République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Supérieure des Sciences et Technologies

Projet semestriel dans le module du génie logiciel

Thème

La gestion du stock

Projet: GEStock

Présenté par :

HADID Rami

MIMOUNI Meriem

Encadré par : Mme Boulkrinat Samia

Résumé

Ce projet s'inscrit dans le cadre de la formation en Génie Logiciel et a pour objectif la conception et la réalisation d'une application informatique dédiée à la gestion de stock. L'entreprise étudiée souhaite automatiser le suivi de ses produits ainsi que les mouvements d'entrée et de sortie afin de réduire les erreurs liées aux méthodes manuelles et d'améliorer la traçabilité des opérations.

L'application proposée permet la gestion complète des produits (ajout, modification, suppression, recherche et consultation) ainsi que l'enregistrement des différents mouvements de stock (réception, retour, revente, utilisation, perte). Chaque opération entraîne une mise à jour automatique des quantités disponibles. De plus, le système offre des fonctionnalités de consultation statistique, telles que l'historique des mouvements et la liste des produits avec leurs quantités actuelles.

La démarche adoptée repose sur une modélisation UML (cas d'utilisation, séquence, classes) et sur l'élaboration d'un modèle logique de données. Le développement est réalisé en C# (WinForms .NET 9.0), avec une base de données relationnelle pour assurer la fiabilité et la cohérence des informations.

Ce travail vise non seulement à répondre aux besoins immédiats de l'entreprise, mais également à fournir une solution évolutive et adaptable, capable d'intégrer de nouvelles fonctionnalités à l'avenir.

Summary

This project is part of the Software Engineering curriculum and aims to design and implement a computer application dedicated to stock management. The studied company intends to automate the monitoring of its products as well as the incoming and outgoing movements in order to reduce errors caused by manual methods and improve the traceability of operations.

The proposed application ensures complete product management (adding, editing, deleting, searching, and consulting) as well as the recording of different stock movements (reception, return, resale, usage, loss). Each operation automatically updates the available quantities. Moreover, the system provides statistical features such as the history of movements and the list of products with their current quantities.

The adopted approach relies on UML modeling (use case, sequence, and class diagrams) and on the design of a logical data model. The development is carried out in C#, with a relational database to guarantee reliability and consistency of information.

This work not only addresses the company's immediate needs but also provides a scalable and adaptable solution, capable of integrating new functionalities in the future.

Sommaire

Chapitre 1 : Introduction générale

1.1 Qu'est-ce que la gestion de stock ?

La gestion de stock est l'ensemble des méthodes et outils permettant de suivre, contrôler et organiser les produits disponibles dans une entreprise. Elle consiste à assurer la disponibilité des articles nécessaires tout en évitant les ruptures ou les surstocks. Une bonne gestion de stock contribue à optimiser les coûts, améliorer la qualité du service et garantir la satisfaction des clients. n

1.2 Problématique

Dans de nombreuses entreprises, la gestion du stock est encore réalisée de manière manuelle, à l'aide de registres papier ou de fichiers Excel. Ces pratiques entraînent plusieurs difficultés :

- Risques élevés d'erreurs humaines.
- Manque de traçabilité des mouvements.
- Difficulté à obtenir des statistiques fiables.
- Perte de temps dans la recherche et la mise à jour des informations.

Face à ces limites, il devient nécessaire de mettre en place une solution informatique capable d'automatiser et de sécuriser la gestion des stocks.

1.3 Objectifs

Le projet vise à développer une application informatique permettant :

- La gestion complète des produits (ajout, modification, suppression ect ..).
- L'enregistrement des mouvements de stock (entrées et sorties).
- La mise à jour automatique des quantités disponibles.
- La génération de statistiques et d'historiques pour faciliter la prise de décision.

Au-delà de ces objectifs immédiats, l'application doit être évolutive afin d'intégrer de nouvelles fonctionnalités selon les besoins futurs de l'entreprise.

1.4 Organisation du rapport

Ce rapport est structuré en plusieurs chapitres :

- **Chapitre 1 : Introduction générale** – Présentation du contexte, de la problématique et des objectifs.
- **Chapitre 2 : Étude de l'existant** – Analyse des méthodes actuelles et identification des limites.
- **Chapitre 3 : Conception** – Expression des besoins, modélisation UML et conception du modèle de données.
- **Chapitre 4 : Réalisation** – Présentation des outils utilisés et de l'application développée.
- **Chapitre 5 : Conclusion** – Bilan du projet, difficultés rencontrées et perspectives d'amélioration.

Chapitre 2 : Étude de l'existant

2.1 Introduction

Avant de concevoir une nouvelle solution informatique, il est essentiel d'analyser les méthodes actuellement utilisées pour la gestion de stock. Cette étude permet de mettre en évidence les pratiques existantes, leurs avantages et leurs limites, afin de mieux comprendre les besoins réels de l'entreprise et de justifier la mise en place d'un système automatisé.

2.2 Méthodes de gestion actuelles

Dans de nombreuses entreprises, la gestion de stock repose encore sur des outils simples ou des méthodes traditionnelles :

- **Gestion manuelle** : utilisation de registres papier pour noter les entrées et sorties.
- **Tableurs (Excel)** : saisie des produits et des mouvements dans des fichiers, avec calculs manuels des quantités.
- **Logiciels basiques** : certaines entreprises utilisent des solutions génériques, souvent limitées et non adaptées à leurs besoins spécifiques.

2.3 Limites des solutions existantes

Ces méthodes présentent plusieurs inconvénients :

- **Manque de fiabilité** : risque élevé d'erreurs humaines lors de la saisie ou du calcul.
- **Absence de traçabilité** : difficulté à retrouver l'historique des mouvements.
- **Temps perdu** : recherche et mise à jour des informations longues et fastidieuses.
- **Manque de statistiques** : impossibilité de générer des rapports détaillés pour la prise de décision.

2.4 Besoins non couverts

L'analyse de l'existant montre que plusieurs besoins restent insatisfaits :

- Centralisation des données dans une base unique et sécurisée.
- Mise à jour automatique des quantités après chaque mouvement.
- Génération d'états statistiques et d'historiques filtrés.
- Interface conviviale et adaptée aux utilisateurs.

Chapitre 3 : Conception

3.1 Introduction

La phase de conception constitue une étape essentielle dans le développement d'un projet logiciel. Elle permet de transformer les besoins exprimés en modèles clairs et précis, facilitant ainsi la réalisation technique. Dans ce projet, nous avons adopté une démarche basée sur la modélisation UML afin de représenter les aspects fonctionnels et statiques du système de gestion de stock.

3.2 Expression des besoins

3.2.1 Besoins fonctionnels

Le système doit permettre :

- La gestion des produits (ajout, modification, suppression, recherche, consultation).
- La gestion des mouvements de stock (entrées : réception, retour ; sorties : revente, utilisation, perte).
- La mise à jour automatique des quantités disponibles après chaque mouvement.
- La consultation de l'historique des mouvements filtré par produit.
- L'affichage de statistiques (quantités disponibles, mouvements récents).

3.2.2 Besoins non fonctionnels

- **Sécurité** : protection des données contre les accès non autorisés.
- **Fiabilité** : cohérence des informations après chaque opération.
- **Performance** : rapidité d'exécution des traitements.
- **Ergonomie** : interface simple et intuitive.
- **Évolutivité** : possibilité d'ajouter de nouvelles fonctionnalités.

3.3 Démarche suivie

Nous avons choisi UML (Unified Modeling Language) car il s'agit d'un langage standard de modélisation largement utilisé en ingénierie logicielle. UML permet de représenter visuellement les besoins, les interactions et la structure du système, ce qui facilite la communication entre les concepteurs et les développeurs.

3.4 Identification des acteurs

Les principaux acteurs du système sont :

- **Administrateur** : gère les produits et supervise les mouvements.
- **Magasinier** : enregistre les entrées et sorties de stock.
- **Utilisateur simple** : consulte les produits et les statistiques.

3.5 Aspect fonctionnel

Pour garantir une compréhension approfondie du fonctionnement de notre application, nous entamons le processus par la création du diagramme de cas d'utilisation (Use Case). Celui-ci représente l'interaction entre le système informatique et les différents acteurs impliqués, fournissant ainsi une vue claire des fonctionnalités disponibles pour chaque utilisateur.
Par la suite, nous passerons au diagramme de séquence afin de mieux comprendre l'interaction avec le système.

3.5.1 Diagramme de cas d'utilisation

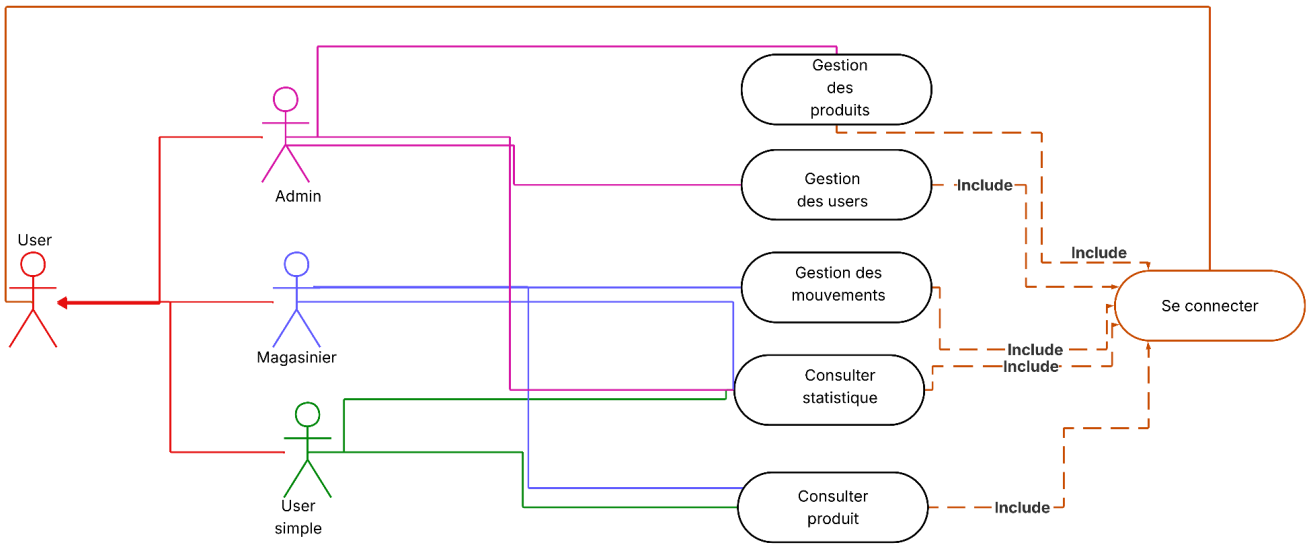
Représente les interactions entre les acteurs et le système (ex. : ajouter produit, enregistrer mouvement, consulter historique).

3.5.1.1 Diagramme de cas d'utilisation global :

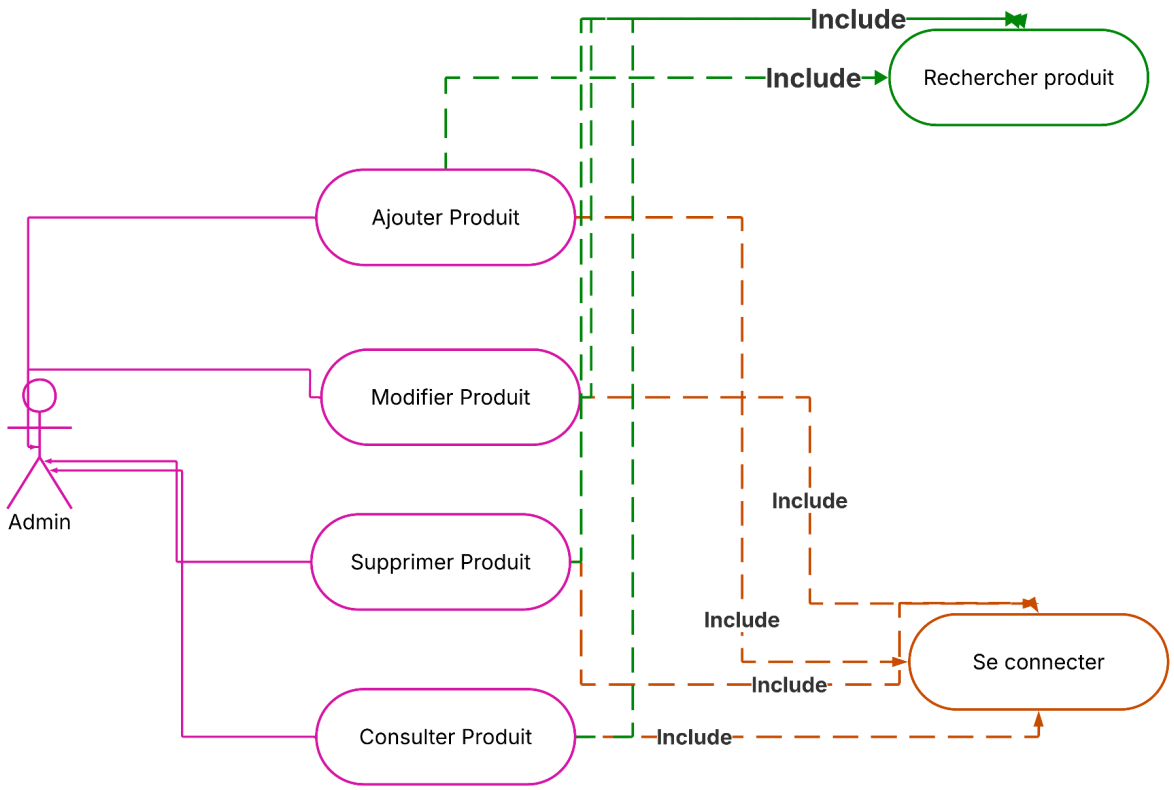
Le diagramme de cas d'utilisation global représente l'ensemble des fonctionnalités offertes par le système de gestion de stock ainsi que les interactions entre le système et les différents acteurs.

Il permet d'avoir une vue d'ensemble du fonctionnement de l'application et de définir clairement les responsabilités de chaque acteur.

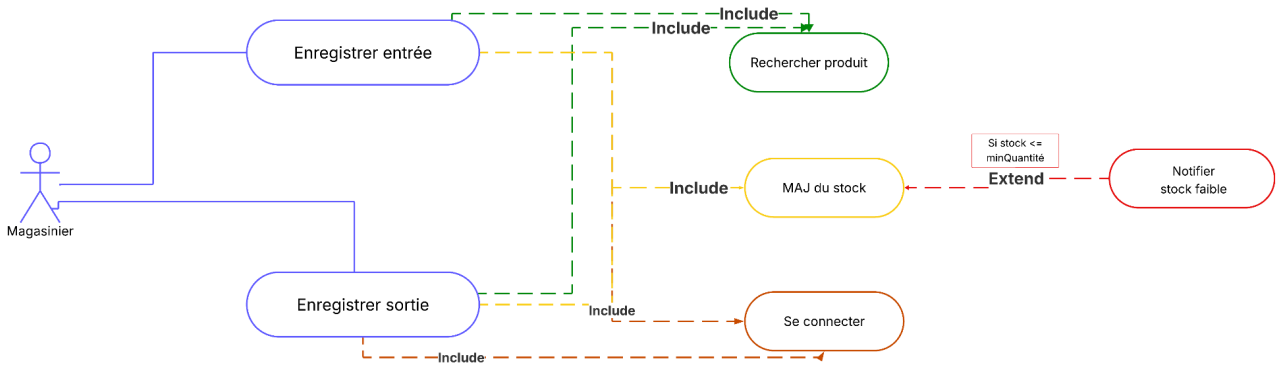
Trois acteurs principaux ont été identifiés : l'Administrateur, le Magasinier et l'Utilisateur simple. Chacun dispose de droits spécifiques lui permettant d'accéder à certaines fonctionnalités du système.



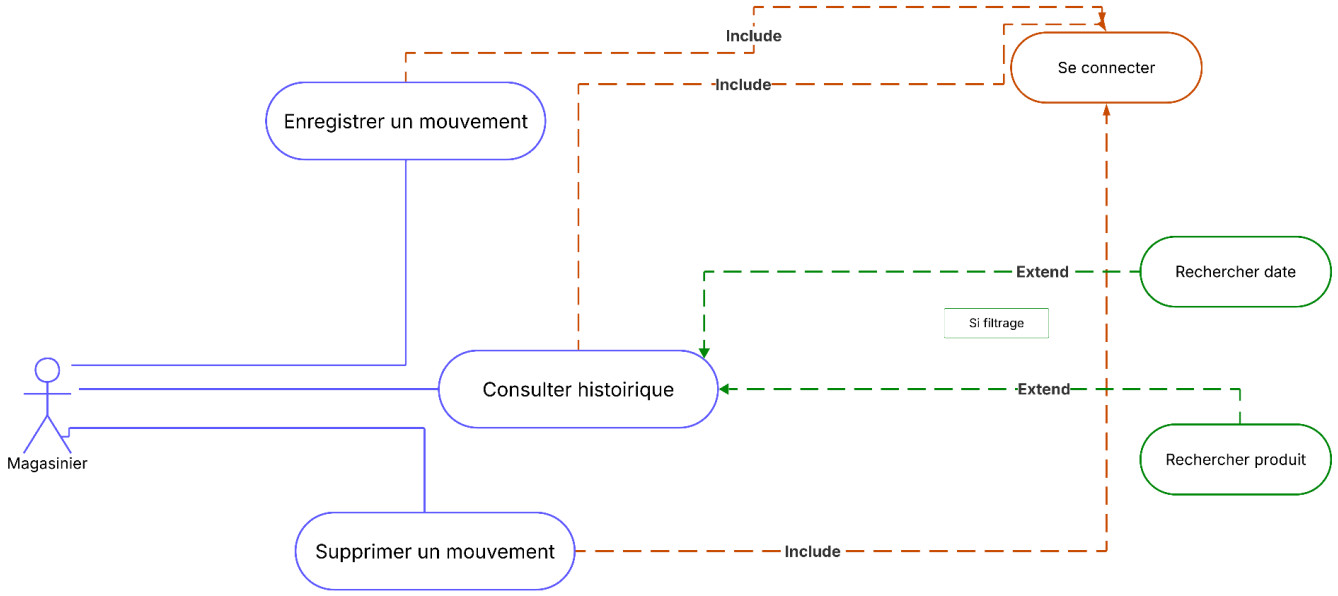
3.5.1.2 Diagramme de cas d'utilisation "Gestion des produits " :



3.5.1.3 Diagramme de cas d'utilisation "Enregistrer un mouvement "



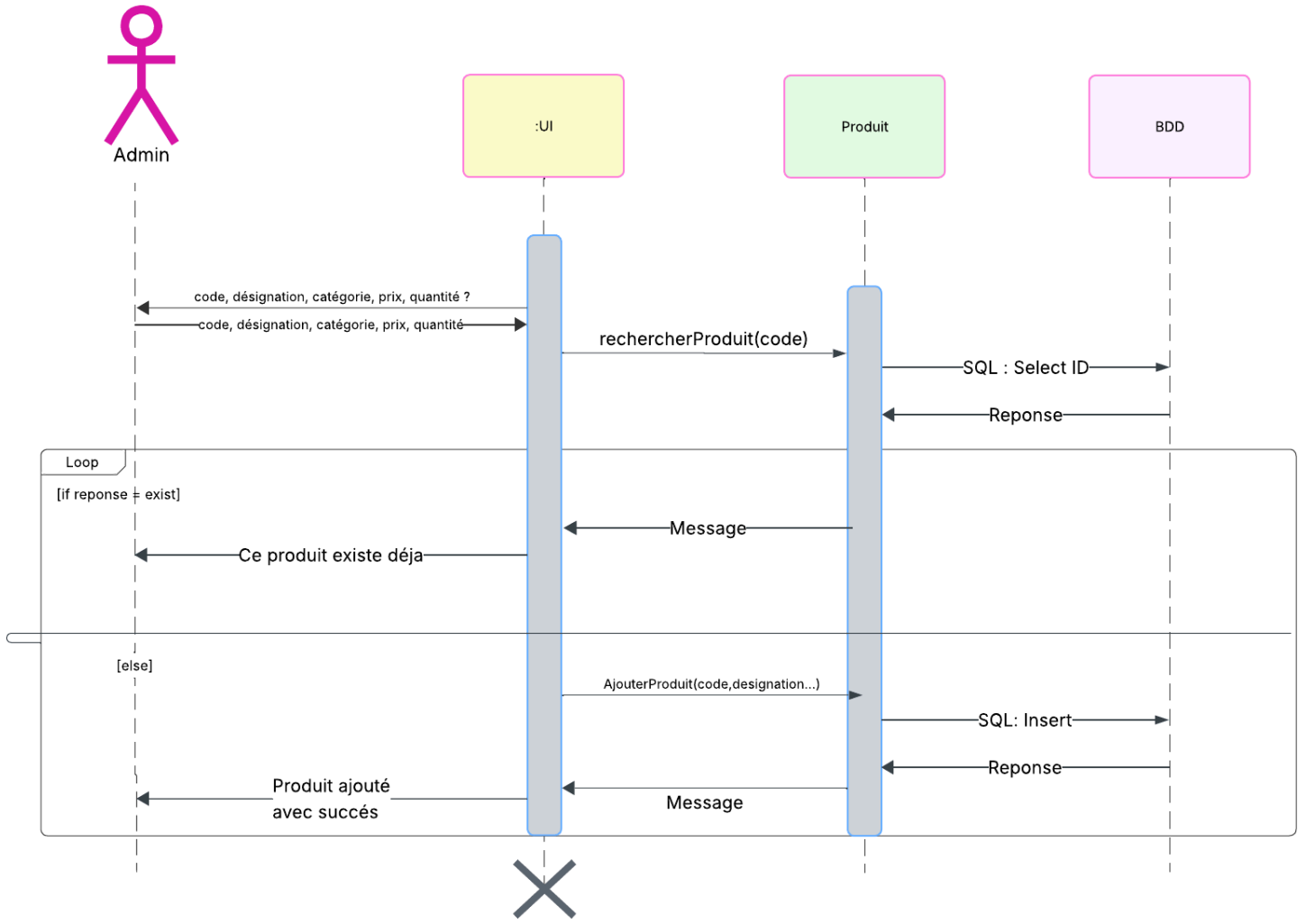
3.5.1.4 Diagramme de cas d'utilisation "**Gestion des mouvements** "



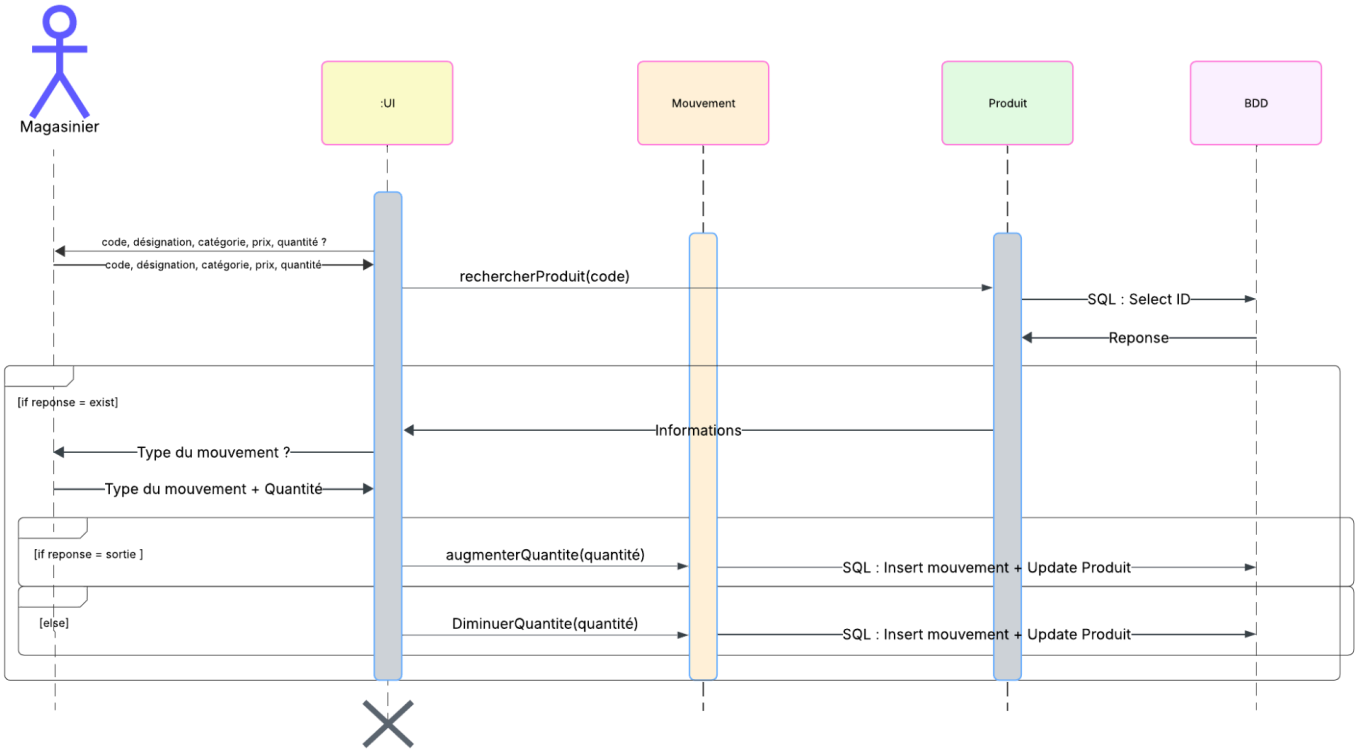
3.5.2 Diagramme de séquence

Montre le déroulement des scénarios dynamiques, comme l'ajout d'un produit ou la sortie de stock.

3.5.2.1 Diagramme de séquence "Ajout d'un produit"



3.5.2.3 Diagramme de séquence “Ajout d’un mouvement”

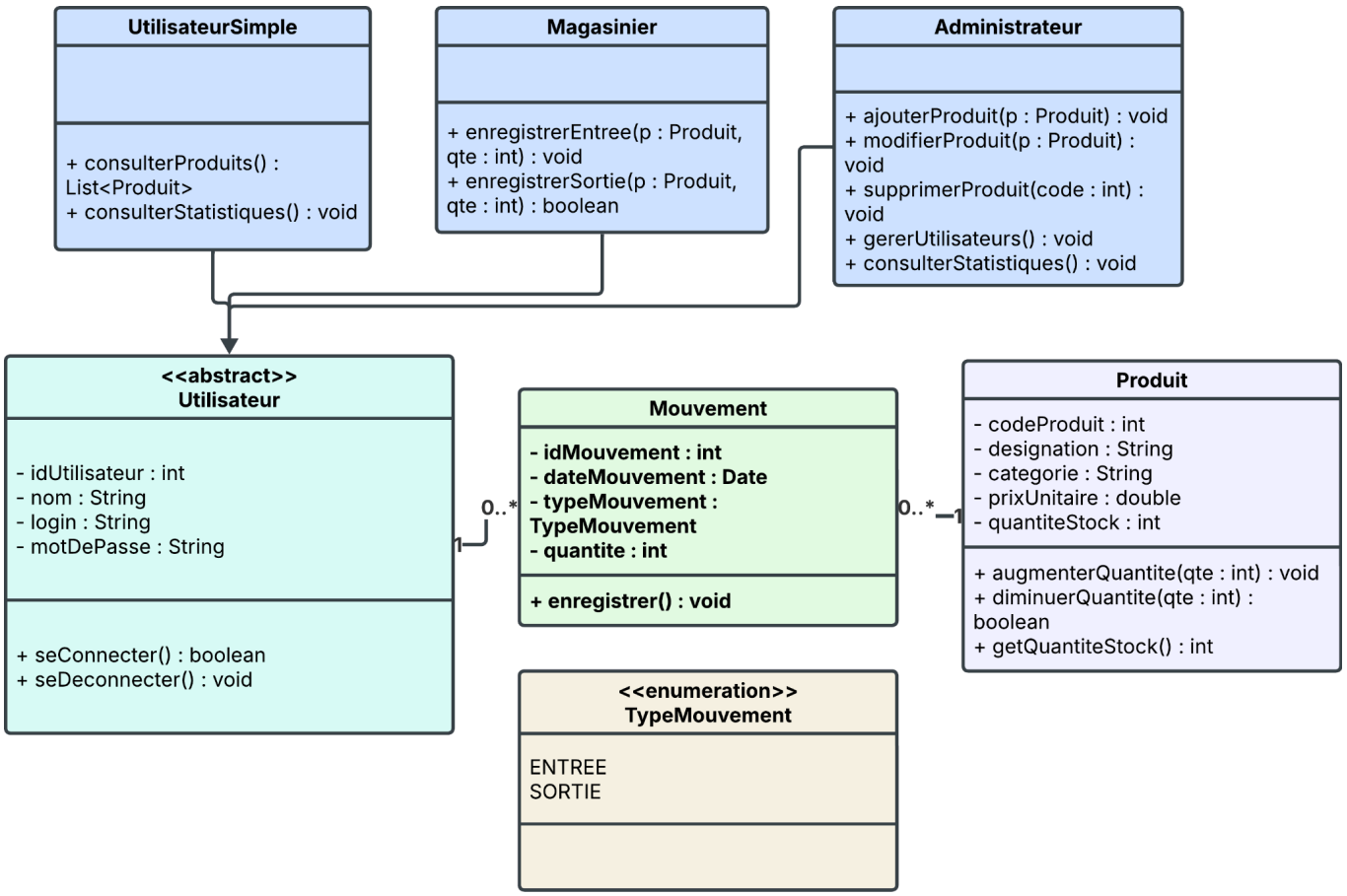


3.6 Aspect statique

3.6.1 Diagramme de classes

Le diagramme de classes occupe une place centrale dans le processus de modélisation d'un système orienté objet. Il fournit une représentation claire et structurée de l'architecture statique du système en décrivant les différentes classes, leurs attributs, leurs méthodes ainsi que les relations qui existent entre elles.

Dans cette section, nous présentons le diagramme de classes de notre système de gestion de stock, illustré à la Figure ci dessous. Ce diagramme permet de mettre en évidence les entités principales du système et d'assurer une bonne compréhension de l'organisation des données et des responsabilités de chaque composant.



Les principales classes identifiées sont les suivantes :

Produit : Cette classe représente les produits gérés par le système. Elle permet de stocker les informations essentielles relatives à chaque article ainsi que la quantité disponible en stock.

Mouvement *:* La classe *Mouvement* permet de tracer l'ensemble des entrées et sorties de stock. Chaque mouvement est associé à un produit et à un utilisateur, garantissant ainsi la traçabilité des opérations.

Utilisateur : Cette classe représente les différents utilisateurs du système. Elle est définie comme une classe abstraite dont héritent les classes *Administrateur*, *Magasinier* et *UtilisateurSimple*, permettant ainsi de factoriser les fonctionnalités communes telles que l'authentification.

Relations entre les classes

- Un Produit peut être associé à plusieurs Mouvements *(relation 1..*)*

- Un Utilisateur peut enregistrer plusieurs Mouvements *(relation 1..)**

Ces relations assurent la cohérence des données et permettent de suivre précisément l'historique des opérations de stock.

3.6.2 Modèle relationnel

Le modèle relationnel représente la traduction du diagramme de classes UML vers une structure adaptée à une base de données relationnelle. Dans ce modèle, les données sont organisées sous forme de relations équivalentes à des tables.

Le passage du modèle objet vers le modèle relationnel s'effectue en appliquant les règles de transformation suivantes :

1. Transformation des classes
   Chaque classe du diagramme de classes UML est transformée en une relation. Un attribut de la classe est choisi comme clé primaire afin d'identifier de manière unique chaque enregistrement. Les autres attributs deviennent des colonnes de la table correspondante.

2. Transformation des associations
   Selon la nature des associations, les règles suivantes sont appliquées :

   - Association 1..1 : une clé étrangère est ajoutée dans la relation correspondant à la classe ayant une multiplicité minimale égale à 1.

   - Association 1..* : une clé étrangère est ajoutée dans la relation du côté * en référence à la relation du côté 1.

   - Association .. : la relation est transformée en une table associative dont la clé primaire est composée des clés primaires des classes associées.

3. Transformation de l'héritage
   Plusieurs stratégies peuvent être adoptées pour traduire l'héritage en modèle relationnel. Dans ce projet, nous avons choisi de migrer la clé primaire de la classe mère (*Utilisateur*) vers les classes filles, tout en conservant une table *UTILISATEUR* contenant les informations communes.

Sur la base de ces règles, le modèle relationnel de données simplifié, dérivé du diagramme de classes, est le suivant :

PRODUIT (codeProduit PK, designation, categorie, prixUnitaire, quantiteStock)

UTILISATEUR (idUser PK, nom, login, motDePasse, role)

MOUVEMENT (idMouvement PK, dateMouvement, typeMouvement, quantite,
codeProduit FK → PRODUIT,
idUser FK → UTILISATEUR)

Ce modèle relationnel assure l'intégrité référentielle des données et garantit la cohérence entre les informations relatives aux produits, aux mouvements de stock et aux utilisateurs.

Chapitre 4 : Réalisation

4.1 Introduction

Après la phase de conception, la réalisation constitue l'étape où les modèles théoriques sont traduits en une application concrète. Elle consiste à mettre en œuvre les outils choisis, à développer les fonctionnalités prévues et à tester le système afin de vérifier sa conformité aux besoins exprimés.

4.2 Outils utilisés

Pour développer l'application de gestion de stock, plusieurs outils technologiques ont été mobilisés :

- **Langage de programmation** :
  - *C#* .NET Framework v9.0 WF (WinForms) (via Visual Studio).
  - Ce langage orientés objet permettent une modularité et une maintenance aisée du code.
- **Base de données** :
  - *SQL*ite (Driver intégré) pour stocker les informations relatives aux produits, aux mouvements et aux utilisateurs.
  - Ce système assurent la fiabilité, la sécurité et la cohérence des données.
- **Environnement de développement intégré (IDE)** :
  - Visual Studio, offrant des fonctionnalités avancées pour la compilation, le débogage et la gestion des projets.
- **Outils de modélisation UML** :
  - StarUML, Visual Paradigm ou Lucidchart pour la conception des diagrammes.
- **Autres outils** :
  - Git/GitHub pour la gestion de versions et la collaboration.

  - Microsoft Office (Word, PowerPoint) pour la rédaction du rapport et la préparation de la présentation.

4.3 Présentation de l'application

L'application développée répond aux besoins identifiés et se compose de plusieurs modules :

- **Module de gestion des produits**
  - Ajout, modification, suppression et recherche de produits.
  - Consultation des informations détaillées (code, désignation, catégorie, prix, quantité).
- **Module de gestion des mouvements**
  - Enregistrement des entrées (réception, retour) et des sorties (revente, utilisation, perte).

- Mise à jour automatique des quantités disponibles.
- **Module de statistiques et consultation**
- Affichage de la liste des produits avec leurs quantités actuelles.
- Consultation de l'historique des mouvements filtré par produit ou par période.
- **Interface utilisateur**
- Conçue pour être simple et intuitive.
- Navigation par menus clairs et formulaires ergonomiques.
- **Sécurité et gestion des utilisateurs**
- Authentification par identifiant et mot de passe.
- Différents rôles (administrateur, magasinier, utilisateur simple) avec des droits spécifiques.

Nos Améliorations et Extensions fonctionnelles:

- ***Smart Insights & Analyse Prédictive***: Algorithme d'assistance à la décision générant des recommandations stratégiques automatiques basées sur les flux de mouvements et les seuils critiques de stock.

- ***Dashboard Analytique & KPI*** : Centralisation des indicateurs clés de performance (valeur de l'actif, taux de rotation, monitoring des alertes) via des visualisations graphiques haute fidélité en temps réel.

- ***Reporting & Exportation de Données*** : Moteur d'extraction de données structurées vers Microsoft Excel, garantissant une interopérabilité totale pour l'archivage et l'analyse externe.

- **Gestion Multimédia Intégrée** : Système d'indexation visuelle des produits avec traitement dynamique des images, facilitant l'identification rapide et la réduction des erreurs de saisie.

- ***Expérience Utilisateur & Mode Plein écran***: Mode d'affichage plein écran optimisé pour les postes de contrôle, maximisant la densité d'information et le confort visuel lors des inventaires.

- ***Interface Adaptive (Responsive Design)*** : Architecture UI dynamique exploitant des algorithmes de redimensionnement en temps réel pour une ergonomie parfaite sur toutes les résolutions d'écran.

- ***Contrôle d'Accès Granulaire (RBAC)*** : Sécurisation des données par une gestion stricte des privilèges utilisateurs, adaptant l'interface selon les responsabilités (Admin, Magasinier, Consultant).

Chapitre 5 : Conclusion

5.1 Bilan du projet

Le projet de gestion de stock a permis de mettre en pratique les connaissances acquises en génie logiciel, notamment en matière de modélisation UML, de conception de bases de données et de développement d'applications. L'application réalisée répond aux besoins essentiels identifiés : gestion des produits, enregistrement des mouvements, mise à jour automatique des quantités et génération de statistiques.

Ce travail a également permis de renforcer les compétences en analyse des besoins, en conception orientée objet et en programmation Java/C#.

5.2 Difficultés rencontrées

Au cours du projet, plusieurs difficultés ont été rencontrées :

- La traduction des besoins exprimés en modèles UML clairs et cohérents.
- La gestion des relations entre les différentes entités dans la base de données.
- L'implémentation de certaines fonctionnalités complexes, comme la mise à jour automatique des stocks après chaque mouvement.
- La coordination et la répartition des tâches au sein du binôme.

Ces obstacles ont été surmontés grâce à une bonne organisation, à la recherche documentaire et à l'utilisation d'outils adaptés.

5.3 Perspectives d'amélioration

Bien que l'application réponde aux besoins de base, plusieurs améliorations peuvent être envisagées :

- Aj● Gestion multi-entrepôts pour les entreprises disposant de plusieurs sites.
- Génération de rapports détaillés exportables (PDF, Excel).
- Intégration d'une interface web ou mobile pour faciliter l'accès au système.
- Mise en place de fonctionnalités avancées de sécurité et de sauvegarde des données.