Plan de formation Node JS

Jour 1 : Présentation de Node JS (Vidéo 1 → Vidéo 5)

- Présentation du coach
- Présentation de l'historique Node JS (depuis le moteur V8 de chrome de jusqu'au Node JS)
- Avantages / inconvénients de Node JS
- Installation de : **Node JS LTS / Postman / VS code / MongoDB + MongoDB Compass**
- Création d'un première application Node JS (en utilisant la commande npm init) et installer la liste des dépendances : **express, cors, morgan**
- Tester votre Hello Word API avec **Postman** et **nodemon**
- Ajouter un script pour lancer l'application en mode dev et en mode prod
- Présentation des REST APIs (Chaque requête dans notre APIs est identifiée par une **METHODE** et un **CHEMIN** d'accès)**.**

Jour 2 : Challenge 1 (Vidéo 6 → Vidéo 13)

- Installation des dépendances : **mongoose**
- Création d'un dossier nommé **models** et dedans créer le modèle **ToDo** → **Bonne pratique :** utilisé les options { versionKey: false, timestamps: true } dans chaque model.
- Création d'un script de connexion avec la base de données
- Création d'un dossiers **routes** contient toutes les routes d'un CRUD
- Tester les APIs avec POSTMAN

Jour 3 : Challenge 2 (Vidéo 14/15)

- Création d'un model **User.**
- Création des APIs demandé (les 5 APIs pour gérer les utilisateurs).
- Création des deux APIs pour affecter et désaffecter une **ToDo** à un **Utilisateur** (utiliser les opérations $pull et $push de mongoose pour ajouter/supprimer l'ID de **ToDo** seulement dans le model **User**)
- Utiliser la méthode **populate()** dans une api qui permet de retourner un ou plusieurs user(s). (**NB :** La méthode « populate » permet d'afficher tout un objet au lieu d'un id seulement).

Jour 4 : Challenge 3

- Création d'une API qui permet d'envoyer un e-mail (sous le format texte).
- Création d'une API qui permet d'envoyer un e-mail (sous le format HTML).
- Création d'une api qui permet d'envoyer un e-mail (sous le format HTML mais cette fois en injectant des variables dans le HTML en utilisant le moteur de template **EJS**).
- Création d'une api qui permet d'envoyer un e-mail et des pièces jointes (c'est-à-dire des **attachments** en utilisant **nodemailer**).

Jour 5 : Challenge 4

- Création d'une API pour sauvegarder un fichier dans un dossier nommé « **uploads »** (en utilisant de l'option **storage** de multer).
- Faire les changements nécessaires pour contrôler les extensions des fichiers à uploader dans le serveur (en utilisant l'option **fileFilter** de multer).
- Ajouter une configuration à expressJS pour dire que le dossier **public** c'est un dossier statique (pour accéder aux fichiers de ce dossier avec une requête GET).

Jour 6 : Challenge 5

- Expliquer le rôle des tâches planifiés (cron tasks) dans notre vie : envoie des SMS de rappelle de payement automatique de dernier jour de chaque mois, envoie des mails automatiques chaque 31 décembre pour vous dire « bonne année », ... Etc
- Création d'un script **cron** en utilisant nodeJs (sans APIs pour être exécuté automatiquement chaque deux minutes).

Jour 7 : Challenge 7

- Création d'une API « registrer » permet de faire l'inscription d'un nouvel utilisateur dans le site.
- Ajouter le cryptage de mot de passe avant l'inscription.
- Ajouter le test si l'email de l'utilisateur existe déjà ou non avant chaque inscription et un message d'erreur sera affiché dans le cas ou l'adresse mail est déjà utilisé (cela peut empêcher l'utilisateur de créer plusieurs mails avec le même adresse e-mail)

Jour 8 : Challenge 6

- Créer une API « login » qui permet de faire l'authentification en utilisant l'adresse email et le mot de passe.
- Modifier l'API « login » pour retourner un jeton d'accès JWT (utiliser la date d'expiration de ce token)
- Expliquer comment on peut décoder le token à travers le site **jwt.io**
- Expliquer l'utilisation de JWT dans les applications Web.
- Utiliser le module « **dotenv** » pour stocker les variables d'environnement.

Jour 9 : Challenge 6

- Expliquer pourquoi on va sécuriser nos APIs (pour des raisons de sécurité).
- Installer les modules nécessaires.
- Dans un dossier « passport-stratégies » il faut créer un script nommé bearer.js qui contient le code de Bearer stratégie. Par la suite, fait l'appel nécessaire dans le fichier principal (généralement index.js)
- Ajouter le middleware **authenticate()** dans les APIs qui peuvent être sécurisé.

Jour 10 : Mini-projet (Optionnel)

- Créer une application Angular/React qui permet de faire la gestion (CRUD) des utilisateurs.
- Créer le backend nécessaire.
- Vous pouvez ajouter la partie authentification des utilisateurs (register/login dans les deux projets) et dans ce cas il faut ajouter dans le projet front les parties suivantes :
  - AuthGuard : pour sécuriser le front.
  - TokenInterceptor : pour injecter le JWT token dans le header les requêtes http.
  - Utiliser les fichiers d'environnements pour configurer le

Jour 11 :

- Choisir un thème de projet.
- Créer des comptes sur **Trello** pour chaque membre dans l'équipe.
- Créer une board et ajouter les tâches à faire dans le projet.
- Chercher et installer une Template Angular/React et publier le projet dans Github.
- Publier le projet backend dans Github.

- Ajouter les collaborateurs dans des deux repositories.
- Créer et publier une branche Github pour chaque membre dans l'équipe.

Jour 12 : Projet (Code Review & Suivi des candidats)

Jour 13 : Projet (Code Review & Suivi des candidats)

Jour 14 : Projet (Code Review & Suivi des candidats)

Jour 15 : Projet (Code Review & Suivi des candidats)

Jour 16 : Projet (Code Review & Suivi des candidats)

Jour 17 : Projet (Code Review & Suivi des candidats)

Jour 18 : Projet (Code Review & Suivi des candidats)

Jour 19 : Projet (Code Review & Suivi des candidats)

Jour 20 : Projet (Code Review & Suivi des candidats)

Jour 21 : Projet (Code Review & Suivi des candidats)

Jour 22 : Projet (Code Review & Suivi des candidats)

Jour 23 : Projet (Code Review & Suivi des candidats)

Jour 24 : Projet (Code Review & Suivi des candidats)

Jour 25 : Projet (Code Review & Suivi des candidats)

Jour 26 : Projet (Code Review & Suivi des candidats)

Jour 27 : Projet (Code Review & Suivi des candidats)

Jour 28 : Projet (Code Review & Suivi des candidats)

Jour 29 : Projet (Code Review & Suivi des candidats)

Jour 30 : Projet (Code Review & Suivi des candidats)