Express API - TP 02

01 . La sécurité, c'est mon métier

L'objectif de ce TP sera de mettre en place un système d'authentification pour les utilisateur (les dresseurs). Vous pourrez reprendre les bases que vous avez vu dans l'apprentissage 3. Nous allons rajouter une couche à tout cela : les permissions.

Effectivement, certains utilisateur ont plus de droits que d'autre. Il serait inconcevable qu'un utilisateur non authentifier puisse remplir un pokédex, mais il ne faudrait pas qu'il puisse créer des Pokémon ... Imaginez la FOLIE !

Dans la suite, je me baserais donc dans l'idée que vous avez un modèle utilisateur et un middleware d'authentification. De plus, vous pouvez récupérer les routes `login`, `register` et `checkUser`.

```
Attention, il vous faudra implémenter un username. Libre à vous de
garder ou non l'email
```

02. Permissions

Actuellement, nos Users n'ont pas de rôles et donc nous ne pouvons faire qu'une séparation connecté/non connecté.

Prenons 2 minutes pour réfléchir aux différentes possibilités que va offrir notre projet :

- Créer un Dresseur => C'est une liaison direct à l'utilisateur
- Créer/modifier/supprimer des Pokémons => Il semblerait qu'un rôle soit nécessaire
- Consulter des Pokémons => Tout le monde peut le faire
- Tagguez des Pokémons rencontrés/capturés => Lié à un utilisateurs.

Nous aurons donc besoin d'une différence entre ceux qui peuvent manipuler les Pokémons et ceux qui ne peuvent pas.

Quels choix avons-nous ?

- Un rôle géré par une string (ADMIN,USER,...)
- Un rôle géré par un nombre
- Une liste de permissions (CAN_CREATE_PKMN, CAN_EDIT_PKMN, ...)

Dans un soucis de lisibilité, je vous invite à peut être éviter le nombre.

Votre première étape sera donc de modifier votre schéma User. Ajoutez un champs qui vous permettra d'établir ce que peux faire l'utilisateur (Vous êtes libre de choisir la méthode).

N'oubliez de mettre une valeur par défaut. Si vous voulez créer un admin lors du `/register`, cela ne doit pas être possible. Le seul moyen d'avoir les pouvoir de création de Pokémons ne devrait être accordé qu'a la main (donc en modifiant le document en BD directement)

03. Middleware de perm

Comme le middleware d'authentification, il va nous falloir un middleware de permission.

Cependant, comme nous avons plusieurs possibilité, celui-ci pourra avoir accès à un paramètre (nom du rôle, nom de la permission par exemple).

Pour faire cette magie en JS, voici un exemple. A vous d'adapter car ce middleware ne fait pas grand chose

```
exports.fonction_to_use_in_router = (param) => {
        return (req, res, next) => {
                if (param)
                        return next();
                else
                        return res.status(403).send();
        };
};
```

C'est une fonction qui retourne une fonction ! Ce qui vous permet de l'utiliser dans le routeur

```
app.get('/users', [
        MiddleWare1.fonction_to_use_in_router(param),
        MiddleWare2.fonction_to_use_in_router(param),
        UsersController.list ]);
```

Testez alors différente possibilité sur vos routes (ne pas hésiter à créer des routes pour tester cela) avec différents types d'utilisateurs.

Si tout est bon, alors notre système d'utilisateur sera fonctionnel pour le restant de l'application ! Rien ne vous empêche d'ajouter des rôles/permissions durant la suite du TP, c'est tout l'intérêt !