|  |
| --- |
| Cégep Marie-Victorin |
| TP3-API |
| Document expliquant les mesures appliquées pour répondre aux exigences mentionnées dans ce TP. |

|  |
| --- |
| Mathieu Gouveia Sousa  11-17-2024 |

Contents

[La base de données MongoDB 2](#_Toc182773213)

[Modélisation des Produits 2](#_Toc182773214)

[Mise à jour des Endpoints pour utiliser MongoDB 2](#_Toc182773215)

[Tests Automatisés 2](#_Toc182773216)

[Serveur de Test Sécurisé 3](#_Toc182773217)

# La base de données MongoDB

Pour réussir la connexion au cluster Atlas, je me suis fier à la documentation pour les connections avec Mongoose, disponible ici: <https://mongoosejs.com/docs/connections.html>

La configuration du fichier «.env » était simple en suivant les notes du cours.

# Modélisation des Produits

Créer un modèle Mongoose en utilisant Typescript nécessitait une démarche légèrement différente qu’avec Javascript, je me suis basé encore sur le site de documentation pour Mongoose: <https://mongoosejs.com/docs/typescript.html>

# Mise à jour des Endpoints pour utiliser MongoDB

La majorité de cette tache consistait la modification des méthodes au service layer utiliser pour la version 1 de l’api, les controllers n’ont requis presque aucun changement.

Faire les Swagger docs étais aussi simple, j’ai fait une copie avec quelques changements pour refléter la version et vérifié que les requêtes soient avec les bon endpoints. J’ai donc deux endpoints pour les docs, un pour chaque version de l’api : « /v1/api-docs » et « /v2/api-docs »

# Tests Automatisés

Les tests on requis que je change beaucoup le fonctionnement de l’application. Par exemple, ma méthode POST retournait un message et pas les données qui était posté sur la DB, j’ai dû les modifier pour finir les vérifications avec mes tests.

Pour écrire mes tests avec Mocha, Chai et Supertest, je me suis fier sur ces ressources :

<https://medium.com/spidernitt/testing-with-mocha-and-chai-b8da8d2e10f2>

<https://medium.com/@ehtemam/writing-test-with-supertest-and-mocha-for-expressjs-routes-555d2910d2c2>

Puis pour configurer mon fichier Artillery et faire mes load tests, celle-ci :

<https://blog.appsignal.com/2021/11/10/a-guide-to-load-testing-nodejs-apis-with-artillery.html>

# Serveur de Test Sécurisé

Configurer les environnements et les certificats sont des tâches que j’avais fait au début du TP3 en suivant les notes sur la page du cours. Ensuite, pour créer le pipeline avec Github Actions, je me suis basé sur un exemple ici :

<https://github.com/marketplace/actions/setup-node-js-environment>