Création d'une API de gestion de voyages organisés

Créer une API pour une application de gestion de voyages organisés avec authentification JWT, gestion des rôles (admin / membre), inscriptions limitées et suivi des documents.

Contexte

L'agence TravelNow organise des voyages groupés avec un nombre de places limité.
Les administrateurs créent des voyages, gèrent les inscriptions, les paiements d'acompte, les documents requis et peuvent modifier les statuts des voyages.
Les membres peuvent consulter les voyages disponibles, s'inscrire s'il reste des places, payer un acompte et suivre leur dossier.

Fonctionnalités attendues

Côté admin

- CRUD des voyages
- Suivi des inscriptions
- Modification des statuts des voyages (ouvert / complet / annulé)
- Gestion des documents obligatoires (passeport, vaccins, etc.)
- Visualisation des paiements d'acompte

Côté membre

- Consultation des voyages disponibles
- Inscription à un voyage
- Paiement d'un acompte (simulation, pas de paiement réel)
- Gestion de son profil utilisateur
- Suivi de ses inscriptions et de l'avancement (paiement, validation documents)

Authentification

- JWT avec deux rôles : admin et membre
- Middlewares de vérification des rôles pour protéger les routes

Routes détaillées

Authentification			
POST	/api/auth/signup	Public	Inscription
POST	/api/auth/login	Public	Connexion
Voyages			
GET	/api/trips	Public	Lister les voyages disponibles
GET	/api/trips/:id	Public	Détails d'un voyage
POST	/api/trips	Admin	Créer un voyage
PUT	/api/trips/:id	Admin	Modifier un voyage
DELETE	/api/trips/:id	Admin	Supprimer un voyage
Inscriptions			
POST	/api/registrations/:tripId	Membre	S'inscrire à un voyage
GET	/api/registrations/me	Membre	Voir mes inscriptions
DELETE	/api/registrations/:id	Membre	Annuler mon inscription
GET	/api/registrations/trip/:id	Admin	Voir les inscriptions d'un voyage
Paiements			
POST	/api/payments/:registrationId	Membre	Simuler un paiement d'acompte
GET	/api/payments/me	Membre	Voir mes paiements
GET	/api/payments/trip/:id	Admin	Voir les paiements liés à un voyage
Documents requis			
POST	/api/documents	Admin	Ajouter un type de document requis
GET	/api/documents	Public	Voir les documents requis
POST	/api/documents/:registrationId	Membre	Soumettre un document
GET	/api/documents/me	Membre	Voir mes documents
GET	/api/documents/trip/:id	Admin	Voir les documents reçus pour un voyage
Utilisateurs			
GET	/api/user/profile	Membre	Voir mon profil
PUT	/api/user/profile	Membre	Modifier mon profil

Base de données

Vous pouvez choisir entre SQL ou NoSQL. Voici quelques notions à garder en tête :

- Un voyage peut avoir plusieurs inscriptions
- Une inscription appartient à un utilisateur
- Une inscription peut avoir plusieurs documents
- Un utilisateur peut avoir plusieurs inscriptions

Aucun modèle de données n'est spécifié, l'objectif est de se projeter dans le contexte pour définir les modèles les plus cohérents, libre à vous pour l'interprétation tant que les routes sont respectées.

Tests et validations

Vous devrez utiliser des services dans vos contrôleurs. Vous devrez également réaliser les tests unitaires de ces services (au minimum tester 2 routes par service, et un service par contrôleur)
Utilisez également express validator pour ajouter un middleware de validation sur vos routes (au minimum 15 validation, une validation est égal à une fonction utilisée, par exemple not null)

Frontend minimal

A minima développer le frontend côté membre avec les fonctionnalités suivantes :

- Liste des voyages avec possibilité d'inscription (/connexion)
- Dashboard utilisateur avec mes inscriptions
- Formulaire de paiement d'acompte (simulation)
- Upload de documents via formulaire

Pour les plus avancés, n'hésitez pas à pousser le frontend au maximum pour terminer le projet.
Le choix du framework frontend est le vôtre. Vous pouvez également utiliser la notion de EJS templates vue en cours.