Audit V1:

Cet audit de la première version (V1) du système de gestion de stock a permis de détecter plusieurs problèmes liés à la sécurité, la validation des données et la logique métier. La version 2 (V2) propose des solutions pour corriger ces failles et améliorer la robustesse du système.

1. Failles de sécurité

- Injection SQL :

Le code utilise des requêtes SQL construites par concaténation de chaînes de caractères. Cela expose l'application à des risques d'injection SQL, car un attaquant pourrait manipuler les entrées pour exécuter des requêtes non autorisées.

Exemple vulnérable :

const query = `INSERT INTO produits (reference, nom, prix_unitaire, quantite, id_categorie, id_fournisseur) VALUES ('${reference}', '${nom}', ${prix_unitaire}, ${quantite}, ${id_categorie}, ${id_fournisseur})`;

Dans cet exemple, l'attaquant peut facilement effectuer une injection SQL en changeant l'url : **http://localhost:3000/clients/1 OR 1=1.**
En faisant ceci, tous les clients de la base de données sont retournées, l'attaquant dispose donc de toutes données sensibles des clients qui sont contenus dans la table clients.

Solution:

Il faudra donc utiliser des requêtes paramétrées pour éviter l'injection SQL.

- Manque de validation des données :

Les champs comme **quantite**, **prix_unitaire**, ou **id_categorie** ne sont pas validés avant d'être insérés ou mis à jour dans la base de données. Cela peut entraîner des valeurs incorrectes (ex. quantités négatives, prix non valides, etc.).

Solution : Ajouter des validations côté serveur pour vérifier les types de données, les valeurs minimales/maximales, et les contraintes métier.
Absence d'authentification :

L'API ne comporte aucune authentification, ce qui signifie que n'importe qui ayant accès au réseau interne peut interagir avec l'API.

Solution:
Mettre en place une authentification de base (par exemple, avec des tokens JWT) pour sécuriser l'accès à l'API.