Lycée LOUIS JOUVET Accompagner pour réussir	Javascript *TP API avec Node.js*	***BTS CIEL*** ***Semestre 1*** ***2025_2026***

Objectif : Créer une api coté serveur http avec node.js

Création du conteneur ubuntu

Créer un conteneur sur le serveur avec l'adresse IP : 10.20.222.1XX

Installation de Node.js et NPM

```
apt update
apt install curl
curl -fsSL https://deb.nodesource.com/setup_18.x | bash -
apt install nodejs
apt install npm
apt install nano
```

Créer un répertoire de travail

```
mkdir projet_nodejs
```

Initialisation du projet

Initialisez votre projet Node.js avec la commande suivante :

```
npm init -y
```

Cela créera un fichier package.json qui contiendra les informations sur le projet et les dépendances.

Installation d'Express.js

Installez Express.js, un framework web pour Node.js, avec la commande suivante :

```
npm install express
```

Création de l'application

Créer un nouveau fichier appelé app.js dans le dossier de projet.

Dans ce fichier, créer une application web simple avec Express.js.

```
const express = require('express');
const app = express();
const port = 3000;
app.get('/', (req, res) => {
        res.send('Bonjour, bienvenue sur mon application Node.js !');
});
app.listen(port, () => {
        console.log(`L'application est en ecoute sur le port ${port}`);
});
```

Exécution de votre application

Exécuter l'application avec la commande suivante :

```
node app.js
```

Tester l'application

Ouvrir un navigateur web et allez à l'adresse http://@ip:3000.

Explication du code :

const express = require('express');

Cette ligne importe le module Express dans l'application. Express est un framework pour Node.js qui fournit des fonctionnalités utiles pour développer des applications web.

const app = express();

Ici, nous initialisons une nouvelle application Express. L'objet app a des méthodes pour les routes et les performances HTTP (GET, POST, PUT, DELETE, etc.), pour spécifier le moteur de rendu des vues, et pour spécifier où se trouvent les vues.

const port = 3000;

Nous définissons le port sur lequel notre application sera exécutée. Vous pouvez choisir n'importe quel numéro de port qui n'est pas déjà utilisé par une autre application.

```
app.get('/', (req, res) => {
    res.send('Bonjour, bienvenue sur mon application Node.js !');
});
```

Ici, nous définissons une route pour l'URL de base ("/") de notre application. Lorsqu'un client envoie une requête HTTP GET à cette URL, la fonction de rappel que nous fournissons est appelée. Cette fonction prend deux arguments : un objet req (requête) qui contient des informations sur la requête HTTP, et un objet res (réponse) qui est utilisé pour définir ce que nous renvoyons au client. Dans ce cas, nous renvoyons simplement le texte "Bienvenue sur mon API !".

```
app.listen(port, () => {
    console.log(`L'application est en écoute sur le port ${port}`);
});
```

Enfin, nous disons à notre application de commencer à écouter les requêtes HTTP sur le port que nous avons défini. La fonction de rappel est appelée une fois que l'application a commencé à écouter sur ce port. Dans ce cas, nous écrivons simplement un message dans la console pour nous informer que l'application est en cours d'exécution.