1) Contexte et vision

L'objectif du projet est de concevoir et développer une API REST modulaire dédiée à la gestion logistique.

Cette application permet de gérer les produits (SKU), les stocks multi-entrepôts, les commandes clients, les approvisionnements fournisseurs, ainsi que les expéditions et le suivi des livraisons.

L'objectif principal est de garantir la traçabilité complète des opérations logistiques, d'assurer un contrôle rigoureux du stock (aucun stock négatif), et d'automatiser les règles métier critiques telles que la réservation, l'expédition et la réception.

La solution repose sur Spring Boot, Spring Data JPA, MapStruct, Swagger/OpenAPI et des tests unitaires avec JUnit 5 et Mockito.

2) Objectifs pédagogiques

L'application doit permettre aux apprenants de :

Comprendre et appliquer une architecture multicouche : Controller, Service, Repository, DTO.

Implémenter la persistance avec JPA et manipuler correctement les dates et statuts métiers.

Développer des endpoints REST CRUD avec pagination, validation et cohérence métier.

Gérer les exceptions de manière centralisée à l'aide d'un contrôleur global d'erreurs.

Mettre en œuvre des règles logistiques avancées : réservation de stock, picking, expédition, réapprovisionnement.

Écrire des tests unitaires et d'intégration (JUnit 5, Mockito) pour valider la logique métier.

3) Périmètre fonctionnel

Le système couvre les modules suivants :

Authentification et profils :

Gestion simple des utilisateurs avec rôles prédéfinis : ADMIN, WAREHOUSE_MANAGER et CLIENT.

Pas de sécurité JWT, mais simulation des rôles dans la logique métier.

Produits et stocks :

- Gestion complète des produits : création, mise à jour, suppression, activation et désactivation.
- Gestion de l'inventaire par entrepôt : qtyOnHand (quantité physique) et qtyReserved (quantité réservée).
- Calcul automatique de la disponibilité : available = qtyOnHand - qtyReserved.

Gestion des mouvements de stock : INBOUND (entrée), OUTBOUND (sortie), ADJUSTMENT (correction manuelle).

Fournisseurs et approvisionnements :

Gestion des fournisseurs et des bons de commande (Purchase Orders).

Réception complète ou partielle d'un bon de commande.

Mise à jour automatique du stock lors de la réception.

Commandes clients :

Création et gestion de commandes (Sales Orders) avec plusieurs lignes.

Cycle de vie : **CREATED → RESERVED → SHIPPED → DELIVERED → CANCELED**.

Réservation obligatoire avant expédition et interdiction du stock négatif.

Gestion automatique des backorders pour les quantités manquantes.

Expéditions et tracking :

Création d'expéditions liées à une commande.

Attribution d'un transporteur et numéro de suivi.

Statuts : **PLANNED, IN_TRANSIT, DELIVERED.**

Respect du cut-off horaire logistique.

Reporting et recherche :

Filtrage par date, statut ou type de mouvement.

Calcul de statistiques de base : nombre de commandes, taux de livraison, détection des ruptures.

4) Rôles et acteurs

Le rôle ADMIN gère les utilisateurs, les produits, les entrepôts et les paramètres globaux.

Le rôle WAREHOUSE_MANAGER gère les stocks, les mouvements, les réservations et les expéditions.

Le rôle CLIENT crée des commandes et consulte leur statut ou leur tracking.

5) Règles métier avancées

- Interdiction stricte du stock négatif : toute sortie doit vérifier la disponibilité.
- Réservation obligatoire avant expédition : une commande ne peut être expédiée que si elle est réservée.
- Allocation multi-entrepôts : en cas d'insuffisance, l'application répartit sur plusieurs entrepôts selon la priorité.
- Création automatique de backorders pour les quantités non disponibles.
- Cut-off logistique : les commandes créées après 15h sont planifiées pour le jour ouvré suivant.
- Réservation temporaire (TTL) : au-delà d'un délai (ex. 24h), la réservation est libérée.

Capacité maximale d'expédition par créneau : si la limite est atteinte, les expéditions sont replanifiées.

6) Modèle conceptuel

Les principales entités sont :

User (id, email, passwordHash, rôle, actif).

Product (id, sku, name, category, active).

Warehouse (id, code, name).

Inventory (id, product, warehouse, qtyOnHand, qtyReserved).

InventoryMovement (id, product, warehouse, type, qty, occurredAt).

Supplier (id, name, contact).

PurchaseOrder (id, supplier, status, createdAt) et POLine (product, qty).

Client (id, name).

SalesOrder (id, client, warehouse, status) et SOLine (product, qty, price).

Shipment (id, salesOrder, carrier, trackingNumber, status).

7) Architecture logicielle

L'application suit une architecture en couches claire et testable :

La couche Controller gère les endpoints REST et la validation des données d'entrée à l'aide des annotations @Valid et @Validated.

La couche Service implémente les règles métier : réservation, ajustement de stock, transitions de statut, génération de backorders.

La couche Repository utilise Spring Data JPA pour interagir avec la base.

Les DTO représentent les données échangées entre le backend et le client sans exposer les entités JPA.

Les mappers MapStruct convertissent les entités vers leurs DTO et inversement.

Les tests utilisent JUnit 5 et Mockito pour valider la cohérence métier.

8) Validation et gestion des exceptions

Validation des données

Toutes les requêtes REST doivent être validées avant exécution :

Les annotations @NotNull, @NotBlank, @Min, @Max, @Email, etc. sont utilisées pour contrôler les entrées.

La validation s'applique aux DTO des requêtes (ex. ProductDTO, OrderDTO).

En cas d'erreur, un message explicite est renvoyé au format JSON décrivant le champ invalide et la contrainte violée.

Exemple de message de retour attendu :

{ "timestamp": "2025-10-26T11:00:00Z", "status": 400, "message": "SKU must not be blank", "path": "/api/products" }.

Gestion des exceptions globales

L'application doit centraliser la gestion des erreurs via un composant @ControllerAdvice.

Ce composant capture toutes les exceptions levées dans les contrôleurs et renvoie une réponse structurée au client.

Les exceptions gérées incluent :

ResourceNotFoundException : ressource inexistante (404).

BusinessException : violation d'une règle métier (400).

ValidationException : erreur de validation (400).

StockUnavailableException : tentative de réservation au-delà du stock disponible (409).

GenericException : erreur interne inattendue (500).

Chaque réponse d'erreur est renvoyée au format JSON standardisé contenant :

le timestamp,

le code HTTP,

le message explicatif,

le path de la requête,

et éventuellement un detail (ex. champ invalide ou valeur incorrecte).

L'objectif pédagogique est de montrer l'importance de séparer la logique métier du traitement des erreurs et d'améliorer la robustesse et la lisibilité des réponses de l'API.

9) DTO et mappers

Chaque entité exposée dispose d'un DTO dédié (ProductDTO, InventoryDTO, SalesOrderDTO, etc.).

Les DTO sont validés avec @Valid et @NotNull selon les contraintes fonctionnelles.

Aucune logique métier n'est incluse dans les DTO.

Les conversions entre entités et DTO sont gérées par des mappers MapStruct contenant les méthodes toDto() et toEntity().

10) Tests unitaires

Les tests unitaires doivent couvrir :

L'interdiction du stock négatif.

La réservation et la libération correcte des quantités.

Les transitions de statut d'une commande.

La création de backorders en cas de stock partiel.

Le respect du cut-off horaire.

La planification d'expédition et le respect des capacités de créneaux.

Le déclenchement des exceptions métier (StockUnavailableException, BusinessException, etc.).

La validation des DTO (erreurs de champs et messages d'erreur).

Les tests de mappers MapStruct.

Les tests utilisent JUnit 5 et Mockito et s'exécutent via la commande mvn test.

11) User stories principales

1) Epic CLIENT — Passer commande et suivre la livraison

US1 — Créer un compte client et compléter le profil
En tant que client, je veux créer mon compte et renseigner mon nom afin de

pouvoir passer des commandes.
Critères d’acceptation :

- Given une adresse email valide et un nom, When je soumets le formulaire, Then mon compte client est créé et actif.

- Given un email déjà utilisé, When je soumets le formulaire, Then je reçois un message d’erreur explicite.

US2 — Rechercher un produit par SKU
En tant que client, je veux rechercher un produit via son SKU pour vérifier sa disponibilité.
Critères d’acceptation :

- Given un SKU existant, When je consulte le produit, Then je vois le nom, la catégorie et l’état actif.

- Given un SKU inactif, When je consulte le produit, Then je vois que le produit est indisponible à la vente.

US3 — Créer une commande client (SalesOrder)
En tant que client, je veux créer une commande contenant une ou plusieurs lignes de produits avec quantité.
Critères d’acceptation :

- Given des produits actifs et un entrepôt source sélectionné, When je crée la commande, Then le statut initial est CREATED et les lignes sont enregistrées.

- Given un produit inexistant ou inactif, When je tente d’ajouter une ligne, Then l’ajout est refusé avec un message clair.

US4 — Réserver le stock de ma commande
En tant que client, je veux que le système réserve le stock avant expédition pour garantir l’envoi.
Critères d’acceptation :

- Given une commande en statut CREATED, When je demande la réservation, Then la réservation réussit si la disponibilité (qtyOnHand -

qtyReserved) de l’entrepôt source couvre toutes les lignes et le statut passe à RESERVED.

- Given une disponibilité partielle, When je réserve, Then le système réserve ce qu’il peut et crée un backorder pour le reste (ou refuse selon paramètre), avec message explicite.

US5 — Suivre mon expédition
En tant que client, je veux voir le numéro de suivi et le statut d’expédition (PLANNED, IN_TRANSIT, DELIVERED).
Critères d’acceptation :

- Given une commande RESERVED, When l’expédition est créée, Then je vois le trackingNumber et le statut PLANNED.

- Given un colis en cours, When le transporteur met à jour, Then je vois IN_TRANSIT puis DELIVERED à la livraison.

Cas limites CLIENT :

- Réservation expirée (TTL) non confirmée sous 24h : la réservation est libérée, la commande revient à CREATED.

- Commande après 15h (cut-off) : planification au jour ouvré suivant, information visible sur la commande.

---

2) Epic WAREHOUSE_MANAGER — Gérer stocks, réservations et expéditions

US6 — Enregistrer un INBOUND (réception)
En tant que gestionnaire d’entrepôt, je veux enregistrer une entrée de stock pour un produit dans un entrepôt.
Critères d’acceptation :

- Given un produit et un entrepôt valides, When j’enregistre un INBOUND, Then qtyOnHand augmente et un InventoryMovement de

type INBOUND est historisé avec occurredAt et referenceDoc.

- Given une quantité négative, When j’essaie d’enregistrer, Then l’opération est refusée.

US7 — Enregistrer un OUTBOUND (sortie)
En tant que gestionnaire d’entrepôt, je veux enregistrer une sortie de stock suite à une expédition.
Critères d’acceptation :

- Given un stock disponible suffisant (available ≥ quantité), When j’enregistre un OUTBOUND, Then qtyOnHand diminue et le mouvement est historisé.

- Given un stock insuffisant, When j’enregistre, Then l’opération est refusée (zéro stock négatif).

US8 — Ajuster le stock (ADJUSTMENT)
En tant que gestionnaire d’entrepôt, je veux corriger un écart inventaire via un ajustement.
Critères d’acceptation :

- Given un ajustement négatif, When j’applique l’ajustement, Then qtyOnHand ne doit jamais tomber sous qtyReserved ; sinon, l’opération est refusée.

- Given un ajustement, When il est validé, Then un InventoryMovement ADJUSTMENT est historisé.

US9 — Réserver la commande (SalesOrder → RESERVED)
En tant que gestionnaire d’entrepôt, je veux réserver le stock d’une commande créée.
Critères d’acceptation :

- Given une commande CREATED, When je lance la réservation, Then qtyReserved augmente pour chaque ligne et la commande passe à RESERVED si toutes les lignes sont couvertes.

- Given une couverture partielle, When je tente la réservation, Then j'obtiens soit un backorder enregistré, soit un refus selon la règle configurée.

US10 — Créer l'expédition (Shipment)
En tant que gestionnaire d'entrepôt, je veux créer une expédition pour une commande RESERVED et planifier son départ.
Critères d'acceptation :

- Given une commande RESERVED, When je crée le Shipment, Then le statut Shipment est PLANNED et la commande reste RESERVED.

- Given l'heure après le cut-off, When je planifie, Then la date de départ est automatiquement positionnée au jour ouvré suivant.

US11 — Marquer la commande SHIPPED puis DELIVERED
En tant que gestionnaire d'entrepôt, je veux marquer la commande comme expédiée puis livrée.
Critères d'acceptation :

- Given une commande RESERVED, When l'expédition démarre, Then le statut de la commande passe à SHIPPED, les OUTBOUND correspondants sont historisés, et qtyReserved diminue d'autant.

- Given un avis de livraison, When je reçois la preuve, Then je passe la commande à DELIVERED et le Shipment à DELIVERED.

Cas limites WH :

- Conflit de réservation concurrente : si deux réservations simultanées provoquent un disponible négatif, l'une échoue et un message clair est renvoyé.

- Surcapacité créneau : si le slot d'expédition atteint son plafond, la création de Shipment force une replanification automatique vers le prochain slot.

---

3) Epic ADMIN — Gouvernance catalogue, entrepôts, achats

US12 — Gérer les produits (CRUD)
En tant qu'admin, je veux créer, modifier, activer ou désactiver un produit.
Critères d'acceptation :

- Given un SKU unique, When je crée le produit, Then il est actif par défaut.

- Given un produit lié à des commandes, When je le désactive, Then il n'est plus sélectionnable pour de nouvelles commandes mais reste consultable.

US13 — Gérer les entrepôts (CRUD)
En tant qu'admin, je veux créer et paramétrer les entrepôts (codes uniques).
Critères d'acceptation :

- Given un code unique, When je crée l'entrepôt, Then il devient disponible pour les mouvements et les commandes.

- Given un entrepôt sans mouvements, When je le supprime, Then l'opération réussit ; sinon, une désactivation est requise.

US14 — Créer un Purchase Order et réceptionner (PO → RECEIVED)
En tant qu'admin, je veux émettre un PO à un fournisseur et réceptionner partiellement ou totalement.
Critères d'acceptation :

- Given un PO APPROVED, When je réceptionne partiellement, Then plusieurs INBOUND s'enregistrent et le solde reste en attente.

- Given la dernière réception, When le total des lignes est atteint, Then le PO passe à RECEIVED.

US15 — Annuler une commande (CANCELED)
En tant qu'admin, je veux annuler une commande avant expédition pour

libérer les réservations.
Critères d’acceptation :

- Given une commande CREATED, When je l’annule, Then elle passe à CANCELED et aucun mouvement stock n’est affecté.

- Given une commande RESERVED, When je l’annule, Then les qtyReserved sont libérées avant passage à CANCELED.

- Given une commande SHIPPED, When j’essaie d’annuler, Then l’opération est refusée (processus de retour à utiliser).

Cas limites ADMIN :

- Suppression d’un produit ou entrepôt référencé : refus, proposer désactivation.

- PO avec fournisseur inconnu : refus avec message explicite.

4) Stories “explicatives” du diagramme (mapping concept → action)

Story A — Inventaire et mouvements
Le diagramme montre Inventory (qtyOnHand, qtyReserved) rattaché à Product et Warehouse, et InventoryMovement pour chaque opération.
Interprétation : chaque INBOUND, OUTBOUND ou ADJUSTMENT crée une ligne de mouvement et met à jour les quantités de l’inventaire. L’historique est traçable, le stock disponible est toujours qtyOnHand - qtyReserved.

Story B — Commande client → réservation → expédition
SalesOrder (CREATED → RESERVED → SHIPPED → DELIVERED) est lié à SalesOrderLine (produit, quantité) et à un Warehouse source.
Interprétation : la réservation augmente qtyReserved, l’expédition déclenche OUTBOUND et libère les réserves, la livraison ferme le flux. Shipment porte le tracking et le statut d’acheminement.

Story C — Achats et réception
 PurchaseOrder (APPROVED → RECEIVED) porte des PurchaseOrderLine. La réception génère des INBOUND et augmente qtyOnHand dans l'entrepôt cible.
 Interprétation : le PO synchronise l'approvisionnement, chaque réception partielle est traçable, le statut passe à RECEIVED quand tout est livré.

Story D — Rôles et gouvernance
 User et Role montrent l'authz logique (même si tu n'actives pas la sécurité). ADMIN gouverne catalogue et entrepôts, WAREHOUSE_MANAGER pilote stock et expédition, CLIENT initie la demande.
 Interprétation : la séparation des rôles correspond directement aux responsabilités décrites dans les stories.

Diagramme de classe: