**📝 Compte Rendu des résolutions de problèmes dans le Projet E-commerce API**

**📌 Introduction**

Le projet **E-commerce API** a été une véritable aventure technique, jalonnée de défis, de résolutions de conflits et d’optimisations. Comme un chef d’orchestre guidant un ensemble musical, nous avons dirigé chaque instrument (Docker, Git, Node.js, MongoDB) pour qu’ils jouent en parfaite harmonie.

**🚀 1. Mise en Place du Projet**

**📂 Clonage et Installation**

Nous avons commencé par **cloner le projet depuis GitHub**, comme si nous avions récupéré les plans d’un chantier de construction. Ensuite, nous avons mis en place l’environnement avec **Docker et Node.js**, qui ont servi de fondations solides pour bâtir notre API.

📌 **Commandes exécutées :**

git clone https://github.com/Meisseu/ecommerce-api.git

cd ecommerce-api

Ensuite, nous avons lancé l’API avec Docker, en créant une **"autoroute de communication"** entre nos services :

bash run.sh

Mais comme toute construction, quelques imprévus sont apparus… 🏗️

**🔍 2. Dépannage et Résolution de Problèmes**

**🔹 Problème de Connexion MongoDB**

MongoDB refusait de se connecter, comme un maçon qui avait oublié son ciment. Après analyse, nous avons compris que la version installée nécessitait un processeur **avec support AVX**, ce que notre machine ne possédait pas.

💡 **Solution** : Installer MongoDB 4.4 à la place de 5.0+ :

docker stop db-container

docker rm db-container

docker run -d --name db-container --network backend-test-network --network-alias db-container mongo:4.4

**🔹 Problème avec run.sh et test.sh**

Nos scripts ne s’exécutaient pas correctement, comme un GPS mal paramétré qui nous envoie dans une impasse. Nous avons dû **modifier les permissions et adapter les commandes pour Windows**.

💡 **Correction des permissions :**

chmod +x run.sh test.sh

bash test.sh

Résultat : nos routes API ont été testées automatiquement ! 🚦

**🔹 Conflits Git (Le Duel des Versions)**

À force de modifier des fichiers en local, **Git a rencontré des conflits** entre notre code et celui sur GitHub. C’était comme si deux architectes avaient dessiné des plans différents pour la même maison. Nous avons dû choisir quelles parties conserver.

💡 **Vérification des conflits :**

git status

💡 **Résolution manuelle des conflits** (dans run.sh et server.js) en supprimant les sections conflictuelles et en fusionnant les versions correctes.

💡 **Finalisation du rebase :**

git add run.sh server.js

git rebase --continue

git push origin main

Après cette correction, nous avons pu pousser notre code à GitHub, comme si nous avions finalement validé les plans de construction. 🏠✅

**🛠 3. Test Automatisé de l’API**

Nous avons ensuite testé notre API grâce au script test.sh, qui a simulé **la gestion complète d’une boutique** :

1. Création d’un client
2. Ajout d’un produit
3. Passage d’une commande

💡 **Exécution du test automatique :**

bash test.sh

📌 **Résultat attendu :**

{

"message": "Order created successfully",

"order": {

"\_id": "65a42b1fcf6a1c3d9b999999",

"customer": "65a42b1fcf6a1c3d9b123456",

"products": ["65a42b1fcf6a1c3d9b654321"]

}

}

Comme un robot effectuant un contrôle qualité sur une chaîne de production, notre test a confirmé que l’API fonctionnait correctement !

**🎯 4. Finalisation et Nettoyage**

Après avoir validé que tout fonctionnait bien, nous avons fait **le ménage** dans notre environnement en supprimant les anciens fichiers clonés et les conteneurs inutiles.

💡 **Suppression propre du projet cloné :**

rmdir /s /q ecommerce-api # Windows

rm -rf ecommerce-api # Linux/macOS

💡 **Arrêt et suppression des conteneurs Docker :**

docker stop express-app db-container

docker rm express-app db-container

docker network rm backend-test-network

💡 **Synchronisation finale avec GitHub :**

git pull origin main --rebase

git push origin main

Tout était propre, comme après un grand ménage de printemps. 🧹✨

**📌 Conclusion**

Nous avons **installé, corrigé, testé et synchronisé** notre API avec **succès**. 🎉

* **Docker et MongoDB ont été configurés** comme une usine fonctionnelle 🏭.
* **Les conflits Git ont été résolus**, évitant une guerre civile dans notre code ⚔️.
* **L’API a été testée et validée**, garantissant une expérience utilisateur fluide ✅.

**🔥 Prochaines Étapes**

1. **Optimisation du code** pour améliorer la gestion des erreurs.
2. **Ajout de la documentation avec Swagger** pour rendre l’API plus accessible.
3. **Mise en place d’un système d’authentification** pour sécuriser l’API.

Nous avons construit une API robuste et fonctionnelle, prête à évoluer ! 🚀

💡 **Moralité :**

"Le développement, c'est comme construire une maison : il faut des fondations solides, résoudre les imprévus et tester chaque porte avant d'y habiter !" 🏠💻