Hướng dẫn xây dựng API với Node.js, Express và MongoDB

Tác giả: Đặng Kim Thi

1. Giới thiệu

Bài viết này hướng dẫn bạn cách xây dựng một API RESTful cơ bản để quản lý danh sách sản phẩm, sử dụng Node.js, Express và MongoDB. API này hỗ trợ các thao tác CRUD (Create, Read, Update, Delete) trên dữ liệu sản phẩm.

2. Các công nghệ sử dụng

2.1. Node.js

Node.js là một môi trường chạy JavaScript phía máy chủ, giúp phát triển các ứng dụng web hiệu quả và có hiệu suất cao.

2.2. Express.js

Express.js là một framework nhẹ và mạnh mẽ của Node.js, giúp dễ dàng xây dựng API và quản lý request/response.

2.3. MongoDB

MongoDB là một cơ sở dữ liệu NoSQL, lưu trữ dữ liệu dưới dạng JSON-like, giúp quản lý dữ liệu linh hoạt và mở rộng dễ dàng.

2.4. Mongoose

Mongoose là một thư viện ODM (Object Data Modeling) giúp làm việc với MongoDB một cách dễ dàng hơn thông qua schema và model.

3. Cách tư duy và các bước thực hiện

3.1. Xác định yêu cầu hệ thống

- API cần có khả năng quản lý danh sách sản phẩm.
- Hỗ trợ các thao tác CRUD trên sản phẩm.
- Dữ liệu cần được lưu trữ trong MongoDB.

3.2. Thiết kế cấu trúc

- Xây dựng một server Express.js.
- Kết nối MongoDB và sử dụng Mongoose để quản lý dữ liệu.
- Định nghĩa schema và model cho sản phẩm.
- Xây dựng các API endpoint cho CRUD.

3.3. Cách tiếp cận

- Viết code theo từng bước từ đơn giản đến phức tạp.
- Sử dụng các nguyên tắc RESTful để thiết kế API.
- Kiểm tra từng phần trước khi tích hợp toàn bộ hệ thống.

4. Cài đặt môi trường

4.1. Cài đặt Node.js và npm

Trước tiên, cần cài đặt Node.js nếu chưa có. Bạn có thể tải về từ nodejs.org.

Sau khi cài đặt, kiểm tra phiên bản Node.js và npm bằng lệnh:

```
node -v
npm -v
```

4.2. Cài đặt MongoDB

MongoDB có thể được cài đặt cục bộ hoặc sử dụng dịch vụ MongoDB Atlas.

Nếu cài đặt cục bộ, bạn có thể tải MongoDB từ mongodb.com.

Chạy MongoDB bằng lệnh:

```
mongod --dbpath /path/to/your/data/directory
```

(Tương tự như phần hướng dẫn cũ)

5. Xây dựng server API

Tạo file `server.js` và nhập mã sau:

5.1. Khởi tạo ứng dụng và kết nối MongoDB

```
const express = require('express');
const mongoose = require('mongoose');
const app = express();
const PORT = 3000;

// Middleware để parse JSON
app.use(express.json());

// Kết nối MongoDB
mongoose.connect('mongodb://localhost:27017/productapi', {
    useNewUrlParser: true,
    useUnifiedTopology: true
```

```
}).then(() => console.log('Kết nối thành công với MongoDB'))
  .catch(err => console.error('Lỗi kết nối', err));
```

5.2. Định nghĩa Schema và Model

```
const productSchema = new mongoose.Schema({
    name: String,
    price: Number
});
const Product = mongoose.model('Product', productSchema);
```

5.3. API CRUD với giải thích chi tiết

Lấy danh sách sản phẩm

```
app.get('/products', async (req, res) => {
    const products = await Product.find(); // Lấy toàn bộ sản phẩm từ
database
    res.json(products);
});
```

Lấy sản phẩm theo ID

```
app.get('/products/:id', async (req, res) => {
    const product = await Product.findById(req.params.id); // Tìm sản phẩm
theo ID
    if (!product) return res.status(404).json({ message: 'Sản phẩm không
tồn tại' });
    res.json(product);
});
```

Thêm sản phẩm mới

```
app.post('/products', async (req, res) => {
    const newProduct = new Product(req.body); // Tạo sản phẩm mới từ dữ
liệu yêu cầu
    await newProduct.save();
    res.status(201).json(newProduct);
});
```

Cập nhật sản phẩm

```
app.put('/products/:id', async (req, res) => {
    const updatedProduct = await Product.findByIdAndUpdate(req.params.id,
req.body, { new: true });
    res.json(updatedProduct);
});
```

Xóa sản phẩm

```
app.delete('/products/:id', async (req, res) => {
    await Product.findByIdAndDelete(req.params.id);
    res.json({ message: 'Sản phẩm đã bị xóa' });
});
```

5.4. Khởi động server

```
app.listen(PORT, () => console.log(`Server đang chạy tại
http://localhost:${PORT}`));
```

6. Kiểm thử API

6.1. Chạy server

Khởi động server bằng lệnh:

```
node server.js
```

6.2. Kiểm thử với Postman hoặc curl

6.2.1. Lấy danh sách sản phẩm

- **Method**: GET
- **URL**: http://localhost:3000/products
- **Kết quả mong đợi**:

```
[
  { "_id": "123", "name": "Tai nghe Bose 700", "price": 9000000 },
  { "_id": "456", "name": "Tai nghe Sony WH-1000XM4", "price": 8000000 }
]
```

6.2.2. Lấy sản phẩm theo ID

- **Method**: GET
- **URL**: http://localhost:3000/products/{id}
- **Ví dụ**: http://localhost:3000/products/123
- **Kết quả mong đợi**:

```
{ "_id": "123", "name": "Tai nghe Bose 700", "price": 9000000 }
```

6.2.3. Thêm sản phẩm mới

- **Method**: POST
- **URL**: http://localhost:3000/products
- **Headers**:
  - Content-Type: application/json
- **Body**:

```
{ "name": "Bàn phím cơ Keychron K6", "price": 1800000 }
```

- **Kết quả mong đợi**:

```
{ "_id": "789", "name": "Bàn phím cơ Keychron K6", "price": 1800000 }
```

6.2.4. Cập nhật sản phẩm

- **Method**: PUT
- **URL**: http://localhost:3000/products/{id}
- **Ví dụ**: http://localhost:3000/products/789
- **Headers**:
  - Content-Type: application/json
- **Body**:

```
{ "name": "Bàn phím cơ Keychron K8", "price": 2000000 }
```

- **Kết quả mong đợi**:

```
{ "_id": "789", "name": "Bàn phím cơ Keychron K8", "price": 2000000 }
```

6.2.5. Xóa sản phẩm

- **Method**: DELETE
- **URL**: http://localhost:3000/products/{id}

- **Ví dụ**: `http://localhost:3000/products/789`
- **Kết quả mong đợi**:

```
{ "message": "Sản phẩm có ID 789 đã bị xóa" }
```

7. Kết luận

Bài viết này hướng dẫn cách xây dựng API RESTful cơ bản với Node.js, Express và MongoDB. Bạn có thể mở rộng thêm chức năng như xác thực người dùng, phân trang dữ liệu hoặc cải thiện hiệu suất bằng caching.

Tác giả: Đặng Kim Thi