



Welcome home !

Règlement intérieur

Poster une issue

JavaScript

Backend & Strapi

Prérequis

👋 Semaine 1

Introduction au concept de backend

Bases de Strapi

👋 Semaine 2

Les références

Customisation des routes

👋 Semaine 3

L'authentification

Les différentes BDD utilisables

Stocker des fichiers

Policy et nouvelles routes

🎥 Vidéo

Créer une nouvelle route

Créer une Policy isAuthorized

👋 Semaine 4

Recherche et tri

Git

Git & Github

Northflank : héberger le serveur et la BDD

HTML / CSS

Vue.js

Après la formation

Créer une Policy IsAuthorized

Nous avons vu, précédemment, le fonctionnement du plugin **Users & Permissions**. Il permet, entre autres, de créer, une collection **User**, des routes pour créer un compte et pour se connecter, et un système de token utilisant **JWT**.

Ce plugin permet, également, de décider si les routes de nos **CRUD** sont accessibles à :

- personne
- tout le monde
- seulement les utilisateurs authentifiés qui envoient leurs **JWT** dans les **headers** de la requête

Ce troisième scénario nous permet de vérifier, lorsqu'une requête arrive sur notre serveur, qu'un **JWT** connu du serveur est bien présent dans ses headers. Si c'est le cas, le **controller** de la route en question est exécuté, sinon une erreur est renvoyée au client.

Imaginons que nous ayons une collection **Restaurant** comprenant, entre autre, un champ **owner** faisant référence à un document de la collection **User**.

Nous voudrions sécuriser la route permettant de supprimer un restaurant (**delete**) ; afin que seuls des utilisateurs connectés puissent l'utiliser. Cela, nous pouvons le faire en utilisant le plugin **Users & Permissions** : il suffit de ne rendre cette route disponible que pour des utilisateurs authentifiés.

Différence entre authentification et autorisation