Rapport de Recommandations

# Introduction

Suite à l'analyse concernant votre code, nous avons identifié plusieurs problèmes liés à la maintenabilité. Ce rapport synthétise ces problèmes et propose des recommandations pour les résoudre.

# Résumé des Problèmes

Les problèmes identifiés concernent principalement la maintenabilité du code. Voici un résumé :

| Impact | Nature | Nombre de Problèmes |
| --- | --- | --- |
| Moyen | Maintenabilité | 3 |
| Haut | Maintenabilité | 4 |
| Faible | Maintenabilité | 2 |

# Problèmes Détails et Recommandations

## Risque : Réécrire la valeur de la variable ailleurs dans le projet.

Occurrences : 3

Proposition d'amélioration : Les variables doivent être déclarées avec 'const' comme elles ne seront pas modifiées.

## Risque : Difficulté de compréhension du code par de nouveaux développeurs sur le projet.

Occurrences : 1

Proposition d'amélioration : Décomposer la structure du code en réalisant plusieurs fonctions, appelées depuis la fonction callback. Chaque fonction sera chargé de réaliser une tâche donnée. Par exemple, une fonction pourrait se charger de calculer les points de fidélité, une autre de modifier le statut de la commande...

## Risque : Difficulté de compréhension du code par les développeurs + perte de performances

Occurrences : 1

Proposition d'amélioration : Retirer la déclaration de la variable.

## Risque : Difficulté de compréhension du code par les développeurs + variable inaccessible en dehors de son scope

Occurrences : 1

Proposition d'amélioration : Déclarer la variable 'loyaltyPoints' avant les conditionnelles et lui assigner une valeur dans les conditionnelles.

## Risque : Les programmeurs ne doivent pas commenter le code, car cela alourdit les programmes et en réduit la lisibilité.

Occurrences : 1

Proposition d'amélioration : Le code inutilisé doit être supprimé et peut être récupéré dans l'historique du contrôle de la source si nécessaire.

## Risque : Oubli de réaliser la tâche (ou de corriger un bug) et dégradation de la qualité du code si les commentaires 'TODO' s'accumulent.

Occurrences : 1

Proposition d'amélioration : Ne pas push une fonctionnalité tant qu'elle n'aura pas été totalement réalisée.

## Risque : Perte de perfomances

Occurrences : 1

Proposition d'amélioration : Remplacer la boucle for par une boucle for of pour améliorer les performances et la compréhension du code.

# Recommandations Générales

## Risque : Exploitation d'éventuelles failles de sécurité sur le framework utilisé.

Occurrences : 1

Proposition d'amélioration : Désactriver l'entête 'x-powered-by' dans la requête HTTP.

## Risque : Injection SQL

Occurrences : 3

Proposition d'amélioration : Utiliser des requêtes préparées.

## Risque : Récupération des identifiants de connexion à la BDD par une personne malveillante.

Occurrences : 4

Proposition d'amélioration : Placer les données sensibles dans des variables d'environnement et ne pas envoyer les fichiers

# Recommandations Générales

- Nettoyez le code : Supprimez le code inutile ou commenté. Utilisez un système de contrôle de version pour l'historique.

- Améliorez la lisibilité : Décomposez les gros blocs de code en fonctions ou méthodes plus petites.

- Utilisez des variables appropriées : Utilisez 'const' et 'let' judicieusement. Évitez 'var'.

- Gestion des tâches : Évitez de laisser des commentaires 'TODO' dans le code.

- Optimisation : Surveillez les boucles et les structures pour l'optimisation.

- Variables d'environnement : Stockez les informations sensibles dans des variables d'environnement.

# Problèmes de Sécurité Détails et Recommandations

## Risque : Injection SQL

Emplacement : api/routes/auth.js(l29,30)

Description : Des requêtes SQL sont directement placées dans le code.

# Problèmes de Sécurité Détails et Recommandations

## Risque : Injection SQL

Emplacement : api/routes/auth.js(l29,30)

Description : Des requêtes SQL sont directement placées dans le code.

Proposition d'amélioration : Utiliser des requêtes préparées.

## Risque : Injection SQL

Emplacement : api/routes/orders.js(l53,55)

Description : Des requêtes SQL sont directement placées dans le code.

Proposition d'amélioration : Utiliser des requêtes préparées.

## Risque : Injection SQL

Emplacement : api/routes/pizzas.js(l43,45)

Description : Des requêtes SQL sont directement placées dans le code.

Proposition d'amélioration : Utiliser des requêtes préparées.

## Risque : Récupération des identifiants de connexion à la BDD par une personne malveillante.

Emplacement : api/app.js(l18)

Description : Des identifiants d'accès à la base de données sont écrits en dur dans le code source.

Proposition d'amélioration : Placer les données sensibles dans des variables d'environnement et ne pas envoyer les fichiers

## Risque : Récupération des identifiants de connexion à la BDD par une personne malveillante.

Emplacement : api/routes/auth.js(l17)

Description : Des identifiants d'accès à la base de données sont écrits en dur dans le code source.

Proposition d'amélioration : Placer les données sensibles dans des variables d'environnement et ne pas envoyer les fichiers

## Risque : Récupération des identifiants de connexion à la BDD par une personne malveillante.

Emplacement : api/routes/orders.js(l15)

Description : Des identifiants d'accès à la base de données sont écrits en dur dans le code source.

Proposition d'amélioration : Placer les données sensibles dans des variables d'environnement et ne pas envoyer les fichiers

## Risque : Récupération des identifiants de connexion à la BDD par une personne malveillante.

Emplacement : api/routes/pizzas.js(l14)

Description : Des identifiants d'accès à la base de données sont écrits en dur dans le code source.

Proposition d'amélioration : Placer les données sensibles dans des variables d'environnement et ne pas envoyer les fichiers

## Risque : Exploitation d'éventuelles failles de sécurité sur le framework utilisé.

Emplacement : api/app.js(l7)

Description : La version d'Expres utilisée est implicitement donnée dans le code.

Proposition d'amélioration : Désactriver l'entête 'x-powered-by' dans la requête HTTP.

# Notes Supplémentaires

1. La route /orders ne devrait pas être accessible publiquement. Il est essentiel de s'assurer que seuls les utilisateurs autorisés peuvent accéder à cette route pour garantir la sécurité des données.

2. Lors de la vérification de l'égalité en JavaScript, il est recommandé d'utiliser '===' au lieu de '=='. Cela garantit une comparaison à la fois de la valeur et du type, évitant ainsi des résultats inattendus, comme par exemple : "0" == 0 renvoyant true.

## Emplacement : Table pizzas

Nature : Données dupliquées

Risque : Incohérence des données

Description : Pizza 'Margherita' a des entrées dupliquées.

Proposition d'amélioration : Supprimer les doublons et s'assurer que chaque pizza est unique (s.

## Emplacement : Table orders\_pizzas

Nature : Quantité nulle

Risque : Incohérence des données

Description : Commande ID 4 a une quantité de pizza 0.

Proposition d'amélioration : Supprimer ou corriger les entrées avec une quantité de pizza de 0.

## Emplacement : Table pizzas (Lines: [51, 61])

Nature : Prix variables

Risque : Incohérence des données

Description : Pizza 'Margherita' à des prix différents.

Proposition d'amélioration : Standardiser le prix pour la pizza 'Margherita'.

## Emplacement : Table orders\_pizzas - Champ quantity

Nature : Conception

Risque : Incohérence des données

Description : Le champ quantity permet des valeurs non positives (0 ou négatives).

Proposition d'amélioration : Modifier la définition du champ pour n'accepter que des valeurs entières positives.

## Emplacement : Docker Compose - Service: db

Nature : Sécurité

Risque : Exposition de données sensibles

Description : Mots de passe en clair

Proposition d'amélioration : Utiliser des mécanismes de gestion des secrets ou des fichiers .env sécurisés.

## Emplacement : Docker Compose - Service: db, adminer

Nature : Sécurité

Risque : Accès non autorisé

Description : Exposition des ports

Proposition d'amélioration : Exposer uniquement les ports nécessaires et utiliser des règles de pare-feu pour restreindre l'accès.

## Emplacement : Docker Compose - Tous les services

Nature : Maintenabilité

Risque : Changements inattendus lors des reconstructions

Description : Image non versionnée

Proposition d'amélioration : Spécifier des versions exactes pour les images Docker.

## Emplacement : Docker Compose - Service: api

Nature : Mauvaises pratiques

Risque : Dépendance sur une fonctionnalité obsolète

Description : Utilisation de links

Proposition d'amélioration : Utiliser les réseaux Docker à la place de 'links'.

## Emplacement : Docker Compose - Service: api

Nature : Performance

Risque : Performance réduite lors du développement

Description : Un volume pour l'api

Proposition d'amélioration : Copier le code dans l'image lors de la construction pour la production.

## Emplacement : Docker Compose - Tous les services

Nature : Lisibilité

Risque : Difficulté à identifier les conteneurs

Description : Nom des conteneurs non standard

Proposition d'amélioration : Utiliser des noms de conteneurs descriptifs et cohérents.

## Emplacement : Docker Compose

Nature : Maintenabilité

Risque : Difficulté à comprendre la configuration

Description : Absence de commentaires

Proposition d'amélioration : Ajouter des commentaires pour expliquer la configuration et les choix.

## Emplacement : Docker Compose - Tous les services

Nature : Performance

Risque : Utilisation de ressources inutiles

Description : Images non optimisées

Proposition d'amélioration : Utiliser des variantes 'slim' ou 'alpine' des images.

## Emplacement : Docker Compose - Service: db

Nature : Sécurité

Risque : Attaques par force brute

Description : Mots de passe trop simples

Proposition d'amélioration : Utiliser des mots de passe complexes combinant lettres, chiffres et caractères spéciaux.

## Emplacement : Docker Compose - Service: db

Nature : Sécurité

Risque : Manque de rigueur

Description : Même mot de passe pour plusieurs comptes/utilisateurs

Proposition d'amélioration : Utiliser des mots de passe uniques pour chaque compte ou utilisateur.

# Journalisation des Erreurs

## Emplacement : Journalisation de l'application

Nature : Sécurité et Maintenabilité

Risque : Exposition d'informations sensibles et suivi inefficace des erreurs

Description : Les erreurs sont actuellement journalisées directement dans la console. Cela présente des risques potentiels en production et peut compliquer le suivi des problèmes.

Proposition d'amélioration : Envisager une solution de journalisation robuste pour stocker, analyser et alerter sur les erreurs de manière sécurisée.

# Utilisation de Nodemon en Production

## Emplacement : Configuration

Nature : Stabilité, Sécurité, Performance

Risque : Utilisation de Nodemon en production

Description : Nodemon, conçu pour le développement, offre un redémarrage automatique du serveur à chaque modification de fichier. Cela présente plusieurs risques en production.

1. Assurer la stabilité : Éviter les redémarrages automatiques qui peuvent entraîner des interruptions de service et des erreurs de redémarrage non traitées.  
 2. Préserver la sécurité : Prévenir l'exposition d'informations sensibles lors des plantages ou redémarrages inattendus.  
 3. Optimiser la performance : Éviter les redémarrages inutiles qui peuvent surcharger le serveur et affecter les performances.

Proposition d'amélioration :

une fois en production remplacer Nodemon par Node .

# Stockage des Mot de passe en MD5

Emplacement : Fichiers de l'application  
Nature : Sécurité  
Risque : Collision des mot de passe   
Description : Il a été observé que certains mots de passe sont stockés sous forme de hash MD5 dans différents fichiers de l'application. Le MD5 est considéré comme cryptographiquement vulnérable.  
Proposition d'amélioration : Pour renforcer la sécurité, il est recommandé de ne pas utiliser MD5 pour le hachage des mots de passe. À la place, utilisez un algorithme de hachage moderne et sécurisé, comme bcrypt.

# Stockage des Mot de passe en clair

Emplacement : Fichiers de l'application

Nature : Sécurité

Risque : Faille de sécurité / risque si partage du code sources

Description : les mots de passe sont stockés en clair et en dur dans les fichiers de l'application. De plus, ces mots de passe sont utilisés pour établir des connexions à des bases de données, ce qui est particulièrement risqué.

Proposition d'amélioration : Utilisez des variables d'environnement ou des solutions de gestion des mdp pour gérer et accéder à ces informations.