



self-service-machine-api - STEP 1

Introduction

Notre but est de réaliser le backend de l'application suivante :



Big Mac
\$ 5.99




Mc Chicken
\$ 4.99


-

1


+




Double Cheese Burger
\$ 2.99




Fries
\$ 2.99




Mc Nuggets
\$ 3.49




Salad
\$ 2.79



Coke
\$ 1.99



Ice Tea
\$ 1.99



Water
\$ 1.49

Order Details

Item	Total
1x Mc Chicken	4.99
<hr/>	
Total	4.99

Nous allons utiliser `node.js` et `Express.js` .

Ce backend sera une API REST permettant d'exposer les produits ('Big Mac', 'Mc Chicken', 'Double Cheese Burger', etc) stockés dans une base de données MySQL.

Etape n°1

Dans cette 1ère étape, nous allons :

- configurer le projet
- installer nos premières dépendances
- faire le HelloWorld

Prérequis

Avoir installé la dernière version LTS de node.js

IDE

Nous allons utiliser [Visual Studio Code](#) ainsi que l'extension [Prettier](#)

Grâce à l'extension prettier, votre code doit être formaté automatiquement lorsque vous faites un `Ctrl+S`.

A vous d'installer et de configurer cette extension !

Initialiser le projet

Pour commencer, nous allons configurer notre projet grâce à la commande suivante :

```
npm init
```

A vous de répondre aux différentes questions.

Ne craignez rien ! Vous pourrez toujours modifier les valeurs par la suite.

Cette commande a créé un fichier `package.json`.

Ce fichier, nous permet de gérer nos dépendances, de définir des commandes, etc

Installer express

Nous allons installer notre 1ère dépendance à savoir `Express.js`.

Il s'agit d'un framework pour node.js qui sera l'outil principal de notre backend.

```
npm install express --save
```

A noter l'option `--save` qui indique que nous aurons besoin de cette dépendance pour faire fonctionner cette application en production.

A chaque fois que nous installons une dépendance, une nouvelle entrée est présente dans le fichier `package.json` et le code de la librairie est placé dans le répertoire `node_modules`.

Installer nodemon

La 2ème dépendance est `nodemon` qui permet de redémarrer "à chaud" notre serveur de développement.

```
npm install nodemon --save-dev
```

A noter l'option `--save-dev` qui indique que nous n'utiliserons cette dépendance que pour le développement (mais pas en production).

Modifier le fichier `package.json` :

Nous allons modifier le fichier `package.json` afin de :

- pouvoir faire un `npm start` pour démarrer l'application (script start)
- pouvoir utiliser la syntaxe des modules ES au lieu de la syntaxe commonJS des modules (l'entrée 'type')

```
{
  "name": "self-service-machine-api",
  "type": "module",
  "version": "1.0.0",
  "description": "API REST to self-service-machine application",
  "main": "src/app.mjs ",
  "scripts": {
    "start": "nodemon src/app.mjs"
  },
  "author": "Charmier Grégory",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2"
  },
  "devDependencies": {
    "nodemon": "^3.0.2"
  }
}
```

Créer un fichier `app.mjs` dans un dossier `src`

Dans ce fichier `app.mjs` nous allons faire le HelloWorld.

Prendre exemple sur <https://expressjs.com/fr/starter/hello-world.html> pour afficher le HelloWorld.

Attention ! Vous devez modifier les imports en notation CommonJS et les écrire en notation ES.

```
import express from "express";

const app = express();
const port = 3000;

app.get("/", (req, res) => {
  res.send("Hello World!");
});

app.listen(port, () => {
  console.log(`Example app listening on port http://localhost:${port}`);
});
```

Vous pouvez maintenant exécuter l'application : `npm start`

Passons à l'étape n°2