ÉNONCE TRAVAUX PRATIQUE – SUJET 5

Albums et Mongoose

Thèmes

- CRUD
- https
- Mongoose

Présentation

L'objectif de la séance est de réaliser une API permettant de gérer des livres. Les manipulations de base seront permises, à savoir : ajouter un livre, récupérer un ou des livres, mettre à jour un livre et supprimer un livre. Pour rappel, ces fonctionnalités primaires d'une API s'appelle généralement **CRUD** (Create, Read, Update & Delete).
Les informations relatives aux livres que nous utiliserons seront enregistrées dans une base de données **MongoDB**, cela nous permettra d'utiliser **Mongoose** qui est un **ODM** destiné à cette base.
Pour sécuriser le service, les clients devrons utiliser **https** pour communiquer vers l'API, mais le serveur respectera les configurations de bonnes pratiques étudiées en cours.

Ressources indispensables :

- **https://owasp.org**
- **https://www.mongodb.com/fr-fr**
- **https://mongoosejs.com**

Étape 1 – C'est la base

Cette première étape consiste à obtenir un service **MongoDB** fonctionnel sur votre machine. La documentation sur le site officiel **MongoDB** vous indiquera comment obtenir **MongoDB Community** et comment l'installer grâce à la procédure détaillée que vous autre à suivre. En vous connectant avec l'outil **mongosh** de votre service, la commande **show dbs** devrait vous afficher les trois bases de données existant par défaut : **admin**, **config** et **local**.

De même, avec **PhpSotrm**, vous pouvez utiliser l'outil intégré de base de données pour tester et ajouter votre service **MongoDB** et certifier ainsi son bon fonctionnement.

Étape 2 – C'est la base

Dans le projet **NodeJS** que vous allez créer, il faut commencer à installer des dépendances **fastify** et **mongoose**. Comme vous savez le faire, générez des clés de chiffrements pour utiliser **https** avec **fastify**.
Après avoir démarrer et sécurisé le serveur **fastify**, vous devrez relier **NodeJS** à votre service **MongoDB**.
Je vous conseille de créer un fichier de configuration dans un dossier, **databases** par exemple, et d'en exporter votre fonction de connexion. En spécifiant un nouveau nom de base dans l'**URI** de votre connexion, **mongoose** s'occupera de la créer.
L'ODM **mongoose** se base sur un duo **Schema/Model** qu'il faut créer. Organisez votre code pour avoir un fichier décrivant le format des données que nous manipulerons. Nous souhaitons enregistrer un livre avec un **titre**, un **auteur**, une **description** et un **format**. Le **titre** et **l'auteur** sont obligatoire. Le **format** sera soit « poche », « manga » ou « audio », sachant que le format « poche » est le format par défaut. L'ODM se chargera de gérer les identifiants.

Étape 3 – Les routes

Dans un fichier routes.js, vous intègrerez toutes les routes de notre projet. Avec **register**, vous ajouterez vos descriptions à **fastify** sous forme de **plugin**. Les **handlers** de vos routes seront définies dans des fichiers différents dans un dossier **controllers**.
Pour valider les données qui emprunterons ces routes, vous définirez des **Json-Schema** qui seront utilisé en entrée et en sortie. Nous veillerons à ne fournir en sortie uniquement les champs titre, auteur, description et format. Les autres champs créés ne devront pas être visible.

Bon courage…