Conception complète d'une application mobile SaaS **de gestion de stock**

Module : Développement mobile

Réalisé par :
Othman Bekhouche
Wassim Jaafri
Hamza Azouagh

Supervisé par :
Pr. Reda Ghafir

Année universitaire 2024-2025

Table des matières

Chapitre 1

Introduction générale

Dans un environnement économique de plus en plus compétitif et numérisé, la maîtrise de la chaîne logistique est devenue un enjeu stratégique majeur pour les entreprises, quelle que soit leur taille. La gestion des stocks, autrefois limitée à de simples registres papier ou des tableurs isolés, exige désormais réactivité, précision et mobilité.

C'est dans cette optique que s'inscrit notre projet : la conception et le développement d'une application mobile en mode SaaS (Software as a Service) dédiée à la gestion de stock. Cette solution vise à transformer la gestion des inventaires en un processus automatisé, centralisé et accessible en temps réel depuis n'importe quel terminal mobile.

L'objectif principal de ce travail est d'offrir aux petites et moyennes entreprises un outil moderne capable de synchroniser instantanément les ventes, les achats et les mouvements de stock via le Cloud. En s'appuyant sur la puissance du WLangage avec WinDev Mobile et la robustesse d'une architecture API REST, nous proposons une plateforme multi-utilisateur garantissant la sécurité et l'intégrité des données commerciales.

Ce rapport détaille la démarche de réalisation du projet à travers quatre axes principaux :

- **L'analyse fonctionnelle :** Identification des besoins métiers et des acteurs du système.
- **La conception logicielle :** Modélisation UML des interactions et de la structure des données.
- **Les choix techniques :** Présentation de la stack technologique (WinDev Mobile, MySQL, Spring Boot/Node.js).
- **L'architecture système :** Description de l'organisation en 3-tiers pour assurer la scalabilité de la solution.

1.1 Objectifs du projet

L'objectif central de ce projet est de fournir une solution logicielle clé en main qui répond aux problématiques de gestion de flux physiques dans un environnement mobile. Ce projet s'articule autour de trois axes majeurs :

1. Optimisation de la gestion opérationnelle

- **Mobilité totale :** Permettre aux gestionnaires de stock et au personnel de terrain d'effectuer des inventaires, des réceptions et des ventes directement depuis leur smartphone, éliminant ainsi les ressaisies sur ordinateur.

— **Suivi en temps réel :** Assurer une visibilité instantanée sur les niveaux de stock (disponibles, réservés, en commande) pour éviter les ruptures de stock ou le sur-stockage.
— **Automatisation documentaire :** Générer automatiquement les pièces justificatives (factures, bons de commande, rapports de mouvements) pour gagner en productivité.

2. Fiabilité et Sécurité des données

— **Centralisation SaaS :** Utiliser le mode SaaS pour garantir que tous les utilisateurs, quel que soit leur emplacement, travaillent sur une base de données unique et synchronisée.
— **Traçabilité rigoureuse :** Enregistrer l'historique de chaque mouvement de stock (qui a fait quoi et quand) afin de responsabiliser les acteurs et faciliter les audits.
— **Contrôle d'accès :** Mettre en place un système de rôles (ACL) pour restreindre l'accès aux données sensibles (marges bénéficiaires, gestion des comptes) selon le profil de l'utilisateur.

3. Défis Techniques et Apprentissage

— **Interopérabilité :** Maîtriser la communication entre une application native **WinDev Mobile** et un backend distant via des services web REST.
— **Modélisation complexe :** Concevoir une base de données relationnelle capable de gérer des relations complexes entre produits, fournisseurs, commandes et mouvements de stock.
— **Expérience Utilisateur (UX) :** Créer une interface ergonomique et intuitive, minimisant le nombre de clics pour les opérations courantes en entrepôt.

1.2 Méthodologie de travail

Pour mener à bien ce projet de développement, nous avons adopté une méthodologie rigoureuse structurée en plusieurs phases clés, alliant les standards de l'ingénierie logicielle et l'agilité.

1. Phase d'Analyse et de Spécification

Cette étape initiale a consisté à recueillir les besoins métiers spécifiques à la gestion de stock. Nous avons défini les processus critiques (entrées/sorties, gestion des alertes) et identifié les différents profils d'utilisateurs. Cette phase a abouti à la rédaction du cahier des charges fonctionnel.

2. Conception et Modélisation UML

Avant l'écriture du code, nous avons utilisé le langage de modélisation UML (Unified Modeling Language) pour concevoir l'architecture du système :

— **Diagrammes de Cas d'Utilisation :** Pour définir le périmètre du système et les interactions acteurs-fonctions.
— **Diagramme de Classes :** Pour structurer la base de données relationnelle et définir les entités.
— **Diagrammes de Séquence :** Pour modéliser la dynamique des échanges entre l'application mobile et le serveur lors de processus critiques comme la commande ou l'authentification.

3. Développement et Architecture 3-Tiers

Le développement a été réalisé selon une architecture découplée afin de garantir la scalabilité et la maintenance du système :

— **Développement Frontend :** Utilisation de l'AGL **WinDev Mobile** pour la création de l'interface utilisateur native, en mettant l'accent sur l'ergonomie mobile.
— **Développement Backend :** Mise en place d'une couche middleware (API REST) pour traiter la logique métier, assurant l'interface entre le client mobile et les données.
— **Persistance :** Structuration de la base de données sous **MySQL** pour assurer la fiabilité des transactions.

Chapitre 2

Étude fonctionnelle

2.1 Acteurs du système

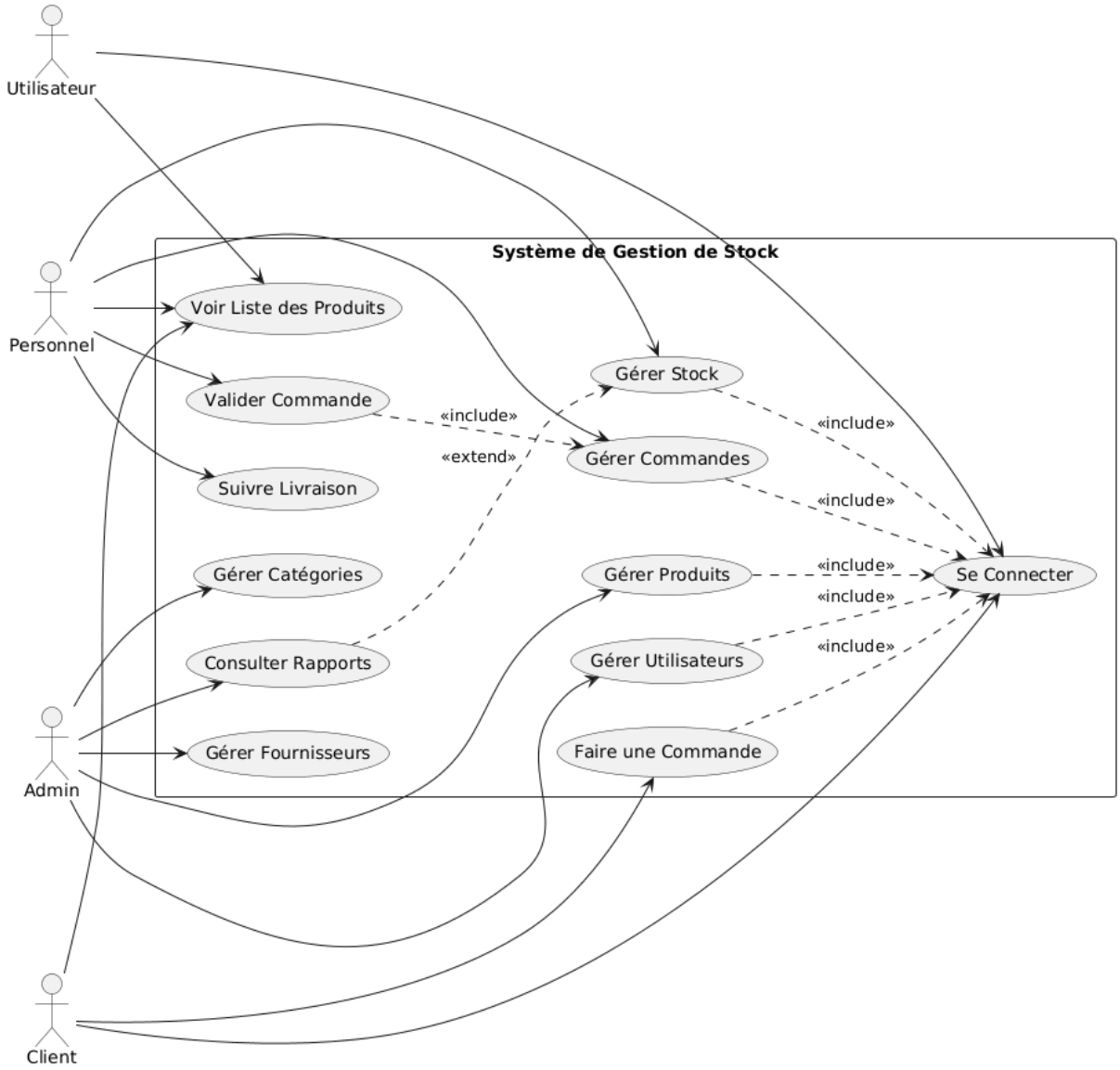
- **Administrateur :** C'est le rôle possédant le plus haut niveau d'accréditation. Il supervise l'intégralité de la chaîne logistique et la configuration du système, gère les utilisateurs et les données.
- **Personnel :** effectue les ventes, achats et met à jour le stock.
- **Client :** Le client interagit avec le système principalement pour ses propres besoins d'approvisionnement, consulte le catalogue et passe des commandes.
- **Utilisateur :** représente n'importe quelle personne interagissant avec l'interface mobile.

2.2 Fonctionnalités principales

- Sécurité et Accès
  Authentification sécurisée : Mise en œuvre via JWT (JSON Web Token) pour des échanges sécurisés avec les Webservices ou via la gestion native des sessions WinDev.
  Gestion des profils : Attribution de rôles (Admin, Personnel, Client) permettant de restreindre l'accès aux données sensibles conformément au diagramme de cas d'utilisation.
- Gestion dynamique du Stock (ajout, suppression, modification)
  L'objectif est d'assurer une visibilité totale sur les actifs physiques de l'entreprise.
- Gestion des fournisseurs et des commandes d'achat.
- Gestion des clients et des ventes.
- Génération automatique de factures.
- Consultation de statistiques et rapports (ventes, marges, ruptures).

Chapitre 3

Diagramme de cas d'utilisation (Use Case)



Le diagramme de cas d'utilisation (UML) ci-dessus définit le périmètre de notre système de gestion de stock. Il identifie les interactions entre les différents acteurs et les fonctionnalités offertes par l'application.

3.1 Identification des Acteurs

Utilisateur : L'acteur de base (généraliste) pouvant consulter les produits.

Personnel : Responsable du suivi opérationnel (validation des commandes, livraisons).

Admin : Détient les droits d'administration (gestion des utilisateurs, catégories, fournisseurs et rapports).

Client : Acteur externe qui interagit avec le système pour passer des commandes.

3.2 Description des fonctionnalités

Authentification

Tous les processus critiques (Gérer Stock, Commandes, Produits, etc.) incluent obligatoirement l'étape "Se Connecter" via une relation «include».

Ventes et commandes

Le Client peut "Faire une commande". Le Personnel assure ensuite la "Validation" et le "Suivi de livraison".

Gestion des stocks

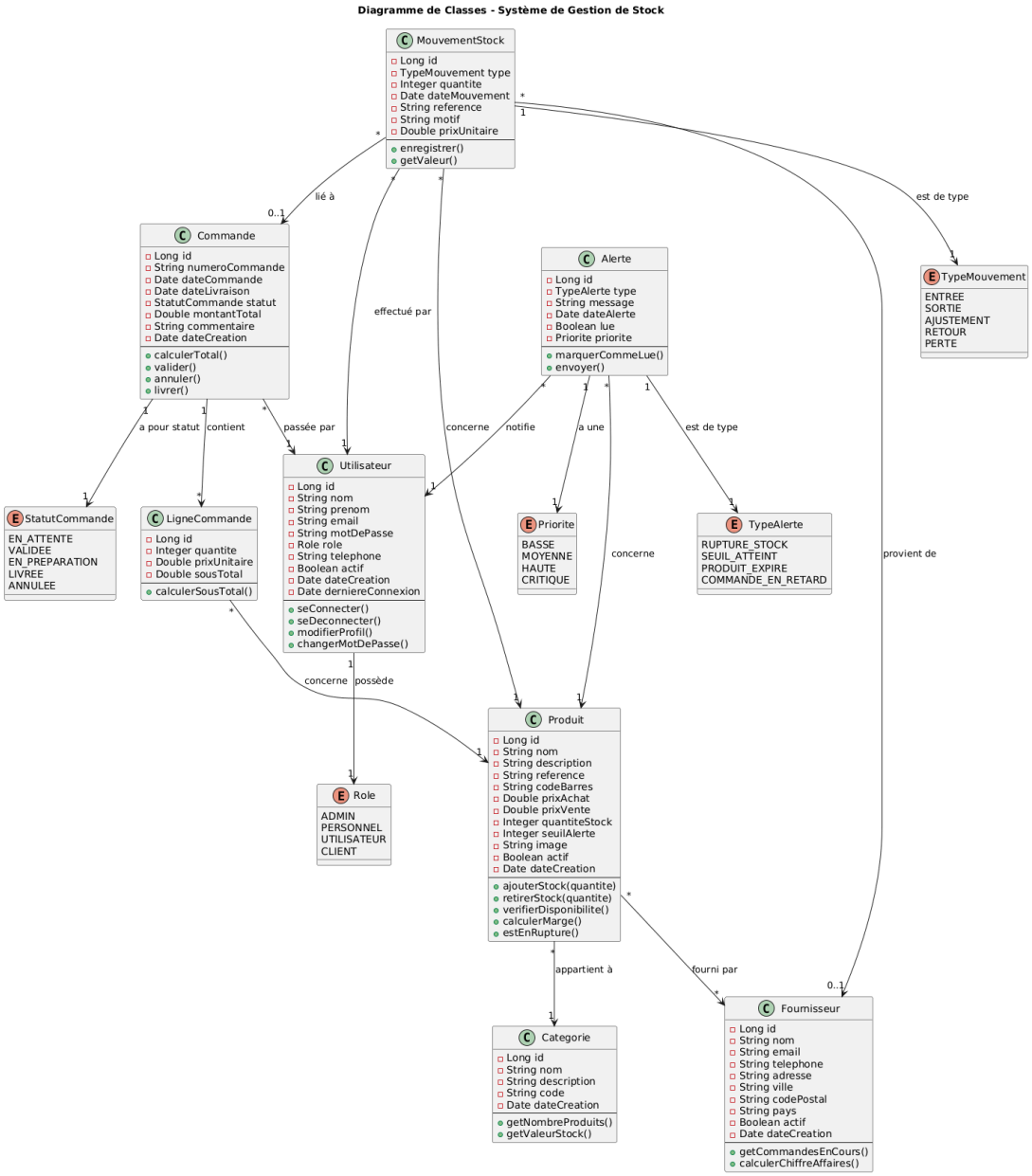
L'Utilisateur peut "Gérer le Stock" et "Gérer les Commandes". À noter que la validation d'une commande peut entraîner une mise à jour des stocks via l'extension «extend».

Administration

L'Admin a un accès exclusif à la configuration du système : gestion des fournisseurs, des catégories de produits et des comptes utilisateurs.

Chapitre 4

Diagramme de classes



Le diagramme de classes ci-dessus présente la structure statique du système de gestion de stock. Il définit les entités du domaine, leurs attributs, leurs comportements (méthodes)

ainsi que les relations qui les unissent.

Analyse des entités principales

Le système s'articule autour de quatre piliers majeurs :

1. Gestion du Catalogue (Produit et Categorie) : Chaque Produit est caractérisé par ses prix (achat/vente) et son niveau de stock. Il appartient obligatoirement à une Categorie. Le système intègre des méthodes de calcul de marge et de vérification de rupture.
2. Flux de Stock (MouvementStock) : Cette entité trace chaque entrée, sortie ou ajustement. Un mouvement est lié à un Produit, effectué par un Utilisateur, et peut être associé à une Commande.
3. Processus de Vente (Commande et LigneCommande) : Une Commande regroupe plusieurs lignes de commande. Son cycle de vie est régi par l'énumération StatutCommande (En attente, Validée, Livrée, etc.).
4. Système d'Alertes : La classe Alerte permet de notifier les utilisateurs selon des seuils critiques (rupture, produit expiré) avec différents niveaux de priorité.

Relations et règles de gestion

Le diagramme met en évidence plusieurs règles métier critiques :

— Traçabilité : Chaque mouvement de stock est "effectué par" un Utilisateur, garantissant un historique complet des actions.

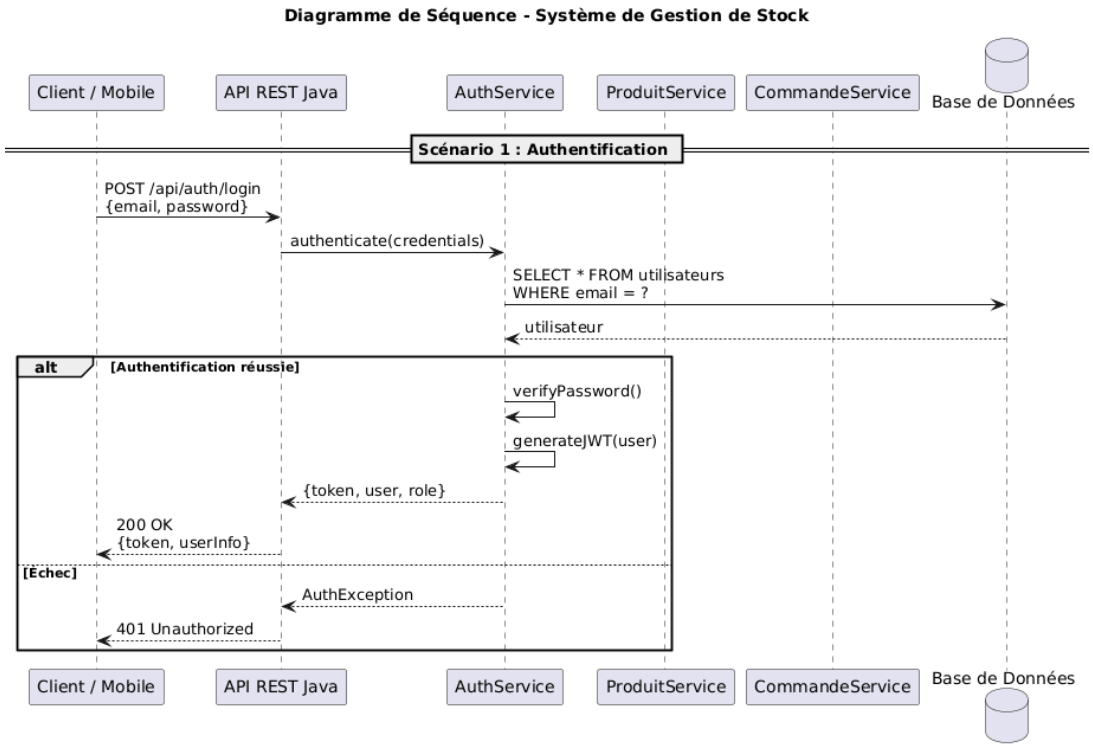
— Approvisionnement : Un Produit est fourni par un Fournisseur (relation 1..*), permettant de gérer les contacts et le chiffre d'affaires généré par partenaire.

— Sécurité : L'accès aux fonctionnalités est protégé par un système de rôles (ADMIN, PERSONNEL, CLIENT) défini dans l'entité Utilisateur.

Chapitre 5

Diagrammes de séquence

5.1 Diagrammes de Séquence - Authentification



Ce diagramme de séquence illustre les interactions entre les différents composants du système lors de l'exécution de scénarios spécifiques. Il permet de visualiser le flux de messages entre le client, l'API et les services métier.

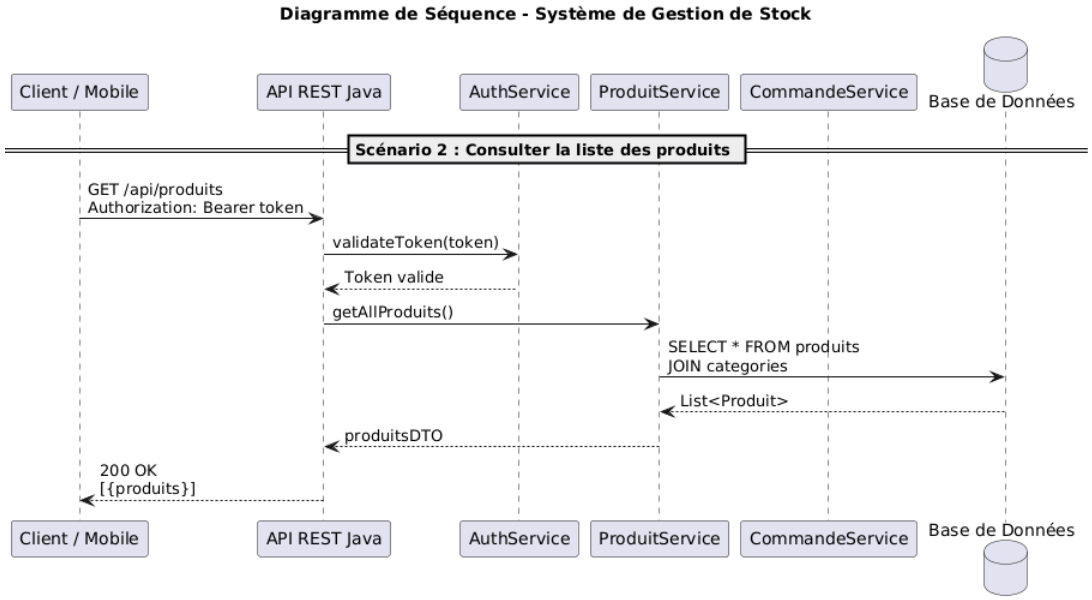
Ce premier scénario détaille le processus de connexion sécurisée d'un utilisateur au système de gestion de stock.

1. Initialisation de la requête :
   Requête Client : Le client (Mobile ou Web) envoie une requête POST /api/auth/login contenant les identifiants (email et password) vers l'API REST Java.
   Délégation au Service : L'API délègue la logique de sécurité à l' AuthService via l'appel de la méthode authenticate(credentials).
2. Vérification et Traitement métier :
   Accès aux Données : L' AuthService interroge la base de données pour récupérer

les informations de l'utilisateur correspondant à l'email fourni (SELECT * FROM utilisateurs...).
Validation : Une fois l'utilisateur récupéré, le service effectue deux actions internes : verifyPassword() : Pour comparer le mot de passe saisi avec celui stocké en base. generateJWT(user) : En cas de succès, un jeton de sécurité (JSON Web Token) est généré pour maintenir la session de l'utilisateur.

3. (Bloc "Alt") :
Le système gère deux résultats possibles pour cette transaction :
Succès (Authentification réussie) : Le service renvoie le jeton, les informations utilisateur et son rôle à l'API. Le client reçoit alors un code 200 OK avec ces données pour autoriser l'accès à l'interface.
Échec : En cas d'identifiants incorrects, une exception AuthException est levée par le service. L'API renvoie alors une erreur 401 Unauthorized au client, lui interdisant l'accès.

5.2 Diagrammes de Séquence - Consulter la liste des produits

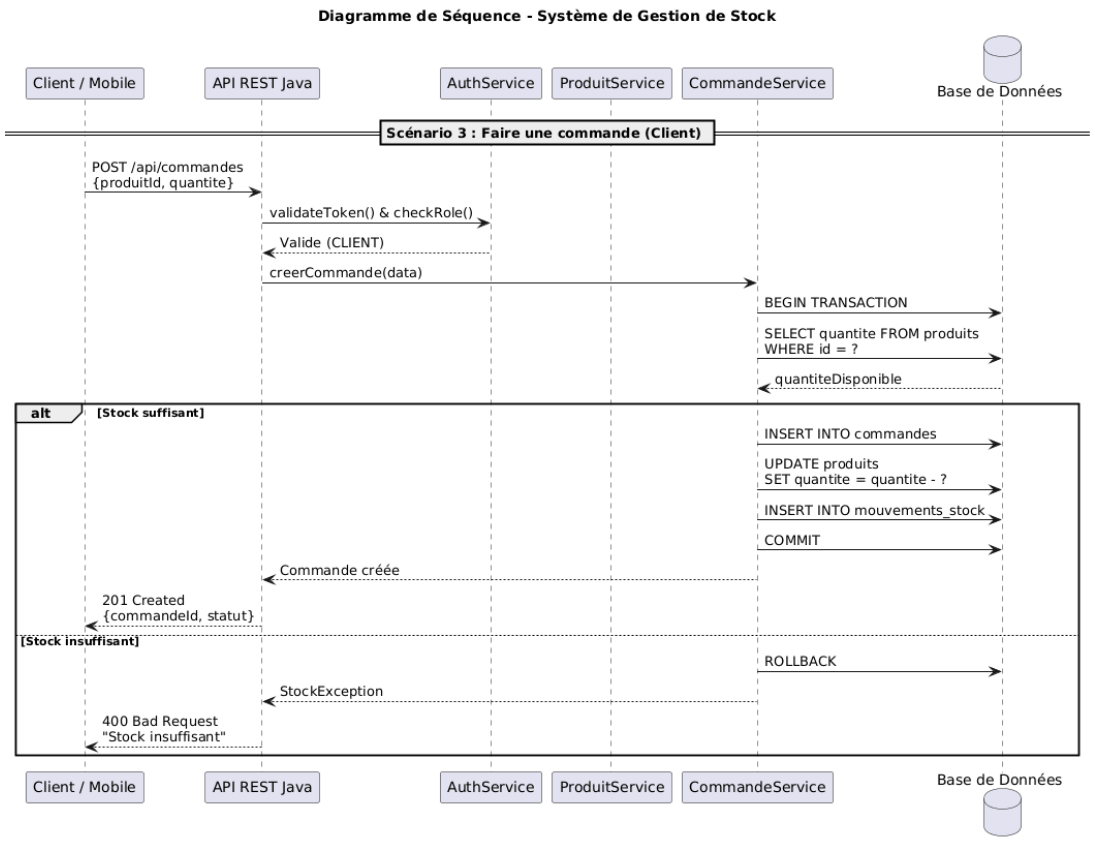


Ce diagramme illustre le processus de récupération de l'inventaire complet par un utilisateur authentifié. Il met en évidence l'interaction entre le Client Mobile, l'API REST Java, les services métiers et la persistance des données.

1. Authentification et Sécurité : Le client envoie une requête GET vers l'endpoint /api/produits. L'API extrait le Bearer Token du header pour le faire valider par l'AuthService.
2. Logique Métier : Une fois le jeton validé, l'API délègue la récupération des données au ProduitService via la méthode getAllProduits().
3. Accès aux Données : Le service interroge la Base de Données en effectuant une jointure (JOIN categories) pour récupérer les produits avec leurs catégories respectives.

4. Transformation et Réponse : Les entités SQL sont transformées en objets de transfert de données (DTO), puis renvoyées au client sous forme de flux JSON avec un code de succès 200 OK.

5.3 Diagrammes de Séquence - Faire une commande (Client)



Ce scénario décrit le processus critique de création d'une commande par un utilisateur. Contrairement à la simple consultation, ce flux nécessite une vérification stricte des droits d'accès, une validation des stocks et une intégrité des données via une transaction SQL.

1. Vérification Initiale : Le processus débute par une requête POST contenant l'identifiant du produit et la quantité souhaitée. L'API REST Java sollicite l'AuthService pour valider non seulement le jeton, mais aussi le rôle spécifique (CLIENT) de l'utilisateur.
2. Gestion Transactionnelle : Une fois l'appel transmis au CommandeService, une transaction est ouverte (BEGIN TRANSACTION) auprès de la Base de Données pour garantir l'atomicité des opérations suivantes.
3. Cas Nominal (Stock suffisant) : Le système vérifie la quantité disponible.

   Si le stock est suffisant, trois opérations sont effectuées : insertion de la commande, mise à jour du stock produit et enregistrement du mouvement de stock.

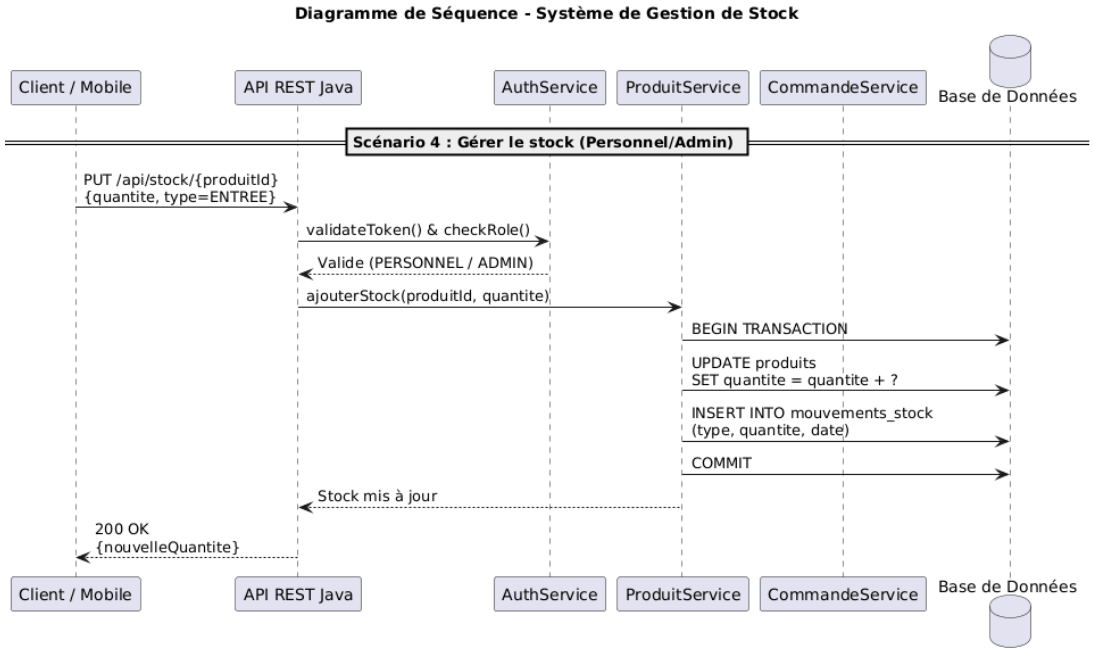
   La transaction est validée (COMMIT), et le client reçoit une confirmation 201 Created.

4. Cas d'Échec (Stock insuffisant) :

   Si la quantité demandée dépasse le stock, une opération de ROLLBACK est déclenchée pour annuler toute modification en cours.

   Une exception métier (StockException) est remontée, aboutissant à une erreur 400 Bad Request retournée à l'utilisateur.

5.4 Diagrammes de Séquence - Gérer le stock



Ce diagramme illustre le processus par lequel un membre du personnel ou un administrateur met à jour la quantité d'un produit en stock. Il met en évidence l'interaction entre le front-end (Client Mobile), l'API REST, les services métiers et la persistance des données, tout en garantissant l'intégrité des transactions.

1. Sécurisation et Authentification : Avant toute modification, l'API REST Java interroge l'AuthService. On vérifie non seulement la validité du jeton (Token), mais aussi si l'utilisateur possède les droits requis (PERSONNEL ou ADMIN). Cela garantit qu'un client standard ne peut pas modifier les niveaux de stock.
2. Logique Métier et Transactionnelle : Une fois autorisé, le ProduitService prend le relais. L'aspect le plus important ici est l'utilisation d'une transaction SQL (BEGIN TRANSACTION ... COMMIT). Cela assure l'atomicité de l'opération :
   (a) La table produits est mise à jour avec la nouvelle quantité.
   (b) Une trace est insérée dans la table mouvements-stock pour l'historique. Si l'une de ces étapes échoue, aucune modification n'est enregistrée.
3. Retour d'information : Une fois la base de données mise à jour, le système confirme le succès de l'opération en renvoyant un code HTTP 200 OK au mobile, accompagné de la nouvelle valeur de la quantité pour une mise à jour immédiate de l'interface utilisateur.

Chapitre 6

Choix technique

6.1 Environnement

— IDE : **WinDev Mobile 2025**

WinDev est un atelier de génie logiciel (AGL) édité par la société française PC SOFT désormais détenue par l'actionnaire canadien Constellation Software Inc. PC SOFT a mis fin aux licences perpétuelles, imposant désormais un abonnement pour accéder aux sources. Windev est conçu pour développer des applications, principalement orientées données pour Windows et également pour Linux, .NET et Java. Il propose son propre langage : le WLangage. La première version de WinDev est sortie en 1993.

— Base de données : **MySQL**

MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde[5], autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, PostgreSQL et Microsoft SQL Server.

— API : **REST**

Une API REST (Representational State Transfer) est un style architectural pour les API web qui définit un ensemble de contraintes pour la communication entre systèmes, utilisant le protocole HTTP et permettant une interaction simple, évolutive et découplée grâce à des requêtes standardisées (GET, POST, PUT, DELETE) et une gestion sans état (stateless) des transactions, idéal pour les architectures distribuées comme le web et les microservices.

6.2 Architecture

Le système repose sur une architecture 3-tiers, une structure standard garantissant que les modifications apportées à une couche (ex : changement d'interface) n'impactent pas directement les autres (ex : base de données).
Architecture **3-tiers** :

1. Couche de Présentation (Frontend) :
   Cette couche représente l'interface visible par l'utilisateur final, Développée avec WinDev Mobile, elle gère l'affichage des données et l'interaction utilisateur. Elle communique avec la couche métier via des requêtes HTTP (REST). Exemple : Lorsqu'un utilisateur clique sur "Consulter la liste des produits", le client mobile émet un GET /api/produits accompagné d'un jeton d'authentification.

2. Couche de Logique Métier (Middleware / Backend) :
   C'est le "cerveau" de l'application, situé entre l'interface et les données. Elle s'appuie sur une API REST Java composée de services spécialisés : AuthService, ProduitService et CommandeService.

3. Couche de Données (Backend) :
   Cette couche est responsable du stockage persistant et de l'intégrité des informations.
   — Technologie : Système de Gestion de Base de Données Relationnelle (SGBDR) MySQL.
   — Fonctionnement : Les données sont organisées en tables reliées (ex : produits liées aux categories).
   — Fiabilité : Elle supporte les opérations transactionnelles (COMMIT / ROLLBACK), essentielles pour garantir qu'une commande ne soit jamais enregistrée si la mise à jour du stock échoue.

Chapitre 7

Stack technique

Composant	**Technologie utilisée**
Frontend mobile	Windev Mobile (WLanguage)
Backend API	Node.js / PHP Symfony / Java Spring Boot
Base de données	MySQL
Authentification	JSON Web Token (JWT)
Communication	API REST (HTTP + JSON)

WINDEV Mobile : est un Atelier de Génie Logiciel (AGL) complet de la société PC SOFT, conçu pour créer rapidement des applications natives pour Android et iOS (iPhone/iPad) en utilisant un langage de programmation unique, le Langage W (WLangage).

Node.js : L'environnement d'exécution qui permet d'utiliser JavaScript côté serveur. Il assure la fluidité entre le frontend et le backend.

Symfony : c'est un framework d'applications web PHP libre et open source, ainsi qu'un ensemble de bibliothèques de composants PHP réutilisables. Il vise à accélérer la création et la maintenance d'applications web et à automatiser les tâches de codage répétitives.

Spring Boot : c'est un framework Java open source utilisé pour programmer des applications Spring autonomes et prêtes pour la production, grâce à un ensemble de bibliothèques qui simplifient le démarrage et la gestion des projets.

MySQL : est le système de gestion de bases de données (SGBD) relationnel le plus utilisé au monde. Il s'agit d'un logiciel open source qui permet de stocker, d'organiser et de récupérer des données de manière structurée.

Chapitre 8

Conclusion générale

La réalisation de ce projet de conception d'une application mobile de gestion de stock a permis de couvrir l'intégralité du cycle de développement d'une solution logicielle moderne, de l'analyse fonctionnelle jusqu'aux choix d'architectures techniques.

À travers l'utilisation de la modélisation UML (cas d'utilisation, classes et séquences), nous avons pu structurer une réponse précise aux besoins des entreprises : une gestion fluide des flux de marchandises, une traçabilité rigoureuse des mouvements de stock et une interface intuitive adaptée à la mobilité. L'architecture 3-tiers adoptée garantit à la fois la sécurité des échanges via l'implémentation de jetons JWT et l'intégrité des données grâce à la gestion transactionnelle de MySQL.

L'utilisation de WinDev Mobile comme environnement de développement, couplée à une architecture API REST, démontre la pertinence des outils de génie logiciel pour produire des applications robustes et multi-plateformes en un temps optimisé. Cette approche SaaS offre ainsi une flexibilité d'évolution majeure, permettant l'ajout futur de fonctionnalités telles que l'analyse prédictive des ventes ou l'intégration de scanners de codes-barres plus avancés.

En conclusion, ce travail nous a permis de consolider nos compétences en développement mobile et en ingénierie logicielle.