



W1 - Stack JavaScript

W-JSC-502

Piscine MERN J02

MongoDB



Piscine MERN J02

repository name: Piscine_MERN_Jour_02
language: Node.js



- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

COMPÉTENCES à ACQUÉRIR

- Javascript
- Node.js
- MongoDB



DÉTAILS ADMINISTRATIFS

- Sauf indication contraire, chaque exercice doit être rendu sous la forme `ex_<numero_exercice>.js` à la racine du répertoire de rendu (par exemple : `ex_42.js` pour l'exercice numéro 42).
- Ne pas rendre le dossier `node_modules`

INTRODUCTION

Pour cette seconde journée, nous allons travailler avec notre Node.js et une base de données MongoDB.



EXERCICE 1 (2 POINTS)

Dans ce premier exercice, vous devez installer MongoDB. Celui-ci doit se lancer automatiquement sur le port 27042 au démarrage.

Vous devez ensuite lancer mongod pour démarrer votre serveur de base de données.

EXERCICE 2 (1 POINT)

Avec l'interpréteur de commande fourni par mongod, vous devez créer une nouvelle base de données nommée « mern-pool ». La création doit être explicite (avec la commande spécialement faite pour) et non, automatisée par mongod.

Tous les prochains exercices seront effectués sur cette base de données.

EXERCICE 3 (4 POINTS)

Vous devez créer votre première collection. Celle-ci doit s'appeler « students ».

Elle doit contenir les champs suivants (avec les options de validation obligatoires) :

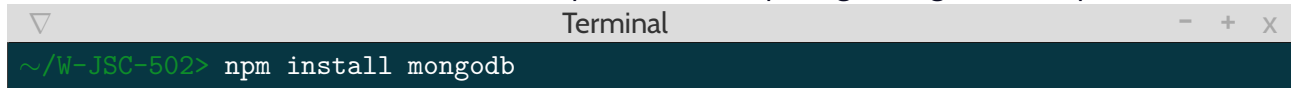
- id : doit être de type « int ».
- lastname : doit être de type « string ».
- firstname : doit être de type « string ».
- email : doit respecter cette regex : `^[\w-\.\.]+@([\w-]+\.\.)+[\w-]{2,4}$`
- phone : doit respecter la syntaxe d'un numéro de téléphone (à vous de choisir une regex ou une autre règle adaptée).
- validated : doit pouvoir contenir ces valeurs suivantes seulement : « in progress », « validated », « rejected ».
- admin : doit être un booléen.

Si une insertion ou une modification d'un document par la suite n'est pas valide, l'opération ne doit pas être effectuée.

EXERCICE 4 (3 POINTS)

Pour cet exercice, vous allez devoir connecter votre serveur Node.js avec votre base de données MongoDB pour qu'ils puissent communiquer ensemble.

Pour ce faire, il vous faut utiliser la commande pour installer le package mongodb avec npm :

A terminal window titled "Terminal" with a dark background. The prompt is "~ / W-JSC-502 >". The command "npm install mongodb" has been entered and executed, with "npm" in green and "install mongodb" in white.

```
~ / W-JSC-502 > npm install mongodb
```

Vous devez écrire une application dans Node.js qui se connecte à votre base de données MongoDB et qui affiche « Connection successfull. » en cas de succès ou qui affiche « Connection failed. » en cas d'échec (si le serveur de la base de données n'est pas démarré par exemple).

EXERCICE 5 (2 POINTS)

Vous devez écrire une application avec Node.js qui affiche un formulaire à l'utilisateur lui permettant de créer un nouvel étudiant.

Lorsque l'utilisateur remplit les différents champs du formulaire et envoie celui-ci, l'application doit remplir avec les informations reçues la collection « students » de votre base de données.

En cas de succès, l'utilisateur se voit affiché « Collection saved. », en cas d'échec (numéro de téléphone non valide par exemple) : « Failed to save the collection. ».

EXERCICE 6 (2 POINTS)

Vous devez écrire une application avec Node.js qui affiche une page à l'utilisateur et qui liste tous les documents de votre collection « students » qui ont le champ « validated » avec la valeur « in progress » et triée par ordre alphabétique du champ « lastname ».

EXERCICE 7 (4 POINTS)

Il est temps de passer aux choses sérieuses. Vous devez créer sous Node.js une application qui vous permet de gérer administrativement votre collection « students ».

Il doit être possible :

- D'afficher toutes les données de votre collection et de les trier par `id`, `lastname` ou `firstname`.
- De rechercher des données (la recherche doit être possible dans tous les champs).
- D'ajouter une nouvelle entrée dans la collection.
- De modifier les valeurs d'une entrée existante.
- De supprimer une entrée existante.



Sur la page d'affichage de toutes les données de votre collection, il doit également être possible d'effectuer des filtres de recherches sur les champs « `id` », « `validated` », « `admin` ». Par exemple, afficher seulement les `id > 15`, ou seulement les `admin à true`, etc.



Aucun cadre n'est fourni pour votre interface, assurez-vous tout de même qu'elle soit ergonomique.



La seule obligation est de pouvoir réaliser les tâches énumérées ci-dessus.

EXERCICE 8 (2 POINTS)

Vous devez réaliser une sauvegarde de votre collection « students ».

Ensuite, vous effacerez toutes les entrées de votre collection, puis vous devrez les restaurer avec votre backup.

N'oubliez pas d'inclure chaque commande dans un script shell nommé `ex_08.sh`.



Non, vous n'avez pas besoin de créer de fichier `ex_08.js` pour cet exercice !



```
Terminal
~/W-JSC-502> ls
ex_08.sh
~/W-JSC-502> ./ex_08.sh save
Backup saved in folder backup.
~/W-JSC-502> ls
ex_08.sh
backup/
~/W-JSC-502> ./ex_08.sh restore
Database restored from backup folder.
~/W-JSC-502> ls
ex_08.sh
```