**BỘ CÔNG THƯƠNG**



**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙜🙢🙠🙞



**ĐỒ ÁN MÔN HỌC**

**ĐỀ TÀI: XÂY DỰNG WEBSITE BÁN HÀNG ĐIỆN TỬ THEO CÔNG NGHỆ MÃ NGUỒN MỞ**

**(NODEJS + MONGODB)**

Giảng viên hướng dẫn :Nguyễn Thị Bích Ngân

Vũ Đình Ái

Sinh viên thực hiện :

1. 2001150151 – Hồ Minh Tiến

2. 2001150152 – Trần Anh Khoa

3. 2001150224 – Lưu Quang Bình

TP. HỒ CHÍ MINH – 2018

**NHẬN XÉT CỦA GIÁO VIÊN**

**LỜI CẢM ƠN**

Đầu tiên cho phép chúng em được gửi lời cảm ơn chân thành và sâu sắc nhất đến quý thầy cô giáo khoa Công Nghệ Thông Tin, trường Đại Học Công Nghiệp Thực Phẩm thành phố Hồ Chí Minh, đặc biệt là thầy Vũ Đình Ái và cô Nguyễn Thị Bích Ngân đã tận tình chỉ dạy, quan tâm, hướng dẫn và góp ý cho chúng em trong suốt quá trình thực hiện và hoàn thành môn học của mình.

Nhóm em xin chân thành cảm ơn.

Tp Hồ Chí Minh, tháng 11 năm 2018

MỤC LỤC

[PHẦN 1. GIỚI THIỆU NODEJS VÀ MONGODB 1](#_Toc530844894)

[CHƯƠNG 1 TỔNG QUAN VỀ NODEJS 1](#_Toc530844895)

[1.1. NODEJS LÀ GÌ? 1](#_Toc530844896)

[1.2. CÁC GÓI CÀI ĐẶT PHỔ BIẾN CỦA NODEJS 1](#_Toc530844897)

[1.3. CÁC IDE NÊN DÙNG KHI LÀM VIỆC VỚI NODEJS 9](#_Toc530844898)

[1.4. CÁC CẤU TRÚC LỆNH CƠ BẢN TRONG NODEJS 10](#_Toc530844899)

[1.5. FUNCTION VÀ TỪ KHÓA KHAI BÁO TRONG JAVASCRIPT ES6 14](#_Toc530844900)

[1.5.1. Từ khóa let và const 14](#_Toc530844901)

[1.5.2. Arrow function trong ES6 16](#_Toc530844902)

[1.6. CALLBACK VÀ PROMISE 17](#_Toc530844903)

[1.6.1. Đồng bộ và bất đồng bộ 17](#_Toc530844904)

[1.6.2. Callback 18](#_Toc530844905)

[1.6.3. Promise 19](#_Toc530844906)

[1.7. KẾT CHƯƠNG 20](#_Toc530844907)

[1.7.1. Ưu điểm 20](#_Toc530844908)

[1.7.2. Nhược điểm 20](#_Toc530844909)

[CHƯƠNG 2 TỔNG QUAN VỀ MONGODB 22](#_Toc530844910)

[2.1. GIỚI THIỆU VỀ MONGODB 22](#_Toc530844913)

[2.1.1. Nosql là gì? 22](#_Toc530844914)

[2.1.2. MongoDB là gì? 22](#_Toc530844915)

[2.1.3. Các kiểu dữ liệu trong MongoDB 22](#_Toc530844916)

[2.2. SO SÁNH MONGODB VỚI CƠ SỞ DỮ LIỆU QUAN HỆ 24](#_Toc530844917)

[2.3. NGUYÊN TẮC HOẠT ĐỘNG CỦA MONGODB 28](#_Toc530844918)

[2.4. SỬ DỤNG MONGODB 29](#_Toc530844919)

[2.5. THAO TÁC VỚI MONGODB TRONG NODEJS 35](#_Toc530844920)

[2.6. KẾT CHƯƠNG 37](#_Toc530844921)

[2.6.1. Ưu điểm 37](#_Toc530844922)

[2.6.2. Nhược điểm 37](#_Toc530844923)

[PHẦN 2. PHÂN TÍCH HỆ THỐNG BÁN HÀNG DIỆN TỬ 38](#_Toc530844924)

[CHƯƠNG 1 KHẢO SÁT HỆ THỐNG 38](#_Toc530844925)

[1.1. MÔ TẢ HỆ THỐNG HIỆN TẠI 38](#_Toc530844926)

[1.2. YÊU CẦU HỆ THỐNG 38](#_Toc530844927)

[1.3. YÊU CẦU CHỨC NĂNG 39](#_Toc530844928)

[1.3.1. Đặt hàng 39](#_Toc530844929)

[1.3.2. Tìm kiếm sản phẩm 40](#_Toc530844930)

[1.3.3. Thanh toán 40](#_Toc530844931)

[1.3.4. Tra cứu 40](#_Toc530844932)

[1.3.5. Nhập hàng 40](#_Toc530844933)

[1.3.6. Quản lý 40](#_Toc530844934)

[1.3.7. Thống kê và in ấn 40](#_Toc530844935)

[1.3.8. Tổng kết 41](#_Toc530844936)

[1.4. QUY TRÌNH NGHIỆP VỤ 42](#_Toc530844937)

[CHƯƠNG 2 PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 44](#_Toc530844938)

[2.1. MÔ HÌNH XỬ LÝ NGHIỆP VỤ BPM 44](#_Toc530844941)

[2.1.1. Đặt hàng 44](#_Toc530844942)

[2.1.2. Nhập hàng 46](#_Toc530844943)

[2.1.3. Quản lý 48](#_Toc530844944)

[2.2. MÔ HÌNH USECASE 50](#_Toc530844945)

[2.3. Mô hình thực thể kết hợp ERD 51](#_Toc530844946)

[2.4. Mô hình cơ sở dữ liệu quan hệ CDM 52](#_Toc530844947)

[2.5. Mô hình dữ liệu vật lý PDM 53](#_Toc530844948)

[2.6. CÁC RÀNG BUỘC TOÀN VẸN 54](#_Toc530844949)

[2.6.1. Ràng buộc toàn vẹn liên bộ 54](#_Toc530844950)

[2.6.2. Ràng buộc toàn vẹn về miền giá trị 54](#_Toc530844951)

[2.6.3. Ràng buộc toàn vẹn liên thuộc tính 57](#_Toc530844952)

[2.6.4. Ràng buộc toàn vẹn liên bộ - liên quan hệ 57](#_Toc530844953)

[2.6.5. Ràng buộc toàn vẹn do thuộc tính tổng hợp 58](#_Toc530844954)

[2.7. MÔ HÌNH CHỨC NĂNG BFD 59](#_Toc530844955)

[2.8. MÔ HÌNH XỬ LÝ DFD 60](#_Toc530844956)

[CHƯƠNG 3 TRIỂN KHAI HỆ THỐNG WEBSITE NODEJS 65](#_Toc530844957)

[3.1. TẠO PROJECT 65](#_Toc530844959)

[3.2. TẠO VÀ CÁT GIAO DIỆN HTML THÀNH EJS TEMPLATE 65](#_Toc530844960)

[3.3. TẠO CƠ SỞ DỮ LIỆU MONGODB 67](#_Toc530844961)

[3.4. KẾT NỐI NODEJS VỚI MONGODB 69](#_Toc530844962)

[3.5. KẾT QUẢ THỰC HIỆN 71](#_Toc530844963)

[KẾT LUẬN 75](#_Toc530844964)

[TÀI LIỆU THAM KHẢO 76](#_Toc530844965)

**MỤC LỤC HÌNH ẢNH**

[Hình 1.1.1 Tạo project 2](#_Toc530844805)

[Hình 1.1.2 Nội dung file package.json 3](#_Toc530844806)

[Hình 1.1.3 Lệnh cài Nodemon 3](#_Toc530844807)

[Hình 1.1.4 Lệnh cài đặt Express 4](#_Toc530844808)

[Hình 1.1.5 Lệnh cài đặt Express generator 4](#_Toc530844809)

[Hình 1.1.6 thư mục trong express-generator 4](#_Toc530844810)

[Hình 1.1.7 Tạo project sử dụng EJS template 5](#_Toc530844811)

[Hình 1.1.8 Thư mục EJS template 5](#_Toc530844812)

[Hình 1.1.9 npm install 5](#_Toc530844813)

[Hình 1.1.10 Thư mục project NodeJS 6](#_Toc530844814)

[Hình 1.1.11 Lệnh chạy server 6](#_Toc530844815)

[Hình 1.1.12 Chạy web nodejs trên trình duyệt 6](#_Toc530844816)

[Hình 1.1.13 Lệnh cài đặt Mongoose 7](#_Toc530844817)

[Hình 1.1.14 Lệnh cài đặt body-parser 7](#_Toc530844818)

[Hình 1.1.15 Lệnh cài đặt EJS 8](#_Toc530844819)

[Hình 1.1.16 Lệnh cài đặt multer 8](#_Toc530844820)

[Hình 1.1.17 Lệnh cài passport.js 9](#_Toc530844821)

[Hình 1.1.18 Lệnh cài đặt nodemailer 9](#_Toc530844822)

[Hình 1.1.19 Sử dụng terminal mở kết nối 13](#_Toc530844823)

[Hình 1.1.20 Ứng dụng helloword 14](#_Toc530844824)

[Hình 1.1.21 ví dụ từ khóa var 14](#_Toc530844825)

[Hình 1.1.22 kết quả chạy từ khóa var 15](#_Toc530844826)

[Hình 1.1.23 kết quả chạy từ khóa let 15](#_Toc530844827)

[Hình 1.1.24 kết quả chạy từ khóa const 16](#_Toc530844828)

[Hình 1.1.25 Ví dụ bất đồng bộ 18](#_Toc530844829)

[Hình 1.1.26 Ví dụ callback 18](#_Toc530844830)

[Hình 1.1.27 Ví dụ promise 20](#_Toc530844831)

[Hình 1.2.1 Mô hình cơ sở dữ liệu mysql 24](#_Toc530844832)

[Hình 1.2.2 Cách hoạt động của MongoDB 28](#_Toc530844833)

[Hình 1.2.3 Lệnh hiện tấ cả database 29](#_Toc530844834)

[Hình 1.2.4 Lệnh chọn database 29](#_Toc530844835)

[Hình 1.2.5 Lệnh tạo collection 30](#_Toc530844836)

[Hình 1.2.6 Lệnh tìm tất cả document 30](#_Toc530844837)

[Hình 1.2.7 Lệnh thêm một document 30](#_Toc530844838)

[Hình 1.2.8 Lệnh thêm nhiều document 30](#_Toc530844839)

[Hình 1.2.9 Lệnh update document 31](#_Toc530844840)

[Hình 1.2.10 Lệnh update không thay đổi thuộc tính khác 31](#_Toc530844841)

[Hình 1.2.11 Lệnh xóa document 31](#_Toc530844842)

[Hình 1.2.12 So sánh bằng 32](#_Toc530844843)

[Hình 1.2.13 So sánh nhỏ hơn 32](#_Toc530844844)

[Hình 1.2.14 So sánh nhỏ hơn bằng 32](#_Toc530844845)

[Hình 1.2.15 So sánh lớn hơn 32](#_Toc530844846)

[Hình 1.2.16 So sánh lớn hơn bằng 33](#_Toc530844847)

[Hình 1.2.17 So sánh khác 33](#_Toc530844848)

[Hình 1.2.18 Phép truy vấn 'in' 33](#_Toc530844849)

[Hình 1.2.19 Phép truy vấn 'not in' 33](#_Toc530844850)

[Hình 1.2.20 Phép 'and' 34](#_Toc530844851)

[Hình 1.2.21 Phép 'or' 34](#_Toc530844852)

[Hình 1.2.22 Phép 'or' và 'and' 34](#_Toc530844853)

[Hình 1.2.23 Lấy tên, hiện cả id 35](#_Toc530844854)

[Hình 1.2.24 Lấy tên, không hiện id 35](#_Toc530844855)

[Hình 1. 2.25 Lệnh kết nối mongodb trong nodejs 35](#_Toc530844856)

[Hình 1.2.26 Thao tác insert 35](#_Toc530844857)

[Hình 1.2.27 Thao tác update 36](#_Toc530844858)

[Hình 1.2.28 Thao tác delete 36](#_Toc530844859)

[Hình 1.2.29 Thao tác tìm kiếm 36](#_Toc530844860)

[Hình 2.2.30 BPM đặt hàng 45](#_Toc530844861)

[Hình 2.2.31 BPM nhập hàng 47](#_Toc530844862)

[Hình 2.2.32 BPM quản lý 49](#_Toc530844863)

[Hình 2.2.33 Mô hình usecase 50](#_Toc530844864)

[Hình 2.2.34 Mô hình ERD 51](#_Toc530844865)

[Hình 2.2.35 Mô hình CDM 52](#_Toc530844866)

[Hình 2.2.36 Mô hình PDM 53](#_Toc530844867)

[Hình 2.2.37 Mô hình BFD 59](#_Toc530844868)

[Hình 2.2.38 DFD đỉnh 60](#_Toc530844869)

[Hình 2.2.39 DFD mức 1 60](#_Toc530844870)

[Hình 2.2.40 DFD đặt hàng 61](#_Toc530844871)

[Hình 2.2.41 DFD thanh toán 62](#_Toc530844872)

[Hình 2.2.42 DFD thống kê 62](#_Toc530844873)

[Hình 2.2.43 DFD tra cứu 63](#_Toc530844874)

[Hình 2.2.44 DFD quản lý 64](#_Toc530844875)

[Hình 2.2.45 DFD nhập hàng 64](#_Toc530844876)

[Hình 2.2.46 Tạo project NodeJS 65](#_Toc530844877)

[Hình 2.2.47 Trang html sau khi chia nhỏ 66](#_Toc530844878)

[Hình 2.2.48 câu lệnh include trong file .ejs 66](#_Toc530844879)

[Hình 2.2.49 Tạo route cho trang giới thiệu và liên hệ 67](#_Toc530844880)

[Hình 2.2.50 Thiết lập route trong file app.js 67](#_Toc530844881)

[Hình 2.2.51 Giao diện tạo database 68](#_Toc530844882)

[Hình 2.2.52 Giao diện tạo collection 69](#_Toc530844883)

[Hình 2.2.53 Giao diện thêm document 69](#_Toc530844884)

[Hình 2.2.54 Lấy dữ liệu bằng MongoClient 70](#_Toc530844885)

[Hình 2.2.55 Tạo schema collection san\_pham 70](#_Toc530844886)

[Hình 2.2.56 Lấy dữ liệu bằng mongoose 71](#_Toc530844887)

[Hình 2.2.57 Giao diện trang chủ 71](#_Toc530844888)

[Hình 2.2.58 Giao diện sản phẩm theo từng loại 72](#_Toc530844889)

[Hình 2.2.59 Giao diện chi tiết sản phẩm 72](#_Toc530844890)

[Hình 2.2.60 Giao diện đăng ký 73](#_Toc530844891)

[Hình 2.2.61 Giao diện đăng nhập 73](#_Toc530844892)

[Hình 2.2.62 Giao diện giỏ hàng 74](#_Toc530844893)

**MỤC LUC BIỂU MẪU**

[Biểu mẫu 2.1.1 Biểu mẫu đặt hàng 39](#_Toc530739372)

[Biểu mẫu 2.1.2 Biểu mẫu hóa đơn 40](#_Toc530739373)

[Biểu mẫu 2.1.3 Biểu mẫu thống kê sản phẩm 41](#_Toc530739374)

[Biểu mẫu 2.1.4 Biểu mẫu thống kê hóa đơn 41](#_Toc530739375)

**MỞ ĐẦU**

Mã nguồn mở là những phần mềm được cung cấp dưới cả dạng mã và nguồn, không chỉ là miễn phí về giá mua mà chủ yếu là miễn phí về bản quyền, người dùng có quyền sửa đổi, cải tiến, phát triển mà không cần xin phép ai, điều mà họ không được phép làm đối với các phần mềm nguồn đóng (tức là phần mềm thương mại).

Tiện ích mà Open Source mang lại chính là quyền tự do sử dụng chương trình cho mọi mục đích, quyền tự do để nghiên cứu cấu trúc của chương trình, chỉnh sửa phù hợp với nhu cầu, truy cập vào mã nguồn, quyền tự do phân phối lại các phiên bản cho nhiều người, quyền tự do cải tiến chương trình và phát hành những bản cải tiến vì mục đích công cộng. Phần mềm có thể được sao chép hoàn toàn miễn phí. Các định dạng ﬁle không hoàn toàn bị kiểm soát bởi một vài nhà cung cấp. Hầu hết các sản phẩm Open Source đều có khả năng bảo mật tuyệt vời, khi một vết nứt được tìm thấy, nó thường được trám nhanh hơn phần mềm có bản quyền vì một cộng đồng hỗ trợ lớn và không bị phụ thuộc vào một công ty nào.

Chính vì những tiện ích đó mà nhóm quyết định tìm hiểu và áp dụng mã nguồn mở vào hệ thống quản lý bán hàng diện tử, cụ thể là sử dụng NodeJS và MongoDB. Đề tài gồm hai phần chính:

* Phần 1: Giới thiệu NodeJS và MongoDB
* Phần 2: Phân tích hệ thống website bán hàng điện tử

## GIỚI THIỆU NODEJS VÀ MONGODB

# TỔNG QUAN VỀ NODEJS

### NODEJS LÀ GÌ?

NodeJS là một phần mềm mã nguồn mở được viết dựa trên ngôn ngữ JavaScript cho phép lập trình viên có thể xây dựng các ứng dụng chạy trên máy chủ.

**JavaScript** là một ngôn ngữ dựa trên sự kiện, vì vậy bất cứ thứ gì xảy ra trên server đều tạo ra một sự kiện. Mỗi kết nối mới sinh ra một sự kiện; dữ liệu nhận được từ một upload form sẽ sinh ra một sự kiện, việc truy vấn dữ liệu từ database cũng sinh ra một sự kiện. Trong thực tế, điều này có nghĩa là một trang web NodeJS sẽ chẳng bao giờ bị khóa và có thể hỗ trợ cho hàng chục nghìn user truy cập cùng lúc.

Ban đầu, NodeJS được phát triển bởi Ryan Dahl. Phiên bản đầu tiên của NodeJS được cho ra mắt vào năm 2009, là một nền tảng chạy trên môi trường Engine JavaScript (V8 Engine). NodeJS đóng vai trò của server - Apache và thông dịch mã ứng dụng chạy trên nó. Giống như Apache, có rất nhiều module (thư viện) có thể được cài đặt để bổ sung thêm các đặc trưng và chức năng - như lưu trữ dữ liệu, hỗ trợ file Zip, đăng nhập bằng Facebook, hoặc các cổng thanh toán. Dĩ nhiên, nó không có nhiều thư viện như PHP, nhưng NodeJS vẫn đang ở trong giai đoạn ban đầu và có một cộng đồng rất mạnh mẽ ở đằng sau nó.

Trong một môi trường server điển hình mỗi một kết nối tới server sẽ sinh ra một luồng mới, và điều này khiến ứng dụng nhanh chóng trở nên chậm chạp hoặc quá tải - cách duy nhất để hỗ trợ nhiều người dùng hơn là bằng cách bổ sung thêm nhiều máy chủ. Đơn giản là nó không có khả năng mở rộng tốt. Nhưng với NodeJS thì điều này không phải là vấn đề.

### CÁC GÓI CÀI ĐẶT PHỔ BIẾN CỦA NODEJS

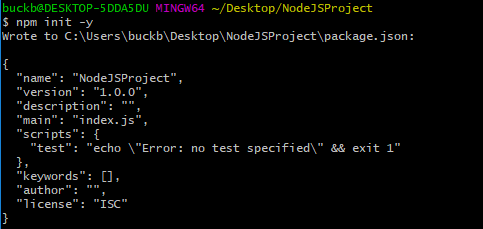
* NPM

NPM viết tắt của Node Package Manager là một công cụ (chương trình) quản lý các thư viện lập trình Javascript cho NodeJS, công cụ này là thật sự cần thiết cho thế giới mã nguồn mở. Trong cộng đồng Javascript, các lập trình viên chia sẻ hàng trăm nghìn các đoạn code giúp cho các dự án mới tránh phải viết lại các thành phần cơ bản, các thư viện lập trình hay thậm chí cả các framework. Mỗi đoạn code này có thể phụ thuộc vào rất nhiều các mã nguồn mở khác, và NPM là công cụ để giúp quản lý các thư viện đó. Hiện tại NPM được đóng gói sẵn cùng với NodeJS từ phiên bản v0.6.3 nên chúng ta không cần phải cái đặt nó.

* Init

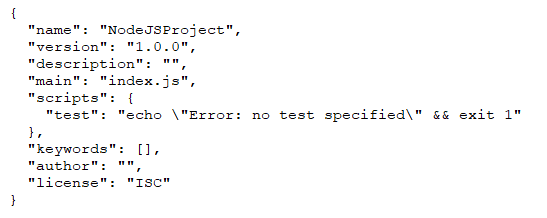
Để có thể bắt đầu làm việc với một project NodeJS, ta cần phải khởi tạo một project với các thông số cần thiết. Để tạo project ta có thể dùng một trong hai lệnh sau: *npm init* sau đó tự nhập các thông số cấu hình theo ý muốn hoặc *npm int –y* để dùng cấu hình mặc định. Các thông số khi tạo project

* **name**: Tên của Project
* **version**: Version của Project
* **description**: Mô tả cho Project
* **entry point**: File chạy chính (chạy đầu tiên) của Project
* **test command**: Danh sách các khai báo cấu hình bổ sung cho npm. Như trong chuỗi trên thì giá trị của test chính là câu thông báo và dừng chương trình khi bị lỗi.
* **author**: Tên tác giả của Project
* **license**: License của Project, giá trị mặc định là ISC.



Hình 1.1.1 Tạo project

Sau khi hoàn tất các lệnh tạo project, một file package.json sẽ được sinh ra trong project với các thông số đã nhập trước đó.

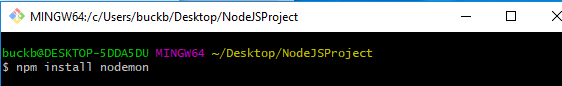


Hình 1.1.2 Nội dung file package.json

* Nodemon

Nếu như đối với PHP, khi cần chỉnh sửa một vài đoạn code trong project thì chỉ cần F5 lại trình duyệt và kiểm chứng,. Còn NodeJS thì khác, mỗi khi có bất cứ thay đổi gì trong project thì bắt buộc phải restart lại server, điều này có nghĩa ta phải Ctrl+C và chạy lại sever nhiều lần sẽ gây ra nhiều phiền toái. Chính vì lý do đó mà Nodemon được ra đời, sau khi cài đặt Nodemon, nếu có thay đổi trong project thì cũng không cần phải chạy lại server.

Lệnh cài đặt Nodemon



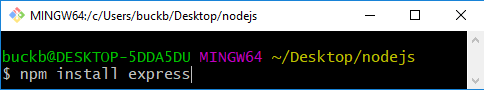
Hình 1.1.3 Lệnh cài Nodemon

* Express Framework

Express là một framework dành cho NodeJS. Nó cung cấp cho chúng ta rất nhiều tính năng mạnh mẽ trên nền tảng web cũng như trên các ứng dụng di động. Express hỗ rợ các phương thức HTTP tạo ra một ứng dụng vô cùng mạnh mẽ. Có thể tổng hợp một số chức năng chính của express như sau:

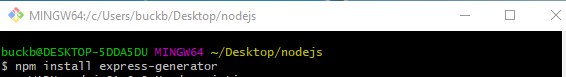
* Express hỗ trợ việc phát triển ứng dụng theo mô hình MVC, mô hình phổ biến cho việc lập trình web hiện nay
* Thiết lập các lớp trung gian để trả về các HTTP request
* Định nghĩa router cho phép sử dụng với các hành động khác nhau dựa trên phương thức HTTP và URL
* Cho phép trả về các trang HTML dựa vào các tham số.

Lệnh cài đặt Express:



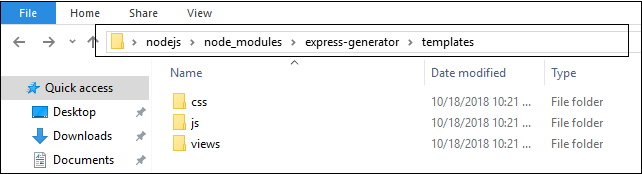
##### Hình 1.1.4 Lệnh cài đặt Express

Ngoài cú pháp *express* là web framework chính, thì còn có *express-generator* là module giúp tạo một project Express mẫu, tức là nó tự tạo cho chúng ta các file và thư mục cần thiết.



##### Hình 1.1.5 Lệnh cài đặt Express generator

Sau khi cài đặt *express-generator*  một module sẽ được tạo ra và có sẵn các thư mục như sau:

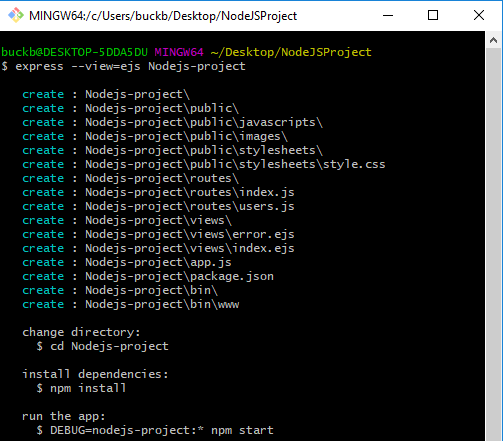


##### Hình 1.1.6 thư mục trong express-generator

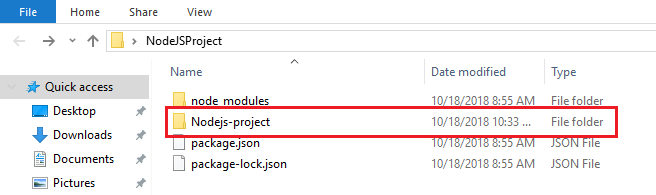
Bên cạnh việc tạo thủ công các thư mục trong project thì *express-generator*  có thể tạo nhanh ra một ứng dụng từ *express* với cú pháp:

*$ express --view=ten\_template ten\_ung\_dung*

Ví dụ: để tạo nhanh một ứng dụng có tên Nodejs-project sử dụng *EJS template*

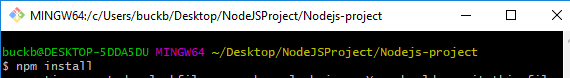


##### Hình 1.1.7 Tạo project sử dụng EJS template



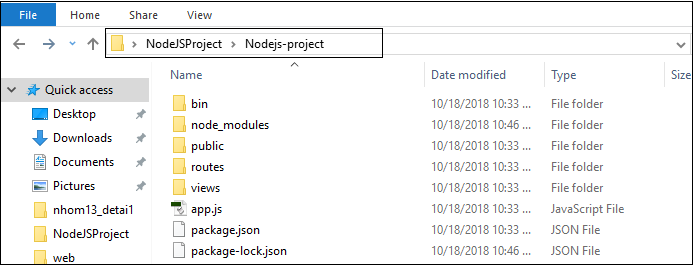
##### Hình 1.1.8 Thư mục EJS template

Sau khi tạo project ta cần phải cài đặt thêm các module từ câu lệnh npm install



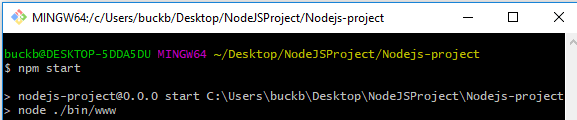
##### Hình 1.1.9 npm install

Như vậy, một project NodeJS đã được tạo ra với các thư mục như sau:



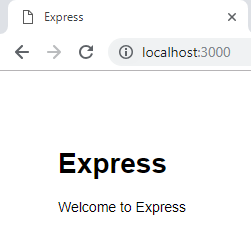
##### Hình 1.1.10 Thư mục project NodeJS

* Khởi chạy server:



##### Hình 1.1.11 Lệnh chạy server

Để chạy server từ *express-generator*  ta dùng lệnh npm start sau đó chạy thử host 3000 trên trình duyệt



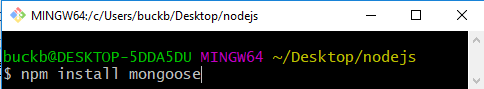
##### Hình 1.1.12 Chạy web nodejs trên trình duyệt

Như vậy, quá trình khỏi tạo nhanh một ứng dụng NodeJS đã thành công.

* Mongoose

[Mongoose](http://mongoosejs.com/) nó là một bộ thư viện giúp ta giao tiếp, xử lý MongoDB với NodeJS nhanh gọn hơn. Để dễ hiểu hơn thì nó tương tự như một driver như dùng JDBC (Java Database Connectivity) để kết với MySQL sử dụng Java vậy, thì ở đây ta dùng Mongoose để giao tiếp với MongoDB với NodeJS.

Lệnh cài đặt Mongoose:



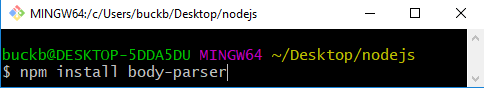
##### Hình 1.1.13 Lệnh cài đặt Mongoose

* Body-parser

Trong NodeJS mỗi khi người dùng gửi yêu cầu đến server để xử lý. Yêu cầu của người dùng không chỉ đơn thuần là các route mà còn bao gồm cả phần dữ liệu mà form post lên trên server. Để server có thể nhận các request từ form post ta sử dụng body-parser để nhận và xử lý (req.body), sau đó respone lại cho người dùng.

* bodyParser.json(): phân tích dữ liệu văn bản dưới dạng json và hiển thị kết quả trên req.body
* bodyParser.urlencoded: phân tích dữ liệu văn bản dưới dạng URL thường được truyền từ form lên và hiển thị kết quả trên req.body:
* bodyParser.urlencoded: đọc bộ đệm dưới dạng văn bản và hiển thị chuỗi kết quả lên req.body

Lệnh cài đặt body-parser:



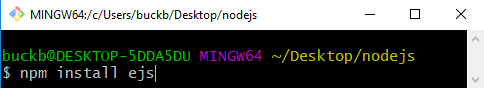
##### Hình 1.1.14 Lệnh cài đặt body-parser

* EJS

EJS là viết tắt của "Embedded JavaScript Templating", đây là một thư viện, được sử

dụng để phân tích các tập tin có phần đuôi mở rộng là ejs, và tạo ra HTML trả về cho client (Trình duyệt).

Lệnh cài đặt EJS:

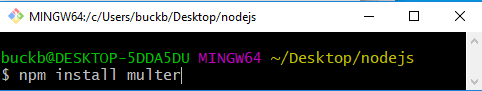


##### Hình 1.1.15 Lệnh cài đặt EJS

* Multer

Đây là một thành phần trung gian trong NodeJS để xử lí phần multipart/form-data, đó chính là xử lý việc upload file từ client về phía server.

Lệnh cài đặt multer:



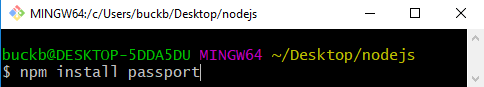
##### Hình 1.1.16 Lệnh cài đặt multer

* Passport.js

Passport là một thư viện hỗ trợ xác thực người dùng authentication trong NodeJS. Nó được thiết kế thành một module vì vậy có thể làm cho Passport dễ dàng tích hợp vào trong ứng dụng của bạn. Trong mô hình web hiện đại, Authen được phát triển thành rất nhiều hình thức khác nhau.

Ngoài việc xác thực bằng cách truyền thống là cần user\_name và password thì trong thời đại của mạng xã hội đang phát triển như Facebook, Twitter..... đòi hỏi có một cách đăng nhập dựa vào tài khoản của các trang mạng xã hội. Vì vậy module Passport được xây dựng để đáp ứng được nhu cầu này. Passport cung cấp các cơ chế Authen [[1]](#footnote-1) đến từng nhà phát hành mạng xã hội khác nhau, vì vậy bạn có thể lựa chọn các cách Authen phù hợp đến mỗi mạng xã hội.

Lệnh cài passport.js:

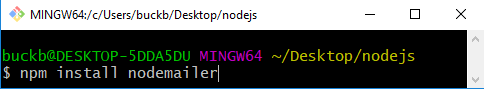


##### Hình 1.1.17 Lệnh cài passport.js

* Nodemailer

Đúng như cái nên mà nó mang, nodemailer có chức năng chính là gửi email cho người dùng. NodeMailer là một trong những module cần thiết nhất nếu như bạn muốn xây dựng một website hoàn hảo. Chúng ta cần phải đặc biệt quan tâm tới việc bảo mật thông tin trong việc gửi và nhận mail nhưng với module nodemailer điều này trở thành một việc rất đơn giản và dễ dàng nhờ nodemailer.

Lệnh cài đặt nodemailer:



##### Hình 1.1.18 Lệnh cài đặt nodemailer

### CÁC IDE NÊN DÙNG KHI LÀM VIỆC VỚI NODEJS

* Visual Studio Code

Visual Studio Code là một trình biên tập mã được phát triển bởi [Microsoft](https://vi.wikipedia.org/wiki/Microsoft) dành cho [Windows](https://vi.wikipedia.org/wiki/Microsoft_Windows), [Linux](https://vi.wikipedia.org/wiki/Linux) và [macOS](https://vi.wikipedia.org/wiki/MacOS). Nó hỗ trợ chức năng debug, đi kèm với [Git](https://vi.wikipedia.org/wiki/Git_(ph%E1%BA%A7n_m%E1%BB%81m)), có syntax highlighting, tự hoàn thành mã thông mình, [snippets](https://vi.wikipedia.org/w/index.php?title=Snippets&action=edit&redlink=1), và [cải tiến mã nguồn](https://vi.wikipedia.org/wiki/C%E1%BA%A3i_ti%E1%BA%BFn_m%C3%A3_ngu%E1%BB%93n). Nó cũng cho phép tùy chỉnh, do đó, người dùng có thể thay đổi theme, phím tắt, và các tùy chọn khác. Nó miễn phí và là [phần mềm mã nguồn mở](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_t%E1%BB%B1_do_ngu%E1%BB%93n_m%E1%BB%9F), mặc dù gói tải xuống chính thì là [có giấy phép](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_s%E1%BB%9F_h%E1%BB%AFu_%C4%91%E1%BB%99c_quy%E1%BB%81n).

Trong cuộc khảo sát vào năm 2017 trên Stack Overflow, Visual Studio Code được xếp hạng là trình biên tập mã phổ biến nhất, với 34.9% của 75398 người trả lời tuyên bố sử dụng nó.

* Atom

Atom là một trình soạn thảo văn bản (và mã nguồn) miễn phí, mã nguồn mở, hỗ trợ các hệ điều hành Linux, Mac OS, Windows. Nó hỗ trợ các plugin để lập trình với NodeJS và nhúng Git-Control, phần mềm này được phát triển bởi GitHub.

Atom được đánh giá là phần mềm soạn thảo văn bản (và mã nguồn) có giao diện hiện đại, dễ sử dụng và có các tính năng vượt trội so với các phần mềm tương tự.

Các ngôn ngữ được hỗ trợ bởi Atom: C/C++, C#, CSS, Git, HTML, JavaScript, Java, JSON, Objective-C, PHP,Python, Ruby on Rails, Ruby, SQL, XML

* Sublime Text

Đây là ứng dụng được viết bằng C++ và Python bởi tác giả Jon Skinner và đã thu hút được hơn 2,5 triệu người sử dụng kể từ khi ra mắt vào năm 2008. Những nhận xét về nó chủ yếu là tích cực và trong khi đây không phải là trình soạn thảo code phức tạp nhất, nhưng chúng ta dựa vào nó để thiết lập ra những tiêu chuẩn cho các trình soạn thảo kế tiếp.Các tính năng nổi bật bao gồm:

* Hỗ trợ ngôn ngữ toàn diện với cú pháp tương thích TextMate
* Đa lựa chọn và chỉnh sửa cột
* Điều hướng nhanh chóng tới các vị trí file và code
* Một command palette để truy cập tất cả các tính năng
* Một mini-map và các code snippet
* Tùy biến mở rộng, nhiều theme tuyệt vời và rất nhiều plugin
* Đẹp, nhanh và ổn định

### CÁC CẤU TRÚC LỆNH CƠ BẢN TRONG NODEJS

* **Module**

**Module là các đoạn code được đóng gói lại với nhau và được giữ Private, chỉ các hàm và biến được định nghĩa bên trong Module là có thể truy cập và thao tác với nhau.** NodeJS sử dụng Module để đơn giản hóa việc tạo ra các ứng dụng phức tạp. Module là giống như các thư viện trong PHP, C, C#,… Mỗi module chứa một tập các hàm chức năng có liên quan đến một đối tượng của Module qua đó giúp việc viết và quản lý mã lệnh của chương trình được dễ dàng hơn. Một module có thể đơn giản là một hàm hay một đối tượng. Mỗi module thường được khai bảo ở một tập tin riêng rẽ.

Ví dụ, http là Module chứa các hàm cụ thể liên quan đến thiết lập HTTP. NodeJS cung cấp một vài các Module core kèm theo để hỗ trợ chúng ta truy cập file trên hệ thống, tạo các máy chủ HTTP, TCP/UDP, và các hàm tiện ích nhỏ hữu dụng khác.

* **Require**

Trong NodeJS, sau khi đã có các Module việc tiếp theo cần làm là sử dụng được các Module đó. Để sử dụng Module, ta đơn giản chỉ cần khai báo với hàm require(), như sau:

Ví dụ: var http = require("http");

Hàm require() là hàm trả về tham chiếu tới một Module cụ thể. Trong trường hợp của đoạn mã trên, chúng ta đang khai báo một tham chiếu tới http Module và lưu nó vào biến http.

Trong đoạn mã trên, chúng truyền một tham số là tên của Module. Điều này báo cho NodeJS sẽ tìm một Module tên là http trong thư mục node\_modules của ứng dụng. Nếu nó không thấy, Node sẽ tiếp tục tìm Module đó ở thư mục global cài đặt node.

Ngoài ra ta cũng có thể chỉ rõ file bằng việc truyền vào tham số là đường dẫn tương đối ./path /module.js hoặc tuyệt đối /path /module.js.

Ví dụ: var http = require("./path /module.js");

* **Export**

Như đã trình bày ở phía trên, module là các đoạn code được đóng gói lại với nhau,... Code trong một Module thường là private – nghĩa là các hàm, biến được định nghĩa và truy cập bởi bên trong của Module. Để đưa các hàm hoặc biến để sử dụng bên ngoài Module bạn cần sử dụng exports.

Có thể hiểu theo cách đơn giản là: khi làm việc với NodeJS chúng ta có thể tạo ra mọi hàm cùng nằm trong một module duy nhất. Nhưng về lâu về dài sau này, ứng dụng của chúng ta sẽ trở nên chậm chạm vì phải mất nhiều thời gian biên dịch hết tất cả các code nằm trong module đó. Chính vì vậy mà ta cần phải chia nhỏ module đó ra thành các module nhỏ hơn để tăng tốc độ xử lý và khả năng bảo trì sau này. Việc chi nhỏ module sẽ nhờ vào Export.

Ví dụ: để tạo ra một module trả về câu ’Hello word’ ta sử dụng export như sau:

var xinchao=function(){

return "Hello Word"

};

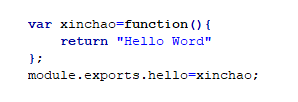
module.exports.hello=xinchao;

* **Tạo một project đơn giản**

Sau khi đã hiểu các khái niệm về module, export, require ta có thể làm một project NodeJS đơn giản như sau: xuất câu ‘Hello Word’ lên trình duyệt web.

* **Bước 1**

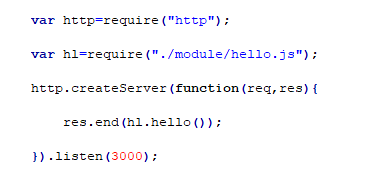
Tạo một folder module chứa tập tin hello.js với nội dung như sau:



File hello.js đóng vai trò là một module và export ra một function có tên hello với nội dung được lấy từ function xinchao trong hello.js

* **Bước 2**

Tạo file helloword.js với nội dung



Trong các câu lệnh trên

var http=require("http");

ta khởi tạo ra biến http từ module HTTP, module này có nhiệm vụ khởi tạo một cổng kết nối HTTP server trả về client.

var hl=require("./module/hello.js");

Biến *hl* chính là module đã được export ra từ bước 1. Tiếp đó để khởi tạo server trong HTTP module chúng ta sử dụng 2 phương thức createServer().listen() của http. Bên trong phương thức createServer chứa một phương thức ẩn danh có 2 tham số truyền vào:

- Tham số thứ nhất: là biến chứa các thông số liên quan đến request mà người dùng gửi lên.

-Tham số thứ hai là biến chứa các thông số liên quan đến response mà chúng ta muốn trả về client.

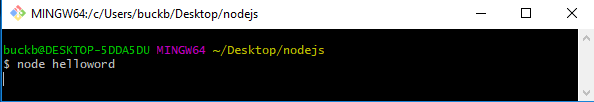
listen(3000) được sử dụng để phía server chờ và lắng nghe các truy cập đến port 3000, nếu có bất kỳ request nào được gửi đến port 3000, server sẽ bắt được và xử lý nó.

res.end(hl.hello());

*req.end* là câu lệnh mà phía server sẽ trả các respone về lại cho client. Trong trường hợp này, giá trị trả về sẽ là function hello của module mà ta đã tạo ra ở bước 1.

* **Bước 3**

Sử dụng terminal để mở kết nối đến server

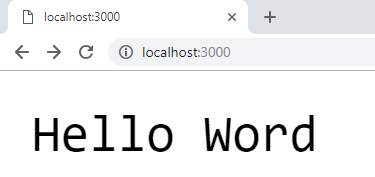


##### Hình 1.1.19 Sử dụng terminal mở kết nối

Lệnh ‘node helloword’ sẽ gọi đến file helloword.js, trong file này có chứa module http để mở kết nối server và sẵn sàn nhân các request từ client đến port 3000

* **Bước 4**

Mở trình trình duyệt web lên và truy cập vào đường dẫn <http://localhost:3000/>



##### Hình 1.1.20 Ứng dụng helloword

Server đã bắt được request của người dùng sau đó xử lý và trả dữ liệu ‘Hello Word’ cho client.

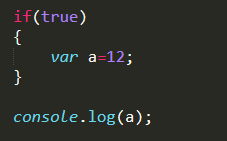
### FUNCTION VÀ TỪ KHÓA KHAI BÁO TRONG JAVASCRIPT ES6

### Từ khóa let và const

Đối với javascript thông thường, để khai báo biến ta dùng từ khóa *var* cho mọi đối tượng. Nhưng trong ES6 ta được cung cấp thêm 2 từ khóa nữa là *let* và *const*, vậy ba từ khóa này khác nhau như thế nào?

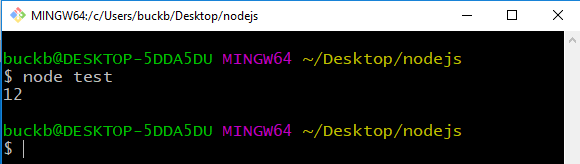
* ***Let***

*let* tạo ra một biến chỉ có thể truy cập được trong *block { }* bao quanh nó, khác với *var* - tạo ra một biến có phạm vi truy cập xuyên suốt cả *function* chứa nó. Để biết được sự khác nhau giữa *let* và *var* ta cùng xem ví dụ sau:



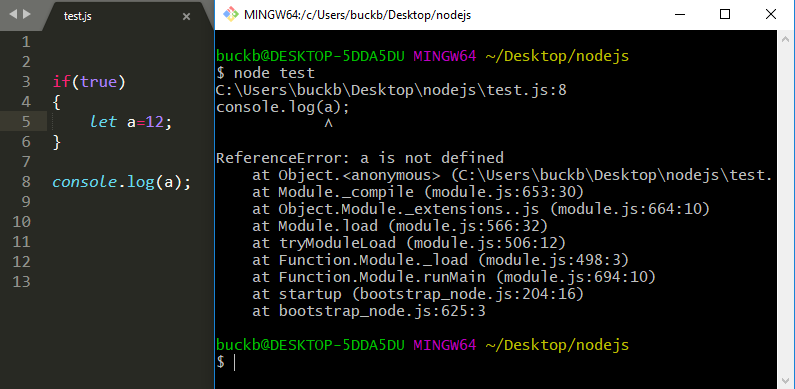
##### Hình 1.1.21 ví dụ từ khóa var

Trong ví dụ này, câu lệnh *if* được xem là một *block* và từ khóa *var* khai báo một biết *a=12* thì kết quả sau khi thực thi sẽ là:



##### Hình 1.1.22 kết quả chạy từ khóa var

Nhưng nếu chúng ta thay thế từ khóa var bằng từ khóa let trong trường hợp phía trên thì:

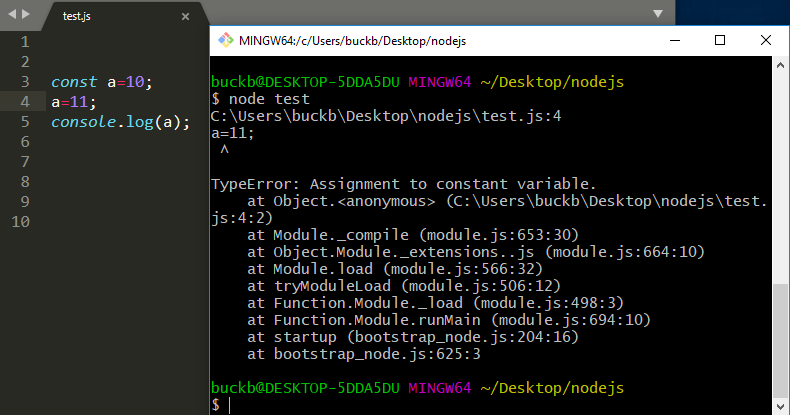


##### Hình 1.1.23 kết quả chạy từ khóa let

Lỗi đã xảy ra tại biến *a*, điều này cho thấy từ khóa *let* chỉ được sử dụng trong một *block*, khi ra khỏi *block* thì biết đó sẽ không còn tồn tại.

* ***const***

Khi khai báo biến bằng từ khóa *const*, biến lúc này sẽ được hiểu là một hằng và chỉ được phép sử dụng chứ không thể thay đổi. Ví dụ:

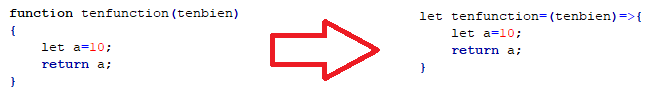


##### Hình 1.1.24 kết quả chạy từ khóa const

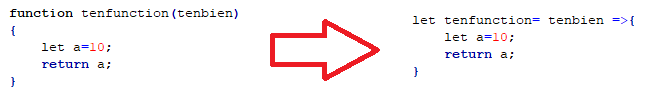
Lỗi xảy ra ở dấu *=* khi cố thay đổi giá trị của hằng số *a*

### Arrow function trong ES6

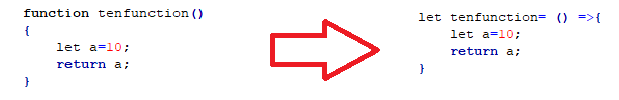
Arrow function được sinh ra với mục đích làm thu gọn code function trong javascript, để thấy được sự khác nhau giữa function bình thường và arrow function ta sét các ví dụ sau:



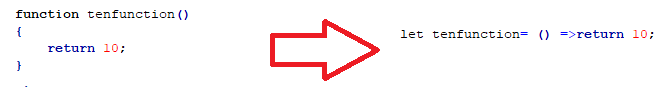
Trong trường hợp function chỉ có duy nhất một tham số truyền vào ta có thể lược bỏ luôn cặp dấu ( ) bao quanh tham số.



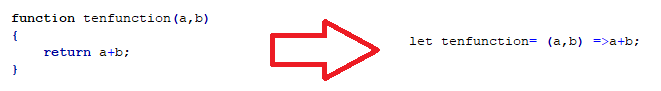
Lưu ý: chỉ có function có một tham số thì mới được lược bỏ cặp ( ), nếu không có tham số nào thì vẫn phải giữ lại cặp ( )



Nếu function được tạo ra chỉ để xử lý một câu lệnh hay nói cách khác là chỉ có một câu lệnh được chứa trong function ta có thể rút gọn như sau:



Ngoài ra nếu function chỉ có một lệnh duy nhất và được trả về với từ khóa return thì ta có thể lược bỏ luôn *return* vì => đã bao hàm luôn cả ý nghĩa của *return*



### CALLBACK VÀ PROMISE

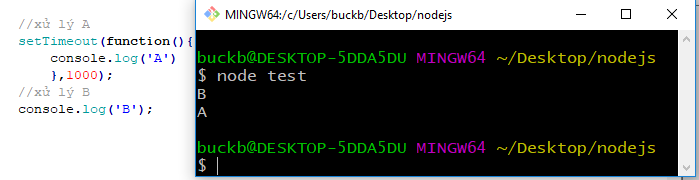
### Đồng bộ và bất đồng bộ

* Đồng bộ

Xử lý đồng bộ nghĩa là chương trình sẽ chạy theo từng bước từ trên xuống dưới và chỉ khi nào bước 1 thực hiện xong thì mới nhảy sang bước 2, khi nào chương trình này chạy xong mới nhảy qua chương trình khác. Chương trình sẽ chạy theo đúng thứ tự và có nguyên tắc nên sẽ không mắc phải các lỗi về tiến trình không cần thiết. Chương trình chạy theo thứ tự đồng bộ nên sẽ sinh ra trạng thái chờ và là không cần thiết trong một số trường hợp, lúc này bộ nhớ sẽ dễ bị tràn vì phải lưu trữ các trạng thái chờ.

* Bất đồng bộ

Ngược lại với xử lý đồng bộ, trong xử lý bất đống bộ hàm A được gọi và thực thi trước hàm B nhưng hàm A không đảm bảo là sẽ kết thúc trước hàm B và hàm B cũng không đảm bảo sẽ thực thi sau hàm A. NodeJS được xử lý theo hướng bất đồng bộ, xét ví dụ đơn giản sau:



##### Hình 1.1.25 Ví dụ bất đồng bộ

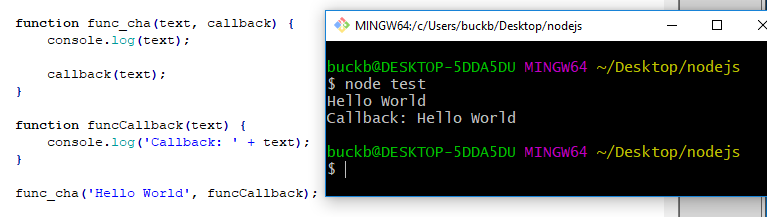
Trong ví dụ trên, rõ ràng xử lý A được gọi và thực thi trước nhưng kết quả lại thực thi xử lý B trước rồi mới đến A, vì lúc này hàm xử lý A phải chờ trong khoản thời gian 1s mới được xử lý, trong khoản thời gian đó NodeJS sẽ thực thi các hàm khác.

-Ưu điểm bất đống bộ: nếu như server có nhiều request cùng một lúc, thì bất đồng bộ sẽ không phải chờ xử lý lần lượt các request đó.

-Nhược điểm: vì bất đống bộ sẽ không lần lượt chạy theo trình tự, nên sẽ rất khó kiểm soát nếu không cẩn thận, dẫn đến kết quả trả ra sai lệnh.

### Callback

Callback function là một function A được truyền vào một function B thông qua danh sách các tham số của B. Lúc này tại hàm B sẽ gọi đến hàm A để thực hiện một chức năng nào đó mà A đang nắm giữ. Ví dụ:



##### Hình 1.1.26 Ví dụ callback

Function *func\_cha* có một tham số truyền vào kiểu function là *callback* và *callback* sẽ được gọi lại trong chính *func\_cha*

### Promise

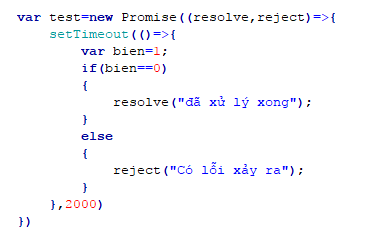
Promise là một đối tượng đặc biệt dùng cho các xử lý bất đồng bộ. Nó đại diện cho một xử lý bất đồng bộ và chứa kết quả cũng như các lỗi xảy ra từ xử lý bất đồng bộ đó. Promise sinh ra để xử lý kết quả của một hành động cụ thể, có ý nghĩa gần giống như callback. Nhưng cách viết của Promise sẽ dễ dàng hơn là sử dụng callback.

Tại mỗi thời điểm Promise sẽ có thể ở một trong các trạng thái sau:

-Pending: Trạng thái chờ xử lý kết thúc. Trạng thái này chính là trạng thái ban đầu của Promise, nó thể hiện rằng thao tác xử lý của ta chưa kết thúc.

-Resolve: Trạng thái xử lý thành công. Trạng thái này thể hiện rằng thao tác xử lý của ta đã kết thúc và thành công.

-Reject: Trạng thái xử lý thất bại. Trạng thái này thể hiện thao tác xử lý đã kết thúc và thất bại. Ví dụ:

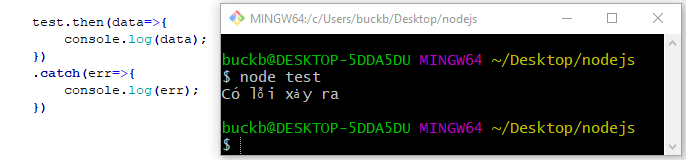


Các giá trị trả về khi ta gọi trong hàm *resolve* , *reject* sẽ được trả về tương ứng với *then* và *catch*. Then và cath chính là hai phương thức gọi promise sau khi sử lý xong

-Then(): khi xử lý của promise thành công thì các lệnh xử lý trong phương thức then sẽ được thực thi.

-Catch(): khi xử lý của promise thất bại thì các lệnh xử lý trong phương thức catch sẽ được thực thi.

Xét tiếp ví dụ ở trên, ta thực hiện như sau:



##### Hình 1.1.27 Ví dụ promise

Vì *bien=1* nên *reject* sẽ được thực thi, khi gọi đến promise thì catch sẽ thực hiện việc xử lý và trả kết quả như hình.

### KẾT CHƯƠNG

### Ưu điểm

Ưu điểm lớn nhất của NodeJS là về tốc độ thực thi và khả năng mở rộng. NodeJS có tốc độ rất nhanh. Đó là một yêu cầu khá quan trọng khi xây dựng, tạo ra một sản phẩm lớn và muốn đảm bảo có thể mở rộng nhanh chóng, đáp ứng được một lượng lớn người dùng khi trang web phát triển lên.

Đặc điểm nổi bật của NodeJS là nó nhận và xử lý nhiều kết nối chỉ với một single-thread. Điều này giúp hệ thống tốn ít RAM nhất và chạy nhanh nhất khi không phải tạo luồng mới cho mỗi truy vấn giống PHP. Ngoài ra, tận dụng ưu điểm bất đồng bộ của Javascript mà NodeJS tận dụng tối đa tài nguyên của server mà không tạo ra độ trễ như PHP.

Với sự ra đời của các ứng dụng di động & HTML5 nên NodeJS rất hiệu quả khi xây dựng những ứng dụng thời gian thực (real-time applications) như ứng dụng chat, các dịch vụ mạng xã hội như Facebook, Twitter,…

### Nhược điểm

Giống như hầu hết các công nghệ mới, việc triển khai NodeJS trên host không phải là điều dễ dàng. NodeJS sẽ làm ứng dụng nặng và tốn tài nguyên Nếu bạn cần xử lý các ứng dụng tốn tài nguyên CPU như encoding video, convert file, decoding encryption... hoặc các ứng dụng tương tự như vậy thì không nên dùng NodeJS (Lý do: NodeJS được viết bằng C++ & Javascript, nên phải thông qua thêm một trình biên dịch của NodeJS sẽ lâu hơn).

Một nhược điểm lớn khác của NodeJS đó là nó vẫn đang trong giai đoạn phát triển ban đầu, điều này có nghĩa là một số đặc trưng sẽ thay đổi trong quá trình phát triển tiếp theo.

# TỔNG QUAN VỀ MONGODB



### GIỚI THIỆU VỀ MONGODB

### Nosql là gì?

NoSQL ra đời năm 1998 bởi Carlo Strozzi khi ông lập mới một hệ cơ sở dữ liệu quan hệ mã nguồn mở nhanh và nhẹ không liên quan đến SQL Vào năm 2009, Eric Evans, nhân viên của Rackspace giới thiệu lại thuật ngữ NoSQL khi Johan Oskarsson của Last.fm muốn tổ chức một hội thảo về cơ sở dữ liệu mã nguồn mở phân tán. Thuật ngữ NoSQL đánh dấu bước phát triển của thế hệ CSDL mới: phân tán (distributed) + không ràng buộc (non-relational).

NoSQL nghĩa là Non SQL hay cũng có thể gọi là Not only SQL. Hiểu một cách đơn giản nhất thì NoSQL là một hệ cơ sở dữ liệu không sử dụng mô hình quan hệ như các hệ quản trị cơ sở phổ biến khác hiện nay như là MySQL hay SQL server. Khi các hệ thống lớn ra đời, sự mở rộng về mặt quy mô cũng như tốc độ truy suất trở nên cần thiết hơn bao giờ hết. Cơ sở dữ liệu quan hệ khó có khả năng giải quyết triệt để những vấn đề này. Chính vì thế NoSQL ra đời nhằm khắc phục những vấn đề trên.

### MongoDB là gì?

MongoDB là hệ quản trị cơ sở dữ liệu NoSQL hướng văn bản, được phát triển bởi công ty phần mềm 10gen vào năm 2007. MongoDB được phát hành mã nguồn mở vào năm 2009.

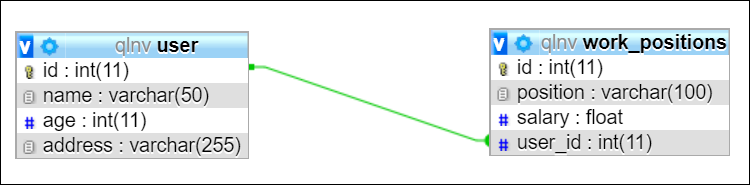
MongoDB được thiết kế theo kiểu hướng đối tượng trong đó các bảng được cấu trúc một cách linh hoạt cho phép các dữ liệu lưu trên bảng không cần phải tuân theo một dạng cấu trúc nhất định nào. Chính do cấu trúc linh hoạt này nên MongoDB có thể được dùng để lưu trữ các dữ liệu có cấu trúc phức tạp và đa dạng và không cố định

### Các kiểu dữ liệu trong MongoDB

|  |  |
| --- | --- |
| **Kiểu** | **Ký hiệu** |
| **Double** | **“double”** |
| **String** | **“string”** |
| **Object** | **“object”** |
| **Array** | **“array”** |
| **Binary** | **“binData”** |
| **Undefined** | **“undefined”** |
| **Objectld** | **“objectld”** |
| **Boolean** | **“bool”** |
| **Date** | **“date”** |
| **Null** | **“null”** |
| **Regular Expression** | **“regex”** |
| **DBPointer** | **“dbPointer”** |
| **JavaScript** | **“javascript”** |
| **Symbol** | **“symbol”** |
| **JavaScript (with scope)** | **“javascriptWithScope”** |
| **32-bit integer** | **“int”** |
| **Timestamp** | **“timestamp”** |
| **61-bit integer** | **“long”** |
| **Decimal128** | **“decimal”** |
| **Min key** | **“minKey”** |
| **Max key** | **“maxKey”** |

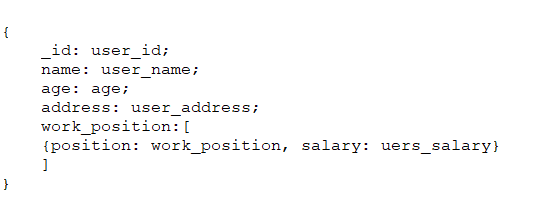
### SO SÁNH MONGODB VỚI CƠ SỞ DỮ LIỆU QUAN HỆ

Xét một ví dụ cụ thể giữa MongoDB và MySQL như sau:



Hình 1.. Mô hình cơ sở dữ liệu mysql

Trong mô hình của mysql, để lưu thông tin của một User, ta phải tạo ra hai table và liên kết các table đó lại với nhau. Đối với MongoDB sẽ chỉ cần có một collection như sau:



- Một số thay đổi về khái niệm:

|  | **Mysql** | **MongoDB** |
| --- | --- | --- |
| Create | Insert | Insert |
| Read | Select | Find |
| Update | Update | Update |
| Delete | Delete | Remove |
| Table | Table | Collection |
| Row | Row | Document |

* **Các thuật ngữ hay sử dụng trong MongoDB**

**Primary Key**: Mặc định primary key trong MongoDB là field \_id, lúc insert nếu bạn không chỉ rõ trường \_id thì nó sẽ tự động gán giá trị cho \_id.

**\_id**: Là trường bắt buộc có trong mỗi document. Trường \_id đại diện cho một giá trị duy nhất trong document MongoDB. Trường \_id cũng có thể được hiểu là khóa chính trong document. Nếu bạn thêm mới một document thì MongoDB sẽ tự động sinh ra một \_id đại diện cho document đó và là duy nhất trong cơ sở dữ liệu MongoDB.

**Collection**: Là nhóm của nhiều document trong MongoDB. Collection có thể được hiểu là một bảng tương ứng trong cơ sở dữ liệu RDBMS (Relational Database Management System). Collection nằm trong một cơ sở dữ liệu duy nhất. Các collection không phải định nghĩa các cột, các hàng hay kiểu dữ liệu trước.

**Cursor**: Đây là một con trỏ đến tập kết quả của một truy vấn. Máy khách có thể lặp qua một con trỏ để lấy kết quả.

**Database**: Nơi chứa các Collection, giống với cơ sở dữ liệu RDBMS chúng chứa các bảng. Mỗi Database có một tập tin riêng lưu trữ trên bộ nhớ vật lý. Một máy chủ MongoDB có thể chứa nhiều Database.

**Document**: Một bản ghi thuộc một Collection thì được gọi là một Document. Các Document lần lượt bao gồm các trường tên và giá trị trong MongoDB có cấu trúc tương tự như kiểu dữ liệu JSON, nghĩa là sẽ có các cặp (key => giá trị) nên nó có tính năng động rất lớn. Document ta có thể hiểu nó giống như các record dữ liệu trong MYSQL, tuy nhiên nó có sự khác biệt là các cặp (key => value) có thể không giống nhau ở mỗi document.

**Embedded Documents**: Nếu trong RDBMS, chúng ta có kiểu join các bảng, ví dụ bảng person join với bảng address để biết một person có những address nào thì trong MongoDB, document của person sẽ chứa 1 array các document address.

**Field**: Là một cặp name – value trong một document. Một document có thể có không hoặc nhiều trường. Các trường giống các cột ở cơ sở dữ liệu quan hệ.

**JSON**: Viết tắt của JavaScript Object Notation. Con người có thể đọc được ở định dạng văn bản đơn giản thể hiện cho các dữ liệu có cấu trúc. Hiện tại JSON đang hỗ trợ rất nhiều ngôn ngữ lập trình.

**Index**: Là những cấu trúc dữ liệu đặc biệt, dùng để chứa một phần nhỏ của các tập dữ liệu một cách dễ dàng để quét. Chỉ số lưu trữ giá trị của một fields cụ thể hoặc thiết lập các fields, sắp xếp theo giá trị của các fields này. Index hỗ trợ độ phân tích một cách hiệu quả các truy vấn. Nếu không có chỉ mục, MongoDB sẽ phải quét tất cả các documents của collection để chọn ra những document phù hợp với câu truy vấn. Quá trình quét này là không hiệu quả và yêu cầu MongoDB để xử lý một khối lượng lớn dữ liệu.

Ví dụ minh họa:

|  |  |  |
| --- | --- | --- |
| **Câu lệnh** | **SQL** | **MongoDB** |
| Create table | CREATE TABLE people (id MEDIUMINT NOT NULL AUTO\_INCREMENT, user\_id Varchar(30), age Number, status char(1), PRIMARY KEY (id)) | db.people.insertOne({User\_id: “abc123”, Age: 55, Status: “A”}) |
| Drop table | DROP TABLE people | db.people.drop() |
| Select | SELECT \*FROM people | db.people.find() |
| SELECT \* FROM people WHERE status = "A" | db.people.find( { status: "A" } ) |
| Update records | UPDATE people SET status = "C" WHERE age > 25 | db.people.updateMany( { age: { $gt: 25 } }, { $set: { status: "C" } } ) |
| UPDATE people SET age = age + 3 WHERE status = "A" | db.people.updateMany( { status: "A" } , { $inc: { age: 3 } } ) |
| Delete Records | DELETE FROM people WHERE status = "D" | db.people.deleteMany( { status: "D" } ) |
| DELETE FROM people | db.people.deleteMany({}) |

* **Một số lợi thế của MongoDB so với các cơ sở dữ liệu quan hệ**

+Ít Schema hơn: MongoDB là một cơ sở dữ liệu dựa trên Document, trong đó một Collection giữ các Document khác nhau. Số trường, nội dung và kích cỡ của Document này có thể khác với Document khác.

+Cấu trúc của một đối tượng là rõ ràng.

+Không có các Join phức tạp.

+Khả năng truy vấn sâu hơn. MongoDB hỗ trợ các truy vấn động trên các Document bởi sử dụng một ngôn ngữ truy vấn dựa trên Document mà mạnh mẽ như SQL.

+MongoDB dễ dàng để mở rộng.

+Việc chuyển đổi/ánh xạ của các đối tượng ứng dụng đến các đối tượng cơ sở dữ liệu là không cần thiết.

+Sử dụng bộ nhớ nội tại để lưu giữ phần công việc, giúp truy cập dữ liệu nhanh hơn.

* **Một số lưu ý khi sủ dụng MongoDB**

+MongoDB sinh ra để tăng tốc độ truy xuất dữ liệu, phù hợp cho các ứng dụng cần tốc độ phản hồi nhanh (realtime như facebook chẳng hạn). Còn các tác nghiệp cần tính toàn vẹn dữ liệu(trong banking) thì Nosql sẽ ko bao giờ là 1 giải pháp cả mà người ta sẽ dùng Mysql.

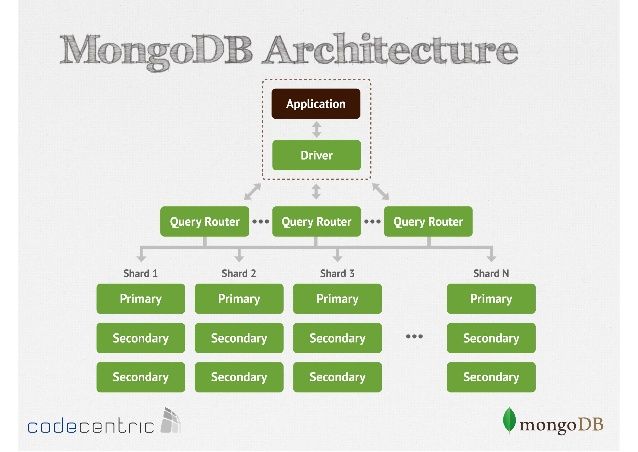
+MongoDB không có tính ràng buộc, một điều tồi tệ trong Database vì vậy sẽ rất cần sự cẩn thận khi thao tác trên các collection có quan hệ dữ liệu với nhau.

+MongoDB đẩy trách nghiệm thao tác Database cho tầng ứng dụng nên sẽ tốn tài nguyên(tài nguyên bây giờ không còn là vấn đề quá lớn nữa).

+MongoDb có thể mở rộng theo chiều ngang (scale out) phương pháp tăng cường khả năng lưu trữ và xử lý là dùng nhiều máy tính phân tán. MongoDb còn có thể mở rộng theo chiều dọc (scale up) tăng cấu hình server.

+MongoDb có thể dùng nhiều máy tính phân tán để lưu trữ dữ liệu nên chi phí sẽ rẻ hơn Mysql. Mysql sử dụng những máy chủ hàng khủng, độc quyền nên sẽ đắt đỏ hơn.

### NGUYÊN TẮC HOẠT ĐỘNG CỦA MONGODB



Hình 1.2.2 Cách hoạt động của MongoDB

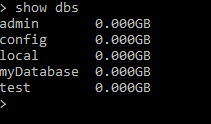
*(Nguồn https://viblo.asia/p/tim-hieu-ve-mongodb-4P856ajGlY3)*

* MongoDB hoạt động dưới một tiến trình ngầm service, luôn mở một cổng (Cổng mặc định là 27017) để lắng nghe các yêu cầu truy vấn, thao tác từ các ứng dụng gửi vào sau đó mới tiến hành xử lý.
* Mỗi một bản ghi của MongoDB được tự động gắn thêm một field có tên “\_id” thuộc kiểu dữ liệu ObjectId mà nó quy định để xác định được tính duy nhất của bản ghi này so với bản ghi khác, cũng như phục vụ các thao tác tìm kiếm và truy vấn thông tin về sau. Trường dữ liệu “\_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.
* Mỗi khi có một truy vấn dữ liệu, bản ghi được cache (ghi đệm) lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.Khi có yêu cầu thêm/sửa/xóa bản ghi, để đảm bảo hiệu suất của ứng dụng mặc định MongoDB sẽ chưa cập nhật xuống ổ cứng ngay, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng.

### SỬ DỤNG MONGODB

* **Hiện tất cả database**

*show dbs*



Hình 1.. Lệnh hiện tấ cả database

* **Tạo database**

*use tendatabase*



Hình 1.. Lệnh chọn database

Lưu ý: ở lệnh này, ta sẽ chọn database cần sử dụng, nếu database đó chưa có thì sẽ tự động tạo ra một database mới.

* **Tạo một collection**

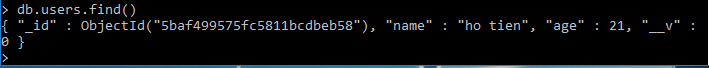
*db.createCollection(‘tên collection’)*



Hình 1.. Lệnh tạo collection

* **Lấy tất các các document của một collection**

*db.tencollection.find()*



Hình 1.. Lệnh tìm tất cả document

* **Thêm document**

-Thêm một document:

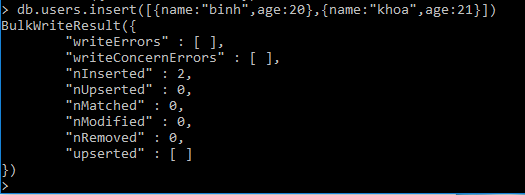
*db.tencollection.insert({name:”tien”,age:20})*



Hình 1.. Lệnh thêm một document

-Thêm nhiều document:

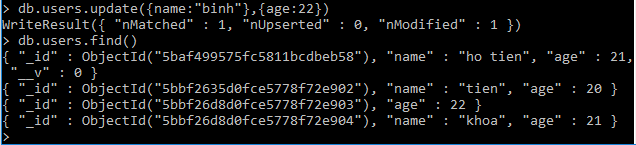
*db.tencollection.insert([{name:”binh”,age:20},{name:”khoa”,age:21}])*



Hình 1.. Lệnh thêm nhiều document

* **Update document**

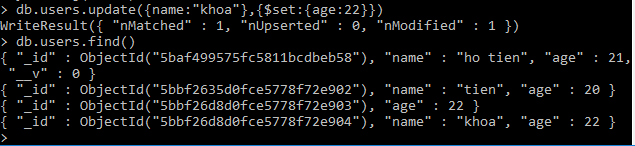
*db.tencollection.update({//điều kiện},{//thay đổi thông số})*



Hình 1.. Lệnh update document

Lưu ý: ở câu lệnh phía trên, dù chỉ thay đổi một thuộc tính là *age* của *user* có *name* là “*binh*”, nhưng tất cả các thuộc tính khách đều bị thay đổi. Để tránh bị thay đổi dữ liệu như trường hợp trên khi update ta dùng thêm từ khóa $set như sau:

*db.tencollection.update({//điều kiện},{$set:{//thay đổi thông số}})*

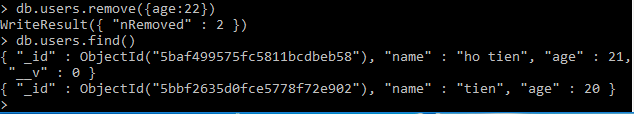


Hình 1.. Lệnh update không thay đổi thuộc tính khác

User có *name* “*khoa*” chỉ bị thay đổi thuộc tính *age* mà không thay đổi các thuộc tính khác.

* **Xóa document**

*db.tencollection.remove({//điều kiện xóa})*

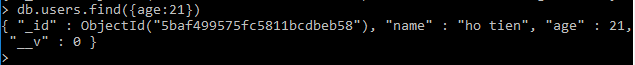


Hình 1.. Lệnh xóa document

* **Truy vấn có điều kiện**

- Phép bằng

*{key:value}*



Hình 1.. So sánh bằng

-Phép nhỏ hơn

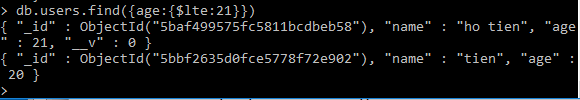
*{key: {$lt: value}}*



Hình 1.. So sánh nhỏ hơn

-phép nhỏ hơn bằng

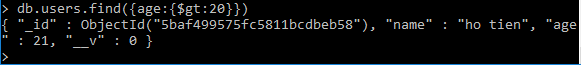
*{key: {$lte: value}}*



Hình 1.. So sánh nhỏ hơn bằng

-Phép lớn hơn

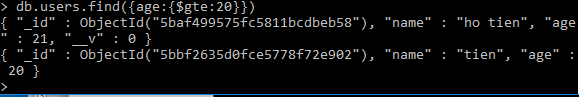
*{key: {$gt: value}}*



Hình 1.. So sánh lớn hơn

-Phép lớn hơn bằng

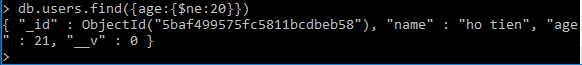
*{key: {$gte: value}}*



Hình 1.. So sánh lớn hơn bằng

-Phép khác

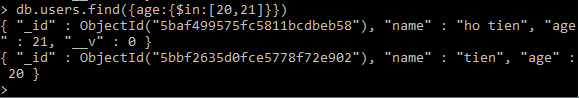
*{key: {$ne: value}}*



Hình 1.. So sánh khác

-Phép trong (in)

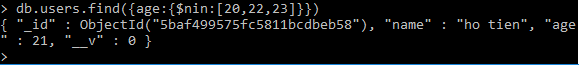
*{key: {$in: [value1, value2,..]}}*



Hình 1.. Phép truy vấn 'in'

-Phép không thuộc (not in)

*{key: {$nin: [value1, value2,..]}}*



Hình 1.. Phép truy vấn 'not in'

* **Truy vần nhiều điều kiện**

-AND

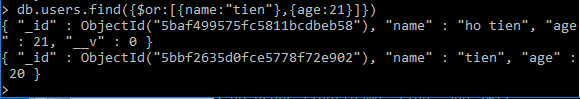
*db.tencollection.find({//các tham số điều kiện})*



Hình 1.. Phép 'and'

-OR

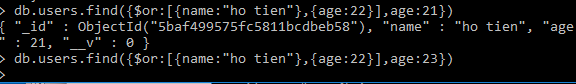
*db.tencollection.find({$or:[{key1:value1},{key2:value2}… ]})*



Hình 1.. Phép 'or'

-Kết hợp cả OR và AND

*db.tencollection.find({$or:[//điều kiện OR],//điều kiện AND})*



Hình 1.. Phép 'or' và 'and'

* **Chọn lọc các trường cần lấy**

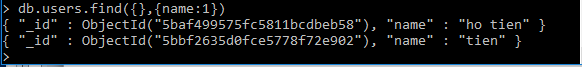
*db.collectionName.find(objectwhere,objectselect)*

Trong đó:

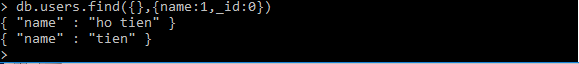
**objectwhere** là object chứa các điều kiện ở các phần trên. Nếu không muốn lọc theo điều kiện thì ta để một object rỗng vào.

**objectselect** là obejct chứa các trường dữ liệu cần lấy ra. Mặc định thì nó sẽ lấy cả \_id, nên nếu như không muốn hiển thị \_id thì ta cần thêm \_id: 0 vào object.

Ví dụ: Chỉ lấy ra tên nhân viên mà không lấy ra tuổi



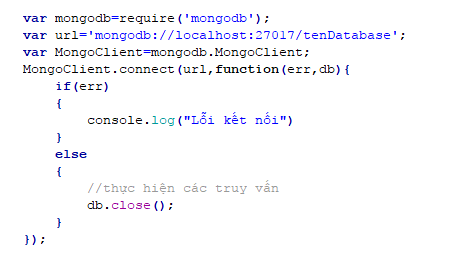
Hình 1.. Lấy tên, hiện cả id



Hình 1.. Lấy tên, không hiện id

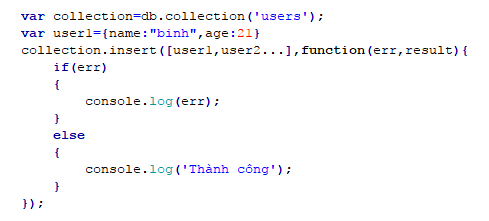
### THAO TÁC VỚI MONGODB TRONG NODEJS

* **Kết nối MongoDB với NodeJS**



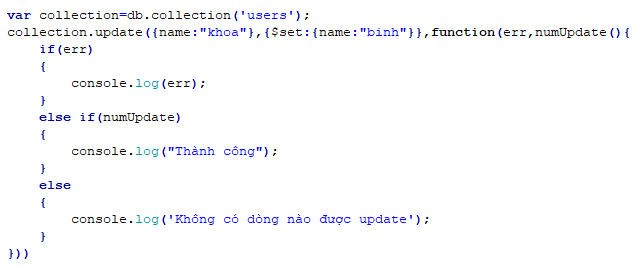
Hình 1. . Lệnh kết nối mongodb trong nodejs

* **Thao tác insert**



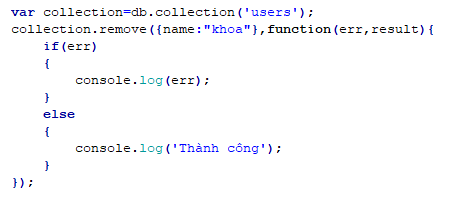
Hình 1.. Thao tác insert

* **Thao tác update**



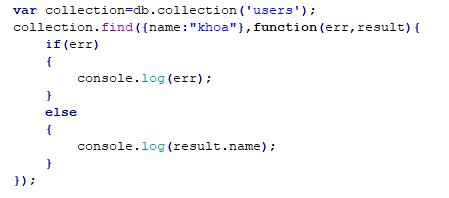
Hình 1.. Thao tác update

* Thao tác delete



Hình 1.. Thao tác delete

* Thao tác find



Hình 1.. Thao tác tìm kiếm

### KẾT CHƯƠNG

### Ưu điểm

* Schema linh hoạt: Do MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ các các kích cỡ và các document khác nhau.
* Cấu trúc đối tượng rõ ràng: Tuy rằng cấu trúc của dữ liệu là linh hoạt nhưng đối tượng của nó được xác định rất rõ ràng.
* Sử dụng bộ nhớ nội tại, nên truy vấn sẽ rất nhanh.
* MongoDB rất dễ mở rộng.
* Không có các join: Điều này cũng góp phần tạo nên tốc độ truy vấn cực nhanh trên mongoDB.
* MongoDB phù hợp cho các ứng dụng realtime.

### Nhược điểm

* MongoDB không có các tính chất ràng buộc như trong RDBMS –> dễ bị làm sai dữ liệu.
* Không hỗ trợ join giống như RDBMS nên khi viết function join trong code ta phải làm bằng tay khiến cho tốc độ truy vấn bị giảm.
* Sử dụng nhiều bộ nhớ: do dữ liệu lưu dưới dạng key-value, các collection chỉ khác về value do đó key sẽ bị lặp lại. Không hỗ trợ join nên sẽ bị thừa dữ liệu (trong RDBMS thì ta chỉ cần lưu 1 bản ghi rồi các bản ghi khác tham chiếu tới còn trong MongoDB thì không).
* Bị giới hạn kích thước bản ghi: mỗi document không được có kích thước >16Mb

## PHÂN TÍCH HỆ THỐNG BÁN HÀNG DIỆN TỬ

## KHẢO SÁT HỆ THỐNG

### MÔ TẢ HỆ THỐNG HIỆN TẠI

***Những khó khăn hiện tại***

Sau quá trình khảo sát và tìm hiểu nhóm nhận thấy hiện tại vẫn còn rất nhiều của hàng điện tử chỉ bán hàng thông qua hình thức giao dịch trực tiếp tại cửa hàng. Trên thực tế hiện nay, quá trình một khách hàng tìm tới một của hàng kinh doanh hàng điện tử mà không tìm hiểu trước thông qua internet là rất ít. Ngoài ra với cách quản lý, bán hàng truyền thống còn dẫn đến một số khó khăn sau:

* Khối lượng giấy tờ sử dụng và lưu trữ nhiều
* Thông tin về tình trạng sản phẩm hiện tại của cửa hàng hay thay đổi thường xuyên, thực hiện thủ công sẽ gây ra nhưng khó khăn cho nhân viên phục vụ
* Thông tin quản lý không đa dạng, khả năng bảo mật thấp
* Việc tra cứu tìm kiếm thông tin gặp nhiều khó khăn và tốn thời gian
* Tốn nhiều thời gian cho việc tổng hợp các báo cáo định kỳ hàng tháng
* Đòi hỏi tốn nhiều nhân lực mà hiệu quả quản lý không cao

Tuy nhiên với cách quản lý đó yêu cầu, đòi hỏi trình độ không cao, cách quản lý đơn giản mà không cần phải hiểu biết về công nghệ thông tin.

### YÊU CẦU HỆ THỐNG

Để khắc phục những nhược điểm hiện tại của một của hàng bán hàng điện tử theo cách truyền thống thì việc xây dựng một phần mềm ứng dụng công nghệ thông tin vào lĩnh vực quản lý, lưu trữ các thông tin của cửa hàng sẽ rất cần thiết nhằm mục tiêu:

* Rút ngăn thời gian làm việc của cả khách hàng và nhân viên
* Cập nhật nhanh chóng và chính xác thông tin sản phẩm và danh sách các món hàng mà khách hàng đã mua.
* Cập nhật và lưu trữ được số lượng lớn thông tin về sổ sách
* Phục vụ việc tra cứu, tìm kiếm thông tin theo yêu cầu nhanh chóng
* Thống kê được những báo cáo phục vụ việc quản lý (Báo cáo về số lượng bán của sản phẩm, báo cáo các hóa đơn trong tháng,… )

### YÊU CẦU CHỨC NĂNG

Danh sách các chức năng của hệ thống:

|  |  |
| --- | --- |
| **STT** | **Tên chức năng** |
| 1 | Đặt hàng |
| 2 | Tìm kiếm sản phẩm |
| 3 | Thanh toán |
| 4 | Tra cứu |
| 5 | Nhập hàng |
| 6 | Quản lý |
| 7 | Thống kê và in ấn |

### Đặt hàng

Lựa chọn sản phẩm theo yêu cầu của chính khách hàng

Biểu mẫu:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tên** | | **Phiếu đặt hàng** | | | |
| Ngày: | | | | | |
| **STT** | **Tên sản phẩm** | | **Thể loại** | **Đơn giá** | **Số lượng** |
| 1 |  | |  |  |  |

Biểu mẫu 2.. Biểu mẫu đặt hàng

### Tìm kiếm sản phẩm

Chức năng có nhiệm vụ tìm ra sản phẩm theo từ khóa mà khách hàng cung cấp

### Thanh toán

Tiến hành tổng thanh toán dựa trên các yêu cầu mà khách hàng đã đặt

**Biểu mẫu:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tên** | | **Hóa Đơn Thanh Toán** | | | | | |
| Nhân viên: | | | | Ngày lập: | | Mã hóa đơn: | |
| **STT** | **Tên sản phẩm** | | **Số lượng** | | **Đơn giá** | | **Thành tiền** |
| 1 |  | |  | |  | |  |
| 2 |  | |  | |  | |  |
| Tổng tiền: | | | | | | | |

Biểu mẫu 2.. Biểu mẫu hóa đơn

### Tra cứu

Tìm kiếm thông tin sản phẩm, hóa đơn, thông tin nhân viên, khách hàng

### Nhập hàng

Nhập sản phẩm vào kho chứa của cửa hàng.

### Quản lý

Chức năng dành cho người quản lý nhà hàng để quản lý thông tin khách hàng, sản phẩm, nhân viên,...

### Thống kê và in ấn

Thống kê các hóa đơn đã xuất và số lượng sản phẩm được gọi theo tháng

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tên** | | **Thống Kê Sản Phẩm** | | | |
| Tháng: | | | | Nhân viên lập: | |
| **STT** | **Tên món** | | **Thể loại** | | **Số lượng bán** |
| 1 |  | |  | |  |
| 2 |  | |  | |  |

Biểu mẫu 2.. Biểu mẫu thống kê sản phẩm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tên** | | **Thống Kê Hóa đơn** | | | |
| Tháng: | | | | Nhân viên lập: | |
| **STT** | **Mã hóa đơn** | | **Tên khách hàng** | | **Tổng tiền** |
| 1 |  | |  | |  |
| 2 |  | |  | |  |

Biểu mẫu 2.. Biểu mẫu thống kê hóa đơn

### Tổng kết

Qua các chức năng đã liệt kê ở trên, chúng ta có thể thấy nếu thực hiện theo cách truyền thống thì sẽ mất rất nhiều thời gian và công sức của cả khách hàng và nhân viên nên hiệu quả làm việc sẽ không cao. Đặt biệt là hai chức năng tra cứu và thống kê, khi cần tra cứu thông tin của một hóa đơn hay một nhân viên thì cần phải sử dụng sổ sách và tìm một cách thủ công sẽ lãng phí thời gian và đôi khi không tìm thấy. Cũng giống như tra cứu, chức năng thống kê hàng tháng là cực kỳ quan trọng và đòi hỏi phải có sự chính xác tuyệt đối nên nếu thực hiện một cách thủ công thì khả năng sai sót là rất cao. Chinh vì vậy mà một phần mềm quản lý cho cửa hàng được ra đời là việc vô cùng cần thiết để đáp ứng nhu cầu làm việc nhanh chóng và chính xác của cửa hàng để đạt được kết quả và hiệu suất cao hơn so với cách làm truyền thống.

### QUY TRÌNH NGHIỆP VỤ

Danh sách các nghiệp vụ cần giải quyết

|  |  |
| --- | --- |
| **STT** | **Tên nghiệp vụ** |
| 1 | Đặt hàng |
| 2 | Thanh toán |
| 3 | Tra cứu |
| 4 | Nhập hàng |
| 5 | Quản lý |
| 6 | Thống kê và in ấn |

**Mô tả nghiệp vụ:**

**Tìm kiếm và đặt hàng**

Khi khách truy cập vào website của cửa hàng và bắt đầu tiến hành tìm kiếm thông qua các từ khóa mà khách cung cấp hoặc tìm theo loại sản phẩm, các sản phẩm được khách hàng chọn sẽ được lưu vào giỏ hàng với số lượng tương ứng đã chọn. Các sản sẩn phẩm được chọn trong giỏ có thể điều chỉnh theo ý muốn của khách.

**Thanh toán**

Sau khi đã chọn được các sản phẩm cần mua, khách hàng tiến hành thanh toán. Khách hàng có thể chọn thanh toán thông qua nhiều hình thức khác nhau. Sau khi thanh toán, thông tin chi tiết về hóa đơn, tính trang đơn hàng sẽ được lưu lại để khách hàng có thể kiểm tra.

**Tra cứu**

+Khách hàng : khi khách có nhu cầu tìm kiếm thông tin của sản phẩm hay một hóa đơn, khách hàng sẽ cung cấp từ khóa để tìm kiếm trên website hoặc đăng nhập để kiểm tra thông tin các đơn hàng

+Nhân viên:

* Hóa đơn: Tra cứu lại thông tin chi tiết của hóa đơn về mã hóa đơn, khách hàng, nhân viên thanh toán khi có khách hàng hoặc nhân viên có yêu cầu cần tra cứu.
* Tra cứu khách hàng: Nhân viên có thể xem thông tin của các khách hàng
* Tra cứu nhân viên: Giúp nhân viên có thể tra cứu thông tin của nhau và liên lạc khi cần thiết.

**Thống kê và in ấn**

+Hóa đơn: Khi có nhu cầu cần kiểm tra lại thông tin của các hóa đơn mà cửa hàng đã phục vụ trong một tháng nào đó, nhân viên có thể chọn tháng cần xem và xuất ra để kiểm tra, lưu trữ.

+Sản phẩm: Giống như hóa đơn, nhân viên có thể xuất ra danh sách số lượng các sản phẩm đã được phục vụ trong một tháng nào đó để xem xét tính khả thi của sản phẩm và loại bỏ nếu cần thiết.

+Phiếu nhập: khi nhập nguyên liệu, các phiếu nhập sẽ được lưu lại và người dùng có thể truy xuất lại để phục vụ tùy theo nhu cầu.

**Quản lý**

Chức năng quản lý chỉ dành cho nhân viên quản trị của hệ thống

+Sản phẩm: có thể thêm mới, chỉnh sửa hoặc xóa bỏ (ẩn) thông tin của sản phẩm theo nhu cầu

+Loại sản phẩm và thương hiệu: có thể thêm mới hoặc chỉnh sửa thông tin của loại sản phẩm và thương hiệu theo nhu cầu

+Nhân viên: có thể thêm mới hoặc chỉnh sửa thông tin của nhân viên theo nhu cầu, tuy nhiên không được thay đổi email và mật khẩu.

+Khách hàng: có thể tra cứu và xem các thông tin mà khách hàng đã cung cấp khi đăng ký.

**Nhập hàng**

Khi kho sản phẩm của cửa hàng cần phải được nhập thêm, nhân viên sẽ được sự chỉ đạo của người quản lý để liên hệ với nhà cung cấp nhập hàng hoá. Sau khi nhận hàng, nhân viên căn cứ theo chứng từ từ phiếu nhập để tiến hành nhập hàng vào kho của cửa hàng và thông tin nhập kho sẽ được lưu lại.

## PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG



### MÔ HÌNH XỬ LÝ NGHIỆP VỤ BPM

### Đặt hàng

Các thông tin quan trọng trong quá trình đặt hàng là *mã hóa đơn, mã khách hàng, ngày lập hóa đơn, tên khách hàng, địa chỉ nhận hàng, số điện thoại và hình thức thanh toán.*

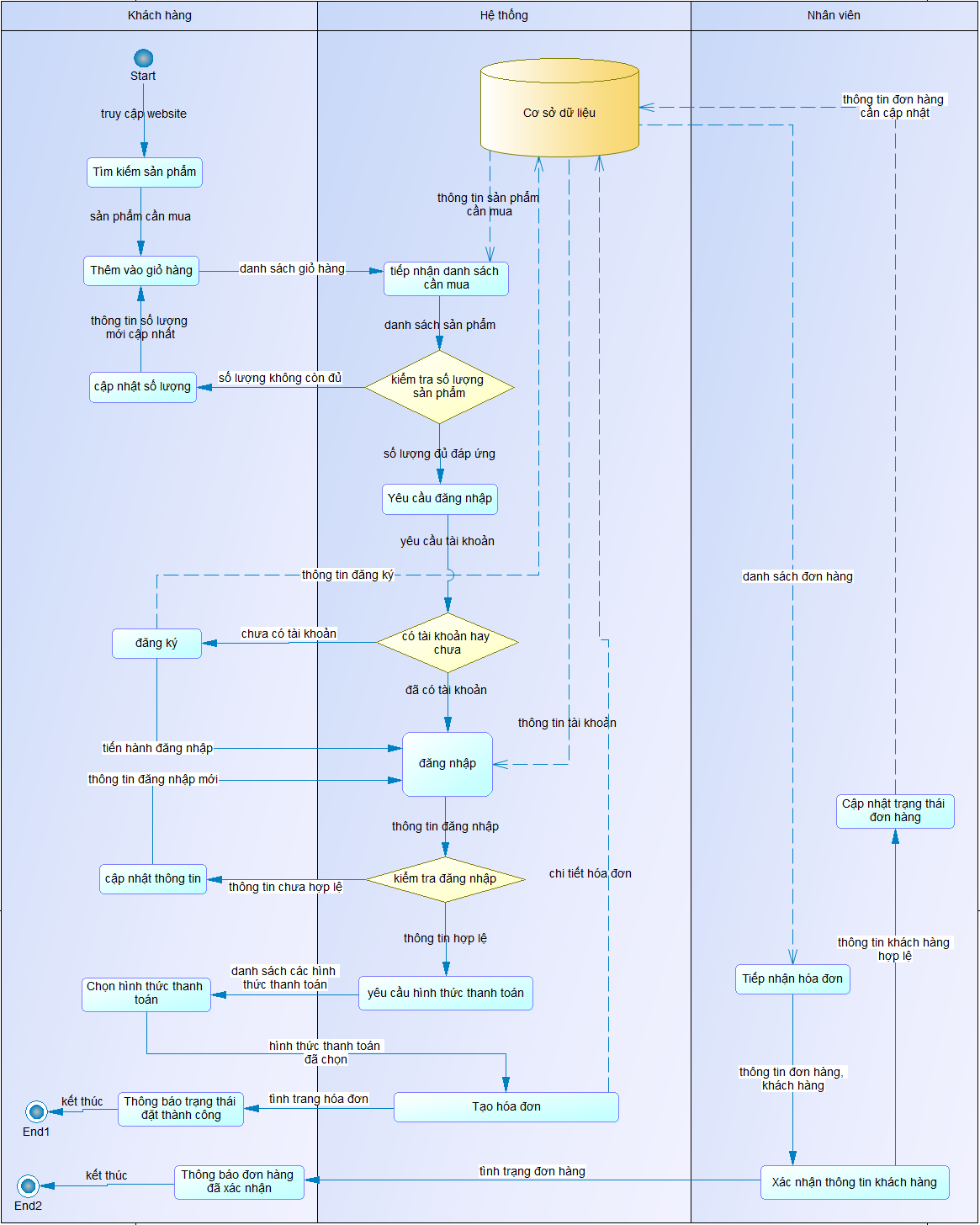
Trong mỗi hóa đơn có các chi tiết về những mặt hàng đã mua như *mã sản phẩm, tên sản phẩm, số lượng mua, giá bán.*

Bình luận chỉ cho phép người đã mua sản phẩm. Tất cả mọi người đề có thể xem được các bình luận đó.

* Mô tả

Khách hàng cần mua sản phẩm nào, sau khi truy cập vào website có thể tìm kiếm các sản phẩm theo yêu cầu cá nhân và thêm các món đồ đó vào giỏ hàng. Khi đã chọn được các món đồ cần mua cho vào giỏ và nhấn thanh toán, hệ thống sẽ kiểm tra lại số lượng tồn kho của các sản phẩm khách đã chọn, nếu số lượng không còn đủ thì sẽ yêu cầu khách hàng thay đổi số lượng hoặc chọn món khác. Ngược lại nếu số lượng thõa yêu cầu của giỏ hàng thì khách hàng sẽ tiến đến giao diện đăng nhập (nếu chưa đăng nhập) hoặc đăng ký. Sau khi đã đăng nhập thành công khách hàng được chọn hình thức thanh toán và hệ thống sẽ tạo hóa đơn với trạng thái ‘Chờ xác nhận’ và thông báo đặt hàng thành công cho khách hàng. Đơn đặt hàng này sẽ được chuyển đến nhân viên, nhân viên tiếp nhận đơn hàng, kiểm tra lại thông tin của khách hàng, nếu thông tin khách hàng là hợp lệ thì cập nhật trạng thái hóa đơn ‘Đã xác nhận’ sau đó hệ thống sẽ thông báo cho khách hàng.

* Sơ đồ



Hình 2.. BPM đặt hàng

### Nhập hàng

Các thông tin cần thiết của một sản phẩm bao gồm: *mã sản phẩm, tên sản phẩm, đơn giá, hình ảnh, số lượng, thời gian bảo hành, sản phẩm mới, nội dung sản phẩm.*

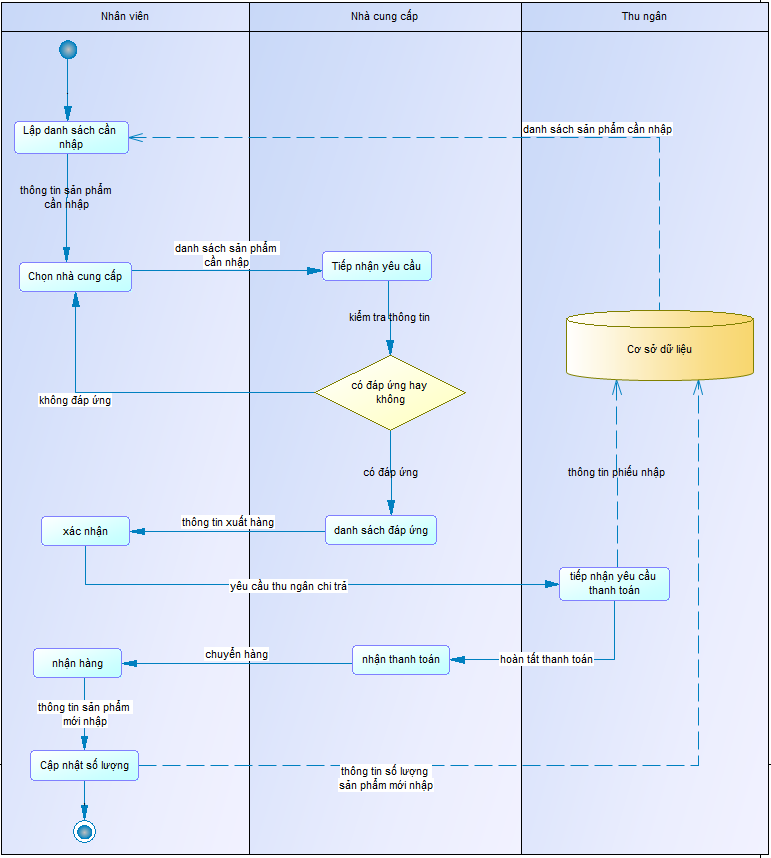
Bên cạnh đó mỗi sản phẩm còn có một trạng thái *xóa.* Ví dụ trang thái *xóa=1* sẽ ẩn thông tin của sản phẩm đó và *xóa=0* thì sản phẩm đó được phiếp hiện lên.

Ngoài ra, mỗi sản phẩm còn thuộc một loại sản phẩm và một thương hiệu nào đó. Loại sản phẩm có những thông tin như mã loại sản phẩm và tên loại sản phẩm, Thương hiệu có những thông tin như mã thương hiệu và tên thương hiệu.

* **Mô tả**

Sau khi kiểm tra số lượng sản phẩm trong kho, nhân viên tiến hành lập danh sách các sản phẩm cần nhập mới sau đó chọn nhà cung cấp để gửi yêu cầu nhập hàng. Nhà cung cấp sẽ nhận yêu cầu và danh sách các sản phẩm cần nhập, nếu nhà cung cấp không đáp ứng được danh sách các sản phẩm đó thì chọn nhà cung cấp khác. Sau khi đã chọn được nhà cung cấp, nhân viên sẽ tiến hành xác nhận thông tin nhập hàng và chuyển yêu cầu thanh toán đến cho thu ngân. Thu ngân sẽ tiếp nhận yêu cầu và thanh toán cho nhà cung cấp, thông tin hóa đơn nhập hàng sẽ được lưu vào cơ sở dữ liệu. Nhà cung cấp xác nhận thanh toán và tiến hành chuyển hàng. Nhân viên nhận hàng và cập nhật lại số lượng trong cơ sở dữ liệu.

* **Sơ đồ**



##### Hình 2.2.31 BPM nhập hàng

### Quản lý

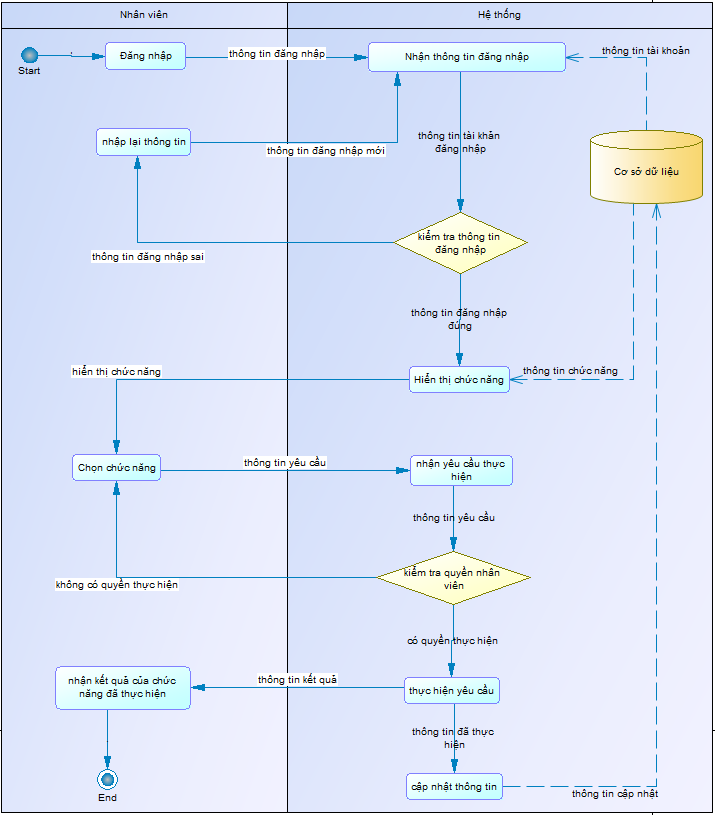
Các nội dung có thể quản lý bao gồm: *trạng thái hóa đơn, thông tin khách hàng, thông tin sản phẩm, thông tin nhân viên, thông tin phiếu nhập hàng.*

Trong mỗi nội dung quản lý, tùy thuộc vào chức vụ nhân viên mà có thể truy cập vào các chức năng quản lý và thực thi khác nhau.

* Mô tả

Để sử dụng được các chức năng của hệ thống nhân viên cần phải đăng nhập đầu tiên. Sau khi nhập xong các thông tin, hệ thống sẽ kiểm tra thông tin tài khoản có trong cơ sở dữ liệu, nếu thông tin tài khoản của người dùng không có trong dữ liệu website hệ thống sẽ thông báo và yêu cầu nhập lại thông tin. Ngược lại, hệ thống sẽ hiện các thông tin, chức năng của website cho nhân viên. Sau khi đăng nhập thành công, nhân viên có thể chọn và thực hiện các chức năng của phần mềm. Nếu tài khoản của người dùng không được phép truy cập vào chức năng đó hệ thống sẽ từ chối truy cập và chỉ nhận thực hiện các chức năng mà người dùng được cho phép. Sau khi thực hiện xong một thao tác truy vấn hệ thống sẽ hiển thị kết quả lại cho người dùng, cập nhật lại thông tin vào cơ sở dữ liệu và kết thúc quá trình.

* Sơ đồ

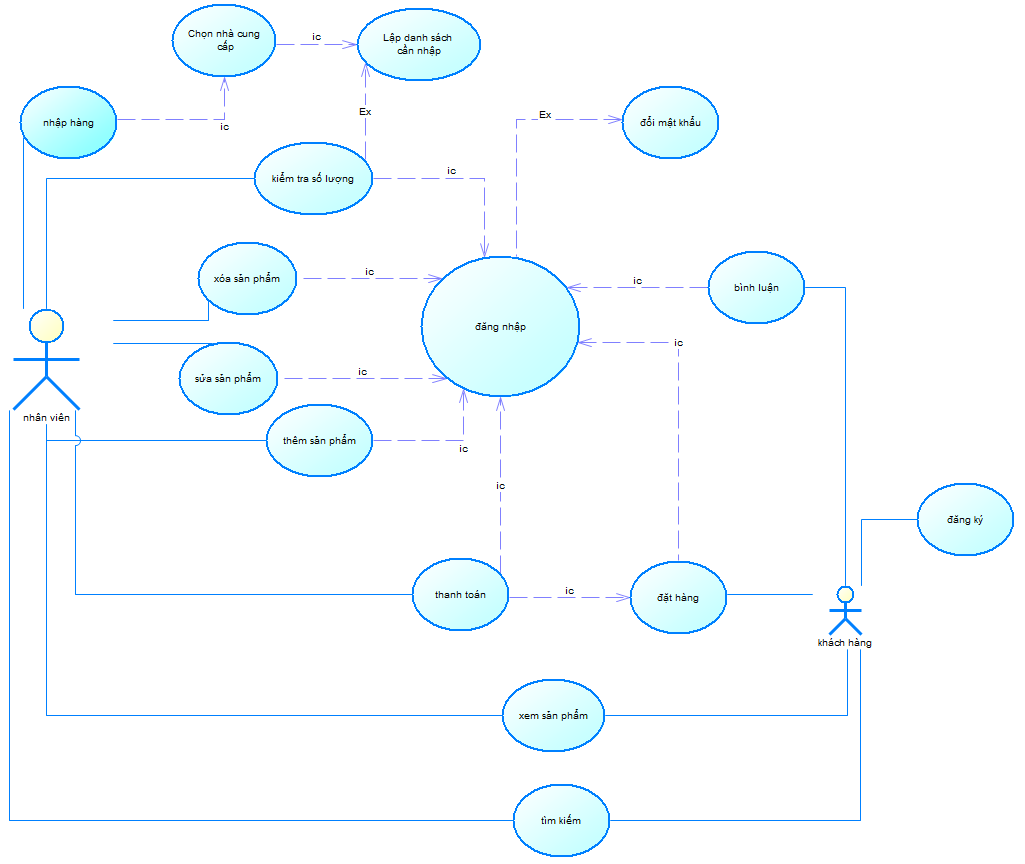


##### Hình 2.2.32 BPM quản lý

### MÔ HÌNH USECASE

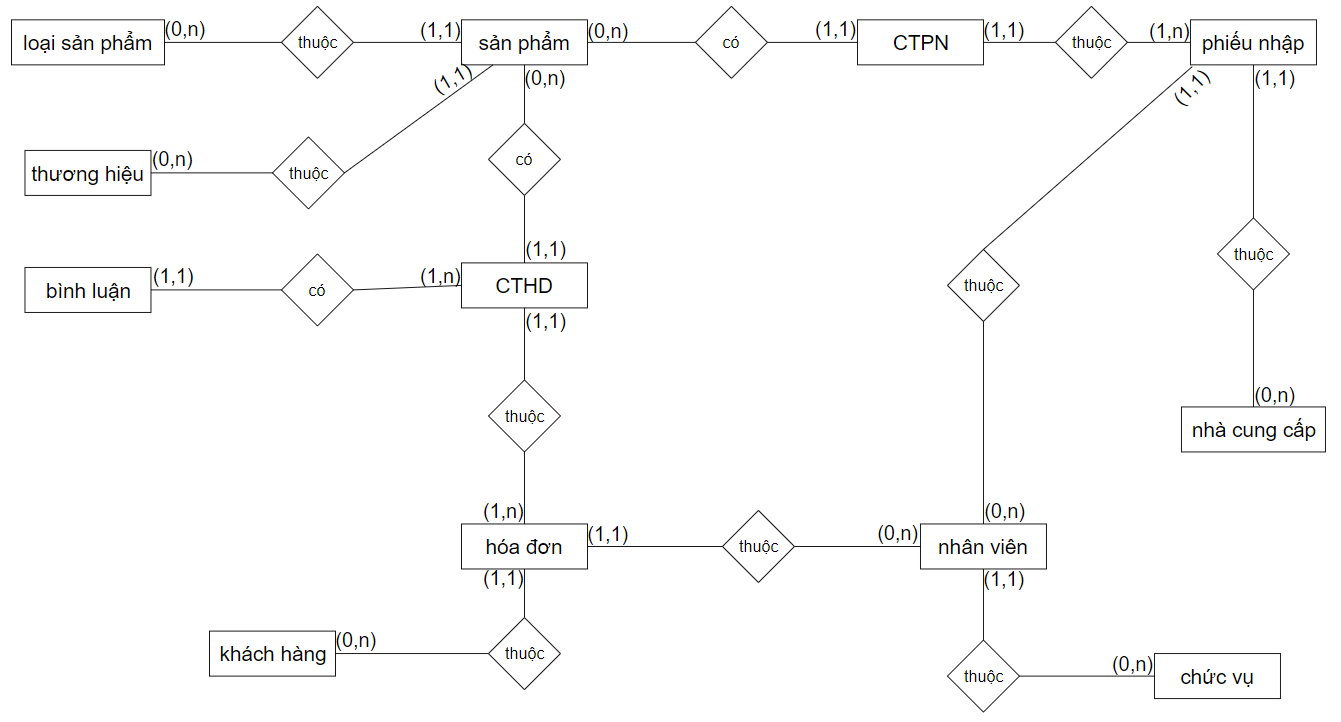
-Khách hàng: Đăng nhập, tạo tài khoản mới, xem thông tin chi tiết các sẩn phẩm, tìm kiếm sản phẩm, tiến hành đặt hàng và xem trạng thái đơn hàng.

-Nhân viên: đối với các thác tác của nhân viên trên hệ thống, trước hết cần phải đăng nhập đầu tiên, ngoài ra còn cần phải được hệ thống cấp quyền, tùy vào quyền mà có thể thực hiện các chức năng khác nhau: đổi mật khẩu, kiểm tra số lượng tồn sau đó lập danh sách và nhập hàng, thêm xóa sửa thông tin sản phẩm, xác nhận trạng thái đơn đặt hàng.

****

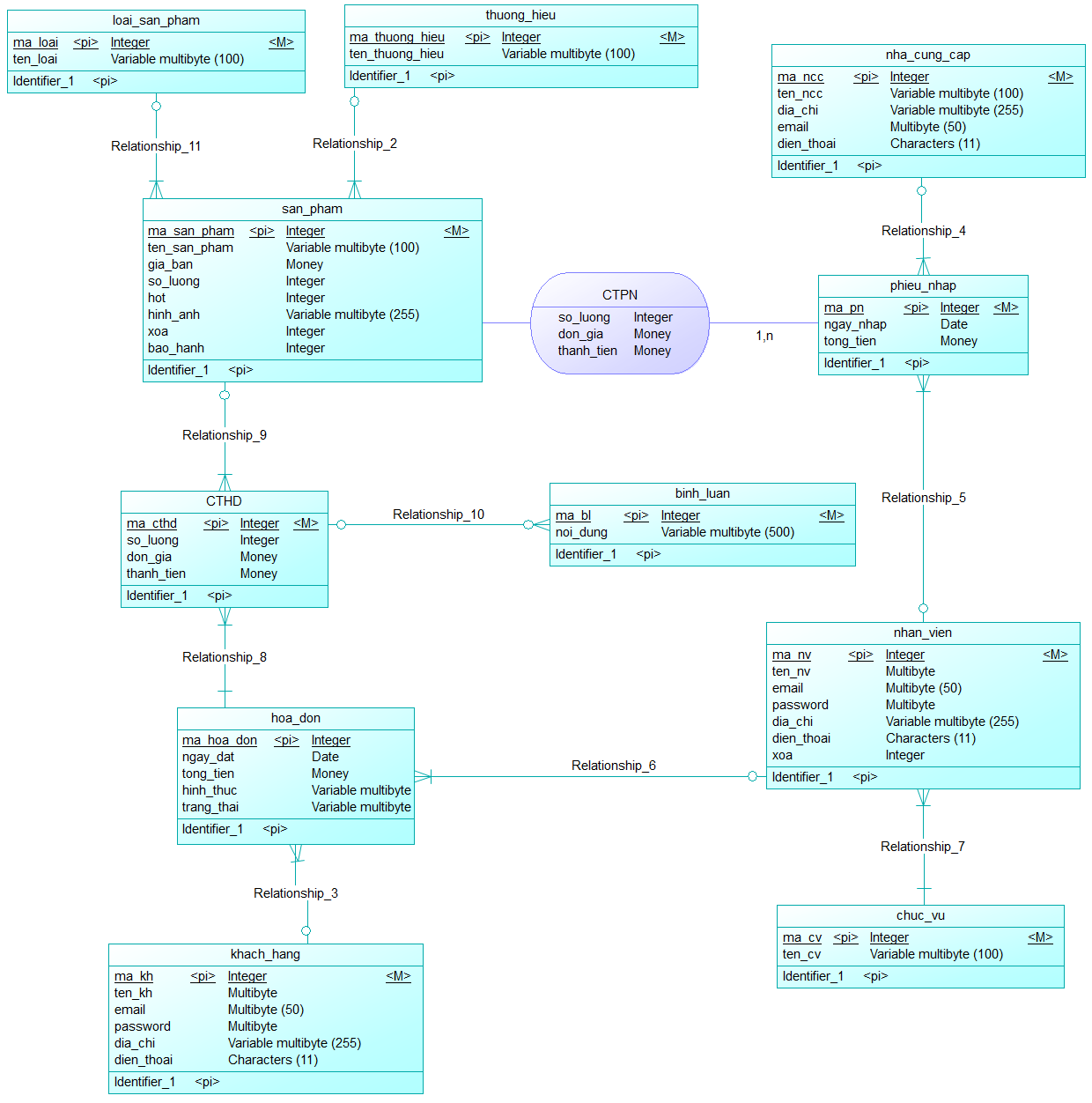
##### Hình 2.2.33 Mô hình usecase

### Mô hình thực thể kết hợp ERD



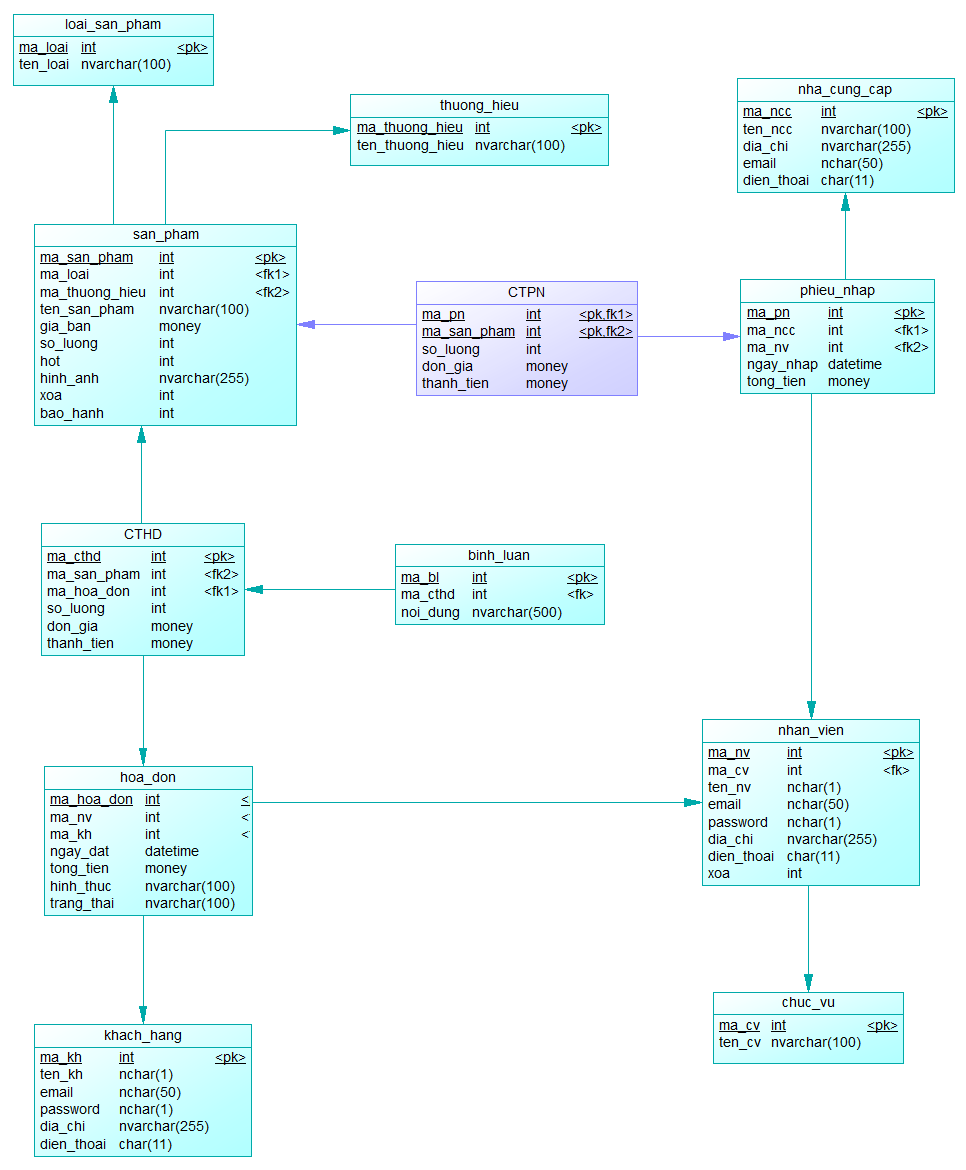
##### Hình 2.2.34 Mô hình ERD

### Mô hình cơ sở dữ liệu quan hệ CDM

****

##### Hình 2.2.35 Mô hình CDM

### Mô hình dữ liệu vật lý PDM

****

##### Hình 2.2.36 Mô hình PDM

### CÁC RÀNG BUỘC TOÀN VẸN

### Ràng buộc toàn vẹn liên bộ

* Mỗi khách hàng trong bảng khach\_hang chỉ có một email duy nhất và không được trùng nhau

∀ k1,k2 ∈ khach\_hang: k1 ≠ k2, ¬∃(k1.email=k2.email)

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| khach\_hang | + | - | +(email) |

* Mỗi nhân viên trong bảng nhan\_vien chỉ có một email duy nhất và không được trùng nhau

∀ n1,n2 ∈ nhan\_vien: n1 ≠ n2, ¬∃(n1.email=n2.email)

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| vhan\_vien | + | - | +(email) |

* Mỗi nhà cung cấp chỉ có một tên duy nhất và không được trùng nhau

∀ n1,n2 ∈ nha\_cung\_cap: n1 ≠ n2, ¬∃(n1.ten\_ncc=n2.ten\_ncc)

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| nha\_cung\_cap | + | - | +(ten\_ncc) |

* Mỗi sản phẩm chỉ có một tên duy nhất và không được trùng nhau

∀ s1,s2 ∈ san\_pham: s1 ≠ s2, ¬∃(s1.ten\_san\_pham=s2. ten\_san\_pham)

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| san\_pham | + | - | +(ten\_mon) |

### Ràng buộc toàn vẹn về miền giá trị

* Đơn giá trong bảng san\_pham phải lớn hơn 0

∀ m ∈ mon\_an: m.don\_gia>0

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| san\_pham | + | - | +(don\_gia) |

* Đơn giá trong bảng CTHD phải lớn hơn 0

∀ c ∈ CTHD: c.don\_gia>0

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| CTHD | + | - | +(don\_gia) |

* Số lượng trong bảng CTHD phải lớn hơn 0

∀ c ∈ CTHD: c.so\_luong>0

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| CTHD | + | - | +(so\_luong) |

* Đơn giá trong bảng CTPN phải lớn hơn 0

∀ c ∈ CTPN: c.don\_gia>0

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| CTPN | + | - | +(don\_gia) |

* Số lượng trong bảng CTPN phải lớn hơn 0

∀ c ∈ CTPN: c.so\_luong>0

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| CTPN | + | - | +(so\_luong) |

* Trạng thái của hoa\_don chỉ nhận ‘Chờ xác nhận’, ‘Đã xác nhận’ và ‘Đã thanh toán’

∀ h ∈ ban: h. hoa\_don =’Chờ xác nhận’ v h. hoa\_don = ‘Đã thanh toán’ v h. hoa\_don = ‘Đã xác nhận’

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| hoa\_don | + | - | +(trang\_thai) |

* Tổng tiền của hoa\_don phải lớn hơn 0

∀ h ∈ hoa\_don: h.tong\_tien>0

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| hoa\_don | + | - | +(tong\_tien) |

* Tổng tiền của phieu\_nhap phải lớn hơn 0

∀ p ∈ phieu\_nhap: p.tong\_tien>0

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| phieu\_nhap | + | - | +(tong\_tien) |

* xoa của san\_pham chỉ nhận giá trị 0 hoặc 1

∀ s ∈ san\_pham: s. xoa=0 v s.xoa= 1

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| san\_pham | + | - | +(xoa) |

* hot của san\_pham chỉ nhận giá trị 0 hoặc 1

∀ s ∈ san\_pham: s. hot=0 v s.hot= 1

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| san\_pham | + | - | +(hot) |

* bao\_hang của san\_pham phải lớn hơn 0

∀ s ∈ san\_pham: s.bao\_hanh>0

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| san\_pham | + | - | +(bao\_hanh) |

* xoa của nhan\_vien chỉ nhận giá trị 0 hoặc 1

∀ s ∈ san\_pham: s. xoa=0 v s.xoa= 1

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| nhan\_vien | + | - | +(xoa) |

### Ràng buộc toàn vẹn liên thuộc tính

* Thành tiền của một sản phẩm trong một hóa đơn phải bằng số lượng nhân đơn giá của sản phẩm đó.

∀ c ∈ CTHD: c.thanh\_tien = c.so\_luong \* t.don\_gia

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| CTHD | + | - | +(thanh\_tien,so\_luong,don\_gia) |

* Thành tiền của một sản phẩm trong một phiếu nhập phải bằng số lượng nhân đơn giá của sản phẩm đó.

∀ c ∈ CTPN: c.thanh\_tien = c.so\_luong \* t.don\_gia

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| CTPN | + | - | +(thanh\_tien,so\_luong,don\_gia) |

### Ràng buộc toàn vẹn liên bộ - liên quan hệ

* Mỗi hóa đơn phải có ít nhất một chi tiết hóa đơn

∀ h ∈ hoa\_don: ∃c ∈ CTHD: h.ma\_hoa\_don =c. ma\_hoa\_don

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| hoa\_don | + | - | +(ma\_hoa\_don) |
| CTHD | - | + | +(ma\_hoa\_don) |

* Mỗi phiếu nhập phải có ít nhất một chi tiết phiếu nhập

∀ p ∈ phieu\_nhap: ∃c ∈ CTPN: p.ma\_pn =c.ma\_pn

|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| phieu\_nhap | + | - | +(ma\_pn) |
| CTPN | - | + | +( ma\_pn) |

### Ràng buộc toàn vẹn do thuộc tính tổng hợp

* Tổng tiền của 1 hóa đơn phải bằng tổng giá trị trong các chi tiết hóa đơn

∀ h ∈ hoa\_don: h.tong\_tien = Σ ( c ∈ CTHD ∧ c.ma\_hoa\_don = h.ma\_hoa\_don) c.so\_luong\*c.don\_gia

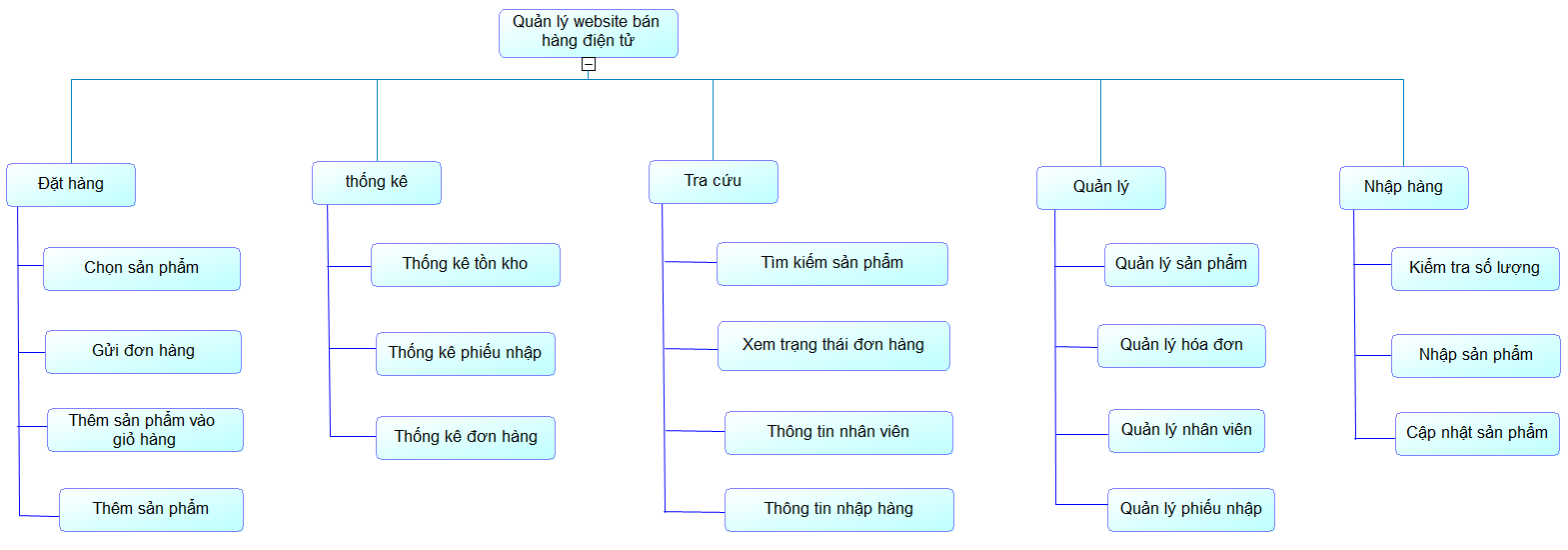
|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| hoa\_don | + | - | +(tong\_tien) |
| CTHD | + | + | +(so\_luong,don\_gia) |

* Tổng tiền của 1 phiếu nhập phải bằng tổng giá trị trong các chi tiết phiếu nhập

∀ p ∈ phieu\_nhap: p.tong\_tien = Σ (c ∈ CTPN ∧ c.ma\_pn = p.ma\_pn) c.so\_luong\*c.don\_gia

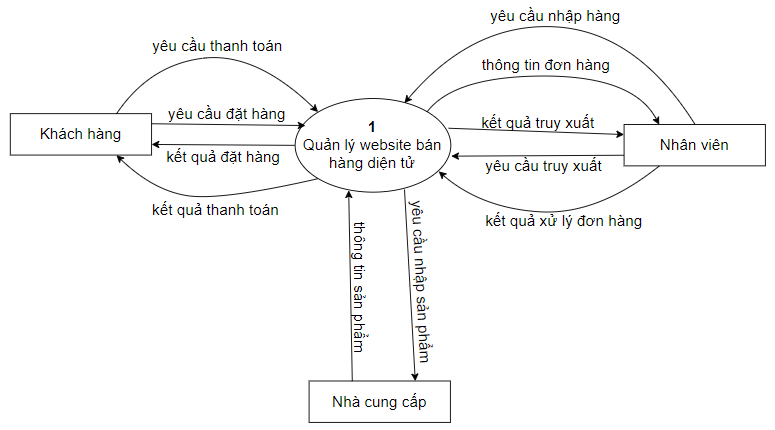
|  |  |  |  |
| --- | --- | --- | --- |
| R | Thêm | Xóa | Sửa |
| phieu\_nhap | + | - | +(tong\_tien) |
| CTPN | - | + | +(so\_luong,don\_gia) |

### MÔ HÌNH CHỨC NĂNG BFD



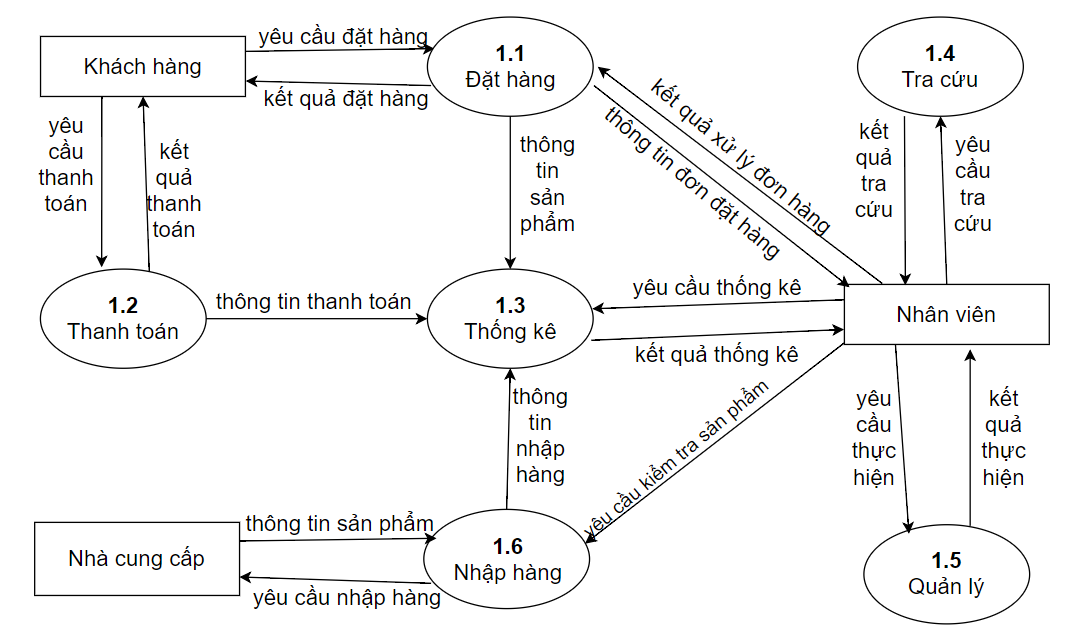
##### Hình 2.2.37 Mô hình BFD

### MÔ HÌNH XỬ LÝ DFD



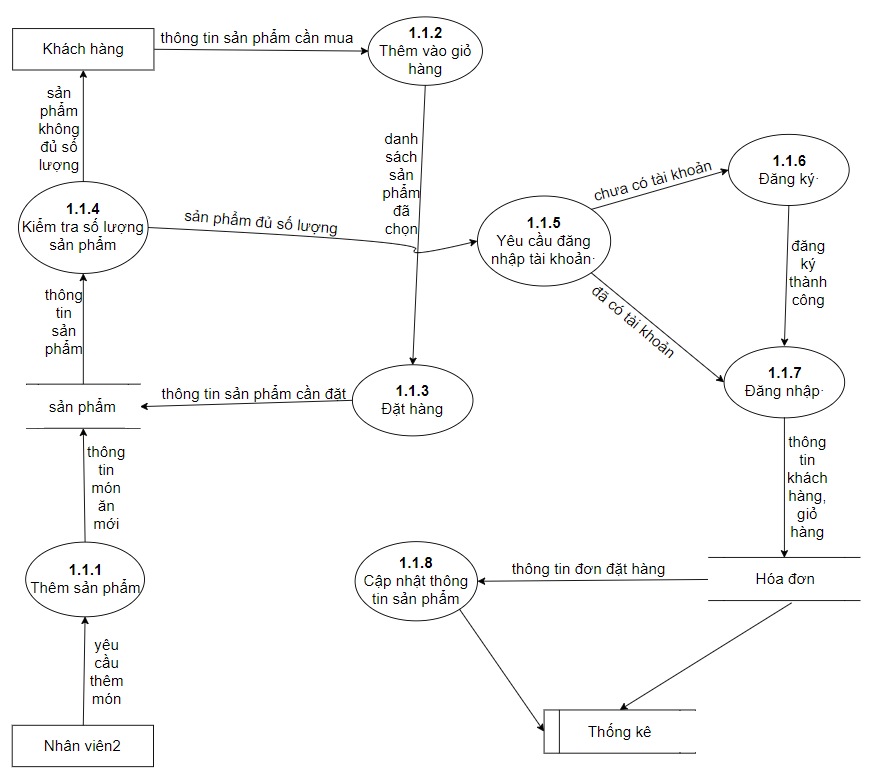
##### Hình 2.2.38 DFD đỉnh

* Phân rã ô xử lý 1 (Quản lý website bán hàng điện tử)



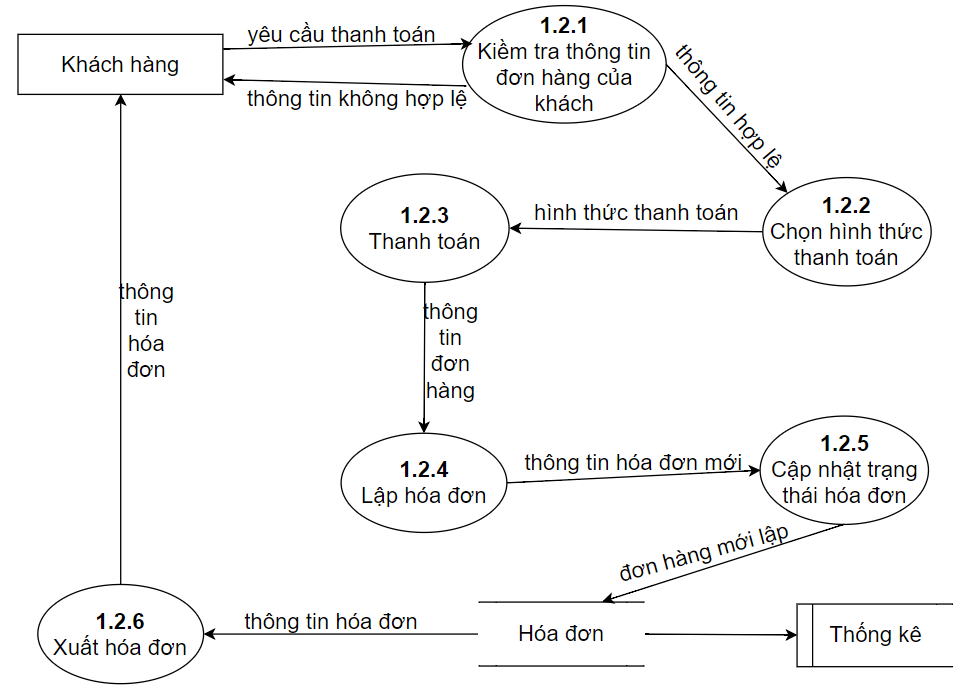
##### Hình 2.2.39 DFD mức 1

* Phân rã ô xử lý 1.1 (Đặt hàng)

****

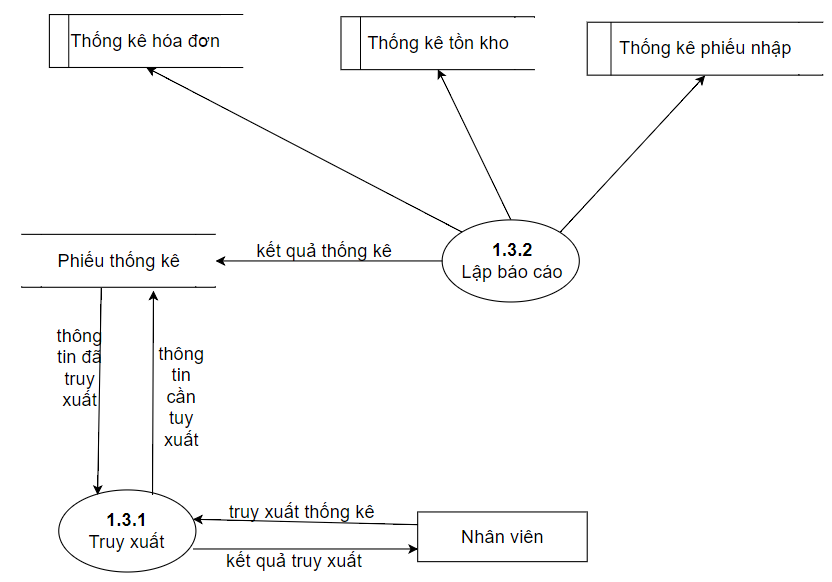
##### Hình 2.2.40 DFD đặt hàng

* Phân rã ô xử lý 1.2 (Thanh toán)

****

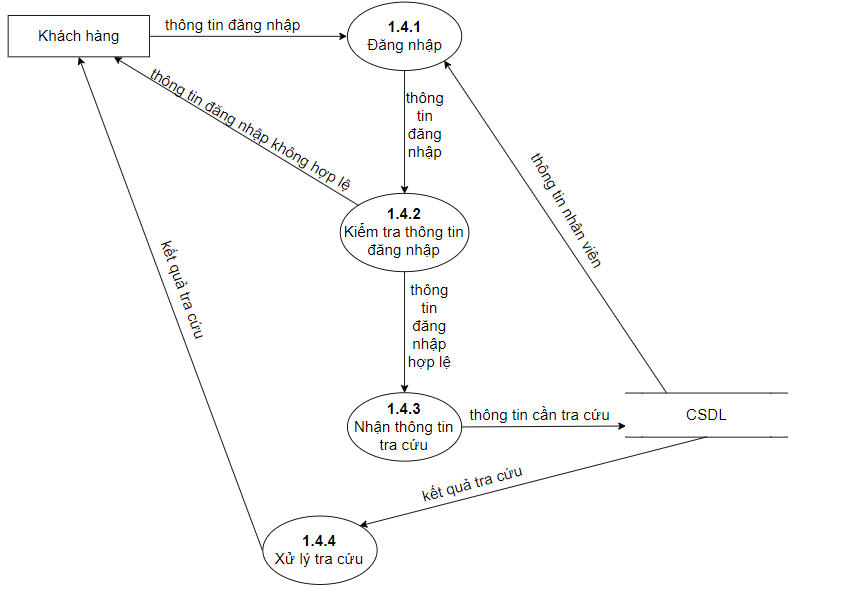
##### Hình 2.2.41 DFD thanh toán

* Phân rã ô xử lý 1.3 (Thống kê)

****

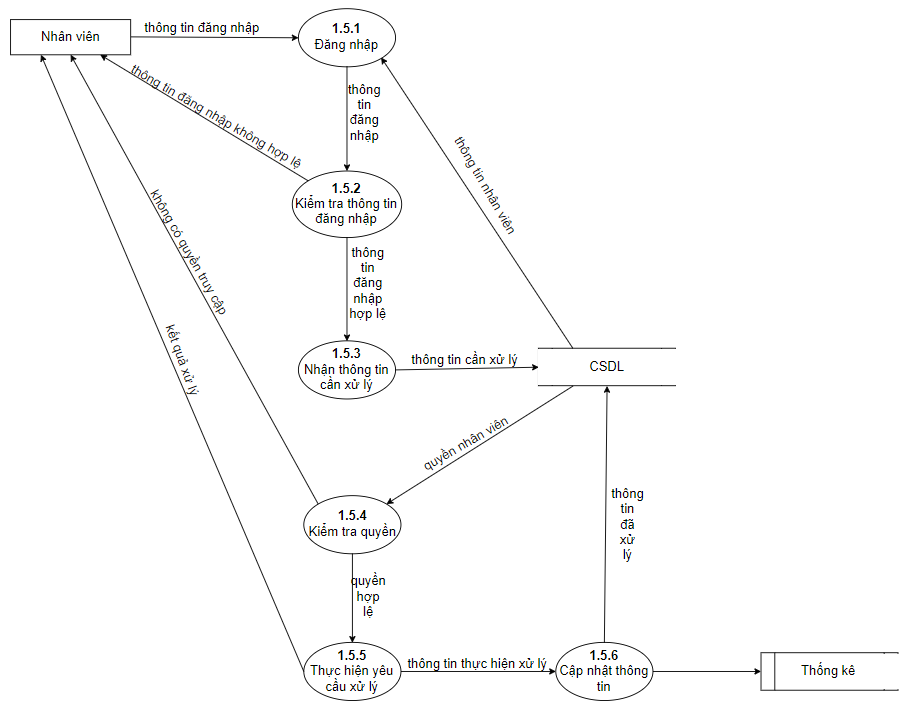
##### Hình 2.2.42 DFD thống kê

* Phân rã ô xử lý 1.4 (Tra cứu)

****

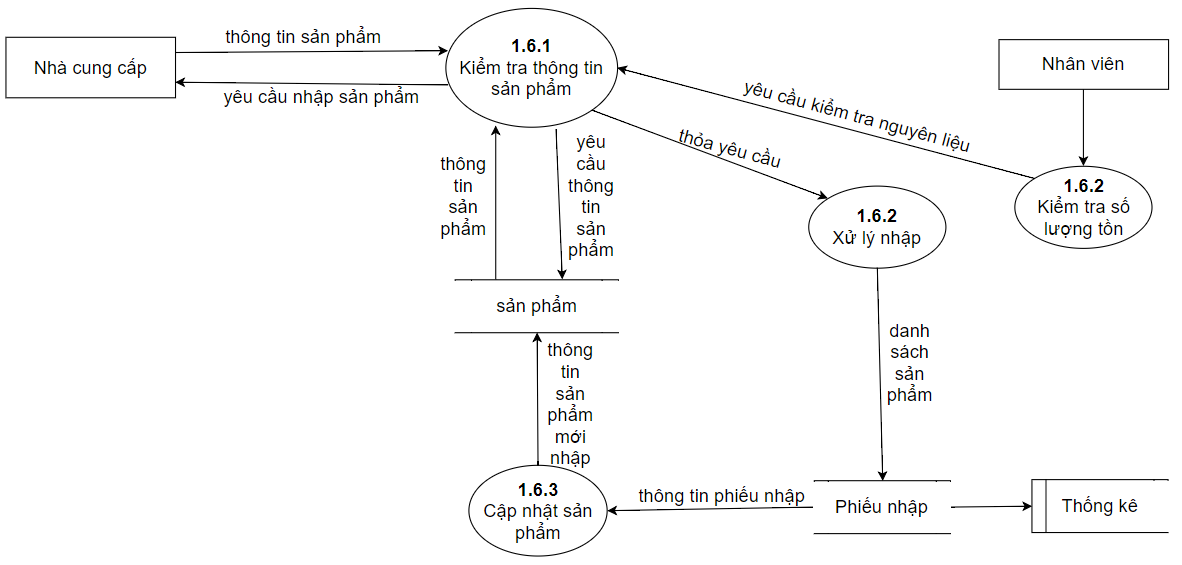
##### Hình 2.2.43 DFD tra cứu

* Phân rã ô xử lý 1.5 (Quản lý)

****

##### Hình 2.2.44 DFD quản lý

* Phân rã ô xử lý 1.6 (Nhập hàng)

****

##### Hình 2.2.45 DFD nhập hàng

## TRIỂN KHAI HỆ THỐNG WEBSITE NODEJS

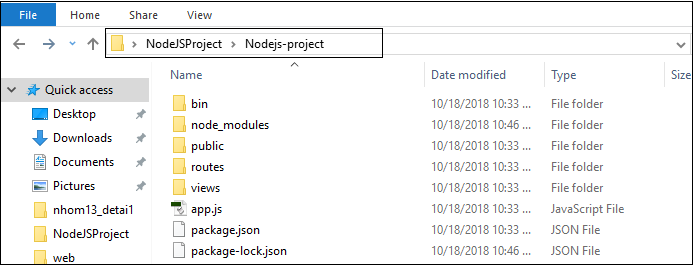


### TẠO PROJECT

Khởi tạo project [[2]](#footnote-2) bằng lệnh *npm init – y* . Xây dựng project dựa trên ejs template, các gói cài đặt được sử dụng bao gồm:

* Express
* Express-generator
* Nodemon
* MongoDB
* Mongoose

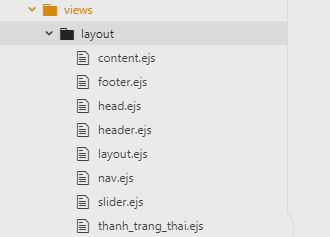
Project đã được khởi tạo thành công.



Hình 2.. Tạo project NodeJS

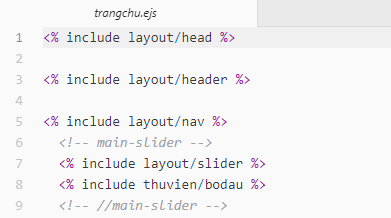
### TẠO VÀ CÁT GIAO DIỆN HTML THÀNH EJS TEMPLATE

Chia nhỏ các thành phần của trang html thành nhiều phần và để trong thư mục view với phần đuôi mở rộng là ejs vì đây là thành phần view của project sử dụng ejs template.



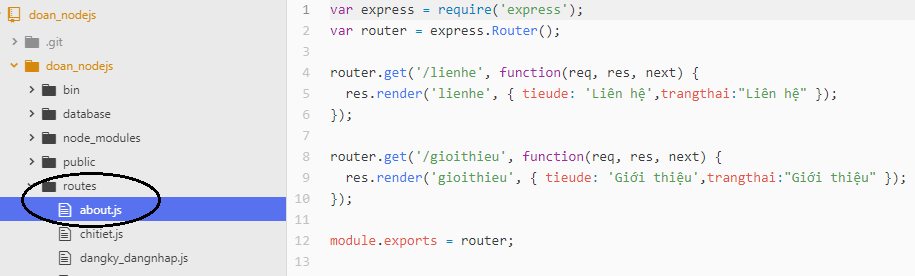
Hình 2.2.47 Trang html sau khi chia nhỏ

Sau khi chia nhỏ các phần của file html, ta tiến hành ghép chúng lại bằng cách sử dụng *include* của ejs. Một trang view hoàn chỉnh sẽ bao gồm các thành phần chính của trang html như header, footer. Các thành phần riêng biệt của từng trang sẽ được xây dựng riêng sau có include các file đã được chia nhỏ trước đó mà không cần phải viết lại từ đầu.



Hình 2.2.48 câu lệnh include trong file .ejs

Sau khi đã hoàn thành các view, ta tiến hành tạo Route [[3]](#footnote-3) cho ứng dụng.



Hình 2.2.49 Tạo route cho trang giới thiệu và liên hệ

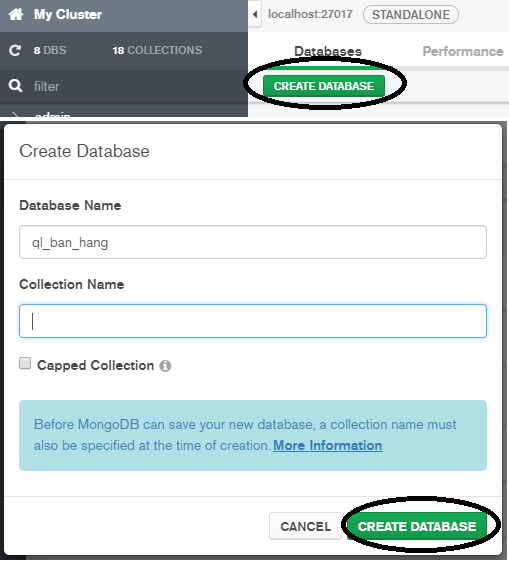
Sau khi route đã tạo xong thì cần phải được cấu hình trong file app.js



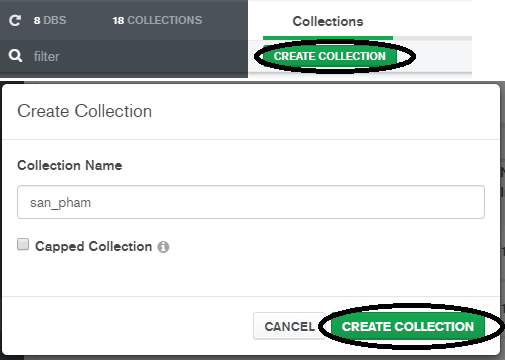
Hình 2.2.50 Thiết lập route trong file app.js

### TẠO CƠ SỞ DỮ LIỆU MONGODB

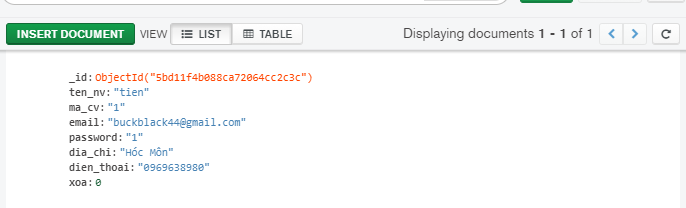
Dữ liệu mẫu được nhập trực tiếp từ giao diện của ứng dụng MongoDB Compass. Bắt đầu bằng việc tạo database: nhập tên database muốn tạo, nhập tên collection nếu muốn tạo cùng lúc với database.

****

Hình 2.2.51 Giao diện tạo database



Hình 2.2.52 Giao diện tạo collection



Hình 2.2.53 Giao diện thêm document

### KẾT NỐI NODEJS VỚI MONGODB

* **Kết nối bằng MongoClient**



Hình 2.2.54 Lấy dữ liệu bằng MongoClient

* **Kết nối bằng Mongoose**

Đối với cách kết nối này, trước tiên cần phảo tạo Schema[[4]](#footnote-4)



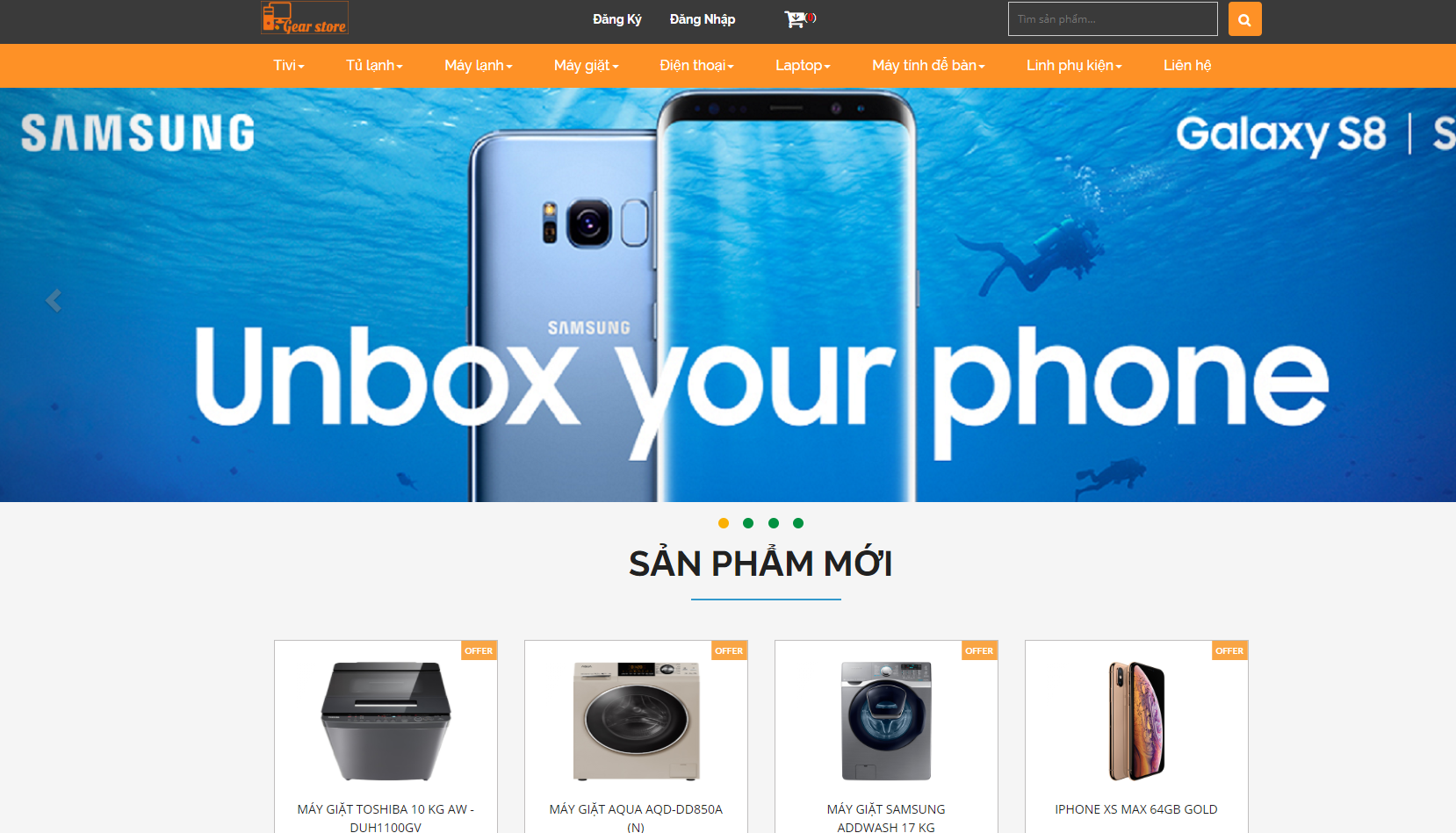
Hình 2.2.55 Tạo schema collection san\_pham

Sau khi đã có Schema dữ liệu sẽ được truy vấn như sau:

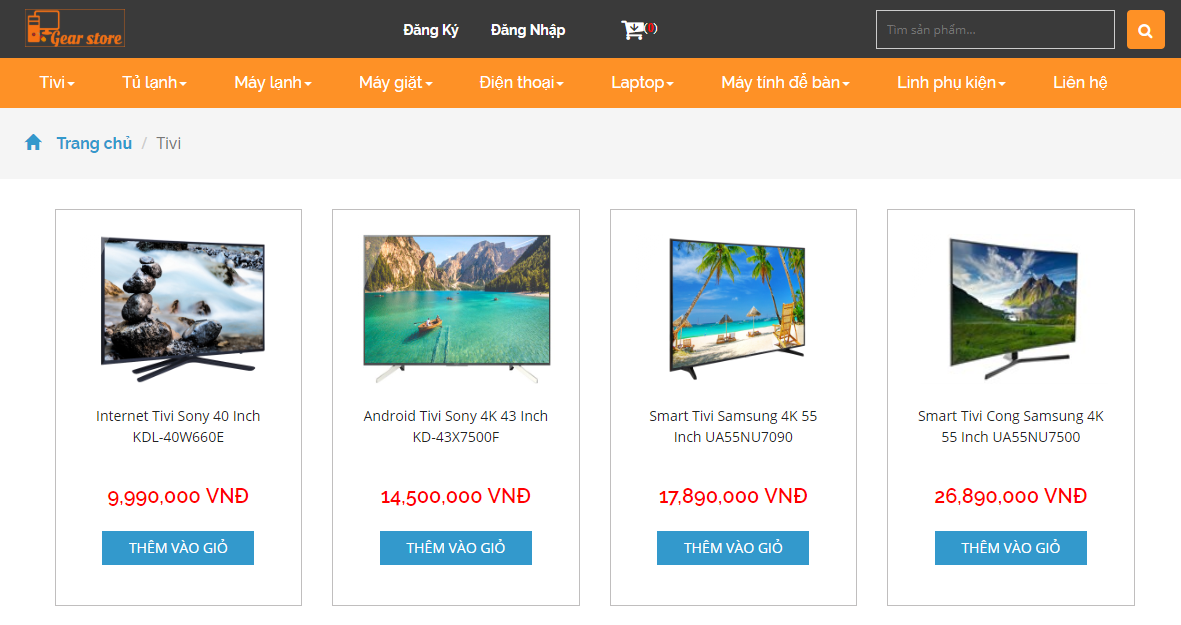


Hình 2.2.56 Lấy dữ liệu bằng mongoose

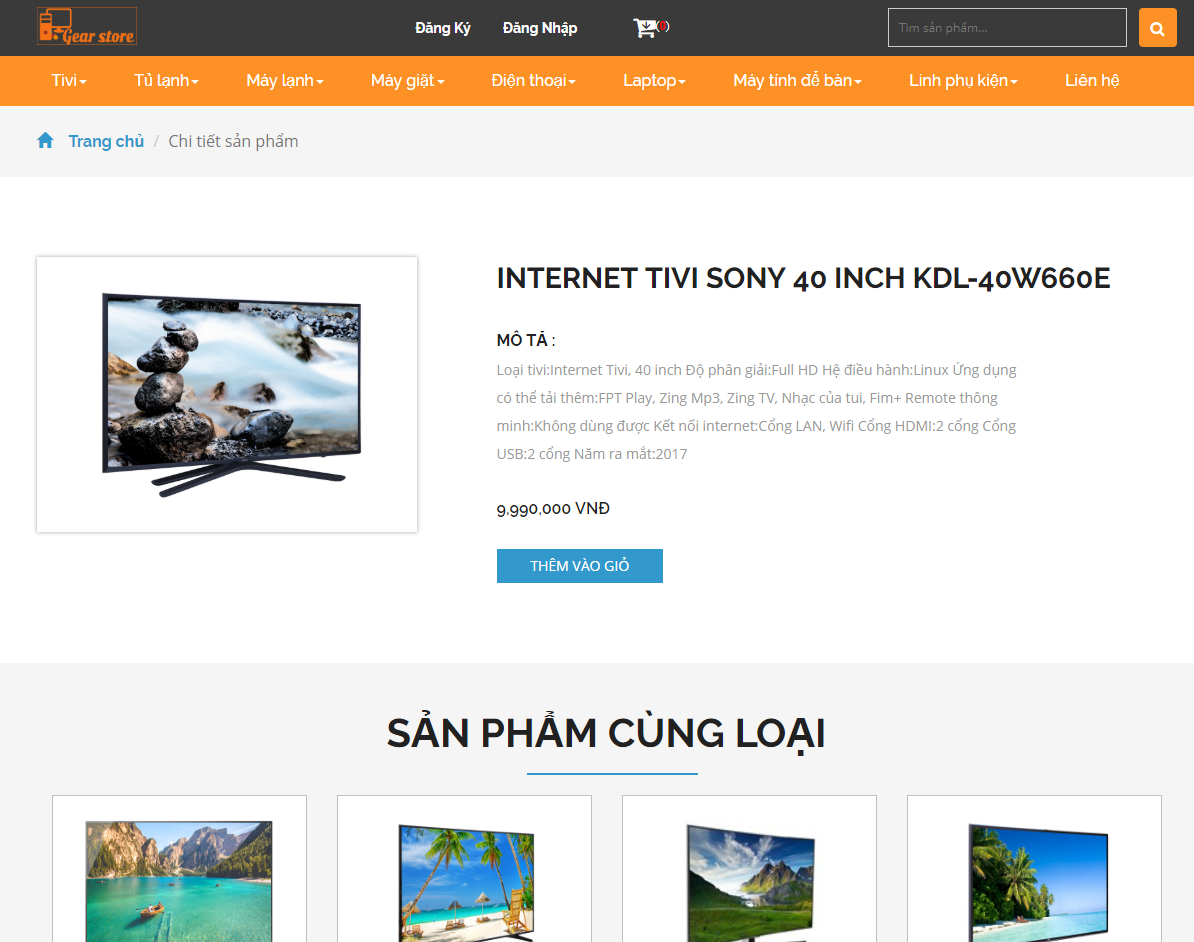
### KẾT QUẢ THỰC HIỆN



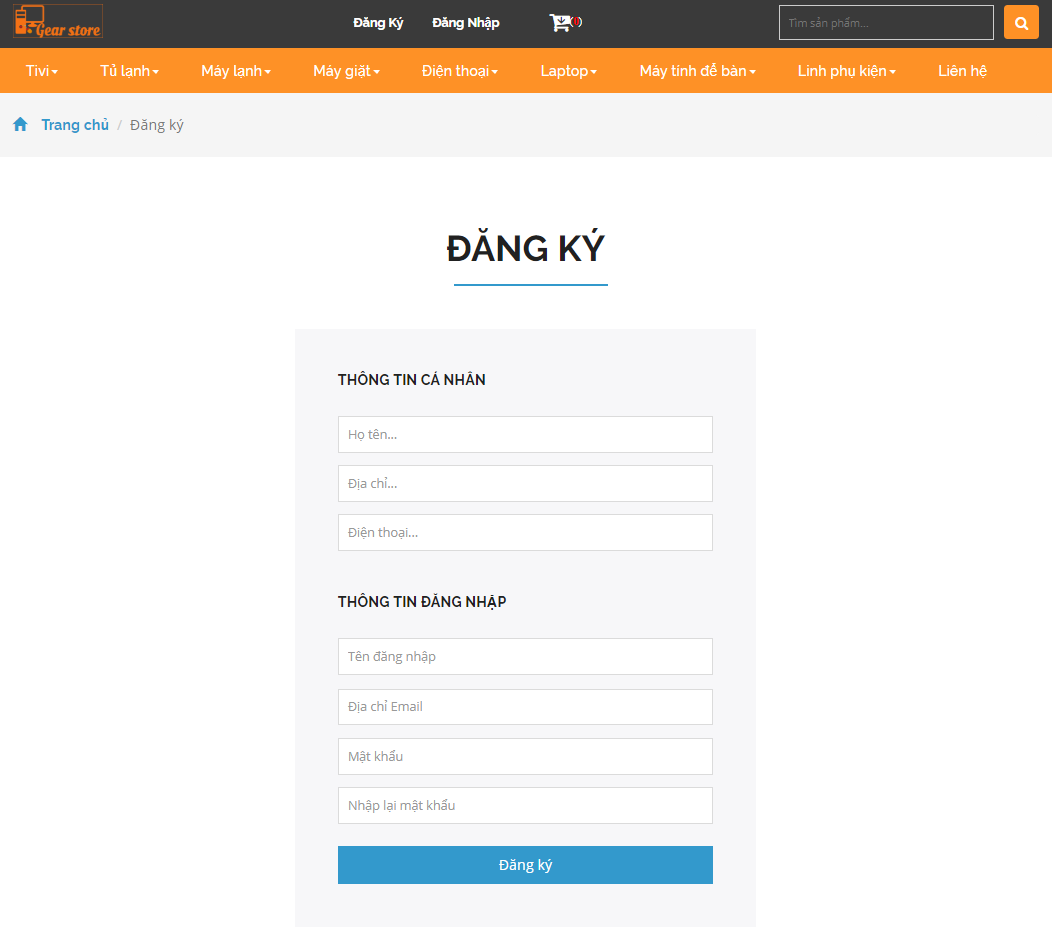
Hình 2.2.57 Giao diện trang chủ



Hình 2.2.58 Giao diện sản phẩm theo từng loại



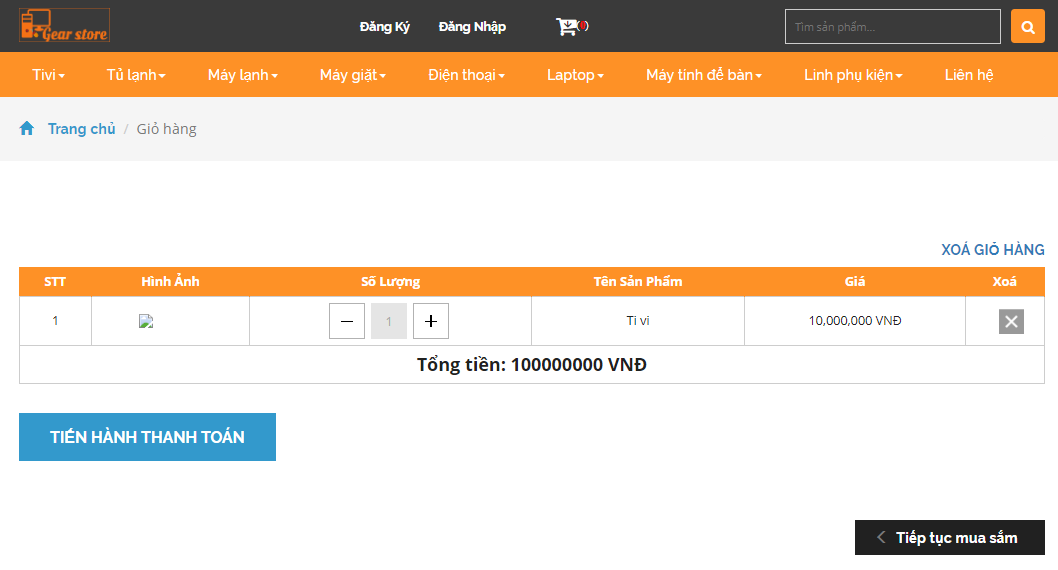
Hình 2.2.59 Giao diện chi tiết sản phẩm



Hình 2.2.60 Giao diện đăng ký



Hình 2.2.61 Giao diện đăng nhập



Hình 2.2.62 Giao diện giỏ hàng

# KẾT LUẬN

NodeJS hiện nay đang phát triển với tốc độ nhanh cùng với đội ngũ người sử dụng vô cùng lớn nên khả năng hỗ trợ lẫn nhau là cực kỳ cao. Ngoài ra điểm chính của NodeJS chính là việc xử lý bất đồng bộ sẽ hạn chế khả năng nghẽn mạng do nhiều request cùng một thời điểm. Còn đối với MongoDB, nó là hệ cơ sở dữ liệu không quan hệ nên việc truy vấn dữ liệu sẽ trở nên linh hoạt. Tuy nhiên bên cạnh những mặt lợi thì NodeJS và cả MongoDB vẫn còn tồn tại những hạn chê nhất định đặc biệt là việc sinh sau đẻ muộn so với các nền tảng khác sẽ khiến cho NodeJS và MongoDB thường xuyên nhận các cập nhật mà bản sau có thể không kế thừa được bản trước. Thông qua đồ án này, nhóm em đã rút ra được một số điểm như sau:

* **Những nội dung đạt được**
* Cơ chế hoạt động của NodeJS
* Sự tương quan giữa MongoDB với cơ sở dữ liệu có quan hệ
* Kết nối MongoDB với NodeJS
* Xây dựng được mô hình cơ sở dữ liệu và truy vấn trên MongoDB
* Thiết kế được giao diện bán hàng cho phân hệ khách hàng
* Khảo sát, phân tích được hệ thống bán hàng điện tử
* Xây dựng được các mô hình hoạt động của hệ thống bán hàng diện tử
* **Những hạn chế của đề tài**
* Giao diện cho khách hàng chưa đẹp và tối ưu
* Quá trình khảo sát chỉ dừng lại ở mức tham khảo các website bán hàng online
* **Hướng phát triển**
* Xây dựng hoàn chỉnh các chức năng của website
* Xây dựng thêm hệ thống website quản lý dành cho phân hệ quản lý
* Đưa hệ thống đi vào hoạt động thực tế

# TÀI LIỆU THAM KHẢO

**Tiếng Việt**

x

|  |  |
| --- | --- |
| [1] | Trường Đại học FPT, *Javascript - Hướng dẫn học qua ví dụ*.: Nhà xuất bản Bách Khoa Hà Nội. |
| [2] | ThS. Nguyễn Văn Lễ ThS. Hoàn Thị Liên Chi, *Cơ sở dữ liệu*.: Trường Đại học Công nghiệp thực phẩm TP.HCM. |
| [3] | ThS. Nguyễn Văn Lễ ThS. Hoàn Thị Liên Chi, *Phân tích thiết kế hệ thông thông tin*.: Trường Đại học Công nghiệp thực phẩm TP.HCM.  **Tiếng Anh** |
| [4] | Ethan Brown, *Web Development with Node and Express*., 2014. |
| [5] | MarkLogic, *Node.js Application Developer’s Guide*., 2017. |
| [6] | Krasimir Tsonev, *Node.js By Example*., 2015. |
| [7] | Sandro Pasquali, *Mastering Node.js*., 2013.  **Website** |
| [8] | Tran Quoc Dat. (2017) Giới thiệu về NoSQL database. [Online]. <https://viblo.asia/p/gioi-thieu-ve-nosql-database-djeZ1a9jZWz> |
| [9] | BUI XUAN HIEN. bodyParser() trong Express.js. [Online]. <https://techblog.vn/bodyparser-trong-expressjs> |
| [10] | Nguyen Van Hien. (2016) Tổng quan về MongoDB. [Online]. <https://viblo.asia/p/tong-quan-ve-mongodb-EoDkQoxqGbV> |
| [11] | Han Van Hiep. (2016) Sử dụng MongoDB với NodeJs. [Online]. <https://viblo.asia/p/su-dung-mongodb-voi-nodejs-ZDEeLXRoeJb> |
| [12] | Phan Van Hieu. (2018) NodeJS căn bản. [Online]. <https://viblo.asia/p/nodejs-can-ban-phan-1-WAyK8MXWZxX> |
| [13] | Nguyen Xuan Huong. (2018) ES6: var, let and const — The battle between function scope and block scope. [Online]. <https://viblo.asia/p/es6-var-let-and-const-the-battle-between-function-scope-and-block-scope-6J3Zg2aPKmB> |
| [14] | Hồ Sỹ Hùng. Node.js cho người mới bắt đầu. [Online]. <https://techmaster.vn/posts/33448/nodejs-cho-nguoi-moi-bat-dau> |
| [15] | KhoaPham. (2018) Khoa Phạm Training. [Online]. <http://online.khoapham.vn/search?q=nodejs> |
| [16] | Le Van Liem. (2017) Tìm hiểu Express js Framework. [Online]. <https://viblo.asia/p/phan-1-tim-hieu-express-js-framework-Qbq5Qq7m5D8> |
| [17] | LAM PHAM. (2018) PHÂN BIỆT ARROW FUNCTION VÀ FUNCTION TRONG JAVASCRIPT. [Online]. <https://completejavascript.com/phan-biet-arrow-function-va-function-trong-javascript/> |
| [18] | Hoang Si. (2018) Cài đặt các Node Modules cần thiết để chạy ứng dụng với NodeJS. [Online]. <https://hoangsi.com/cai-dat-cac-node-modules-can-thiet-de-chay-ung-dung-voi-nodejs.html> |
| [19] | Techmaster. (2015) Node.js là gì? [Online]. <https://techmaster.vn/posts/33428/nodejs-la-gi-va-tai-sao-toi-nen-hoc-lap-trinh-nodejs> |
| [20] | TheHalfHeart. (2016) Arrow Function trong ES6. [Online]. <https://freetuts.net/arrow-function-trong-es6-606.html> |
| [21] | TheHalfHeart. (2014) Tổng quan về MongoDB. [Online]. <https://freetuts.net/tong-quan-ve-mongodb-203.html> |
| [22] | Vũ Thanh Tài. Truy vấn dữ liệu trong MongoDB. [Online]. <https://toidicode.com/truy-van-du-lieu-trong-mongodb-289.html> |
| [23] | vietjack. (2018) Học MongoDB cơ bản và nâng cao. [Online]. <https://vietjack.com/mongodb/> |
| [24] | vietjack. (2018) Học Node.js cơ bản và nâng cao. [Online]. <https://vietjack.com/nodejs/> |
| [25] | MongoDB. MongoDB Documentation. [Online]. <https://docs.mongodb.com/> |
| [26] | npm lnc. (2018) npm - npm. [Online]. <https://www.npmjs.com/> |

x

1. Authen: cơ chế xác thực người dùng [↑](#footnote-ref-1)
2. Chi tiết cách cài đặt các gói nẳm ở phần 1 Tổng quan về NodeJS mục 1.2 [↑](#footnote-ref-2)
3. Route: Đường dẫn trên thanh địa chỉ trình duyệt, trang xử lý [↑](#footnote-ref-3)
4. Schema sẽ quy định cấu trúc cho collection được truy vấn [↑](#footnote-ref-4)