Sujet examen CCP2 - Titre Pro Developpeur Web et Web Mobile

Projet : Gestion des missions et des candidatures

Contexte :

Vous êtes chargé de créer une plateforme permettant aux bénévoles de postuler à des missions proposées par des associations. Une fois qu'un bénévole postule, une association peut accepter ou refuser la candidature.

Optionnel Vous pouvez utiliser un outil de gestion de projet (Linear, Trello, Notion, GitHub Projects, un tableur) pour organiser votre travail.

- Découpez votre projet en tâches avant de commencer à coder.
- Déplacez vos tâches au fur et à mesure de l'avancement.

Fonctionnalités à développer :

1. **Authentification :**
   - Un utilisateur peut se connecter ou s'inscrire sur la plateforme.
   - Deux types d'utilisateurs : **Bénévoles** et **Associations**.
   - L'authentification doit se faire via un système de token (JWT).
2. **Gestion des missions :**
   - Une association peut créer, modifier, et supprimer des missions.
   - Les missions ont un titre, une description, une date, et sont associées à une **association**.
3. **Gestion des candidatures :**
   - Un bénévole peut **postuler à une mission**. À la suite de la candidature, le statut de la candidature est **"En attente"**.

- Une association peut **accepter** ou **refuser** la candidature d’un bénévole.
- Le statut de la candidature doit être **"Acceptée"** ou **"Refusée"** en fonction de l'action de l'association ou “en attente” par défaut.

4. **Affichage des missions :**
    - Un bénévole peut consulter la liste des missions disponibles et postuler à celles qui l'intéressent.
    - Une association peut consulter la liste des candidatures en attente pour chaque mission.

Modélisation de la base de données :

Avant de commencer à coder, vous devez **modéliser votre base de données** en fonction du choix de votre technologie (SQL ou NoSQL). Vous êtes libre de choisir votre solution mais vous devez **justifier votre choix** et montrer clairement la structure de votre base de données.

- Utilisateurs (Bénévoles et Associations)
- Missions
- Candidatures

N’oubliez pas de créer et de fournir un jeu de données qui me permettent de tester l’application.

Documentation :

1. **Endpoints API :**

    Documenter toutes les routes avec leurs fonctionnalités, méthodes HTTP utilisées, paramètres et exemples de requêtes/réponses.

2. **README File :**
    - Présentation du projet
    - Instructions pour installer et lancer l’application
    - Justification du choix technologique (SQL vs NoSQL)
    - Informations complémentaires pour l’utilisation

Conteneurisation avec Docker (Optionnel) :

- **Objectif :** Fournir un environnement facilement déployable.
- **Fichiers nécessaires :**
  - `Dockerfile` pour définir l'environnement de l'application.
  - `docker-compose.yml` pour orchestrer la base de données et l'application.
- Instructions pour la construction et l'exécution du conteneur.

Livrables attendus :

- **Code source complet** de l'application avec un README détaillant le choix technologique (SQL ou MongoDB), les instructions pour démarrer le projet, et les fonctionnalités implémentées via un lien Github
- **Schéma de la base de données**
- **Votre document de gestion des tâches** ou au moins une capture d'écran
- **Documentation d'API**
- **Jeu de données d'essai** pour tester l'application
- **Création et initialisation de la base de données**
- **Optionnel : Conteneurisation avec Docker**

Critères d'évaluation :

- Fonctionnalités respectées (authentification, gestion des missions et candidatures).
- Clarté de la **modélisation de la base de données**.
- **Code propre et bien structuré**, utilisation de bonnes pratiques.
- Justification du choix de la technologie (SQL vs NoSQL).
- **Jeu d'essai et création de la base de données.**
- **Optionnel :** Un Docker pour conteneuriser l'application.

- Gestion des erreurs et des permissions pour les utilisateurs.

Conseils de planification : A titre purement indicatif

1. **Etape 1 : Analyse et Modélisation**
    - **Objectif :** Analyse du sujet, modélisation de la base de données et choix de la technologie (SQL ou MongoDB).
    - Préparation du projet, structure des fichiers et justification du choix technologique.
    - Définition du jeu de données d’essai et initialisation de la base.
2. **Etape 2 : Routes API de base (Sans authentification) + Début des Tests**
    - **Objectif :** Implémentation des routes pour gérer les missions et les candidatures (sans authentification).
    - Développement des routes de création, modification, suppression de missions et gestion des candidatures.
    - **Tests au fur et à mesure** pour valider les fonctionnalités des routes API.
    - Début de la rédaction de la documentation API (endpoints).
3. **Etape 3 : Authentification, Gestion des Rôles et Tests**
    - **Objectif :** Implémentation de l'authentification des utilisateurs et gestion des rôles.
    - Ajout des fonctionnalités de connexion, inscription, et gestion des rôles (bénévoles vs associations).
    - **Tests continus** sur les fonctionnalités d'authentification et les routes protégées.
    - Documentation des permissions des utilisateurs.
4. **Etape 4 : Affichage des Missions, Finalisation et Documentation**
    - **Objectif :** Affichage des missions pour les bénévoles et gestion des candidatures pour les associations.

- Tester l'ensemble des fonctionnalités, en assurant que les routes fonctionnent correctement avec l'authentification.
- **Finalisation de la documentation API et du README.**
- Appliquer des **améliorations finales** (gestion des erreurs, validation des données, etc.).
- Optionnel : Mise en place de Docker et test du déploiement avec le conteneur.

Conseils supplémentaires :

- **Ignorer l’authentification et les rôles au départ** : Tout le monde peut ajouter et gérer les missions.
- **N'oubliez pas de bien tester** votre application avec Postman ou Thunderclient. N’oubliez pas de tester les cas d’erreur qui ne doivent pas fonctionner.
- Assurez-vous que chaque fonctionnalité est bien couverte et fonctionne avant l’envoi.
- Ne mettez en place Docker que si vous avez fini, c’est optionnel.
