teach-node-laranode

Vous devez faire une API de gestion d'utilisateur CRUD en node sans framework.

Aucun package n'est autorisé (sauf mention contraire)

Vous devez donc créer une API REST qui renvoi une ressource CRUD User (bien respecter les méthodes et urls suivantes : https://www.restapitutorial.com/lessons/httpmethods.html)

Structure

Vous devez respecter la structure suivante

```
laranode/
├── app/
│    ├── Controllers/
│    │   └── UserController.js
│    ├── Models/
│    │   └── User.js
│    ├── Repository/
│    │   └── UserRepository.js
│    └── Validator/
│        └── UserValidator.js
├── bootstrap/
│   ├── Db.js
│   └── Router.js
├── routes/
│   └── index.js (routes.js)
├── .env
├── .env.example
├── index.js
└── package.json
```

Modules

Vous êtes libre du choix de votre système de module (CommonJS ou d'ECMAScript)

Classes

Les noms de fichiers avec des majuscules correspondent à des classes. Mais vous pouvez utiliser seulement des fonctions, mais vous devez faire du "Dependency injection" dans tous les cas.

Router

Vous devez gérer 2 types d'erreurs

- 404 : File not found
- 405 : Method not allowed

🟠 Vous devrez gérer un paramètre dynamique `id` pour les routes GET, DELETE et POST `/users/{id}` (ou `/users/{id}` ou même votre propre choix de syntaxe) Vous pouvez gérer un routeur respectant cette syntaxe pour donner un côté Laravel à votre projet

```
router.get("/", (req, res) => {
    // Votre code...
  res.write("Hello");
  res.end();
    // Votre code...
});
```

Dans ce cas, vous devez créer les 4 fonctions `router.get()`, `router.post()`, `router.put()`, `router.delete()` correspondants à leur méthode HTTP respective

Controllers

Les controllers doivent être utilisés dans les routes de la façon suivantes

```
router.get("/users", (req, res) => {
  // Votre code...
  const userController = new UserController(req, res);
  userController.all();
    // Votre code...
});
```

Base de données

Vous devez utiliser PostgreSQL : https://www.npmjs.com/package/pg

Vous devez stocker les identifiants de connexion à la base de données dans un fichier `.env` avec https://www.npmjs.com/package/dotenv

Vous devez créer une base de données comprenant une table `users` disposant des champs suivants :

- id (serial) : primary key
- name (varchar)
- email (varchar)
- created_at (timestamp)
- updated_at (timestamp)
- password (varchar)

Vous devez initialiser une connexion à la base de données via la classe `bootstrap/Db.js`

Repository

Les requêtes à la base de données concernant les users (`all()`, `get(id)`, `update(id, user)`, `create(user)`, `delete(id)`) doivent se trouver dans la classe

`app/repositories/UserRepository.js`

Validation

Vous devez valider les paramètres envoyés sous forme de JSON dans le body de la requête pour les routes POST `/users` et PUT `/users/:id`. Dans ce cas, vous devez renvoyer une erreur 422 (Unprocessable Entity) en cas d'invalidation.

Réponses HTTP

- Vous devez renvoyer une erreur 404 si un utilisateur n'existe pas pour les routes GET, DELETE, PUT `/users/:id`
- Vous devez renvoyer une réponse 200 pour les routes GET, PUT `/users/:id` et GET `/users`. Vous renverrai aussi le ou les utilisateurs récupéré/modifié.
- Vous devez renvoyer une réponse 201 pour la route POST `/users`. Vous renverrai l'utilisateur créé sans son mot de passe.
- Vous devez renvoyer une réponse vide 204 pour la route DELETE `/users/:id`.

Model

Vous devrez créer un model User dans la classe `app/Models/User.js` qui permettra d'envoyer les informations d'un utilisateur via l'API en retirant le champ `password` (pour des raisons évidentes de sécurité).

Config

Vous pouvez gérer en option un fichier de configuration pour ne pas directement appeler les variables d'environnement (comme Laravel) dans `config/app.js`

Récupérer le body d'une requête POST/PUT

Voir https://codeparadox.in/create-rest-api-and-crud-in-node-js-without-any-framework (chapitre Create) ou la fonction `getPostData()` https://github.com/bradtraversy/vanilla-node-rest-api/blob/master/utils.js

Aide

- https://codeparadox.in/create-rest-api-and-crud-in-node-js-without-any-framework
- https://www.section.io/engineering-education/a-raw-nodejs-rest-api-without-frameworks-such-as-express/
- https://www.youtube.com/watch?v=_1xa8Bsho6A
- https://github.com/bradtraversy/vanilla-node-rest-api