Laravel Notes API

L'objectif est d'écrire et de mettre en production une API HTTP de gestion de notes personnelles. Cette API sera le back-end d'une application web mobile.

Afin de vérifier son bon fonctionnement, ainsi que le respect du cahier des charges et des critères d'évaluation fournis ci-dessous, le code source rendu sera exécuté et vérifié par une suite de tests automatisés écrits par l'enseignant.

Les tests sont disponibles sur le repo public suivant : https://github.com/cba85/teach-notes-token-auth-api-tests

Cahier des charges

Dans cette section, nous décrivons les fonctionnalités attendues dans l'API à produire.

> Attention : Sachant que ces fonctionnalités seront vérifiées par des tests automatisés, merci de respecter ces spécifications à la lettre. Ceci inclut notamment : le nom des routes, la structure des objets JSON à produire, les chaines de caractères fournies…

Spécifications fonctionnelles

Un utilisateur pour s'inscrire, se connecter et se deconnecter depuis une application cliente.

Pour cela, on utilisera une authentification par token de type Bearer en utilisant Laravel Sanctum.

Lors de son utilisation par une application cliente, l'API à fournir devra permettre à chaque utilisateur de l'application de :

- Retrouver ses notes, dans l'ordre antichronologique, avec leur date de création et de mise à jour;
- Saisir une nouvelle note
- Modifier une note
- Supprimer n'importe laquelle de ses notes.

Les notes seront à stocker en tant que texte brut (c.a.d. non HTML) et doivent pouvoir contenir des sauts de ligne, ainsi que n'importe quel caractère Unicode. (accents, emoji…)

Spécifications techniques

Modèle de données

Toutes les données manipulées par l'API doivent être stockées dans une base de données.

Vous devez créer les collections `notes` et `users` qui contient toutes les notes et les utilisateurs.

Schéma de la collection `users`

Chaque `user` doit contenir les propriétés suivantes :

- `id` (type : integer) : identifiant unique de l'utilisateur, généré automatiquement lors de l'insertion
- `name` (type : string) : nom de l'utilisateur
- `email` (type : string) : adresse e-mail de l'utilisateur
- `password` (type : string) : mot de passe de l'utilisateur
- `created_at` (type : date): date et heure à laquelle l'utilisateur a été créé
- `updated_at` (type : date): date et heure à laquelle l'utilisateur a été mis à jour pour la dernière fois

Schéma de la collection `notes`

Chaque `note` doit contenir les propriétés suivantes :

- `id` (type : integer) : identifiant unique de la note, généré automatiquement lors de l'insertion
- `content` (type : string) : contenu textuel de la note
- `created_at` (type : date): date et heure à laquelle la note a été créé
- `updated_at` (type : date): date et heure à laquelle la note a été mise à jour pour la dernière fois
- `user_id` (type : integer) : identifiant de l'utilisateur a qui appartient la note

Vous devez créer une migration pour la création de ces tables dans votre base de données.

Routes

Les routes doivent être capables d'extraire les paramètres passés dans le corps de chaque requête au format `application/json`.

La réponse envoyée par chacune de ces routes doit aussi être au format JSON. En cas d'erreur, elle doit systématiquement contenir la propriété suivante :

- `error` (type: string ou null) : En cas d'erreur pendant l'exécution de la requête, cette propriété aura pour valeur le message d'erreur correspondant. Sinon elle sera null.

Les autres propriétés de la réponse JSON sont spécifiées dans chaque route à implémenter, tel que décrites dans la documentation accessible à l'adresse https://app.swaggerhub.com/apis-docs/cba85/notes-auth/1.0.0.

Critères d'évaluation

Vous devrez rendre deux URLs:

- l'URL du dépôt GitHub contenant le code source, l'historique de commits
- l'URL à laquelle l'API a été déployée en production