Etapes du projet :

1 => Création du répertoire de projet

2 => On se déplace à l'intérieur du répo

3 => On initialise notre projet à nodejs avec : npm init (pour création du fichier package json)

4 => Création du fichier app.js

5 => Modification du script package json = ajout clé "start" avec valeur "node app"

6 => Test du script avec la commande npm start

7 => Installation de node express avec la commande = npm i express (le dossier node\_modules apparait + package-lock-js + la clé "dependencies" dans le fichier package.json)

8 => Installation uniquement en mode développement de l'outil (watcher) nodemon avec la commande = npm i nodemon --save-dev

8 bis => Modifier la clé "start" du script dans package json la valeur de start devient "nodemon app" (on démarre toujours avec npm start)

9 => Création du serveur app avec module express dans app.js comme ci-suit :

////////////////////////////////////

const express = require('express');

const app = express();

app.get('/', function (req, res) {

    res.send('Hello World')

})

app.listen(3000, () => {

    console.log('http://localhost:3000');

}))

////////////////////////////////////////

10 => Modification de la route par défaut GET et ajout d'un code de statut de sa réponse (source : https://developer.mozilla.org/fr/docs/Web/HTTP/Status) comme ci-suit :

////////////////////////////////////////

app.get('/dwwm/stagiaires', function (req, res) {

    res.status(200).send('Hello World : I\'ll love JS one day...');

})

////////////////////////////////////////

11 => Convertir la réponse du serveur en objet JSON comme ci-suit :

////////////////////////////////////////

app.get('/dwwm/stagiaires', function (req, res) {

    res

    .status(200)

    .json({

        status : res.statusCode,

        message: 'Hello World : I\'ll love JS one day...!',

        heure : Date.now()

    });

});

////////////////////////////////////////

12 => Ajout de la route POST pour modification base de donnée comme ci-suit :

//////////////////////////////////////////

app.post('/dwwm/stagiaires', (req, res) => {

    res

    .status(200).json({

        status : res.statusCode,

        message : 'Route POST OK'

    });

});

////////////////////////////////////////

12 bis => Test des routes sur l'outil Rest Client

13 => Ajout de la route GET BY ID pour récupérer 1 stagiaire à partir de son identifiant comme ci-suit :

/////////////////////////////////////////////////

//Le parametre :id est indiqué dans la route

app.get('/dwwm/stagiaires/:id', function (req, res) {

    //.params = cible les paramètres indiqués dans la route

    console.log(req.params);

        res

            .status(200)

            .json({

                status : res.statusCode,

                message : 'Route GET BY ID OK'

            });

});

/////////////////////////////////////////////////

14 => Ajout de la route PUT BY ID pour modifier 1 stagiaire en base de donnée à partir de son ID (voir app.js)

15 => Ajout de la route DELETE BY ID pour supprimer 1 stagiaire en base de donnée à partir de son ID (voir app.js)

16 => ORGANISATION MVC

a. Création des controleurs qui stock les fonctions autrefois dans les routes. Exemple comme ci-suit :

/////////////////////////////////////////////////

const allStagiaires = function (req, res) {

    res

        .status(200)

        .json({

            status : res.statusCode,

            message: 'Route GET ALL OK',

        });

};

// La route devient :

app.get('/dwwm/stagiaires', allStagiaires);

/////////////////////////////////////////////////