

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG  
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**



# **BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 3**

## **ĐỀ TÀI XÂY DỰNG WEBSITE ĐẶT VÉ XEM PHIM TRỰC TUYẾN**

**Giảng viên hướng dẫn : Kim Ngọc Bách**

**Họ và tên : Trần Quang Nam**

**Mã sinh viên : B22DCDT208**

**Lớp : E22CQCN02-B**

# Tìm Hiểu Về Node.js Và Express.js

## 1. Giới Thiệu Node.js

### Node.js là gì?

Node.js là một môi trường chạy JavaScript phía server, giúp bạn xây dựng các ứng dụng web nhanh, hiệu quả và có khả năng mở rộng tốt. Nó sử dụng V8 Engine (công cụ chạy JavaScript của Google Chrome) để thực thi mã JavaScript trên máy chủ.

### Tại sao nên sử dụng Node.js?

- Mô hình non-blocking, xử lý bất đồng bộ (Asynchronous I/O): Không giống như PHP hay Python, Node.js có thể xử lý nhiều yêu cầu cùng lúc mà không làm tắc nghẽn hệ thống.
- Chạy nhanh và hiệu suất cao: Nhờ vào V8 Engine, tốc độ thực thi JavaScript của Node.js rất nhanh.
- Sử dụng chung một ngôn ngữ (JavaScript) cho cả Frontend và Backend: Điều này giúp lập trình viên dễ dàng phát triển ứng dụng full-stack.
- Có hệ sinh thái phong phú với NPM (Node Package Manager): Hàng triệu thư viện hỗ trợ giúp tiết kiệm thời gian phát triển.

## 2. Express.js – Framework Web cho Node.js

### Express.js là gì?

Express.js là một framework web phổ biến cho Node.js, giúp việc xây dựng server và API trở nên đơn giản hơn. Nó cung cấp nhiều tính năng mạnh mẽ như:

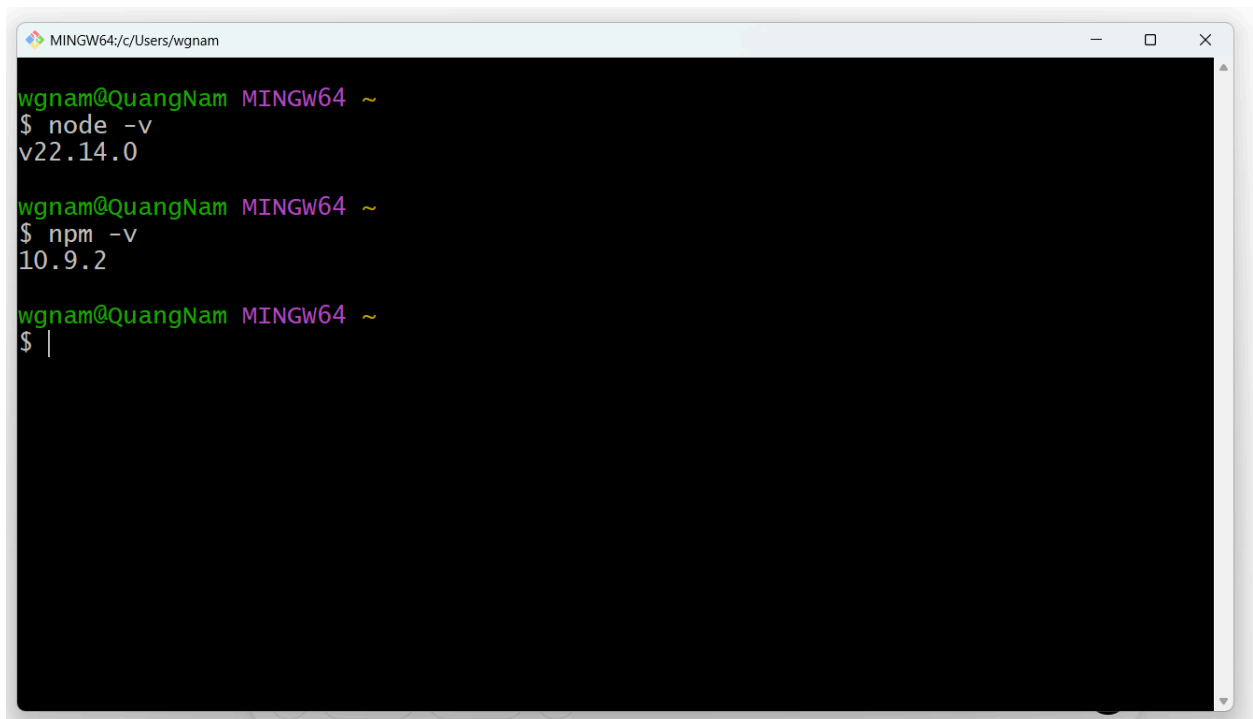
- Routing (định tuyến URL)
- Middleware (xử lý yêu cầu trước khi trả về phản hồi)
- Hỗ trợ REST API dễ dàng
- Quản lý session và cookie

### 3. Cài Đặt Node.js và Express.js

#### Cài đặt Node.js

Có thể tải Node.js từ trang chủ: <https://nodejs.org>

Sau khi cài đặt, kiểm tra phiên bản:



```
MINGW64~/c/Users/wgnam
wgnam@QuangNam MINGW64 ~
$ node -v
v22.14.0

wgnam@QuangNam MINGW64 ~
$ npm -v
10.9.2

wgnam@QuangNam MINGW64 ~
$ |
```

## Tạo Dự Án Node.js với Express.

```
wgnam@QuangNam MINGW64 /d/tìm hiểu
$ npm install express

added 69 packages in 7s

14 packages are looking for funding
  run `npm fund` for details
```

## 4. Tạo Server Cơ Bản với Express

Tạo file server.js và thêm đoạn mã sau:

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

const app = express();

const port = 3000;

app.get('/', (req, res) => {

  res.send('Hello, Express!');

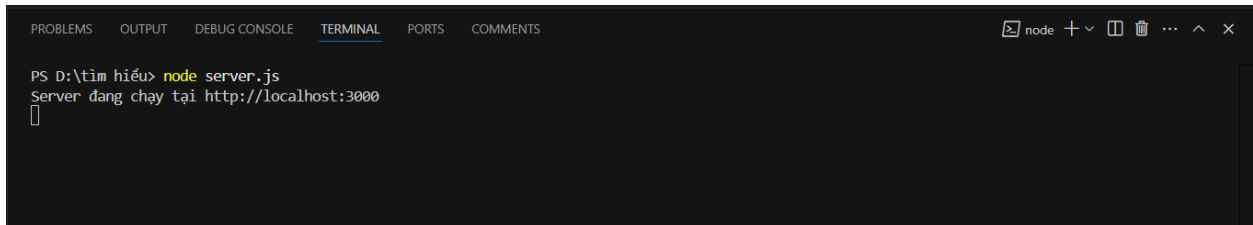
});

app.listen(port, () => {

  console.log(`Server đang chạy tại http://localhost:${port}`);
```

```
});
```

## Chạy server: node server.js



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS D:\tìm hiểu> node server.js
Server đang chạy tại http://localhost:3000
█
```



## 5. Routing trong Express

Routing giúp bạn định nghĩa các đường dẫn trong ứng dụng.

```
app.get('/about', (req, res) => {  
    res.send('Trang Giới Thiệu');  
});  
  
app.get('/contact', (req, res) => {  
    res.send('Trang Liên Hệ');  
});
```

## 6. Sử Dụng Middleware

Middleware là các hàm trung gian giúp xử lý request trước khi gửi response.

Ví dụ: Một middleware đơn giản để log request:

```
app.use((req, res, next) => {  
  
  console.log(`${req.method} ${req.url}`);  
  
  next();  
  
});
```

### Cài đặt body-parser để xử lý JSON:

```
wgnam@QuangNam MINGW64 /d/tìm hiểu  
$ npm install body-parser  
  
added 10 packages, changed 3 packages, and audited 80 packages in 3s  
  
15 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities
```

### Sử dụng trong server:

```
const bodyParser = require('body-parser');  
  
app.use(bodyParser.json());
```

## 7. REST API Với Express.js

Tạo một API CRUD (Create, Read, Update, Delete) cơ bản.

```
const users = [];  
  
app.post('/users', (req, res) => {  
  const user = req.body;  
  users.push(user);  
  res.status(201).json(user);  
});  
  
app.get('/users', (req, res) => {  
  res.json(users);  
});
```

## 8. Kết Nối Với MongoDB (Mongoose)

Cài đặt MongoDB và thư viện mongoose:

```
wgnam@QuangNam MINGW64 /d/tìm hiểu
$ npm install mongoose

added 20 packages, and audited 100 packages in 14s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

## Kết nối với MongoDB

Tạo file db.js:

```
const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/mydb', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => console.log('Kết nối MongoDB thành công'))
.catch(err => console.log(err));
```

## Tạo Model User

Tạo file models/User.js:

```
const mongoose = require('mongoose');
```



```
const UserSchema = new mongoose.Schema({  
  
  name: String,  
  
  email: String  
  
});  
  
module.exports = mongoose.model('User', UserSchema);
```

## Thêm API CRUD cho MongoDB

Trong server.js:

```
const User = require('./models/User');  
  
app.post('/users', async (req, res) => {  
  
  const user = new User(req.body);  
  
  await user.save();  
  
  res.status(201).json(user);  
  
});  
  
app.get('/users', async (req, res) => {
```

```
const users = await User.find();

res.json(users);

});
```

## 11. Tổng Kết - Node.js và Express.js trong Phát Triển Ứng Dụng Web

Sau khi tìm hiểu về Node.js và Express.js, chúng ta có thể thấy rằng đây là một bộ công cụ mạnh mẽ giúp xây dựng các ứng dụng web và API hiện đại. Dưới đây là tóm tắt chi tiết về từng phần quan trọng mà chúng ta đã khám phá.

### 1. Node.js - Sức Mạnh của JavaScript ở Backend

Node.js đã thay đổi cách xây dựng backend bằng cách đưa JavaScript lên máy chủ, mang lại những lợi ích như:

- Hiệu suất cao: Sử dụng V8 Engine giúp thực thi JavaScript cực nhanh.
- Xử lý bất đồng bộ (Asynchronous): Mô hình event-driven và non-blocking I/O giúp ứng dụng mở rộng tốt.
- Nhẹ và linh hoạt: Không cần máy chủ cố định như Apache hoặc Nginx, bạn có thể triển khai dễ dàng trên nhiều nền tảng.
- Hệ sinh thái phong phú: Với NPM (Node Package Manager), hàng triệu thư viện hỗ trợ giúp tăng tốc độ phát triển.

### 2. Express.js - Giúp Việc Xây Dựng API Trở Nên Dễ Dàng

Express.js là một framework web tối giản nhưng mạnh mẽ cho Node.js. Nó cung cấp:

- Routing đơn giản: Giúp định nghĩa các endpoint cho API nhanh chóng.
- Middleware mạnh mẽ: Hỗ trợ xử lý request, response, logging, authentication...
- Khả năng mở rộng cao: Dễ dàng tích hợp với các dịch vụ bên ngoài như cơ sở dữ liệu, xác thực, caching...

Nhờ Express.js, chúng ta có thể dễ dàng xây dựng các REST API hoặc ứng dụng full-stack với frontend như React, Angular, hoặc Vue.js.

### **3. MongoDB - Lưu Trữ Dữ Liệu Hiệu Quả với Mongoose**

MongoDB là một cơ sở dữ liệu NoSQL phổ biến, đặc biệt phù hợp với Node.js nhờ:

- Lưu trữ dữ liệu dưới dạng JSON/BSON: Dễ dàng làm việc với JavaScript.
- Không có schema cố định: Giúp dễ dàng thay đổi cấu trúc dữ liệu.
- Khả năng mở rộng tốt: Hỗ trợ sharding và replication.

Với Mongoose, chúng ta có thể làm việc với MongoDB dễ dàng hơn bằng cách định nghĩa schema, model, và thực hiện các thao tác CRUD nhanh chóng.