Rapport de Projet

Application Online Shopping

Application e-commerce JavaFX

Gestion des utilisateurs et des produits

Réalisé par :

Meftah Ibtihale

Filière : 4IIR G2

Encadré par :

LARHLIMI Abderrahim

Année universitaire 2025–2026

Table des matières

Remerciements **3**

Résumé **4**

1 Introduction générale **5**

Introduction générale **5**

1.1 Objectif 5

1.2 Fonctionnalités 6

2 Technologies Utilisées **7**

3 Architecture et Choix Techniques **8**

3.1 Architecture MVC 8

3.2 Gestion Hybride des Données 9

3.3 Multi-threading 9

4 Modélisation **11**

4.1 Diagramme de classes 11

5 Authentification **13**

5.1 Interface de Connexion 13

6 Espace Administrateur **14**

6.1 Dashboard Administrateur 14

7 Gestion des Utilisateurs **15**

7.1 Interface Gestion des Utilisateurs 15

8 Gestion des Produits **16**

8.1 Interface Gestion des produits 16

9 Conclusion **18**

Table des figures

4.1 Diagramme de classes 12

5.1 Interface de connexion 13

6.1 Dashboard administrateur 14

7.1 Gestion des utilisateurs 15

Remerciements

Je tiens à exprimer mes sincères remerciements à mon encadrant pour son accompagnement, ses conseils et son suivi tout au long de la réalisation de ce projet. Je remercie également l'établissement pour le cadre pédagogique et les moyens mis à disposition. Enfin, je remercie toute personne ayant contribué, de près ou de loin, à la réussite de ce travail.

Résumé

Ce projet consiste à concevoir et développer une application de gestion de produits pour une plateforme de vente en ligne. L'objectif principal est de permettre l'administration, la consultation et la gestion des produits et des utilisateurs de manière efficace. Les technologies utilisées incluent Java, JavaFX, Hibernate et MySQL, avec une base de données déployée via Docker.

Chapitre 1

Introduction générale

Le développement des applications de commerce en ligne occupe une place importante dans le domaine informatique. Ce projet vise à répondre au besoin de gestion centralisée des produits et des utilisateurs dans un système de vente en ligne. Le présent rapport est structuré en plusieurs chapitres décrivant l'analyse, la conception, la réalisation et la validation du système.

1.1 Objectif

L'objectif principal du projet est de consolider les compétences en programmation orientée objet à travers une architecture claire et maintenable. Le projet permet de mettre en œuvre les principes fondamentaux tels que l'encapsulation, l'héritage, le polymorphisme et l'abstraction, tout en respectant les bonnes pratiques de conception logicielle.

L'application vise également à maîtriser la gestion des données persistantes en utilisant Hibernate comme couche d'accès aux données. Cela inclut la configuration du framework ORM, le mapping objet-relationnel, la gestion des transactions et la communication avec une base de données MySQL déployée dans un environnement Docker.

Un autre objectif essentiel est la compréhension et l'application d'une architecture logicielle structurée, notamment le modèle MVC. Cette approche permet de séparer clairement l'interface utilisateur, la logique métier et l'accès aux données, améliorant ainsi la lisibilité du code, sa maintenabilité et son évolutivité.

Le projet a aussi pour but de développer une interface graphique ergonomique avec JavaFX, facilitant l'interaction entre l'utilisateur et le système. L'accent est mis sur la gestion des événements, la validation des données et l'affichage dynamique des informations issues de la base de données.

Enfin, cette application constitue une première approche des problématiques rencontrées dans les systèmes e-commerce réels, telles que la gestion des utilisateurs, des produits, de l'authentification et des erreurs. Elle prépare ainsi à des projets plus complexes intégrant la sécurité avancée, les performances et l'évolutivite du système.

1.2 Fonctionnalités

L'application intègre un système d'authentification des utilisateurs permettant un accès sécurisé à la plateforme. Chaque utilisateur doit s'inscrire puis se connecter à l'aide de ses identifiants personnels. Ce mécanisme garantit que seules les personnes autorisées peuvent accéder aux fonctionnalités de l'application et interagir avec les données stockées dans la base. La validation des informations saisies et la gestion des erreurs d'authentification assurent une expérience utilisateur fiable et cohérente.

La gestion des rôles constitue une fonctionnalité centrale du système. L'application distingue clairement les profils administrateur et client, chacun disposant de droits et d'interfaces spécifiques. L'administrateur peut accéder aux fonctionnalités de gestion avancée, telles que l'ajout, la modification et la suppression des produits ainsi que la supervision des utilisateurs. Le client, quant à lui, dispose d'un accès limité aux fonctionnalités de consultation des produits et d'interaction avec le catalogue. Cette séparation des rôles renforce la sécurité et respecte les principes de contrôle d'accès.

La gestion des produits permet de créer, consulter, mettre à jour et supprimer les articles disponibles sur la plateforme. Chaque produit est défini par des informations essentielles telles que le nom, le prix, la quantité en stock, la description et l'image associée. Le système assure le suivi du stock afin de refléter en temps réel la disponibilité des produits. Cette fonctionnalité garantit une gestion cohérente des données et constitue le cœur du fonctionnement de l'application e-commerce.

Chapitre 2

Technologies Utilisées

— L'application est développée en **Java**, qui constitue le socle de la logique métier. Ce langage permet de structurer le projet selon les principes de la programmation orientée objet, d'assurer une bonne maintenabilité du code et de mettre en œuvre des concepts avancés tels que les DAO, la gestion des exceptions et la séparation des responsabilités entre les différentes couches de l'application.

— L'interface graphique est réalisée avec **JavaFX**. Cette technologie permet de concevoir des interfaces modernes, interactives et ergonomiques. Grâce aux fichiers FXML et aux contrôleurs associés, l'interface est clairement séparée de la logique métier, ce qui facilite l'évolution du design et l'amélioration de l'expérience utilisateur.

— La base de données repose sur **MySQL**, un système de gestion de bases de données relationnelles fiable et largement utilisé. MySQL assure le stockage persistant des données telles que les utilisateurs, les produits et les informations liées au stock. Il permet également l'exécution de requêtes efficaces pour la consultation et la mise à jour des données.

— **Hibernate** est utilisé pour la gestion des utilisateurs. Cet ORM simplifie l'interaction avec la base de données en automatisant la persistance des objets Java vers les tables relationnelles. Il réduit l'écriture de code SQL répétitif, améliore la lisibilité du code et facilite la gestion des entités, des relations et des transactions.

— Pour la gestion des produits, l'application utilise **JDBC et du SQL natif**. Ce choix permet un contrôle précis des requêtes, notamment pour les opérations de gestion du stock et des produits. L'utilisation directe de JDBC offre de bonnes performances et une maîtrise complète des accès aux données.

— **Docker** est utilisé pour la base de données MySQL afin de garantir un environnement de développement stable et reproductible. La base de données est exécutée dans un conteneur, ce qui facilite le déploiement, évite les problèmes de configuration locale et assure la portabilité du projet sur différentes machines.

Chapitre 3

Architecture et Choix Techniques

3.1 Architecture MVC

L'architecture de l'application repose sur le modèle MVC (Model–View–Controller), afin d'assurer une séparation claire des responsabilités et une meilleure maintenabilité du code. Ce choix architectural permet de structurer l'application de manière logique et évolutive, tout en facilitant la compréhension du projet.

Le **Modèle** représente la couche responsable de la gestion des données et de la logique métier. Il regroupe les entités telles que les utilisateurs, les produits, les commandes et les éléments de stock. Cette couche communique directement avec la base de données via Hibernate ou JDBC selon le type de données manipulées. Les collections Java, comme `List` et `Map`, sont utilisées pour stocker temporairement les données récupérées depuis la base, permettant un traitement efficace en mémoire.

La **Vue** correspond à l'interface graphique développée avec JavaFX. Elle est chargée de l'affichage des informations et de l'interaction avec l'utilisateur. Les vues ne contiennent aucune logique métier, ce qui garantit une interface claire, réactive et indépendante des traitements internes. Les données affichées proviennent du contrôleur sous forme de collections prêtes à être parcourues et affichées dans des composants graphiques comme les tableaux et les listes.

Le **Contrôleur** joue un rôle central dans l'application. Il reçoit les actions de l'utilisateur, déclenche les traitements nécessaires dans le modèle et met à jour la vue en conséquence. Les contrôleurs utilisent fréquemment les **Streams Java** pour filtrer, trier ou transformer les collections de données, par exemple pour afficher uniquement les produits disponibles en stock ou rechercher un produit spécifique par son nom. Cette approche rend le code plus lisible, plus concis et plus performant.

3.2 Gestion Hybride des Données

L'application adopte une gestion hybride des données afin de tirer parti des avantages de différentes technologies d'accès à la base de données. Ce choix a été fait pour répondre aux besoins spécifiques de chaque type de données manipulées.

Hibernate est utilisé pour la gestion des utilisateurs. Il simplifie les opérations CRUD grâce à la persistance objet-relationnelle et réduit considérablement la quantité de code SQL à écrire. Les entités utilisateurs sont automatiquement mappées aux tables de la base MySQL, ce qui facilite la gestion des rôles, de l'authentification et des sessions. Les collections Hibernate permettent de charger et manipuler les données de manière structurée avant leur exploitation dans l'application.

Le SQL natif, via JDBC, est utilisé pour la gestion des produits. Ce choix offre un contrôle total sur les requêtes, notamment pour les opérations liées au stock et aux prix. Il permet d'optimiser les performances et d'exécuter des requêtes précises adaptées aux besoins métier. Les résultats des requêtes SQL sont ensuite convertis en objets Java et stockés dans des collections, qui peuvent être traitées à l'aide des Streams pour effectuer des calculs, des filtrages ou des regroupements.

Cette approche hybride améliore à la fois la flexibilité, la performance et la maîtrise des accès aux données, tout en conservant une architecture cohérente et professionnelle.

3.3 Multi-threading

Le multi-threading est intégré dans l'application afin d'améliorer l'expérience utilisateur et d'assurer une interface fluide. Dans une application graphique, certaines opérations peuvent être longues, comme l'accès à la base de données ou la simulation d'un paiement. Sans gestion des threads, ces opérations risqueraient de bloquer l'interface.

Pour éviter ce problème, les traitements lourds sont exécutés dans des threads séparés du thread principal de JavaFX. Cela permet à l'interface de rester réactive, même lorsque des opérations complexes sont en cours. Les résultats de ces traitements sont ensuite transmis à la vue de manière sécurisée.

Le multi-threading est également utilisé pour simuler des scénarios réalistes, comme le traitement d'un paiement ou la validation d'une commande. Les données manipulées

par les threads sont souvent stockées dans des collections thread-safe ou traitées de manière contrôlée afin d'éviter les conflits. Les Streams peuvent être utilisés pour analyser les résultats de ces traitements, par exemple pour mettre à jour l'état des commandes ou recalculer le stock disponible.

Cette approche renforce la robustesse de l'application et démontre l'utilisation de concepts Java avancés dans un contexte réel et professionnel.

Chapitre 4

Modélisation

4.1 Diagramme de classes

Ce diagramme de classes modélise l'architecture objet de l'application Online Shopping en mettant en évidence les entités métier, les couches d'accès aux données, les utilitaires et les contrôleurs. Il reflète une conception structurée respectant les principes de la programmation orientée objet et du modèle MVC.

La couche Model regroupe les classes représentant le cœur métier de l'application. La classe User gère les informations liées aux utilisateurs et à leurs rôles. La classe Product constitue la base commune des produits, avec des spécialisations comme FurnitureProduct et RealEstateProduct, ce qui illustre un usage cohérent de la spécialisation métier. Les classes Order et OrderItem permettent de représenter le processus de commande, tandis que CartItem gère les éléments temporaires du panier.

Les relations entre les classes montrent clairement la logique fonctionnelle du système. Un utilisateur peut passer plusieurs commandes, chaque commande contient plusieurs éléments, et chaque élément du panier est associé à un produit. Ces associations traduisent fidèlement le fonctionnement d'une application e-commerce.

La couche DAO assure la persistance des données. Les classes UserDAO, ProductDAO, OrderDAO et FurnitureDAO encapsulent les opérations CRUD et isolent l'accès à la base de données du reste de l'application. L'utilisation de HibernateUtil centralise la gestion des sessions Hibernate, ce qui renforce la maintenabilité et la cohérence de l'accès aux données.

Les classes utilitaires comme CartManager et UserSession jouent un rôle clé dans la gestion de l'état de l'application. CartManager gère la collection des articles du panier et

calcule le total, tandis que UserSession permet de conserver les informations de l'utilisateur connecté durant la session.

Enfin, la couche Controller regroupe les classes responsables de la logique applicative et de l'interaction avec l'interface JavaFX. Les contrôleurs tels que LoginController, RegisterController, ProductsController, AdminProductsController et CartController orchestrent les actions de l'utilisateur, appellent les DAO appropriés et mettent à jour la vue.

Dans l'ensemble, ce diagramme met en évidence une architecture cohérente, modulaire et évolutive, facilitant la maintenance du projet et son extension vers des fonctionnalités e-commerce plus avancées.

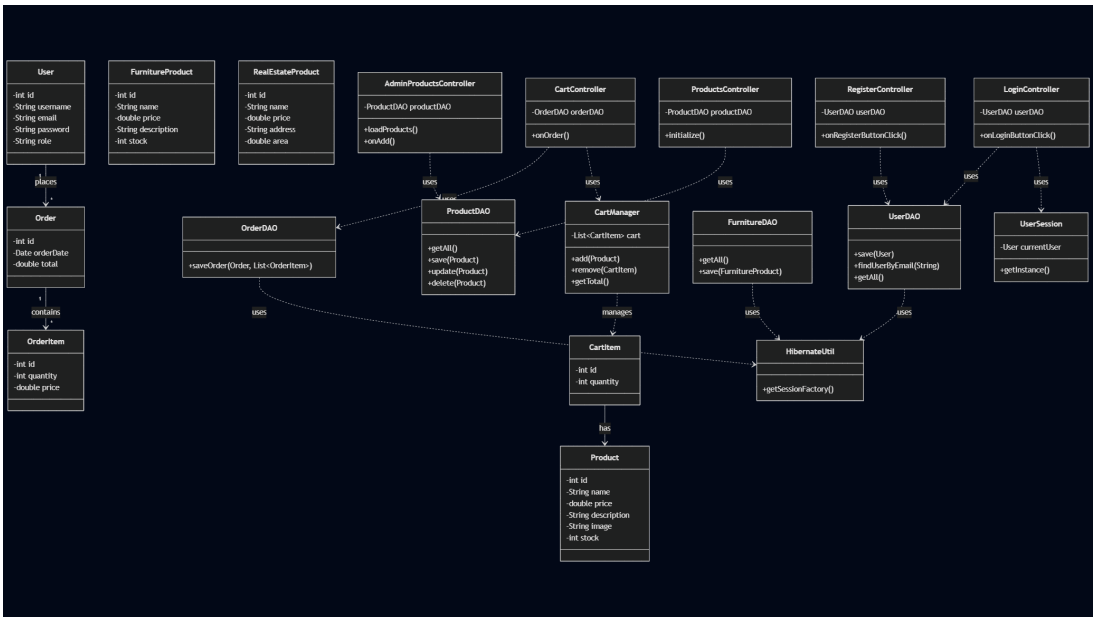


FIGURE 4.1 – Diagramme de classes

Chapitre 5

Authentification

5.1 Interface de Connexion

Cette interface permet à l'utilisateur de saisir son email et son mot de passe. L'accès dépend du rôle associé au compte.

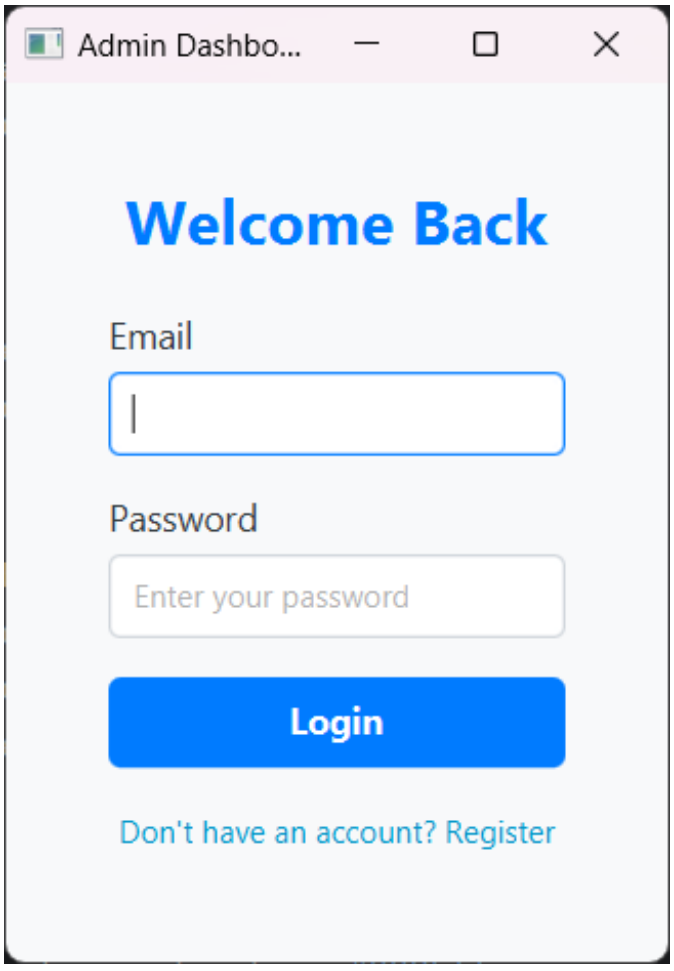


FIGURE 5.1 – Interface de connexion

Chapitre 6

Espace Administrateur

6.1 Dashboard Administrateur

Après connexion, l'administrateur accède à un tableau de bord central. Il permet la navigation entre les différentes fonctionnalités.

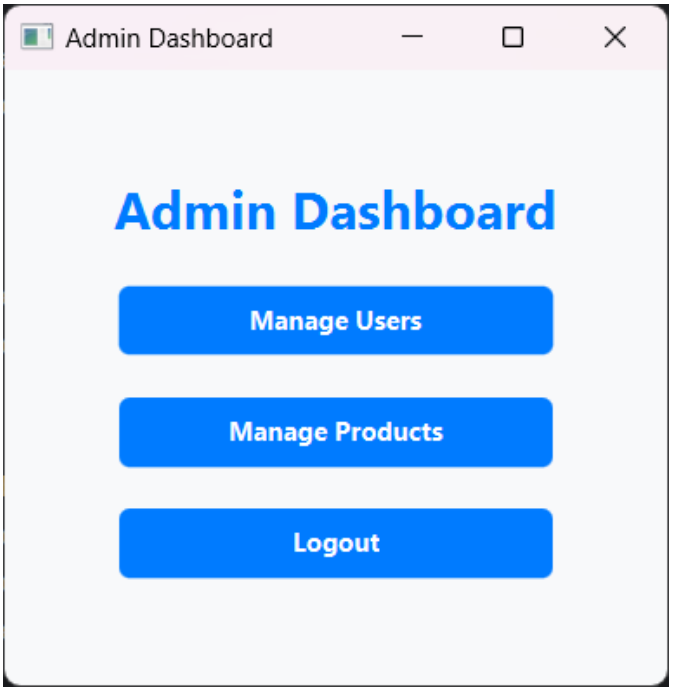


FIGURE 6.1 – Dashboard administrateur

Chapitre 7

Gestion des Utilisateurs

7.1 Interface Gestion des Utilisateurs

Cette partie permet à l'administrateur de gérer les comptes. Il peut ajouter, modifier ou supprimer des utilisateurs.

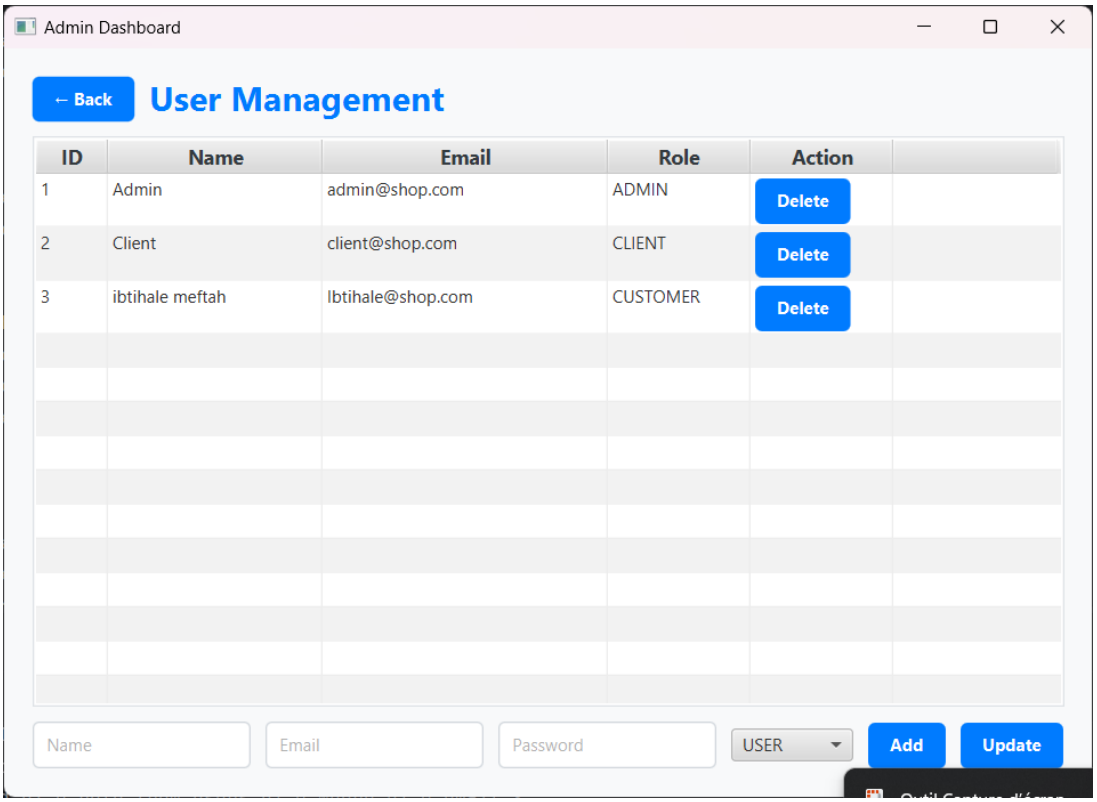


FIGURE 7.1 – Gestion des utilisateurs

Chapitre 8

Gestion des Produits

8.1 Interface Gestion des produits

L'administrateur dispose d'un espace dédié lui permettant de gérer l'ensemble des produits de la plateforme. Cette gestion repose sur le principe du CRUD, qui constitue le cœur du fonctionnement du module d'administration.

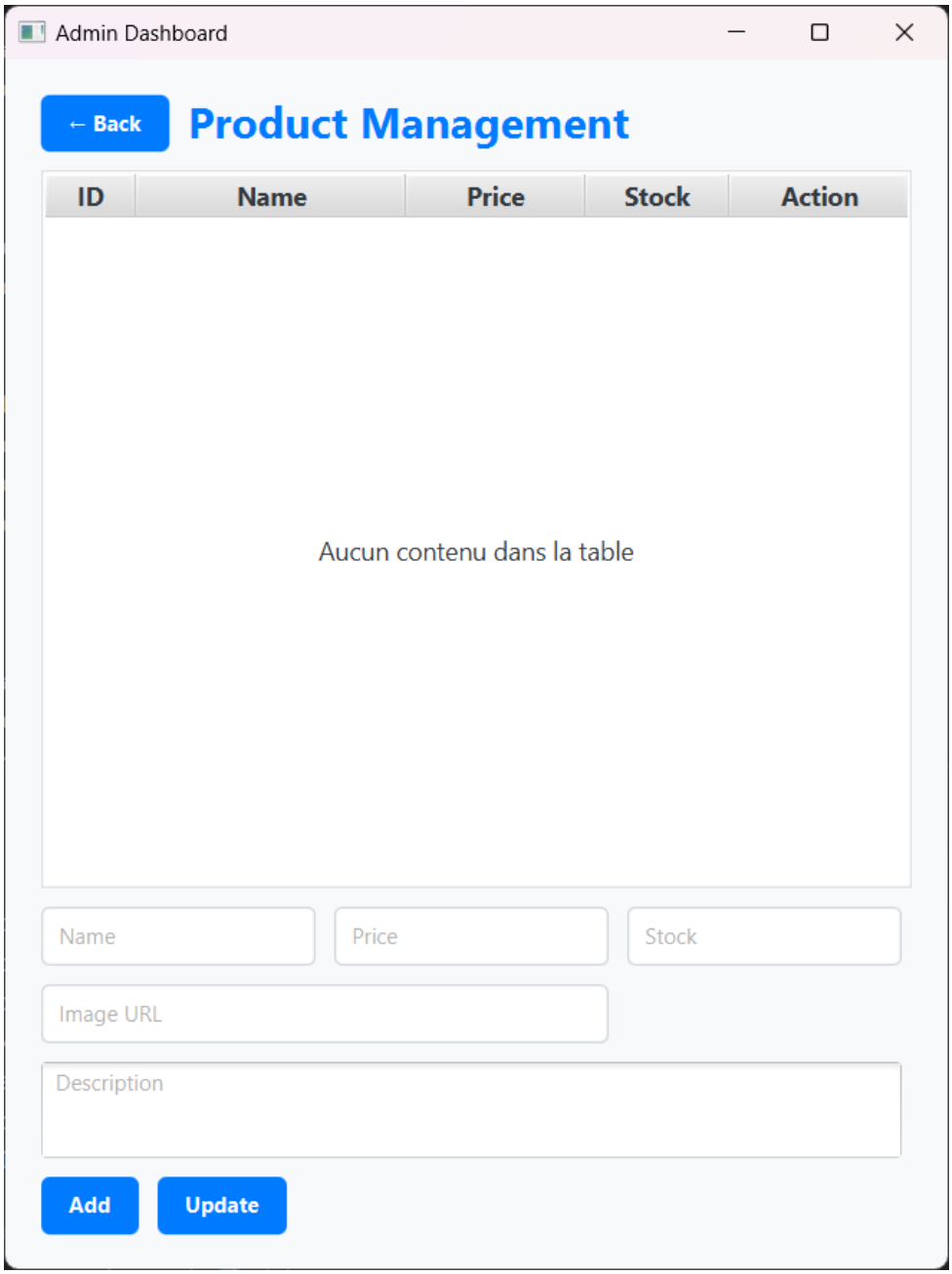
La création des produits permet à l'administrateur d'ajouter de nouveaux articles en définissant leurs informations essentielles, notamment le nom du produit, son prix et la quantité disponible en stock. Ces données sont saisies via une interface graphique claire et contrôlée, afin d'éviter les incohérences et les erreurs de saisie.

La consultation des produits offre une vue globale et structurée de l'ensemble du catalogue. L'administrateur peut visualiser à tout moment la liste des produits disponibles, leurs prix actuels et l'état du stock. Cette fonctionnalité facilite le suivi des articles et la prise de décision liée à la gestion des ventes.

La mise à jour des produits permet de modifier les informations existantes, comme l'ajustement des prix ou la mise à jour du stock après un approvisionnement ou une vente. Cette étape est essentielle pour garantir la cohérence entre les données affichées dans l'application et la réalité du stock.

La suppression des produits donne à l'administrateur la possibilité de retirer des articles devenus obsolètes ou indisponibles. Cette opération est réalisée de manière contrôlée afin de préserver l'intégrité de la base de données.

L'ensemble de ces opérations est directement relié à la base de données MySQL, qui assure le stockage persistant et sécurisé des informations. Chaque action CRUD déclenche des requêtes vers la base de données, garantissant ainsi que les données restent à jour, fiables et cohérentes tout au long du cycle de vie de l'application.


Admin Dashboard
← Back
Product Management
ID
Name
Price
Stock
Action
Aucun contenu dans la table
Name
Price
Stock
Image URL
Description
Add
Update

Chapitre 9

Conclusion

Ce projet met en pratique de manière concrète les concepts avancés du langage Java étudiés au cours de la formation. Il intègre la programmation orientée objet, la gestion des exceptions, l'utilisation des collections, des streams, ainsi que le multi-threading, afin de répondre à des problématiques réelles rencontrées dans le développement d'applications professionnelles. Cette mise en application permet de consolider les acquis théoriques à travers un projet structuré et fonctionnel.

L'application repose sur une architecture claire et bien organisée, basée sur le modèle MVC, garantissant une séparation nette entre la logique métier, l'interface graphique et le contrôle des actions utilisateur. Cette organisation facilite la maintenance, la compréhension du code et l'ajout de nouvelles fonctionnalités. L'interface graphique, développée avec JavaFX, offre une navigation fluide et intuitive, permettant aux utilisateurs et à l'administrateur d'interagir efficacement avec le système.

Enfin, ce projet a été conçu dans une optique d'évolutivité. Sa structure modulaire permet d'envisager facilement des extensions futures, telles que l'ajout d'un système de paiement réel, la gestion des commandes avancées, la mise en place de statistiques ou encore une version web ou mobile. Il constitue ainsi une base solide pour le développement d'une application e-commerce plus complète et plus robuste.