2024 - TP5

RAPPORT DÉVELOPPEMENT AVANCÉ

Mathéo BIENVENU

SOMMAIRE

1 - EXPLICATIONS DE MES CHOIX ET DIFFICULTÉS RENCONTRÉES

Etape 1 :

Au sein de cette étape, rien de très compliqué : j'ai simplement installé MongoDB Community et MongoDB Compass.

Etape 2 :

Lors de cette seconde étape, j'ai commencé par installer la librairie Fastify ainsi que Mongoose. J'ai également généré les clés de chiffrement avec OpenSSL.

J'ai commencé par créer différents dossiers : controller, database, https et routes. Ces dossiers permettent d'avoir une architecture plus propre et plus facilement modulable. J'ai ensuite créé les fichiers app.js et server.js. Le fichier app.js permet d'initialiser Fastify avec les certificats pour HTTPS et de "register" pour importer les routes, la gestion des formulaires et basic-auth.

Dans le dossier database, je me suis aidé de la documentation pour apprendre à utiliser Mongoose. J'ai configuré la connexion à la base de données, m'assurant qu'elle se lance directement après le démarrage du serveur.

Après avoir pris connaissance des informations d'un livre, j'ai créé un schéma grâce à la documentation de Mongoose. Je l'ai placé dans un fichier livreSchemaModel.js, ce qui facilite sa localisation et permettra d'ajouter de futurs fichiers si l'application souhaite gérer d'autres types d'items, tels que des DVD, etc.

J'ai prévu que si aucun format n'est fourni, le format par défaut soit 'poche'. Il ne peut être que 'poche', 'manga' ou 'audio'. Le titre et l'auteur sont obligatoires et de type String, la description est également de type String mais n'est pas obligatoire.

Etape 3 :

Cette étape a été la plus complexe par rapport aux autres mais consistait uniquement en l'application de ce que nous avons appris lors des différents TP.

Dans le dossier routes, un fichier routes.js est présent et permet de rediriger les requêtes vers la bonne fonction présente dans notre contrôleur, nommé livreController.js.

Chacune de ces routes est également composée d'un schéma, créé dans le fichier validatorSchema.js, situé aussi dans le dossier routes. Cela permet de modifier les schémas JSON très facilement sans avoir à modifier le fichier routes.js. Chacun de ces schémas sert de validateur, triant les données entrantes mais aussi sortantes.

Une fois les requêtes traitées, elles sont redirigées vers la fonction appropriée dans notre contrôleur. Dans ce fichier, nous importons le modèle de schéma nommé Livres, ce qui nous permet d'utiliser directement le schéma défini ainsi que notre connexion à la base de données.

Le contrôleur possède quatre fonctions, chacune associée à un type de requête :

- showLivre, GET
- addLivre, POST
- removeLivre, DELETE
- updateLivre, PUT

2 - AMÉLIORATIONS POSSIBLES

Je ne vois pas de réelle amélioration possible ; tout est assez simple et a été exécuté de manière assez propre. L'architecture est certainement perfectible, mais elle reste simple et claire.

3 - CONCLUSION

En conclusion, ce TP a été particulièrement intéressant car il a favorisé une plus grande autonomie. Moins d'instructions ont été fournies durant le TP, nous obligeant à appliquer ce que nous avons appris lors des précédents travaux pratiques et dirigés.