

Compte Rendu – TP MERN Semaine 1

Cours : Fondations du Back-end avec Node.js et Express

Objectifs du TP:

L'objectif de ce TP est de découvrir les bases du développement back-end avec Node.js et Express.js, de comprendre le concept d'architecture MERN et l'approche "API-First", puis de créer une première API simple que l'on testera avec Postman.

À la fin, je veux :

- Créer un projet Node.js avec NPM
- Utiliser Express pour créer un serveur
- Gérer des routes GET et POST
- Tester des endpoints avec Postman

Étape 1 : Configuration de l'Environnement de Développement

1-Installation de Node.js (version LTS)

Vérification :

```
Microsoft Windows [version 10.0.26100.6584]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Lina>node -v
v22.13.1

C:\Users\Lina>npm -v
11.1.0

C:\Users\Lina>
```

2-Installation de VS Code avec les extensions :

- ESLint
- Prettier
- DotENV
- GitLens

3-Installation de Postman pour tester les requêtes HTTP.

Étape 2 : Démarrage du Projet et Dépendances

Création d'un nouveau dossier :

```
C:\Users\Lina>mkdir Mon-Api-Blog
Un sous-répertoire ou un fichier Mon-Api-Blog existe déjà.

C:\Users\Lina>mkdir mon-api-blog
Un sous-répertoire ou un fichier mon-api-blog existe déjà.

C:\Users\Lina>cd mon-api-blog

C:\Users\Lina\mon-api-blog>npm init -y
Wrote to C:\Users\Lina\mon-api-blog\package.json:

{
  "name": "mon-api-blog",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"Error: no test specified\\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs"
}
```

Installation des dépendances :

```
npm install express
```

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

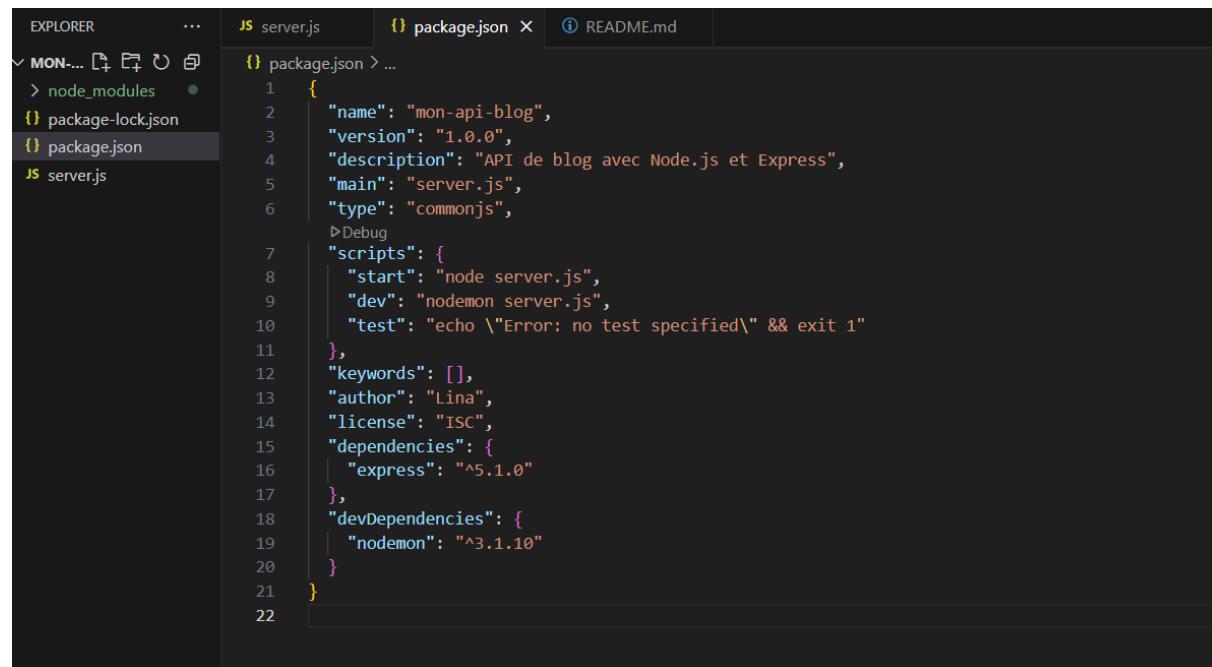
```
C:\Users\Lina\mon-api-blog>npm install express
up to date, audited 96 packages in 983ms
20 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Lina\mon-api-blog>npm install nodemon --save-dev
up to date, audited 96 packages in 1s
20 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Ajout des scripts dans package.json :



The screenshot shows a code editor interface with the following file structure:

- EXPLORER
- ... (dropdown)
- MON-...
- > node_modules
- { package-lock.json
- { package.json
- JS server.js

The package.json file is open in the editor:

```
{
  "name": "mon-api-blog",
  "version": "1.0.0",
  "description": "API de blog avec Node.js et Express",
  "main": "server.js",
  "type": "commonjs",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js",
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "Lina",
  "license": "ISC",
  "dependencies": {
    "express": "^5.1.0"
  },
  "devDependencies": {
    "nodemon": "^3.1.10"
  }
}
```

Étape 3 – Création du serveur Express

Fichier : **server.js**

```
MON-API-BLOG
  node_modules
  package-lock.json
  package.json
JS server.js

JS server.js > ...
1  const express = require('express');
2  const app = express();
3  const PORT = 3000;
4
5  // Middleware pour lire le JSON
6  app.use(express.json());
7
8  // Route d'accueil
9  app.get('/', (req, res) => {
10    res.status(200).send(`<h1>Bienvenue sur l'API de Blog !</h1>`);
11  });
12
13 // Route de test
14 app.get('/api/test', (req, res) => {
15   res.status(200).json({ message: 'Le test a fonctionné !', success: true });
16 });
17
18 // Route POST pour créer un article
19 app.post('/api/articles', (req, res) => {
20   const articleData = req.body;
21   console.log('Données reçues :', articleData);
22   res.status(201).json({
23     message: 'Article créé avec succès !',
24     article: { id: Date.now(), ...articleData }
25   });
26 });
27
28 // Lancer le serveur
29 app.listen(PORT, () => {
30   console.log(`Serveur démarré sur http://localhost:${PORT}`);
31 });

~
```

```
found 0 vulnerabilities
PS C:\Users\Lina\mon-api-blog> npm run dev

> mon-api-blog@1.0.0 dev
> nodemon server.js

> mon-api-blog@1.0.0 dev

> mon-api-blog@1.0.0 dev
> mon-api-blog@1.0.0 dev
> nodemon server.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Serveur démarré sur http://localhost:3000
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
[nodemon] starting `node server.js`
```

Étape 4 – Tests avec Postman

Test 1 : Route d'accueil /

Méthode : **GET**

URL : <http://localhost:3000/>

Résultat : Affiche le message “*Bienvenue sur l'API de Blog !*”

The screenshot shows the Postman application interface. On the left is a sidebar with various icons for collections, environments, and filters. The main workspace has a header bar with tabs for 'server.js', 'HTTP http://localhost:3000/' (which is selected), 'package.json', and 'package-lock.json'. Below the header, there's a search bar and a 'Save' button. The main content area shows a 'New HTTP Request' card with a 'GET' method and the URL 'http://localhost:3000/'. Underneath, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Scripts', and 'Settings'. The 'Params' tab is active, showing a table with one row and two columns: 'Key' and 'Value'. In the 'Body' section, there are tabs for 'Pretty', 'Raw', 'Preview', and 'HTML'. The 'Pretty' tab is selected, showing the response body: '1 <h1> Bienvenue sur l'API de Blog !</h1>'. At the top right of the main area, there are buttons for 'Send' and 'Code Cookies'. Below the main area, status information is displayed: 'Status: 200 OK Time: 49 ms Size: 268 B'. There are also icons for copy, share, and refresh.

Test 2 : Route de test /api/test

Méthode : **GET**

URL : <http://localhost:3000/api/test>

Réponse JSON :

The screenshot shows the Postman interface with a successful API call. The top bar shows the URL `http://localhost:3000/api/test`. The main area displays a GET request with the following details:

- Method:** GET
- URL:** `http://localhost:3000/api/test`
- Params:** An empty table for Query Params.
- Body:** An empty table for Body.
- Headers:** A table with 7 items.
- Test Results:** Status: 200 OK, Time: 7 ms, Size: 287 B.
- Network:** Local Address: ::1:53536, Remote Address: ::1:3000.
- Body (Pretty):**

```
1 [ {  
2   "message": "L'  
3   "success": tr  
4 } ]
```

Test 3 : Route POST /api/articles

- Méthode : **POST**
- URL : `http://localhost:3000/api/articles`
- Onglet **Body → raw → JSON**

The screenshot shows the Postman interface with a successful API call. The top bar shows the URL `http://localhost:3000/api/articles`. The main area displays a POST request with the following details:

- Method:** POST
- URL:** `http://localhost:3000/api/articles`
- Params:** An empty table for Params.
- Body:** A table with 9 items, set to **raw** mode. The body content is:

```
1 {  
2   "title": "Mon premier article",  
3   "content": "Ceci est le contenu de mon article.",  
4   "author": "Lina"  
5 }  
6
```
- Headers:** A table with 9 items.
- Test Results:** Status: 201 Created, Time: 18 ms, Size: 314 B.
- Body (JSON):**

```
1 {  
2   "message": "Article créé avec succès !",  
3   "article": {  
4     "id": 1759777613400  
5   }  
6 }
```

