoris** : Sauvegarder des produits pour plus tard
- **Système de notation** : Notes et avis sur les produits
- **Notifications email** : Confirmation de commande, promotions

### 8.3.2 Fonctionnalités Administrateur

- **Gestion des utilisateurs** : Liste, modification, suppression, statistiques
- **Gestion des commandes** : Validation, expédition, annulation
- **Statistiques avancées** : Charts avec Chart.js, chiffre d'affaires, produits populaires
- **Export de données** : CSV, PDF pour rapports
- **Promotions** : Codes promo, réductions, ventes flash

### 8.3.3 Améliorations Techniques

- **Tests automatisés** : Jest, Supertest, React Testing Library
- **CI/CD** : GitHub Actions pour tests automatiques et déploiement
- **Documentation API** : Swagger/OpenAPI pour documentation interactive
- **Multi-langue** : i18next pour internationalisation (FR/EN/AR)
- **Mode sombre** : Toggle light/dark theme
- **PWA** : Progressive Web App avec offline support
- **WebSockets** : Notifications temps réel avec Socket.io
- **GraphQL** : Alternative à REST pour requêtes flexibles
- **Microservices** : Séparation en services indépendants pour scalabilité

### 8.3.4 Améliorations Sécurité

- **2FA** : Authentification à deux facteurs (SMS, Google Authenticator)
- **Captcha** : Protection contre bots (Google reCAPTCHA)
- **Rate Limiting** : Protection contre brute force et DDoS
- **Audit logs** : Traçabilité complète des actions administrateurs
- **Security headers** : Helmet.js avec CSP stricte
- **Encryption** : Chiffrement des données sensibles au repos
- **Penetration Testing** : Tests d'intrusion professionnels

## 8.4 Conclusion Finale

Ce projet de développement Full Stack a été une expérience enrichissante qui a permis de consolider les connaissances théoriques en sécurité des applications web modernes par une mise en pratique concrète.

L'application développée démontre une maîtrise des technologies web actuelles (React, Node.js, MongoDB) et l'implémentation effective des meilleures pratiques de sécurité (OWASP Top 10, JWT, bcrypt, RBAC).

Le déploiement réussi en production sur des services cloud professionnels (Render, Vercel, MongoDB Atlas) valide la qualité et la robustesse de l'architecture mise en place.

Ce projet constitue une base solide pour de futurs développements et peut servir de référence pour la création d'applications web sécurisées et performantes.

« *La sécurité n'est pas un produit, mais un processus.* »  
— Bruce Schneier

# Bibliographie

- [1] OWASP Foundation. (2021). *OWASP Top Ten 2021*.  
<https://owasp.org/www-project-top-ten/>
- [2] Mozilla Developer Network. (2026). *Web Security*.  
<https://developer.mozilla.org/en-US/docs/Web/Security>
- [3] Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)*. RFC 7519.  
<https://datatracker.ietf.org/doc/html/rfc7519>
- [4] Meta. (2026). *React Documentation*.  
<https://react.dev>
- [5] Express.js Team. (2026). *Express.js Guide*.  
<https://expressjs.com>
- [6] MongoDB Inc. (2026). *MongoDB Manual*.  
<https://docs.mongodb.com>
- [7] OpenJS Foundation. (2026). *Node.js Documentation*.  
<https://nodejs.org/en/docs/>
- [8] Provost, N., & Mazières, D. (1999). *A Future-Adaptable Password Scheme*.  
USENIX Annual Technical Conference.

# Annexe A

## Code Source Important

### A.1 Schéma Mongoose User

```
1 const mongoose = require('mongoose');
2 const bcrypt = require('bcryptjs');
3
4 const UserSchema = new mongoose.Schema({
5   name: {
6     type: String,
7     required: [true, 'Le nom est obligatoire'],
8     trim: true,
9     minlength: 2,
10    maxlength: 50
11  },
12  email: {
13    type: String,
14    required: [true, 'L\'email est obligatoire'],
15    unique: true,
16    lowercase: true,
17    match: [/^[\S+\@\S+\.\S+$/, 'Email invalide']
18  },
19  password: {
20    type: String,
21    required: [true, 'Le mot de passe est obligatoire'],
22    minlength: 6
23  },
24  role: {
25    type: String,
26    enum: ['user', 'admin'],
27    default: 'user'
28  }
29 }, { timestamps: true });
30
31 // Hash du mot de passe avant sauvegarde
32 UserSchema.pre('save', async function() {
33   if (!this.isModified('password')) return;
34   this.password = await bcrypt.hash(this.password, 10);
35 });
36
37 // Méthode de comparaison du mot de passe
38 UserSchema.methods.comparePassword = async function(candidatePassword) {
39   return await bcrypt.compare(candidatePassword, this.password);
40 };
41
42 // Exclure le mot de passe des réponses JSON
43 UserSchema.methods.toJSON = function() {
44   const obj = this.toObject();
45   delete obj.password;
46   return obj;
47 };
```

```

48 module.exports = mongoose.model('User', UserSchema);
49

```

Listing A.1 – Modèle User avec bcrypt

## A.2 Middleware d'Authentification JWT

```

1 const jwt = require('jsonwebtoken');
2 const User = require('../models/User');
3
4 exports.protect = async (req, res, next) => {
5     let token;
6
7     // Extraction du token depuis le header Authorization
8     if (req.headers.authorization?.startsWith('Bearer')) {
9         token = req.headers.authorization.split(' ')[1];
10    }
11
12    if (!token) {
13        return res.status(401).json({
14            success: false,
15            message: 'Accès non autorisé. Veuillez vous connecter.'
16        });
17    }
18
19    try {
20        // Vérification et décodage du token
21        const decoded = jwt.verify(token, process.env.JWT_SECRET);
22
23        // Récupération de l'utilisateur
24        req.user = await User.findById(decoded.id).select('-password');
25
26        if (!req.user) {
27            return res.status(401).json({
28                success: false,
29                message: 'Utilisateur introuvable'
30            });
31        }
32
33        next();
34    } catch (error) {
35        return res.status(401).json({
36            success: false,
37            message: 'Token invalide ou expiré'
38        });
39    }
40};
41
42 // Middleware d'autorisation par rôle
43 exports.authorize = (...roles) => {
44     return (req, res, next) => {
45         if (!roles.includes(req.user.role)) {
46             return res.status(403).json({
47                 success: false,
48                 message: `Le rôle "${req.user.role}" n'est pas autorisé`
49             });

```

```
50     }
51     next();
52 }
53 }
```

Listing A.2 – Middleware de protection des routes

## Annexe B

# Commandes de Déploiement

## B.1 Commandes Git

```
1 # Initialisation
2 git init
3 git add .
4 git commit -m "feat: Initial commit"
5
6 # Création repository GitHub
7 gh repo create websec --public
8
9 # Push
10 git remote add origin https://github.com/redaelhadfi/websec.git
11 git branch -M main
12 git push -u origin main
```

Listing B.1 – Workflow Git

## B.2 Déploiement Backend (Render)

```
1 # Via Render Dashboard:
2 # 1. New Web Service
3 # 2. Connect GitHub repository
4 # 3. Configuration:
5 #     - Build Command: npm install
6 #     - Start Command: npm start
7 #     - Environment: Node
8 #
9 # 4. Environment Variables:
10 NODE_ENV=production
11 MONGODB_URI=mongodb+srv://...@cluster0.mongodb.net/ecommerce
12 JWT_SECRET=your_secret_here
13 CLIENT_URL=https://websec-bice.vercel.app
```

Listing B.2 – Configuration Render

## B.3 Déploiement Frontend (Vercel)

```
1 # Via Vercel Dashboard:
2 # 1. New Project
3 # 2. Import Git Repository
4 # 3. Configuration:
5 #     - Framework: Vite
6 #     - Build Command: npm run build
7 #     - Output Directory: dist
```

```
8 #      - Root Directory: frontend
9 #
10 # 4. Environment Variables:
11 VITE_API_URL=https://redaelhadfii.onrender.com
```

Listing B.3 – Configuration Vercel