École Marocaine des Sciences de l'Ingénieur

Cahier des Charges Technique

Dropshipping

Présenté par : guermellou & modakir
Programme : Computer Science

Version : 1.0

Année universitaire 2024–2025

Table des matières

1 Introduction et Contexte **2**
1.1 Présentation du Projet 2
1.2 Objectifs 2
1.3 Périmètre (Scope) 2

2 Architecture Technique **3**
2.1 Technologies Utilisées 3
2.2 Architecture Globale 3

3 Spécifications Fonctionnelles Détaillées **4**
3.1 Module Authentification et Clients 4
3.2 Module Catalogue et Produits 4
3.3 Module Panier d'Achat 4
3.4 Module Commande et Paiement (Checkout) 5
3.5 Module Administration (Back-office) 5

4 Exigences Non-Fonctionnelles **6**
4.1 Sécurité 6
4.2 Performance 6
4.3 Évolutivité 6

Chapitre 1

Introduction et Contexte

1.1 Présentation du Projet

DropShop est une solution complète de commerce électronique développée spécifiquement pour le modèle de *dropshipping*. Contrairement au e-commerce traditionnel, cette plateforme est conçue pour gérer des produits qui ne sont pas stockés physiquement par le vendeur, mais expédiés directement depuis les fournisseurs vers les clients finaux.

1.2 Objectifs

L'objectif principal est de fournir une application web robuste, sécurisée et évolutive permettant :

- Aux **Clients** de naviguer, commander et suivre leurs achats de manière intuitive.
- Aux **Administrateurs** de gérer le catalogue produit, les fournisseurs et le cycle de vie des commandes via un tableau de bord centralisé.

1.3 Périmètre (Scope)

Le projet couvre l'ensemble du flux transactionnel : de l'authentification des utilisateurs à la validation du paiement (simulation) et la gestion des statuts de commande, en passant par la gestion complexe du panier d'achat persistant.

Chapitre 2

Architecture Technique

2.1 Technologies Utilisées

Le projet repose sur la stack technologique Microsoft .NET moderne, garantissant performance et maintenabilité.

Technologie	**Usage dans le projet**	**Version**
ASP.NET Core MVC	Framework principal (Architecture Model-View-Controller)	.NET 8
Entity Framework Core	ORM pour l'interaction avec la base de données	8.0
SQL Server	Système de gestion de base de données relationnelle	LocalDB/Prod
Bootstrap 5	Framework CSS pour le responsive design	5.3
BCrypt.Net	Sécurité et hachage des mots de passe	-

TABLE 2.1 – Stack Technologique

2.2 Architecture Globale

L'application suit strictement le modèle architectural MVC (Modèle-Vue-Contrôleur) avec une séparation claire des responsabilités :

- **Controllers :** Gèrent les requêtes HTTP, la logique métier et l'orchestration (ex : `AdminController`, `CommandeController`).
- **Models (Domain) :** Représentent les entités persistantes de la base de données.
- **ViewModels (DTO) :** Objets optimisés pour le transfert de données entre le contrôleur et la vue, évitant d'exposer directement les entités de la base de données.
- **Views :** Interface utilisateur générée via le moteur de template Razor (.cshtml).

Chapitre 3

Spécifications Fonctionnelles Détaillées

Cette section détaille les fonctionnalités analysées directement depuis le code source.

3.1 Module Authentification et Clients

Source : ClientController.cs, Program.cs

— **Inscription :** Validation des champs (Nom, Prénom, Email). Vérification de l'unicité de l'email en base de données. Hachage du mot de passe via **BCrypt** avant stockage.
— **Connexion :** Authentification par Cookie sécurisé (`HttpOnly`, expiration 7 jours). Gestion des sessions via `HttpContext.Session`.
— **Gestion de Profil :** Modification des informations personnelles.
— **Carnet d'adresses :** CRUD (Create, Read, Update, Delete) complet des adresses de livraison. Gestion d'un indicateur "Adresse Principale" qui désactive automatiquement l'ancien flag lors d'un changement.

3.2 Module Catalogue et Produits

Source : ProduitController.cs

— **Affichage :** Liste paginée (12 articles par page par défaut).
— **Filtrage Avancé :**
  — Recherche textuelle (Titre et Description).
  — Plage de prix (Min / Max).
  — Par Fournisseur.
— **Tri :** Par prix (croissant/décroissant), alphabétique, ou nouveauté.
— **Recherche AJAX :** Endpoint API JSON pour l'autocomplétion dans la barre de recherche.

3.3 Module Panier d'Achat

Source : PanierController.cs, CommandeController.cs

— **Persistance :** Contrairement à un panier simple en session, le panier est stocké en base de données (`Panier` et `LignePanier`), permettant à l'utilisateur de retrouver ses articles après reconnexion.

— **Gestion :** Ajout d'articles, modification des quantités, suppression.
— **Indicateur visuel :** Un `ViewComponent` met à jour dynamiquement le badge du panier dans la navigation.

3.4 Module Commande et Paiement (Checkout)

Source : CommandeController.cs
— **Vérification des stocks :** Avant la validation, le système revérifie le stock disponible. Si le stock est insuffisant, l'utilisateur est redirigé vers le panier.
— **Création de commande :**
1. Sélection ou création d'une adresse de livraison.
2. Calcul du montant total.
3. **Règle Métier :** Frais de port de 4,99 € si la commande est inférieure à 50 €. Gratuits au-delà.
4. Décrémentation atomique des stocks produits.
5. Changement du statut à "En Attente".

— **Historique :** Le client peut consulter ses commandes passées et leur statut.

3.5 Module Administration (Back-office)

Source : AdminController.cs Ce module est protégé par l'attribut `[Authorize(Roles = "Admin")]`.
— **Dashboard Analytique :**
  — Calcul du chiffre d'affaires (excluant les commandes annulées).
  — Compteurs : Commandes en attente, produits en stock faible (< 10 unités).
  — Top 5 des produits les plus vendus.
— **Gestion des Produits :** Ajout/Modif avec upload d'image (stockée dans `wwwroot/uploads`). Suppression logique (soft delete) ou désactivation si le produit a déjà été commandé pour préserver l'intégrité référentielle.
— **Gestion des Fournisseurs :** CRUD complet. Empêche la suppression d'un fournisseur s'il possède des produits liés.
— **Gestion des Commandes :** Vue détaillée, changement de statut (En Attente → En Cours → Expédiée → Livrée).

Chapitre 4

Exigences Non-Fonctionnelles

4.1 Sécurité

— **CSRF (Cross-Site Request Forgery) :** Protection active sur tous les formulaires POST via `ValidateAntiForgeryToken`.
— **XSS :** Encodage automatique des sorties via le moteur Razor.
— **Contrôle d'accès :** Utilisation stricte de l'annotation `[Authorize]` pour protéger les contrôleurs sensibles (Panier, Commande, Admin).

4.2 Performance

— Utilisation systématique de la programmation asynchrone (`async/await`) pour toutes les opérations I/O (Base de données, Fichiers) afin de ne pas bloquer le thread principal.
— Chargement optimisé des données liées (Eager Loading via `.Include()`) pour éviter le problème du N+1.

4.3 Évolutivité

— Architecture prête pour la migration de SQL Server LocalDB vers Azure SQL ou SQL Server Enterprise via simple changement de `ConnectionString`.
— Injection de dépendances configurée dans `Program.cs` facilitant les tests unitaires futurs.