ETUDE DE CAS

Création de l'API des articles

Nom et prénom du stagiaire (à compléter) :

Antoine Tartare

Les réponses doivent être répertoriées sur un compte GitHub ou GitLab. Assurez-vous que le lien envoyé soit public.

Veuillez noter votre lien GitHub ou GitLab :
https://github.com/Inaure/etudedecasiscod

1. Rédigez un diagramme UML de la base de données

Récupérez les sources sur le lien suivant : https://github.com/Oktogone/Nodejs-Approfondissement
Ces sources vous serviront tout au long de l'étude de cas

Aide :

Installation :

1. Démarrez MongoDB
2.

npm install
npm run dev

Lancez votre navigateur sur le port 3000

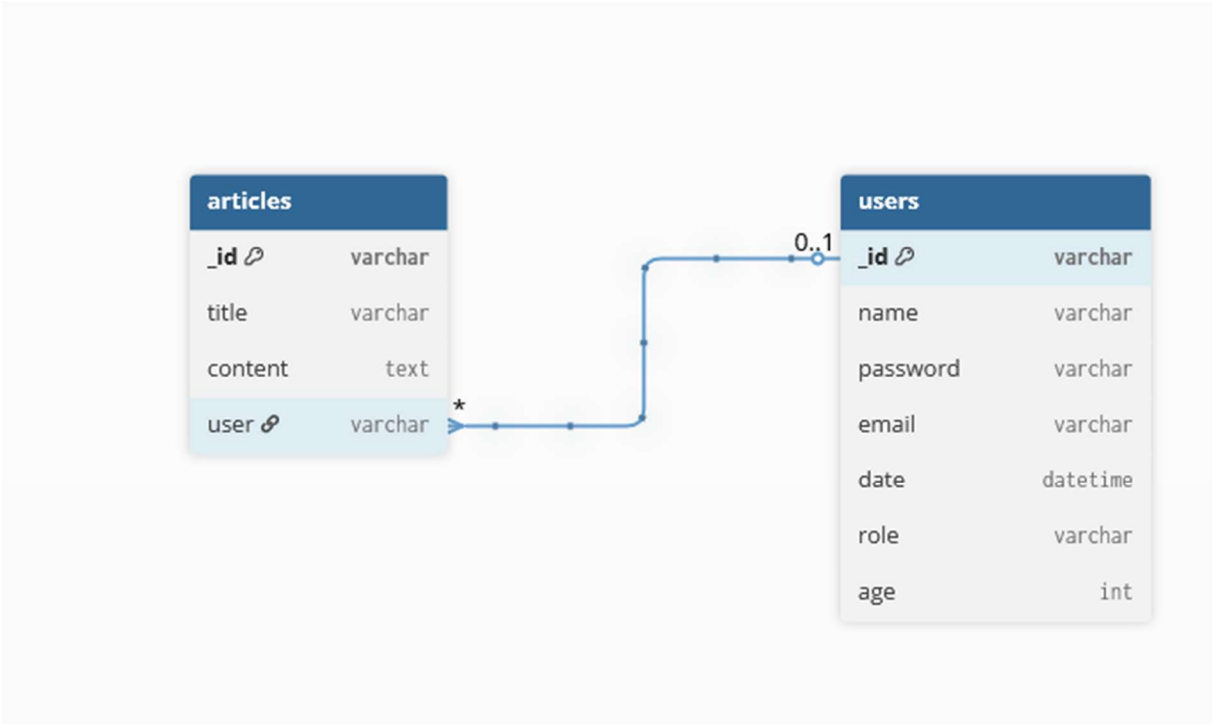
Si vous souhaitez lancer les tests : npm test

Retrouvez les schémas dans le dossier « api » de chaque ressource. A partir de ces schémas, dessinez le diagramme UML de la base de données

VISIPLUS
digital learning

Feuille de travail

ETUDE DE CAS



2. Ajouter l'énumération dans le schéma de article.schema.js avec deux valeurs : draft, published

Rédigez le code dans *api*/articles/articles.schema.js

Aide :

Documentation : https://mongoosejs.com/docs/schematypes.html#string-validators

3. Créer les endpoints de création, mise à jour et suppression d'un article. L'utilisateur doit être connecté pour effectuer la création.

1. Rédigez le code dans /*api*/articles/articles.service.js avec les 3 méthodes (création, mise à jour, suppression)
2. Rédigez le code dans *api/*articles/articles.controller.js avec les 3 méthodes correspondant au contrôleur
3. Rédigez les routes dans *api/*articles/articles.router.js. Pensez à l'ajouter dans votre application (server.js)
4. Pensez à ajouter le temps réel dans le contrôleur
5. Modifiez le middleware d'authentification afin de faire récupérer toutes les informations d'un utilisateur et les faire passer dans « req » (et pas seulement l'id de l'utilisateur)
6. Faites en sorte que la modification et la suppression (dans le contrôleur) s'effectuent seulement si l'utilisateur est « admin » (propriété « role » sur l'utilisateur).
7. Lors de la création, faites un enregistrement en utilisant l'id de l'utilisateur connecté

Aide :

Calquez vous sur la réalisation de l'API *api*/users réalisée durant cette formation

4. Créer le endpoint public pour afficher les articles d'un utilisateur. Le endpoint doit être sous la forme api/users/:userId/articles

Plus compliqué.
Attention, le endpoint doit être public donc pensez à ne pas la bloquer avec le middleware d'authentification sans le vouloir !
Le endpoint contient 1 paramètre : l'identifiant de l'utilisateur donc
1. Ajoutez le le code dans *api*/users/users.controller.js
2. Ecrivez la méthode dans le service (api/articles/articles.service.js) en utilisant la méthode populate() de Mongoose. Récupérer tous les articles d'un utilisateur, (ne pas afficher le mot de passe)

Aide :

N'hésitez pas à revoir l'utilisation des populations dans Mongoose (https://mongoosejs.com/docs/populate.html)

5. Créer les tests unitaires du point n°2 (création, mise à jour , suppression)

A partir d'un nouveau fichier dans le dossier « tests », nommez le « articles.spec.js », rédigez les 3 tests en utilisant « supertest ».
1. Vérifiez le code de la réponse de la requête
2. Ecrivez les mocks pour donner des réponses fictives (vous pouvez utilisez un module comme mockingoose, si vous le souhaitez)

Aide :

Calquez vous sur les tests réalisés sur la ressource « users »

6. Etablir une configuration de déploiement

Ouvrez le fichier « ecosystem.config.js », et ajoutez les besoins suivants :

- On doit avoir un fichier log en cas d'erreur s'enregistrant dans /logs/err.log
- Utilisation de la mémoire maximum: 200 Mo

Enfin, donnez la commande la commande PM2 pour lancer votre application avec 3 instances en parallèle

pm2 start ecosystem.config.js --env production

Aidez vous de la documentation de PM2 :
https://pm2.keymetrics.io/docs/usage/application-declaration/