Exercice : Gestion des suppressions en cascade avec Mongoose

Propriétaire	Ⓜ Marine
Étiquettes	

Exercice : Gestion des suppressions en cascade avec Mongoose

Contexte

Tu dois modéliser une base de données pour une application de blog. Chaque utilisateur peut publier plusieurs articles. Lorsque l'utilisateur est supprimé, tous ses articles doivent être supprimés automatiquement pour éviter les données orphelines.

Niveau Facile : Suppression en cascade sans middleware

1. **Objectif** : Supprime un utilisateur et tous ses articles associés **manuellement** dans une seule opération.
2. **Instructions** :
   - Crée deux modèles :
     - `User` (avec les champs : `name`, `email`).
     - `Article` (avec les champs : `title`, `content`, et une référence à l'utilisateur `userId`).
   - **Seed de la base de données** :
     - Crée une fonction de seed pour peupler la base de données avec des utilisateurs et des articles associés.
     - Utilise cette fonction pour générer des données fictives avant de procéder à la suppression en cascade.
     - Assure-toi que chaque utilisateur ait plusieurs articles.

- Implémente un bouton ou une méthode pour supprimer un utilisateur.
- Lorsqu'un utilisateur est supprimé, supprime **manuellement** tous ses articles avec une commande MongoDB.

1. **Quiz :**
   - Pourquoi la suppression en cascade est importante dans ce contexte ?
   - Quelles seraient les conséquences de ne pas gérer la cascade (par exemple, laisser des articles orphelins) ?

Niveau Intermédiaire : Suppression en cascade avec middleware `pre`

1. **Objectif** : Utilise un middleware `pre` dans le modèle `User` pour gérer automatiquement la suppression des articles lorsque l'utilisateur est supprimé.
2. **Instructions** :
   - Ajoute un middleware `pre('remove')` dans le schéma `User`.
   - Utilise ce middleware pour supprimer automatiquement tous les articles liés avant la suppression de l'utilisateur.
3. **Étapes pour tester** :

- Crée un utilisateur et plusieurs articles associés.
- Supprime l'utilisateur avec `user.remove()` et vérifie que les articles sont bien supprimés automatiquement.
- **Quiz :**
  - Pourquoi utilise-t-on `pre('remove')` au lieu de `post('remove')` ici ?
  - Que se passe-t-il si une erreur survient dans le middleware `pre` ?

Quiz :

- Pourquoi utiliser un middleware `post` pour journaliser les suppressions plutôt que `pre` ?
- Que ferais-tu si la journalisation échoue ? Le processus de suppression doit-il être annulé ?

Bonus : Gestion des relations multiples et suppressions en cascade

Contexte

Tu dois modéliser une application de blog dans laquelle :

- Un **utilisateur** peut écrire plusieurs **articles**.
- Un **article** peut avoir plusieurs **commentaires** et des **likes**.
- Lorsqu'un utilisateur est supprimé :
    1. Tous ses articles doivent être supprimés.
    2. Tous les commentaires associés à ses articles doivent être supprimés.
    3. Tous les likes associés à ses articles doivent également être supprimés.

Nous allons gérer cela avec des **middlewares Mongoose**, en utilisant `pre` et des transactions MongoDB pour assurer une suppression atomique.