**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO BÀI TẬP LỚN**

**Môn học: Lập trình hướng đối tượng**

**Đề tài: Xây dựng website bán điện thoại**

|  |  |
| --- | --- |
| **Giảng Viên** | **: Nguyễn Mạnh Sơn** |
| **Nhóm** | **: 13** |
| **Nhóm BTL** | **: 03** |
| **Thành Viên** | **: B22DCCN847 - Nguyễn Tiến Thực**  **: B22DCCN607 - Nguyễn Việt Pháp**  **: B22DCCN139 - Trần Đức Dũng** |

***Hà Nội – 2024***

Mục Lục

[Chương 1: giới thiệu bài toán 4](#_Toc183201379)

[I. Đặt vấn đề 4](#_Toc183201380)

[II. Đề xuất giải pháp 5](#_Toc183201381)

[1. Thiết Kế Giao Diện Người Dùng (UI/UX) 5](#_Toc183201382)

[2. Tích Hợp Chức Năng Cần Thiết 5](#_Toc183201383)

[Chương 2: Kỹ thuật công nghệ sử dụng 6](#_Toc183201384)

[I. Git/ GitHub 6](#_Toc183201385)

[II. Database 6](#_Toc183201386)

[III. Server 7](#_Toc183201387)

[IV. Backend 8](#_Toc183201388)

[1. SERVLET 8](#_Toc183201389)

[2. JDBC (Java Database Connectivity) 9](#_Toc183201390)

[V. Frontend 9](#_Toc183201391)

[1. JSP (Java Server Pages) 9](#_Toc183201392)

[2. HTML/ CSS/ JS 1](#_Toc183201393)0

[3. Boostraps 1](#_Toc183201394)1

[Chương 3: Kiến trúc hệ thống 1](#_Toc183201395)2

[I. Database (SQL Server) 1](#_Toc183201396)2

[1. Thiết kế Database 1](#_Toc183201397)2

[2. Tạo các bảng, các ràng buộc, chỉ mục: 1](#_Toc183201398)2

[3. Cung cấp dữ liệu cho bảng 1](#_Toc183201399)5

[II. Hệ thống phía Backend 1](#_Toc183201400)8

[1. Model 19](#_Toc183201401)

[2. Dal 22](#_Toc183201402)

[3. Controller.client 25](#_Toc183201403)

4. Controller.admin..............................................................................................................................28

[III. Hệ thống phía Frontend 3](#_Toc183201404)0

[Chương 4: Chạy và thử nghiệm trang Web 3](#_Toc183201405)2

[I. Trang home của web 3](#_Toc183201406)2

[II. Trang xem chi tiết điện thoại: 3](#_Toc183201407)3

[III. Trang xem chi tiết giỏ hàng 3](#_Toc183201408)5

[IV. Trang xem chi tiết đơn hàng 3](#_Toc183201409)5

[V. Trang thông tin cá nhân 3](#_Toc183201410)7

[VI. Trang đăng ký, đăng nhập 3](#_Toc183201411)8

**VII. Trang Admin** **........................................................................................................................39**

[VIII. Đánh giá và hướng phát triển trong tương lai 4](#_Toc183201412)2

[1. Ưu điểm 4](#_Toc183201413)2

[2. Nhược điểm 4](#_Toc183201414)2

[3. Hướng phát triển trong tương lai 4](#_Toc183201415)2

[TÀI LIỆU THAM KHẢO 4](#_Toc183201416)3

**Lời mở đầu**

Lời đầu tiên, chúng em không biết nói gì hơn ngoài bày tỏ lòng biết ơn sâu sắc đến thầy bằng việc gửi lời cảm ơn chân thành và sâu sắc đến giảng viên đã hướng dẫn chúng em, thầy Nguyễn Mạnh Sơn, vì đã luôn sát sao quan tâm và tạo điều kiện cho chúng em có thể tích lũy thêm nhiều kiến thức không chỉ ở trong môn Lập trình hướng đối tượng mà còn là các kĩ năng và kiến thức mà sinh viên nào cũng cần nắm được để phục vụ cho công việc sau này. Ngoài ra, nhóm xin gửi lời cảm ơn tới chính bản thân các thành viên vì đã dành rất nhiều thời gian, công sức và cả tâm huyết để thực hiện và hoàn thành thành công bài tập lớn môn Lập trình hướng đối tượng. Với điều kiện thời gian cũng như kinh nghiệm còn hạn chế của người học viên, báo cáo này không thể tránh được những thiếu sót. Chúng em rất mong nhận được sự chỉ bảo, đóng góp ý

kiến của các thầy để có điều kiện bổ sung, nâng cao ý thức của mình, phục vụ tốt hơn công tác thực tế sau này.

Chúng em xin trân trọng cảm ơn!

**Phân chia công việc**

Sau khi cả nhóm họp và cũng nhau thống nhất, định hình ra mô hình bài toán, chúng em đã phân chia công việc cụ thể như sau:

|  |  |  |
| --- | --- | --- |
| **Tên** | **Công việc** | **Đóng góp** |
| **Nguyễn Tiến Thực** | **- Code phía backend, sử dụng java-servlet, tạo đối tượng, controller.** | **40%** |
| **Trần Đức Dũng** | **- Code phía frontend, tạo trang JSP, nhúng HTML, CSS, JS.** | **30%** |
| **Nguyễn Việt Pháp** | **- Thiết kế database, thêm dữ liệu, truy vấn cơ sở dữ liệu với SQL Server** | **30%** |

# Chương 1: giới thiệu bài toán

## Đặt vấn đề

Trong thời đại công nghệ 4.0, sự phát triển nhanh chóng của Internet và các thiết bị di động đã thay đổi hoàn toàn cách con người mua sắm và tiếp cận sản phẩm. Thương mại điện tử không chỉ trở thành xu hướng mà còn là nhu cầu tất yếu của doanh nghiệp và người tiêu dùng. Đặc biệt, lĩnh vực kinh doanh điện thoại di động, một sản phẩm công nghệ phổ biến và thiết yếu, đang chứng kiến sự gia tăng mạnh mẽ về nhu cầu mua sắm trực tuyến.

Ngày nay, khách hàng không chỉ tìm kiếm sự tiện lợi mà còn mong muốn trải nghiệm mua sắm phong phú, thông tin minh bạch và dịch vụ nhanh chóng. Tuy nhiên, tại Việt Nam, nhiều cửa hàng điện thoại truyền thống gặp phải giới hạn về quy mô kinh doanh và khả năng tiếp cận khách hàng rộng lớn do thiếu kênh phân phối trực tuyến. Trong khi đó, các nền tảng mua sắm trực tuyến lớn tuy đa dạng nhưng không thể đảm bảo tính chuyên sâu và cá nhân hóa dịch vụ cho từng thương hiệu hoặc cửa hàng. Điều này dẫn đến khoảng trống cần được lấp đầy bởi các website thương mại điện tử chuyên biệt, nơi các cửa hàng có thể xây dựng thương hiệu riêng và cung cấp dịch vụ tốt hơn cho khách hàng.

Một website bán điện thoại không chỉ đơn thuần là nơi trưng bày sản phẩm mà còn phải đáp ứng các yêu cầu như:

* **Giao diện thân thiện và hấp dẫn**: Khách hàng có thể dễ dàng tìm kiếm, so sánh sản phẩm, xem thông tin chi tiết như thông số kỹ thuật, giá cả và khuyến mãi.
* **Hệ thống quản lý sản phẩm hiện đại**: Cho phép chủ cửa hàng cập nhật nhanh chóng danh mục sản phẩm, tình trạng tồn kho và chương trình ưu đãi.
* **Chức năng hỗ trợ khách hàng**: Bao gồm hệ thống đánh giá, bình luận.

Ngoài ra, website còn là công cụ giúp doanh nghiệp tối ưu hóa chi phí vận hành, tăng tính cạnh tranh trên thị trường và tiếp cận khách hàng một cách hiệu quả hơn.

## Đề xuất giải pháp

Trong bối cảnh đó, việc xây dựng một **website bán điện thoại** chuyên nghiệp là một giải pháp mang tính chiến lược, không chỉ giúp cửa hàng mở rộng thị trường mà còn tạo lợi thế cạnh tranh bền vững trong ngành. Báo cáo này sẽ trình bày chi tiết quá trình nghiên cứu, thiết kế và triển khai website, tập trung vào các vấn đề chính như:

### Thiết Kế Giao Diện Người Dùng (UI/UX)

* **Giao diện thân thiện và dễ sử dụng:** Đảm bảo rằng người dùng có thể dễ dàng tìm kiếm và điều hướng trang web.

### Tích Hợp Chức Năng Cần Thiết

**Tìm kiếm và lọc sản phẩm**

* **Tìm kiếm và lọc sản phẩm:** Cho phép người dùng tìm kiếm sản phẩm theo nhiều tiêu chí như giá, thương hiệu .
* **Đánh giá và nhận xét:** Tích hợp chức năng cho phép khách hàng để lại đánh giá và nhận xét về sản phẩm.
* **Quản lý sản phẩm:** thêm bớt giỏ hàng.
* **Tài khoản:** quên mật khẩu**,** đăng nhập, đăng kí tài khoản.

# Chương 2: Kỹ thuật công nghệ sử dụng

1. **Git/ GitHub**

Sử dụng Git va GitHub để làm việc nhóm,

****

**Git** và **GitHub** là công cụ quan trọng và mạnh mẽ cho quản lý dự án, phát triển phần mềm, và cộng tác nhóm.

**Git** là một **hệ thống quản lý phiên bản phân tán (Distributed Version Control System)**.Mục đích Theo dõi lịch sử thay đổi của mã nguồn hoặc tài liệu trong dự án, giúp quản lý và hợp tác hiệu quả hơn.

**GitHub** là một **nền tảng lưu trữ mã nguồn dựa trên Git**, hoạt động trực tuyến. giúp lưu trữ các **Git** trên nền tảng đám mây, hỗ trợ làm việc nhóm cộng tác giữa những thành viên trong nhóm

## Database

sử dụng **SQL Server**

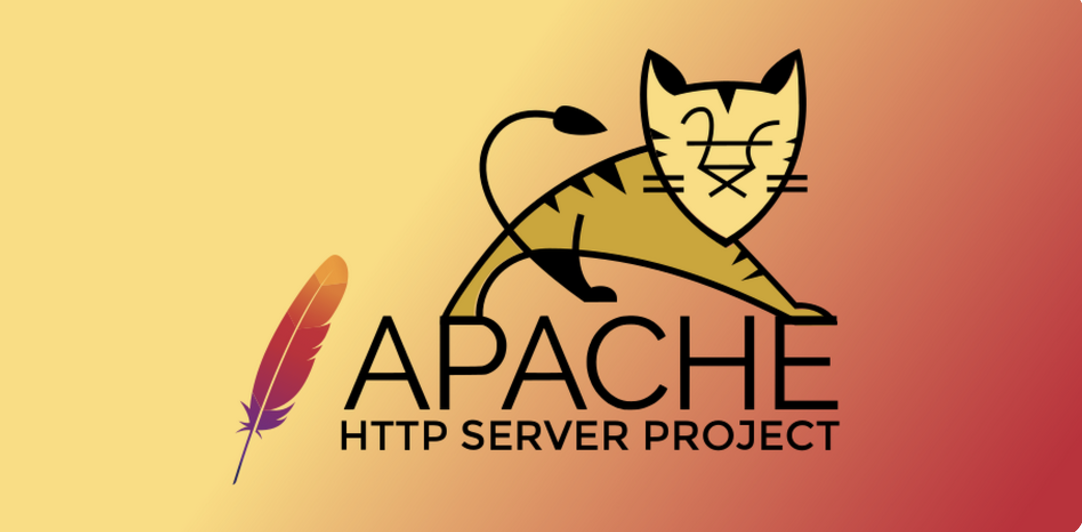
****

**SQL Server** (hay Microsoft SQL Server) là một hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) được phát triển bởi Microsoft.

SQL Server cung cấp cho người dùng các công cụ và tính năng để quản lý, lưu trữ, xử lý các truy vấn dữ liệu, kiểm soát truy cập, xử lý giao dịch và hỗ trợ tích hợp dữ liệu từ nhiều nguồn khác nhau.

## Server

sử dụng **Apache Tomcat 10**

****

**Apache Tomcat** là dạng máy chủ ứng dụng web **mã nguồn mở** được phát triển bởi Apache Software Foundation. Cụ thể hơn, Tomcat được xem là một thành phần phổ biến trong việc triển khai ứng dụng web Java.

## Backend

### SERVLET

**Servlet** là một **thành phần phần mềm phía server** trong Java được sử dụng để xử lý các yêu cầu từ phía khách hàng (client), thường là các trình duyệt web, và tạo ra các phản hồi động (dynamic response). Servlet chạy trên một **container servlet** (như Apache Tomcat, Jetty) và được sử dụng để xây dựng các ứng dụng web.

**Kiến trúc Servlet**



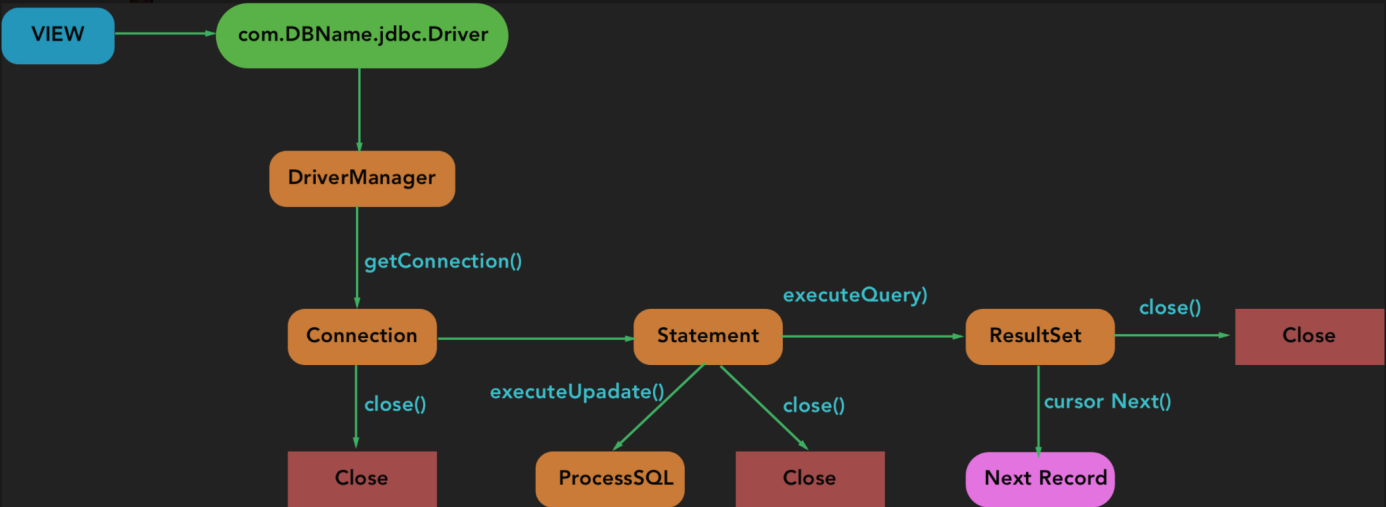
**Các phương thức chính trong Servlet**

1. **init(ServletConfig config)**
   * Được gọi một lần khi Servlet được khởi tạo.
2. **service(HttpServletRequest request, HttpServletResponse response)**
   * Xử lý các yêu cầu từ client và tạo phản hồi.
3. **doGet(HttpServletRequest request, HttpServletResponse response)**
   * Xử lý yêu cầu HTTP GET.
4. **doPost(HttpServletRequest request, HttpServletResponse response)**
   * Xử lý yêu cầu HTTP POST.
5. **destroy()**
   * Được gọi khi Servlet bị dừng hoặc container đóng lại.

### JDBC (Java Database Connectivity)

**JDBC** - Java Database Connectivity, là một API chuẩn để kết nối giữa ngôn ngữ lập trình Java và các database. Nó là một phần của JavaSE (Java Standard Edition). API JDBC sử dụng JDBC drivers để kết nối với cơ sở dữ liệu.

**luồng thực hiện** trong **JDBC**



Các bước kết nối với **JDBC**:

* Load driver
* Tạo kết nối (Create Connection)
* Tạo câu lệnh truy vấn SQL (Statement)
* Thực thi câu lệnh truy vấn SQL (Excute query)
* Đóng kết nối (Close Connection)

## Frontend

### JSP (Java Server Pages)

****

**JSP**, hay **Java Server Pages**, là một công nghệ tiêu chuẩn cho phép các lập trình viên tạo web page động bằng cách sử dụng kết hợp mã HTML, XML và Java. Các trang **JSP** được thực thi trên web server và kết quả đầu ra sẽ được gửi tới trình duyệt web của khách truy cập.

các thành phần chính của JSP:

* **Thẻ** **root**: Chứa các thuộc tính và tất cả thông tin của một trang JSP.
* **Comment**: Bạn hoàn toàn có thể sử dụng tính năng comment trong JSP tương tự như trong trang HTML.
* **Declaration**: Thành phần này dùng để khai báo các biến hoặc phương thức của Java trong trang JSP. Tuy nhiên, bạn cần lưu ý, nếu khai báo quá nhiều, tình trạng nhầm lẫn code JSP với code Java có thể xảy ra.
* **Expression**: Được dùng để chèn trực tiếp một giá trị nào đó vào trang.
* **Thẻ biểu thức JSP**: Được dùng để đánh giá biểu thức, định hướng các output đến trình duyệt website thích hợp.

### HTML/ CSS/ JS

****

**HTML**, **CSS**, và **JavaScript** là **ba công nghệ cơ bản** của lập trình web, giúp xây dựng và thiết kế các trang web tương tác và hấp dẫn. Mỗi công nghệ đóng một vai trò khác nhau trong quá trình phát triển giao diện người dùng.

**HTML** là một ngôn ngữ đánh dấu được sử dụng để tạo cấu trúc và nội dung cho một trang web. Nó giúp định nghĩa các phần tử như văn bản, hình ảnh, liên kết, danh sách, và các thành phần khác trên trang web.

**CSS** là ngôn ngữ định kiểu, dùng để thiết kế và định dạng giao diện của trang web. Nó kiểm soát cách các phần tử HTML hiển thị trên trình duyệt như màu sắc, phông chữ, bố cục, khoảng cách, v.v.

**JavaScript** là ngôn ngữ lập trình động giúp trang web trở nên tương tác và linh hoạt hơn. Nó chạy trực tiếp trên trình duyệt của người dùng.

### Boostraps

****

**Bootstrap** là một Framework Front-End phổ biến được sử dụng để thiết kế giao diện Website. Bootstrap cung cấp một bộ thư viện các thành phần và công cụ giúp thiết kế Website nhanh hơn và dễ dàng hơn. Đây là mã nguồn mở miễn phí, có thể được tải về và sử dụng dễ dàng, cung cấp mẫu giao diện Website với các thành phần có sẵn như menu điều hướng, form, button, carousel, modal...

1. **Font Awesome**

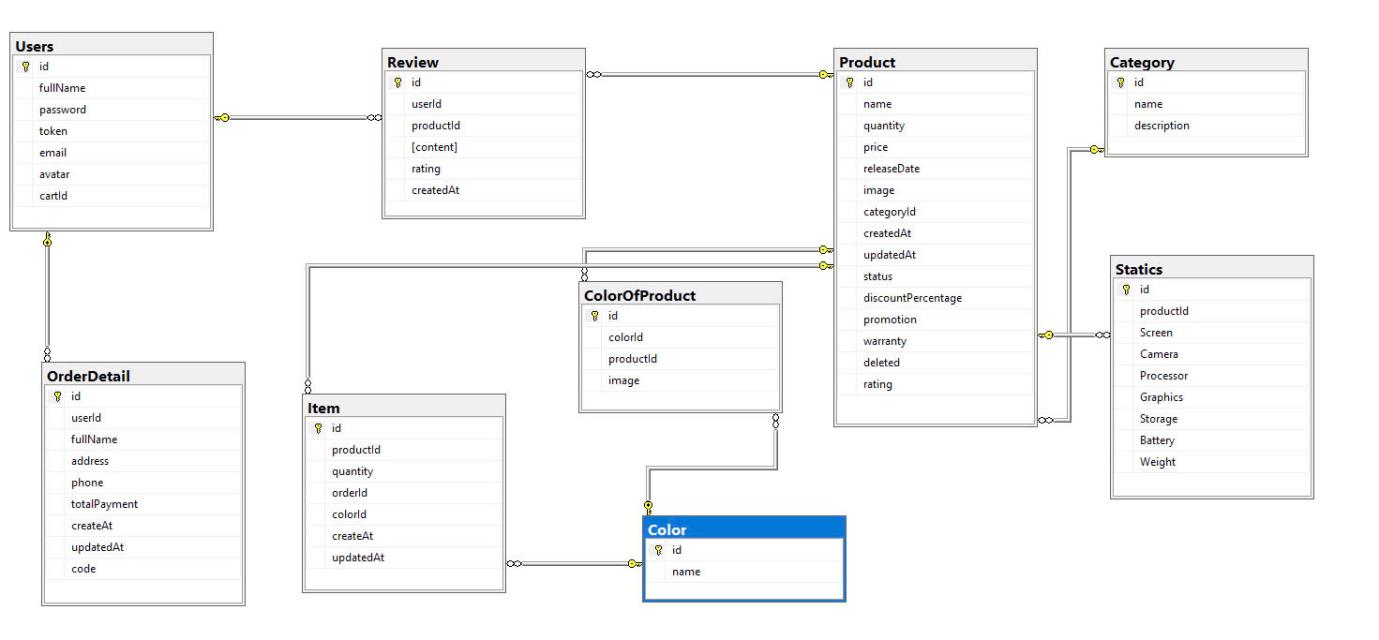
****

**Font Awesome** là một thư viện biểu tượng **icon** phổ biến được sử dụng trong thiết kế giao diện web và ứng dụng. Nó cung cấp hàng ngàn biểu tượng chất lượng cao, có thể dễ dàng tùy chỉnh về kích thước, màu sắc, hoặc hiệu ứng, giúp giao diện trở nên sinh động và chuyên nghiệp hơn.

# Chương 3: Kiến trúc hệ thống

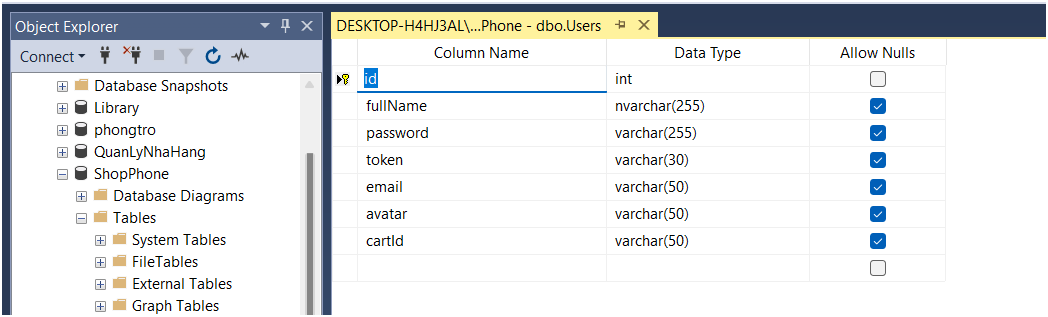
## Database (SQL Server)

1. Thiết kế Database

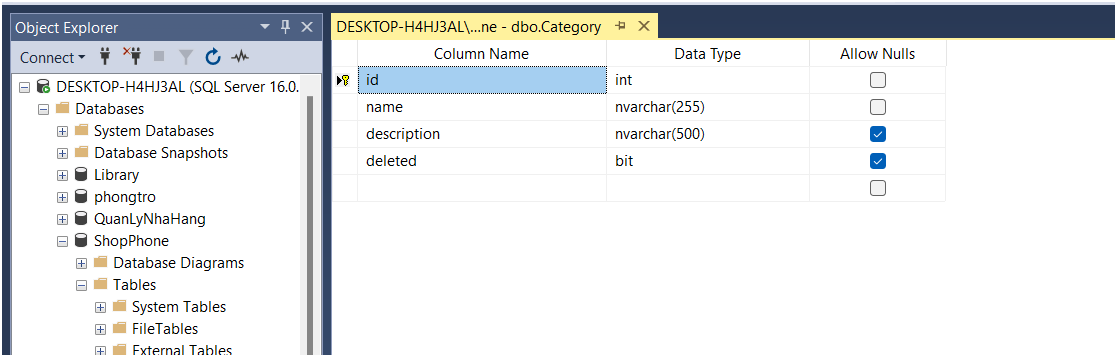


1. Tạo các bảng, các ràng buộc, chỉ mục:

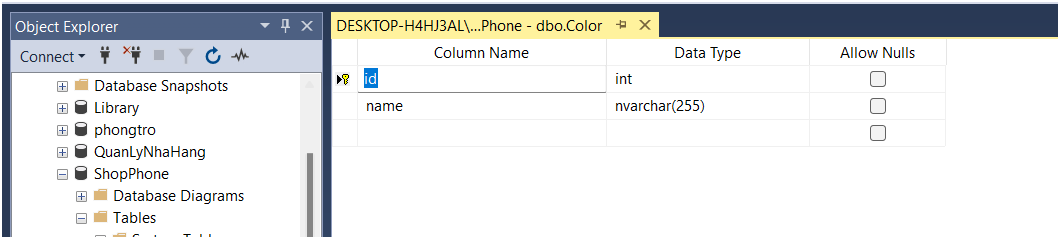
- Bảng **Users** đại diện cho một lớp “tài khoản người dùng “



- Bảng **Category**, đại diện cho thể loại các điện thoại



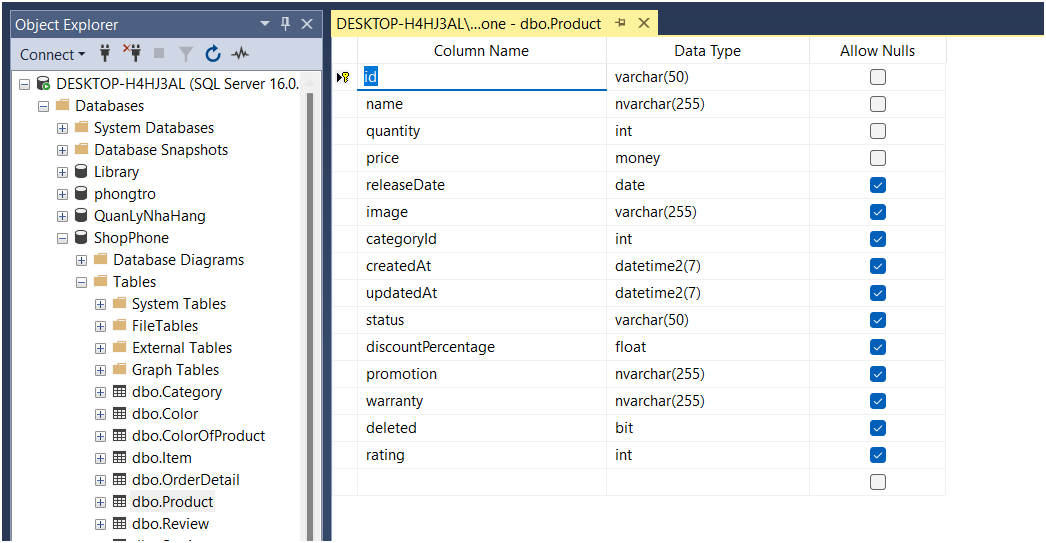
- Bảng **Color**, đại diện cho các loại màu điện thoại



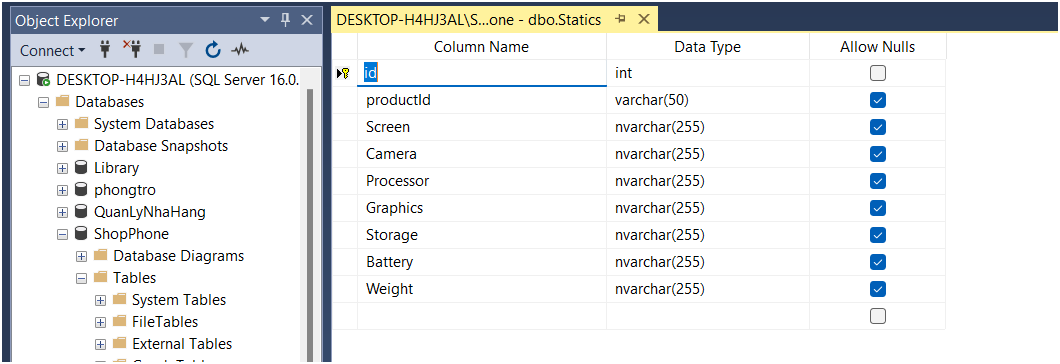
- Bảng **ColorOfProduct**, đại diện chi tiết từng màu của mỗi điện thoại



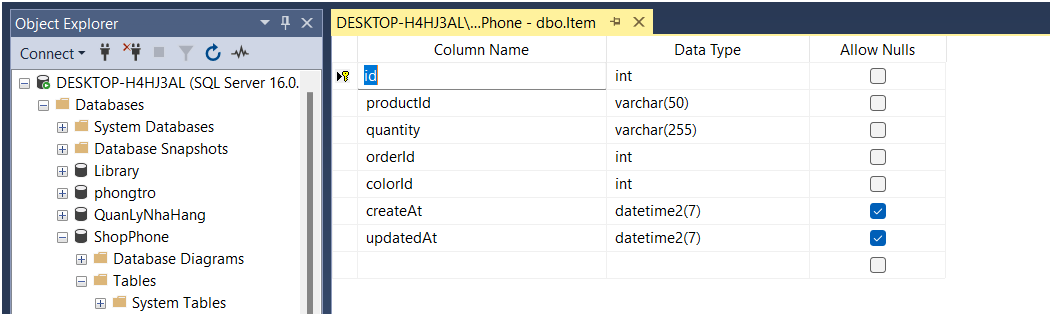
- Bảng **Product**, đại diện cho điện thoại mà admin đăng lên



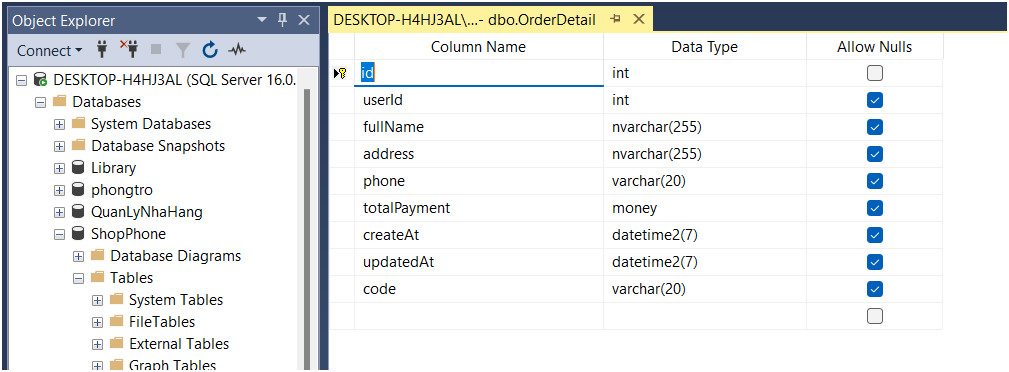
- Bảng **Statics**, đại diện cho thông số kỹ thuật điện thoại



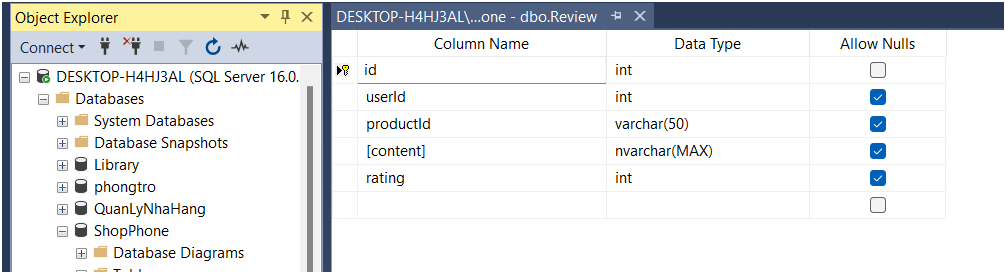
- Bảng **Item**, đại diện cho đơn hàng mà khách hàng đã đặt



- Bảng **OrderDetail**, đại diện cho chi tiết mỗi đơn hàng

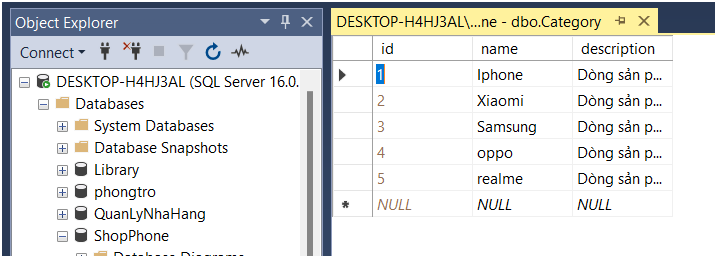


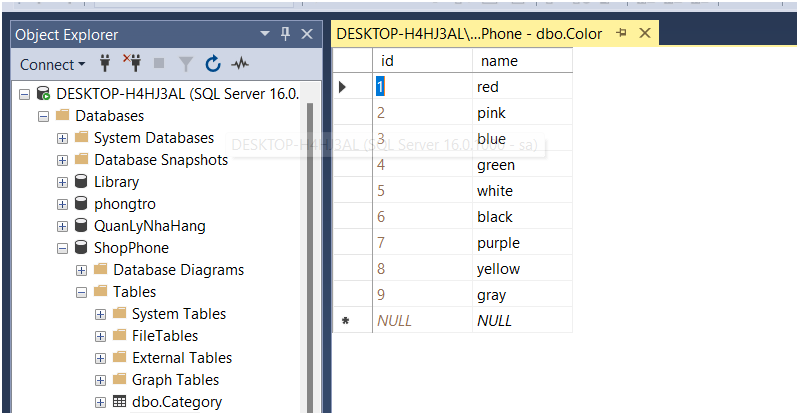
- Bảng **Review**, đại diện cho đánh giá của khách hàng

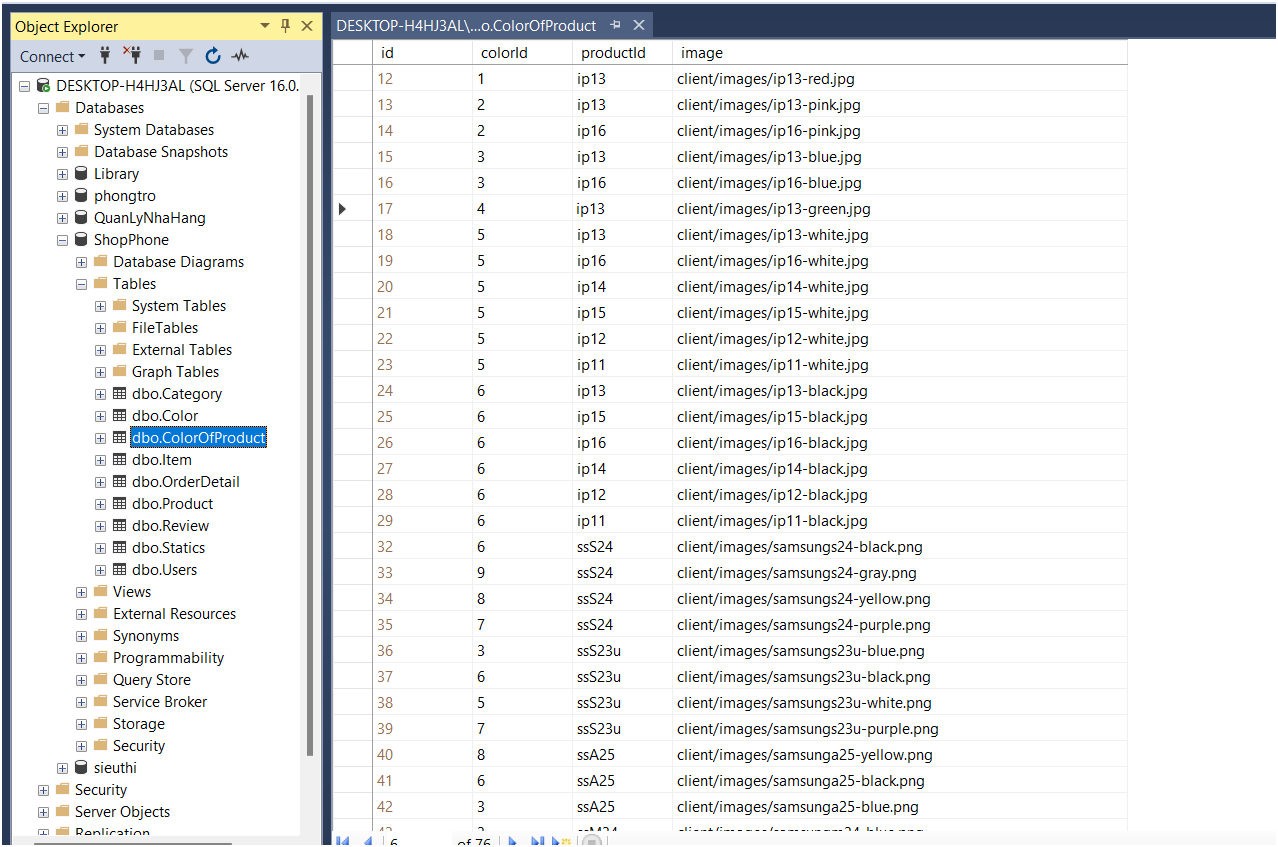


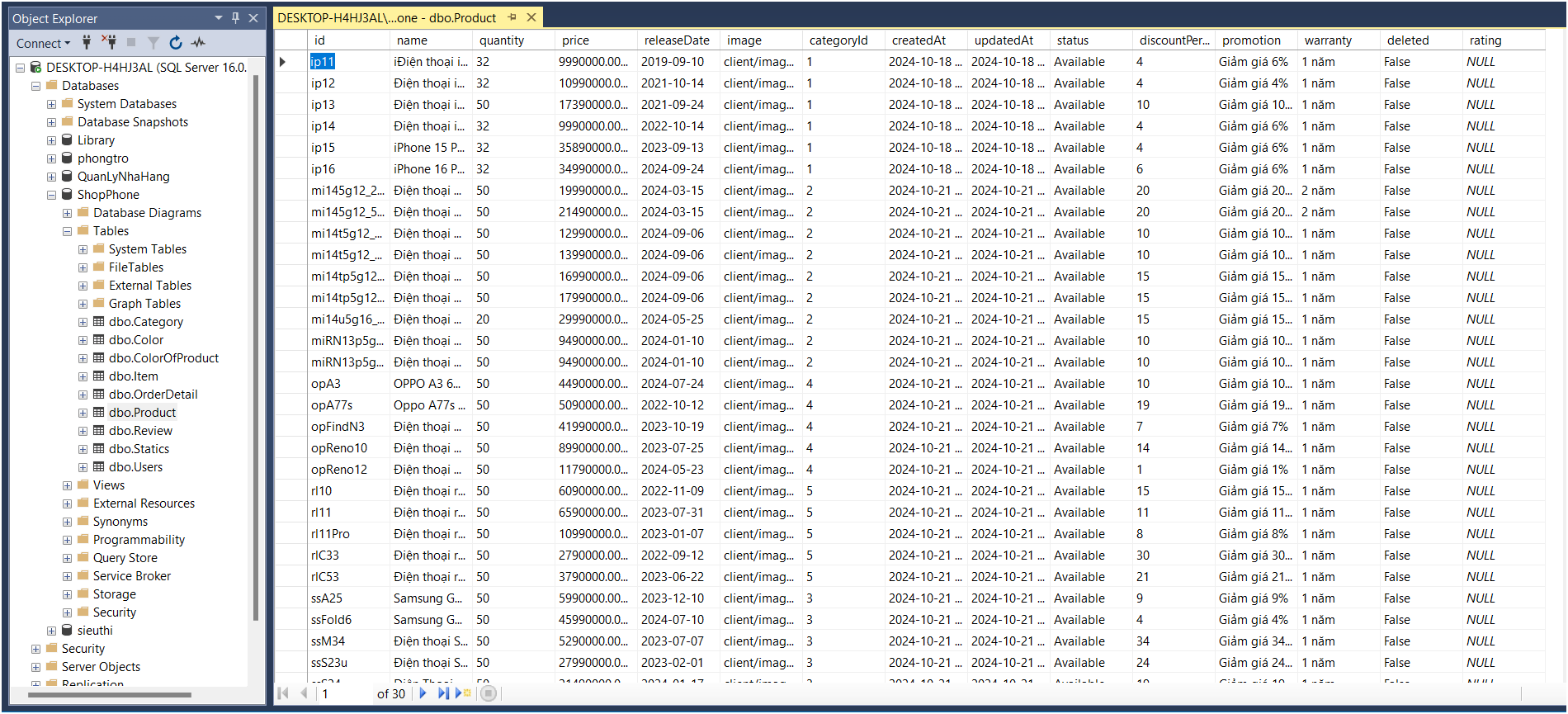
1. Cung cấp dữ liệu cho bảng

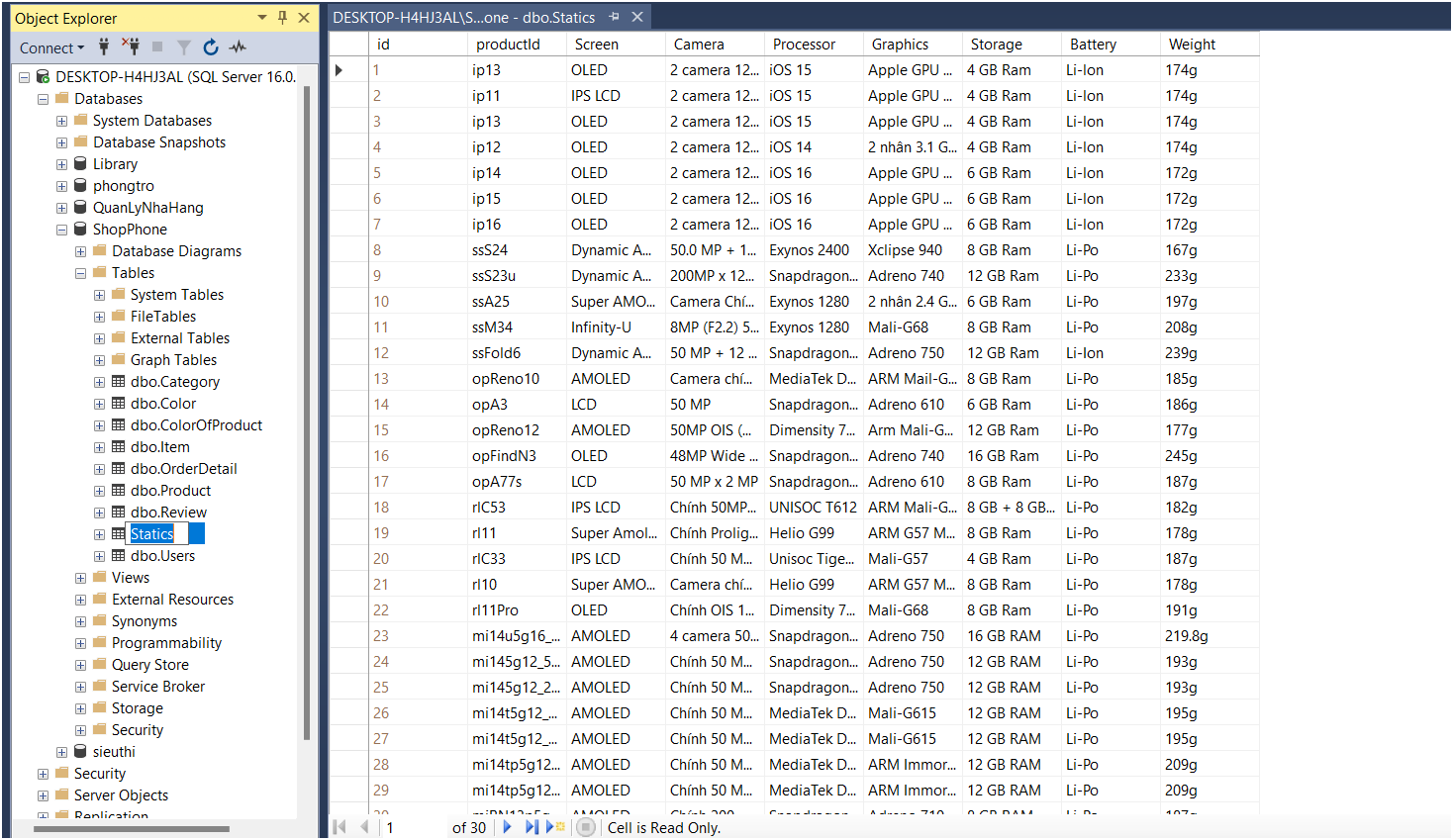
Với các bảng Category, Color, ColorOfProduct, Product, Statics ta chèn được các dữ liệu tương ứng như sau: (các bảng còn lại như Users, Item, OrderDetail, Review sẽ tự động được thêm vào database khi nhập dữ liệu trên web)



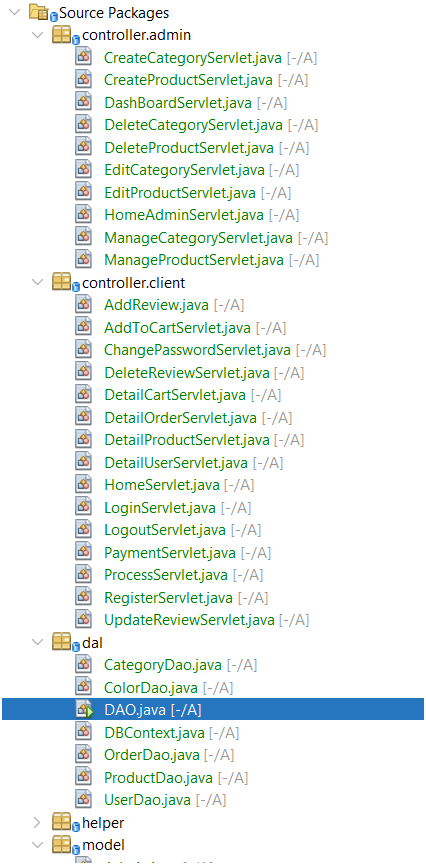








1. **Hệ thống phía Backend**

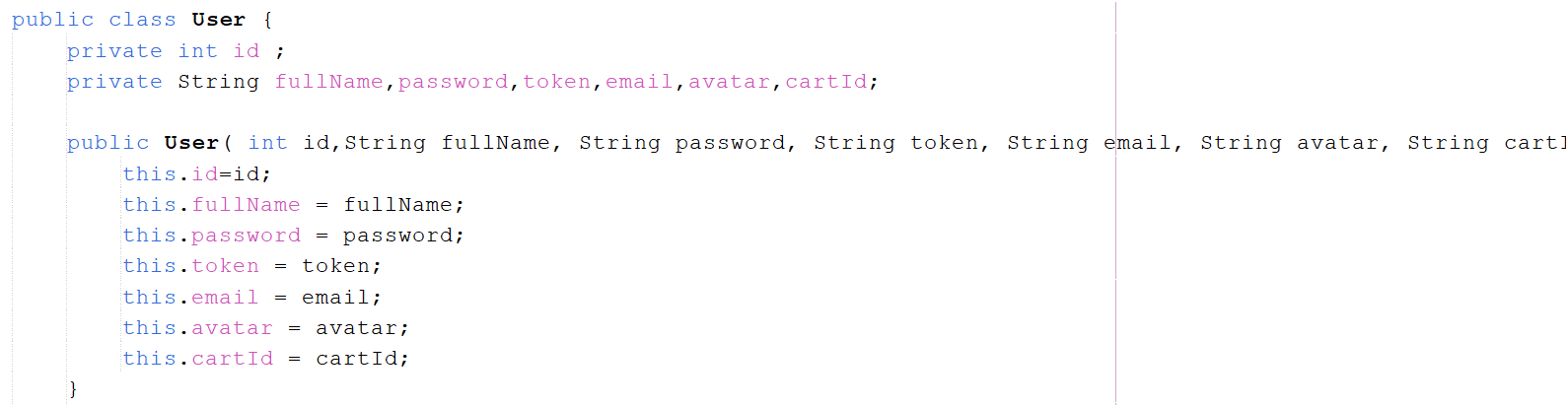




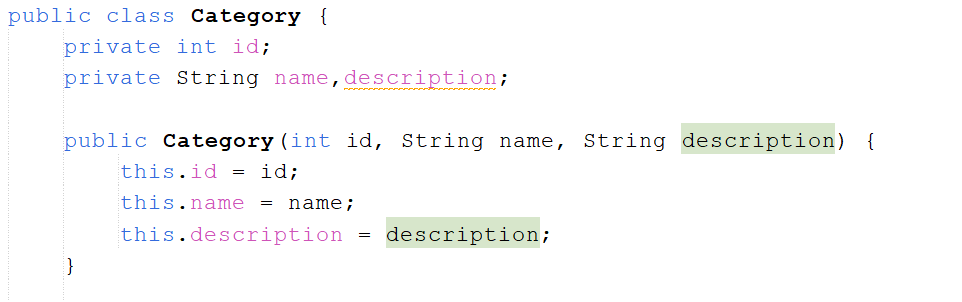
1. Model

Package chứa toàn bộ thực thể của website:

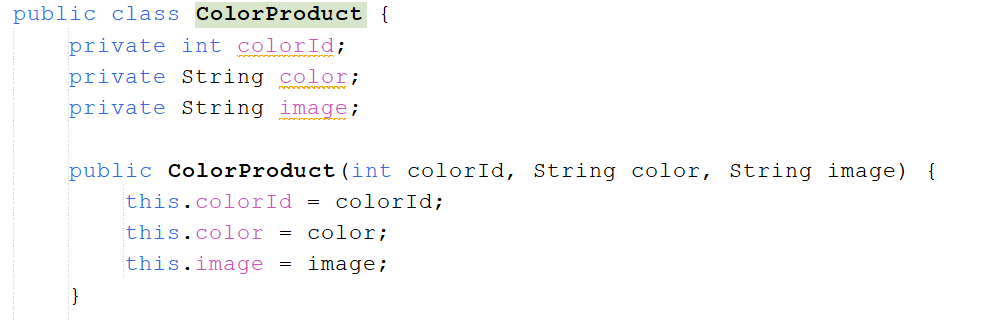
- User:



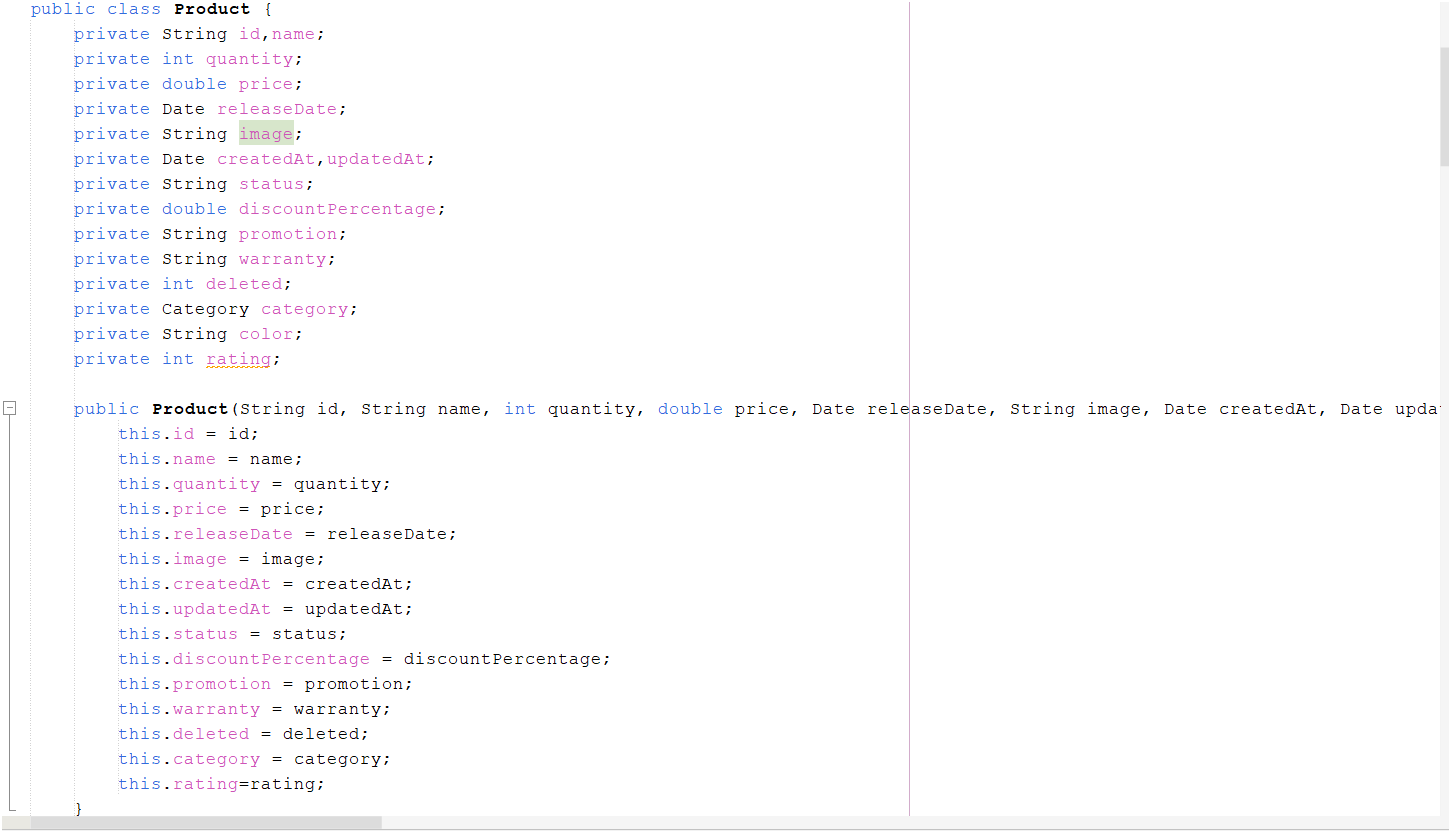
- Category:



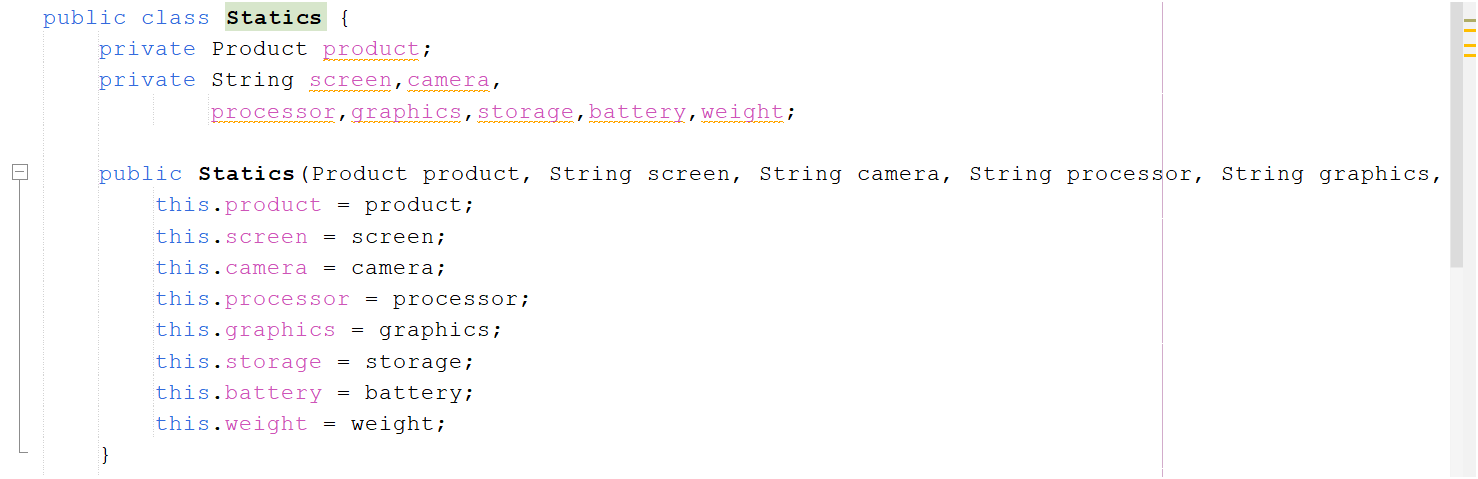
- ColorProduct:



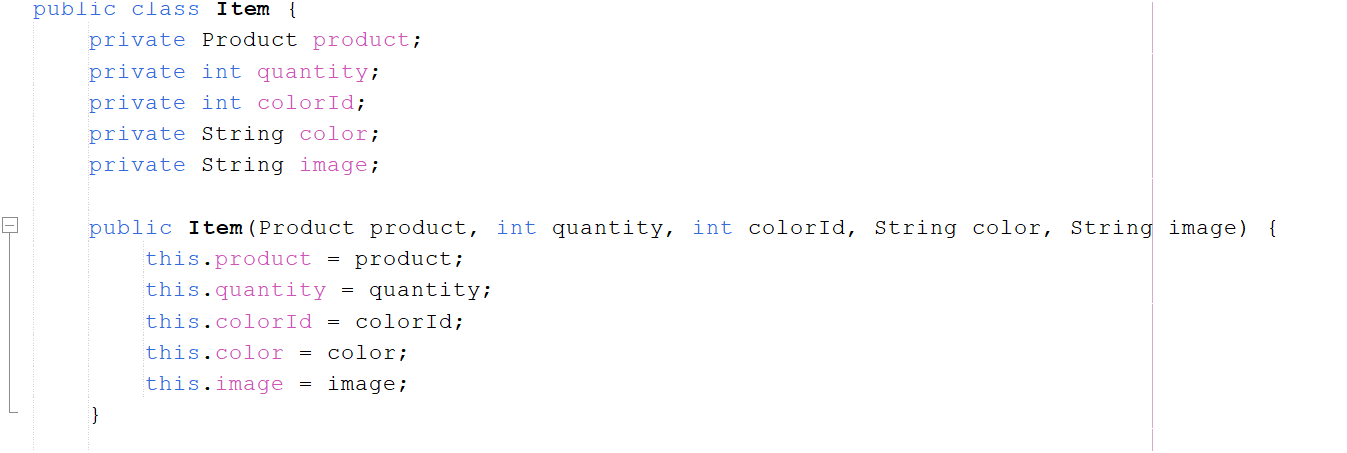
- Product:



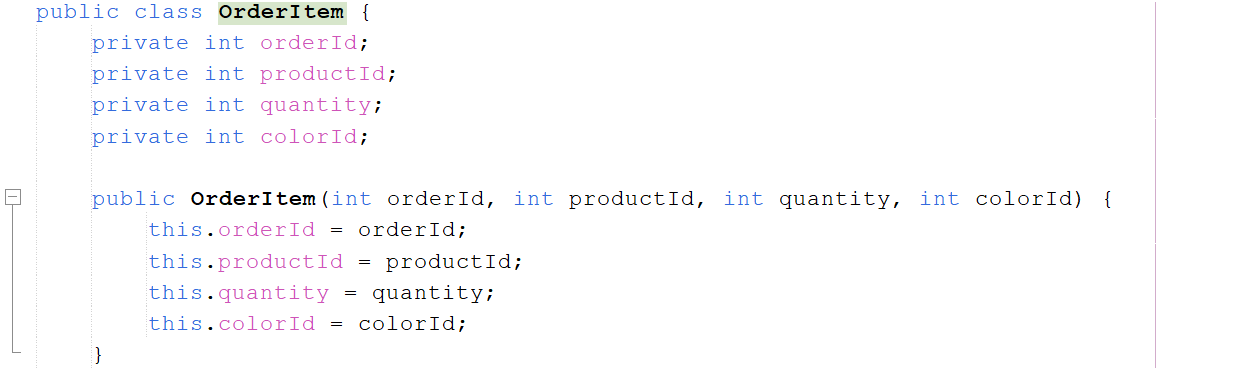
- Statics:



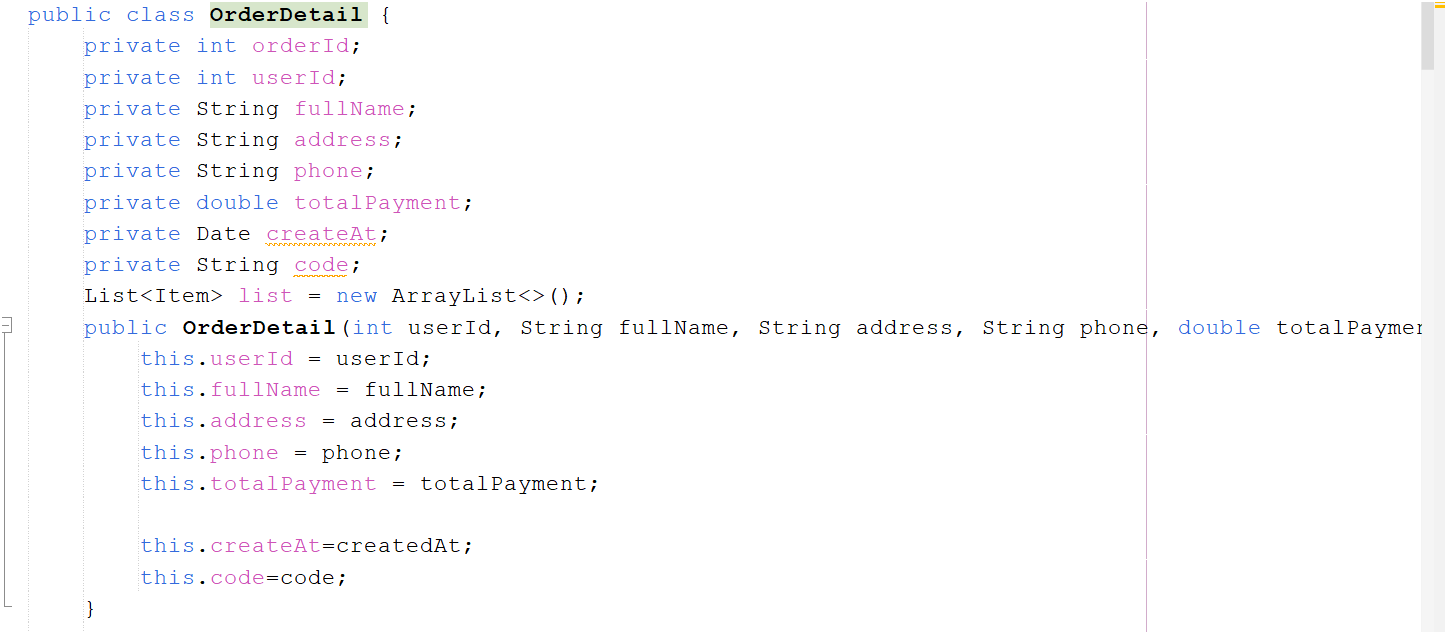
- Item:



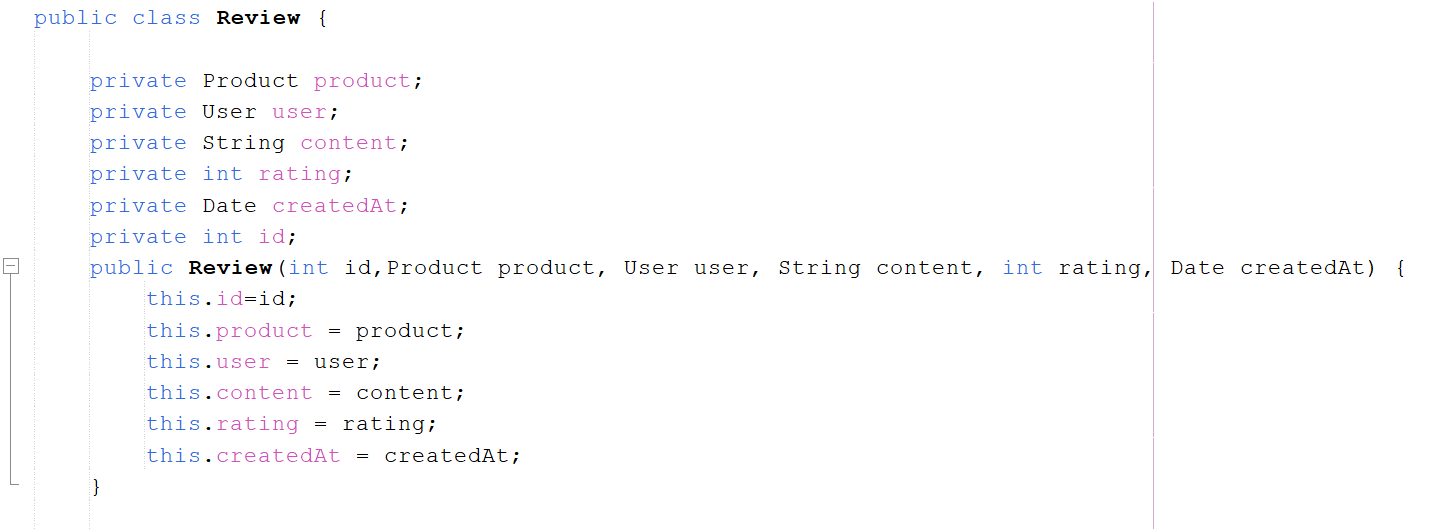
- OrderItem:



- OrderDetail:



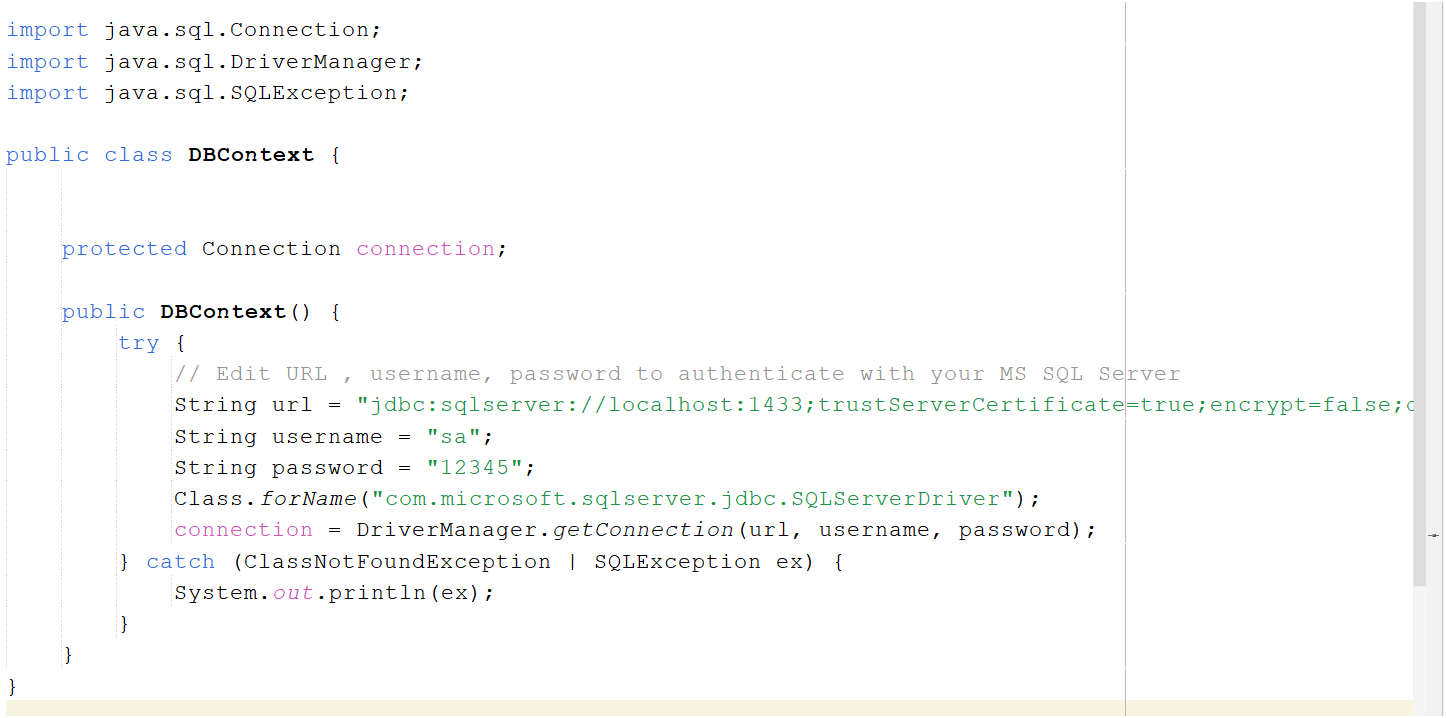
- Review:



1. Dal

****

Đây là package để truy vấn với cơ sở dữ liệu, file **DBContext.java** kết nối Back-end với CSDL



- **DAO**:



Các phương thức chính trong lớp này như:

- getAll(): Lấy danh sách tất cả các danh mục (Category) chưa bị xóa từ cơ sở dữ liệu

- getAllProducts(): Lấy danh sách tất cả các sản phẩm (Product) chưa bị xóa từ cơ sở dữ liệu

- getCategoryByID(int id): Trả về thông tin danh mục theo ID

- getListByPage(ArrayList<Product> list, int begin, int end): Phân trang danh sách sản phẩm

- getProductbyCondition(int id, String radioPrice, String keyword): Lọc sản phẩm theo danh mục, khoảng giá và từ khóa

- sortProduct(List<Product> list, String sortKey, String sortValue): Sắp xếp danh sách sản phẩm theo tiêu đề hoặc giá

- getProduct(String id): Lấy thông tin chi tiết của một sản phẩm theo ID

- getColorsProduct(String id): Lấy danh sách màu sắc của một sản phẩm

- getColorName(int id): Lấy tên màu theo ID

- getColorImage(int id): Lấy hình ảnh màu theo ID

- getUserById(int id): Lấy thông tin người dùng theo ID

- getUserByEmail(String email): Lấy thông tin người dùng theo email

- getUser(String email, String password): Xác thực người dùng bằng email và mật khẩu

- getOrderId(String keyword): Lấy ID đơn hàng mới nhất hoặc đếm số lượng đơn hàng

- inserUsertoDB(): Thêm người dùng mới vào cơ sở dữ liệu

- getUserByToken(String token): Lấy thông tin người dùng bằng token

- insertOrdertoDB(OrderDetail o): Thêm chi tiết đơn hàng vào cơ sở dữ liệu

- insertOneItemtoDB(Item item, int orderId): Thêm một mục hàng vào đơn hàng

- insertItemstoDB(List<Item> items): Thêm nhiều mục hàng vào đơn hàng

- getOrderByUserId(int userId): Lấy danh sách đơn hàng của một người dùng

- updateUser(): Cập nhật thông tin người dùng

- updatePassword(): Cập nhật mật khẩu người dùng

- getStatic(String productId): Lấy thông số kỹ thuật của sản phẩm

- insertReview(Review r): Thêm đánh giá sản phẩm

- getAllReview(): Lấy tất cả đánh giá

- getAllReviewByProductId(String productId): Lấy đánh giá theo ID sản phẩm

- getProductId(int reviewId): Lấy ID sản phẩm từ ID đánh giá

- deleteReview(int reviewId): Xóa đánh giá

- updateRatingOfProduct(String productId, int rating): Cập nhật điểm đánh giá sản phẩm

- updateReview(): Cập nhật thông tin đánh giá

- getAllOrder(): Lấy tất cả đơn hàng

- getProductByCategoryId(int categoryId): Lấy sản phẩm theo ID danh mục

- deleteProduct(String productId): Xóa sản phẩm (đánh dấu là đã xóa)

- deleteCategory(int categoryId): Xóa danh mục (đánh dấu là đã xóa và xóa các sản phẩm thuộc danh mục)

- insertCategory(): Thêm danh mục mới

- updateCategory(): Cập nhật thông tin danh mục

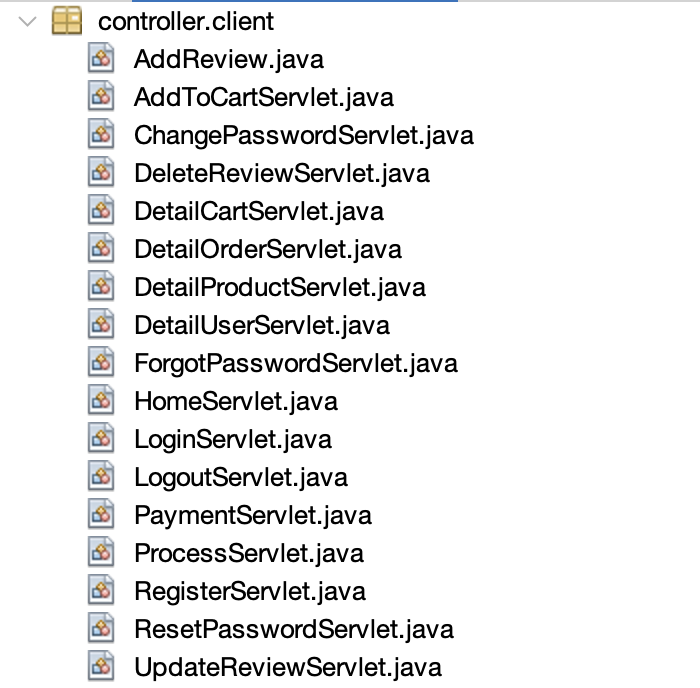
- getAllProduct(): Lấy tất cả sản phẩm (đã được sắp xếp theo ngày tạo)

- insertProduct(Product p): Thêm sản phẩm mới

- updateProduct(Product p, String productId): Cập nhật thông tin sản phẩm

Cả lớp DAO này được thiết kế để tương tác với cơ sở dữ liệu thông qua JDBC (Java Database Connectivity) để thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên các bảng cơ sở dữ liệu.

1. Controller.client



**AddReview t**hực hiện chức năng thêm đánh giá sản phẩm. Kiểm tra người dùng đã đăng nhập hay chưa thông qua cookie token. Nếu chưa, thông báo yêu cầu đăng nhập và chuyển hướng về trang chi tiết sản phẩm. Nếu đã đăng nhập, nhận thông tin đánh giá từ yêu cầu HTTP, thêm đánh giá vào cơ sở dữ liệu qua DAO, cập nhật điểm đánh giá trung bình của sản phẩm, và chuyển hướng người dùng về trang chi tiết sản phẩm.

**AddToCartServlet** có nhiệm vụ Xử lý yêu cầu thêm sản phẩm vào giỏ hàng. Servlet lấy productId từ đường dẫn và colorId từ yêu cầu HTTP, sử dụng DAO để lấy thông tin sản phẩm từ cơ sở dữ liệu. Tiếp theo, kiểm tra cookie cartId để xác định giỏ hàng hiện tại, thêm sản phẩm mới vào danh sách sản phẩm trong giỏ hàng, cập nhật lại cookie giỏ hàng, thông báo đặt hàng thành công và chuyển hướng người dùng về trang chi tiết sản phẩm.

**ChangePasswordServlet** thực hiện chức năng thay đổi mật khẩu của người dùng. Phương thức doGet kiểm tra cookie token để xác định người dùng hiện tại, lấy thông tin từ DAO, và chuyển hướng đến trang change-password.jsp với thông tin người dùng. Nếu có lỗi trong session, lỗi được gửi đến trang JSP và xóa khỏi session. Phương thức doPost xử lý yêu cầu thay đổi mật khẩu bằng cách kiểm tra cookie token, lấy các thông tin về mật khẩu hiện tại và mật khẩu mới từ yêu cầu. Hệ thống kiểm tra các điều kiện: mật khẩu hiện tại phải khớp, mật khẩu mới phải trùng nhau, và có độ dài tối thiểu 8 ký tự. Nếu vi phạm, lỗi sẽ được gửi về trang JSP kèm theo thông tin người dùng. Nếu hợp lệ, mật khẩu mới được cập nhật qua DAO, một thông báo thành công được đặt trong session và người dùng được chuyển hướng lại trang đổi mật khẩu.

**DeleteReviewServlet** thực hiện chức năng xóa đánh giá (review) của sản phẩm. Phương thức doGet xử lý yêu cầu xóa bằng cách lấy thông tin từ đường dẫn (pathInfo), kiểm tra ID review có hợp lệ không (dạng /id). Nếu hợp lệ, ID review được lấy từ đường dẫn và DAO được sử dụng để xóa đánh giá đó khỏi cơ sở dữ liệu. Sau khi xóa thành công, servlet chuyển hướng người dùng về trang chi tiết sản phẩm (/Shop/detail/{productId}) bằng cách lấy productId tương ứng với review đã xóa từ DAO. Nếu ID không hợp lệ, servlet trả về lỗi HTTP 404.

**DetailCartServlet** xử lý việc hiển thị chi tiết giỏ hàng. Phương thức doGet lấy dữ liệu từ cookie để xác định giỏ hàng của người dùng, sử dụng cartId để xây dựng chuỗi dữ liệu giỏ hàng (txt). Nếu người dùng đã đăng nhập (xác định qua cookie token), thông tin tài khoản được lấy từ DAO và gửi tới trang JSP. Một đối tượng Cart được khởi tạo từ dữ liệu giỏ hàng, phân tích các mục (Item) trong giỏ và tính tổng tiền. Các thông tin này (danh sách sản phẩm, tổng tiền) được truyền tới trang /client/cart.jsp để hiển thị. Nếu không có giỏ hàng hợp lệ, giỏ sẽ rỗng.

**DetailOrderServlet** xử lý việc hiển thị chi tiết đơn hàng cho người dùng. Phương thức doGet nhận thông tin từ cookie để xác định người dùng (token). Nếu người dùng chưa đăng nhập (không tìm thấy token), trang JSP sẽ hiển thị thông báo lỗi yêu cầu đăng nhập. Nếu người dùng đã đăng nhập, thông tin tài khoản và danh sách các đơn hàng được lấy từ cơ sở dữ liệu (DAO) và gửi tới trang /client/order.jsp để hiển thị.

**DetailProductServlet** xử lý yêu cầu hiển thị chi tiết sản phẩm. Khi nhận được yêu cầu GET, servlet sẽ kiểm tra xem có lỗi nào trong session không và hiển thị nếu có. Sau đó, nó lấy productId từ URL, sử dụng DAO để lấy thông tin chi tiết sản phẩm, các lựa chọn màu sắc, đánh giá và thống kê liên quan đến sản phẩm. Servlet cũng kiểm tra xem người dùng có đăng nhập không thông qua token trong cookie, nếu có sẽ lấy thông tin người dùng. Các dữ liệu này sau đó được lưu vào thuộc tính request và chuyển tiếp đến trang detail.jsp để hiển thị. Nếu productId không hợp lệ, servlet sẽ trả về lỗi 404.

**DetailUserServlet** xử lý việc hiển thị và cập nhật thông tin người dùng. Khi người dùng gửi yêu cầu GET, servlet sẽ lấy thông tin người dùng từ cookie, kiểm tra và hiển thị lỗi nếu có, rồi chuyển tiếp thông tin đến trang detail-user.jsp. Khi nhận yêu cầu POST, servlet nhận các thông tin như tên đầy đủ, email và ảnh đại diện từ form, tải ảnh lên thư mục lưu trữ, và cập nhật thông tin người dùng vào cơ sở dữ liệu. Sau khi cập nhật, servlet chuyển hướng người dùng về trang chi tiết thông tin cá nhân và thông báo "Cập nhật thành công!".

**HomeServlet** quản lý việc hiển thị sản phẩm trên trang chủ, cung cấp các tính năng như lọc sản phẩm theo danh mục, từ khóa, khoảng giá, sắp xếp sản phẩm, và phân trang. Khi người dùng truy cập trang chủ, servlet sẽ truy xuất danh sách các danh mục sản phẩm và sản phẩm từ cơ sở dữ liệu, áp dụng các điều kiện lọc và sắp xếp được người dùng chọn. Ngoài ra, servlet còn quản lý giỏ hàng bằng cách tạo và theo dõi cartId thông qua cookie, đồng thời xác thực người dùng bằng token. Sau khi xử lý dữ liệu, servlet chuyển tiếp các thuộc tính như danh sách sản phẩm, danh mục, thông tin phân trang và người dùng đến trang home.jsp để hiển thị giao diện cho người dùng.

**LoginServlet** xử lý việc đăng nhập của người dùng. Khi người dùng gửi form đăng nhập, servlet kiểm tra xem email có tồn tại trong cơ sở dữ liệu không, nếu không sẽ thông báo lỗi. Tiếp theo, nếu mật khẩu sai, servlet sẽ thông báo mật khẩu không đúng. Nếu đăng nhập thành công, servlet tạo một cookie token để xác thực người dùng trong phiên làm việc, đồng thời quản lý giỏ hàng của người dùng. Servlet sẽ lấy thông tin giỏ hàng từ các cookie và kết hợp giỏ hàng cũ của người dùng với giỏ hàng mới (nếu có). Cuối cùng, servlet lưu lại giỏ hàng vào các cookie mới và chuyển hướng người dùng đến trang chủ.

**LogoutServlet** xử lý việc đăng xuất của người dùng. Khi người dùng thực hiện đăng xuất (GET request), servlet xóa tất cả các cookie liên quan đến phiên làm việc của người dùng, bao gồm cả cookie token (dùng để xác thực người dùng) và cookie cartId (giỏ hàng của người dùng). Sau khi xóa các cookie, servlet chuyển hướng người dùng đến trang đăng nhập (login).

**PaymentServlet** quản lý quá trình thanh toán của người dùng, xử lý cả yêu cầu GET và POST. Khi nhận yêu cầu GET, servlet kiểm tra lỗi từ session và hiển thị thông báo lỗi nếu có, đồng thời lấy thông tin giỏ hàng từ cookie và kiểm tra trạng thái đăng nhập bằng token. Nếu người dùng đã đăng nhập, thông tin giỏ hàng và tổng tiền được gửi đến trang thanh toán. Khi nhận yêu cầu POST, servlet lấy thông tin thanh toán từ người dùng, kiểm tra xem họ đã đăng nhập chưa (dựa trên token). Nếu chưa, yêu cầu người dùng đăng nhập, nếu đã đăng nhập thì thực hiện thanh toán, lưu thông tin đơn hàng vào cơ sở dữ liệu, xóa giỏ hàng và chuyển hướng về trang chủ sau khi thanh toán thành công.

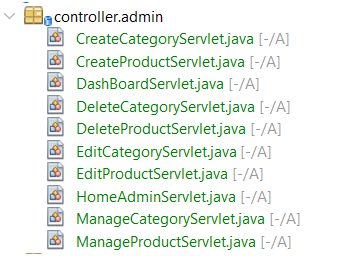
**ProcessServlet** xử lý các yêu cầu liên quan đến việc thay đổi giỏ hàng của người dùng. Khi nhận yêu cầu GET, servlet thực hiện các hành động như giảm hoặc tăng số lượng sản phẩm trong giỏ hàng, hoặc xóa sản phẩm khỏi giỏ hàng. Nó lấy thông tin giỏ hàng từ cookie, xác định hành động cần thực hiện (giảm, tăng, hoặc xóa), và cập nhật lại giỏ hàng. Sau khi cập nhật giỏ hàng, servlet lưu lại thông tin giỏ hàng mới vào cookie và chuyển hướng người dùng đến trang giỏ hàng hoặc trang chủ nếu giỏ hàng đã trống.

**RegisterServlet** xử lý các yêu cầu đăng ký tài khoản người dùng. Khi nhận yêu cầu GET, servlet sẽ chuyển hướng người dùng đến trang đăng ký (JSP). Khi nhận yêu cầu POST, servlet sẽ kiểm tra các thông tin đăng ký của người dùng, bao gồm tên, email, mật khẩu và xác nhận mật khẩu. Nếu email đã tồn tại trong cơ sở dữ liệu, servlet sẽ thông báo lỗi và chuyển hướng lại đến trang đăng ký. Nếu mật khẩu không đủ 8 ký tự hoặc không khớp với mật khẩu xác nhận, servlet cũng sẽ thông báo lỗi tương ứng. Nếu các thông tin hợp lệ, servlet sẽ tạo một người dùng mới, lưu vào cơ sở dữ liệu với thông tin đăng ký và tạo token, mã giỏ hàng (cartId). Sau đó, servlet chuyển hướng người dùng đến trang đăng nhập.

**ResetPasswordServlet** xử lý việc đặt lại mật khẩu cho người dùng trong hệ thống. Khi người dùng gửi yêu cầu POST với các thông tin như email, mã OTP, và mật khẩu mới, servlet sẽ xác thực OTP và email trong cơ sở dữ liệu. Nếu hợp lệ, servlet kiểm tra độ dài và tính khớp của mật khẩu mới. Nếu các điều kiện được đáp ứng, mật khẩu mới sẽ được cập nhật vào cơ sở dữ liệu. Sau khi hoàn tất, servlet chuyển hướng người dùng đến trang đăng nhập. Trong trường hợp có lỗi (như OTP không hợp lệ hoặc mật khẩu không đạt yêu cầu), servlet hiển thị lại giao diện đặt lại mật khẩu cùng thông báo lỗi tương ứng.

**UpdateReviewServlet** xử lý việc cập nhật đánh giá của người dùng đối với một sản phẩm. Khi người dùng gửi yêu cầu POST với các thông tin như mô tả, xếp hạng, và ID sản phẩm, servlet sẽ lấy ID đánh giá từ URL và cập nhật thông tin đánh giá vào cơ sở dữ liệu. Sau đó, servlet tính toán lại điểm đánh giá trung bình của sản phẩm dựa trên các đánh giá hiện có, rồi cập nhật lại điểm trung bình đó trong cơ sở dữ liệu. Sau khi hoàn tất, servlet chuyển hướng người dùng đến trang chi tiết sản phẩm.

1. Controller.admin



**CreateCategoryServlet** chịu trách nhiệm xử lý yêu cầu tạo danh mục mới trong hệ thống quản trị. Khi nhận yêu cầu GET tại đường dẫn /admin/create-category, servlet hiển thị form tạo danh mục để người dùng nhập thông tin. Với yêu cầu POST, servlet nhận dữ liệu từ form (tên và mô tả danh mục), sử dụng DAO để thêm danh mục mới vào cơ sở dữ liệu, và sau đó chuyển hướng người dùng đến trang danh sách danh mục. Nếu xảy ra lỗi trong quá trình xử lý, servlet sẽ in ra thông báo lỗi.

**CreateProductServlet** chịu trách nhiệm xử lý yêu cầu tạo sản phẩm mới trong hệ thống quản trị. Khi nhận yêu cầu GET từ đường dẫn /admin/create-product, servlet lấy danh sách các danh mục từ cơ sở dữ liệu và chuyển tiếp thông tin đến trang JSP create-product.jsp để hiển thị giao diện tạo sản phẩm. Với yêu cầu POST, servlet nhận dữ liệu từ form (bao gồm thông tin sản phẩm và ảnh), lưu ảnh vào thư mục cụ thể, tạo đối tượng Product, và sử dụng DAO để thêm sản phẩm vào cơ sở dữ liệu. Sau khi xử lý, servlet chuyển hướng người dùng đến trang danh sách sản phẩm (/admin/products). Nếu xảy ra lỗi trong quá trình thêm, servlet ghi log lỗi ra console.

**DashBoardServlet** chịu trách nhiệm xử lý yêu cầu và hiển thị thông tin tổng quan trên trang dashboard của hệ thống quản trị. Khi nhận yêu cầu GET từ đường dẫn /admin/dashboard, servlet sử dụng DAO để lấy dữ liệu từ cơ sở dữ liệu, bao gồm danh sách sản phẩm, danh mục, đánh giá và chi tiết đơn hàng. Sau đó, các dữ liệu này được truyền đến trang JSP dashboard.jsp để hiển thị.

**DeleteCategoryServlet** chịu trách nhiệm xử lý yêu cầu xóa danh mục khỏi hệ thống quản trị. Servlet lắng nghe các yêu cầu GET tại đường dẫn /admin/delete-category/\*. Khi nhận được yêu cầu, nó trích xuất categoryId từ phần đường dẫn còn lại, kiểm tra tính hợp lệ của ID, và sử dụng DAO để thực hiện xóa danh mục khỏi cơ sở dữ liệu. Sau khi xóa thành công, servlet chuyển hướng người dùng về trang danh sách danh mục (/admin/categories). Nếu ID không hợp lệ hoặc xảy ra lỗi, servlet ghi log lỗi hoặc trả về lỗi 404.

**DeleteProductServlet** chịu trách nhiệm xử lý yêu cầu xóa sản phẩm trong hệ thống quản trị. Servlet lắng nghe yêu cầu GET tại đường dẫn /admin/delete-product/\*, trích xuất productId từ URL, kiểm tra tính hợp lệ và gọi phương thức deleteProduct(productId) của DAO để xóa sản phẩm khỏi cơ sở dữ liệu. Nếu thành công, servlet chuyển hướng người dùng đến trang danh sách sản phẩm (/admin/products); nếu lỗi hoặc productId không hợp lệ, servlet trả về lỗi 404.

**EditCategoryServlet** xử lý việc chỉnh sửa danh mục sản phẩm trong trang quản trị. Khi nhận yêu cầu GET, servlet lấy categoryId từ URL, gọi DAO để lấy thông tin danh mục qua getCategoryByID, rồi chuyển dữ liệu đến trang /admin/edit-category.jsp để hiển thị. Khi nhận yêu cầu POST, servlet nhận thông tin cập nhật từ biểu mẫu (name và description), dùng updateCategory của DAO để lưu thay đổi vào cơ sở dữ liệu, sau đó chuyển hướng về trang danh sách danh mục (/admin/categories). Nếu categoryId không hợp lệ, servlet trả về lỗi 404.

**EditProductServlet** xử lý việc chỉnh sửa sản phẩm trong trang quản trị. Khi nhận yêu cầu GET, servlet lấy productId từ URL, gọi DAO để lấy thông tin sản phẩm và danh sách danh mục, sau đó chuyển dữ liệu tới trang /admin/edit-product.jsp để hiển thị. Khi nhận yêu cầu POST, servlet lấy thông tin cập nhật từ biểu mẫu, bao gồm tên, danh mục, số lượng, giá cả, tỷ lệ giảm giá, trạng thái, và ảnh sản phẩm. Nếu có ảnh mới, servlet lưu ảnh vào thư mục đã chỉ định và cập nhật sản phẩm vào cơ sở dữ liệu; nếu không có ảnh mới, chỉ cập nhật các thông tin còn lại. Sau đó, servlet chuyển hướng về trang danh sách sản phẩm (/admin/products). Nếu productId không hợp lệ, servlet trả về lỗi 404.

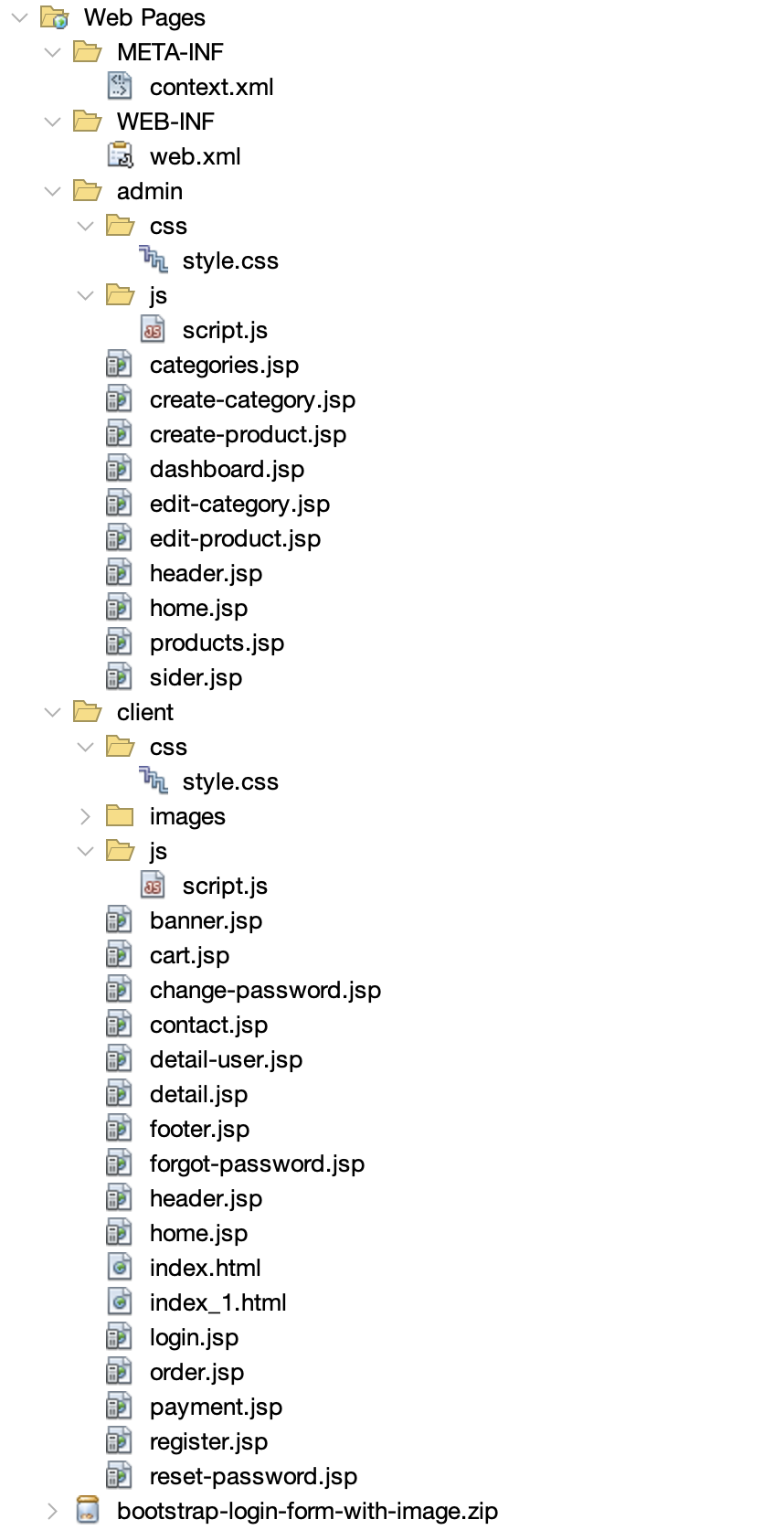
**HomeAdminServlet** xử lý yêu cầu đến trang quản trị của hệ thống. Khi nhận yêu cầu GET, servlet chuyển hướng người dùng tới trang /admin/home.jsp, nơi có thể hiển thị thông tin hoặc giao diện quản lý.

**ManageCategoryServlet** là một servlet quản lý danh mục sản phẩm trong phần quản trị. Khi nhận yêu cầu GET, servlet sẽ lấy tất cả các danh mục từ cơ sở dữ liệu thông qua lớp DAO và truyền chúng vào trang categories.jsp để hiển thị danh sách danh mục cho người dùng.

**ManageProductServlet** là một servlet trong hệ thống quản trị dùng để xử lý các yêu cầu liên quan đến sản phẩm. Khi nhận yêu cầu GET, servlet sẽ truy vấn dữ liệu sản phẩm từ cơ sở dữ liệu thông qua đối tượng DAO và truyền danh sách sản phẩm vào trang products.jsp để hiển thị. Phương thức POST hiện tại không thực hiện xử lý gì, nhưng có thể được mở rộng trong tương lai để hỗ trợ các chức năng như thêm, sửa, hoặc xóa sản phẩm. Servlet này cung cấp giao diện cho quản trị viên để dễ dàng quản lý các sản phẩm trong hệ thống.

## Hệ thống phía Frontend

Cấu trúc mục này sẽ gồm các phần như JSP (nơi mà mã HTML được nhúng với mã Java để tạo nội dung động trên trang web), mục CSS Lưu trữ các file CSS để trang web có thể được trang trí và tạo điều kiện hiển thị hấp dẫn, mục JS Chứa các file JavaScript để xử lý các sự kiện tương tác của người dùng trên trình duyệt, mục image lưu trữ hình ảnh và các tài nguyên hình ảnh cơ bản như logo, background để sử dụng trên trang web.

****

● **META-INF**:Thư mục META-INF chứa các tệp tin cấu hình như context.xml

chứa các cấu hình liên quan đến môi trường thực thi của ứng dụng.

● **WEB-INF**: là một thư mục quan trọng trong ứng dụng web Java và nằm trong thư

mục gốc của ứng dụng web. Trong WEB-INF có chưa web.xml là tệp tin cấu hình

chính của ứng dụng web Java. Nó chứa các khai báo servlet, filter, listener và các

cấu hình khác liên quan đến ứng dụng web.

● **CSS**: chứa các tập tin của Bootstrap được thêm vào dự án phục vụ quá trình làm

giao diện nhanh chóng và thuận tiện hơn

● **images**: Chứa các ảnh (.jpg và .png) phục vụ cho quá trình xây dựng trang web

● **js**: mã JavaScript này dùng để thực hiện các thao tác động trên giao diện người dùng của trang web. Các chức năng chính bao gồm: lọc sản phẩm, phân trang, sắp xếp, xử lý đánh giá, và quản lý trạng thái giao diện.

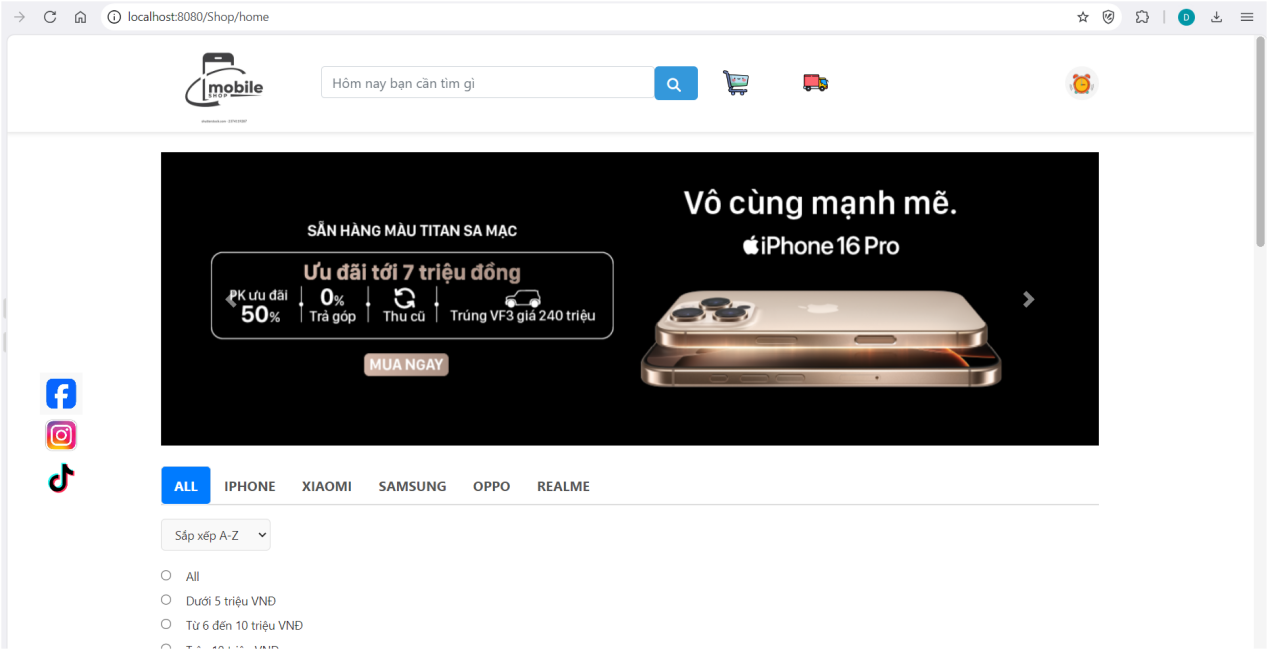
● **bootstrap:** để cấu hình trang login của web

# Chương 4: Chạy và thử nghiệm trang Web

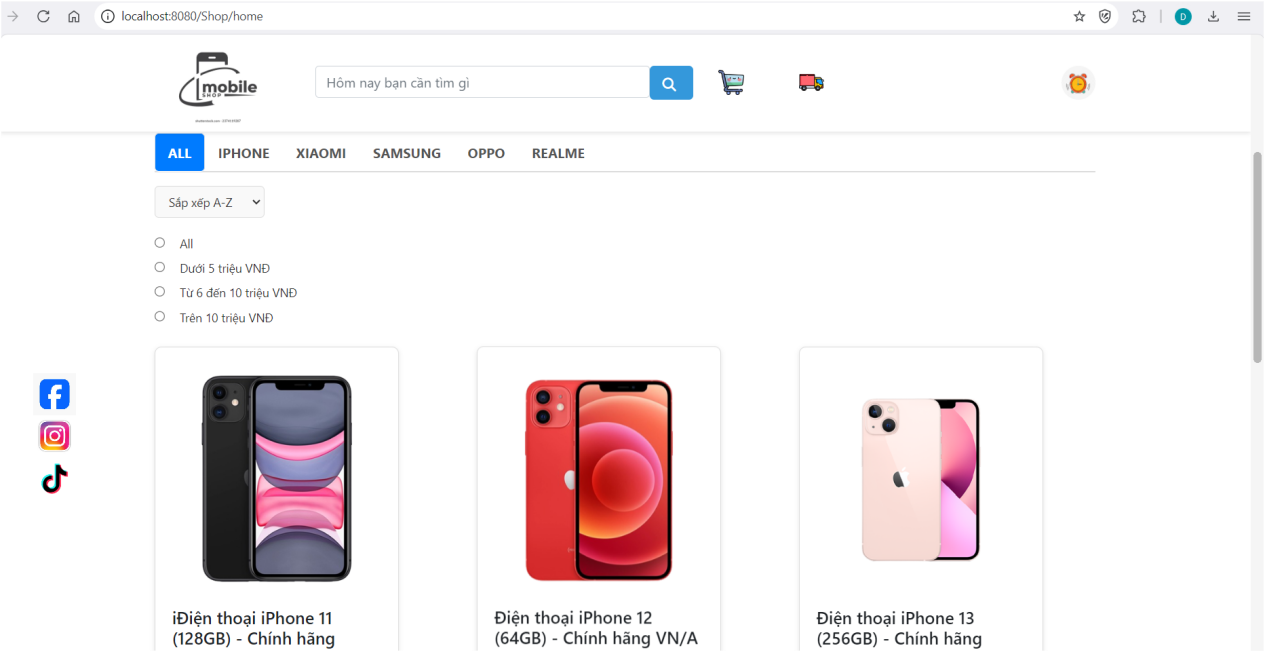
1. **Trang home của web**

Phần đầu của trang chứa icon của trang, thanh tìm kiếm sản phẩm, giỏ hàng, giao hàng, avatar chứa thông tin người dùng.

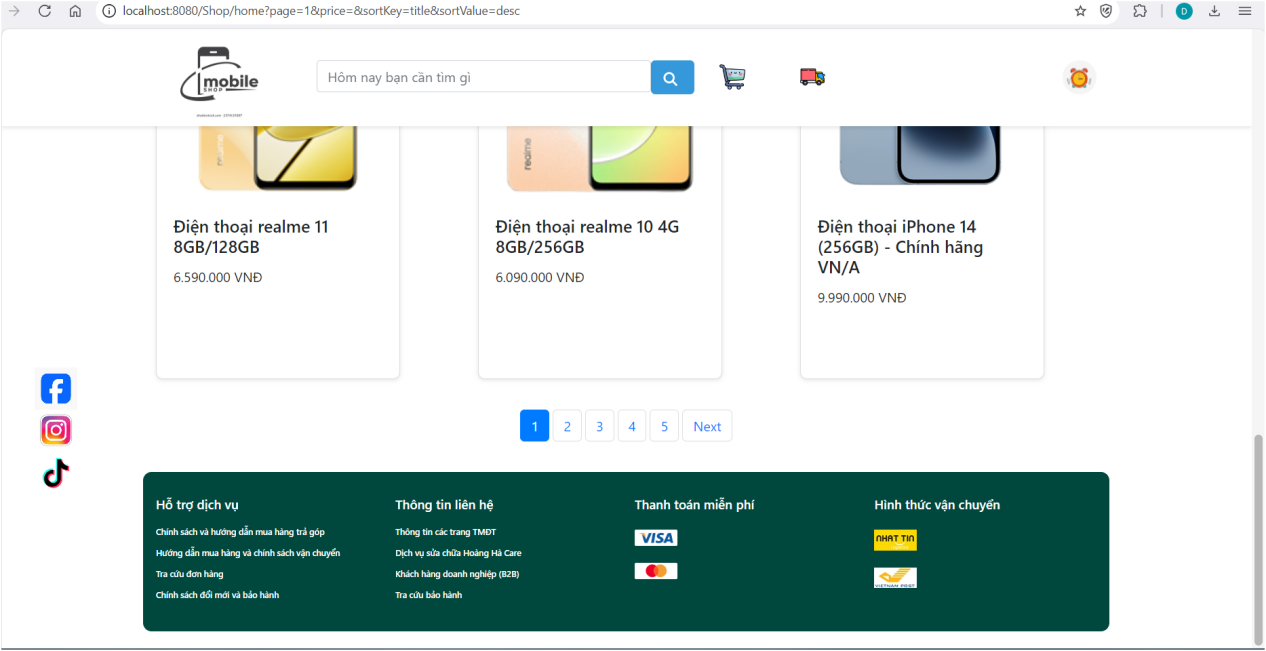
Tiếp đến có banner giới thiệu các sản phẩm điện thoại



Tiếp của phần thân chứa bộ lọc theo hãng, theo tên, theo giá



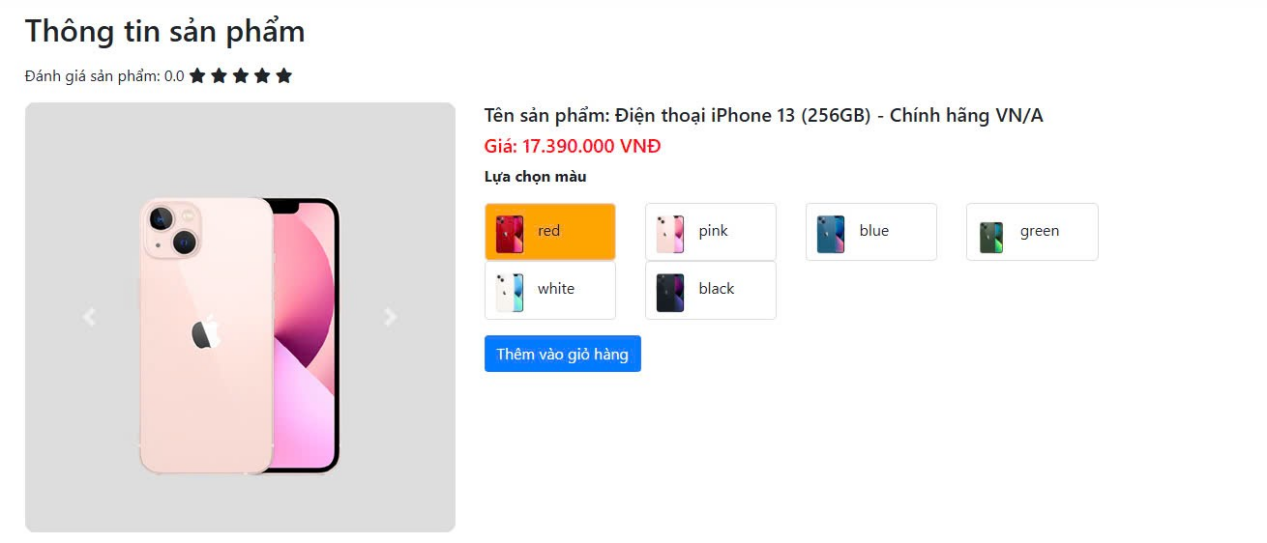
Phần cuối của trang có thêm thông tin và tính năng phân trang



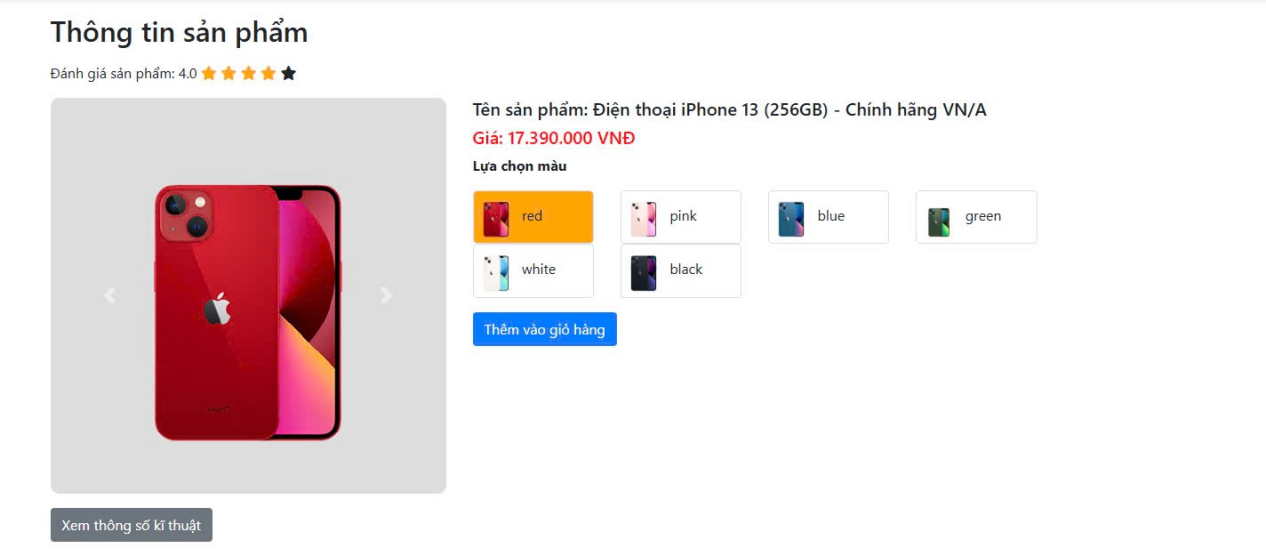
1. **Trang xem chi tiết điện thoại:**

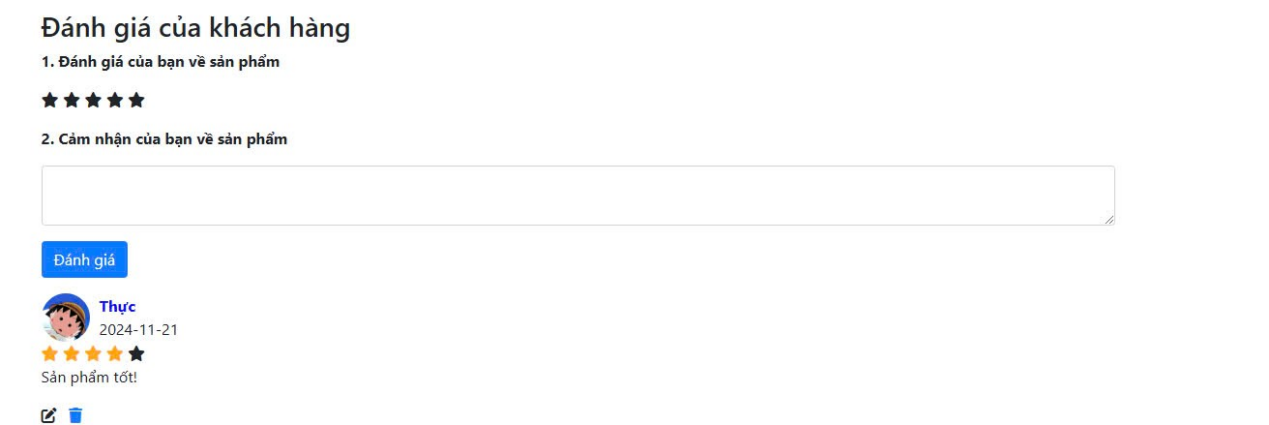
- Trang gồm có hình ảnh điện thoại, tên, giá, màu sắc, thông số kỹ thuật. Bên cạnh còn có phần thêm vào giỏ hàng.

- Phần dưới có thêm chức năng đánh giá sản phẩm.



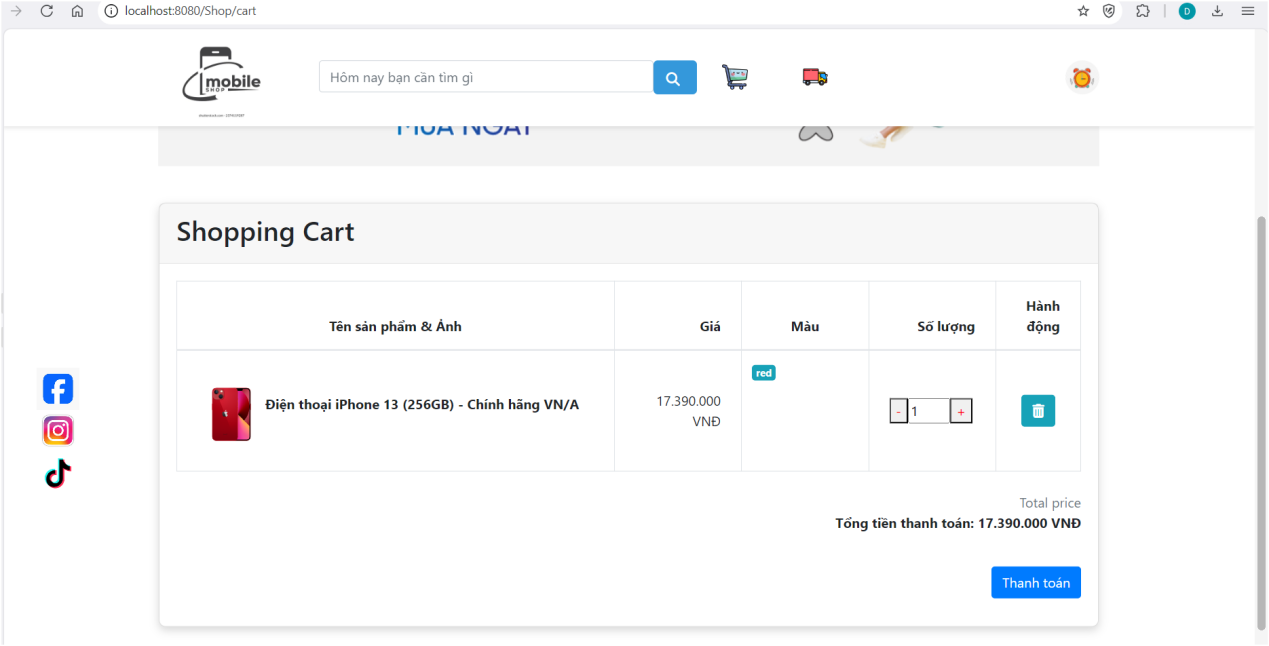






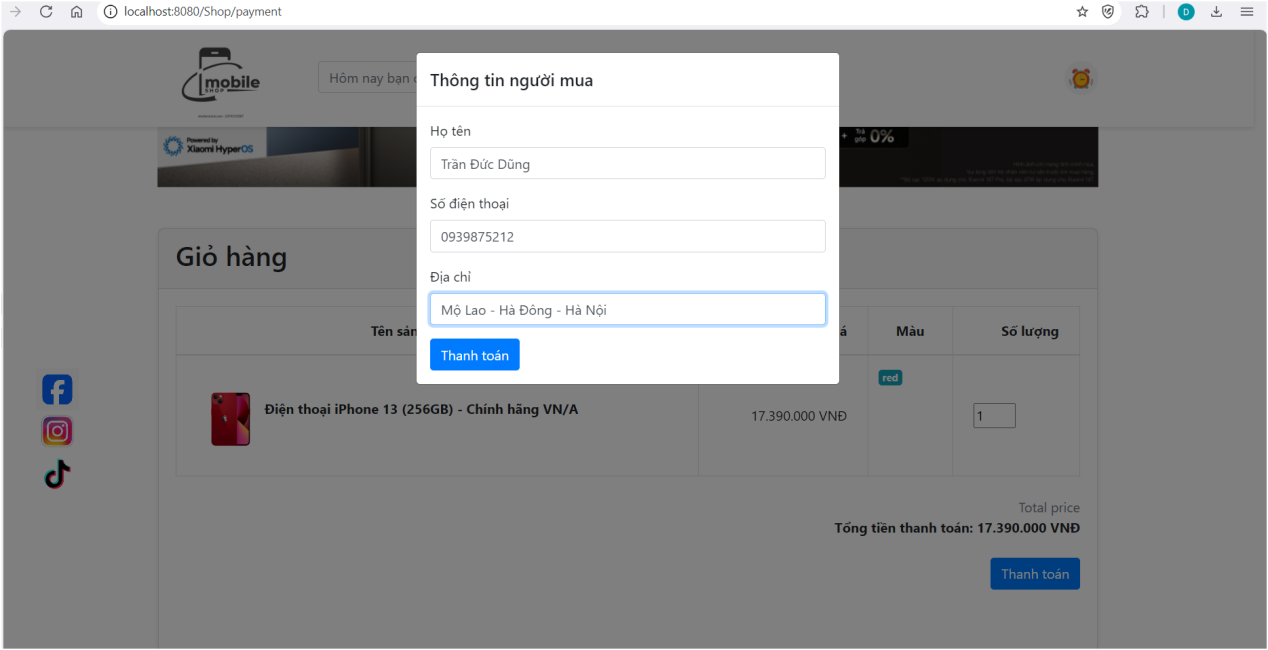
1. **Trang xem chi tiết giỏ hàng**

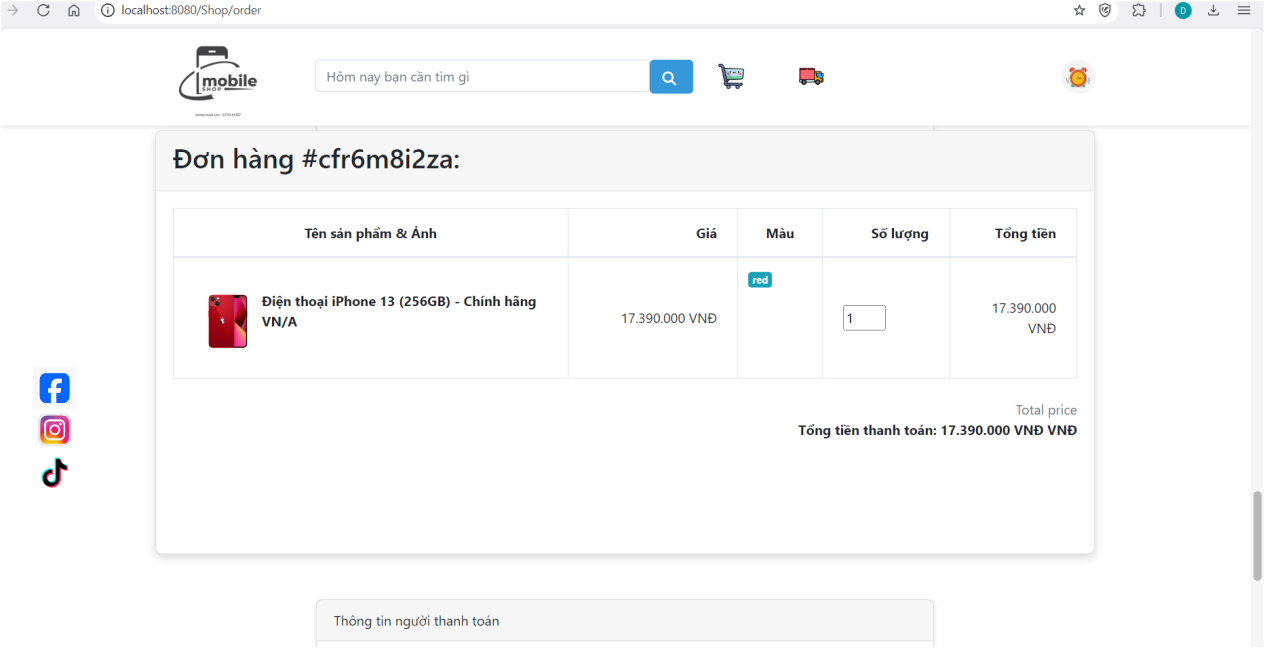
Sau khi khách hàng ấn thêm vào giỏ hàng sẽ được chuyển hướng sang trang chi tiết giỏ hàng. Trang này có thể điều chỉnh số lượng muốn đặt mua hoặc có thể xóa đơn hàng.

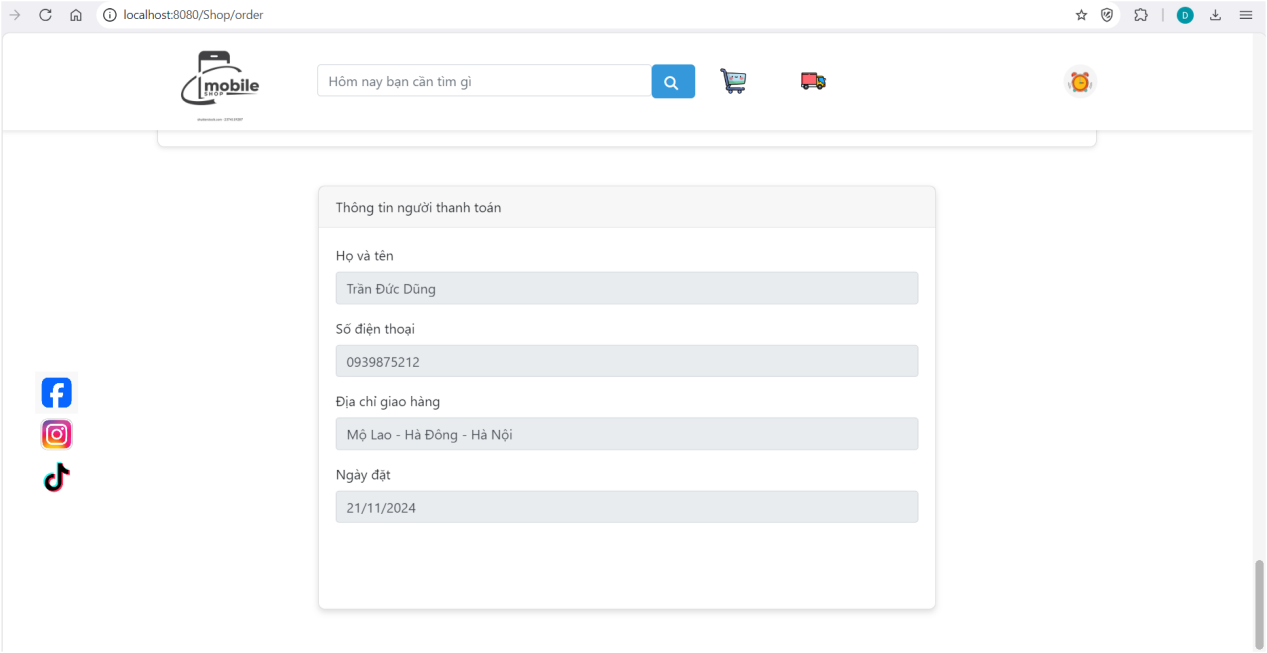


1. **Trang xem chi tiết đơn hàng**

Sau khi khách hàng ấn thanh toán sẽ hiện ra form để điền thông tin cá nhân sau đó ấn thanh toán sẽ được chuyển đến trang đơn hàng.

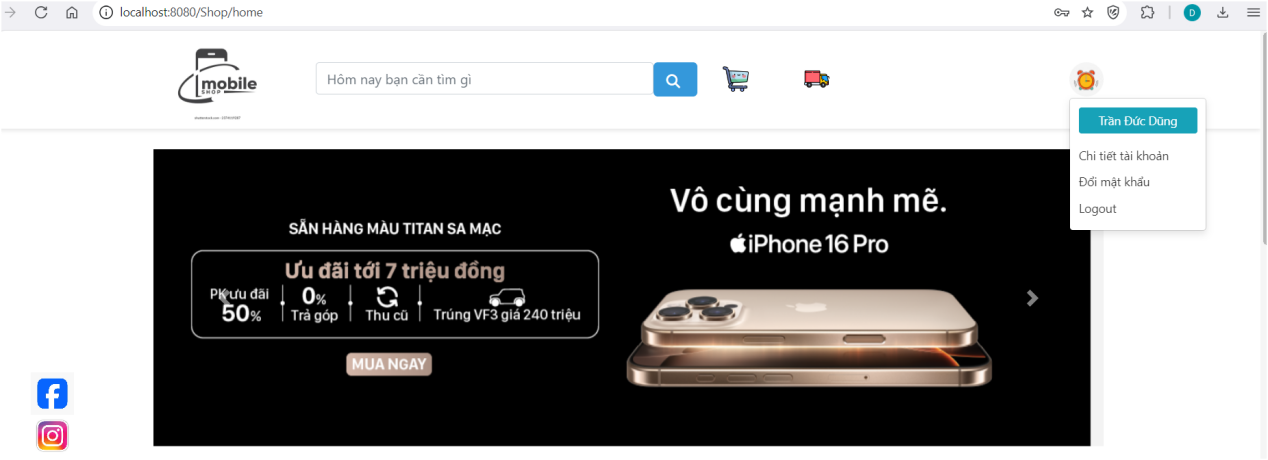




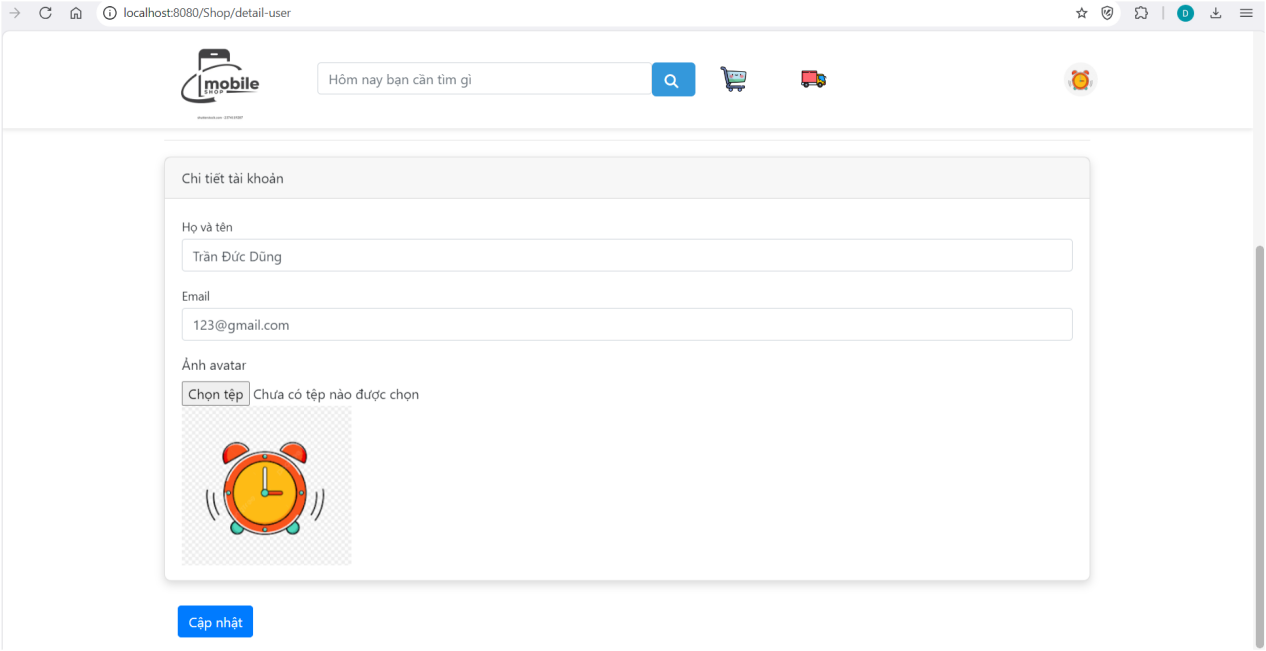


1. **Trang thông tin cá nhân**

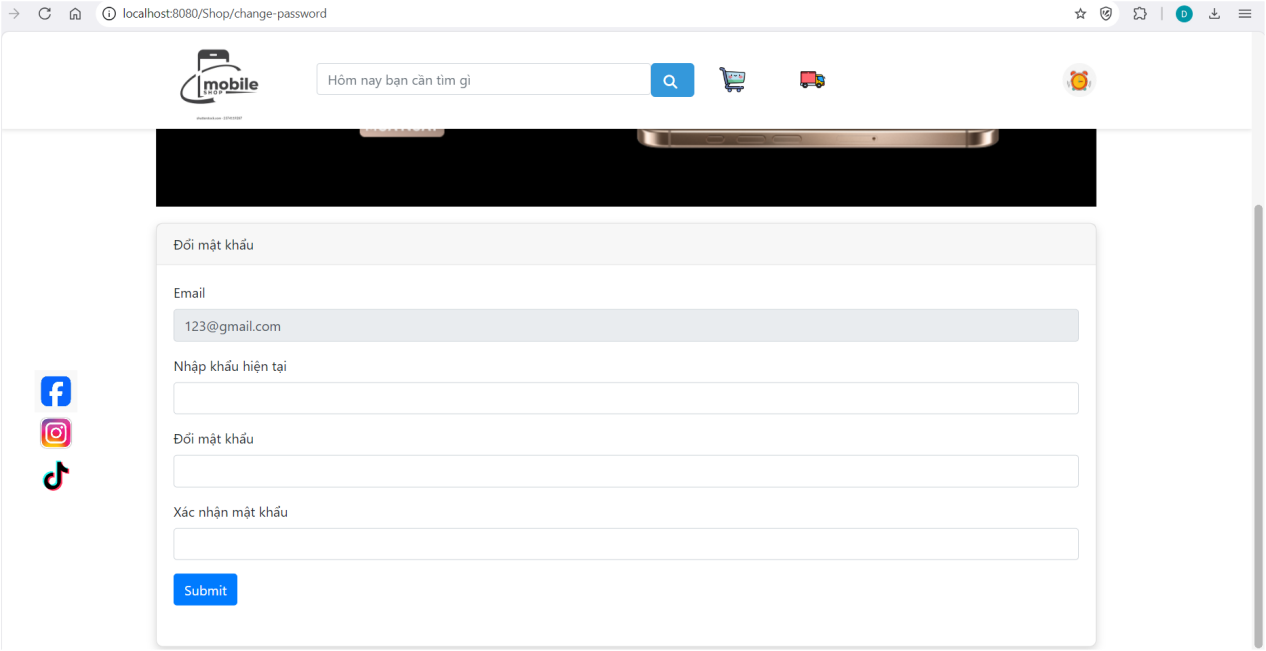
Khi di chuột vào avatar có 3 tùy chọn Chi tiết tài khoản, Đổi mật khẩu và Logout.



Trang Chi tiết tài khoản: có thể thay đổi avatar bằng cách chọn ảnh từ máy

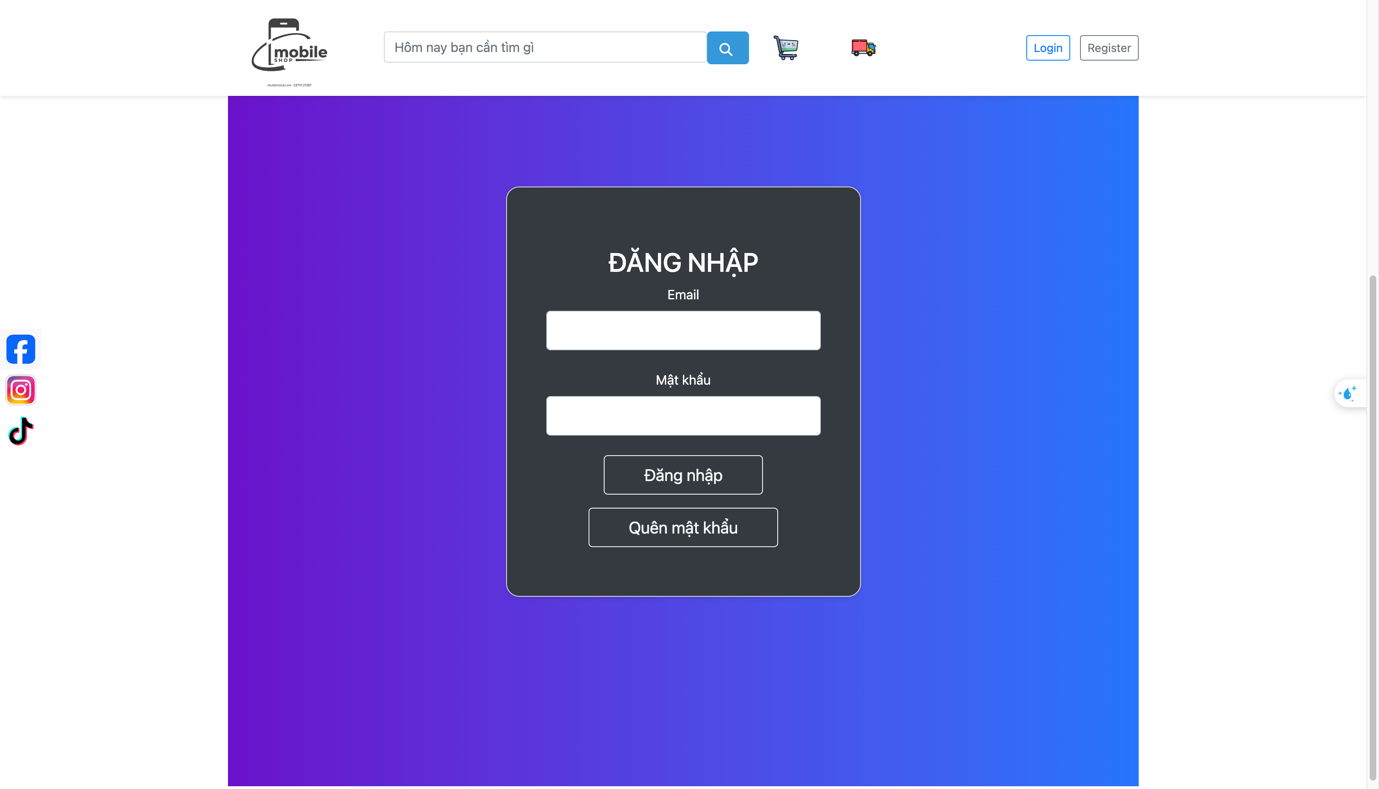


Trang Đổi mật khẩu:

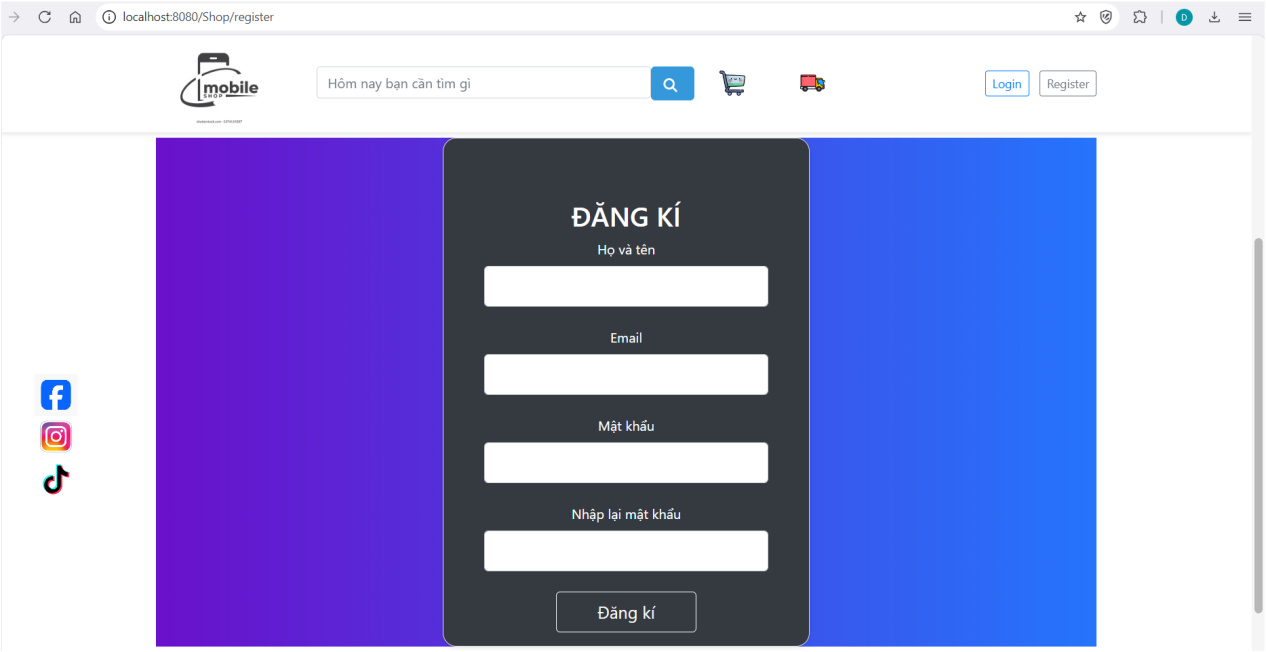


1. **Trang đăng ký, đăng nhập, quên mật khẩu, đổi mật khẩu**

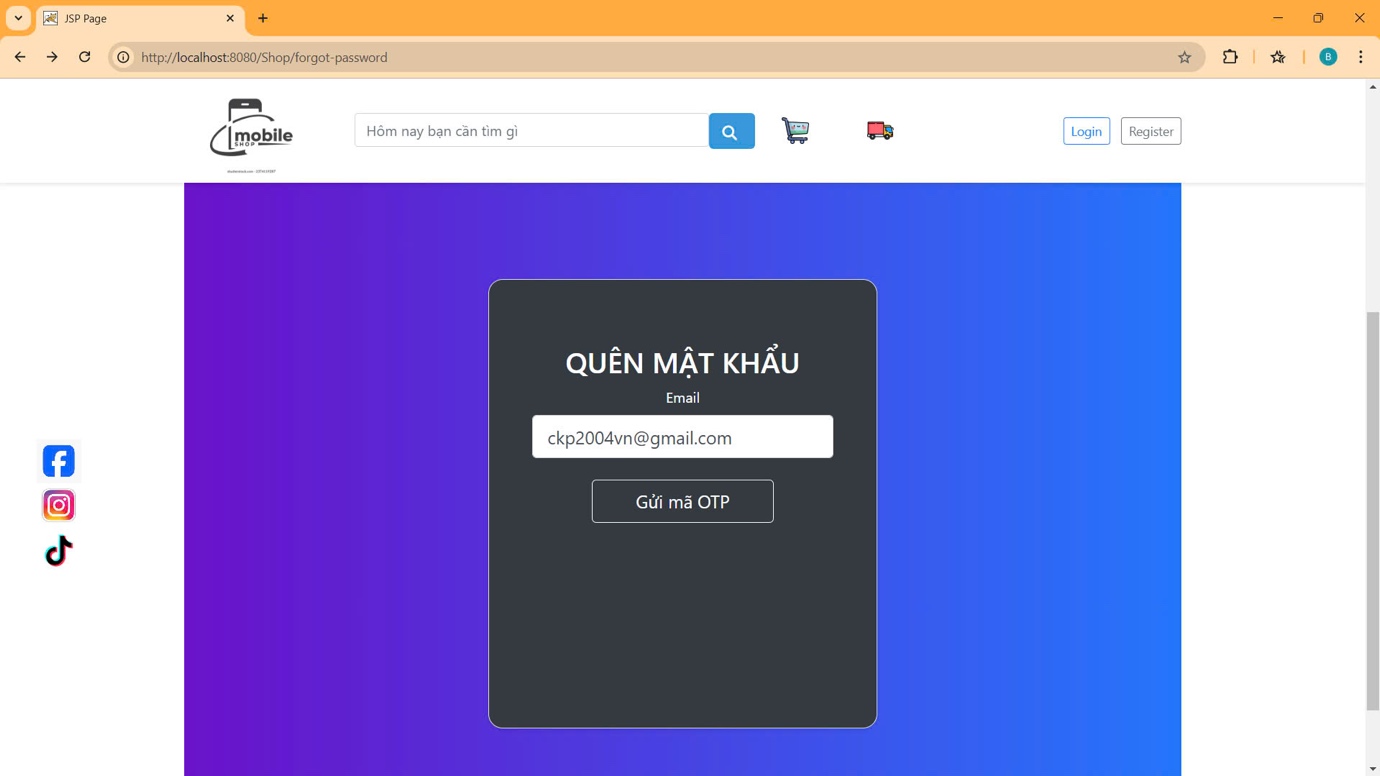
Trang đăng nhập:



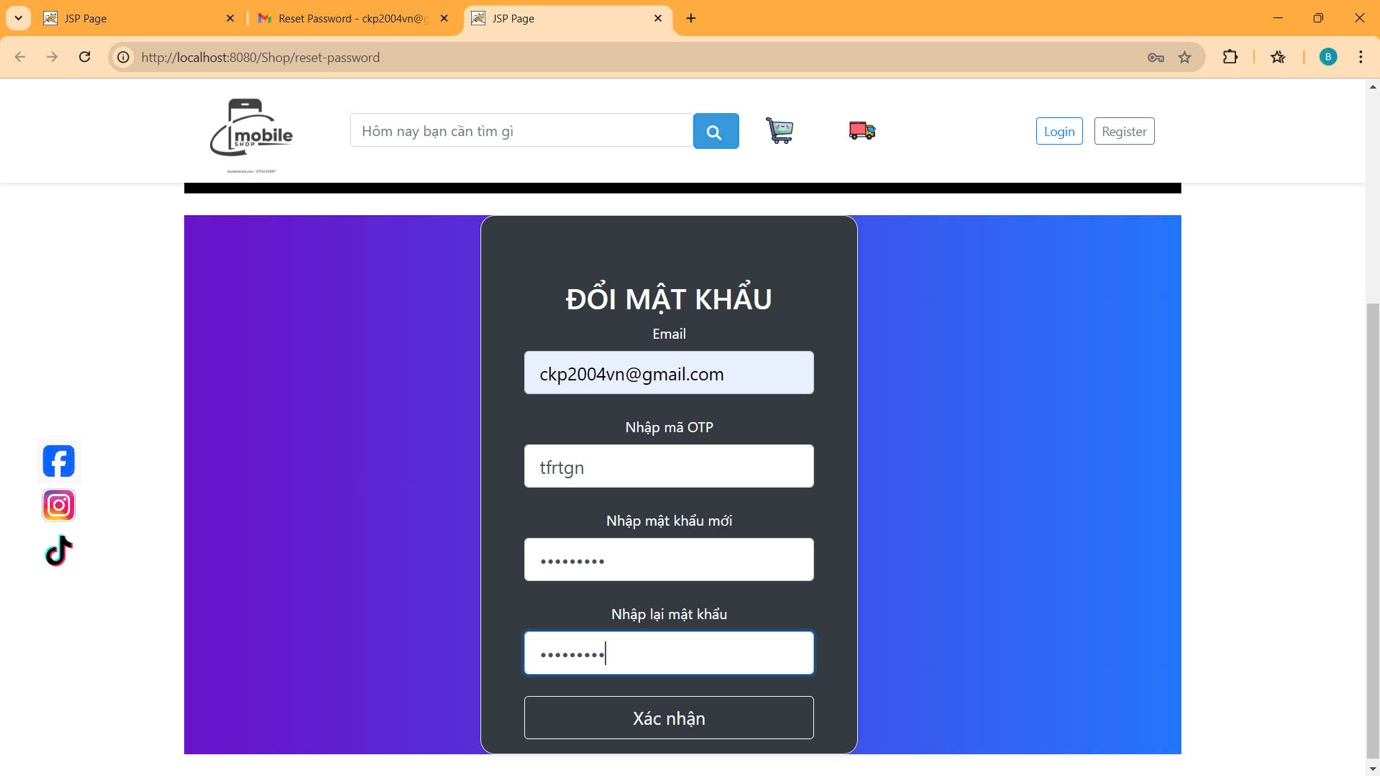
Trang đăng ký:



Trang quên mật khẩu :

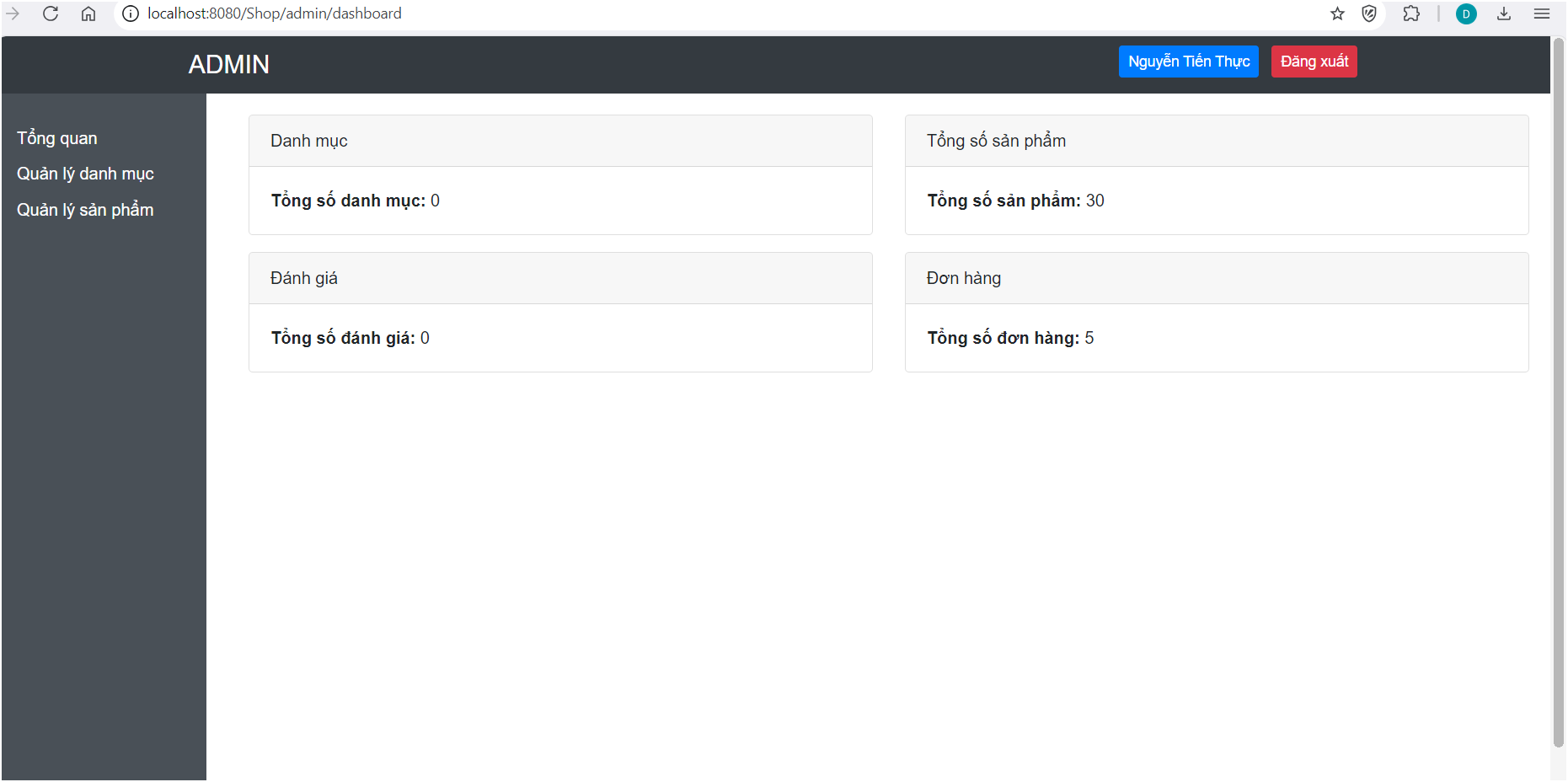
****

Trang đổi mật khẩu:

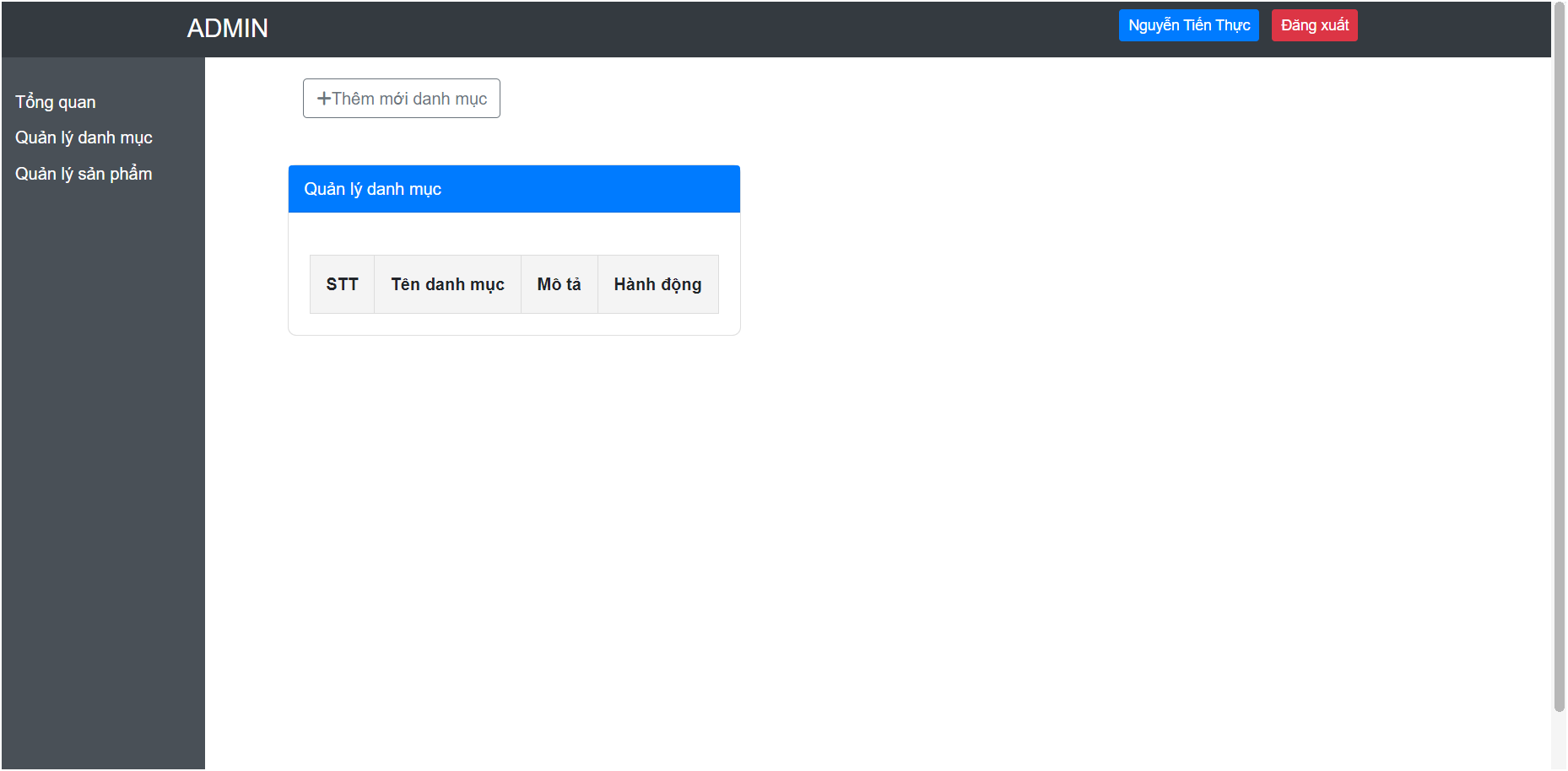


1. **Trang Admin**

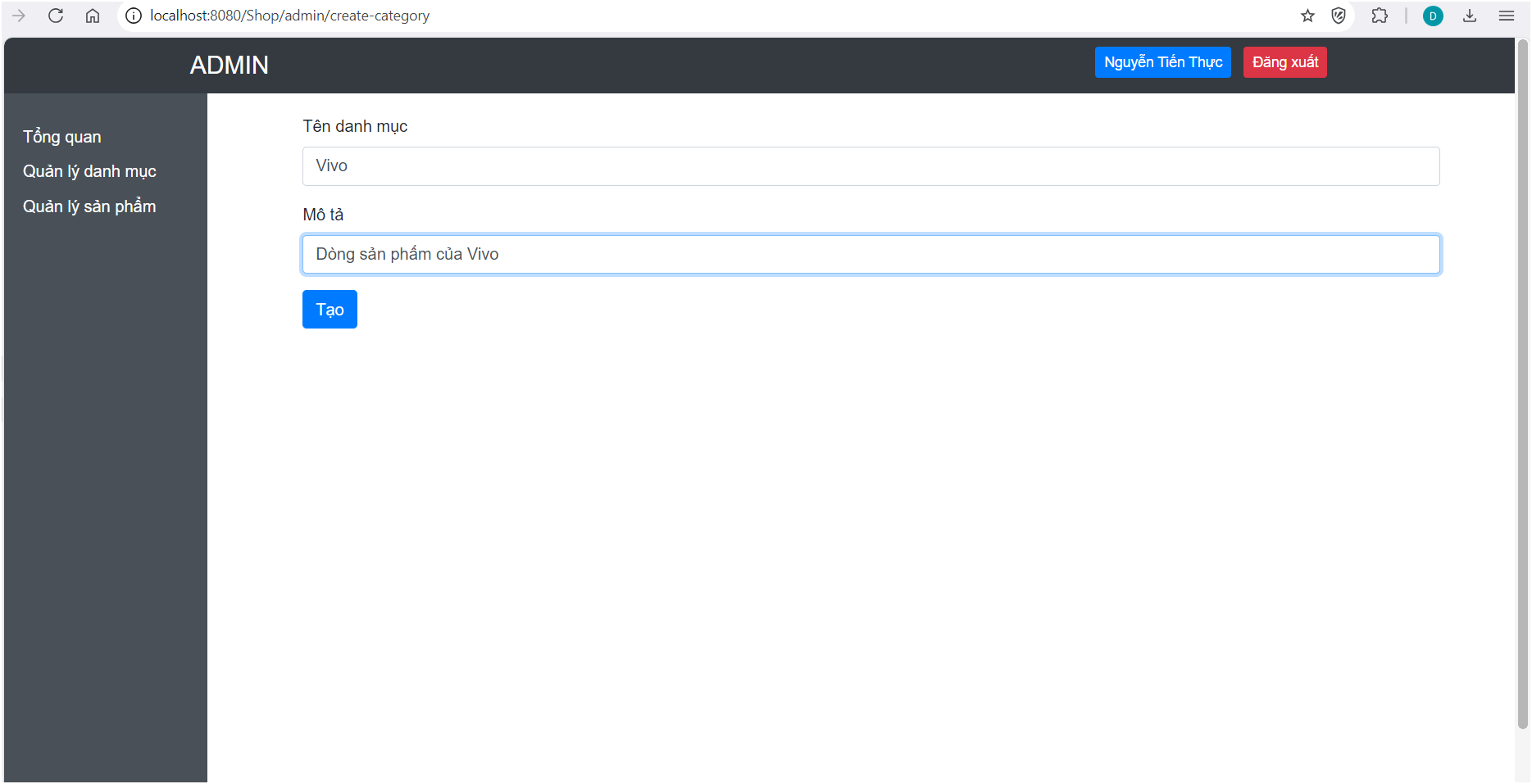
Tổng quan:

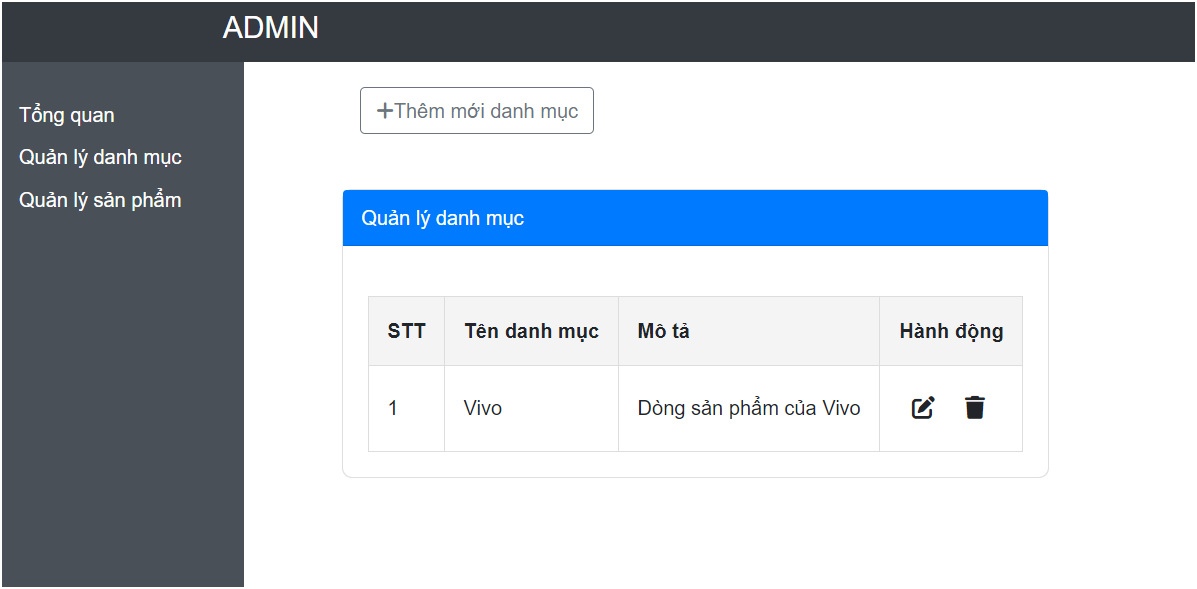


Quản lý danh mục:

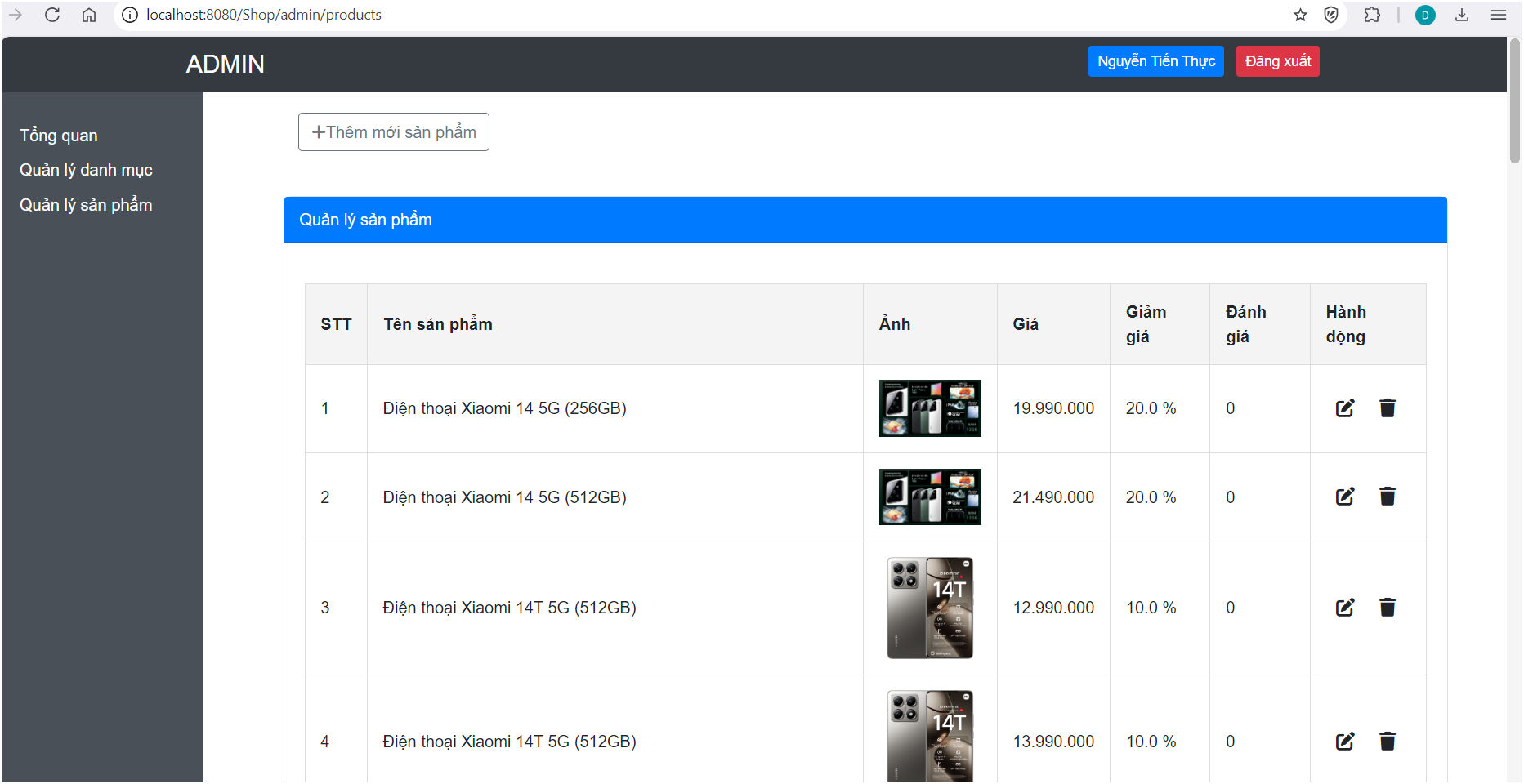


Thêm mới danh mục:

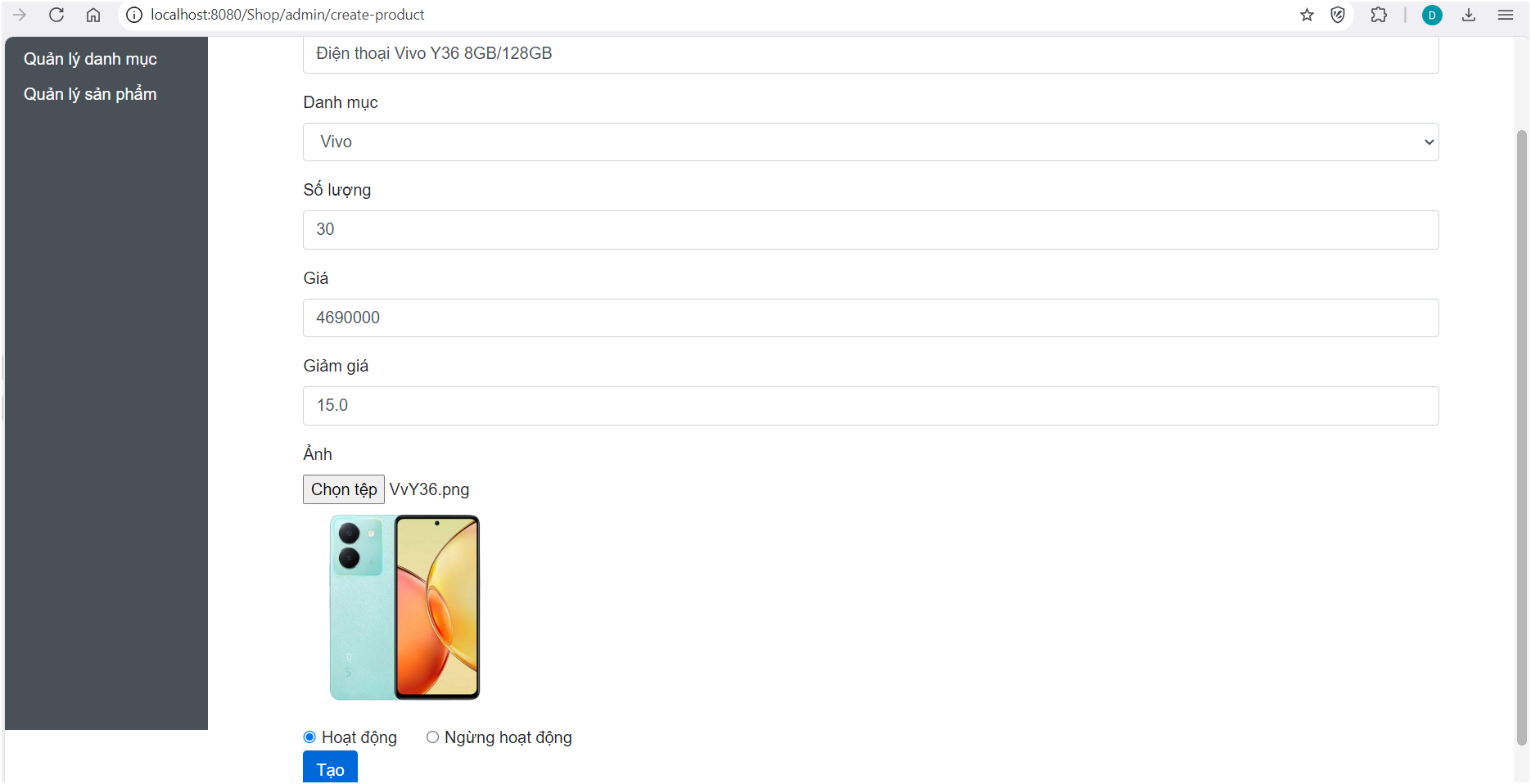




Quản lý sản phẩm: có thể thêm, sửa, xóa bất kỳ sản phẩm nào



Thêm mới sản phẩm:



**VIII. Đánh giá và hướng phát triển trong tương lai**

1. Ưu điểm

- Chương trình có sử dụng truy vấn cơ sở dữ liệu

- Chương trình được viết bằng JSP/Servlet với giao diện diện đơn giản, dễ

nhìn, dễ sử dụng.

- Chương trình được xây dựng có 1 số tính năng cơ bản như: xem thông tin các dòng điện thoại, thông số kỹ thuật, đặt hàng và thanh toán.

1. Nhược điểm

- Chương trình còn đơn giản.

- Nhiều tính năng chưa được hoàn thiện như theo dõi thói quen mua hàng từ khách hàng từ đó gợi ý những sản phẩm mà khách hàng đang muốn mua.

- Chưa có tích hợp các API thanh toán (Momo, ZaloPay, VNPay).

- Chưa có chương trình dành cho khách hàng thân thiết.

1. Hướng phát triển trong tương lai

- Cải tiến giao diện.

- Hiển thị các sản phẩm bán chạy, sản phẩm mới, sản phẩm khuyến mãi.

- Sử dụng AI để đề xuất dựa trên lịch sử mua hàng.

- Tích hợp cổng thanh toán (Momo, ZaloPay, VNPay).

- Tích điểm, nhận ưu đãi dành cho những khách hàng thường xuyên mua hàng.

# TÀI LIỆU THAM KHẢO

[1].<https://www.youtube.com/playlist?list=PLD8zSU7U1L2HJR6ti3WflVvEqBHRAro2h>( khóa học lập trình web bằng Servlet - JSP - cô Vân Anh - PTIT 2022)

1. <https://getbootstrap.com/docs/4.6/getting-started/introduction/> (Boostrap 4.6)
2. <https://fontawesome.com/> (fontawesome)
3. <https://www.youtube.com/watch?v=wkNwSV7ajzI/> (forget pasword)