

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2023 - 2024

Nghiên cứu NodeJS và xây dựng ứng dụng bán hàng trực tuyến

Giáo viên hướng dẫn:
ThS. Ngô Thanh Huy

Sinh viên thực hiện:
Họ tên: Nguyễn Hoàng Khang
MSSV: 110121035
Lớp: DA21TTB

Trà Vinh, tháng 01 năm 2024

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2023 - 2024

Nghiên cứu NodeJS và xây dựng ứng dụng bán hàng trực tuyến

Giáo viên hướng dẫn:
ThS. Ngô Thanh Huy

Sinh viên thực hiện:
Họ tên: Nguyễn Hoàng Khang
MSSV: 110121035
Lớp: DA21TTB

Trà Vinh, tháng 01 năm 2024

.....

[illegible]

(Ký tên và ghi rõ họ tên)

This image shows a full page of white paper with horizontal dotted lines, typical of primary school writing paper. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Tôi xin bày tỏ lòng biết ơn chân thành đến Thầy Ngô Thanh Huy đã dành thời gian để hướng dẫn và hỗ trợ tôi trong quá trình thực hiện đề tài. Sự tận tâm và sự đã giúp tôi vượt qua những khó khăn và phát triển những kỹ năng trong suốt quá trình thực hiện đề tài.

Tôi cũng muốn bày tỏ lòng biết ơn đặc biệt đến gia đình, mọi người đã luôn ở bên cạnh và hỗ trợ tôi về tinh thần. Sự động viên đã là nguồn động lực tinh thần lớn, giúp tôi vượt qua khó khăn.

Ngoài ra, tôi tự hào về sự cố gắng và nỗ lực mà bản thân đã đặt vào đề tài này. Tuy đề tài vẫn còn nhiều hạn chế do áp lực thời gian nhưng tôi cảm thấy bản thân đã học được rất nhiều thứ từ khi nhận được đề tài đến khi hoàn thành. Đây sẽ là bước đệm để tôi có thể phát triển thêm trong tương lai.

Đề tài này không chỉ là kết quả của sự nỗ lực bản thân mà còn là sự kết hợp của tất cả những góp ý, động viên đến từ Thầy và gia đình. Xin chân thành cảm ơn tất cả.

MỤC LỤC

| | |
|---|-----------|
| LỜI CẢM ƠN | 3 |
| MỤC LỤC | 4 |
| DANH MỤC HÌNH | 7 |
| DANH MỤC TỪ VIẾT TẮT | 8 |
| TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH | 9 |
| CHƯƠNG 1: TỔNG QUAN | 10 |
| 1.1. Tổng quan về đề tài | 10 |
| 1.2. Các công nghệ sử dụng trong đề tài | 10 |
| 1.3. Kết quả thực hiện..... | 11 |
| CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT | 12 |
| 2.1. Tổng quan về NodeJS..... | 12 |
| 2.1.1. Giới thiệu NodeJS | 12 |
| 2.1.2. Cách hoạt động của NodeJS | 12 |
| 2.1.3. Các tính năng chính của NodeJS | 13 |
| 2.1.4. Ưu điểm của NodeJS | 13 |
| 2.1.5. Nhược điểm của NodeJS | 13 |
| 2.2. Tổng quan về Express Framework | 14 |
| 2.2.1. Giới thiệu Express Framework..... | 14 |
| 2.2.2. Ứng dụng của Express Framework | 14 |
| 2.2.3. Các tính năng chính của Express Framework | 14 |
| 2.2.4. Ưu điểm của Express Framework | 15 |
| 2.2.5. Nhược điểm của Express Framework | 15 |
| 2.2.6. Giới thiệu Template Engine Express Handlebars..... | 15 |
| 2.3. Tổng quan về MongoDB | 16 |
| 2.3.1. Giới thiệu MongoDB..... | 16 |
| 2.3.2. Các tính năng chính của MongoDB | 16 |
| 2.3.3. Ưu điểm của MongoDB | 17 |
| 2.3.4. Nhược điểm của MongoDB | 17 |

| | |
|--|-----------|
| 2.4. Mô hình MVC (Model – View – Controller) | 17 |
| 2.4.1. Giới thiệu về Mô hình MVC | 17 |
| 2.4.2. Các thành phần trong mô hình MVC | 17 |
| 2.4.3. Cách hoạt động của mô hình MVC | 18 |
| 2.4.4. Ưu điểm của mô hình MVC | 19 |
| 2.4.5. Nhược điểm của mô hình MVC | 19 |
| 2.5. RESful API | 20 |
| 2.5.1. Giới thiệu về RESful API | 20 |
| 2.5.2. Cách hoạt động của RESful API | 20 |
| CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU | 21 |
| 3.1. Đặt tả hệ thống | 21 |
| 3.2. Sơ đồ mô tả chức năng tìm kiếm sản phẩm | 21 |
| 3.3. Sơ đồ mô tả chức năng thêm mới sản phẩm | 22 |
| 3.4. Sơ đồ mô tả chức năng cập nhật sản phẩm | 22 |
| 3.4. Sơ đồ mô tả chức năng xóa sản phẩm | 23 |
| 3.5. Mô hình dữ liệu | 23 |
| 3.6. Cấu trúc dự án | 24 |
| 3.7. Thành phần Model | 25 |
| 3.7.1. Model Product | 25 |
| 3.7.2. Model Login | 26 |
| 3.7.3. Model Order | 27 |
| 3.8. Thành phần Controller | 28 |
| 3.8.1. Home Controller | 28 |
| 3.8.2. Product Controller | 29 |
| 3.8.3. Order Controller | 30 |
| 3.9. Thành phần View | 31 |
| 3.9.1. View Home | 31 |
| 3.9.2. View Order | 32 |
| 3.9.3. View Search | 33 |

| | |
|---|-----------|
| CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU | 35 |
| 4.1. Hướng dẫn cài đặt..... | 35 |
| 4.2. Kết quả thử nghiệm | 35 |
| 4.3. Cách hoạt động của JSON..... | 35 |
| 4.4. Giao diện trang Home | 36 |
| 4.5. Giao diện trang thêm sản phẩm | 37 |
| 4.6. Giao diện trang tìm kiếm sản phẩm..... | 37 |
| 4.7. Giao diện trang cập nhật sản phẩm..... | 38 |
| CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN | 39 |
| 5.1. Kết luận..... | 39 |
| 5.2. Hạn chế | 39 |
| 5.3. Hướng phát triển..... | 39 |
| DANH MỤC TÀI LIỆU THAM KHẢO | 40 |
| PHỤ LỤC | 41 |
| BẢNG MÔ TẢ CHI TIẾT CHO CHUỖI JSON MẪU..... | 41 |

DANH MỤC HÌNH

| | |
|--|----|
| Hình 2. 1 Cấu trúc thư mục của Express Handlebars..... | 16 |
| Hình 2. 2 Cách hoạt động của mô hình MVC | 19 |
| Hình 3. 1 Sơ đồ mô tả chức năng tìm kiếm sản phẩm..... | 19 |
| Hình 3. 2 Sơ đồ mô tả chức năng thêm mới sản phẩm..... | 20 |
| Hình 3. 3 Sơ đồ mô tả chức năng cập nhật sản phẩm..... | 20 |
| Hình 3. 4 Sơ đồ mô tả chức năng xóa sản phẩm | 21 |
| Hình 3. 5 Mô hình dữ liệu | 21 |
| Hình 3. 6 Cấu trúc dự án..... | 22 |
| Hình 3. 7 Model Product | 23 |
| Hình 3. 8 Model Login | 24 |
| Hình 3. 9 Model Order | 25 |
| Hình 3. 10 Home Controller | 26 |
| Hình 3. 11 Product Controller | 27 |
| Hình 3. 12 Order Controller | 28 |
| Hình 3. 13 View Home | 29 |
| Hình 3. 14 View Order | 30 |
| Hình 3. 15 View Search | 31 |
| Hình 4. 1 Chuỗi JSON được trả về View | 34 |
| Hình 4. 2 Giao diện trang Home | 34 |
| Hình 4. 3 Giao diện thêm sản phẩm | 35 |
| Hình 4. 4 Giao diện tìm kiếm sản phẩm..... | 35 |
| Hình 4. 5 Giao diện cập nhật sản phẩm..... | 36 |

DANH MỤC TỪ VIẾT TẮT

HTML: Hyper Text Markup Language.

HTTP: Hyper Text Transfer Protocol.

JSON: JavaScript Object Notation.

MVC: Model View Controller.

TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH

Đặt vấn đề

NodeJS là một nền tảng phát triển ứng dụng web dựa trên JavaScript, cho phép xây dựng các ứng dụng đa nền tảng, hiệu năng cao và dễ mở rộng. Trong thời đại công nghệ số, việc xây dựng các ứng dụng bán hàng trực tuyến là một nhu cầu thiết yếu của các doanh nghiệp và người tiêu dùng. Do đó nghiên cứu NodeJS và xây dựng ứng dụng bán hàng trực tuyến là một đề tài mang tính thực tế cao.

Đối tượng nghiên cứu

Đối tượng tập trung nghiên cứu trong đề tài là Framework NodeJS - một môi trường chạy mã JavaScript phía máy chủ, cho phép phát triển các ứng dụng một cách hiệu quả và linh hoạt. Đồng thời, cần nghiên cứu xây dựng ứng dụng bán hàng trực tuyến, nắm rõ các yêu cầu mà một trang thương mại điện tử cần có.

Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài này sẽ tập trung vào việc nghiên cứu NodeJS và sử dụng nó để xây dựng một ứng dụng bán hàng trực tuyến cơ bản, bao gồm các chức năng như hiển thị sản phẩm, chỉnh sửa, tìm kiếm và thanh toán.

Các hướng tiếp cận

Nghiên cứu sâu lý thuyết của NodeJS, bao gồm cách hoạt động, những đặc điểm nổi bật cũng như cần xác định cụ thể yêu cầu, chức năng của một ứng dụng bán hàng trực tuyến. Thực hiện triển khai ứng dụng và kết hợp một số công nghệ liên quan. Thực nghiệm để kiểm tra hiệu suất và đánh giá ứng dụng. So sánh kết quả thử nghiệm với các nền tảng phát triển khác để đưa ra nhận xét.

Kết quả đạt được

Hiểu rõ cách hoạt động của NodeJS, các ưu điểm và hạn chế của nó. Phát triển được một ứng dụng bán hàng trực tuyến với các chức năng cơ bản như hiển thị sản phẩm, thêm, xóa và sửa sản phẩm. Triển khai được ứng dụng trên máy tính cá nhân hoặc hosting và hoạt động ổn định, khả năng mở rộng tốt.

CHƯƠNG 1: TỔNG QUAN

1.1. Tổng quan về đề tài

Đề tài “Nghiên cứu NodeJS và xây dựng ứng dụng bán hàng trực tuyến” tập trung vào việc nghiên cứu NodeJS và áp dụng để phát triển một ứng dụng thương mại điện tử để minh họa khả năng xây dựng ứng dụng của nền tảng này.

Nghiên cứu về NodeJS, nền tảng phát triển ứng dụng dựa trên JavaScript runtime. Đồng thời tìm hiểu cách hoạt động, các ưu điểm và hạn chế của nó so với các nền tảng khác.

Phân tích yêu cầu và những chức năng cơ bản của một ứng dụng bán hàng trực tuyến. Xây dựng cơ sở dữ liệu, thiết kế giao diện, sử dụng các thư viện hỗ trợ. Thực hiện so sánh, đối chiếu với các nền tảng khác để nhận xét tính linh động và hiệu suất của NodeJS.

1.2. Các công nghệ sử dụng trong đề tài

Express: đây là một framework cho NodeJS, giúp xây dựng ứng dụng web và API một cách nhanh chóng và dễ dàng.

Express-Handlebars: là một engine template cho Express, giúp tạo ra giao diện người dùng một cách dễ dàng và linh hoạt.

Method-Override: là một middleware cho Express, giúp hỗ trợ các phương thức HTTP như PUT và DELETE trong các trình duyệt không hỗ trợ.

Mongoose: là một Object Data Modeling library cho MongoDB và NodeJS, giúp quản lý kết nối và quản lý mô hình dữ liệu được chặt chẽ hơn.

Mongoose-Slug-Updater: là một plugin cho Mongoose, giúp tạo và cập nhật các slug cho các mô hình Mongoose.

Morgan: là một middleware logger cho NodeJS, giúp ghi lại các yêu cầu HTTP đến máy chủ.

Nodemon: đây là một công cụ giúp phát triển ứng dụng NodeJS bằng cách tự động khởi động lại ứng dụng khi có thay đổi trong mã nguồn.

1.3. Kết quả thực hiện

Sau quá trình tìm hiểu, nghiên cứu và thực hiện đề tài, tôi đã xây dựng được một trang web có các chức năng cơ bản như hiển thị sản phẩm, tìm kiếm sản phẩm theo tên, chỉnh sửa sản phẩm và đặt mua sản phẩm, đáp ứng được yêu cầu cơ bản của người dùng. Đây sẽ là cơ sở để phát triển thêm các chức năng phức tạp trong tương lai.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1. Tổng quan về NodeJS

2.1.1. Giới thiệu NodeJS

NodeJS là một môi trường chạy mã JavaScript phía máy chủ, được xây dựng trên nền tảng V8 JavaScript engine của Google Chrome. Điều này cho phép NodeJS tận dụng sức mạnh của V8, giúp thực thi mã JavaScript với tốc độ nhanh.

NodeJS sử dụng mô hình lập trình không đồng bộ, giúp xử lý nhiều yêu cầu cùng lúc mà không cần chờ đợi. Điều này tăng hiệu suất và khả năng mở rộng của ứng dụng, giúp ứng dụng có thể phục vụ hàng ngàn người dùng cùng lúc mà không gặp phải vấn đề về hiệu suất.

NodeJS được phát triển với mục tiêu xây dựng các ứng dụng mạng có khả năng mở rộng cao. Nó kết hợp với JavaScript, ngôn ngữ lập trình phổ biến nhất trên web, tạo ra một nền tảng mạnh mẽ cho việc phát triển ứng dụng web phía máy chủ. Điều này giúp lập trình viên có thể sử dụng cùng một ngôn ngữ lập trình cho cả phía máy chủ và phía người dùng, giảm bớt độ phức tạp khi phát triển ứng dụng.

NodeJS cũng hỗ trợ cộng đồng lập trình viên rộng lớn thông qua npm (Node Package Manager), một công cụ quản lý gói cho phép lập trình viên chia sẻ và sử dụng lại mã nguồn một cách dễ dàng. Npm cung cấp hàng ngàn thư viện và công cụ, giúp lập trình viên có thể tận dụng mã nguồn đã được viết sẵn, giảm bớt thời gian và công sức cần thiết để phát triển ứng dụng từ đầu.

Tóm lại, NodeJS là một công cụ mạnh mẽ cho việc phát triển ứng dụng web hiện đại, đáp ứng nhu cầu của thế giới số hóa ngày càng tăng. NodeJS giúp xây dựng ứng dụng có khả năng mở rộng, hiệu suất cao và dễ dàng bảo trì, đáp ứng nhu cầu của thị trường thương mại điện tử ngày càng phát triển.

2.1.2. Cách hoạt động của NodeJS

NodeJS hoạt động dựa trên mô hình không đồng bộ, điều này có nghĩa là nó không chờ đợi một tác vụ hoàn thành trước khi chuyển sang tác vụ tiếp theo. Thay vào đó, nó

đăng ký một callback để thông báo khi tác vụ hoàn thành, cho phép nó tiếp tục xử lý các tác vụ khác.

Điều này giúp NodeJS xử lý hàng ngàn yêu cầu đồng thời mà không cần nhiều tài nguyên hệ thống. Điều này làm cho NodeJS trở thành một lựa chọn lý tưởng cho việc xây dựng các ứng dụng mạng có khả năng mở rộng cao, như các ứng dụng bán hàng trực tuyến.

2.1.3. Các tính năng chính của NodeJS

Đơn luồng: NodeJS hoạt động trên một luồng duy nhất, điều này giúp NodeJS có khả năng mở rộng cao. Tuy nhiên, nếu có lỗi xảy ra trong luồng đó, toàn bộ ứng dụng có thể bị dừng lại.

Tương thích đa nền tảng: NodeJS hoạt động trên nhiều nền tảng khác nhau, bao gồm Microsoft Windows và Linux. Điều này giúp NodeJS trở thành một lựa chọn tốt cho việc phát triển ứng dụng đa nền tảng.

Hiệu suất cao: NodeJS đạt được hiệu suất cực kỳ cao trong khi hoạt động thông qua một luồng duy nhất, qua đó giúp xử lý hàng ngàn yêu cầu đồng thời mà không cần nhiều tài nguyên hệ thống.

2.1.4. Ưu điểm của NodeJS

Dễ học: NodeJS được xây dựng trên nền tảng JavaScript, ngôn ngữ lập trình phổ biến nhất trên web. Giúp cho mọi người có thể dễ dàng tiếp cận NodeJS.

Cộng đồng lớn: NodeJS có một cộng đồng lập trình viên lớn và tích cực. Giúp cho việc sửa lỗi hoặc tìm kiếm tài liệu dễ dàng hơn.

Tương thích đa nền tảng: NodeJS hoạt động trên nhiều nền tảng khác nhau nên có thể phát triển đa nền tảng

Dễ bảo trì: NodeJS có một hệ thống file đơn giản, giúp việc bảo trì ứng dụng trở nên dễ dàng hơn.

2.1.5. Nhược điểm của NodeJS

Khó xử lý các tác vụ tốn nhiều CPU: NodeJS được thiết kế để xử lý các tác vụ I/O không đồng bộ, trong khi các tác vụ tốn nhiều CPU thường là các tác vụ đồng bộ. Do đó, NodeJS có thể gặp khó khăn khi xử lý các tác vụ này.

Hỗ trợ thư viện không đầy đủ: NodeJS có một cộng đồng phát triển lớn và tích cực, nhưng vẫn có một số thư viện không được hỗ trợ đầy đủ.

Thiếu cấu trúc chặt chẽ: NodeJS không có một cấu trúc nghiêm ngặt, điều này có thể khiến việc tổ chức dự án và quản lý mã nguồn trở nên khó khăn, đặc biệt khi ứng dụng phát triển lớn và phức tạp.

2.2. Tổng quan về Express Framework

2.2.1. Giới thiệu Express Framework

Express Framework được xây dựng trên nền tảng NodeJS. Express cung cấp cho các nhà phát triển một bộ các tính năng và tiện ích sẵn có, giúp họ xây dựng các ứng dụng web và API nhanh chóng và hiệu quả. Express được phát triển bởi TJ Holowaychuk, và được phát hành lần đầu tiên vào năm 2010. Express nhanh chóng trở nên phổ biến và được sử dụng bởi hàng triệu nhà phát triển trên toàn thế giới.

2.2.2. Ứng dụng của Express Framework

Express có thể được sử dụng để xây dựng nhiều loại ứng dụng web và API, bao gồm:

Ứng dụng web tĩnh: ứng dụng web tĩnh là ứng dụng web chỉ chứa nội dung tĩnh, chẳng hạn như HTML, CSS và JavaScript. Express có thể được sử dụng để phục vụ nội dung tĩnh cho các ứng dụng web tĩnh.

Ứng dụng web động: ứng dụng web động là ứng dụng web có thể tạo ra nội dung động, chẳng hạn như nội dung được tạo bởi cơ sở dữ liệu hoặc các tính toán. Express có thể được sử dụng để xây dựng ứng dụng web động.

RESTful API: là một API dựa trên các nguyên tắc REST. Express có thể được sử dụng để xây dựng API một cách nhanh chóng và hiệu quả.

2.2.3. Các tính năng chính của Express Framework

Các tính năng cơ bản đặc trưng của Express Framework bao gồm:

Routing: Express cho phép định nghĩa các đường dẫn (routes) cho ứng dụng. Mỗi đường dẫn có thể được kết nối với một hàm xử lý (handler), chịu trách nhiệm xử lý các yêu cầu HTTP đến đường dẫn đó.

Mô hình lập trình vòng lặp sự kiện (event loop): Express sử dụng mô hình lập trình vòng lặp sự kiện để xử lý các yêu cầu HTTP. Mô hình này cho phép Express xử lý nhiều yêu cầu đồng thời một cách hiệu quả.

Hỗ trợ thư viện rộng rãi: Express có một cộng đồng phát triển lớn và tích cực, cung cấp một số lượng lớn các thư viện bổ sung cho Express. Các thư viện này có thể được sử dụng để thêm các tính năng mới hoặc cải thiện hiệu suất của ứng dụng.

2.2.4. Ưu điểm của Express Framework

Tốc độ xử lý nhanh: Express sử dụng mô hình lập trình vòng lặp sự kiện để xử lý các yêu cầu HTTP. Mô hình này cho phép Express xử lý nhiều yêu cầu đồng thời một cách hiệu quả.

Cộng đồng lớn mạnh: Express có một cộng đồng phát triển lớn và tích cực, cung cấp một số lượng lớn các thư viện bổ sung cho Express giúp cải thiện hiệu suất của ứng dụng.

Dễ học: Express sử dụng JavaScript, một ngôn ngữ lập trình phổ biến và dễ học, giúp cho những người mới bắt đầu sử dụng Express một cách nhanh chóng và dễ dàng.

2.2.5. Nhược điểm của Express Framework

Cấu trúc không chặt chẽ: Express không có một cấu trúc nghiêm ngặt, điều này có thể khiến việc tổ chức dự án và quản lý mã nguồn trở nên khó khăn, đặc biệt khi ứng dụng phát triển lớn và phức tạp.

2.2.6. Giới thiệu Template Engine Express Handlebars

Express Handlebars là một Template Engine được sử dụng trong Express. Nó cho phép tạo các mẫu HTML với theo một quy ước cụ thể. Đây là một công cụ hỗ trợ hiển thị ra các trang HTML với nội dung động.

Handlebars giúp tách phần logic và phần giao diện và cho phép tái sử dụng HTML. Để sử dụng được Express Handlebars chúng ta cần cài đặt bằng câu lệnh:

\$ npm install express-handlebars

Sau khi cài đặt cần cấu trúc thư mục như sau:



Hình 2. 1 Cấu trúc thư mục của Express Handlebars

Tạo nội dung HTML trong thư mục layouts và thêm nội dung cần thiết. Cuối cùng cần tạo các route trong ứng dụng để xử lý yêu cầu từ người dùng trả về các trang HTML được tạo bởi thư viện.

2.3. Tổng quan về MongoDB

2.3.1. Giới thiệu MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở đa nền tảng viết bằng C++. Bản ghi trong MongoDB được lưu trữ dạng một dữ liệu văn bản, là một cấu trúc dữ liệu bao gồm các cặp giá trị và trường tương tự như các đối tượng JSON.

MongoDB được phát triển bởi 10gen (nay là MongoDB, Inc.) và được phát hành lần đầu tiên vào năm 2009. MongoDB nhanh chóng trở nên phổ biến và được sử dụng bởi hàng triệu nhà phát triển trên toàn thế giới.

2.3.2. Các tính năng chính của MongoDB

MongoDB cung cấp một số tính năng chính, bao gồm:

Dữ liệu dạng văn bản: MongoDB lưu trữ dữ liệu dưới dạng văn bản, điều này giúp cho dữ liệu trở nên linh hoạt và dễ sử dụng hơn.

Cơ sở dữ liệu phân tán: MongoDB có thể được phân tán trên nhiều máy chủ, giúp cho cơ sở dữ liệu có thể mở rộng quy mô và đáp ứng được nhu cầu của các ứng dụng có khối lượng dữ liệu lớn.

Mô hình dữ liệu linh hoạt: MongoDB không có một mô hình dữ liệu cố định, giúp cho dữ liệu có thể lưu trữ theo cách phù hợp nhất với nhu cầu của ứng dụng.

2.3.3. Ưu điểm của MongoDB

Khả năng linh hoạt cao: MongoDB cung cấp một mô hình dữ liệu linh hoạt, cho phép lưu trữ dữ liệu phù hợp nhất với ứng dụng.

Tốc độ truy vấn cao: MongoDB sử dụng mô hình dữ liệu phân tán, giúp cho cơ sở dữ liệu có thể xử lý các truy vấn một cách nhanh chóng và hiệu quả.

Dễ sử dụng: MongoDB cung cấp một API đơn giản và dễ sử dụng, điều này giúp cho những người mới làm quen cơ sở dữ liệu MongoDB có thể sử dụng một cách nhanh chóng, tiết kiệm thời gian.

2.3.4. Nhược điểm của MongoDB

Không có tính nhất quán toàn vẹn: MongoDB không đảm bảo tính nhất quán toàn vẹn, có thể dẫn đến các vấn đề khi các truy vấn thay đổi dữ liệu trong cùng một thời điểm.

Không hỗ trợ các truy vấn phức tạp: MongoDB không hỗ trợ các truy vấn phức tạp, chẳng hạn như việc gộp nhiều collection với nhau.

2.4. Mô hình MVC (Model – View – Controller)

2.4.1. Giới thiệu về Mô hình MVC

Kiến trúc MVC được thảo luận lần đầu vào năm 1979 bởi Trygve Reenskaug, một nhà khoa học máy tính người Na Uy. MVC là một mô hình thiết kế phần mềm phân tách một ứng dụng thành ba thành phần chính:

Model: quản lý dữ liệu và xử lý logic nghiệp vụ.

View: hiển thị dữ liệu cho người dùng.

Controller: điều khiển sự tương tác giữa Model và View.

Mô hình MVC giúp tách biệt các trách nhiệm của ứng dụng, giúp cho ứng dụng dễ dàng phát triển, bảo trì và mở rộng. MVC đã trở nên phổ biến trong phát triển ứng dụng web, đặc biệt là các ứng dụng web dựa trên hướng đối tượng. Nhiều ngôn ngữ lập trình web hiện đại đều hỗ trợ mô hình MVC, chẳng hạn như Java, PHP, Python, Ruby,...

2.4.2. Các thành phần trong mô hình MVC

Model: là thành phần chịu trách nhiệm quản lý dữ liệu của ứng dụng. Model có thể bao gồm các thành phần sau:

Lớp dữ liệu: lưu trữ dữ liệu của ứng dụng.

Lớp logic nghiệp vụ: xử lý các yêu cầu của người dùng đối với dữ liệu.

View: là thành phần chịu trách nhiệm hiển thị dữ liệu cho người dùng. View có thể bao gồm các thành phần sau:

HTML: định nghĩa cấu trúc và nội dung của giao diện.

CSS: định nghĩa kiểu dáng của giao diện.

JavaScript: xử lý các tương tác của người dùng với giao diện.

Controller: là thành phần trung gian giữa Model và View. Controller chịu trách nhiệm tiếp nhận yêu cầu từ người dùng, xử lý yêu cầu và trả về kết quả cho View. Controller có thể bao gồm các thành phần sau:

Lớp xử lý yêu cầu: tiếp nhận yêu cầu từ người dùng.

Lớp xử lý dữ liệu: gọi tới các phương thức trong Model để lấy hoặc cập nhật dữ liệu.

Lớp xử lý hiển thị: xác định giao diện hiển thị dữ liệu trên View.

2.4.3. Cách hoạt động của mô hình MVC

Quy trình hoạt động của mô hình MVC bao gồm các bước sau:

Người dùng tương tác với View.

View gửi thông tin đến Controller.

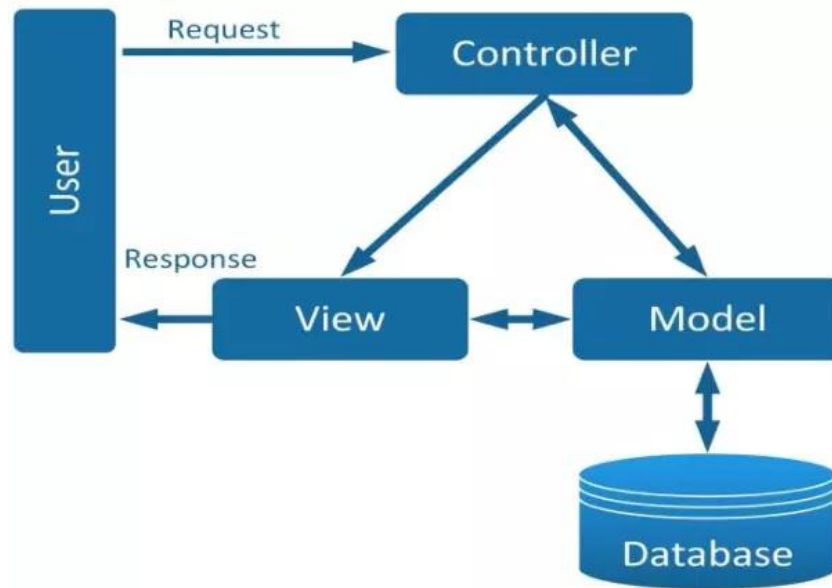
Controller nhận thông tin từ View và xử lý yêu cầu.

Controller gọi các phương thức trong Model để lấy hoặc cập nhật dữ liệu.

Model lấy thông tin từ cơ sở dữ liệu hoặc thực hiện các thao tác xử lý dữ liệu. Sau đó, trả về kết quả cho Controller.

Controller nhận kết quả từ Model và quyết định cách hiển thị dữ liệu trên View.

View cập nhật giao diện để hiển thị thông tin mới nhận được từ Model.



Hình 2. 2 Cách hoạt động của mô hình MVC

2.4.4. Ưu điểm của mô hình MVC

Tải nhanh, tối ưu lượng băng thông: Mô hình MVC tách biệt dữ liệu và giao diện, giúp cho ứng dụng tải nhanh hơn và tối ưu lượng băng thông.

Kiểm tra dễ dàng: mô hình MVC giúp cho việc kiểm tra ứng dụng dễ dàng hơn, bởi vì các thành phần của ứng dụng được tách biệt rõ ràng.

Khả năng cung cấp nhiều chế độ xem: mô hình MVC cho phép ứng dụng cung cấp nhiều chế độ xem khác nhau cho người dùng, chẳng hạn như chế độ xem web, chế độ xem di động,...

Dữ liệu trả về không cần định dạng: mô hình MVC cho phép Model trả về dữ liệu thô, giúp cho View có thể định dạng dữ liệu theo cách phù hợp với giao diện của mình.

2.4.5. Nhược điểm của mô hình MVC

Tăng độ phức tạp của dữ liệu: mô hình MVC có thể làm tăng độ phức tạp của dữ liệu, vì dữ liệu cần được truyền giữa các thành phần của ứng dụng.

Tốn nhiều thời gian tìm hiểu: mô hình MVC có thể tốn nhiều thời gian tìm, bởi vì mô hình này có khá nhiều khái niệm và nguyên tắc cần phải nắm vững nên sẽ khó khăn trong việc tiếp cận mô hình.

Khó khăn khi tổ chức và quản lý file: mô hình MVC có thể gặp khó khăn khi tổ chức và quản lý file, các thành phần của ứng dụng thường được lưu trữ trong các file riêng biệt nên sẽ gây khó khăn trong quá trình tìm kiếm và chỉnh sửa file.

2.5. RESful API

2.5.1. Giới thiệu về RESful API

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web để tiện cho việc quản lý các tài nguyên. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video...), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

REST (REpresentational State Transfer) là một kiểu kiến trúc dùng để thiết kế API, cho phép chuyển đổi cấu trúc dữ liệu. REST sử dụng các phương thức HTTP đơn giản như GET, POST, DELETE để tạo ra giao tiếp giữa các máy.

API (Application Programming Interface) là một bộ quy định và cơ chế cho phép một ứng dụng hoặc một thành phần tương tác với ứng dụng hoặc thành phần khác. API có thể cung cấp dữ liệu mà ứng dụng cần trong các định dạng dữ liệu thông dụng như JSON hoặc XML.

RESTful API không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một RESTful API.

2.5.2. Cách hoạt động của RESful API

RESTful API hoạt động chủ yếu dựa vào giao thức HTTP để truyền tải dữ liệu là các đối tượng JSON bằng các phương thức:

GET: trả về một tài nguyên hoặc một danh sách các tài nguyên.

POST: tạo mới một tài nguyên.

PUT: cập nhật thông tin cho tài nguyên.

DELETE: xóa một tài nguyên.

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1. Đặt tả hệ thống

Hệ thống có các chức năng cơ bản như hiển thị sản phẩm, thêm, xóa, sửa sản phẩm. Ngoài ra chức năng tìm kiếm sản phẩm và đặt mua sản phẩm cũng được triển khai trong hệ thống. Người mua hàng có thể xem, tìm kiếm và đặt mua sản phẩm. Quản trị viên có thể thêm sản phẩm, sửa và xóa sản phẩm.

Giao diện người dùng sử dụng framework Express Handlebar kết hợp với thư viện Bootstrap để hiển thị danh sách sản phẩm, chi tiết sản phẩm, và quản lý sản phẩm. Sử dụng giao thức HTTPS để giao tiếp với Backend thông qua các RESTful API. Ngoài ra hệ thống còn sử dụng MongoDB để lưu trữ thông tin người dùng, sản phẩm và đơn hàng. Kết nối Backend đến cơ sở dữ liệu để thực hiện các truy vấn và lấy dữ liệu bằng các phương thức GET, POST, UPDATE và DELETE. Hệ thống phân chia rõ ràng và linh hoạt giữa Frontend và Backend, sử dụng NodeJS kết hợp cơ sở dữ liệu MongoDB để tạo nên hệ thống linh hoạt và hiệu quả.

3.2. Sơ đồ mô tả chức năng tìm kiếm sản phẩm



Hình 3. 1 Sơ đồ mô tả chức năng tìm kiếm sản phẩm

Người dùng sẽ nhập từ khóa vào form tìm kiếm, sau đó form sẽ gửi yêu cầu đến Controller tìm kiếm tương ứng bằng phương thức GET. Controller tiếp nhận thông tin xử lý và truy xuất thông tin tương ứng từ cơ sở dữ liệu MongoDB. Cơ sở dữ liệu sẽ trả về chuỗi JSON thông tin sản phẩm tìm kiếm và Controller tiếp nhận sẽ chuyển người dùng đến trang tìm kiếm sản phẩm và hiển thị nội dung người dùng cần tìm kiếm.

3.3. Sơ đồ mô tả chức năng thêm mới sản phẩm



Hình 3. 2 Sơ đồ mô tả chức năng thêm mới sản phẩm

Người dùng nhập thông tin cần thêm sản phẩm mới bao gồm tên, mô tả sản phẩm, hình ảnh, giá bán vào form nhập thông tin sản phẩm. Sau đó form sẽ gửi yêu cầu đến Controller tương ứng với phương thức là POST để đưa chuỗi JSON từ form đến cơ sở dữ liệu. MongoDB sẽ tiếp nhận thông tin và lưu vào các documents của collection tương ứng. Sau khi lưu vào cơ sở dữ liệu người dùng sẽ được chuyển hướng đến trang hiển thị giao diện của sản phẩm.

3.4. Sơ đồ mô tả chức năng cập nhật sản phẩm



Hình 3. 3 Sơ đồ mô tả chức năng cập nhật sản phẩm

Giao diện cập nhật thông tin sản phẩm sẽ hiển thị toàn bộ danh sách các sản phẩm, người dùng sẽ chọn sản phẩm cần cập nhật sau đó sẽ cập nhật lại thông tin của sản phẩm bao gồm tên, mô tả sản phẩm, hình ảnh, giá bán. Khi người dùng ấn Cập Nhật thì form sẽ gửi yêu cầu đến Controller cập nhật với phương thức là UPDATE để đưa chuỗi JSON từ form đến cơ sở dữ liệu. Sau đó người dùng sẽ chuyển đến trang cập nhật sản phẩm để tiếp tục cập nhật sản phẩm.

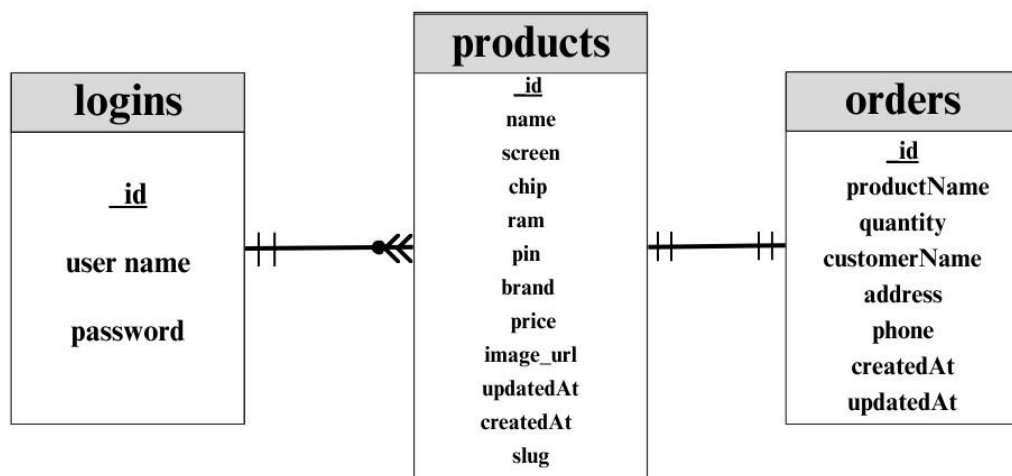
3.4. Sơ đồ mô tả chức năng xóa sản phẩm



Hình 3. 4 Sơ đồ mô tả chức năng xóa sản phẩm

Giao diện xóa sản phẩm sẽ hiển thị danh sách các sản phẩm. Người dùng lựa chọn sản phẩm cần xóa. Sau khi người dùng xác nhận xóa dữ liệu trong form sẽ được gửi yêu cầu đến Controller xóa sản phẩm với phương thức là DELETE để thực hiện việc xóa sản phẩm trong cơ sở dữ liệu. Sau khi xóa dữ liệu thành công, trang danh sách sản phẩm sẽ được cập nhật lại các sản phẩm hiện tại.

3.5. Mô hình dữ liệu



Hình 3. 5 Mô hình dữ liệu

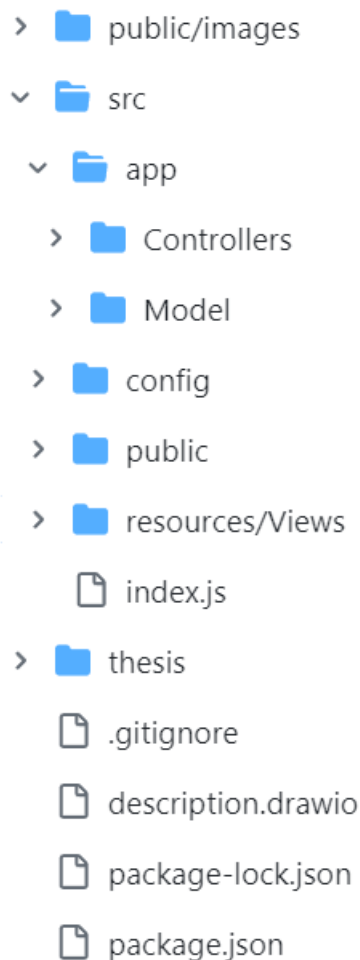
Collection “logins”: chứa thông tin người dùng bao gồm tên đăng nhập và mật khẩu, mỗi document trong bảng đại diện cho một người dùng.

Collection “products”: bảng chứa thông tin của sản phẩm như tên sản phẩm, thông tin chi tiết của sản phẩm, giá bán, hình ảnh và ngày tạo, ngày cập nhật sản phẩm. Mỗi document trong bảng đại diện cho một loại sản phẩm.

Collection “orders”: Bảng này liên kết thông tin từ bảng “products” cho biết sản phẩm này cho đơn hàng nào. Bảng chứa các thông tin bao gồm tên sản phẩm, số lượng, thông tin chi tiết người mua hàng, ngày tạo và ngày cập nhật.

Mỗi người dùng có thể đặt nhiều đơn hàng, nhưng mỗi đơn hàng chỉ thuộc về một người dùng. Mỗi sản phẩm có thể xuất hiện trong nhiều đơn hàng nhưng mỗi đơn hàng chỉ chứa một sản phẩm.

3.6. Cấu trúc dự án



Hình 3. 6 Cấu trúc dự án

Thư mục **public/images**: nơi lưu trữ hình ảnh được sử dụng trong dự án.

Thư mục **src/app**: chứa các thư mục con như:

Model: định nghĩa cấu trúc dữ liệu.

config: kết nối với cơ sở dữ liệu MongoDB.

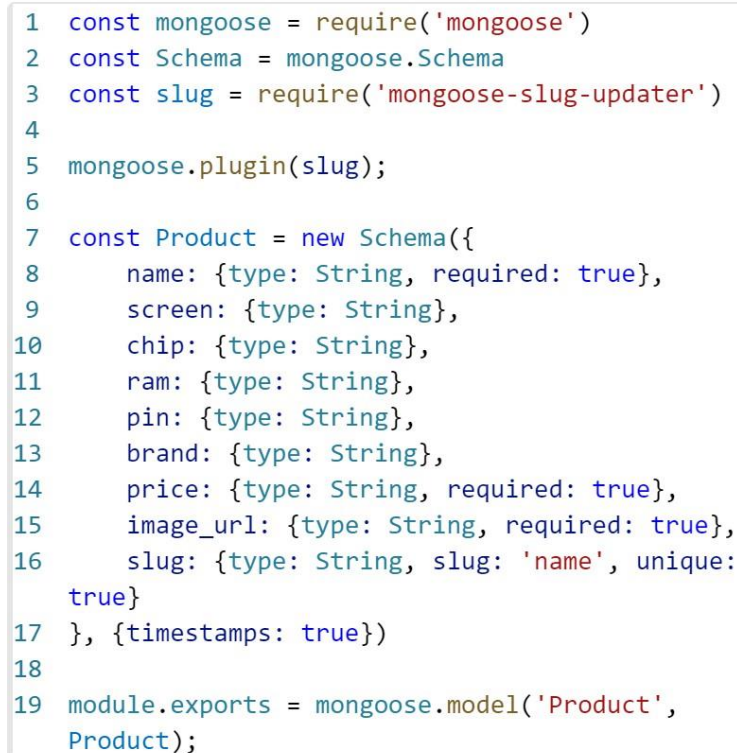
Views: hiển thị dữ liệu, giao diện cho người dùng.

Controller: Điều khiển sự tương tác giữa hai thành phần là **Model** và **Views**.

File **index.js**: Là nơi định nghĩa các route, khởi tạo các thư viện cần thiết cho dự án, cấu hình middleware, kết nối cơ sở dữ liệu.

3.7. Thành phần Model

3.7.1. Model Product




```
1 const mongoose = require('mongoose')
2 const Schema = mongoose.Schema
3 const slug = require('mongoose-slug-updater')
4
5 mongoose.plugin(slug);
6
7 const Product = new Schema({
8   name: {type: String, required: true},
9   screen: {type: String},
10  chip: {type: String},
11  ram: {type: String},
12  pin: {type: String},
13  brand: {type: String},
14  price: {type: String, required: true},
15  image_url: {type: String, required: true},
16  slug: {type: String, slug: 'name', unique:
    true}
17 }, {timestamps: true})
18
19 module.exports = mongoose.model('Product',
    Product);
```

Hình 3. 7 Model Product

Trong Model “**Product**” có các thư viện cần thiết như Mongoose giúp quản lý dữ liệu chặt chẽ hơn, đồng thời plugin mongoose-slug-updater giúp tạo và cập nhật các slug

tự động. Định nghĩa một Schema mới cho sản phẩm chứa các trường dữ liệu như name, screen, chip, ram, pin, brand, price, image_url, và slug với mỗi trường được định nghĩa kiểu dữ liệu và các thuộc tính khác nhau. Model này sẽ được sử dụng để tạo và truy vấn các document từ collection “**products**” trong MongoDB.

3.7.2. Model Login

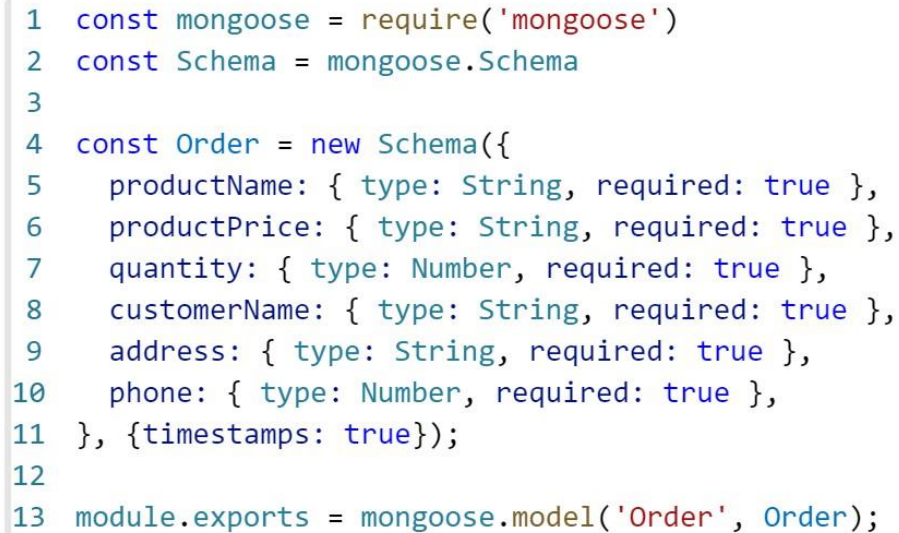
A screenshot of a code editor window with a light blue background. The editor has a white background and a title bar with three colored circles (red, yellow, green). The code is as follows:

```
1  const mongoose = require('mongoose')
2  const Schema = mongoose.Schema
3
4  const Login = new Schema({
5    username: { type: String, required: true },
6    password: { type: String, required: true },
7  })
8
9  module.exports = mongoose.model('Login', Login);
```

Hình 3. 8 Model Login

Trong Model “**Login**” sử dụng thư viện Mongoose để tạo ra một Schema cho việc đăng nhập, bao gồm username và password. Model này sẽ được sử dụng để tạo và truy vấn các document từ collection “**logins**” trong MongoDB.

3.7.3. Model Order



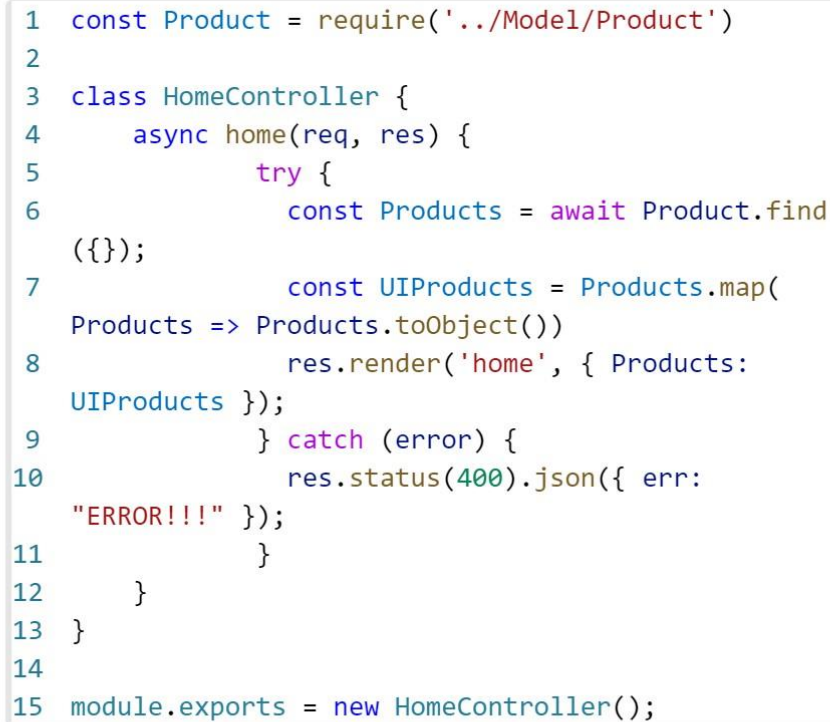
```
1  const mongoose = require('mongoose')
2  const Schema = mongoose.Schema
3
4  const Order = new Schema({
5    productName: { type: String, required: true },
6    productPrice: { type: String, required: true },
7    quantity: { type: Number, required: true },
8    customerName: { type: String, required: true },
9    address: { type: String, required: true },
10   phone: { type: Number, required: true },
11  }, {timestamps: true});
12
13  module.exports = mongoose.model('Order', Order);
```

Hình 3. 9 Model Order

Trong Model “**Order**” chứa các trường dữ liệu như productName, productPrice, quantity, customerName, address, phone với mỗi trường được định nghĩa các kiểu dữ liệu khác nhau. Model này sẽ được sử dụng để tạo và truy vấn các document từ collection “**orders**” trong MongoDB.

3.8. Thành phần Controller

3.8.1. Home Controller

A screenshot of a code editor window with a light blue background. The editor has a white background with three colored window control buttons (red, yellow, green) in the top left corner. The code is written in a light blue monospace font. It defines a class HomeController with an async home method. The method uses require to load a Product model, then uses Product.find to get products. It maps the products to UIProducts and renders the 'home' view with the products. A catch block handles errors by setting the status to 400 and returning an error message. Finally, it exports a new instance of the HomeController class.

```
1  const Product = require('../Model/Product')
2
3  class HomeController {
4    async home(req, res) {
5      try {
6        const Products = await Product.find
7          ({});
8        const UIProducts = Products.map(
9          Products => Products.toObject())
10       res.render('home', { Products:
11         UIProducts });
12     } catch (error) {
13       res.status(400).json({ err:
14         "ERROR!!!" });
15     }
16   }
17 }
18
19 module.exports = new HomeController();
```

Hình 3. 10 Home Controller

Định nghĩa một lớp HomeController với một phương thức **home** để xử lý yêu cầu HTTP tới trang chủ của ứng dụng.

Phương thức **home** này sẽ truy vấn cơ sở dữ liệu để lấy thông tin về tất cả các sản phẩm, sau đó gửi thông tin này tới trình duyệt để hiển thị trên trang chủ.

Nếu có lỗi xảy ra trong quá trình xử lý yêu cầu, ứng dụng sẽ trả về mã lỗi 400 và thông báo lỗi là “ERROR!!!”.

3.8.2. Product Controller

A code editor window with a light blue background and a white code area. The code is written in JavaScript/Node.js and defines a ProductController class. The code is as follows:

```
1  const Product = require('../Model/Product');
2
3  class ProductController {
4    async detail(req, res) {
5      try {
6        const product = await Product.findOne({ slug
7          : req.params.slug })
8        res.render('detail', { productdetail:
9          product.toObject() })
10       } catch (error) {
11         res.status(400).json({ err: "Error!!!" })
12       }
13     }
14   }
15   module.exports = new ProductController();
```

Hình 3. 11 Product Controller

Một lớp ProductController với một phương thức **detail** được tạo ra trong đoạn mã trên. Phương thức **detail** nhận vào hai tham số: req và res. req là đối tượng yêu cầu, chứa thông tin về yêu cầu HTTP gửi từ trình duyệt, bao gồm URL, phương thức HTTP (GET, POST, v.v.), và bất kỳ dữ liệu nào được gửi cùng với yêu cầu. res là đối tượng phản hồi, được sử dụng để gửi dữ liệu trở lại cho trình duyệt.

Trong phương thức **detail**, đầu tiên, mã thực hiện một truy vấn cơ sở dữ liệu để tìm một sản phẩm có slug trùng với slug trong yêu cầu từ client. slug thường là một chuỗi không dấu được tạo từ tên của sản phẩm, được sử dụng trong URL để xác định sản phẩm cụ thể. Kết quả của truy vấn cơ sở dữ liệu được lưu vào biến product.

Tiếp theo, mã gửi một trang HTML tới client. Trang này được render từ template có tên 'detail', với dữ liệu là { productdetail: product.toObject() }. Phương thức toObject

được sử dụng để chuyển đổi sản phẩm từ dạng Mongoose document sang dạng JavaScript object thông thường.

3.8.3. Order Controller

A screenshot of a code editor window with a light blue background. The editor has three colored window control buttons (red, yellow, green) in the top left corner. The code is written in a light blue monospace font on a white background. It defines a class OrderController with an async method order that takes req and res as arguments. Inside the order method, a try block creates a new Order object from req.body, saves it, and catches any errors to return a 400 status with an error message. The class is then instantiated and exported as the module's default export.

```
1  const Order = require('../Model/Order');
2
3  class OrderController {
4    // POST /order
5    async order(req, res) {
6      try {
7        const order = new Order(req.body);
8        await order.save();
9      } catch (error) {
10       res.status(400).json({ err: "ERROR!!!" });
11     }
12   }
13 }
14
15 module.exports = new OrderController();
```

Hình 3. 12 Order Controller

Khởi tạo một lớp OrderController với một phương thức order. Phương thức order nhận vào hai tham số: req và res. req là đối tượng yêu cầu, chứa thông tin về yêu cầu HTTP gửi từ trình duyệt, bao gồm URL, phương thức HTTP, và bất kỳ dữ liệu nào được gửi cùng với yêu cầu. res là đối tượng phản hồi, được sử dụng để gửi dữ liệu trở lại cho trình duyệt. Trong phương thức order tạo một đối tượng order mới từ model Order với dữ liệu từ req.body bằng phương thức POST. Sau đó lưu đối tượng order vào cơ sở dữ liệu bằng phương thức save của Mongoose.

3.9. Thành phần View

3.9.1. View Home



Hình 3. 13 View Home

Tạo ra một container và một hàng sử dụng Bootstrap, để căn chỉnh giao diện trang web. Sử dụng Handlebars để lặp qua từng phần tử trong đối tượng "Products". Tạo ra một card sản phẩm với kích thước được định nghĩa để hiển thị trong lưới của Bootstrap. Chiều cao của card được đặt là 100%. Hiển thị hình ảnh sản phẩm bên trong một thẻ `<a>` để tạo liên kết đến trang chi tiết sản phẩm. Hiển thị tên và giá sản phẩm bên trong phần thân của card.

3.9.2. View Order

```

1 <div class="container mt-5">
2   <h2 class="mb-4">Đặt hàng sản phẩm</h2>
3   <form method="POST" action="/order">
4     <div class="form-group">
5       <label for="productName">Tên sản phẩm:</label>
6       <input type="text" class="form-control readonly-input" name="productName" value="{{
Product.name}}" readonly>
7     </div>
8     <div class="form-group">
9       <label for="productPrice">Giá sản phẩm</label>
10      <input type="text" id="formatMoney" class="form-control readonly-input money" name=
"productPrice" value="{{Product.price}}đ /1 sản phẩm" readonly>
11    </div>
12    <div class="form-group">
13      <label for="quantity">Số lượng:</label>
14      <input type="number" class="form-control" name="quantity" value="1" min="1" required>
15    </div>
16    <div class="form-group">
17      <label for="customerName">Tên khách hàng:</label>
18      <input type="text" class="form-control" name="customerName" required>
19    </div>
20    <div class="form-group">
21      <label for="address">Địa chỉ giao hàng:</label>
22      <input type="text" class="form-control" name="address" required >
23    </div>
24    <div class="form-group">
25      <label for="phone">Số điện thoại:</label>
26      <input type="tel" class="form-control" name="phone" required>
27    </div>
28    <button type="submit" class="btn btn-primary" data-toggle="modal" data-target="#buy_prod">Đặt hàng
29  </button>
30 </form>
31 </div>
32
33 <div id="buy_prod" class="modal" tabindex="-1" role="dialog">
34   <div class="modal-dialog" role="document">
35     <div class="modal-content">
36       <div class="modal-header">
37         <h5 class="modal-title">THÔNG BÁO</h5>
38         <button type="button" class="close" data-dismiss="modal" aria-label="Close">
39           <span aria-hidden="true">&times;</span>
40         </button>
41       </div>
42       <div class="modal-body">
43         <p>Bạn đã đặt hàng thành công!</p>
44       </div>
45       <div class="modal-footer">
46         <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
47       </div>
48     </div>
49   </div>
50 </div>

```

Hình 3. 14 View Order

Form đặt hàng với các trường nhập thông tin như tên sản phẩm, giá, số lượng, tên khách hàng, địa chỉ giao hàng và số điện thoại. Khi người dùng nhấn nút "Đặt hàng", dữ liệu sẽ được gửi đến địa chỉ "/order", và một modal thông báo sẽ xuất hiện với thông

điệp "Bạn đã đặt hàng thành công!". Sử dụng Bootstrap để tạo giao diện thân thiện và thuận tiện cho người dùng.

3.9.3. View Search



Hình 3. 15 View Search

Một container Bootstrap với một tiêu đề h3 hiển thị kết quả tìm kiếm dựa trên từ khóa ("{{keyword}}"). Danh sách sản phẩm hiển thị trong các card Bootstrap, mỗi card chứa hình ảnh, tên sản phẩm, giá và liên kết đến trang chi tiết sản phẩm ("/Product/{{this.slug}}").

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1. Hướng dẫn cài đặt

Sao chép đề tài từ Github: mở terminal hoặc command prompt, di chuyển đến thư mục muốn sao chép dự án vào và chạy lệnh:

```
git clone https://github.com/NHKhanq/csn-da21ttb-NguyenHoangKhang-webbanhang-nodejs
```

Di chuyển vào thư mục mới tạo bằng lệnh:

```
cd csn-da21ttb-NguyenHoangKhang-webbanhang-nodejs
```

Cài đặt các dependencies sử dụng npm bằng lệnh:

```
npm install
```

Cài đặt MongoDB Compass và tạo 3 collection lần lượt là logins, products và orders.

Sau khi cài đặt xong, chạy ứng dụng bằng lệnh:

```
npm start
```

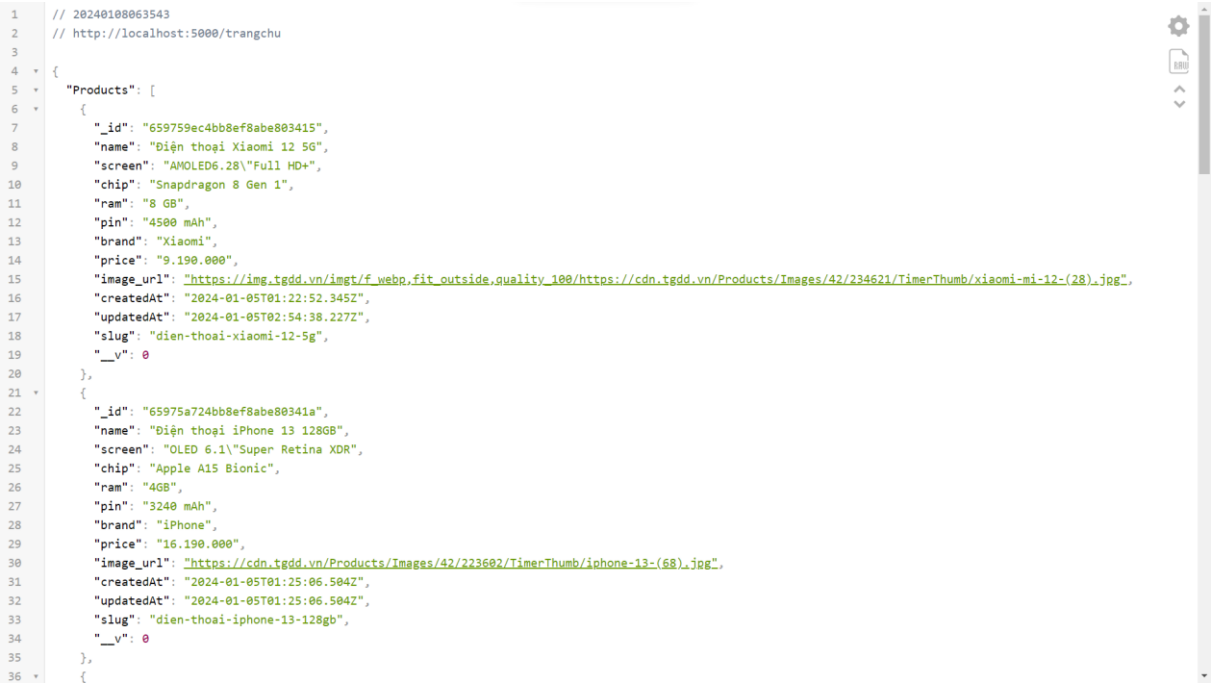
Mở trình duyệt và truy cập <http://localhost:5000> để xem ứng dụng.

4.2. Kết quả thử nghiệm

Đề tài được triển khai ở máy chủ localhost:5000 với cơ sở dữ liệu MongoDB được kết nối. Kết quả runtime của đề tài dưới 1s. Đáp ứng được các yêu cầu cơ bản của một trang web bán hàng. Hỗ trợ được việc xem, chỉnh sửa và tìm kiếm sản phẩm.

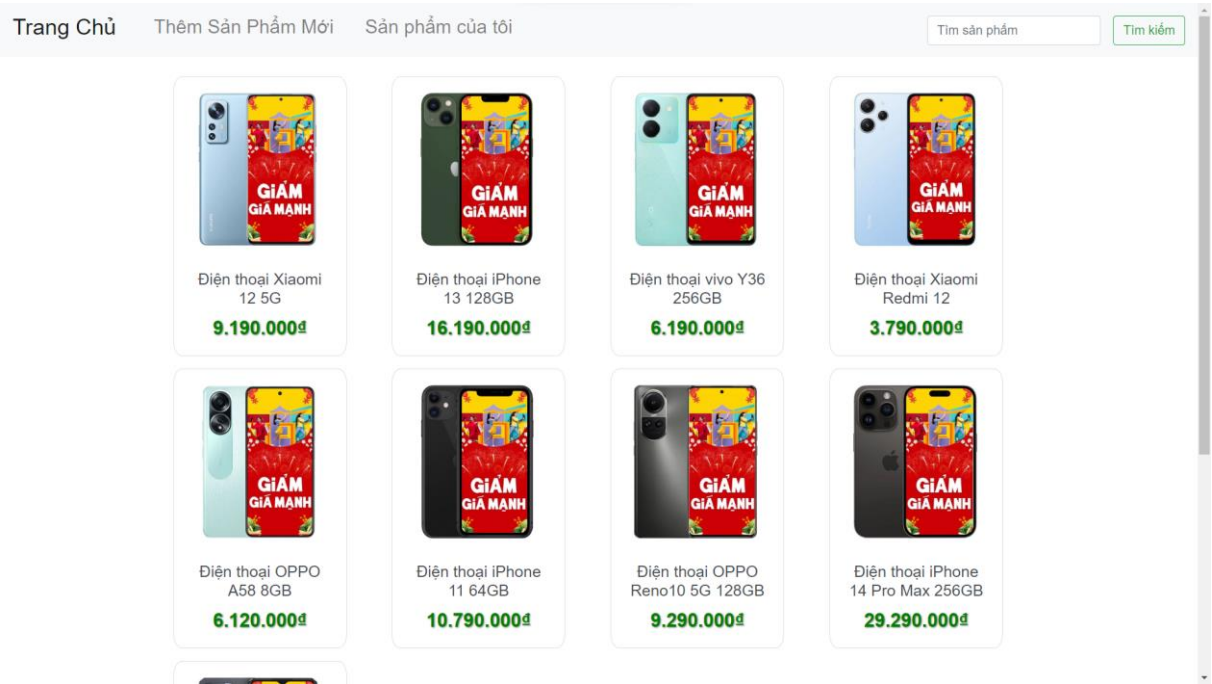
4.3. Cách hoạt động của JSON

Đầu tiên, người dùng sẽ tương tác với View bằng cách điền thông tin vào các form để yêu cầu đến Controller tương ứng. Nếu yêu cầu đó liên quan đến việc lấy dữ liệu, Controller sẽ gọi đến Model. Model nhận thông tin từ Controller, thực thi các yêu cầu và trả kết quả về lại cho Controller được biểu diễn dưới dạng đối tượng và chuyển đổi thành dạng JSON. Controller nhận được các đối tượng JSON sẽ gửi về View, sau khi View nhận được dữ liệu dạng JSON từ Controller nó sẽ hiển thị dữ liệu này lên giao diện người dùng.



Hình 4. 1 Chuỗi JSON được trả về View

4.4. Giao diện trang Home



Hình 4. 2 Giao diện trang Home

4.5. Giao diện trang thêm sản phẩm

[Trang Chủ](#) [Thêm Sản Phẩm Mới](#) [Sản phẩm của tôi](#)

Thêm Sản Phẩm

Tên Sản Phẩm:

Màn Hình: Pin:

Chip: Hãng:

Hình ảnh: RAM:

Giá:


©110121035 - Nguyễn Hoàng Khang - ĐỒ ÁN CƠ SỞ NGÀNH.

Hình 4. 3 Giao diện thêm sản phẩm


4.6. Giao diện trang tìm kiếm sản phẩm

[Trang Chủ](#) [Thêm Sản Phẩm Mới](#) [Sản phẩm của tôi](#)


Kết quả tìm kiếm cho "iphone"



Điện thoại iPhone
13 128GB
16.190.000đ



Điện thoại iPhone
11 64GB
10.790.000đ



Điện thoại iPhone
14 Pro Max 256GB
29.290.000đ

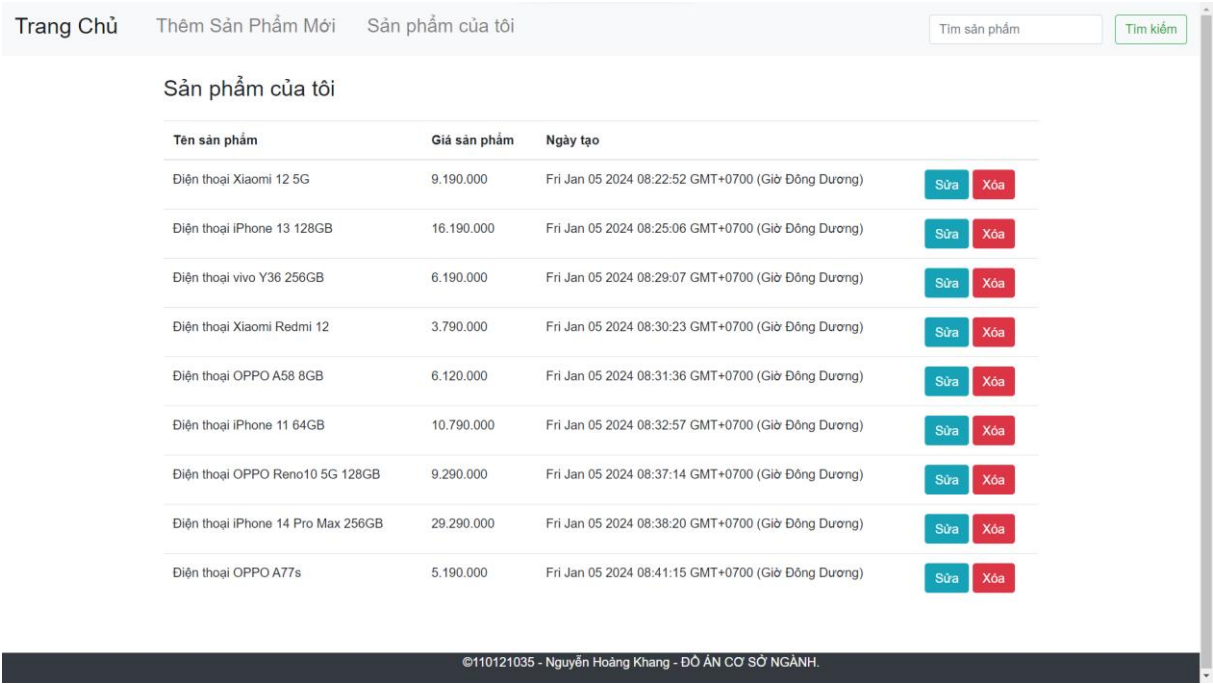
©110121035 - Nguyễn Hoàng Khang - ĐỒ ÁN CƠ SỞ NGÀNH.

Hình 4. 4 Giao diện tìm kiếm sản phẩm

Nguyễn Hoàng Khang

37

4.7. Giao diện trang cập nhật sản phẩm



Hình 4. 5 Giao diện cập nhật sản phẩm

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Đề tài “Nghiên cứu NodeJS và xây dựng ứng dụng bán hàng trực tuyến” đã xây dựng được một trang web có các chức năng cơ bản như hiển thị sản phẩm, thêm, xóa, sửa và tìm kiếm sản phẩm theo mô hình MVC, làm tiền đề để phát triển và mở rộng dự án trong tương lai. Trang web cũng đã áp dụng một số công nghệ mới như NodeJS, Express, MongoDB, Bootstrap để tăng hiệu năng và giao diện thân thiện với người dùng.

5.2. Hạn chế

Bên cạnh những mặt đạt được, vẫn tồn đọng nhiều hạn chế như:

Chức năng đăng nhập vẫn chưa hoàn thiện, vẫn chưa phân quyền cho người dùng.

Chưa xây dựng được chức năng giỏ hàng.

Hệ thống còn đơn giản chưa đáp ứng được người dùng.

5.3. Hướng phát triển

Phát triển hoàn thiện chức năng đăng nhập và phân quyền cho người dùng.

Xây dựng chức năng thêm sản phẩm vào giỏ hàng.

Cần phát triển hệ thống để đáp ứng nhu cầu của người dùng.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] Ajit Singh, Sultan Ahmad, MongoDB Simply In Depth, 2019.
- [2] R. Carlisle, The Nodejs Handbook, 2016.
- [3] S. Dang, "F8 - Học Lập Trình Để Đi Làm," 04 03 2022. [Online]. Available: <https://fullstack.edu.vn/>.
- [4] E. Brown, Web Development with Node & Express, 2014.
- [5] P. P. Ltd, Mastering Node.js, November 2013, 2012.

PHỤ LỤC

BẢNG MÔ TẢ CHI TIẾT CHO CHUỖI JSON MẪU

| Chuỗi JSON | Chi tiết dữ liệu |
|---|--|
| "_id": "659759ec4bb8ef8abe803415", | Mã số duy nhất của sản phẩm. |
| "name": "Điện thoại Xiaomi 12 5G", | Tên sản phẩm (Điện thoại Xiaomi 12 5G). |
| "screen": "AMOLED6.28 "Full HD+", | Mô tả màn hình của điện thoại |
| "chip": "Snapdragon 8 Gen 1", | Mô tả chip xử lý |
| "pin": "4500 mAh", | Mô tả dung lượng pin cho sản phẩm |
| "brand": "Xiaomi", | Thương hiệu sản phẩm (Xiaomi). |
| "price": "9.190.000", | Giá của sản phẩm (9.190.000 VND). |
| "image_url": "https://img.tgdd.vn/Images/xiaomi-mi-12-.jpg", | URL hình ảnh sản phẩm. |
| "createdAt": "2024-01-05T01:22:52.345Z", | Ngày và giờ tạo (5 tháng 1 năm 2024, 01:22:52) |
| "updatedAt": "2024-01-05T02:54:38.227Z", | Ngày và giờ cập nhật (5 tháng 1 năm 2024, 02:54:38) |
| "__v": 0 | Phiên bản của tài liệu (version) |