PIIQUANTE

Application API Requirements

Contexte du projet

Piiquante se dédie à la création de sauces épicées dont les recettes sont gardées secrètes. Pour tirer parti de son succès et générer davantage de buzz, l'entreprise souhaite créer une application web dans laquelle les utilisateurs peuvent ajouter leurs sauces préférées et liker ou disliker les sauces ajoutées par les autres.

Spécifications de l'API

	Point d'accès	Request body (le cas échéant)	Type de réponse attendue	Fonction
POST	/api/auth/signup	{ email: string, password: string }	{ message: string }	Hachage du mot de passe de l'utilisateur, ajout de l'utilisateur à la base de données.
POST	/api/auth/login	{ email: string, password: string }	{ userId: string, token: string }	Vérification des informations d'identification de l'utilisateur, renvoie l _id de l'utilisateur depuis la base de données et un token web JSON signé (contenant également l'_id de l'utilisateur).
GET	/api/sauces	-	Array of sauces	Renvoie un tableau de toutes les sauces de la base de données.
GET	/api/sauces/:id	-	Single sauce	Renvoie la sauce avec l'_id fourni.
POST	/api/sauces	{ sauce: String, image: File }	{ message: String } **Verb**	Capture et enregistre l'image, analyse la sauce transformée en chaîne de caractères et l'enregistre dans la base de données en définissant correctement son imageUrl. Initialise les likes et dislikes de la sauce à 0 et les usersLiked et usersDisliked avec des tableaux vides. Remarquez que le corps de la demande initiale est vide ; lorsque multer est ajouté, il renvoie une chaîne pour le corps de la demande en fonction des données soumises avec le fichier.

PUT	/api/sauces/:id	EITHER Sauce as JSON OR { sauce: String, image: File }	{ message: String }	Met à jour la sauce avec l'_id fourni. Si une image est téléchargée, elle est capturée et l'imageUrl de la sauce est mise à jour. Si aucun fichier n'est fourni, les informations sur la sauce se trouvent directement dans le corps de la requête (req.body.name, req.body.heat, etc.). Si un fichier est fourni, la sauce transformée en chaîne de caractères se trouve dans req.body.sauce. Notez que le corps de la demande initiale est vide ; lorsque multer est ajouté, il renvoie une chaîne du corps de la demande basée sur les données soumises avec le fichier.
DELETE	/api/sauces/:id	-	{ message: String }	Supprime la sauce avec l'_id fourni.
POST	/api/sauces/:id/like	{ userId: String, like: Number }	{ message: String }	Définit le statut « Like » pour l' userId fourni. Si like = 1, l'utilisateur aime (= like) la sauce. Si like = 0, l'utilisateur annule son like ou son dislike. Si like = -1, l'utilisateur n'aime pas (= dislike) la sauce. L'ID de l'utilisateur doit être ajouté ou retiré du tableau approprié. Cela permet de garder une trace de leurs préférences et les empêche de liker ou de ne pas disliker la même sauce plusieurs fois : un utilisateur ne peut avoir qu'une seule valeur pour chaque sauce. Le nombre total de « Like » et de « Dislike » est mis à jour à chaque nouvelle notation.

API Errors

Les erreurs éventuelles doivent être renvoyées telles qu'elles sont produites, sans modification ni ajout. Si nécessaire, utilisez une nouvelle Error().

API Routes

Toutes les routes sauce pour les sauces doivent disposer d'une autorisation (le token est envoyé par le front-end avec l'en-tête d'autorisation : « Bearer <token> »). Avant que l'utilisateur puisse apporter des modifications à la route sauce, le code doit vérifier si l'userId actuel correspond à l'userId de la sauce. Si l'userId ne correspond pas, renvoyer « 403: unauthorized request. » Cela permet de s'assurer que seul le propriétaire de la sauce peut apporter des modifications à celle-ci.

Data ModelsSauce

- **userId** : *String* — l'identifiant MongoDB unique de l'utilisateur qui a créé la sauce
- **name** : *String* — nom de la sauce
- **manufacturer** : *String* — fabricant de la sauce
- **description** : *String* — description de la sauce
- **mainPepper** : *String* — le principal ingrédient épicé de la sauce
- **imageUrl** : *String* — l'URL de l'image de la sauce téléchargée par l'utilisateur
- **heat** : *Number* — nombre entre 1 et 10 décrivant la sauce
- **likes** : *Number* — nombre d'utilisateurs qui aiment (= likent) la sauce
- **dislikes** : *Number* — nombre d'utilisateurs qui n'aiment pas (= dislike) la sauce
- **usersLiked** : *["String <userId>"]* — tableau des identifiants des utilisateurs qui ont aimé (= liked) la sauce
- **usersDisliked** : *["String <userId>"]* — tableau des identifiants des utilisateurs qui n'ont pas aimé (= disliked) la sauce

Utilisateur

- **email** : *String* — adresse e-mail de l'utilisateur **[unique]**
- **password** : *String* — mot de passe de l'utilisateur haché

Exigences de sécurité

- Le mot de passe de l'utilisateur doit être haché.
- L'authentification doit être renforcée sur toutes les routes sauce requises.
- Les adresses électroniques dans la base de données sont uniques et un plugin Mongoose approprié est utilisé pour garantir leur unicité et signaler les erreurs.
- La sécurité de la base de données MongoDB (à partir d'un service tel que MongoDB Atlas) ne doit pas empêcher l'application de se lancer sur la machine d'un utilisateur.
- Un plugin Mongoose doit assurer la remontée des erreurs issues de la base de données.
- Les versions les plus récentes des logiciels sont utilisées avec des correctifs de sécurité actualisés.
- Le contenu du dossier images ne doit pas être téléchargé sur GitHub.

Repository GitHub

Retirez le code de l'application front-end du repository du projet et suivez les étapes suivantes :

1. Clonez le repository
2. Ouvrez un terminal (Linux/Mac) ou une invite de commande/PowerShell (Windows)
3. Exécutez npm install à partir du répertoire du projet
4. Exécutez npm start
5. Exécutez le back-end sur http://localhost:3000 seulement

Si vous utilisez VSCode, utilisez l'extension LiveShare pour faire fonctionner le serveur front-end sans avoir recours à npm install.