Lumen User Management

API de gestion d'utilisateurs (CRUD) utilisant Lumen.

Créer une API permettant de gérer des utilisateurs :

- Listing des utilisateurs
- Affichage d'un utilisateur
- Création d'un utilisateur
- Mise à jour d'un utilisateur
- Suppression d'un utilisateur

Vous devez aussi une API accessible via l'adresse `/api/users`.

Compétences

- Savoir utiliser un routeur
- Savoir utiliser des messages HTTP (requêtes et réponses) PSR-7
- Utiliser une structure MVC
- Valider des formulaires côté serveur
- Se connecter à une base de données
- Savoir utiliser un container PSR-11
- Utiliser un moteur de template
- Créer une API RESTfull
- Savoir créer une API fonctionnelle en utilisant les principes de PHP moderne via un microframework

Frameworks

Vous devez utiliser Lumen pour ce projet :

- [Lumen - PHP Micro-Framework By Laravel]

Base de données

Vous devez respecter le schéma utilisateur suivant :

Schema

```sql
CREATE TABLE `users` (
`id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`created_at` timestamp NULL DEFAULT NULL,
`updated_at` timestamp NULL DEFAULT NULL,
`first_name` varchar(255) DEFAULT NULL,
`last_name` varchar(255) DEFAULT NULL,
`email` varchar(255) DEFAULT NULL,
`password` varchar(255) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Données

```sql
INSERT INTO `users` (`id`, `created_at`, `updated_at`, `first_name`,
`last_name`, `email`, `password`) VALUES
(1,NULL,NULL,'Alexandrie','Weiss','jeannine00@tiscali.fr','b0fa503be939d44fd
27b7869899e2d1f'),
(2,NULL,NULL,'Astrid','Tanguy','audrey61@torres.com','8041a2966dc722518d10a6
6432c7f005'),
(3,NULL,NULL,'Michel','Jacquot','anais.julien@lopes.fr','be81691af9edf69b3ae
b93b6f9b4c718'),
(4,NULL,NULL,'Daniel','Masson','agnes.gomez@voila.fr','5249360dddafdb903dca4
5e0e13b0500'),
(5,NULL,NULL,'Thibaut','Leveque','nathalie85@pereira.com','0d99dd63159fcd098
a79e128101e6d06'),(6,NULL,NULL,'Laure','Le
Roux','bmorel@didier.org','71c695c0725dd2285bbf19aad65dc65e'),
(7,NULL,NULL,'Louis','Le
Roux','alfred09@laposte.net','c60528f06bbe473aea196b521951aad6'),
(8,NULL,NULL,'Sylvie','De
Sousa','bernadette51@tiscali.fr','a7b5689018f7d247af4dfea7091f864b'),
(9,NULL,NULL,'Pierre','Lefebvre','thomas.stephane@ifrance.com','f813771d4c2f
1d657eb2655b0bebf33b'),
(10,NULL,NULL,'Gérard','Fabre','arthur38@orange.fr','3a046eb49bff24b18f84f48
02252da0a');
```

Database

Migration

Créer le schéma de la base de données via une migration Laravel : Database: Migrations

Seeder

Ajouter des utilisateurs fake dans la base de données : Database: Seeding

Routes et controleurs

Vous devez utiliser un système RESTFull.

Remarques

- Vous devez renvoyer l'utilisateur modifié/créé en guise de confirmation sur l'API. En cas de suppression, vous devez renvoyer une réponse vide.
- Vous devez valider vos formulaires côté serveur : Validation
- Vous devez chiffrer les mots de passe lors de la création et de la modification
- Vous devez utiliser un code logique (`Api\UserController` pour le controller de l'API par example).
- Vous devez coder avec un nommage anglais
- Faire un rendu propre sur Github (pas de dossier `vendor/`, `composer.lock`, `tmp/`...) et messages de "commit" propres
- Evitez la duplication de code