**TRƯỜNG CAO ĐẲNG FPT POLYTECHNIC**



**BÁO CÁO ASSIGNMENT JAVA 5**

**XÂY DỰNG WEBSITE BÁN MÔ HÌNH**

Giảng viên hướng dẫn: Huỳnh Khắc Duy

Nhóm thực hiện: Nhóm 7

Thành viên: Đinh Tăng Đức Tiến

Thành phố Hồ Chí Minh, 10-06-2004

MỤC LỤC

[I. TỔNG QUAN VỀ DỰ ÁN 4](#_Toc168884097)

[Phạm vi của dự án bao gồm: 4](#_Toc168884098)

[Lý do lựa chọn Spring Boot và ReactJS cho dự án: 4](#_Toc168884099)

[II. CÔNG NGHỆ SỬ DỤNG 5](#_Toc168884100)

[Spring Boot 5](#_Toc168884101)

[ReactJS 5](#_Toc168884102)

[Các thư viện/công cụ hỗ trợ khác 5](#_Toc168884103)

[III. THIẾT KẾ HỆ THỐNG 6](#_Toc168884104)

[Kiến trúc tổng quan 6](#_Toc168884105)

[Mô hình dữ liệu ERD 8](#_Toc168884106)

[Các thành phần chính (components, services, controllers, ...) 9](#_Toc168884107)

[IV. TRIỂN KHAI BACK-END 9](#_Toc168884108)

[Cấu trúc dự án 9](#_Toc168884109)

[Triển khai dữ liệu với Spring Data JPA 9](#_Toc168884110)

[Triển khai API RESTful với Spring MVC 11](#_Toc168884111)

[Xử lý bảo mật với Spring Security 12](#_Toc168884112)

[Thanh toán online 12](#_Toc168884113)

[V. TRIỂN KHAI FONT-END REACT JS 15](#_Toc168884114)

[Cấu trúc dự án 15](#_Toc168884115)

[Thiết kế giao diện người dùng (UI/UX) 15](#_Toc168884116)

[Điều hướng với React Router 15](#_Toc168884117)

[Kết nối với API Back-end 16](#_Toc168884118)

[VI. QUY TRÌNH TRIỂN KHAI VÀ PHÁT TRIỂN 16](#_Toc168884119)

[Môi trường phát triển và công cụ sử dụng 16](#_Toc168884120)

[Quản lý mã nguồn (Git, GitHub, ...) 16](#_Toc168884121)

[Quy trình làm việc (Agile) 16](#_Toc168884122)

[VII. KẾT LUẬN VÀ ĐỀ XUẤT PHÁT TRIỂN 17](#_Toc168884123)

[Tổng kết dự án 17](#_Toc168884124)

[Những thách thức gặp phải và cách giải quyết 17](#_Toc168884125)

[Đề xuất phát triển tiếp theo 17](#_Toc168884126)

[VIII. TÀI LIỆU THAM KHẢO 18](#_Toc168884127)

# TỔNG QUAN VỀ DỰ ÁN

Mục đích của dự án là xây dựng một website thương mại điện tử chuyên bán các mô hình (models) đa dạng, bao gồm mô hình xe hơi, máy bay, tàu thủy, và nhiều loại mô hình khác nhau. Website sẽ cung cấp giao diện trực quan và thân thiện với người dùng, cho phép khách hàng dễ dàng tìm kiếm, xem thông tin chi tiết, và đặt mua các mô hình mong muốn.

## Phạm vi của dự án bao gồm:

* Xây dựng hệ thống quản lý sản phẩm, đơn hàng, và người dùng
* Thiết kế giao diện người dùng (UI/UX) tương tác, hiện đại và thân thiện
* Cung cấp tính năng tìm kiếm, lọc sản phẩm theo các tiêu chí khác nhau
* Xây dựng hệ thống giỏ hàng và thanh toán an toàn
* Triển khai hệ thống quản lý tài khoản người dùng, bao gồm đăng ký, đăng nhập, và quản lý thông tin cá nhân
* Tích hợp các tính năng bổ sung như đánh giá sản phẩm, gợi ý sản phẩm, và hỗ trợ khách hàng

## Lý do lựa chọn Spring Boot và ReactJS cho dự án:

1. **Spring Boot**:

* Spring Boot là một framework back-end mạnh mẽ và linh hoạt, giúp phát triển các ứng dụng web hiệu quả và nhanh chóng.
* Cung cấp môi trường phát triển đơn giản, tự động cấu hình, và tích hợp sẵn nhiều thư viện phổ biến như Spring MVC, Spring Data JPA, Spring Security.
* Hỗ trợ tốt cho việc xây dựng API RESTful, quản lý cơ sở dữ liệu, xử lý bảo mật, và triển khai ứng dụng.
* Cộng đồng người dùng lớn, dễ dàng tìm kiếm hỗ trợ và tài liệu.

1. **ReactJS**:

* ReactJS là một thư viện JavaScript phổ biến và mạnh mẽ để xây dựng giao diện người dùng (UI) cho ứng dụng web.
* Hỗ trợ xây dựng các thành phần UI tái sử dụng, hiệu năng cao và dễ bảo trì.
* Cung cấp tính năng quản lý trạng thái ứng dụng (state management) hiệu quả với Redux hoặc Context API.
* Hỗ trợ tốt cho việc xây dựng ứng dụng đơn trang (Single Page Application - SPA), điều hướng với React Router.
* Cộng đồng người dùng lớn, nhiều thư viện và công cụ hỗ trợ phát triển.

Sự kết hợp của Spring Boot làm back-end và ReactJS làm front-end sẽ giúp xây dựng một ứng dụng web hiện đại, hiệu quả và tương tác tốt với người dùng. Đồng thời, việc lựa chọn hai công nghệ phổ biến này cũng giúp dễ dàng tìm kiếm hỗ trợ, tài liệu và nguồn nhân lực trong quá trình phát triển và bảo trì dự án.

# CÔNG NGHỆ SỬ DỤNG

## ****Spring Boot****

Spring Boot là một dự án được phát triển bởi Pivotal Team, giúp đơn giản hóa việc xây dựng, cấu hình và triển khai ứng dụng Spring. Một số tính năng chính của Spring Boot:

* Khởi tạo dự án nhanh chóng với cơ chế "starters" tích hợp sẵn các thư viện phụ thuộc cần thiết.
* Tự động cấu hình ứng dụng dựa trên môi trường lập trình và thư viện được sử dụng.
* Cung cấp các Embedded HTTP servers (ví dụ: Tomcat, Jetty) để chạy ứng dụng web một cách dễ dàng.
* Hỗ trợ tốt cho việc phát triển các ứng dụng production-ready với các tính năng như metrics, health checks, external configs.
* Tích hợp Spring Data JPA để làm việc với cơ sở dữ liệu một cách dễ dàng.
* Hỗ trợ Spring Security để bảo mật ứng dụng web.

## ****ReactJS****

ReactJS là một thư viện JavaScript mã nguồn mở được phát triển và duy trì bởi Facebook. Nó được sử dụng chủ yếu để xây dựng giao diện người dùng (UI) cho ứng dụng web và di động. Một số tính năng chính của ReactJS:

* Sử dụng cú pháp JSX để kết hợp HTML và JavaScript, giúp viết mã một cách trực quan và dễ dàng.
* Xây dựng UI bằng cách chia nhỏ thành các components tái sử dụng.
* Sử dụng Virtual DOM để tối ưu hiệu suất bằng cách chỉ cập nhật lại các phần thay đổi trong DOM thực tế.
* Hỗ trợ quản lý trạng thái ứng dụng (state management) với Redux hoặc Context API.
* Cung cấp các thư viện hỗ trợ như React Router cho điều hướng trong ứng dụng.
* Cộng đồng lớn với nhiều thư viện và công cụ hỗ trợ phát triển.

## ****Các thư viện/công cụ hỗ trợ khác****

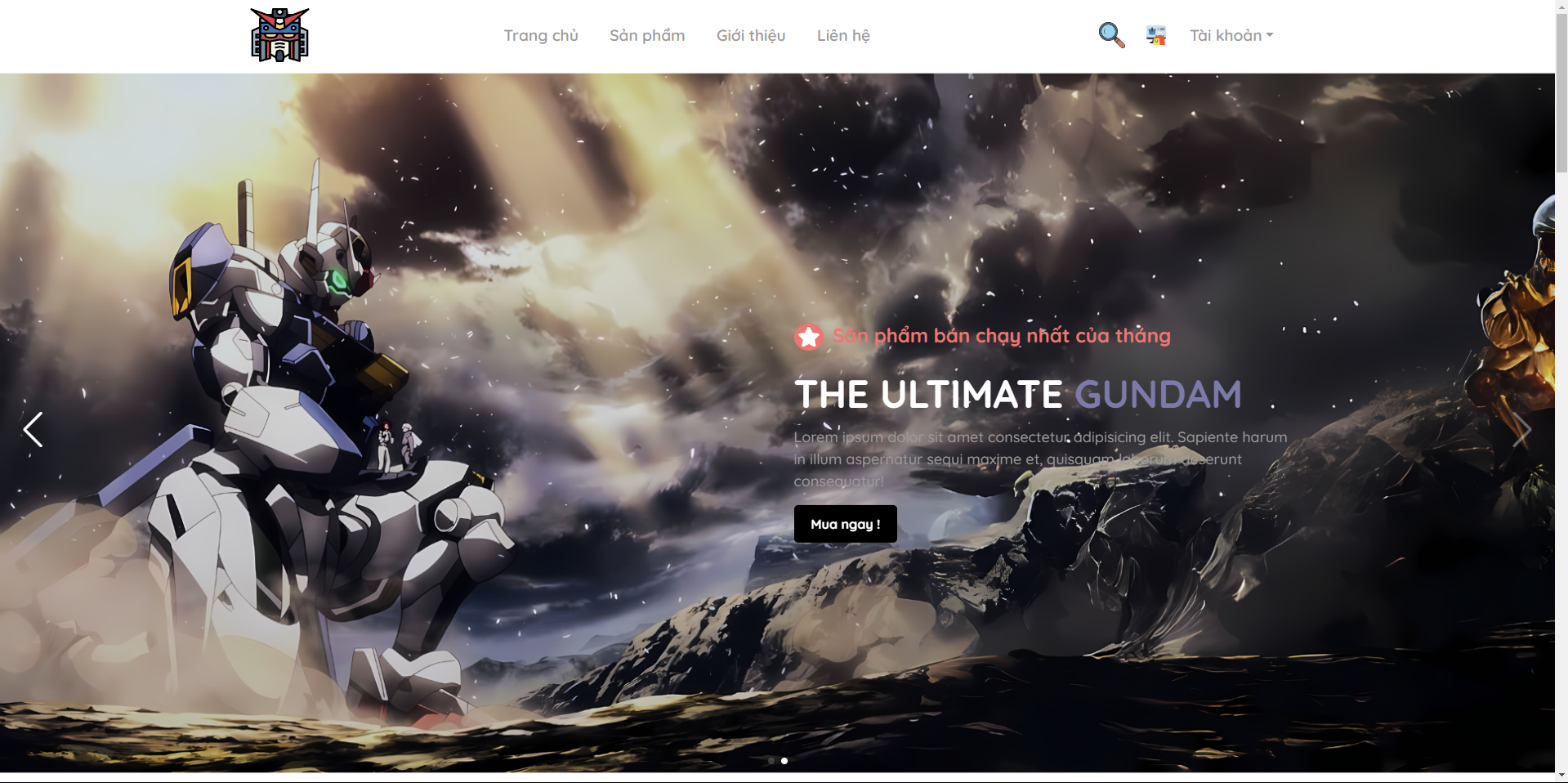
* **Spring Data JPA**: Module của Spring Framework cung cấp lớp trừu tượng hóa để làm việc với các cơ sở dữ liệu quan hệ thông qua Java Persistence API (JPA).
* **React Router**: Thư viện điều hướng cho ReactJS, giúp xây dựng các ứng dụng Single Page Application (SPA) với URL định tuyến.
* **Axios**: Thư viện JavaScript để gửi yêu cầu HTTP từ phía client đến server.
* **Bootstrap**: Framework CSS phổ biến để thiết kế giao diện người dùng responsive và hiện đại.
* **SQL Server**: Cơ sở dữ liệu nơi để lưu trữ các dữ liệu của trang web.
* **MS SQL Driver**: Driver kết nối Spring Boot App với CSDL SQL Server
* **Lombook**: hỗ trợ viết các getter/setter và contructor nhanh chóng

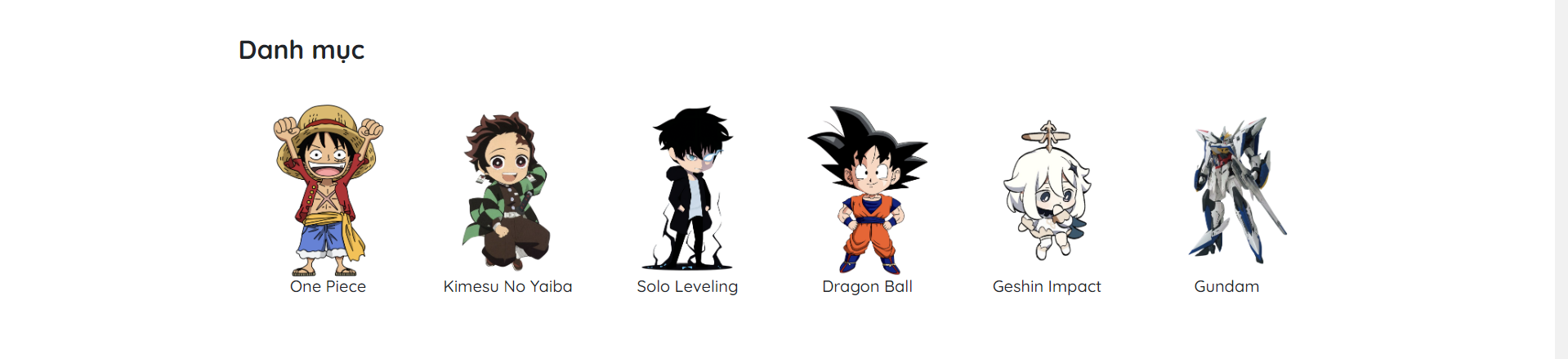
Các công nghệ và thư viện này sẽ giúp xây dựng một ứng dụng web hiện đại, đáp ứng các yêu cầu về hiệu suất, bảo mật, và trải nghiệm người dùng tốt.

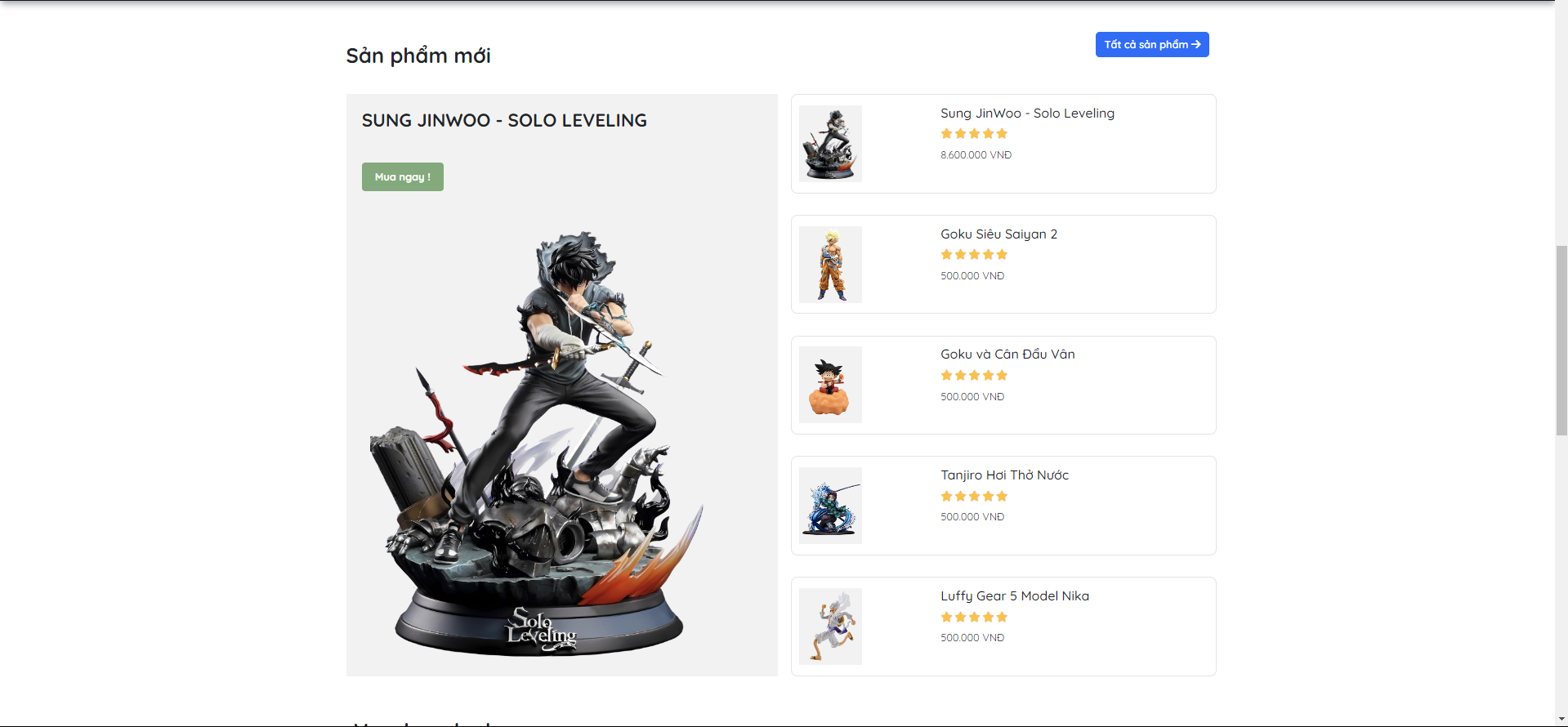
# THIẾT KẾ HỆ THỐNG

## Kiến trúc tổng quan

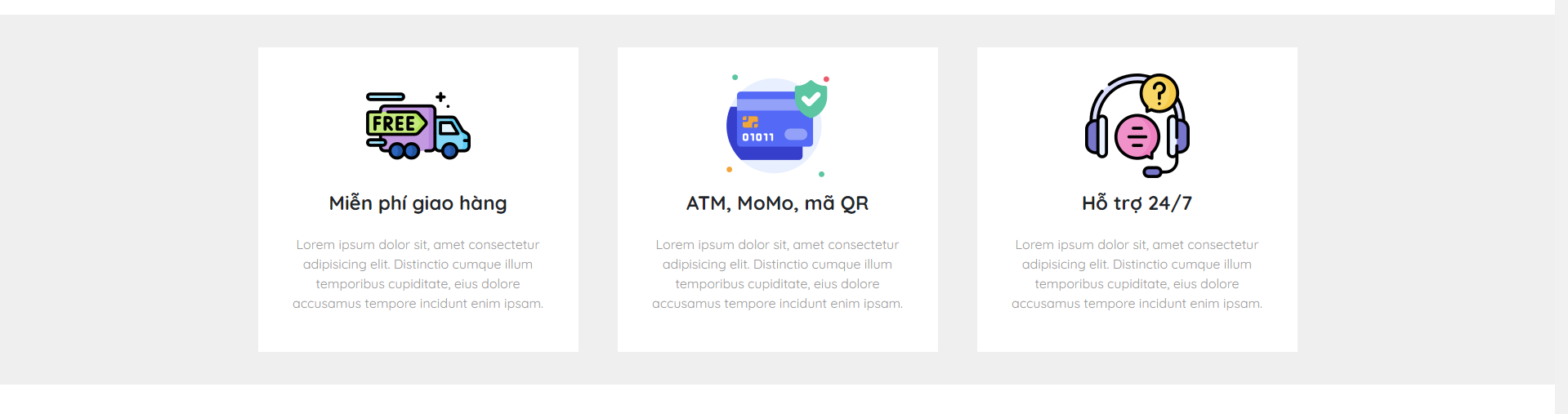
* Website hướng tới sự thân thiện với người dùng nên vẫn đầy đủ Header, Main, Footer. Header chứa các link điều hướng tới các trang như **sản phẩm**, **liên hệ**, **giới thiệu** và một số chức năng như **đăng ký/đăng nhập**, **tìm kiếm**, **giỏ hàng**. Main chứa các danh mục (loại hàng) mà người chủ đang kinh doanh, một vài sản phẩm phổ biến, được dựa trên số liệu phân tích từ hóa đơn mua hàng của người dùng. Có hiển thị được các sản phẩm mới được nhập về những sản phẩm đang trend hoặc sắp tới sẽ lên trending, hiển thị được quà tặng cho hóa đơn từ 1tr000 trở lên, bạn sẽ được tặng 1 bộ dụng cụ để lắp ghép mô hình. Hiển thị được các phương thức dịch vụ mà web mang lại như là: **miễn phí giao hàng**, **thanh toán bằng ATM / MOMO / QR CODE**, **hỗ trợ khách hàng 24/7**. Footer hiển thị các thông tin để khách hàng có thể liên hệ và một vài đường link mà khách hàng có thể sẽ cần.

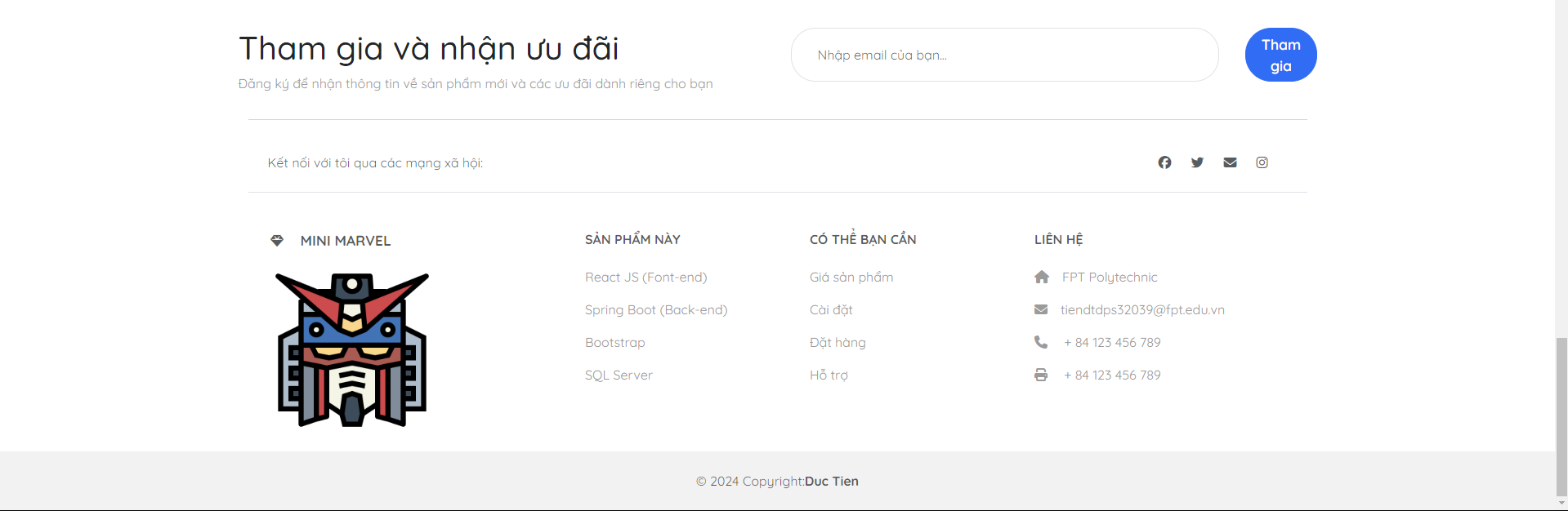




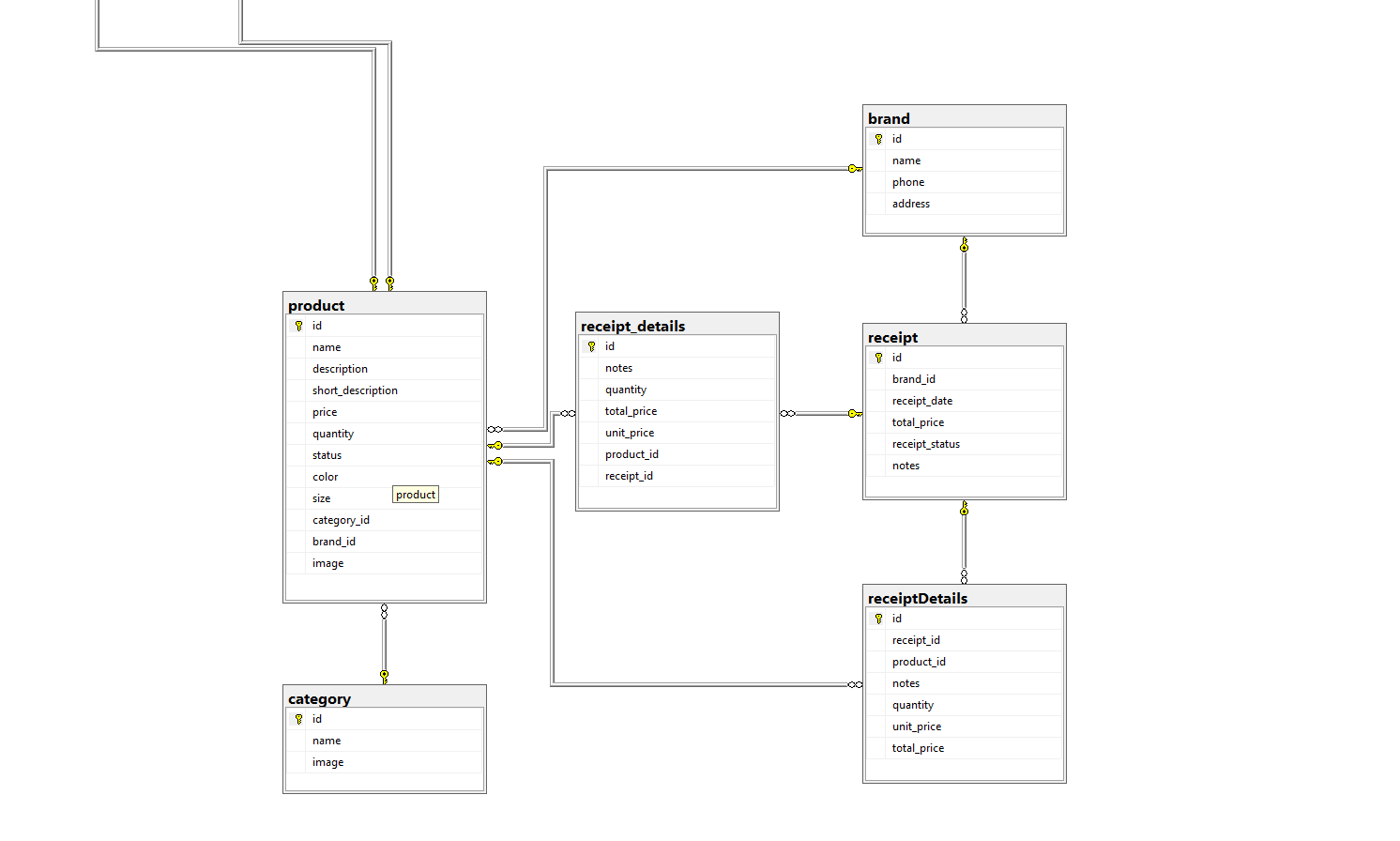
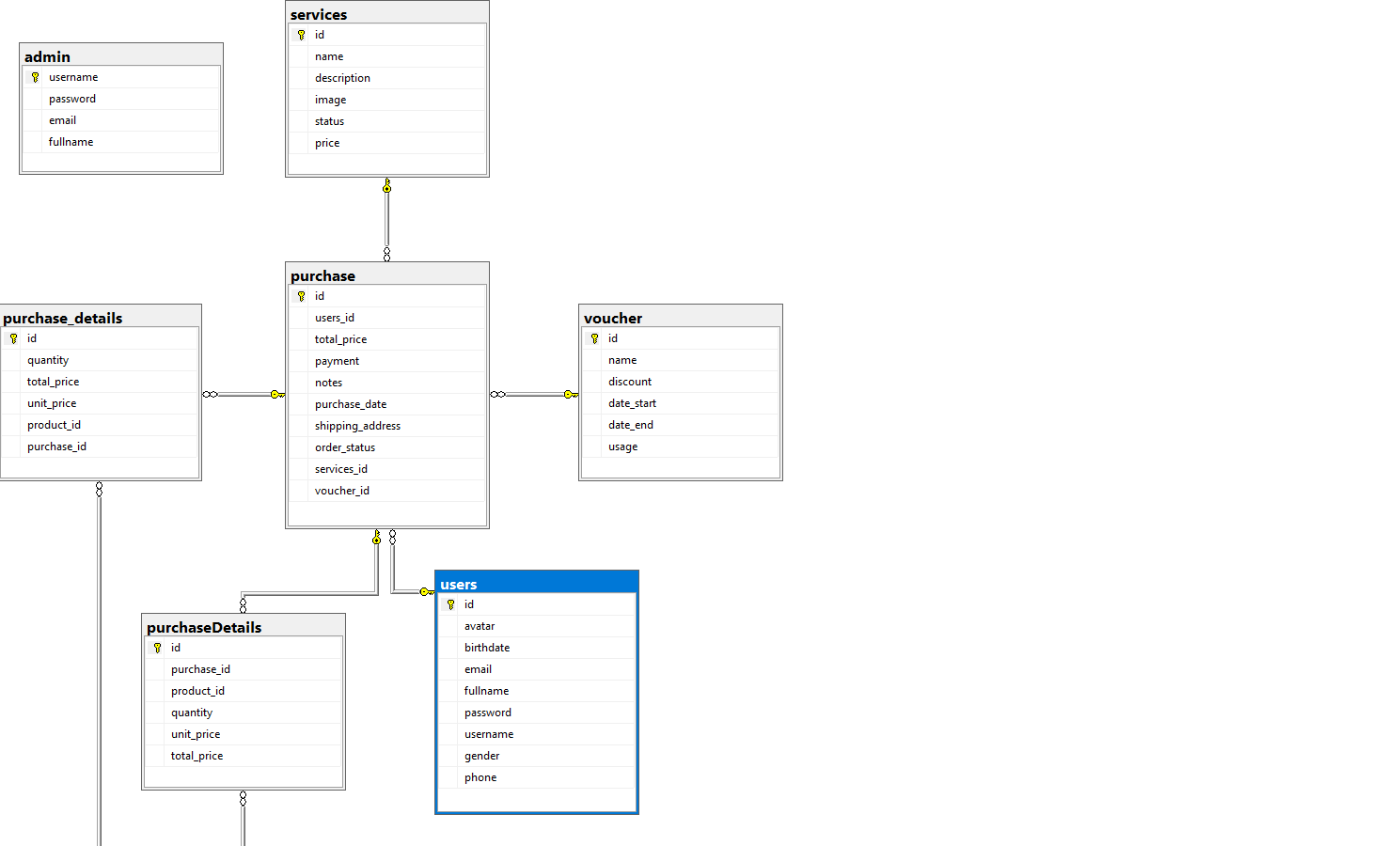








## Mô hình dữ liệu ERD



## Các thành phần chính (components, services, controllers, ...)

Components (reactjs)

* Home: chứa các component khác như banner, cart, login, sigup,…
* App.jsx: cấu hình các router cho website

Service (spring boot)

* Các service giúp thực hiện các công viêc CRUD của các bảng như category, product, user, voucher,…

Controller (sping boot)

* Các controller đều sử dụng annotation @RestController, @CrossOrigin nhằm thực hiện việc API RESTfulvà cho phép truy xuất từ localhost:5050 ( fontend ).

# TRIỂN KHAI BACK-END

## Cấu trúc dự án

 src/main/java/com/minimarvel: Chứa mã nguồn Java

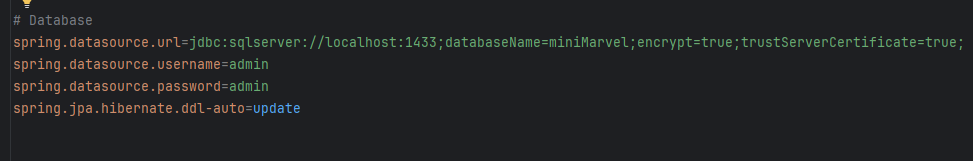
* controller: Chứa các lớp controller xử lý yêu cầu từ client và trả về phản hồi.
* service: Chứa logic nghiệp vụ của ứng dụng.
* repository: Chứa các lớp tương tác với cơ sở dữ liệu.
* entity: Chứa các lớp đối tượng dữ liệu (entity).
* model/dto: chứa các thực thể dto được viết từ các entity
* model/mapper: chứa các lớp có phương thức chuyển đổi giữa enity và dto

 src/main/resources: Chứa tài nguyên như file cấu hình, dữ liệu tĩnh,...

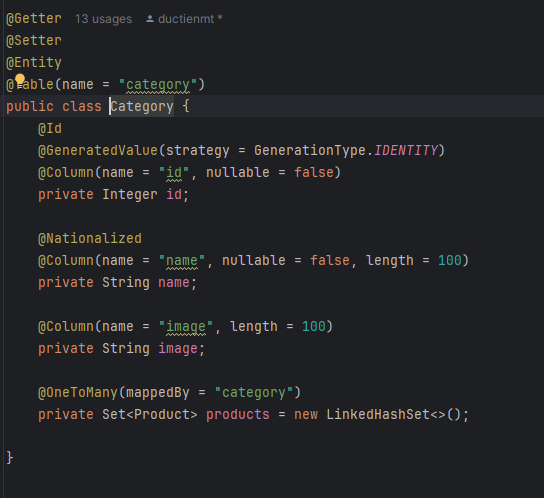
## Triển khai dữ liệu với Spring Data JPA

Spring Data JPA cung cấp lớp trừu tượng để làm việc với cơ sở dữ liệu quan hệ thông qua Java Persistence API (JPA). Các bước chính:

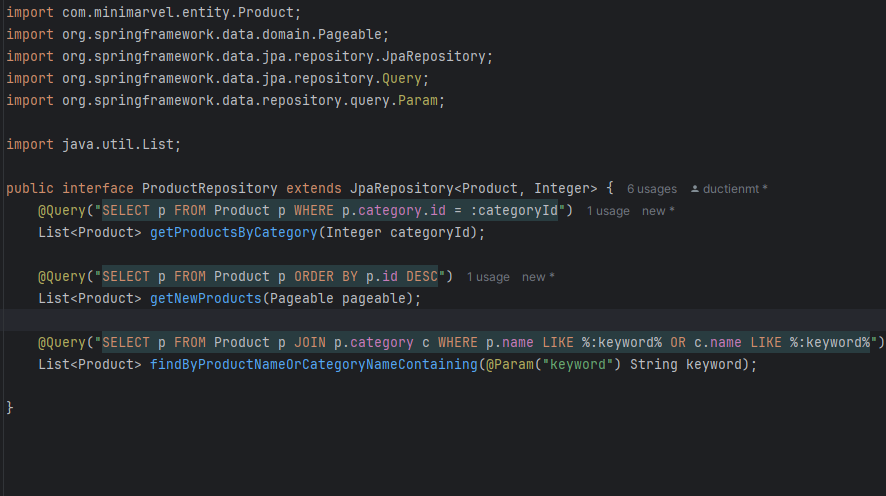
1. **Cấu hình kết nối cơ sở dữ liệu** trong file application.properties



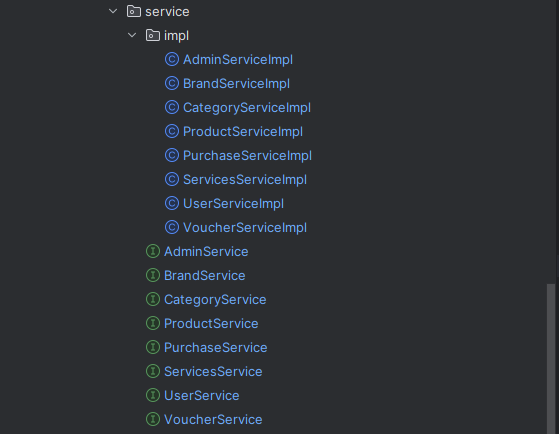
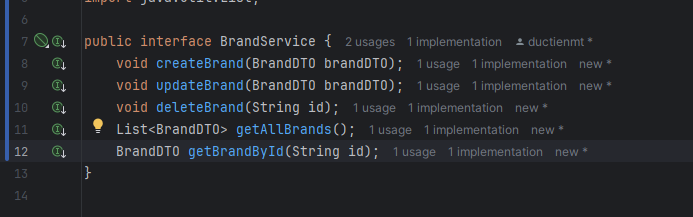
1. **Tạo entity class** tương ứng với bảng trong cơ sở dữ liệu, sử dụng các annotation như @Entity, @Id, @OneToMany,...

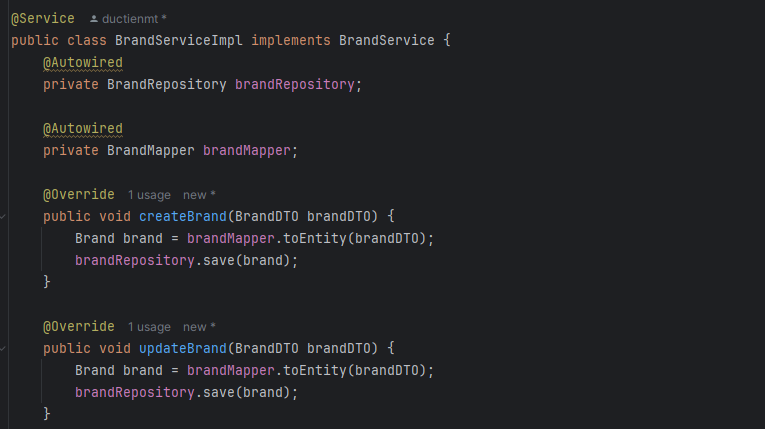


1. **Kế thừa interface JpaRepository** trong lớp repository, cung cấp các phương thức CRUD cơ bản.



1. **Triển khai logic nghiệp vụ** trong lớp service, sử dụng lớp repository để tương tác với cơ sở dữ liệu.





## ****Triển khai API RESTful với Spring MVC****

Spring MVC hỗ trợ xây dựng API RESTful một cách đơn giản với các annotation như @RestController, @RequestMapping, @GetMapping, @PostMapping,...

1. **Tạo controller class** và sử dụng @RestController để đánh dấu đây là một controller xử lý yêu cầu HTTP.
2. **Định nghĩa các mapping method** tương ứng với các đường dẫn API, sử dụng @RequestMapping hoặc các annotation tương đương.
3. **Xử lý yêu cầu và trả về phản hồi** trong các phương thức của controller, có thể sử dụng @RequestBody để đọc dữ liệu từ request, @PathVariable để truy cập tham số trong URL.
4. **Quản lý trạng thái phản hồi HTTP** bằng cách trả về các đối tượng ResponseEntity với mã trạng thái phù hợp.

## ****Xử lý bảo mật với Spring Security****

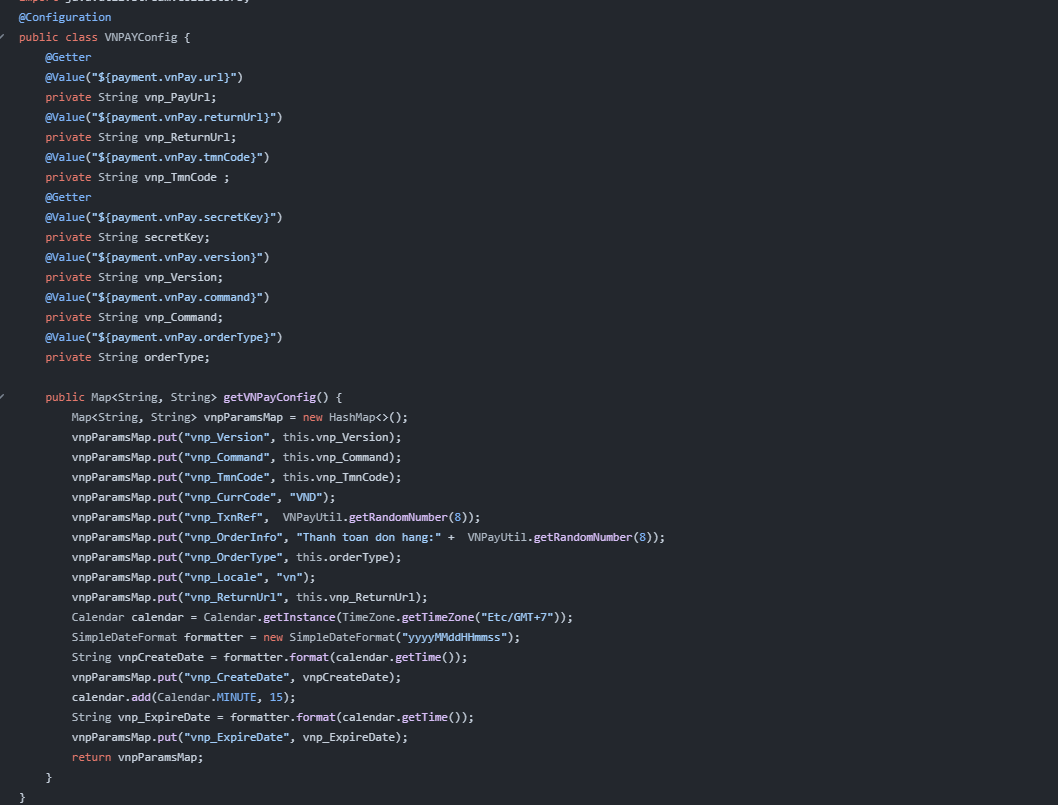
Hiện tại tính năng này còn đang gặp lỗi nên chưa đưa vào hoạt động

**Spring Security** cung cấp các tính năng bảo mật cho ứng dụng web, bao gồm xác thực người dùng, phân quyền truy cập, bảo vệ chống tấn công,...

1. **Cấu hình bảo mật** tạo một lớp cấu hình riêng kế thừa từ WebSecurityConfigurerAdapter.
2. **Tạo lớp UserDetailsService** để tải thông tin người dùng từ cơ sở dữ liệu hoặc nguồn dữ liệu khác.
3. **Cấu hình xác thực** bằng cách ghi đè phương thức configure(AuthenticationManagerBuilder) và sử dụng userDetailsService.
4. **Cấu hình phân quyền truy cập** bằng cách ghi đè phương thức configure(HttpSecurity) và sử dụng các phương thức như antMatchers(), hasRole(), permitAll(),...
5. **Áp dụng bảo mật cho API** bằng cách sử dụng annotation @PreAuthorize hoặc @Secured trong controller.

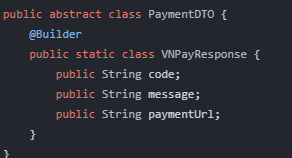
## Thanh toán online

**Thanh toán online** thông qua ví **VNPAY và MOMO** (còn gặp lỗi nên chưa đưa vào hoạt động).

**Cấu hình các đường dẫn** cần thiết cho việc thanh toán

**Tạo lớp PaymentController và PaymentDTO, Service** nhằm cung cấp các dịch vụ để theo tác.





**Tạo file Util** nhằm cung cấp các chức năng cho controller và service, em có viết 2 chức năng là băm nhỏ và lấp ip



# TRIỂN KHAI FONT-END REACT JS

## Cấu trúc dự án

 src/components: Chứa các component tái sử dụng cho

giao diện người dùng.

 src/pages: Chứa các trang chính của ứng dụng, kết hợp

nhiều component.

 src/services: Chứa các service để gọi API từ server.

 src/utils: Chứa các hàm tiện ích, hằng số, ...

 public: Chứa các hình ảnh và favicon được hiển thị lên

trên web

 src/App.jsx: File App cấu hình các router và là file chính

để chạy dự án

## ****Thiết kế giao diện người dùng (UI/UX)****

ReactJS hỗ trợ tốt cho việc xây dựng giao diện người dùng với các component tái sử dụng. Một số kỹ thuật thiết kế UI/UX phổ biến:

* Sử dụng thư viện UI như Bootstrap, Material-UI để đơn giản hóa việc thiết kế.
* Áp dụng quy ước đặt tên và cấu trúc component rõ ràng để dễ bảo trì.
* Chia nhỏ các component thành các component con để tăng tính tái sử dụng.
* Sử dụng CSS Module hoặc CSS-in-JS để quản lý styles cho component.
* Áp dụng các nguyên tắc thiết kế UI/UX như responsive design, accessibility, ...

## ****Điều hướng với React Router****

React Router là một thư viện điều hướng phổ biến cho ReactJS, cho phép xây dựng ứng dụng đơn trang (Single Page Application - SPA) với URL định tuyến.

1. **Cài đặt và import React Router**.
2. **Tạo file định nghĩa các routes** sử dụng BrowserRouter, Routes, Route, và Link (hoặc NavLink).
3. **Kết nối routes với các component tương ứng** bằng cách sử dụng Route với path và element.
4. **Điều hướng giữa các trang** bằng cách sử dụng Link (hoặc NavLink) với to prop.
5. **Truy cập tham số URL** trong component với useParams hook.
6. **Điều hướng lập trình** bằng cách sử dụng useNavigate hook hoặc history object từ React Router.

## ****Kết nối với API Back-end****

Để giao tiếp với API Back-end từ phía Front-end, thường sử dụng thư viện gọi HTTP axios.

1. **Tạo service để gọi API**: Định nghĩa các hàm gọi API tương ứng trong các file service.
2. **Gọi API từ component**: Import và sử dụng các hàm service trong component.
3. **Xử lý phản hồi API**: Xử lý trạng thái phản hồi, hiển thị dữ liệu từ API hoặc xử lý lỗi.

Với ReactJS, việc triển khai front-end sẽ trở nên linh hoạt và dễ dàng hơn nhờ vào khả năng tái sử dụng component, và hỗ trợ tốt cho việc xây dựng ứng dụng đơn trang.

# QUY TRÌNH TRIỂN KHAI VÀ PHÁT TRIỂN

## ****Môi trường phát triển và công cụ sử dụng****

Để phát triển dự án, cần chuẩn bị môi trường phát triển bao gồm:

* **Java Development Kit (JDK 22)**: Môi trường lập trình Java.
* **Node.js và NPM**: Môi trường chạy và quản lý các gói JavaScript cho ReactJS.
* **IDE hoặc code editor**: IntelliJ IDEA, Visual Studio Code, ...
* **Cơ sở dữ liệu**: SQL Server.
* **Công cụ quản lý phần mềm**: Maven phía back-end và NPM cho phía front-end.
* **Công cụ đóng gói và triển khai**: Ví dụ Maven, Docker, ...

## ****Quản lý mã nguồn (Git, GitHub, ...)****

Sử dụng hệ thống quản lý mã nguồn phân tán như Git và nền tảng lưu trữ mã nguồn như GitHub, GitLab, ... để:

* Theo dõi và quản lý thay đổi mã nguồn.
* Phân nhánh để phát triển các tính năng mới.
* Hợp nhất mã nguồn từ nhiều nguồn khác nhau.
* Đảm bảo an toàn và dễ dàng khôi phục mã nguồn.

## ****Quy trình làm việc (Agile)****

Áp dụng quy trình phát triển phần mềm linh hoạt như Agile để quản lý dự án hiệu quả:

* Chia nhỏ dự án thành các sprint ngắn (2-4 tuần).
* Xác định các yêu cầu, ưu tiên và ước lượng công việc trong mỗi sprint.
* Tổ chức các cuộc họp hàng ngày để theo dõi tiến độ và giải quyết vấn đề.
* Tổ chức cuộc họp đánh giá và tổng kết sau mỗi sprint.
* Liên tục cải tiến quy trình để nâng cao hiệu quả.

# KẾT LUẬN VÀ ĐỀ XUẤT PHÁT TRIỂN

## ****Tổng kết dự án****

Dự án xây dựng website bán mô hình từ Spring Boot và ReactJS đã đạt được các mục tiêu đề ra ban đầu. Ứng dụng cung cấp một giao diện người dùng hiện đại, thân thiện và tương tác cao, cho phép khách hàng dễ dàng tìm kiếm, xem thông tin chi tiết và đặt mua các mô hình mong muốn. Hệ thống back-end được xây dựng trên nền tảng Spring Boot, đảm bảo hiệu suất, bảo mật và khả năng mở rộng cao. Các tính năng chính như quản lý sản phẩm, đơn hàng, người dùng, thanh toán và xác thực đã được triển khai thành công.

## ****Những thách thức gặp phải và cách giải quyết****

Trong quá trình phát triển, nhóm dự án đã gặp phải một số thách thức sau:

1. **Tích hợp giữa front-end và back-end**: Việc đồng bộ hóa và kết nối giữa ReactJS (front-end) và Spring Boot (back-end) đã gặp phải một số vấn đề liên quan đến định dạng dữ liệu, xử lý lỗi và bảo mật. Nhóm đã giải quyết vấn đề này bằng cách thiết lập các chuẩn giao tiếp giữa hai phần, sử dụng các công cụ như Axios để gửi yêu cầu HTTP, và áp dụng các biện pháp bảo mật theo khuyến nghị của Spring Security.
2. **Quản lý trạng thái ứng dụng phức tạp**: Với ứng dụng có nhiều tính năng và luồng dữ liệu phức tạp, việc quản lý trạng thái ứng dụng trở nên khó khăn. Nhóm đã giải quyết vấn đề này bằng cách sử dụng Redux để quản lý trạng thái tập trung, đồng thời áp dụng các nguyên tắc và mẫu thiết kế phù hợp để duy trì tính nhất quán và dễ bảo trì.
3. **Tối ưu hóa hiệu suất và trải nghiệm người dùng**: Với lượng dữ liệu lớn và yêu cầu về giao diện tương tác cao, việc đảm bảo hiệu suất và trải nghiệm người dùng tốt là một thách thức. Nhóm đã giải quyết vấn đề này bằng cách áp dụng các kỹ thuật tối ưu hóa hiệu suất cho ReactJS như sử dụng React.memo, code-splitting, và lazy loading. Đồng thời, nhóm cũng tối ưu hóa các truy vấn cơ sở dữ liệu và caching dữ liệu phía server để cải thiện hiệu suất ứng dụng.

## ****Đề xuất phát triển tiếp theo****

Dựa trên các phản hồi từ người dùng và xu hướng phát triển của thị trường, nhóm dự án đề xuất một số hướng phát triển tiếp theo cho ứng dụng:

1. **Tích hợp hệ thống đánh giá và nhận xét sản phẩm**: Cho phép người dùng đánh giá, nhận xét và chia sẻ trải nghiệm sử dụng sản phẩm. Điều này sẽ cải thiện trải nghiệm người dùng và tăng niềm tin cho khách hàng mới.
2. **Triển khai tính năng đề xuất sản phẩm**: Sử dụng các thuật toán máy học và phân tích dữ liệu để đề xuất các sản phẩm phù hợp với sở thích và hành vi của người dùng, từ đó tăng doanh thu và tỷ lệ chuyển đổi.
3. **Xây dựng ứng dụng di động**: Phát triển ứng dụng di động trên nền tảng iOS và Android để mang đến trải nghiệm mua sắm thuận tiện hơn cho người dùng di động.
4. **Tích hợp hệ thống thanh toán đa dạng**: Hỗ trợ thêm nhiều phương thức thanh toán khác nhau như thẻ tín dụng, ví điện tử, chuyển khoản ngân hàng, ... để đáp ứng nhu cầu đa dạng của khách hàng.
5. **Mở rộng hệ thống logistics và vận chuyển**: Tích hợp với các đơn vị vận chuyển và logistics để cải thiện quá trình giao hàng, theo dõi đơn hàng và quản lý kho hàng hiệu quả hơn.

# TÀI LIỆU THAM KHẢO

* Spring Boot Documentation - <https://spring.io/projects/spring-boot>
* React Documentation - <https://reactjs.org/docs/getting-started.html>
* Spring Data JPA Reference Documentation - <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>
* Spring Security Reference Documentation - <https://docs.spring.io/spring-security/reference/>
* React Router Documentation - <https://reactrouter.com/web/guides/quick-start>
* Redux Documentation - <https://redux.js.org/introduction/getting-started>
* Axios Documentation - <https://axios-http.com/docs/intro>
* Git Documentation - <https://git-scm.com/doc>
* Docker Documentation - <https://docs.docker.com/>
* AWS Documentation - <https://docs.aws.amazon.com/>
* Jenkins Documentation - <https://www.jenkins.io/doc/>
* Kubernetes Documentation - <https://kubernetes.io/docs/home/>

Các tài liệu khác:

* "Spring Boot in Action" by Craig Walls
* "React Quickly" by Azat Mardan
* "Pro React 16" by Adam Freeman
* "Spring Security in Action" by Laurențiu Spilcă
* "Designing Data-Intensive Applications" by Martin Kleppmann
* "Refactoring UI" by Adam Wathan and Steve Schoger
* "Web Development with Node and Express" by Ethan Brown
* "Clean Code" by Robert C. Martin
* "The Pragmatic Programmer" by Andrew Hunt and David Thomas

Các trang web và cộng đồng:

* Stack Overflow (<https://stackoverflow.com/>)
* Spring Community (<https://spring.io/community>)
* React Community (<https://reactjs.org/community/support.html>)
* GitHub (<https://github.com/>)
* Medium (<https://medium.com/>)
* DEV Community (<https://dev.to/>)