Documentation API node.js

Cette documentation présente une API conçue avec node.js. Cette Api implémente les fonctionnalités permettant d'assurer la gestion de films grâce aux opérations ***CRUD*** (CREATE, READ, UPDATE et DELETE).

Afin de recréer cette API, vous devez suivre les étapes suivantes :

- **Création d'un nouveau répertoire et initialisation de projet node.js.**

Vous devez créer un répertoire dédié au projet par exemple **projetNodejs.** Accédez au répertoire à l'aide d'un terminal tel que (PowerShell, Bash ou autre). Tapez la commande suivante dans votre terminal afin d'initialiser un nouveau projet : **npm init**. Ensuite suivez les instructions pour créer un nouveau package.js par défaut.

- **Installation des dépendances.**

Comme pour tout projet node.js, il faut installer les dépendances nécessaires au projet. Pour ce faire, placez-vous dans le répertoire créé précédemment avec terminal :

  - ✓ Tapez ***: npm install express,*** *pour installer Framework pour le serveur web****).***
  - ✓ *Tapez* ***: npm insatll mysql2,*** *module pour la connexion à base de données.*
  - ✓ *Pour celui qui dispose d'un client front-end tel que(angular, react ...), tapez* ***npm install cors****, pour éviter les blocages de requêtes http .*

- **Configuration du serveur et importation des modules.**

Créez d'abord un fichier index.js à la racine du projet, en suite importez les modules express, mysql2, cors dans des variables différentes. Grâce à la variable qui importe le module express, vous pouvez créer l'application express par exemple(const app=express ()). Le serveur écoute sur le port 3000, vous pouvez également créer une var pour cette valeur.

- **Configuration de la connexion à la base de données.**

La connexion à la base de données nécessite des configurations. En effet, il faut un créer un fichier de configuration qui va abriter les informations de connexion à notre base de données, vous devez créer donc un fichier nommé config.js configuré, comme suit l'exemple suivant :

module.exports={ host:"localhost", user:"root", password:"", database:"devoirapi" }, importez ensuite ce fichier dans index.js.

Vous créerez une base donnée **devoirapi** qui contiendra une table **film** dont vous pouvez créer avec le code sql qui suit :

```
DROP TABLE IF EXISTS `film`;

CREATE TABLE IF NOT EXISTS `film` (

 `id` int NOT NULL AUTO_INCREMENT,

 `nom` varchar(25) NOT NULL,

 `titre` varchar(50) NOT NULL,

 `acteur` varchar(25) NOT NULL,

 `duree` int NOT NULL,

 PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

On peut désormais établir la connexion à la base de données grâce à la méthode createConnection() de l'objet mysql2 importé qui prend en paramètre le fichier config.js. Vous pouvez alors tester le succès ou l'erreur potentielle de la connexion à la base en affichant un message. La méthode connect(), qui prend en paramètre un callback permet de gérer cette tâche.

- **Les opérations CRUD.**

Après les installations et configurations de notre environnement de travail, il faut passe à l'implémentation des routes de notre application.

Avant l'implémentation des routes, il faut configurer l'application pour que celle-ci n'accepte que les données aux formats JSON (Javascript Object Notation). Vous devez écrire cette ligne avant les routes dans le fichier index.js : **app.use(express.json()).** Ce middleware permette de vérifier à requête que les données échangées sont de type JSON.

Une route est une partie dans notre application dédiée à un traitement spécifique. Une route composée de méthode (http), Endpoint et callback dans express. Voici les routes que l'application dispose et leur fonction respective.

- ✓ app.get("/films") Affiches tous les films.
- ✓ app.post("/ajoute/film") Ajoute un nouveau film.
- ✓ aPP.put("/modifie/film/:id") Modifie un film existant.
- ✓ app.delete('/supprime/film/:id'), supprime un film.

Pour tester cette api, vous pouvez utiliser des outils tels que postman,isomnia...