Documentation

1. Vue d'ensemble

L'objectif était ici de développer une application web full-stack permettant de gérer des tâches au sein de projets d'équipe. Mon but était de rendre l'expérience agréable, que la solution ait une bonne ergonomie tout en étant assez efficace côté serveur, dans la mesure de mes compétences.

Les fonctionnalités principales sont l'authentification / création de profil, la création et la gestion de projets, de tâches et l'assignation d'utilisateur, le tout sous la forme d'un tableau Kanban, à la manière de Trello, seul outil du genre que j'ai déjà utilisé.

2. Architecture

- Frontend : Vue 3, Axios pour les requêtes HTTP, Vue Router pour la navigation.
- Backend : FastAPI, JWT pour l'authentification, gestion des routes CRUD.
- Base de données : MySQL (initialement locale, maintenant dockerisée).
- Communication : frontend ↔ backend via API REST.

3. Choix des technologies

- **Vue 3** : Réactif, modulable, simple à prendre en main pour quelqu'un comme moi qui ne connaît pas React
- **FastAPI** : Rapide, scalable, structure simple et intégration native avec Pydantic pour la validation des données.
- **JWT** : Solution efficace et stateless pour l'authentification, plus sécurisée que de simples tokens stockés en localStorage.
- **Docker pour la base de données** : Portabilité, pas besoin d'installer MySQL/Workbench sur chaque machine. M'a permis d'avoir une première expérience pratique de Docker.
- **BCrypt** : Hashage sécurisé des mots de passe côté serveur.

Côté frontend, Vue s'est imposé par sa simplicité, sa réactivité et mon expérience limitée sur React. Côté backend, j'ai choisi FastApi pour m'accorder à la stack technique de Kanbios et retrouver ce framework que j'avais déjà utilisé, tout en profitant de sa structure simple, sa rapidité et sa propension à la scalabilité. Cela fut aussi l'occasion d'utiliser Pydantic qui a facilité les interactions avec la Database. Voulant prendre de bons reflexes par rapport à la sécurité, et en prenant en compte votre feedback lors de notre entretien, j'ai décidé d'utiliser JWT ainsi que BCrypt pour le hashage des mots de passe.

Résumé de la sécurité :

- JWT stocké côté client et transmis dans le header Authorization
- Routes critiques (création d'utilisateurs, projets, tâches) protégées via dépendances FastAPI.
- Hashage des mots de passe avec Bcrypt

Risques restants :

- Pas de rafraîchissement automatique du token : risque d'expiration
- Permissions sur certaines routes (édition/suppression de projets ou tâches) ne sont pas encore complètement restreintes aux propriétaires ou utilisateurs assignés.
- Dans certains contextes, possibilité de failles côté frontend si l'utilisateur modifie les requêtes via l'inspecteur de navigateur.

4. Améliorations futures

1. Implémenter un rafraîchissement automatique des JWT pour prolonger les sessions en toute sécurité.
2. Ajouter une gestion fine des permissions : seul le propriétaire ou les utilisateurs assignés peuvent modifier ou supprimer des projets/tâches.
3. Dockeriser l'ensemble du projet plutôt que seulement la DB, pour faciliter le déploiement et le partage.
4. Ajouter notifications en temps réel avec WebSockets.
5. Mettre en place tests automatisés pour le backend et le frontend afin d'assurer la stabilité du système.