COMMENT CREER UNE APP AVEC NODEJS

* Il faut déjà avoir NODEJS installer sur sa machine.
* Il faut créer un dossier et l’ouvrir dans vsCode.
* Dans le terminal il faut taper npm init pour initialiser un projet NodeJS
* Lors de l’initialisation on peut éventuellement indiquer un repository git
* Il faut insérer un point d’entrée (nom du fichier exemple app.js)
* Indiquer l’auteur

Une fois le projet jsnode initialiser on créer un nouveau fichier (le point d’entrée) et on installe express et fs dans dependencies

EXPRESS : permet de créer une application sans avoir à utilisé le http. Il sert à définir les routes.

FS (FileSystem): permet de gérer et créer des fichiers pour stocker et lire diverses informations

* Dans le terminal, npm install express fs

Dans le fichier app.js on enregistre dans des constantes « express » et « fs », On indique la route pour envoyer les requêtes et recevoir les réponses avec le verbe rest « GET ». Afin de pouvoir récupérer les données, on indique la route (en rajoutant data), on indique a fs dans quel fichier récupérer les donnée Au cas où une erreur se produirait on ajoute un if else. S’il y a une erreur (500), l’app indiquera une erreur. Si au contraire tout s’est bien passée l’app renverra les données au format json

Pour finir, On ajoute un port d’écoute sur lequel on fait tourner l’app.

*const* express = require('express');

*const* fs = require('fs');

*const* app = express();

app.get('/', (*requete*, *response*) *=>*{

*response*.send("salut ca fart !");

})

app.get('/data', (*request*, *response*) *=>*{

    fs.readFile('data.json', (*err*, *data*) *=>*{

        if (*err*) {

*response*.status(500).json({

                message:"Erreur lors de la lecture des données",

            });

        } else {

*response*.status(200).json(JSON.parse(*data*));

        }

    });

});

app.listen(3000, () *=>*{

    console.log("Je tourne sur le port 3000");

})

Afin de ne pas surcharger l’appli, on installe nodemon dans la partie dev

* Dans le terminal, npm install nodemon –save-dev
* Dans le fichier package.json, au niveau des script (dans le code) on écrit :

‘’start’’ : ‘nodemon app.js’

* Dans le terminal, on tape npm start pour lancer le serveur grâce à nodemon il se redémarre automatiquement lorsque l’on sauvegarde dans vsCode

{

*"name"*: "nodejs",

*"version"*: "1.0.0",

*"description"*: "Cours NodeJs",

*"main"*: "app.js",

*"scripts"*: {

*"test"*: "echo \"Error: no test specified\" && exit 1",

*"start"*: "nodemon app.js"

  },

*"author"*: "Bibi",

*"license"*: "ISC",

*"dependencies"*: {

*"express"*: "^4.18.2",

*"fs"*: "^0.0.1-security"

  },

*"devDependencies"*: {

*"nodemon"*: "^2.0.20"

  }

}

On créer un fichier nommer data.js dans lequel on enregistre les données sous la forme d’un JSON

Dans thunder Client on appui sur newRequest, comme URL on note <http://localhost:3000/data>

Localhost = serveur local

3000 = port d’écoute de l’app

Data = ce qu’on souhaite récupérer

RECUPERER DONNEES PAR ID

On créer une nouvelle route avec le verbe Rest « GET » en ajoutant / :id

// C'est une route qui me permet de récupérer une data par son id

// GET "/data/:id"

// Ex : http://localhost:3000/data/:id

app.get('/data/:id', (*request*, *response*) *=>*{

    // Je vais utiliser la méthode readFile du module fs pour pouvoir récuperer la totalité du fichier

fs.readFile("data.json", (*err*, *data*) *=>* {

// Je met une condition si il y a une erreur dans le callback

if ( *err* ) {

    // Je renvoie une réponse status 500 avec un message et l'erreur

    // réponse 500 + erreur + "problémes"

*response*.status(500).json({

        message: "Erreur lors de la lecture des données",

        error: *err*,

        });

        }else{

        // Je parse la chaine de caracteres en Json pour le transformer en JSON manipulable

*const* jsonData = JSON.parse(*data*);

            // Je vais cherchez dans ce fichier si l'id correspondants en paramètres existe dans le contenue

*const* dataById = jsonData.data.find((*obj*) *=>*

*obj*.id === parseInt(*request*.params.id)

            )

            // Si on trouve un objet avec cet id

            if(dataById) {

                // On renvoie une reponse avec un statut 200 et l'objet

*response*.status(200).json(dataById);

            } else {

                // On renvoie une reponse avec un statut 404 avec un message d'erreur

*response*.status(404).json({

                    message:"Aucun objet trouvé avec cet id"

                })

            }

        }

    });

});

CREER UNE NOUVELLE DONNEE DANS LE FICHIER JSON.DATA

// C'est une route qui me permet d'insérer de la donnée dans le fichier data.json

// POST "/data"

// Ex : http://localhost:3000/data

app.post("/data", (*request*, *response*) *=>*{

    // Lire le contenu du fichier

    fs.readFile("data.json", (*err*, *data*) *=>*{

        // Si une erreur sur la lecture du fichier

        if (*err*) {

*response*.status(500).json({

                message: "Erreur lors de la lecture des données",

            });

        } else {

        // Stocker les données existante

*const* existingData = JSON.parse(*data*);

            // rajouter ma donnée

            existingData.data.push(*request*.body)

            // Je vais écrire le fichier avec les nouvelles données

            fs.writeFile('data.json', JSON.stringify(existingData), (*writeErr*) *=>*{

                // Si il y a une erreur renvoi l'erreur 500  + message + error

                if (*err*) {

*response*.status(500).json({

                        message: "Erreur lors de l'ecriture des données",

                    });

                    // renvoie d'une réponse status 200 + message

                } else {

*response*.status(200).json({

                        message: "les données ont été ajouter avec succés",

                    });

                }

            });

        }

    });

});