# W1 - Javascript Avancé

W-JSC-502

# Piscine MERN J03

Express avancé

3.1

{EPITECH.}

# Piscine MERN J03

**repository name:** Piscine_MERN_Jour_03
**repository rights:** ramassage-tek
**language:** Node.js

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).

## COMPÉTENCES À ACQUÉRIR

- Javascript
- Node.js
- MongoDB
- Express.js

## Détails Administratifs

- Sauf indication contraire, chaque exercice doit être rendu dans un fichier nommé `'server.js'` dans son propre dossier nommé 'ex_<numero_exercice>' situé à la racine du projet (par exemple : ex_42/server.js pour l'exercice numéro 42).

## Introduction

Pour cette troisième journée, nous allons travailler avec notre serveur Node.js, notre base de données MongoDB et le framework Express.js.

Nous vous laissons un grand choix quant à l'implémentation des fonctionnalités demandées ; c'est à vous de décider et de défendre vos choix lors de la soutenance.

## Exercice 0 (0 point)

Lire tout le sujet avant de commencer, cela peut aider quant à l'organisation de votre architecture.

## Exercice 1 (0 point)

Si ce n'est pas déjà fait, vous devez installer via le gestionnaire de paquets NPM le paquet Express sur votre serveur Node.js.
Si ce n'est pas déjà fait, vous devez installer MongoDB.

Votre serveur `mondgod` devra être lancé sur le port **27042**, et votre API Node devra être lancée sur le port **4242**.

## Exercice 2 (6 points)

Vous devez créer une application Node.js avec Express et MongoDB.
Celle-ci doit être un espace membre.
Pour cela, implémentez deux fonctionnalités simples :

- Une inscription, reliée à la route `'POST /register'`, qui possèdera comme paramètre `POST`
  - `login` : Le login de l'utilisateur
  - `email` : L'email de l'utilisateur
  - `password` : Le mot de passe de l'utilisateur
  - `passwordConfirm` : Le mot de passe de confirmation de l'utilisateur
- Une connexion, reliée à la route `'POST /login'`
  - `email` : L'email de l'utilisateur
  - `password` : Le mot de passe de l'utilisateur

Les champs de votre base de données décrivant un membre doivent être :

- Un id : de type « int ».
- Un login : de type « string », ayant une longueur comprise entre 5 et 20 caractères ; ce login doit être unique dans la collection.

- Un email : de type « string », qui doit être unique dans la collection.
- Un mot de passe : de type « string » ; celui-ci doit être hashé avec sha1 dans la collection.
- Un type : de type booléen ; celui-ci détermine si le membre est administrateur ou non.

Pour l'inscription, l'utilisateur doit fournir un login, un email, un mot de passe et une confirmation de mot de passe.
Pour la connexion, l'utilisateur se connecte avec son login et son mot de passe.
Par défaut, un membre n'est pas administrateur.
En cas d'erreur, l'utilisateur est informé de l'erreur rencontrée. De plus, quelle que l'erreur soit, le statut HTTP **400** est renvoyé par votre API.
Après une connexion ou une inscription réussie, l'utilisateur doit être redirigé vers une page membre affichant le message « Welcome `<login>` ! » (statut HTTP **200**).

## Exercice 3 (6 points)

Maintenant que vous avez votre espace membre, vous allez devoir créer votre propre boutique en ligne. Dans chaque boutique, il existe des produits à vendre, et c'est par là que nous allons commencer.
Vous devez donc avoir deux pages supplémentaires (accessibles seulement aux membres ou aux administrateurs) dans votre application :

- Une page « boutique » qui affiche tous les produits, avec au minimum :
- Le titre du produit
- Le prix du produit
- Une page « boutique/`<id_produit>` » qui affiche un produit en particulier (après un clic sur un produit de la boutique), et qui donne les détails suivants :
- Le titre du produit
- Le prix du produit
- Une description du produit

## Exercice 4 (4 points)

Vous devez ajouter une page « admin/add » qui permet aux administrateurs d'ajouter des nouveaux produits.
Un produit contient :

- Un titre
- Un prix
- Une description

À vous de définir les types de champs dans votre collection de produits MongoDB ; vous devez faire des choix logiques et les défendre en soutenance.

## Exercice 5 (4 points)

Il va falloir maintenant revoir l'architecture de votre base de données et de votre code. En effet, que serait une boutique sans catégories de produits ?
Dans la partie admin, ajoutez la possibilité de créer une catégorie dans « admin/add/category ».
Dans la partie admin, ajoutez la possibilité de choisir une catégorie pour un produit dans « admin/add/product ».

« admin/add/product » n'est autre que « admin/add » de l'exercice 4.

Dans la partie boutique, ajoutez l'affichage de la catégorie du produit.

## Bonus (10 points)

C'est presque terminé, il ne reste plus qu'à finaliser le tout pour avoir un prototype présentable à de futurs clients.

1. Complétez les CRUD pour les collections suivantes :

- Membre
- Article

- Catégorie

2. Permettez aux admin d'ajouter une photo à leurs produits
3. Ajoutez un système de recherche et de filtre permettant de :

- Rechercher un produit (recherche dans le titre ou dans la description)
- Trier les produits par :
    - Ordre alphabétique sur le titre
    - Ordre de croissance/de décroissance sur le prix
    - Ordre de croissance/de décroissance sur la date de sortie