MEAN : Application de Gestion de Portefeuille

Objectifs

1. Comprendre comment configurer une base de données MongoDB.
2. Apprendre à créer un serveur Express.js.
3. Implémenter l'authentification utilisateur avec JWT.
4. Créer et intégrer un frontend Angular 17.
5. Implémenter une fonctionnalité de gestion de portefeuille avec des opérations CRUD.

Prérequis

1. Node.js et npm installés
2. MongoDB installé et en cours d'exécution
3. Angular CLI installé (`npm install -g @angular/cli`)

Partie 1 : Configuration du Backend avec Express.js

Étape 1 : Initialiser le Projet

1. Ouvrez un terminal et créez un nouveau répertoire pour le projet.
2. Initialisez un nouveau projet Node.js.
3. Installez les dépendances requises (`express`, `mongoose`, `bcryptjs`, `jsonwebtoken`, `body-parser`, `cors`).

Question :

- Que fait la commande `npm init -y` ?

Étape 2 : Créer la Structure du Projet

1. Créez les répertoires nécessaires : `backend`, `backend/models` et `backend/routes`.
2. Créez un fichier d'entrée `backend/server.js`.

Étape 3 : Configurer la Connexion MongoDB

1. Ouvrez `backend/server.js`.
2. Configurez Express et la connexion à MongoDB avec Mongoose.
3. Ajoutez les middleware pour `body-parser` et `cors`.
4. Configurez un écouteur pour le serveur sur le port 3000.
5. Créez les gestionnaires de routes pour l'authentification et le portefeuille.

Questions :

- Quel est le rôle de `body-parser` dans une application Express ?
- Pourquoi utilise-t-on `cors` dans la configuration du serveur ?

Étape 4 : Créer le Modèle Utilisateur

1. Créez un fichier `backend/models/User.js`.
2. Définissez un schéma Mongoose pour le modèle Utilisateur avec des champs pour `username` et `password`.

Question :

- Quelle est la signification de `unique: true` dans le champ `username` du schéma ?

Étape 5 : Créer le Modèle Portefeuille

1. Créez un fichier `backend/models/Portfolio.js`.
2. Définissez un schéma Mongoose pour le modèle Portefeuille avec des champs pour `userId`, `asset`, `quantity`, et `value`.

Question :

- Comment la référence `userId` dans le modèle Portefeuille aide-t-elle à associer des actifs aux utilisateurs ?

Étape 6 : Créer les Routes d'Authentification

1. Créez un fichier `backend/routes/auth.js`.
2. Implémentez des routes pour l'inscription et la connexion des utilisateurs.
3. Utilisez `bcryptjs` pour hacher les mots de passe et `jsonwebtoken` pour créer des jetons JWT.

Questions :

- Pourquoi est-il important de hacher les mots de passe avant de les stocker dans la base de données ?
- Comment JWT aide-t-il à maintenir les sessions utilisateur ?

Étape 7 : Créer les Routes de Portefeuille

1. Créez un fichier `backend/routes/portfolio.js`.
2. Implémentez des routes pour ajouter des actifs et récupérer le portefeuille d'un utilisateur.
3. Ajoutez un middleware pour authentifier les requêtes à l'aide de JWT.

Questions :

- Quel est le rôle des middleware dans les applications Express ?
- Comment l'authentification JWT améliore-t-elle la sécurité de votre application ?

Partie 2 : Configuration du Frontend avec Angular 17

Étape 1 : Créer le Projet Angular

1. Utilisez Angular CLI pour créer un nouveau projet Angular nommé `frontend`.
2. Naviguez dans le répertoire du projet.

Étape 2 : Installer les Dépendances

1. Installez le package `@auth0/angular-jwt`.

Étape 3 : Créer le Service d'Authentification

1. Créez un service pour gérer l'inscription, la connexion et la déconnexion des utilisateurs.
2. Stockez les jetons JWT dans le stockage local.
3. Créez des méthodes pour vérifier si un utilisateur est authentifié et pour récupérer le jeton.

Question :

- Comment le stockage des jetons JWT dans le stockage local aide-t-il à maintenir les sessions utilisateur ?

Étape 4 : Créer le Service de Portefeuille

1. Créez un service pour gérer la récupération et l'ajout d'actifs au portefeuille.
2. Assurez-vous que les requêtes HTTP incluent le jeton JWT dans les en-têtes.

Question :

- Pourquoi est-il nécessaire d'inclure le jeton JWT dans les en-têtes des requêtes HTTP ?

Étape 5 : Configurer le Gardien d'Authentification

1. Créez un gardien pour protéger les routes nécessitant une authentification.
2. Redirigez les utilisateurs vers la page de connexion s'ils ne sont pas authentifiés.

Question :

- Comment un gardien d'authentification améliore-t-il la sécurité de votre application ?

Étape 6 : Créer le Composant de Connexion

1. Créez un composant pour la page de connexion.
2. Implémentez un formulaire pour que les utilisateurs saisissent leur nom d'utilisateur et leur mot de passe.
3. Appelez le service d'authentification pour connecter l'utilisateur.

Question :

- Pourquoi est-il important de gérer la connexion des utilisateurs sur le frontend ?

Étape 7 : Créer le Composant d'Inscription

1. Créez un composant pour la page d'inscription.
2. Implémentez un formulaire pour que les utilisateurs saisissent leur nom d'utilisateur et leur mot de passe.
3. Appelez le service d'authentification pour inscrire l'utilisateur.

Question :

- Quels sont les avantages d'avoir un composant d'inscription séparé ?

Étape 8 : Créer le Composant de Portefeuille

1. Créez un composant pour afficher et gérer le portefeuille.
2. Implémentez un formulaire pour ajouter de nouveaux actifs.
3. Récupérez et affichez les actifs du portefeuille depuis le backend.

Question :

- Comment pouvez-vous vous assurer que les données du portefeuille sont toujours à jour ?

Étape 9 : Configurer le Routage

1. Configurez les routes pour les composants de connexion, d'inscription et de portefeuille.
2. Protégez la route du portefeuille en utilisant le gardien d'authentification.

Question :

- Pourquoi le routage est-il important dans une application monopage (SPA) ?

Étape 10 : Configurer le Module Principal

1. Importez les modules nécessaires et déclarez les composants dans le module principal Angular.
2. Configurez le composant de démarrage pour l'application Angular.

Résumé

Vous avez maintenant une application de gestion de portefeuille de base. Le backend est un serveur Express.js avec MongoDB pour stocker les utilisateurs et les actifs du portefeuille, utilisant JWT pour l'authentification. Le frontend est une application Angular 17 qui permet aux utilisateurs de s'inscrire, de se connecter et de gérer leurs portefeuilles.