**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

LUẬN VĂN TỐT NGHIỆP

ĐỀ TÀI:

XÂY DỰNG WEBSITE THƯƠNG MẠI ĐIỆN TỬ

CHO NHÃN

THỜI TRANG TARIFA

**Ngành :**

**Niên khóa :**

**Lớp :**

**SVTH :**

TP. HỒ CHÍ MINH, tháng 7 năm 2017

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**



LUẬN VĂN TỐT NGHIỆP

ĐỀ TÀI:

XÂY DỰNG WEBSITE THƯƠNG MẠI ĐIỆN TỬ

CHO NHÃN

THỜI TRANG TARIFA

**Ngành :**

**Niên khóa :**

**Lớp :**

**SVTH :**

TP. HỒ CHÍ MINH, tháng 7 năm 2017

**CÔNG TRÌNH HOÀN TẤT TẠI**

**TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM**

Cán bộ hướng dẫn: Thạc sĩ Mai Anh Thơ

Cán bộ phản biện: Thạc sĩ Trần Lê Như Quỳnh

Luận văn cử nhân được bảo vệ tại HỘI ĐỒNG CHẤM LUẬN VĂN CỬ NHÂN TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM ngày tháng năm

*Nhận xét của giáo viên hướng dẫn*

*Nhận xét của giáo viên phản biện*

**NHIỆM VỤ LUẬN VĂN CỬ NHÂN**

Phái: **Nam**

Nơi sinh: Bình Phước

Ngành: Công nghệ thông tin

Họ tên sinh viên: **TẠ NGỌC BẢO LÂM**

Ngày tháng năm sinh: 03/07/1994

Chuyên ngành: Công nghệ thông tin

**CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc Lập-Tự Do-Hạnh Phúc**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐH NÔNG LÂM TPHCM**

Phái: **Nữ**

Nơi sinh: Vĩnh Long

Ngành: Công nghệ thông tin

Họ tên sinh viên: **VÕ THỊ KIM CƯƠNG**

Ngày tháng năm sinh: 12/02/1995

Chuyên ngành: Công nghệ thông tin

1. TÊN ĐỀ TÀI: **Xây dựng website thương mại điện tử cho nhãn thời trang Tarifa**
2. NHIỆM VỤ VÀ NỘI DUNG

* Nhiệm vụ:
* Nội dung:

1. NGÀY GIAO NHIỆM VỤ: 03/03/2017
2. NGÀY HOÀN THÀNH NHIỆM VỤ: 25/08/2017
3. HỌ VÀ TÊN CÁN BỘ HƯỚNG DẪN: Thạc sĩ Mai Anh Thơ

Ngày / /

CÁN BỘ HƯỚNG DẪN

(Ký và ghi rõ họ tên)

Ngày / /

KHOA CNTT

(Ký và ghi rõ họ tên)

Ngày / /

CÁN BỘ PHẢN BIỆN

(Ký và ghi rõ họ tên)

**LỜI CẢM ƠN**

Trong bốn năm học tập và rèn luyện tại trường đại học Nông Lâm thành phố Hồ Chí Minh, chúng em được sự dạy dỗ chân thành và tận tâm từ thành cô. Chúng em đã tích lũy được nhiều kiến thức bổ ích và các kỹ năng cần thiết cho cuộc sống.

Đầu tiên, chúng em xin bày tỏ lòng biết ơn sâu sắc đến Cô ThS.Mai Anh Thơ. Cô đã tận tình hướng dẫn, truyền đạt kiến thức và kinh nghiệm cho chúng em trong suốt quá trình thực hiện.

Tiếp theo, chúng em xin chân thành cảm ơn quý Thầy Cô Khoa công nghệ thông tin trường đại học Nông Lâm thành phố Hồ Chí Minh đã truyền đạt kiến thức cho chúng em trong bốn năm học tập và tạo điều kiện thuận lợi cho chúng em hoàn thành khóa luận này.

Bên cạnh đó, chúng mình xin gửi lời cảm ơn đến các bạn lớp DH13DT đã chia sẻ và giúp đỡ chúng mình trong suốt quá trình học tập.

Cuối cùng, chúng con xin chân thành cảm ơn cha mẹ đã sinh chúng con, nuôi dưỡng, tạo mọi điều kiện vật chất lẫn tinh thần cho chúng con học tập đến ngày hôm nay.

Trong quá trình thực hiện, mặc dù chúng em đã cố gắng tìm hiểu, học tập rất nhiều nhưng chúng em cũng không tránh khỏi những thiếu sót. Kính mong sự thông cảm và chỉ bảo của Thầy Cô.

Chúng em xin kính chúc cha mẹ, Thầy Cô tràn đầy sức khỏe, hạnh phúc và làm việc tốt.

Chúng em xin chân thành cảm ơn!

Sinh viên thực hiện

Võ Thị Kim Cương

DANH SÁCH CHỮ VIẾT TẮT

HTML5 Hypertext Markup Language 5

CSS3 Cascading Style Sheets 3

JS Java Script

JPA Java Persistence API

JavaEE Java Enterprise Edition

JSB JavaScript Bean

DI Dependency Injection

IoC Inversion of Control

JDBC Java Database Connectivity

DAO Data Access Object

ORM Object-relational mapping

JDO Java Data Objects

SQL Structured Query Language

JSF JavaServer Faces

JSP JavaServer Pages

XML Extensible Markup Language

HTTP Hypertext Transfer Protocol

MVC Model View Controller

REST API Representation State Transfer Application Program Interface

SPA Single Page Application

DOM Document Object Model

AJAX Asynchronous JavaScript and XML

SEO Search Engine Optimization

JWT JSON Web Token

JPQL Java Persistence Query Language

DANH MỤC CÁC HÌNH

[Hình 3.1: Mô hình usecase hệ thống 7](#_Toc492019315)

[Hình 3.2: Lược đồ đăng ký 10](#_Toc492019316)

[Hình 3.3: Lược đồ đăng nhập 12](#_Toc492019317)

[Hình 3.4: Lược đồ đặt hàng 13](#_Toc492019318)

[Hình 3.5: Lược đồ lớp 14](#_Toc492019319)

[Hình 3.6: Cơ sở dữ liệu phần hiển thị trang phục 16](#_Toc492019320)

[Hình 3.7: Luồng thực thi của ứng dụng thông thường 20](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019321)

[Hình 3.8: Servlet Container 21](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019322)

[Hình 3.9: Luồng thực thi của ứng dụng sử dụng IoC 22](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019323)

[Hình 3.10: Module của Spring framework 23](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019324)

[Hình 3.11: Luồng dữ liệu của Spring MVC với DispathcherServlet 28](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019325)

[Hình 3.12: Luồng hoạt động của Spring Security 30](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019326)

[Hình 3.13: Ứng dụng web sử dụng Single Page 32](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019327)

[Hình 3.14: Cơ chế data binding 33](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019328)

[Hình 3.15: Cơ chế one-way binding và two-way binding 35](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019329)

[Hình 3.16: Ứng dụng web application 36](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019330)

[Hình 3.17: Ứng dụng sử dụng Ajax 36](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019331)

[Hình 3.18: Ứng dụng sử dụng SPA 37](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019332)

[Hình 3.19: Mô hình MVC 38](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019333)

[Hình 3.20: Giải thuật lưu trữ hình ảnh 43](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019334)

[Hình 3.21: Mô hình kiến trúc hệ thống 47](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019335)

[Hình 3.22: Tìm kiếm và cài đặt Spring Tools trong eclipse 48](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019336)

[Hình 4.1: Giao diện trang chủ 55](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019337)

[Hình 4.2: Giao diện đăng nhập, đăng ký 56](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019338)

[Hình 4.3: Giao diện hiển thị trang phục đơn 57](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019339)

[Hình 4.4: Giao diện hiển thị trang phục phối 58](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019340)

[Hình 4.5: Giao diện hiển thị thông tin chi tiết trang phục 59](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019341)

[Hình 4.6: Giao diện hiển thị giỏ hàng 60](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019342)

[Hình 4.7: Giao diện hiển thị thông tin trang phục 62](file:///C:\Users\Art_Kiss\Desktop\LVTN_v7.docx#_Toc492019343)

[Hình 4.8: Giao diện hiển thị thông tin chi tiết và thao tác sửa đổi thông tin trang phục 63](#_Toc492019344)

[Hình 4.9: Giao diện xác nhận việc xóa trang phục trước khi xóa 63](#_Toc492019345)

[Hình 4.10: Giao diện hiển thị trang phục đơn 64](#_Toc492019346)

[Hình 4.11: Giao diện hiển thị trang phục phối 64](#_Toc492019347)

[Hình 4.12: Giao diện hiển thị hóa đơn của đơn hàng 65](#_Toc492019348)

DANH MỤC CÁC BẢNG

[Bảng 2.1: Bảng so sách các website thời trang vintage 5](#_Toc492019364)

[Bảng 3.1: Bảng mô tả usecase xem thông tin trang phục 8](#_Toc492019365)

[Bảng 3.2: Bảng mô tả usecase đặt hàng 8](#_Toc492019366)

[Bảng 3.3: Bảng mô tả usecase quản lý giỏ hàng 9](#_Toc492019367)

[Bảng 3.4: Bảng mô tả usecase quản lý admin 10](#_Toc492019368)

[Bảng 3.5: Bảng đặc tả đối tượng thông tin trang phục 14](#_Toc492019369)

[Bảng 3.6: Bảng đặc tả đối tượng trang phục phối 15](#_Toc492019370)

[Bảng 3.7: Bảng đặc tả đối tượng trang phục 15](#_Toc492019371)

[Bảng 3.8: Bảng đặc tả đối tượng trang phục đơn 16](#_Toc492019372)

[Bảng 3.9: Bảng information\_product 17](#_Toc492019373)

[Bảng 3.10: Bảng product 18](#_Toc492019374)

[Bảng 3.11: Bảng mô tả chi tiết bảng singleimage 18](#_Toc492019375)

[Bảng 3.12: Bảng mô tả chi tiết miximage 19](#_Toc492019376)

[Bảng 3.13: Bảng phạm vi của bean 25](#_Toc492019377)

**TÓM TẮT**

Ngày nay, nhu cầu mặc đẹp ngày càng được nâng cao. Vì vậy các website thời trang trở thành công cụ hữu ích giúp người mua có thể bước đầu lựa chọn những phong cách thời trang phù hợp với bản thân. Việc xây dựng một website thương mại điện tử cho một nhãn thời trang phù hợp với phong cách mà thương hiệu hướng đến, đáp ứng nhu cầu của khách hàng không phải là một vấn đề dễ dàng.

Luận văn này sẽ tập trung tìm hiểu và giới thiệu về Spring framework, những đặc tính vượt trội được áp dụng, ưu việt của nó trong việc tạo ra các ứng dụng đòi hỏi sự module hóa và có khả năng tái sử dụng cao. Bên cạnh đó, luận văn còn áp dụng Mapstruct làm đơn giản hóa quá trình chuyển đổi kiểu dữ liệu được áp dụng song song với Spring framework Đồng thời, trình bày thêm công nghệ AngularJS đang rất phổ biến hiện nay, các đặc điểm, tính năng nổi bật và các ưu, nhược điểm của nó. Áp dụng các công nghệ đó để tạo nên một ứng dụng website thời trang.

Sau khi tìm hiểu, học tập chúng tôi sẽ vận dụng kết quả tìm hiểu được vào việc xây dựng ứng dung website thời trang phong cách vintage nhằm minh họa cho phần lý thuyết được trình bày. Ứng dụng được phân chia thành hai phần rõ ràng là phía client và phía server. Trong luận văn này chúng tôi sẽ trình bày chi tiết phần xây dựng ứng dụng phía client và server bằng các công nghệ kể trên.

**ABSTRACT**

Today, the demand for beautiful dress get better and better. But many people don’t enough time for shopping. So, shopping online become useful tool for everyone. Making an ecommerce website with a suitable style is not easy. Therefore, making an ecommerce website for vintage style with fast access speed and secure customer’s information is the priority.

This thesis will focus on understanding and briefing introduction about Spring framework, superior properties, outstanding features in creative application that requires modularizing and capacity of reusing. Beside that, this thesis applies Mapstruct to convert request and response datas. And further description of AngularJS with some special features, advantages and disavantages. They will be applied for this thesis to create a fashion web application.

After studying, we are going to use the result of studying to create a real fashion website with vintage style. Application is designed in two modules: client module and server module. We will represent application details with below technologies.

MỤC LỤC

[DANH SÁCH CHỮ VIẾT TẮT ii](#_Toc492020071)

[DANH MỤC CÁC HÌNH iv](#_Toc492020072)

[DANH MỤC CÁC BẢNG vi](#_Toc492020073)

[Chương 1 Mở đầu 2](#_Toc492020074)

[1.1 Lý do chọn đề tài 2](#_Toc492020075)

[1.2 Mục đích, đối tượng và phạm vi nghiên cứu 2](#_Toc492020076)

[1.2.1 Mục đích 2](#_Toc492020077)

[1.2.2 Đối tượng nghiên cứu 3](#_Toc492020078)

[1.2.3 Phạm vi nghiên cứu 3](#_Toc492020079)

[1.3 Nội dung nghiên cứu 4](#_Toc492020080)

[Chương 2 Tổng quan 5](#_Toc492020081)

[2.1 Khảo sát các website thời trang mang phong cách vintage 5](#_Toc492020082)

[2.2 Kết luận, đề xuất giải pháp cho bài toán 6](#_Toc492020083)

[2.2.1 Kết luận 6](#_Toc492020084)

[2.2.2 Giải pháp 7](#_Toc492020085)

[Chương 3 Nội dung và phương pháp nghiên cứu 8](#_Toc492020086)

[3.1 Phân tích và thiết kế bài toán 8](#_Toc492020087)

[3.1.1 Use case 8](#_Toc492020088)

[3.1.2 Sequence diagram 11](#_Toc492020089)

[3.1.3 Class diagram 15](#_Toc492020090)

[3.1.4 Mô hình cơ sở dữ liệu 17](#_Toc492020091)

[3.2 Tìm hiểu về Spring framework 20](#_Toc492020092)

[3.2.1 Giới thiệu về Spring framework 20](#_Toc492020093)

[3.2.2 Một số khái niệm chính 20](#_Toc492020094)

[3.2.3 Các module của Spring 24](#_Toc492020095)

[3.2.4 Spring core 25](#_Toc492020096)

[3.2.5 Spring MVC 28](#_Toc492020097)

[3.2.6 Spring security 30](#_Toc492020098)

[3.3 Tìm hiểu về AngularJS 32](#_Toc492020099)

[3.3.1 Giới thiệu về AngularJS 32](#_Toc492020100)

[3.3.2 Đặc trưng 35](#_Toc492020101)

[3.3.3 Lợi ích và khó khăn của AngularJS 43](#_Toc492020102)

[3.4 Các giải thuật quan trọng 44](#_Toc492020103)

[3.4.1 Lưu trữ hình ảnh 44](#_Toc492020104)

[3.4.2 Web service 46](#_Toc492020105)

[3.4.3 Bảo mật 46](#_Toc492020106)

[3.4.4 Single page 47](#_Toc492020107)

[3.4.5 Đa ngôn ngữ 48](#_Toc492020108)

[3.5 Hiện thực bài toán 49](#_Toc492020109)

[3.5.1 Kiến trúc hệ thống 49](#_Toc492020110)

[3.5.2 Cài đặt framework 50](#_Toc492020111)

[3.5.3 Cấu hình cho hệ thống 51](#_Toc492020112)

[Chương 4 Kết quả đạt được 56](#_Toc492020113)

[4.1 Hệ thống front-end 56](#_Toc492020114)

[4.2 Hệ thống back-end 63](#_Toc492020115)

[Chương 5 Kết luận và kiến nghị 68](#_Toc492020116)

[5.1 Các kết luận 68](#_Toc492020117)

[5.1.1 Các hạn chế 68](#_Toc492020118)

[5.1.2 Các đề xuất mới 68](#_Toc492020119)

[5.2 Các kiến nghị 68](#_Toc492020120)

[TÀI LIỆU THAM KHẢO 69](#_Toc492020121)

[PHỤ LỤC i](#_Toc492020122)

[Phụ lục 1: Mô tả khác của usecase hệ thống i](#_Toc492020123)

[Phụ lục 2: Mô tả chi tiết class diagram ix](#_Toc492020124)

[Phụ lục 3: Mô tả chi tiết cơ sở dữ liệu xii](#_Toc492020125)

# Mở đầu

## Lý do chọn đề tài

Kinh tế phát triển kéo theo nhu cầu về chất lượng cuộc sống của con người phát triển theo. Và nhu cầu mặc đẹp là một trong số những nhu cầu tất yếu đó. Việc lựa chọn một bộ quần áo hợp phong cách, hợp thời trang rất được người mua quan tâm. Một trong những phong cách thời trang đang được ưa chuộng hiện nay là phong cách vintage. Phong cách thời trang vintage không chỉ được các quý cô độ tuổi trung niên yêu chuộng mà bộ phận giới trẻ hiện nay quan tâm. Phong cách thời trang này mang mang hơi hướng của sự sang trọng, thanh lịch, nhẹ nhàng và cổ điển làm người mặc trở nên dịu dàng hơn.

Nếu như các tín đồ thời trang ngày trước phải đến cửa hàng quần áo để chọn cho mình những bộ trang phục phù hợp thì ngày nay trở nên dễ dàng và tiết kiệm hơn thông qua các website thời trang.

Số lượng website nhiều nên người dùng đặt ra nhiều yêu cầu khắt khe hơn ngoài yêu cầu về chất lượng sản phẩm như tốc độ truy cập website nhanh chóng, hình ảnh website đẹp, bảo mật thông tin khách hàng …

Chính vì những lí do trên nên chúng tôi chọn đề tài “Xây dụng website thương mại điện tử cho nhãn thời trang Tarifa”. Nội dung luận văn này sẽ tập trung tìm hiểu và áp dụng những tính năng nổi bật của cộng nghệ Spring bên phía server. Đồng thời, kết hợp tìm hiểu và ứng dụng AngularJS bên phía client để xây dựng ứng dụng website thời trang thực tế đáp ứng các nhu cầu nói trên.

## Mục đích, đối tượng và phạm vi nghiên cứu

### Mục đích

Mục đích quan trọng của việc thực hiện đề tài này là xây dựng website thương mại điện tử cho nhãn thời trang Tarifa. Bên cạnh đó, thực hiện đề tài nhằm đạt được các mục tiêu chi tiết như sau:

* Tìm hiểu các website thời trang mang phong cách vintage như Vivid linen, Muji, Adored, Flax, Spanishmoss bao gồm bảo mật thông tin khách hàng, cách thức thanh toán và các chính sách chăm sóc khách hàng.
* Xây dựng giao diện người dùng theo phong cách vintage bằng HTML5, CSS3, JavaScript, AngularJS
* Xử lý back-end bằng ngôn ngữ Java, Spring framework
* Sử dụng hibernate, JPA nhằm hỗ trợ tương tác giữa database và hệ thống
* Thực hiện tốt công việc đặt hàng trực tuyến, phối hợp, gợi ý các trang phục thích hợp, chia sẻ thông tin qua mạng xã hội, các chương trình chăm sóc khách hàng, ….

### Đối tượng nghiên cứu

Đối tượng nghiên cứu chủ yếu của đề tài là các website thời trang áp dụng thành công thương mại điện tử như Vivid linen, Muji, Adored, Buykud, Spanishmoss.

Ngoài ra, đối tượng nghiên cứu của đề tài là các công nghệ hỗ trợ xây dựng website như Spring framework (đặc biệt là cơ chết bảo mật tuyệt vời của Spring security) phía server, AngularJS phía client và Hibernate JPA cho việc giao tiếp giữa hệ thống và database.

Bên cạnh đó, đề tài còn hướng đến đối tượng giúp làm đơn giản công việc lập trình như Mapstruct.

### Phạm vi nghiên cứu

Để đáp ứng được các mục đích trên, đề tài đưa ra phạm vi nghiên cứu sau:

* Đề tài tập trung tìm hiểu các tính năng nổi bật của các website trên
* Đề tài tập trung trình bày kết quả sau khi áp dụng Spring framework, AngularJS, JPA, Mapstruct. Mỗi công nghệ sẽ tìm hiểu và giới thiệu các phần cơ bản như giới thiệu, kiến trúc và các đặc tính nổi bật của chúng.
* Đề tài sẽ triển khai việc phân quyền cho người dùng và admin.

Đề tài sẽ vận dụng các kết quả nghiên cứu được để xây dựng website thương mại điện tự về thời trang dựa trên các cơ sở lý thuyết sẽ được tìm hiểu bên dưới.

## Nội dung nghiên cứu

Đề tài sẽ tập trung nghiên cứu các vấn đề sau:

* Đề tài sẽ tập trung tìm hiểu về thương mại điện tử như các phương thức thanh toán, cách thức quảng cáo hay lợi ích thương mại điện tử mang lại. Các nội dung đó sẽ được rút ra từ việc khảo sát các website thương mại điện tử như Vivid linen, Muji, Adored, Buykud, Spanishmoss
* Nghiên cứu công nghệ Spring framework hỗ trợ việc lập trình bằng ngôn ngữ Java. Đề tài sẽ khai thác các tính năng tuyệt vời của Spring framework như xây dựng website theo hướng RESTful Service và cơ chế bảo mật hiệu quả của nó.
* Đề tài sẽ nghiên cứu công nghệ AngularJS để xây dựng website phía client nhằm cải thiện tốc độ tải dữ liệu với cơ chế single-page
* Nghiên cứu JPA giúp việc giao tiếp giữa hệ thống và database trở nên đơn giản hơn.
* Đề tài sẽ đưa ra giải pháp cho việc phối hợp và gợi ý trang phục phù hợp.
* Đề tài nghiên cứu xây dựng giao diện người dùng nhẹ nhàng theo phong cách vintage.

Đề tài sẽ tập trung nghiên cứu những nội dung trên để xây dựng website thương mại điện tử mang phong cách thời trang vintage.

# Tổng quan

## Khảo sát các website thời trang mang phong cách vintage

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Vivid linen** | **Muji** | **Adored** |
| **Giao diện** | * Thể hiện rõ phong cách thời trang vintage. * Hình ảnh có độ phân giải cao giúp người xem dễ dàng quan sát được chất liệu vải. Đặc biệt, tại mỗi vị trí hiển thị, hình ảnh sẽ có độ phân giải khác nhau. * Cách bố trí các thành phần của giao diện hợp lý giúp người xem dễ dàng tìm kiếm trang phục yêu thích. | * Giao diện phù hợp với phong cách thời trang vintage. * Cách bố trí các thành phần của giao diện quá phức tạp làm người xem khó tìm kiếm trang phục. * Đưa ra gợi ý trang phục dựa trên các trang phục đã xem trước đó | * Giao diện đậm chất vintage.   Hình ảnh rõ nét.   * Bố cục các thành phần hợp lý. Menu được thiết kế gọn gàng giúp người xem dễ dàng phân biệt từng loại trang phục. Đặc biệt, website tích hợp chức năng chuyển đổi trạng thái hiển thị từ dạng lưới sang dạng danh sách và ngược lại. |
| **Đặt hàng, thanh toán** | * Có luồng thực hiện thanh toán chi tiết giúp người đặt hàng xác định được thông tin trang phục trước khi quyết định mua hàng * Website có tích hợp cổng thanh toán như: PayPal hoặc thanh toán bằng VISA. | Hiển thị đầy đủ các thông tin cần thiết của trang phục như giá từng trang phục, giá tổng cộng của giỏ hàng, số lượng và hình ảnh ảnh đại diện của trang phục. | Chức năng đặt hàng, thanh toán đơn giản. |
| **Chính sách chăm sóc khách hàng** | * Đưa ra các câu hỏi thường gặp ở khách hàng và giải đáp chúng. * Gợi ý chọn size trang phục cho người mua. * Đưa ra chính sách giao hàng cụ thể. * Gửi mail thông báo các chương trình khuyến mãi, hoặc thông báo các sản phẩm mới. | * Có chính sách giao hàng, vận chuyển chi tiết. * Áp dụng các phương thức thanh toán: VISA, Master Card, JSB, American Express, Diners Club. | * Trả lời các câu hỏi thường gặp của khách hàng. * Có tích hợp phương thức thanh toán bằng cổng thanh toán điện tử PayPal. |
| **Tốc độ truy xuất của website** | Tốc độ tải hình ảnh chậm do có quá nhiều hình ảnh với kích thước lớn. | Tốc độ tải trang tương đối nhanh. | Tốc độ tải trang nhanh. |

Bảng .: Bảng so sách các website thời trang vintage

## Kết luận, đề xuất giải pháp cho bài toán

### Kết luận

Sau khi phân tích và đưa ra các nhận xét về các website, chúng tôi thấy rằng các yếu tố về giao diện, chức năng đặt hàng thanh toán và các chính sách chăm sóc khách hàng là yếu tố quan trọng ngoại trừ chất lượng sản phẩm của một website thời trang. Vì vậy xây dựng giao diện là yếu tố cần được quan tâm đầu tiên.

Ngoài ra các chức năng của hệ thống phải hoạt động thật tốt đặc biệt là chức năng đặt hàng, thanh toán để tiết kiệm thời gian và làm đơn giản hóa quá trình mua hàng của khách hàng.

Để giữ chân khách hàng thì hệ thống cần đưa ra các chính sách chăm sóc khách hàng hợp lý như các chương trình khuyến mãi vào các dịp đặc biệt hay quy định rõ ràng cho chính sách giao hàng, đổi trả sản phẩm.

Cải thiện tốc độ tải trang, tải hình ảnh đại diện cho trang phục cũng là một yếu tố quan trọng của một website để tiết kiệm tối đa thời gian của người mua hàng.

### Giải pháp

Nắm được các ưu điểm và khuyết điểm của các website trên để xây dựng nên một website thương mại điện tử mang phong cách vintage với giao diện nhẹ nhàng, chức năng hệ thống hoạt động ổn định, có chính sách chăm sóc khách hàng và tốc độ tải trang, tải hình ảnh trang phục nhanh thì hệ thống cần:

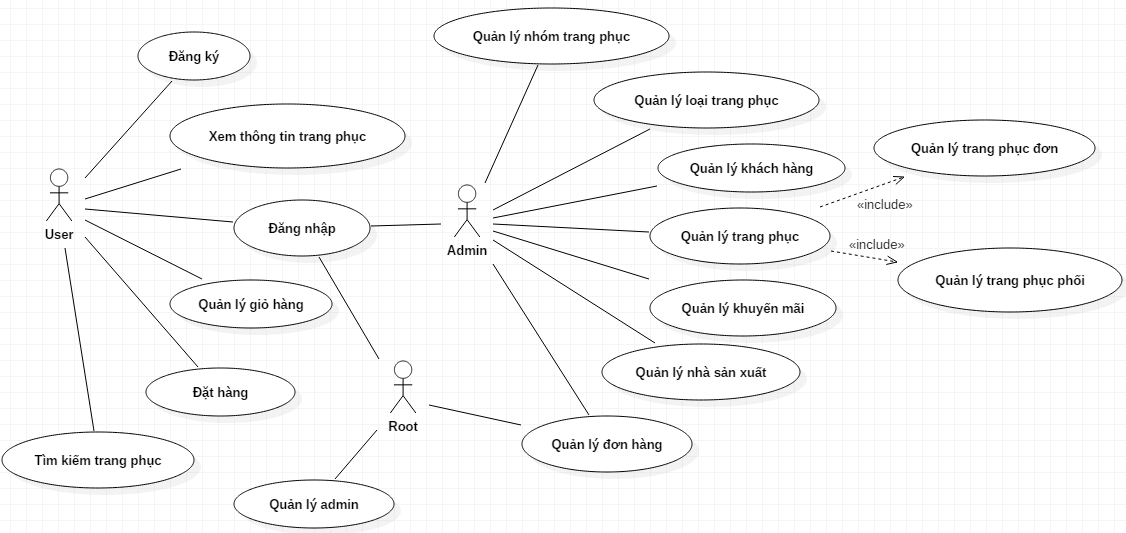
* Sử dụng công nghệ Spring framework với cơ chế bảo mật hiêu quả giúp người dùng bảo mật tài khoản của họ. Ngoài ra sử dụng Spring framework để cải thiện tốc độ lấy và trả dữ liệu của website.
* Sử dựng công nghệ AngularJS để xây dựng website phía người dùng. Dữ liệu được tải lại với tốc độ nhanh chóng, rút ngắn thời gian chờ đợi việc tải trang của khách hàng.
* Ngoài ra website sẽ sử dụng các công nghệ khác như JPA, Mapstruct và công cụ lập trình spring tool suit giúp làm đơn giản hóa công việc lập trình.
* Website sẽ xây dựng các giải thuật giúp giải quyết bài toán về phối đồ cho các sản phẩm phối.

# Nội dung và phương pháp nghiên cứu

## Phân tích và thiết kế bài toán

### Use case

#### Mô hình



Hình .: Mô hình usecase hệ thống

#### Đặc tả một số usecase chính

##### Xem thông tin trang phục

|  |  |
| --- | --- |
| Usecase | Xem thông tin trang phục |
| Actor | Khách hàng, admin, root |
| Description | Usecase cho phép người xem xem thông tin trang phục bao gồm trang phục đơn và các trang phục phối. Người xem còn sem thông tin chi tiết về giá, kích thước, màu sắc ở cả trang phục đơn và trang phục phối. |
| Main flows | 1. Người dùng truy cập vào website ở trang chủ. 2. Người xem chọn xem trang phục theo loại (ở mục category) hoặc xem trang phục theo nhóm (ở menu ngang). 3. Người xem có thể xem màu sắc trang phục. 4. Người xem chọn vào từng trang phục để xem thông tin chi tiết trang phục bao gồm thông tin về giá, màu sắc, hình ảnh. |
| Alternative flows |  |
| Pre-condition |  |
| Post-condition | Người xem biết được các thông tin liên quan tới trang phục. |

Bảng .: Bảng mô tả usecase xem thông tin trang phục

##### Đặt hàng

|  |  |
| --- | --- |
| Usecase | Đặt hàng |
| Actor | Khách hàng |
| Description | Usecase này cho phép khách hàng đặt hàng trang phục |
| Main flows | 1. Chọn trang phục yêu thích. 2. Chọn các yêu cầu bắt buộc về màu sắc, size, số lượng. 3. Chọn chức năng đặt hàng. |
| Alternative flows | 1. Hệ thống yêu cầu nhập các thông tin bắt buộc về màu sắc, size, số lượng đặt hàng. 2. Nếu khách hàng chưa có tài khoản thì hệ thống yêu cầu đăng ký tài khoản mới trước khi đặt hàng. 3. Khách hàng đã có tài khoản nhưng chưa đăng nhập thì hệ thống sẽ yêu cầu đăng nhập trước khi đặt hàng. 4. Hệ thống xác nhận địa chỉ giao hàng theo địa chỉ tài khoản hoặc theo địa chỉ giao hàng mới. |
| Pre-condition | Đã đăng nhập vào hệ thống bằng tài khoản đã đăng ký trước đó. |
| Post-condition | Khách hàng sẽ đặt hàng được trang phục yêu thích. |

Bảng .: Bảng mô tả usecase đặt hàng

##### Quản lý giỏ hàng

|  |  |
| --- | --- |
| Usecase | Quản lý giỏ hàng |
| Actor | Khách hàng |
| Description | Usecase cho phép khách hàng quản lý giỏ hàng của mình với các thao tác: thêm hoặc giảm số lượng trang phục, xóa trang phục ra khỏi giỏ hàng, thêm trang phục mới vào giỏ hàng. |
| Main flows | 1. Vào trang hiển thị trang phục bất kỳ. 2. Chọn hình ảnh đại diện của trang phục để vào trang chi tiết. 3. Chọn các thông số bắt buộc như: số lượng, màu sắc, và kích thước muốn đặt mua. 4. Chọn chức năng thêm vào giỏ hàng. 5. Chọn icon giỏ hàng để xem tất cả trang phục trong giỏ hàng. |
| Alternative flows | Nếu người đặt hàng không chọn các thông số về số lượng, màu sắc và kích thước của trang phục hệ thống sẽ hiển thị thông báo yêu cầu hoàn thành các thông số này. |
| Pre-condition | Người dùng phải có tài khoản và đã đăng nhập vào hệ thống. |
| Post-condition | Người đặt hàng có thể đặt mua trang phục trực tiếp. |

Bảng .: Bảng mô tả usecase quản lý giỏ hàng

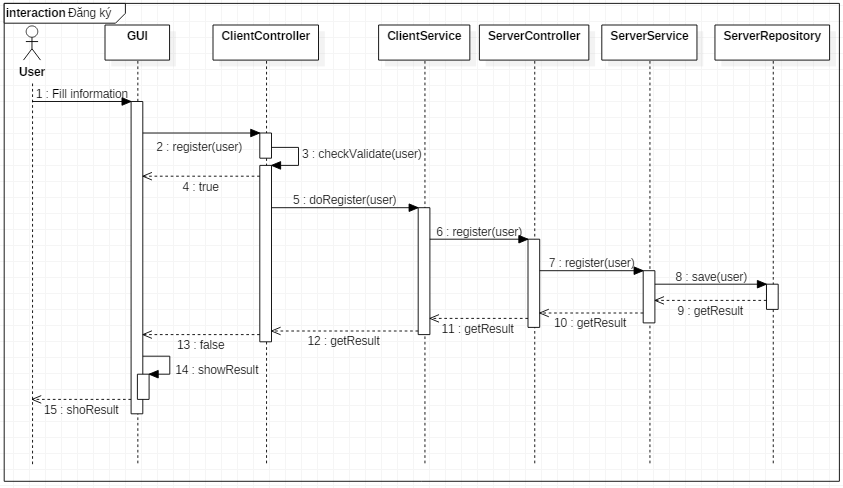
##### Quản lý admin

|  |  |
| --- | --- |
| Usecase | Quản lý admin |
| Actor | Root |
| Description | Usecase cho phép root quản lý admin khác thông qua việc phân quyền cho admin |
| Main flows | 1. Người quản trị là root đăng nhập vào hệ thống bằng tài khoản root. 2. Người quản trị thực hiện chức năng quản lý admin thông qua việc phân quyền cho admin. 3. Người quản trị là root xác nhận lại thông tin thay đổi trước khi quá phân quyền admin được thực hiện. 4. Hoàn tất quá trình phân quyền admin. |
| Alternative flows | Nếu việc phân quyền thực hiện không thành công thì hệ thống sẽ hiện thông báo cho người quản trị là root. |
| Pre-condition | Người quản trị phải đăng nhập vào hệ thống quản lý với tài khoản là root trước khi thực hiện chức năng phân quyền cho admin. |
| Post-condition | Thực hiện thành công việc phân quyền cho admin. |

Bảng .: Bảng mô tả usecase quản lý admin

### Sequence diagram

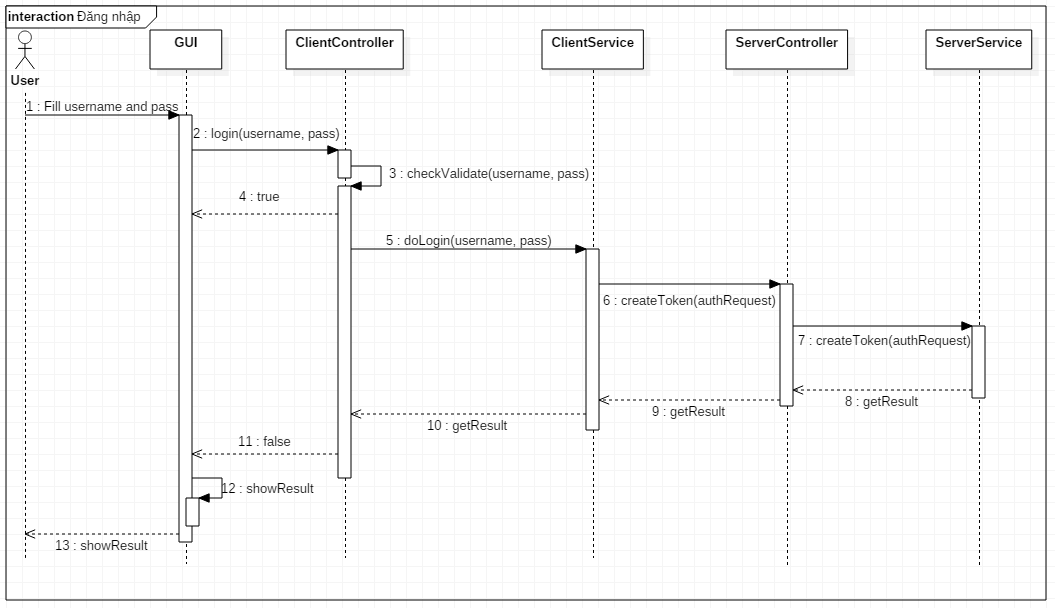
#### Đăng ký



Hình .: Lược đồ đăng ký

* Mô tả lược đồ đăng ký
* Người dùng (user) điền đầy đủ thông tin và gửi yêu cầu đăng ký thông qua giao diện đăng ký.
* Từ giao diện sẽ gửi yêu cầu đăng ký bằng phương thức register với thông tin do người dùng nhập vào tới ClientController.
* Tại vị trí của ClientController, hệ thống sẽ kiểm tra các giá trị nhập vào của người dùng đã hợp lệ chưa.
* Nếu giá trị nhập vào của người dùng hợp lệ, ClientController này sẽ gửi yêu cầu đăng ký bằng hàm doRegister với các giá trị của người dùng nhập vào tới ClientService.
* ClientService sẽ gửi yêu cầu đăng ký này tới ServerController bằng phương thức register.
* ServerController sẽ gọi hàm register tới Service.
* ServerService sẽ gọi hàm save ở ServerRepository để lưu các thông tin của người dùng xuống cơ sở dữ liệu.
* Nếu tại vị trí ClientController, việc kiểm tra giá trị nhập vào của người dùng chưa hợp lệ sẽ trả về thông báo “Giá trị nhập vào chưa hợp lệ”.

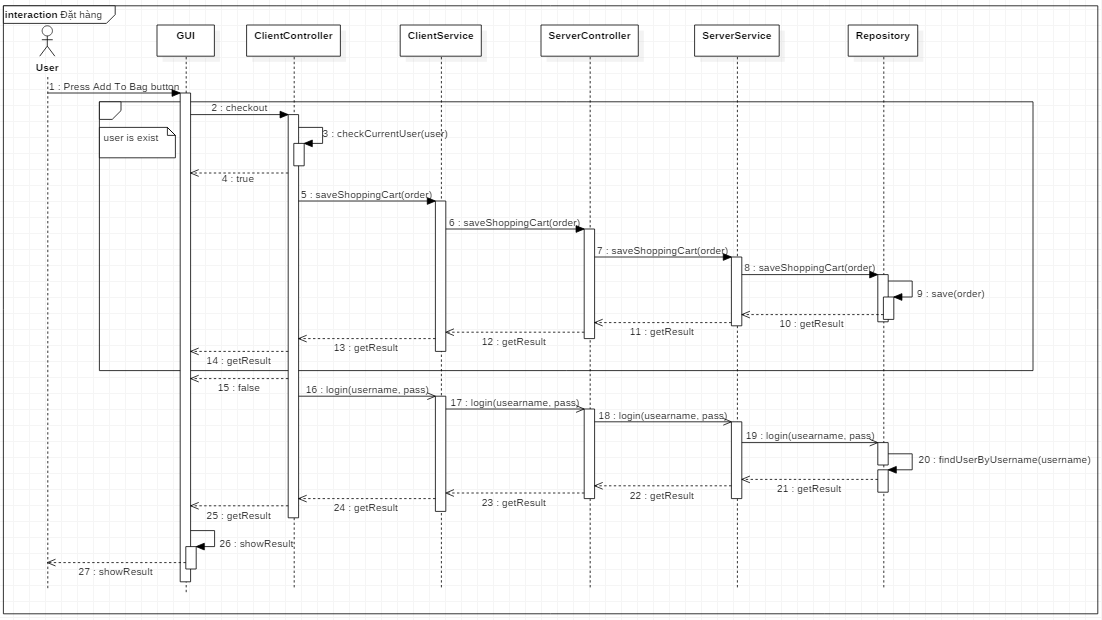
#### Đăng nhập



Hình .: Lược đồ đăng nhập

* Mô tả lược đồ đăng nhập
* Người dùng điền đầy đủ thông tin bao gồm username và password trên giao diện đăng nhập (GUI).
* Giao diện đăng nhập sẽ gọi hàm login với username và password nhập vào ở ClientController.
* Tại ClientContrller, hệ thống sẽ kiểm tra giá trị username và password đã hợp lệ chưa bằng phương thức checkValidate
* Nếu giá trị username và passwork là hợp lệ thì gọi hàm doLogin ở ClientService.
* ClientService sẽ gọi hàm createToken với giá trị nhập vào như ban đầu tới ServerController.
* ServerController sẽ gọi hàm createToken ở ServerService.
* Nếu giá trị nhập vào không hợp lệ, hệ thống sẽ thông báo cho người dùng ở ClientController và hiển thị thông báo tại giao diện đăng nhập.

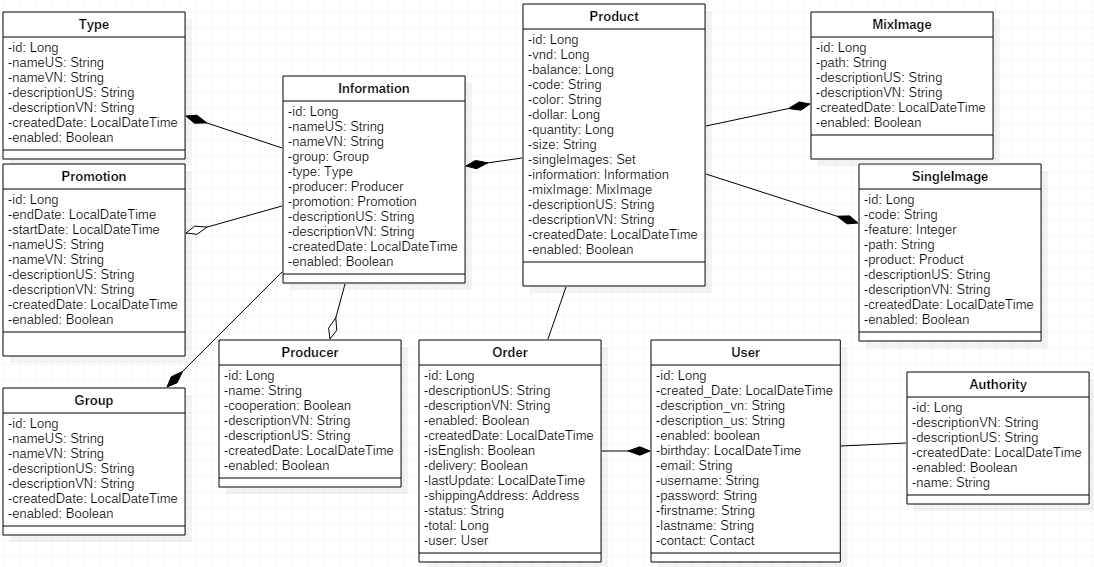
#### Đặt hàng



Hình .: Lược đồ đặt hàng

* Mô tả lược đồ đặt hàng
* Người dùng nhấn nút đặt hàng ở giao diện đặt hàng (GUI).
* Giao diện gửi yêu cầu đặt hàng tới ClientController.
* ClientController sẽ kiểm tra người dùng hiện tại đã đăng nhập chưa.
* Nếu người dùng đã đăng nhập vào hệ thống, ClientController sẽ gọi hàm saveShoppingCart với thông tin trang phục ở ClientService.
* ClientService sẽ gọi hàm saveShoppingCart ở ServerController.
* ServerController sẽ tiếp tục gọi hàm saveShoppingCart ở ServerService.
* ServerService sẽ gọi hàm saveShopping ở ServerRepository.
* ServerRepository sẽ lưu đơn đặt hàng xuống cơ sở sở dữ liệu.
* Tại ClientController, hàm checkCurrentUser trả về giá trị rỗng thì hệ thống sẽ yêu cầu người đặt hàng đăng nhập vào hệ thống.
* Người dùng sẽ thực hiện việc đăng ký giống như lược đồ đăng nhập được miêu tả ở trên.

### Class diagram



Hình .: Lược đồ lớp

#### Information: Thông tin trang phục

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Định danh |
| nameUS | String | Tên trang phục bằng tiếng Anh |
| nameVN | String | Tên trang phục bằng tiếng Việt |
| group | Group | Nhóm trang phục |
| type | Type | Loại trang phục |
| promotion | Promotion | Khuyến mãi cho trang phục |
| producer | Producer | Nhà sản xuất của trang phục |
| descriptionUS | String | Mô tả thông tin trang phục bằng tiếng Anh |
| descriptionVN | String | Mô tả thông tin trang phục bằng tiếng Việt |
| createdDate | LocalDateTime | Ngày tạo thông tin trang phục |
| enabled | Boolean | True là hiển thị False là không hiển thị |

Bảng .: Bảng đặc tả đối tượng thông tin trang phục

#### MixImage: Trang phục phối

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Định danh |
| path | String | Đường dẫn hình ảnh cho trang phục phối |
| descriptionUS | String | Mô tả trang phục phối bằng tiếng Anh |
| descriptionVN | String | Mô tả trang phục phối bằng tiếng Việt |
| createdDate | LocalDateTime | Ngày tạo trang phục phối |
| enabled | Boolean | True là hiển thị False là không hiển thị |

Bảng .: Bảng đặc tả đối tượng trang phục phối

#### Product: Trang phục

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Định danh |
| vnd | Long | Giá tiền trang phục theo đơn vị VNĐ |
| balance | Long | Giá trang phục nhập vào |
| code | String | Mã trang phục |
| color | String | Màu sắc trang phục |
| dollar | Long | Giá trang phục theo đơn vị Dollar |
| quantity | Long | Số lượng trang phục |
| size | String | Size trang phục |
| singleImages | Set<SingleImage> | Danh sách trang phục đơn |
| information | Information | Thông tin trang phục |
| mixImage | MixImage | Trang phục phối |
| descriptionUS | String | Mô tả trang phục bằng tiếng Anh |
| descriptionVN | String | Mô tả trang phục bằng tiếng Việt |
| createdDate | LocalDateTime | Ngày tạo trang phục |
| enabled | Boolean | True là hiển thị, False là không hiển thị |

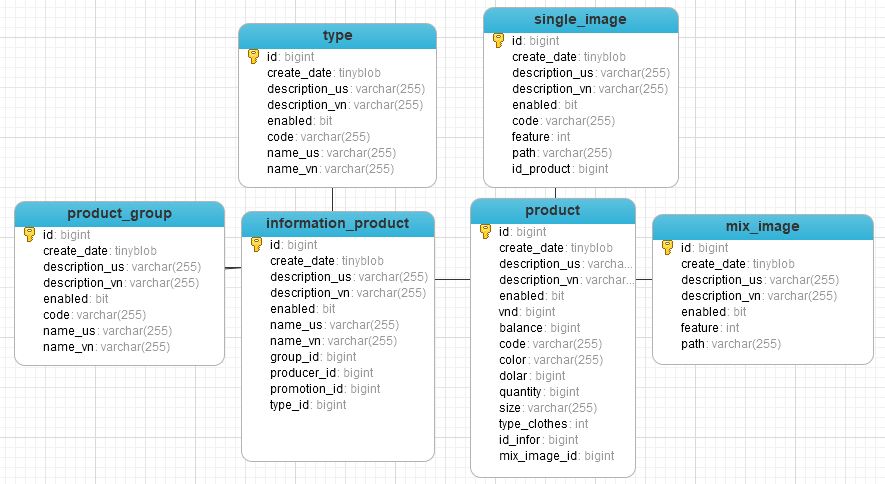
Bảng .: Bảng đặc tả đối tượng trang phục

#### SingleImage: Trang phục đơn

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Định danh |
| code | String | Mã cho nhóm trang phục đơn |
| feature | Integer | Vị trí hiển thị của trang phục đơn |
| path | String | Đường dẫn hình ảnh cho trang phục đơn |
| product | Product | Trang phục |
| descriptionUS | String | Mô tả trang phục đơn bằng tiếng Anh |
| descriptionVN | String | Mô tả trang phục đơn bằng tiếng Việt |
| createdDate | LocalDateTime | Ngày tạo trang phục đơn |
| enabled | Boolean | True là hiển thị False là không hiển thị |

Bảng .: Bảng đặc tả đối tượng trang phục đơn

### Mô hình cơ sở dữ liệu



Hình .: Cơ sở dữ liệu phần hiển thị trang phục

#### Bảng information\_product

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| name\_us | varchar | Tên sản phẩm bằng tiếng Anh |
| name\_vn | varchar | Tên sản phẩm bằng tiếng Việt |
| group\_id | bigint | Định danh của group |
| producer\_id | bigint | Định danh của producer |
| promotion\_id | bigint | Định danh của promotion |
| type\_id | bigint | Định danh của type |

Bảng .: Bảng information\_product

#### Bảng product

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| vnd | bigint | Giá sản phẩm theo đồng Việt |
| balance | bigint | Giá nhập vào của sản phẩm |
| code | varchar | Mã code cho sản phẩm hiển thị ở trang index |
| color | varchar | Màu sắc sản phẩm |
| dollar | bigint | Giá sản phẩm theo đồng Mỹ |
| quantity | bigint | Số lượng sản phẩm |
| size | varchar | Size |
| type\_clothes | int | Loại sản phẩm |
| id\_infor | bigint | Định danh information\_product |
| mix\_image\_id | bigint | Định danh mix\_image |

Bảng .: Bảng product

#### Bảng singleimage

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| feature | int | Vị trí hiển thị hình ảnh |
| code | Varchar | Mã nhóm hình ảnh trang phục |
| path | varchar | Đường dẫn tới hình ảnh |

Bảng .: Bảng mô tả chi tiết bảng singleimage

#### Bảng miximage

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| feature | int | Vị trí hiển thị hình ảnh |
| path | varchar | Đường dẫn tới hình ảnh |
| id\_product | bigint | Định danh product |

Bảng .: Bảng mô tả chi tiết miximage

## Tìm hiểu về Spring framework

### Giới thiệu về Spring framework

Spring là một framework phổ biến dành cho các ứng dụng Java. Nó là một cấu trúc dùng để xây dựng các ứng dụng mã nguồn mở. Phiên bản đầu tiên được ông Rod Johnson viết và đã được ông xuất bản cùng với quyển sách Expert One-on-One JavaEE Design and Development vào tháng 10 năm 2002.

Kiến trúc của Spring framework được ra mắt lần đầu tiên vào tháng 6 năm 2003 dưới giấy phép Apache – phiên bản 2.0. Phiên bản 1.0 đánh dấu sự thành công đầu tiên được phát hành vào tháng 3 năm 2004 và tiếp đó vào tháng 9 năm 2004, tháng 3 năm 2005.

Các tính năng của Spring đặc biệt được sử dụng để xây dựng các ứng dụng web trên nền tảng JavaEE. Spring framework ra đời thay thế cho mô hình Enterprise Java Bean. Spring cung cấp đầy đủ tài liệu hướng dẫn bao gồm lý thuyết và những bài thực hành hỗ trợ cho công việc lập trình.

Spring framework đã làm cho những kỹ thuật chưa được ưa chuộng thành kỹ thuật phổ biến như hiện nay. Có thể kể đến kỹ thuật “Đảo ngược quyền phụ thuộc” (Inversion of Controller hay còn gọi là IoC) hay kỹ thuật “Tiêm sự phụ thuộc” (Dependency Injection hay còn gọi là DI).

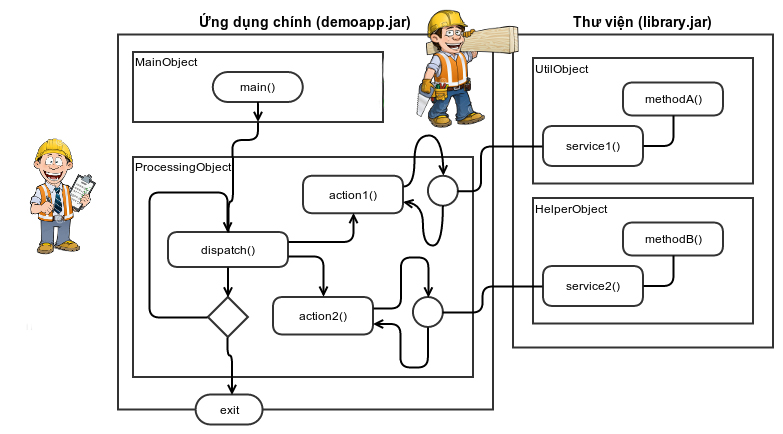
### Một số khái niệm chính

IoC và DI là hai khái niệm quan trọng và nền tảng tạo nên sức mạnh cho Sring framework.

#### Inversion of Control (IoC)

IoC trong Spring được xây dựng theo nguyên tắc đảo ngược sự phụ thuộc (Inversion of Control). Nguyên tắc này được sử dụng trong mẫu thiết kế và được phổ biến nhờ ông Robert C.Martin và Martin Fowler. Đầu tiên cần tìm hiểu về Control (điều khiên) trong một chương trình và Inversion (sự đảo ngược).

Đầu tiên, Control Flow (luồng thực thi) được sử dụng cho trình tự thực hiện các câu lệnh, chỉ thị hoặc lời gọi hàm trong một chương trình khi chương trình này thực thi.

Do các ứng dụng ngày càng phức tạp nên phương pháp lập trình hướng đối tượng được áp dụng phổ biến hơn nhằm phân loại, tách biệt các thành phần, chức năng và gom thành đối tượng cụ thể. Bên cạnh đó, các nhà phát triển còn dựng sẵn các thư viện để có thể sử dụng lại. Các ứng dụng cần gọi phương thức của dối tượng trong ứng dụng và sử dụng các thư viện được xây dựng sẵn. Dưới đây là hình ảnh minh họa cho luồng thực thi của một ứng dụng.

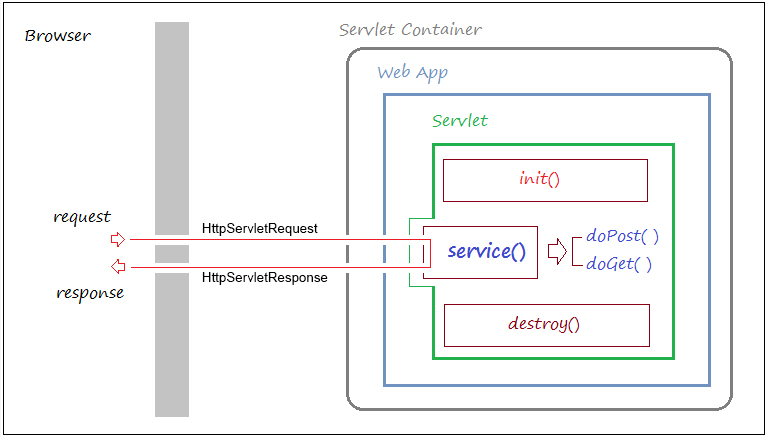
Hình .: Luồng thực thi của ứng dụng thông thường

Một ứng dụng phải thực thi hai nhiệm vụ sau: trực tiếp điều khiển luồng thực thi của chương trình và xây dựng các chức năng để đáp ứng nghiệp vụ của ứng dụng. Nhìn chung, luồng dữ liệu của một chương trình tương đối giống nhau ví dụ như các ứng dụng web sử dụng giao thức HTTP để truyền dữ liệu qua mạng hay phần unit testing trong các ứng dụng, …. Việc tạo dựng và kiếm soát luồng thực thi của chương trình lặp đi lặp lại khi xây dựng ứng dụng sẽ làm mất nhiều thời gian, chi phí. Chính vì yếu điểm này, nguyên lý đảo ngược sự điều khiển được phát triển.

Để hiểu về khái niệm “inversion (đảo ngược)” chúng ta xét ví dụ bên dưới về ứng dụng web trong Java với Java Servlet.

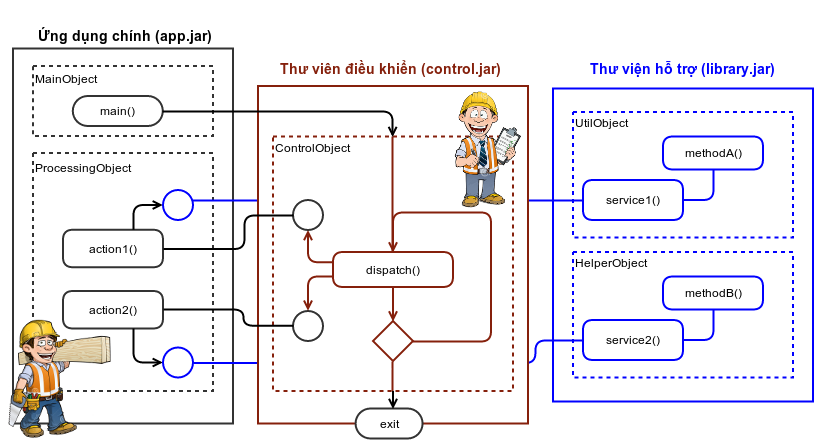
Servlet là công nghệ web đầu tiên đối với người lập trình Java. Các bước lập trình với Java Servlet như sau:

* Tạo lớp đối tượng kế thừa từ HttpServlet, ghi đè (override) phương thức doGet(), doPost(), …
* Đăng ký trong file cấu hình Deployment Descriptor trong ứng dụng Servlet này với đường dẫn xác định.
* Lớp Servlet vừa tạo sẽ được gọi đến khi có một truy vấn HTTP có đường dẫn khớp với đường dẫn khai báo trong Deployment Descriptor.



Hình .: Servlet Container

Servlet Container sẽ kích hoạt ứng dụng mỗi khi có Http Request đến và chịu trách nhiệm chuyển đổi thông điệp HTTP Request, HTTP Response thành các đối tượng HttpServletRequest, HttpServletResponse để truyền cho các hàm doGet(), doPost().

Với ví dụ trên, ứng dụng không còn thực hiện hai công việc cùng lúc là trực tiếp tạo dựng và kiếm soát luồng thực thi, vừa xây dựng chức năng nghiệp vụ. Việc kiểm soát luồng thực thi được tách khỏi chức năng nghiêp vụ và bị đẩy ra bên ngoài. Người lập trình đã ủy thác việc kiểm soát luồng thực thi ứng dụng cho một thành phần (thường là thư viện dựng sẵn) bên ngoài đảm nhiệm, chỉ cần tập trung vào chức năng chính của ứng dụng.

Hình .: Luồng thực thi của ứng dụng sử dụng IoC

Inversion (đảo ngược) là chuyển nhiệm vụ kiểm soát luồng thực thi từ ứng dụng cho một thành phần chuyên trách. Như vậy ứng dụng có thể tập trung thực hiện các chức năng chính của nó. Thư viện điều khiển sẽ dựa trên mô tả trong cấu hình của ứng dụng để điều phối luồng thực thi của ứng dụng. Có thể thấy, thư viện phần mềm khung của IoC tạo dựng luồng thực thi và gọi đến các chức năng của ứng dụng.

Tham khảo tại trang: <http://acegik.net/blog/java/spring/core/tong-quan-ve-ioc-trong-spring.html>

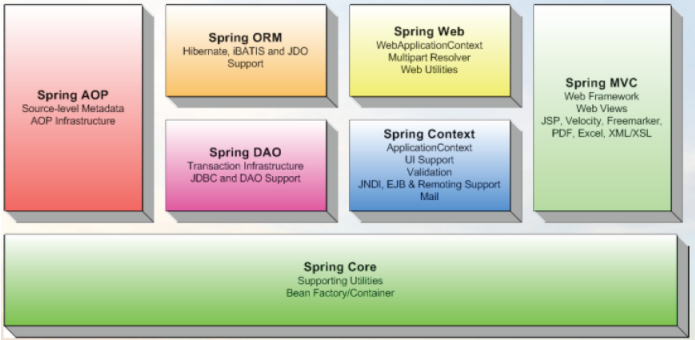
#### Dependency Injection (DI)

Dependency Injection viết tắt là DI là khả năng liên kết giữa các thành phần lại với nhau trong Spring, đó chính là việc các thuộc tính trong một đối tượng được “inject (tiêm chích)” để tham chiếu lần lượt đến các đối tượng khác được quản lý bởi Spring Container.

Với Dependency Injection thì một đối tượng sẽ không phụ thuộc vào đối tượng khác và đối tượng khác cũng vậy. Khi cần sử dụng, đối tượng này sẽ gọi đổi tượng kia và ngược lại. Khung chứa để chứa các đối tượng được tạo ra và khung chứa đó là IoC (đã được giải thích phía trên).

### Các module của Spring

Spring được xây dựng với 7 module chính. Xem hình minh họa bên dưới để hiểu rõ các module của Spring.



Hình .: Module của Spring framework

Trong đề tài này, chúng tôi sử dụng một số module sau:

* **Spring Core**: cung cấp các thành phần cơ bản nhất cho các ứng dụng Spring. Đặc biệt nó cung cấp hai đặc tính cơ bản là IoC (Inversion of Control) và DI (Dependency Injection). Thành phần chính của nó là BeanFactory, một cài đặt của Factory pattern. BeanFactory áp dụng IoC pattern để đặc tả sự phụ thuộc từ code của ứng dụng.
* **Spring DAO**: tầng JDBC và DAO đưa ra một cây phân cấp exception để quản lý kết nối tới database. Tầng exception điều khiển lỗi và giảm khối lượng code. Module này cung cấp các dịch vụ quản lý giao dịch cho các đối tượng trong ứng dụng Spring.
* **Spring ORM**: Spring có thể tích hợp với một vài ORM framework để cung cấp Object Relational tool bao gồm: JDO, Hibernate, OJB và iBatis SQL Maps.
* **Spring Web**: Nằm trên application context module, cung cấp context cho các ứng dụng web. Spring cũng tích hợp với Struts, JSF và Webwork. Spring web cũng làm giảm bớt việc điều khiển nhiều request và gắn các tham số của request vào đối tượng domain.
* **Spring MVC:** được cài đặt với dầu đủ đặc tính của MVC pattern để xây dựng các ứng dụng web. Spring MVC được cấu hình thông qua giao diện và chứa được một số kỹ thuật view bao goofmL JSP, Velocity, Velocity, Tiles và generation of PDF và Excel file.

Đó là các module chính được áp dụng để xây dựng website thời trang Tarifa.

### Spring core

#### Bean

##### Định nghĩa Bean

Các đối tượng trong Spring được quản lý bởi IoC container và được gọi là các bean. Những bean này được tạo ra bởi configuration metadata được cung cấp từ container.

Spring container sẽ quản lý một hoặc nhiều bean. Các bean được tạo ra nhờ vào thông tin cấu hình được chỉ ra trong thẻ <bean/> khi sử dụng XML config. Có ba cách để cung cấp configuration metadata cho Spring container: thông qua XML configuration file, thông qua Annotation-based configuration và thông qua Java-based configuration.

##### Phạm vi của Bean

Phạm vi của bean sẽ quy định việc khởi tạo một hay nhiều thể hiện, khi nào một thể hiện được tạo ra. Spring cung cấp 5 phạm vi của bean.

|  |  |
| --- | --- |
| **Phạm vi** | **Mô tả** |
| Singleton | Chỉ một thể hiện duy nhất được tạo ra cho bean trong mỗi Spring IoC container. Đây là phạm vi mặc định cho mỗi bean nếu không khai báo phạm vi. |
| Prototype | Cho phép nhiều hơn một đối tượng được tạo ra |
| Request | Một thể hiện cụ thể sẽ được tạo ra cho mỗi HTTP request. Phạm vi này chỉ có trong container của các ứng dụng web (WebApplicationContext) |
| Session | Một thể hiện cụ thể sẽ được tạo ra cho mỗi HTTP session. Phạm vi này chỉ có trong container của các ứng dụng web (WebApplicationContext) |
| Global-session | Một thể hiện cụ thể sẽ được tạo ra cho một HTTP session. Phạm vi này chỉ có trong container của các ứng dụng web |

Bảng .: Bảng phạm vi của bean

#### Spring IoC container

BeanFactory là một interface cung cấp các tính năng cơ bản cho DI trong Spring nhưng ApplicationContext có thêm những tính năng nâng cao. Vì thế, ApplicationContext được sử dụng nhiều hơn.

ApplicationContext là một interface được kế thừa từ interface BeanFactory. ApplicationConext có nhiều thể hiện nhưng cần chú ý đến các thể hiện: ClassPathXmlApplicationContext, FileSystemXmlApplicationContext, XmlWebApplicationContext và AnnotationConfigApplicationContext.

#### Dependency Injection

Có 2 cách để tiêm một đối tượng vào một đối tượng khác là dùng constructor và setter. Chúng ta tìm hiểu chi tiết của 2 cách này.

Đầu tiên, setter injection là phương pháp tiêm một đối tượng bằng phương thức setter. Thông thường, nó không cần thiết, một phương thức setter sẽ đi với phương thức getter. Trong trường hợp này nó được thiết lập cho việc ghi và đọc.

Thứ hai là constructor injection. Cách thức tiêm đối tượng này phụ thuộc vào tham số truyền vào là đối tượng.

Chúng ta đã biết cách khai báo để đưa một đối tượng vào một đối tượng. Ngoài ra, Spring còn hỗ trợ đưa một đối tượng và một đối tượng khác bằng cách sử dụng Bean Autowiring. Có 3 cách sử dụng Bean Autowiring là byName, byType và constructor. Chúng ta sẽ tìm hiểu chi tiết về chúng.

* byName: Mỗi bean trong khung chứa đều có 1 id. Dựa vào điều này Spring quy định nếu id của đối tượng phụ thuộc giống với tên một biến nào đó trong đối tượng bị phụ thuộc thì chúng ta sẽ khai báo thêm thuộc tính **autowire = “byName”**. Như vậy Spring sẽ tự động đưa đối tượng đối tượng phụ thuộc vào đối tượng bị phụ thuộc thông qua phương thức setter của đối tượng bị phụ thuộc.
* byType: Đối với cách này chỉ áp dụng cho mỗi đối tượng chỉ có một bean duy nhất trong khung chứa của Spring. Chúng ta sẽ dùng thuộc tính **autowire = “byType”** khi khai báo một bean. Và Spring sẽ tự động đưa đối tượng phụ thuộc vào đối tượng bị phụ thuộc thông qua phương thức setter của đối tượng bị phụ thuộc.
* Constructor: Tương tự như đối với thuộc tính **autowire = “byType”**, cách này áp dụng cho mỗi đối tượng chỉ có một bean duy nhất trong khung chứa của Spring.

### Spring MVC

#### Giới thiệu Spring MVC

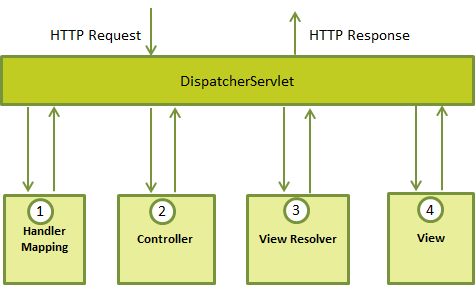
Spring MVC là một module trong Spring framework, cung cấp mô hình kiến trúc MVC và một vài đặc tính khác như chuyển đối kiểu dữ liệu, validation, …

Đặc điểm nổi bật nhất của Spring MVC là tách các thành phần của ứng theo 3 phần: Model, View và Controller. Model dùng để đóng gói dữ liệu ứng dụng. View sẽ chịu trách nhiệm nhận giá trị của model và chỉ trang HTML hiển thị. Cuối cùng, controller chịu trách nhiệm nhận và xử lý các yêu cầu từ người dùng và tạo các model phù hợp, sau đó trả về cho view.

Hoạt động gửi trả dữ liệu của Spring MVC tuân theo luồng xử lý yêu cầu của DispathcherServlet. Chúng ta sẽ tìm hiểu chi tiết trong mục DispacherServlet.

#### DispathcherServlet

Giống như bất kỳ servlet nào, Spring cung cấp DispatcherServlet. Nhiệm vụ của DispathcherServlet là xử lý tất cả HTTP request và response.

Dưới đây chúng ta sẽ tìm hiểu luồng đi của dữ liệu được lấy từ browser gửi đi xử lý thông qua DispathcherServlet và nhận kết quả trả về cho View.

Hình .: Luồng dữ liệu của Spring MVC với DispathcherServlet

Luồng dữ liệu được thể hiện theo các bước sau:

* Bước 1: nhận HTTP request. DispathcherServlet gửi yêu cầu đến HandlerMapping để xác định controller sẽ xử lý yêu cầu.
* Bước 2: Controller sẽ nhận các request và gọi các service phù hợp dựa trên phương thức GET, POST, PUT, DELETE. Nếu yêu cầu đó cần truy xuất cơ sở dữ liệu thì Controller sẽ ủy nhiệm cho logic nghiêp vụ khác để lấy thông tin và gửi dữ liệu về cho Controller. Lúc này Controller sẽ đóng gói dữ liệu và gửi đến DispathcherServlet.
* Bước 3: DispathcherServlet sẽ gửi gói tin cho ViewResolver để tìm View hiển thị.
* Bước 4: Sau khi hoàn tất, DispathcherServlet sẽ chuyển dữ liệu từ model đến view và cuối cùng hiển thị dữ liệu lên trình duyệt.

Nhờ vào DispathchetServlet nên Spring MVC trở thành công nghệ tiện ích cho các ứng dụng web.

### Spring security

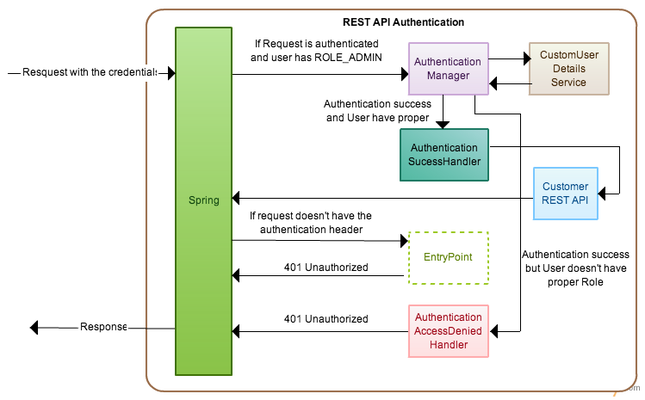
#### Giới thiệu về Spring Security

Spring Security là một module của Spring framework. Nó cung cấp một giải pháp bảo mật hoàn hảo cho các ứng dụng J2EE.

Spring Security xuất hiện vào cuối năm 2003 với một dự án nhỏ có tên là “The Acegi Security System for Spring”. Sau đó, các dịch vụ xác thực Acegi Security được giới thiệu. Một năm sau đó, Acegi Security trở thành một dự án nhỏ của Spring framework. Phiên bản 1.0.0 được phát hành vào 5/2006 sau hơn 2 năm rưỡi phát triển với nhiều cải thiện đáng kể. Cuối năm 2007, Acegi Security chính thức trở thành một một dự án của Spring và được đổi tên thành Spring Security.

#### Kiến trúc

Đặc điểm nổi bật nhất của Spring Security là cơ chế xác thực (authentication) và ủy quyền (authorization). Đây cũng là cơ chế tạo nên sự bảo mật tuyệt vời cho các ứng dụng Java Spring. Bên dưới là hình ảnh minh họa toàn bộ hoạt động của Spring Security



Hình .: Luồng hoạt động của Spring Security

Luồng hoạt động của Spring Security sẽ được thực hiện theo các bước sau:

* Bước 1: Yêu cầu của người dùng được gửi đi và yêu cầu đó được bắt lại bằng Dispathcher.
* Bước 2: Nếu yêu cầu của người dùng đã được xác thực và người dùng đó là người quản trị thì yêu cầu sẽ được gửi đến Authentication Manager. Ngược lại, nếu yêu cầu không được xác nhận sẽ gửi lại thông báo “401 Unauthorized”.
* Bước 3: Yêu cầu đó tiếp tục được gửi tới CustomUser Details Service và trả ngược lại cho Authencication Manager.
* Bước 4: Nếu Authentication Manager xác thực thành công và đã đúng vai trò của người dùng sẽ gửi yêu cầu tới Authentication SucessHandler. Nhưng nếu việc xác thực của Authentication Manager không thành công sẽ trả về thông báo “401 Unauthorized”.
* Bước 5: Authentication SucessHandler sẽ tiếp tục gửi yêu cầu tới Customer REST API sau đó trả lại yêu cầu cho Dispathcher và hirn thị thông báo cho người dùng.

Cơ chế xác thực (Authentication) được hiểu đơn giản như một hàng rào chặn tất cả yêu cầu và chờ xác thực trước khi vào bên trong. Cơ chế này giúp hệ thống được bảo mật tốt hơn vì tất cả yêu cầu vào hệ thống đều được kiểm soát.

Cơ chế phân quyền (Authorization) giúp ứng dụng xác định được vai trò của người dùng để thực hiện các chức năng tương ứng của họ.

Đây là hệ thống bảo mật toàn diện của Spring Security. Các hệ thống sử dụng Spring Security có sự bảo mật tốt hơn bởi cơ chế bảo mật tuyệt vời của nó.

## Tìm hiểu về AngularJS

### Giới thiệu về AngularJS

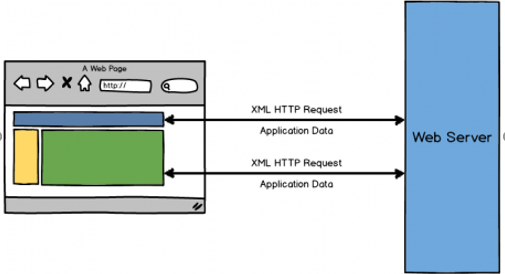
Sự thành công của Jquery làm cho ngôn ngữ JavaScript trở nên phổ biến hơn bởi tính đơn giản và hiệu quả của nó. Tiếp bước sự thành công đó là sự ra đời của AngularJS.

AngularJS là một framework có cấu trúc cho các ứng dụng web động. Nó cho phép sử dụng HTML như một ngôn ngữ mẫu và cho phép mở rộng cú pháp của HTML để trình bày các thành phần của ứng dụng một cách rõ ràng và súc tích. Đây là một framework hỗ trợ xây dựng cơ chế single-page cho ứng dựng website. AngularJS làm đơn giản các ứng dụng phức tạp. Data binding và Dependency injection là hai tính năng cốt lỗi của AngularJS. Với hai tính năng này, AngularJS loại bỏ phần lớn mã nguồn. Vì vậy AngularJS trở thành đối tác lý tưởng cho bất kỳ công nghệ Server nào.

AngularJS được bắt đầu phát triển từ năm 2009 bởi lập trình viên Misko Hevery tại Google. Lúc này, Misko đang tham gia dự án của Google. Việc sử dụng AngularJS phía front-end làm giảm thiểu đáng kể số dòng code, từ 17000 dòng xuống khoảng 1500 dòng. Với sự thành công đó, Misko và nhân viên tham gia dự án của Google đã quyết định phát triển AngularJS. Mặc dù là một framework mới nhưng với các tính năng mạnh mẽ của AngularJS trở thành sự lựa chọn cho nhiều nhà phát triển front-end.

AngularJS là một mã nguồn mở, hoàn toàn miễn phí và được sử dụng bởi hàng ngàn nhà phát triển trên toàn thế giới. Người dùng có thể sửa đổi, đóng góp vào mã nguồn giúp AngularJS trở nên hoàn thiện hơn.

Đặc điểm nổi bật nhất của AngularJS là cơ chế data-binding cho phép các nhà phát triển ngôn ngữ HTML và CSS tiết kiệm thời gian tải dữ liệu và làm đơn giản thao tác của người dùng.

Các ứng dụng website ngày nay thường được xây dựng như một Single Page (SPA). Và AngularJS là một lựa chọn tuyệt vời cho các nhà phát triển muốn phát triển ứng dụng theo hướng trang đơn.

Hình .: Ứng dụng web sử dụng Single Page

Một ứng dụng website xây dựng như một Single Page giúp người dùng có trải nhiệm như đang tương tác với một ứng dụng desktop thật sự. Nghĩa là ứng dụng không phải tải lại toàn bộ tài nguyên mỗi lần thao tác.

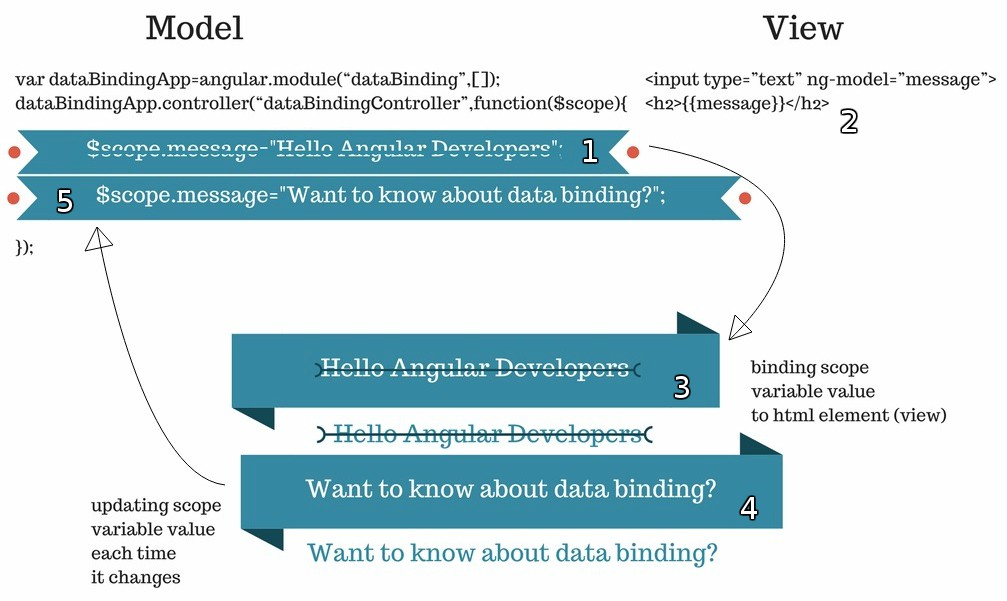
Ngoài ra, AngularJS còn được phát triển theo hướng mô hình kiến trúc MVC. Đây là một cách hữu ích để tách các thành phần khi phát triển một ứng dụng lớn. Mỗi thành phần phục vụ một mục đích riêng. Model là dữ liệu, view là giao diện người dùng và controller là logic nghiệp vụ. Mô hình này được xây dựng để tách sự phụ thuộc vào nhau, tăng tính tái sử dụng code và dễ dàng duy trì, phát triển ứng dụng.

Tài liệu tham khảo chính có thể xem tại đường dẫn: <https://docs.angularjs.org>

### Đặc trưng

Angular có rất nhiều đặc trưng, và dưới đây là một vài đặc điểm của nó: data-binding, Single-page, kiến trúc mô hình MVC, Dependency Injection (DI), các Directives, …

#### Data binding

Trong các ứng dụng truyền thống, giao diện người dùng được tạo ra bằng cách kết hợp mã HTML và dữ liệu được xử lý từ phía server. Bất cứ khi nào có sự thay đổi yêu cầu từ phía người dùng, server phải gửi lại toàn bộ mã HTML và dữ liệu được xử lý ngay cả khi mã HTML đã có sẵn.

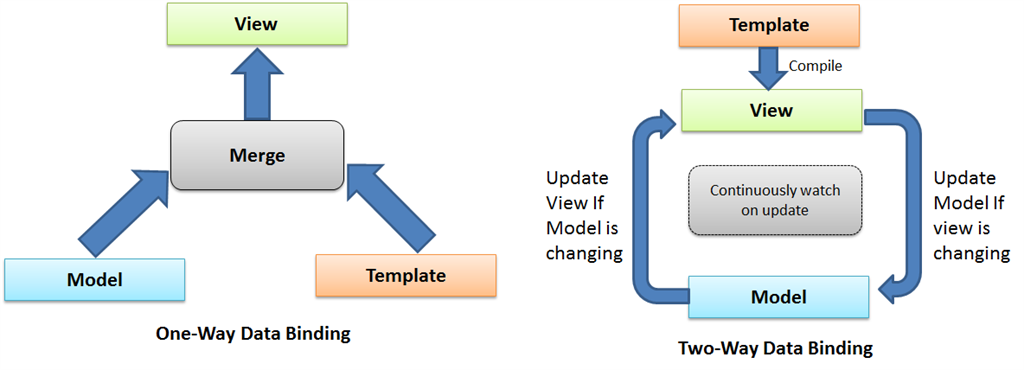
Hình .: Cơ chế data binding

Đó là hình ảnh mô phỏng cho data-binding. Bước 1 là đăng ký giá trị “Hello Angular Developers” cho scope có tên là message. Bước 2, dò tìm biến message ở UI trong thẻ <input> với chỉ thị “ng-model= ‘message’” và sau đó dò tìm biến có scope là message trong thẻ <h2> bằng {{message}}. Bước 3, dò tìm dòng văn bản “Hello Angular Developers” được hiển thị trong thẻ <input>. Bước 4, thay đổi dòng văn bản “Hello Angular Developers” trong thẻ <input> bằng “Want to know about data binding”. Bước 5, cập nhật giá trị mới thay đổi vào biến scope là message bằng cách dùng “$scope.message = “Want to know about data binding” và quá trình này được lặp lại tại bước 2.

Điển hình cho data-binding là cơ chế two-way binding. Two-way binding tạo nên sức mạnh thành công cho AngularJS. Để hiểu rõ hơn về cơ chế two-way binding, đầu tiên chúng ta tìm hiểu cơ chế one-way binding.

One-way binding được thực hiện như sau: AngularJS sẽ binding tìm giá trị biến scope và cập nhật giá trị của biến scope ở giao diện mỗi khi có yêu cầu thay đổi dữ liệu.

Ở cơ chế one-way binding AngularJS chỉ cập nhật dữ liệu theo một chiều. AngularJS có thể cập nhật dữ liệu hai chiều bằng cơ chế two-way binding nhờ tính năng đồng bộ hóa giữa model và view. Quá trình đồng bộ này diễn ra liên tục như một vòng tròn giữa model và view. Mọi sự thay đổi trong lớp này, ngay lập tức lớp kia cũng được thay đổi theo. Chúng ta chỉ binding một lần và việc còn lại là của AngularJS. Hình ảnh bên dưới sẽ mô tả rõ hơn về cơ chế one-way binding và cơ chế two-way binding.



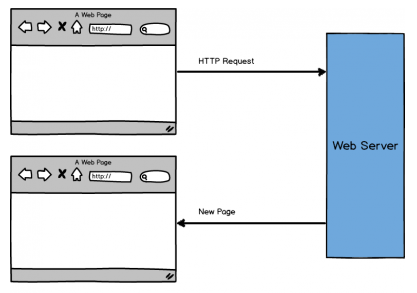
Hình .: Cơ chế one-way binding và two-way binding

Có thể thấy, thành phần ở controller và view rất rõ ràng (chi tiết hơn ở mục kiến trúc MVC). Dữ liệu ở view thực chất là được chuyển đến từ model nên controller không cần thực hiện các thao tác binding phức tạp và phụ thuộc quá nhiều vào cấu trúc DOM của view. Như vậy quá trình debug và viết unit test trở nên dễ dàng hơn.

#### Single-page (SPA)

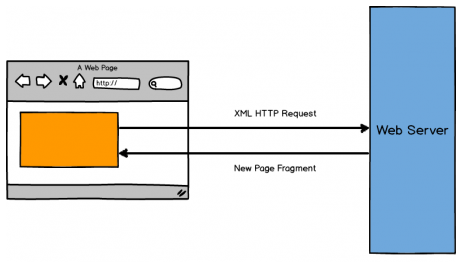
Ứng dụng Single page là ứng dụng web chạy bên trong của trình duyệt web và không yêu cầu tải lại trang trong quá trình sử dụng. Tất cả mã JavaScript, HTML5, CSS3 được tải lên từ yêu cầu đầu tiên. Có thể kể đến các ứng dụng SPAs phổ biến hiện nay như: Gmail, Google Maps, Google Drive, iCloud, GitHub, Hotmail, …

Ứng dụng web tĩnh không còn phù hợp với xu thế hiện nay vì chúng khó quản lý, khó bảo trì. Hình bên dưới minh họa cho quá trình hoạt động của một ứng dụng web tĩnh.



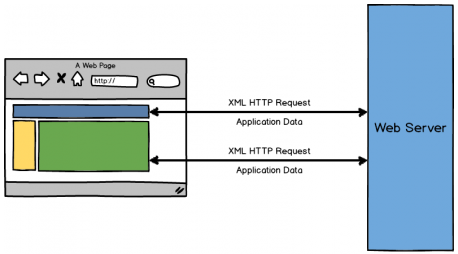
Hình .: Ứng dụng web application

Sau đó, Ajax ra đời như một điển hình cho mô hình MVC. Với sự hỗ trợ của Javascript, mô hình này cho phép sự thay đổi bên trong của trang và có thể hiển thị giá trị mới từ server mà không cần tải lại trang. Bên dưới là mô hình mô tả cách hoạt động của của website sử dụng Ajax.



Hình .: Ứng dụng sử dụng Ajax

Tuy Ajax là một cải tiến mới nhưng không phải là giải pháp hoàn hảo. Hiệu suất của Ajax kém. Việc truyền dữ liệu không cần thiết có thể xảy ra mỗi khi tương tác với người dùng. Ngoài ra ứng dụng web sử dụng Ajax có thể trở nên vô dụng khi bị ngắt kết nối.

Sự ra đời của SPA là giải pháp hiệu quả để giải quyết các vấn đề Ajax không đáp ứng được. Hình ảnh bên dưới là minh họa cho luồng hoạt động của ứng dụng sử dụng Single-page.

Hình .: Ứng dụng sử dụng SPA

.

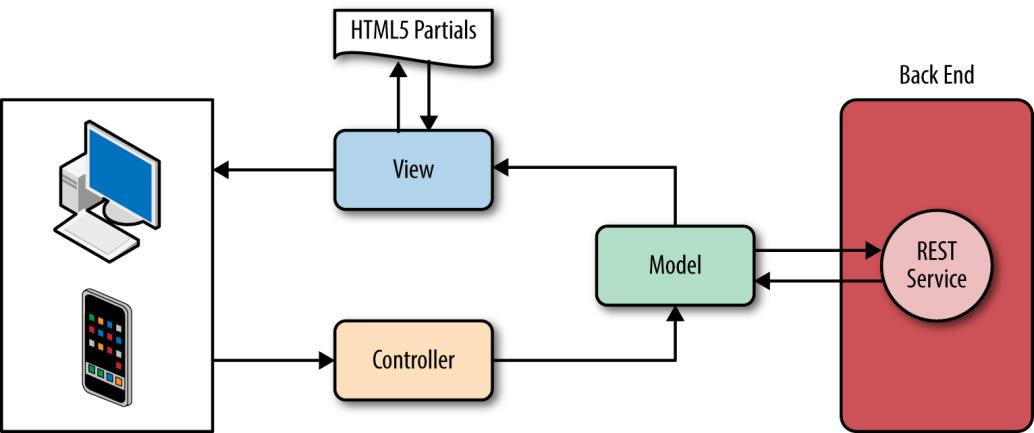
SPA cung cấp tốc độ truy xuất nhanh như một ứng dụng desktop. SPA mang đến một số lợi ích như sau:

* Cải thiện tốc độ truy xuất của website. Các tài nguyên của website như các mã HTML5, CSS3, Javascript được tải một lần duy nhất trong suốt quá trình chạy ứng dụng và nó chỉ cập nhật lại dữ liệu bị thay đổi.
* Đơn giản hóa quá trình kiểm thử. Người lập trình có thể dễ dàng phát hiện lỗi trên trình duyệt vì chúng hoạt động theo mô hình MVC.
* Ứng dụng có thể lưu trữ các biến ở local storage hiệu quả. Các dữ liệu giống nhau chỉ cần tải một lần. Tất cả các yêu cầu sau đó sẽ được lấy từ local storage. Điều này làm giảm tối đa yêu cầu từ phía người dùng

Tuy nhiên, việc sử dụng SPA cho ứng dụng cũng có những mặt hạn chế. Nó gây khó khăn cho quá trình làm SEO vì nội dung chỉ tài một lần duy nhất vì vậy không lưu lại lịch sử tìm kiếm theo đường dẫn trên Google. Và cuối cùng là việc tải ứng dụng lần đầu sẽ mất nhiều thời gian hơn đối với các ứng dụng lớn.

#### Mô hình MVC

MVC (Model-View-Controller) là một kiến trúc phần mềm rất phổ biến hiện nay. MVC không chỉ được sử dụng trong mô hình kiến trúc phía server mà nó còn được áp dụng cho ứng dụng phía client và AngularJS đã tận dụng mô hình kiến trúc này một cách hiệu quả. MVC chia ứng dụng phía client thành ba phần tương tác với nhau để tách biệt giữa cách thức mà thông tin được xử lý nội hàm và thông tin được trình bày, tiếp nhận từ phía người dùng.

Mỗi phần chịu trách nhiệm cho một nhiệm vụ riêng. Model (xử lý và truy xuất dữ liệu), View (giao diện người dùng), Controller (bộ phận điều hướng giữa View và Model)

Hình .: Mô hình MVC

Đối với AngularJS, View là DOM, Controller là các lớp JavaScript, Model là dữ liệu được lưu ở thuộc tính của các đối tượng trong JavaScript.

#### Dependency Injection (DI)

Dependency Injection (DI) là nơi chứa các đối tượng phụ thuộc được định nghĩa sẵn trong ứng dụng. DI là một giải pháp để ghép các đoạn code lại với nhau, điều này đòi hỏi phải phân chia các đoạn code theo các chức năng tách biệt như value, factory, service, provider và constant.

Value là một đối tượng đơn giản của JavaScript. Nó được sử dụng để truyền giá trị tới controller.

Factory được sử dụng để trả về các giá trị. Nó tạo ra giá trị theo yêu cầu bất cứ khi nào nhận được yêu cầu từ server hay controller.

Service là một đối tượng đơn lẻ của JavaScript. Nó định nghĩa các phương thức của service và được thêm vào controller.

Provider được sử dụng để tạo ra các service, factory, … Nó sử dụng phương thức get để lấy giá trị trả về từ value, service và factory.

Constant được sử dụng để đưa giá trị ở giai đoạn cấu hình

DI làm cho các ứng dụng AngularJS trở nên dễ phát triển hơn vì có thể sắp xếp code thành nhiều module khác nhau. Bằng cách này, người lập trình có thể tạo ra thành phần có khả năng tái sử dụng về sau. Nó là một cách phát triển hiệu quả cho các ứng dụng lớn cần được mở rộng, bảo trì và kiểm thử.

#### Directives

Tính năng này thật sự đã tạo nên điểm nhấn mạnh mẽ giữa AngularJS với bất kỳ framework JavaScript khác. Directives giúp làm giảm hàng ngày dòng code cho người lập trình. Nó mở rộng HTML, giảm số lượng thẻ HTML. Code HTML sẽ gòn gàng, dễ hiểu hơn.

Mặc dù HTML5 ra đời với số lượng thẻ phong phú, nhưng để tạo ra một giao diện website hoàn hảo là không thể. AngularJS cung cấp các directives giúp người lập trình tạo ra những thẻ HTML mới hoặc các thuộc tính mới cho tất cả các thẻ trong HTML.

Bên dưới là một số directive quan trọng của AngularJS

* Đầu tiên, ng-app là một directive quan trọng của AngularJS. Khi directive được khởi tạo, Angular sẽ hiểu đây là thành phần gốc của ứng dụng. Nó sẽ khởi tạo các thông số, cấu hình bên trong. Trong cùng một trang HTML chỉ có một directive ng-app được khai báo. Bên cạnh đó, cũng có thể sử dụng directive ng-app mà không cần bất kỳ tham số nào
* Tiếp theo, có thể kể đến directive ng-controller. Đây là directive chỉ cho ứng dụng biết đây là một controller, nó được đính kèm vào DOM của HTML. Các thuộc tính và phương thức được khai báo trong controller được cập nhật tự động vào giao diện thông qua biến scope khi sử dụng directive này. Trong một số trường hợp, cần chia nhỏ controller nếu controller đó quá phức tạp.
* Kế tiếp là directive ng-bind dùng để thay thế nội dung của các đoạn mã HTML và cập nhật giá trị mới từ controller. Có thể sử dụng “{{}}” để thay thế cho thẻ ng-bind. Ví dụ bên dưới có giá trị trả về là một đối tượng “group”. Nhưng view cần hiển thị các thuộc tính của group là id, code, name và description thì dùng directive ng-bind hoặc dùng “{{}}”.
* Directive ng-model được đặt trong thành phần của HTML. Nó được sử dụng nhiều nhất trong thẻ <input>. Mục đích của directive này là dùng để lấy dữ liệu nhập vào từ phía người dùng. Ngoài ra nó còn cung cấp các thao tác validate như kiểm tra trường nhập vào có phải là kiểu số, kiểu email không, …. Nó còn thiết lập các css class trong thẻ HTML bằng các directive: ng-valid, ng-invalid, ng-dirty, ng-touched, …
* ng-repeat là một directive quan trọng trong việc hiển thị một danh sách giá trị lên view. Có thể hiểu ng-repeat như một vòng lặp, nó hiển thị lần lượt từng giá trị trong một danh sách. Ở ví dụ bên dưới có thể thấy “groups” là một danh sách và “group” là giá trị trong danh sách “groups”. Chúng ta dùng directive ng-repeat như cấu trúc bên dưới để lặp một danh sách.
* Một directive dùng để bắt dự kiện được Angular hỗ trợ là ng-click. Directive ng-click xảy ra khi thao tác click chuột vào button bất kỳ. Hàm trong directive sẽ được gửi về controller. Hàm gửi đến controller có thể hoặc không bao gồm dữ liệu.

### Lợi ích và khó khăn của AngularJS

#### Lợi ích

Ứng dụng web sử dụng AngularJS phía client thu được một số lợi ích sau:

* Nhờ vào Single Page mà ứng dụng cải thiện tối đa tốc độ tải tài nguyên. Tài nguyên ứng dụng web chỉ tải vào lần đầu tiên truy cập và thay đổi dữ liệu cần thiết.
* Công việc kiểm thử của lập trình viên trở nên đơn giản hơn do các ứng dụng AngularJS phân chia các thành phần rõ ràng bằng mô hình MVC.
* Mô hình kiến trúc MVC giúp tăng tính tái sử dụng code, tiết kiệm hàng ngàn dòng code cho ứng dụng.
* Cơ chế DI làm giảm thiểu sự phụ thuộc code

#### Khó khăn

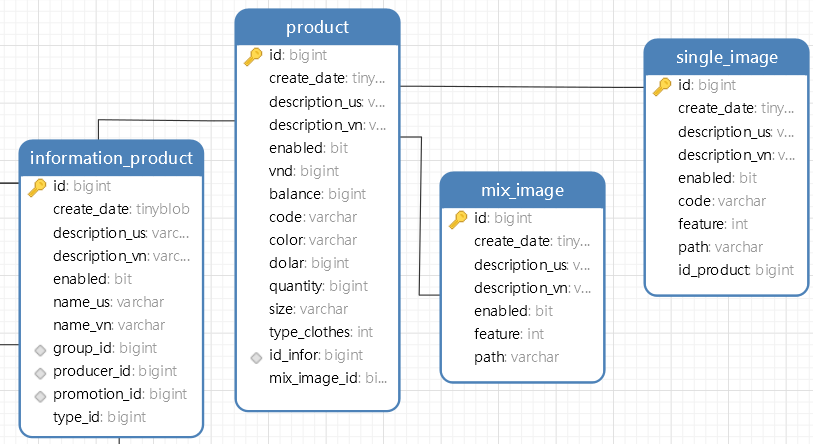
Bên cạnh đó, các ứng dụng áp dụng AngularJS có gặp một số vấn đề sau:

* Các ứng dụng web AngularJS không an toàn. Vì vậy việc phân quyền và các vấn đề bảo mật phải được thực hiện phía server.
* Ứng dụng AngularJS không thể phân rã. Nếu người dùng vô hiệu hóa mã JavaScript và họ chỉ nhìn thấy cấu trúc đơn giản của trang.
* Đặc biệt, các ứng dụng web của AngularJS gây khó khăn cho công tác SEO vì nó ứng dụng đặc tính trang đơn nên không tải lại trang và không hiển thị đường dẫn ở trình duyệt.

## Các giải thuật quan trọng

### Lưu trữ hình ảnh

Website thời trang luôn chú trọng đến hình ảnh hiển thị cho người xem. Hình ảnh càng đẹp, càng bắt mắt sẽ làm tăng sức của khách hàng. Vì vậy vấn đề lưu trữ hình ảnh trở nên khó khăn hơn khi số lượng hình ảnh lớn và kích thước hình ảnh khác nhau tùy thuộc vào vị trí hiển thị của chúng.

Để giải quyết được bài toán làm thế nào để lưu trữ được khối lượng lớn hình ảnh đó lại vừa truy xuất nhanh? Bài toán này sẽ được giải đáp thông qua giải thuật lưu trữ hình ảnh được thực hiện rõ ràng nhất ở cơ sở dữ liệu của hệ thống.

Hình .: Giải thuật lưu trữ hình ảnh

Hình ảnh trang phục được phân ra thành 2 loại cụ thể là trang phục đơn và trang phục phối tương ứng với 2 bảng single\_image và mix\_image. Trang phục phối gồm một hoặc nhiều trang phục đơn hợp thành (có thể là một áo, một quần và một áo khoác). Cụ thể như sau:

* Đối với bảng single\_image: Trường code đánh dấu ba vị trí hiển thị của cùng một trang phục. Nếu ba record trong bảng single\_image có cùng code nghĩa là chúng cùng thể hiện 1 hình ảnh giống nhau, nhưng khác về kích thước và độ phân giải. Trường feature quy định vị trí hiển thị của hình ảnh dựa vào kích thước hình ảnh là hỉnh ảnh nhỏ, vừa, và hình lớn tại trang chi tiết trang phục.

### Web service

Công nghệ phát triển tạo ra nhiều nền tảng mạnh mẽ cho ứng dụng. Công việc lập trình của các lập trình viên sẽ dễ dàng hơn khi một server cung cấp dịch vụ cho nhiều nền tảng sử dụng mà không cần tạo ra nhiều dịch vụ khác nhau. Nghĩa là chỉ cần một server nhưng có thể cung cấp dịch vụ cho nhiều nền tảng khác nhau như IOS, Android, Web, ….

Hệ thống áp dụng web service để tách biệt hai thành phần client và server. Nhiệm vụ của server là cung cấp dịch vụ-cung cấp data cho client. Hệ thống sử dụng Spring framework để cung cấp data dưới dạng JSON. Client có thể sử dụng nhiều công nghệ khác nhau để trình bày hay hiển thị data cho người xem. Công nghệ trong hệ thống được áp dụng là AngularJS. Ngoài ra việc hiển thị data ở client có thể được thực hiện bằng nhiều thư viện khác như VueJS, ReactJS, …

### Bảo mật

Website sử dụng cơ chế phân quyền và xác thực được thể hiện rõ như sau:

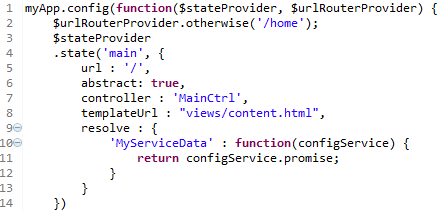
* Website được phân ra thành 4 loại tài khoản như sau:
* Root: tài khoản có quyền cao nhất. tài khoản này có tất cả các quyền bao gồm việc quản lý admin, user đã có tài khoản, có thể chuyển quyền từ một tài khoản thường thành admin hoặc ngược lại. Ngoài ra, root còn có một số quyền quản lý quan trọng mà admin không có.
* Admin: quản lý từng thành phần của hệ thống như: user thường, thông tin sản phẩm, thông tin nhà sản xuất….
* User đã có tài khoản có thể thực hiện việc mua hàng trực tuyến
* User chưa đăng ký tài khoản chỉ có thể xem trang phục, lưu vào giỏ hàng để xem sau lần đăng nhập tiếp theo trên cùng một máy, user không thao tác được các chức năng mua hàng trực tuyến.
* Cơ chế xác thực của website được thể hiện như sau:
* Khi người dùng đăng ký một tài khoản mới, mật khẩu sẽ được mã hóa theo dạng Bcript và lưu xuống database.
* Khi người dùng đăng nhập vào hệ thống với tên tài khoản và mật khẩu nhập vào. Mật khẩu sẽ được mã hóa bằng cơ chế Bcript. Spring Security sẽ thực hiện công việc xác thực bằng cách so sánh tên tài khoản và mật khẩu nhập vào với tên tài khoản và mật khẩu dưới database. Nếu việc xác thực được thực hiện thành công, hệ thống sẽ tạo ra một chuỗi token theo chuẩn JWT (JSON WEB token) chứa thông tin của mật khẩu và ngày tháng đăng nhập. Sau đó, chuỗi token này sẽ được trả về cho client và được lưu trữ ở local storage. Mỗi lần có yêu cầu xác thực từ phía người dùng, client sẽ gửi yêu cầu có đính kèm token trên header. Spring MVC sẽ lọc chuỗi token trước khi vào controller để thực hiện việc xác thực. Tại vị trí này, chuỗi tolen sẽ được lấy ra và hệ thống sẽ lấy được tên tài khoản và mật khẩu để thực hiện công việc xác thực.

### Single page

Website ứng dụng angular js để xây dựng hệ thống single page. Website chỉ tải tài nguyên trong lần truy cập đầu tiên. Các thao tác của người dùng lên giao diện sẽ được giữ nguyên trên một trang. Nó chỉ cập nhật lại những dữ liệu thay đổi. Quá trình này sẽ diễn ra với tốc độ nhanh hơn bởi nó không cần tải lại tài nguyên đã có.

Các bước để tích hợp và cài đặt tính năng này cho website.

* C:\Users\Art_Kiss\AppData\Local\Microsoft\Windows\INetCache\Content.Word\21076614_2415128855379302_887589348_n (1).pngBước 1: Thêm thư viện angularJS và thư viện ui-router
* C:\Users\Art_Kiss\AppData\Local\Microsoft\Windows\INetCache\Content.Word\21121857_2415129052045949_2087307546_n.pngBước 2: Khai báo angularJS với ứng dụng
* C:\Users\Art_Kiss\AppData\Local\Microsoft\Windows\INetCache\Content.Word\21076761_2415129312045923_49712836_n.pngBước 3: Khai báo ui-router cho ứng dụng
* Bước 4: Config ui-router cho ứng dụng



Đó là các bước tạo nên ứng dụng single-page.

### Đa ngôn ngữ

Hệ thống tích hợp chức năng đa ngôn ngữ. Website có thể chuyển đổi ngôn ngữ qua lại giữa tiếng Việt và tiếng Anh.

Cơ chế đa ngôn ngữ được thực hiện như sau:

* Phía client:
* Lựu một biến tên “location” vào local stogare với giá trị “true” hoặc “false” để xác định ngôn ngữ mà người dùng muốn chọn. Giá trị “true” tương ứng với “tiếng Anh”, “false” là “tiếng Việt”. Và mặc định biến này được gắn giá trị là true.
* Đối với những request yêu cầu xác định ngôn ngữ, header của request phải gửi biến location về server, server dựa vào biến location để xác định dữ liệu có ngôn ngữ tương ứng cần trả về cho client
* Phía server:
* Đối với dữ liệu cần đa ngôn ngữ (tên sản phẩm, mô tả sản phẩm…) lưu thành 2 trường là tiếng Anh và tiếng Việt
* Đối với các controller cần trả về cho client dữ liệu đa ngôn ngữ cần phải nhận thêm biến location để xác định vị trí của request cần lấy data, từ đó dùng Spring Data truy xuất dữ liệu từ database, dựa vào biến localtion để chuyển đổi đúng ngôn ngữ vào object data cần trả về client.

## Hiện thực bài toán

### Kiến trúc hệ thống

Áp dụng các cơ sở lý thuyết và các kỹ thuật xây dựng website ở trên, chúng tôi sẽ tiến hành phân tích hệ thống website thời trang Tarifa.

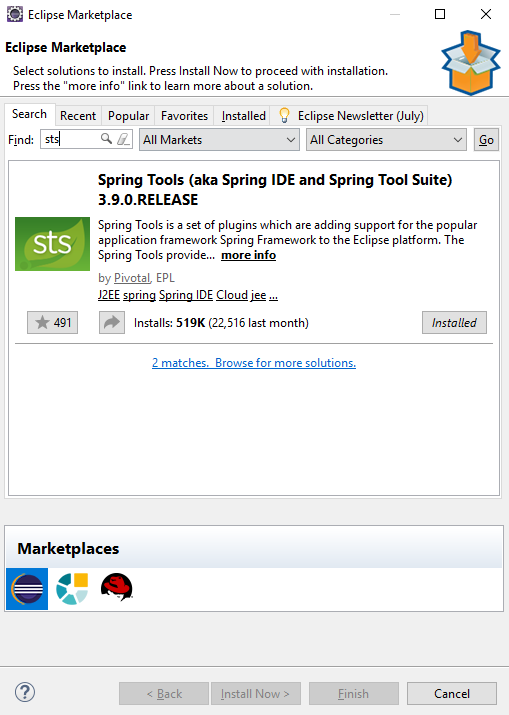
Phía dưới là mô hình kiến trúc tổng quát của website.

Hình .: Mô hình kiến trúc hệ thống

### Cài đặt framework

* Tải JDK 8 tại địa chỉ <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
* Cài đặt biến môi trường
* Tải và cài đặt Spring Tool Suit trong eclipse hoặc tải trực tiếp Spring Tool

Vào **Help** → **Eclipse Marketplace** → tìm theo từ khóa **STS** (Spring Tool Suit). Xem hình bên dưới



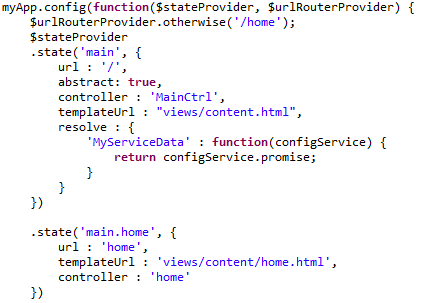
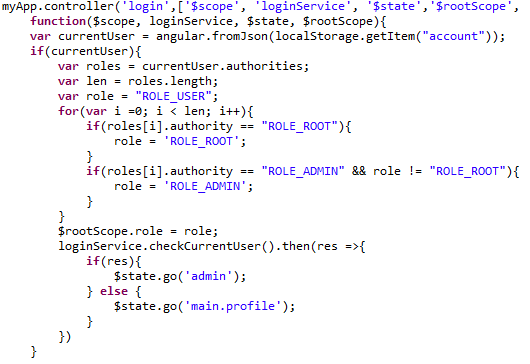
Hình .: Tìm kiếm và cài đặt Spring Tools trong eclipse

### Cấu hình cho hệ thống

#### Cấu hình front-end

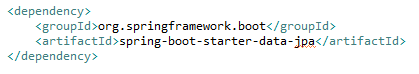
* Bước 1: Tải và tích hợp thư viện của AngularJS
* Bước 2: Thêm thư viện ui-router để cấu hình single-page
* Bước 3: Đăng ký AngularJS cho ứng dụng

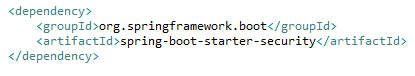


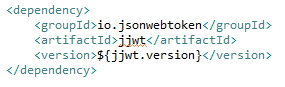
* Bước 4: Cấu hình single-page cho ứng dụng
* Bước 5: Tạo controller
* Bước 6: Tạo service

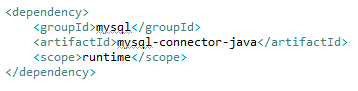
#### Cấu hình phía back-end

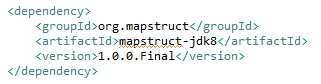
* Bước 1: Thêm thư viện vào file pom.xml. Thư viện được ứng dụng sử dụng bao gồm: JPA, Spring Security, JSON Web Token, MySQL, Mapstruct, Tomcat, Mail.



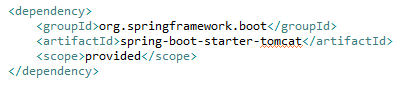


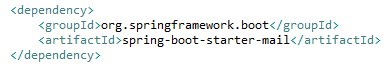




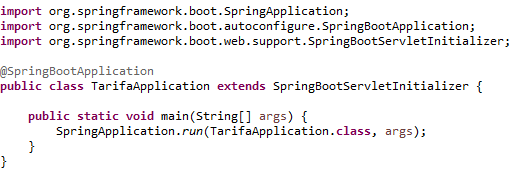








* Bước 4: Chạy hạy ứng dụng



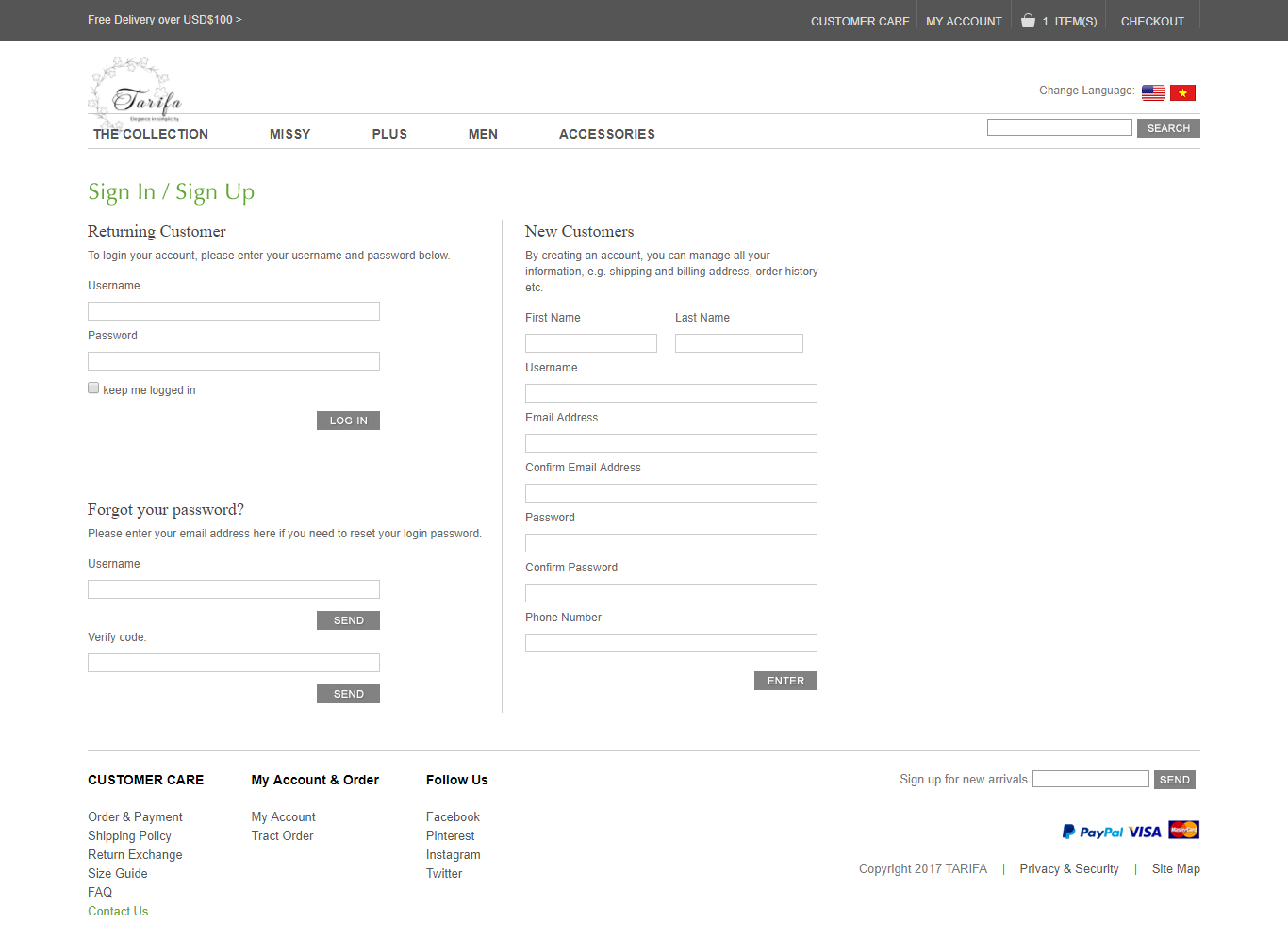
# Kết quả đạt được

## Hệ thống front-end

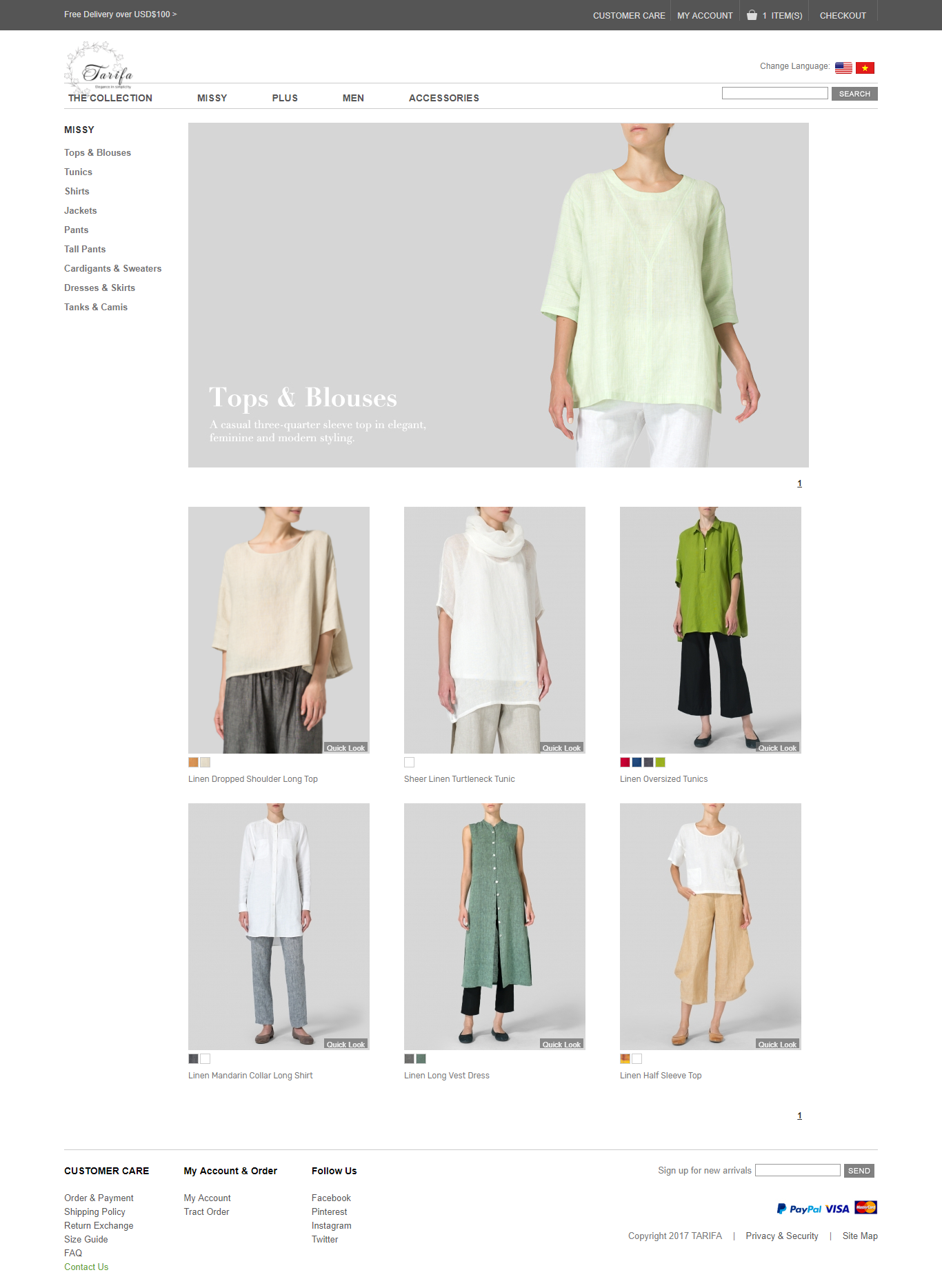
* Giao diện trang nhã, thanh lịch phù hợp với một website kinh doanh thời trang mang phong cách vintage. Bên cạnh đó, hình ảnh của trang phục được tùy chỉnh hợp lý tạo cảm giác thật cho người xem.
* Áp dụng thành công công nghệ AngularJS với các chức năng sau:
* Website được triển khai theo hướng single page application (SPA). Với tính năng này, website hạn chế việc tải lại các tài nguyên, từ đó cải thiện được tốc độ tải một cách tối ưu nhất.
* Mô hình MVC được sử dụng khá linh hoạt giúp phân chia hệ thống phía client một cách rõ ràng. Tách biệt của các thành phần mà hệ thống dễ dàng tái sử dụng giúp giảm việc viết mã nguồn.
* Sử dụng localstogare để tạo giỏ hàng, đa ngôn ngữ, hỗ trợ việc xác thực, phân quyền.
* Các hình ảnh liên quan được thể hiện bên dưới:



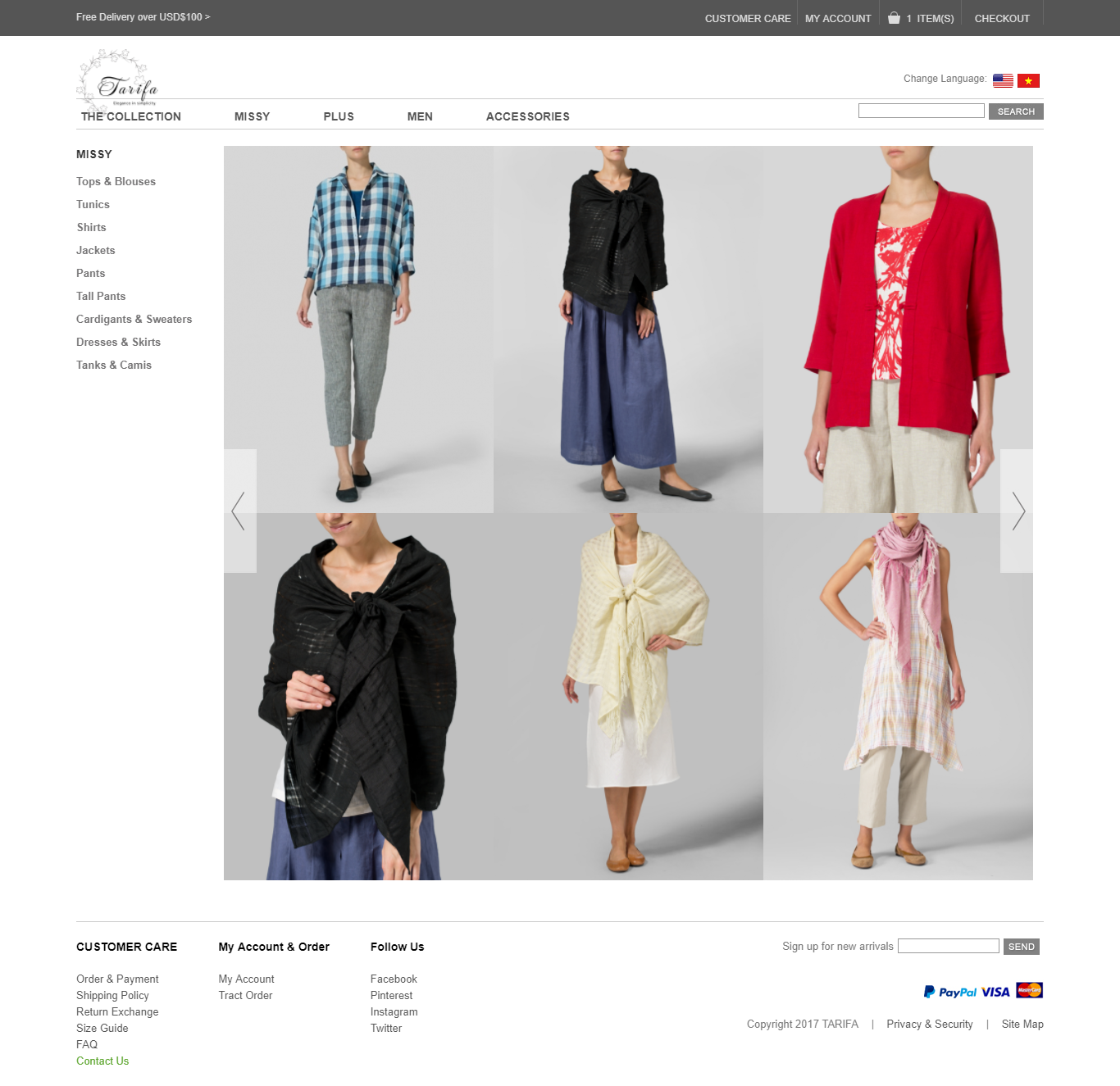
Hình .: Giao diện trang chủ

****

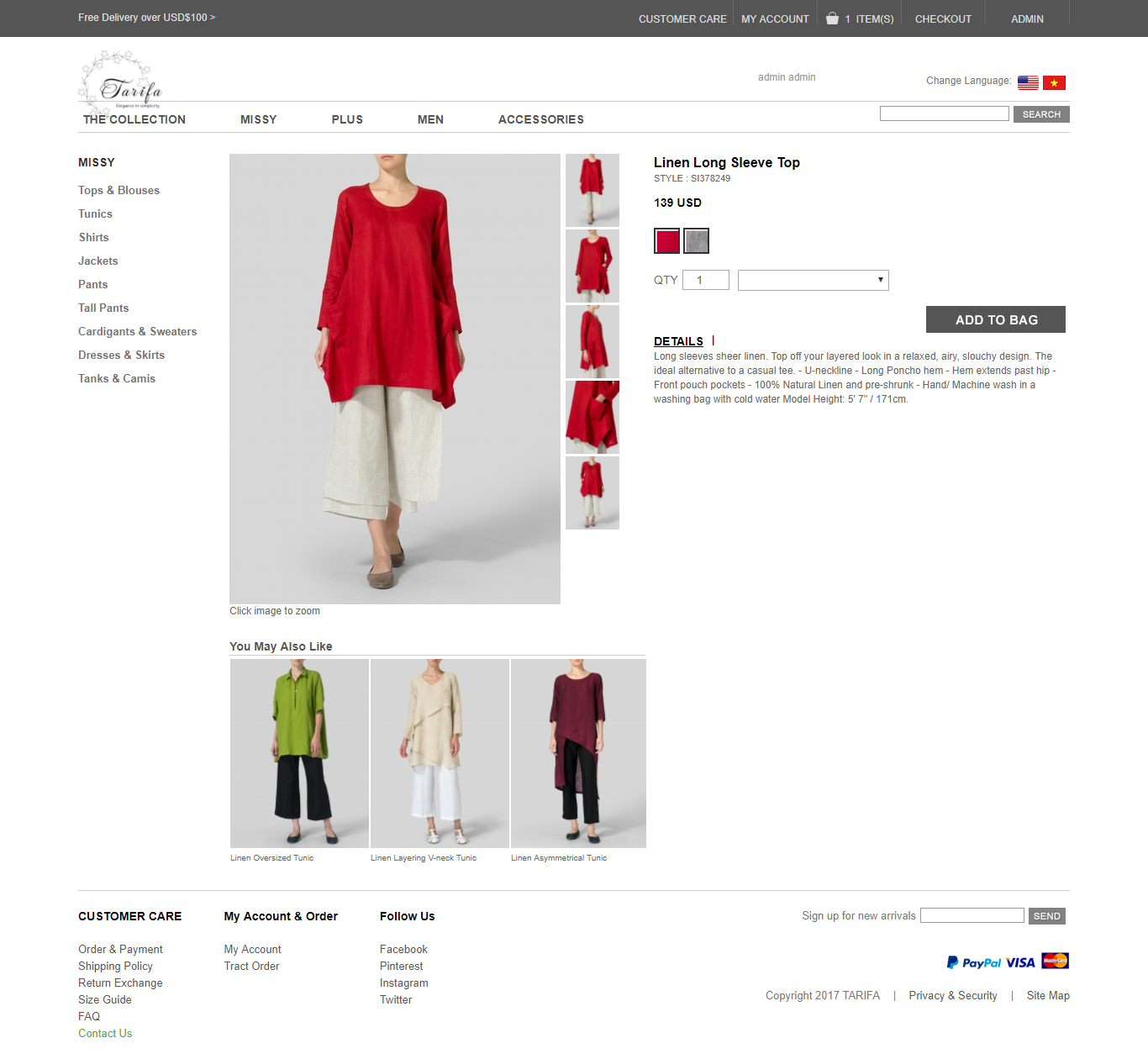
Hình .: Giao diện đăng nhập, đăng ký

****

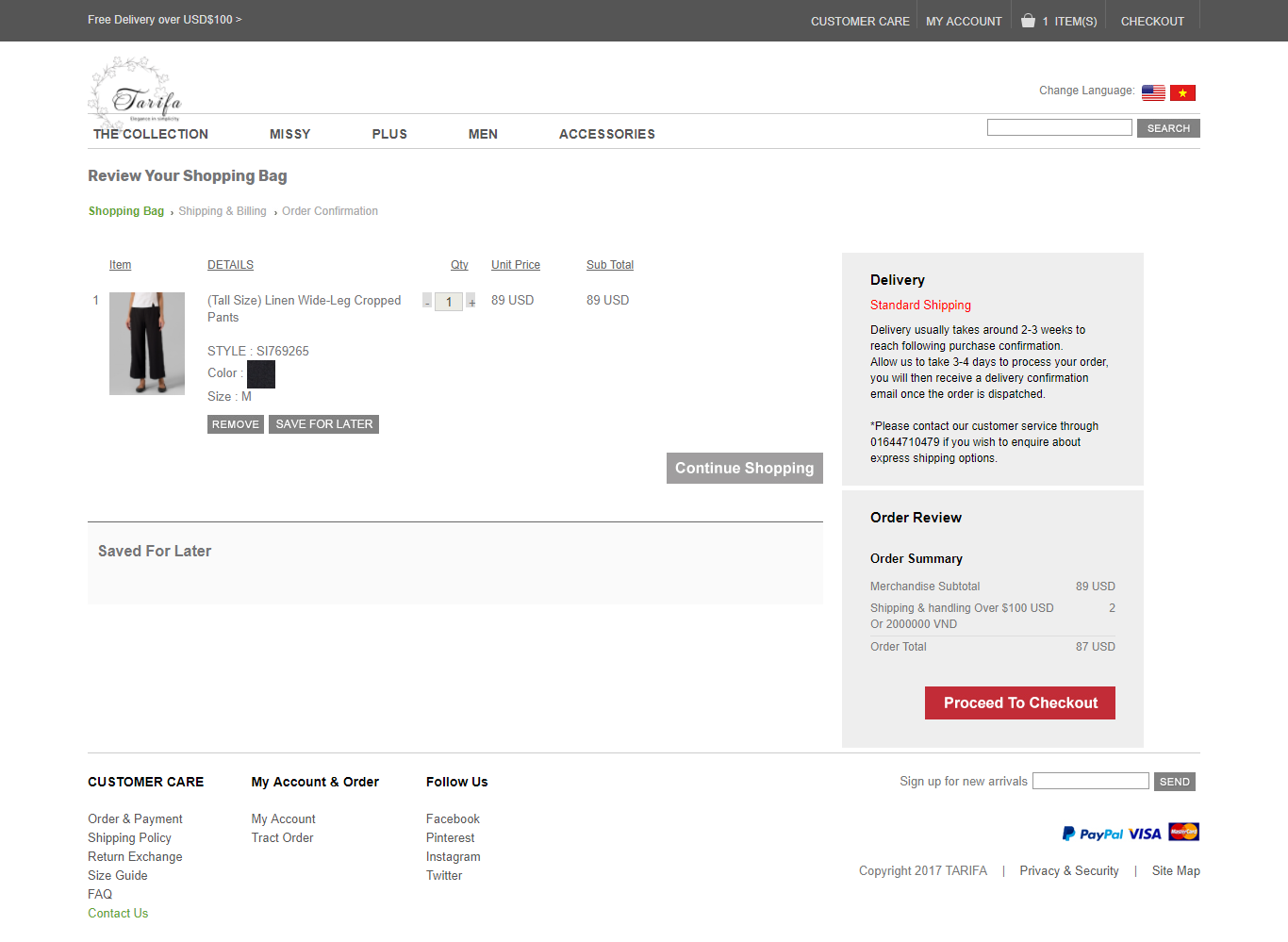
Hình .: Giao diện hiển thị trang phục đơn



Hình .: Giao diện hiển thị trang phục phối

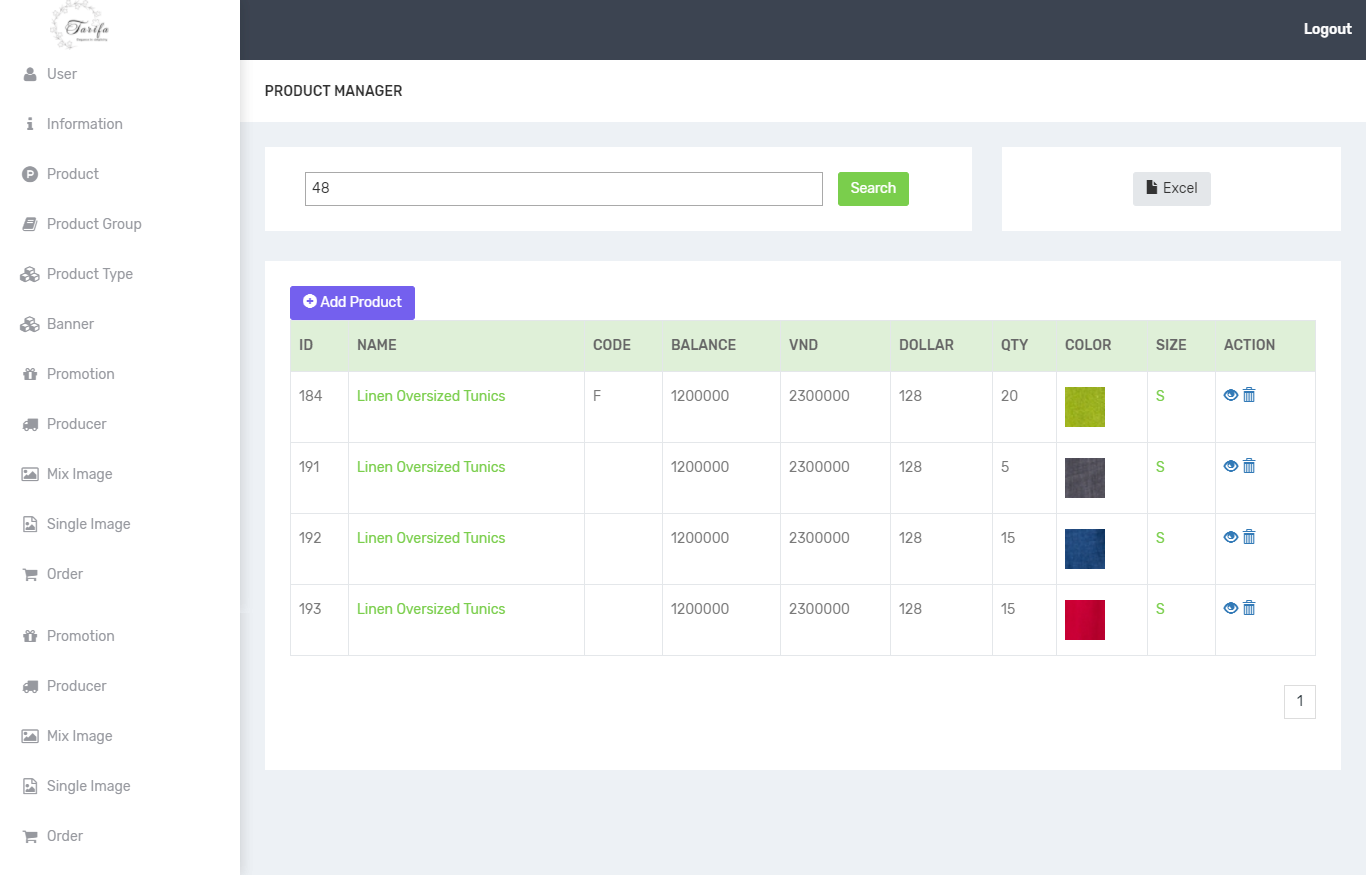


Hình .: Giao diện hiển thị thông tin chi tiết trang phục

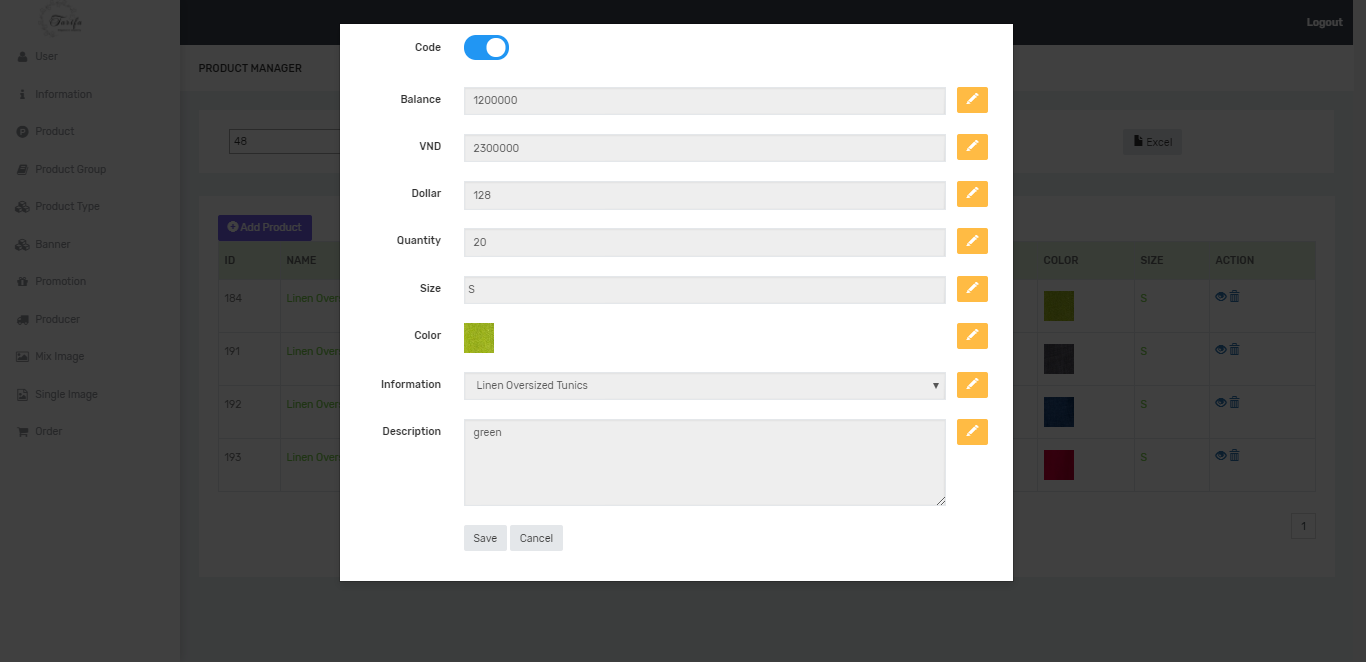


Hình .: Giao diện hiển thị giỏ hàng

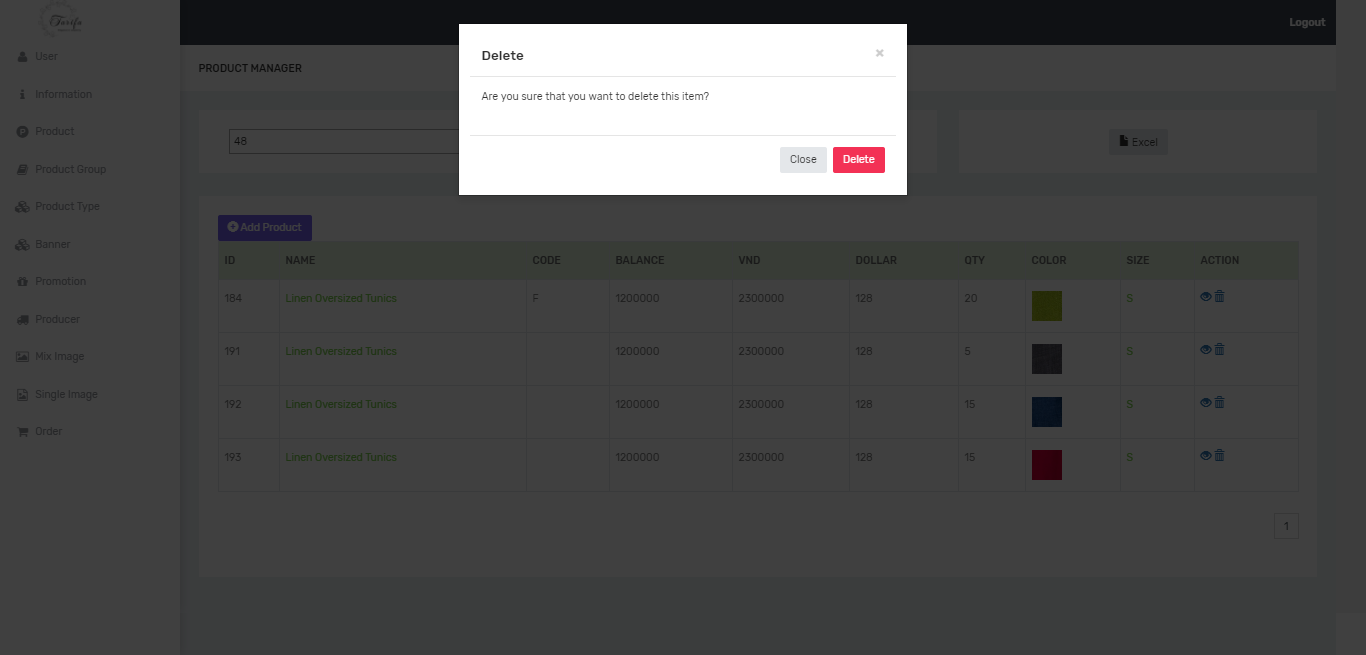
## Hệ thống back-end

* Xây dựng được hệ thống cơ sở dữ liệu hoàn chỉnh giúp truy xuất dữ liệu thuận lợi. Xây dựng thành công giải thuật lưu trữ nhiều hình ảnh và kết hợp các trang phục đơn thành một bộ đồ phối.
* Hiểu được cơ chế hoạt động và ứng dụng thành công các module của Spring framework như:
* Website sử dụng spring security để xác thực và phân quyền, định đúng vai trò, quyền hạng tương tác với hệ thống của từng người dùng. Root là tài khoản quản lý cao nhất. Cấp thấp hơn là admin thường, người dùng đã có tài khoản và cuối cùng là người dùng chưa có tài khoản.
* Spring data được áp dụng để xây dựng cơ chế kết nối giữa hệ thống và database thông qua mô hình Object Relaitional Mapping (ORM) và sử dụng câu truy vấn được định nghĩa sẵn (JPQL) để truy xuất dữ liệu .
* Spring MVC được sử dụng theo hướng restful web service cung cấp dữ liệu cho client. Hai cơ chế DI và IoC giúp ứng dụng tách biệt các thành phần nhằm tăng tính tái sử dụng code và giảm sự phụ thuộc giữa chúng.
* Hệ thống sử dụng thư viện Mapstruct giúp chuyển đổi dữ liệu từ cơ sở dữ liệu thành dữ liệu truyền qua mạng 1 cách ngắn gọn, giảm thiệu viết mã nguồn.
* Sử dụng mẫu thiết kế strategy giúp tách biệt phần hiện thực ra khỏi interface, tách các module service nhằm tái sử dụng.
* Hệ thống xây dựng thành công chức năng đa ngôn ngữ. Người dùng có thể tự chọn ngôn ngữ tiếng Việt hoặc tiếng Anh.
* Hình ảnh minh họa được trình bày bên dưới:

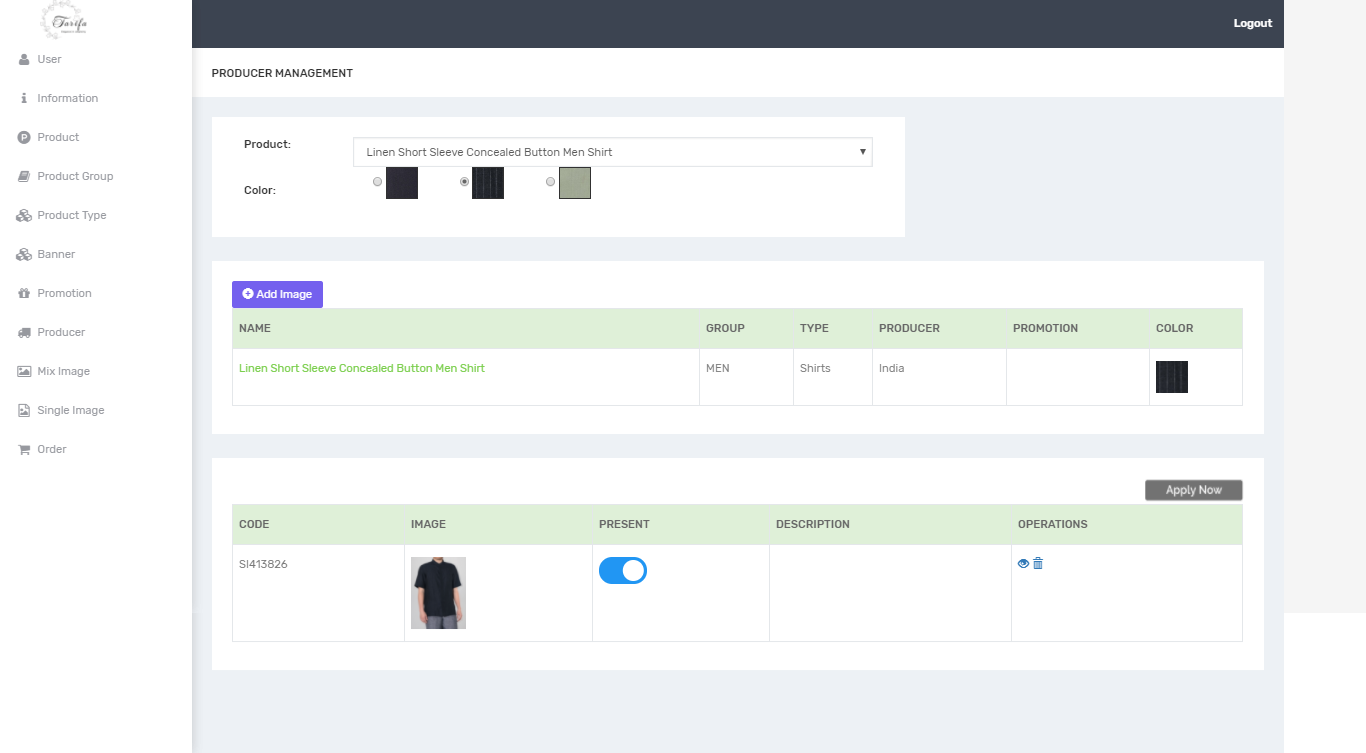
Hình .: Giao diện hiển thị thông tin trang phục



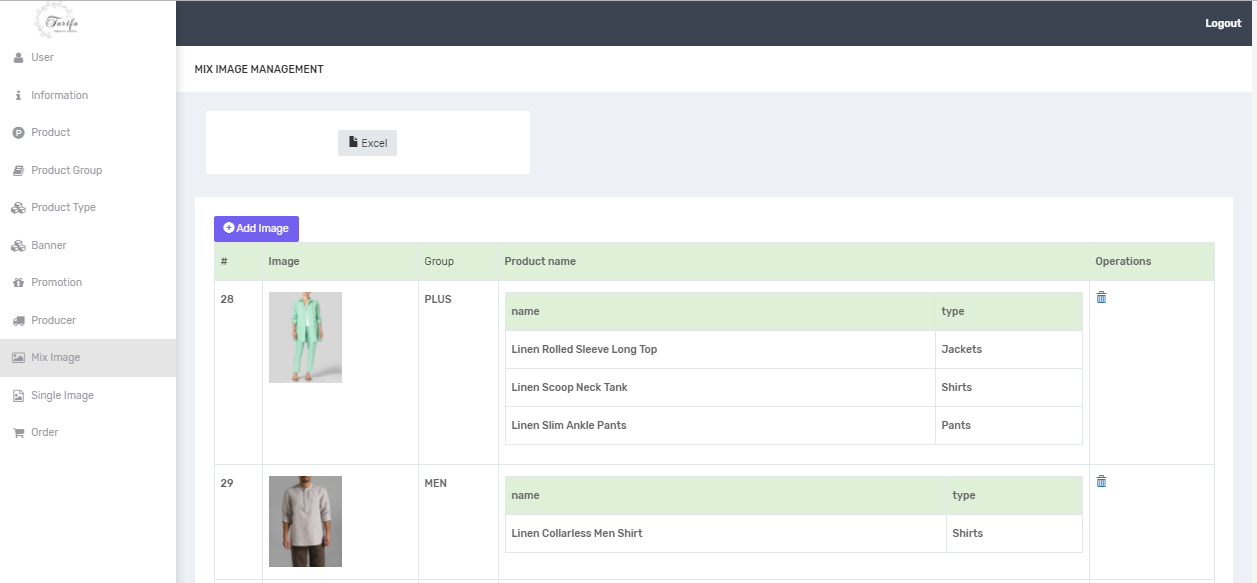
Hình .: Giao diện hiển thị thông tin chi tiết và thao tác sửa đổi thông tin trang phục



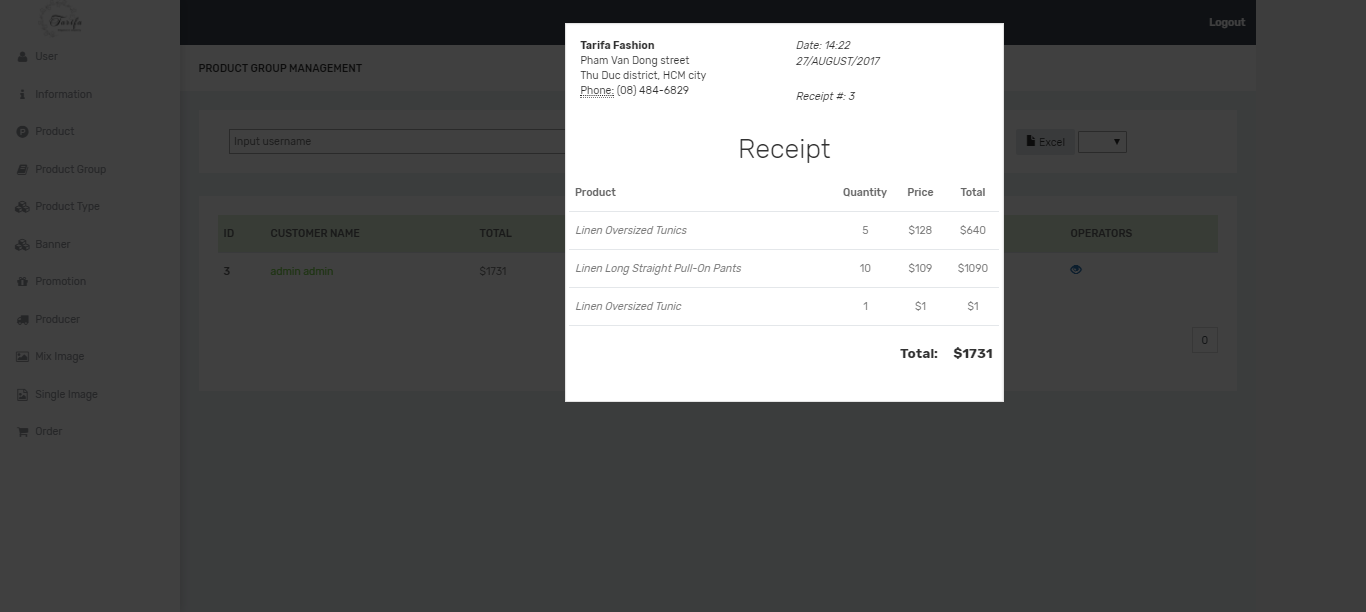
Hình .: Giao diện xác nhận việc xóa trang phục trước khi xóa



Hình .: Giao diện hiển thị trang phục đơn



Hình .: Giao diện hiển thị trang phục phối



Hình .: Giao diện hiển thị hóa đơn của đơn hàng

# Kết luận và kiến nghị

## Các kết luận

### Các hạn chế

* Chưa áp dụng responsive cho giao diện ứng dụng.
* Chưa áp dụng được các chính sách của thương mại điện tử như các chính sách chăm sóc khách hàng thân thiết cũng như các phương thức thanh toán bằng cổng thanh toán điện tử.

### Các đề xuất mới

* Áp dụng các chính sách thương mại điện tử vào website như chính sách chăm sóc khách hàng thân thiết, khách hàng VIP hay tích hợp phương thức thanh toán bằng cổng thanh toán.
* Tích hợp các responsive cho giao diện để người dùng có thể truy xuất trên các thiết bị thông minh khác.

## Các kiến nghị

Sử dụng website như một ứng dụng website kinh doanh thời trang mang phong cách vintage thật thụ. Vì ứng dụng tích hợp các chức năng bán hàng.

Bên cạnh đó, tiếp tục tìm hiểu các module còn lại của Spring framework để khai thác tối đa lợi ích Spring framework mang lại.

TÀI LIỆU THAM KHẢO

[1] The complete E-commerce Book: Design, Build, & Maintain a Successful Web-based Business, Second Edition – Janice Reynolds

[2] Ben Alex, Luke Taylo. Spring Security Reference Documentation

[3] <https://spring.io/>

[4] <http://o7planning.org/>

PHỤ LỤC

Phụ lục 1: Mô tả khác của usecase hệ thống

* **Đăng ký**

|  |  |
| --- | --- |
| Usecase | Đăng ký |
| Actor | Khách hàng |
| Description | Usecase cho phép khách hàng đăng ký tài khoản mới |
| Main flows | 1. Người đăng ký chọn chức năng đăng ký trên giao diện đăng ký thành viên. 2. Người đăng ký điền đầy đủ thông tin bắt buộc vào form đăng ký. 3. Chọn chức năng gửi dữ liệu đăng ký. 4. Hệ thống sẽ gửi lại phản hồi cho người đăng ký. Người đăng ký sẽ nhận được email đăng ký thành công. |
| Alternative flows | 1. Nếu người đăng ký đã có tài khoản thì không cần đăng ký và thực hiện chức năng đăng nhập. 2. Người đăng ký sẽ nhân được phản hồi của hệ thống nếu không điền đầy đủ các thông tin bắt buộc. |
| Pre-condition | Người đăng ký chưa có tài khoản trong hệ thống là không có tên đăng nhập hoặc email đã đăng ký. |
| Post-condition | Khách hàng có tài khoản của hệ thống để thực hiện việc đặt hàng. |

* **Đăng nhập**

|  |  |
| --- | --- |
| Usecase | Đăng nhập |
| Actor | Khách hàng, người quản trị |
| Description | Usecase cho phép khách hàng và người quản trị đăng nhập vào hệ thống. Người quản trị đăng nhập bằng tài khoản của người quản trị. Khách hàng đăng nhập vào hệ thống bằng tài khoản đã được cấp với tên đăng nhập và mật khẩu tương ứng. |
| Main flows | 1. Chọn chức năng đăng nhập 2. Điền đầy đủ username và password vào form đăng nhập 3. Nhấp vào nút submit để đăng nhập |
| Alternative flows | 1. Hệ thống sẽ hiển thị thông báo nếu nhập sai username và password. 2. Nếu khách hàng chưa có tài khoản thì chọn chức năng đăng ký. 3. Nếu người đăng nhập chưa hoàn thành các yêu cầu bắt buộc của form đăng nhập hệ thống sẽ hiển thị thông báo để người đăng nhập hoàn thành chúng. |
| Pre-condition | Đã có tài khoản trong hệ thống. Đối với quản trị viên phải biết được tài khoản quản trị của mình. |
| Post-condition | Khi người đăng nhập thực hiện thành công chức năng này, người đăng nhập sẽ thực hiện được các chức năng khác của hệ thống. |

* **Tìm kiếm trang phục**

|  |  |
| --- | --- |
| Usecase | Tìm kiếm trang phục |
| Actor | Khách hàng |
| Description | Usecase cho phép khách hàng tìm kiếm trang phục yêu thích dựa vào tên và mã trang phục. |
| Main flows | 1. Người tìm kiếm điền tên hoặc mã trang phục cần tìm và form tìm kiếm. 2. Người dùng chọn chức năng tìm kiếm. 3. Trang phục tìm kiếm được sẽ hiển thị tại trang tìm kiếm. |
| Alternative flows | Nếu người tìm kiếm không điền thông tin tìm kiếm và chọn chức năng tìm kiếm thì hệ thống sẽ hiển thị thông báo chưa hoàn thành. |
| Pre-condition |  |
| Post-condition | Người dùng sẽ thực hiện được các chức năng tiếp theo dự trên kết quả tìm kiếm được |

* **Quản lý nhóm trang phục**

|  |  |
| --- | --- |
| Usecase | Quản lý nhóm trang phục |
| Actor | Người quản trị nhóm trang phục |
| Description | Usecase cho phép người quản trị quản lý nhóm trang phục với các chức năng xem chi tiết, thêm mới, sửa đổi và xóa nhóm trang phục. |
| Main flows | 1. Người quản trị đăng nhập vào hệ thống bằng tài khoản của người quản trị. 2. Người quản trị thực hiện chức năng quản lý nhóm trang phục thông qua việc thêm, xóa, sửa nhóm trang phục. 3. Người quản trị xác nhận lại thông tin thay đổi trước khi quá trình thêm, xóa, sửa nhóm trang phục được thực hiện. 4. Hoàn tất quá trình thêm, xóa, sửa nhóm trang phục. |
| Alternative flows | Nếu việc thêm, xóa và sửa nhóm trang phục không được thực hiện thì hệ thống sẽ thông báo cho người quản trị. |
| Pre-condition | Người quản trị phải đăng nhập vào hệ thống quản lý trước khi thực hiện chức năng quản lý nhóm trang phục. |
| Post-condition | Thực hiện thành công việc thêm, xóa và sửa nhóm trang phục. |

* **Quản lý loại trang phục**

|  |  |
| --- | --- |
| Usecase | Quản lý loại trang phục |
| Actor | Người quản trị loại trang phục |
| Description | Usecase cho phép người quản trị quản lý loại trang phục với các chức năng xem chi tiết, thêm mới, sửa đổi và xóa loại trang phục. |
| Main flows | 1. Người quản trị đăng nhập vào hệ thống bằng tài khoản của người quản trị. 2. Người quản trị thực hiện chức năng quản lý loại trang phục thông qua việc thêm, xóa, sửa loại trang phục. 3. Người quản trị xác nhận lại thông tin thay đổi trước khi quá trình thêm, xóa, sửa loại trang phục được thực hiện. 4. Hoàn tất quá trình thêm, xóa, sửa loại trang phục. |
| Alternative flows | Nếu việc thêm, xóa và sửa loại trang phục không được thực hiện thì hệ thống sẽ thông báo cho người quản trị. |
| Pre-condition | Người quản trị phải đăng nhập vào hệ thống quản lý trước khi thực hiện chức năng quản lý loại trang phục. |
| Post-condition | Thực hiện thành công việc thêm, xóa và sửa loại trang phục. |

* **Quản lý khách hàng**

|  |  |
| --- | --- |
| Usecase | Quản lý khách hàng |
| Actor | Người quản trị khách hàng |
| Description | Usecase cho phép người quản trị quản lý các thông tin của khách hàng với các chức năng xem chi tiết, thêm mới, sửa đổi và khóa tài khoản khách. |
| Main flows | 1. Người quản trị đăng nhập vào hệ thống bằng tài khoản của người quản trị. 2. Người quản trị thực hiện chức năng quản lý khách hàng thông qua việc thêm, sửa và khóa tài khoản khách hàng. 3. Người quản trị xác nhận lại thông tin thay đổi trước khi quá trình thêm, sửa và khóa tài khoản khách hàng được thực hiện. 4. Hoàn tất quá trình thêm, sửa và khóa tài khoản khách hàng. |
| Alternative flows | Nếu việc thêm, sửa và khóa tài khoản khách hàng không được thực hiện thì hệ thống sẽ thông báo cho người quản trị. |
| Pre-condition | Người quản trị phải đăng nhập vào hệ thống quản lý trước khi thực hiện chức năng quản lý khách hàng. |
| Post-condition | Thực hiện thành công việc thêm, sửa và khóa tài khoản khách hàng. |

* **Quản lý trang phục**

|  |  |
| --- | --- |
| Usecase | Quản lý trang phục |
| Actor | Người quản trị trang phục |
| Description | Usecase cho phép người quản trị quản lý trang phục với các chức năng như xem chi tiết, thêm mới, sửa đổi và xóa trang phục |
| Main flows | 1. Người quản trị đăng nhập vào hệ thống bằng tài khoản của người quản trị. 2. Người quản trị thực hiện chức năng quản lý trang phục thông qua việc thêm, xóa và sửa trang phục. 3. Người quản trị xác nhận lại thông tin thay đổi trước khi quá trình thêm, xóa, và sửa trang phục được thực hiện. 4. Hoàn tất quá trình thêm, xóa và sửa trang phục. |
| Alternative flows | Nếu việc thêm, xóa và sửa trang phục không được thực hiện thì hệ thống sẽ thông báo cho người quản trị. |
| Pre-condition | Người quản trị phải đăng nhập vào hệ thống quản lý trước khi thực hiện chức năng quản lý trang phục. |
| Post-condition | Thực hiện thành công việc thêm, xóa và sửa trang phục. |

* **Quản lý khuyến mãi**

|  |  |
| --- | --- |
| Usecase | Quản lý khuyến mãi |
| Actor | Người quản trị khuyến mãi |
| Description | Usecase cho phép người quản trị quản lý các khuyến mãi với các chức năng như xem chi tiết, thêm mới, sửa đổi và xóa khuyến mãi. |
| Main flows | 1. Người quản trị đăng nhập vào hệ thống bằng tài khoản của người quản trị. 2. Người quản trị thực hiện chức năng quản lý khuyến mãi thông qua việc thêm, xóa và sửa khuyến mãi. 3. Người quản trị xác nhận lại thông tin thay đổi trước khi quá trình thêm, xóa, và sửa khuyến mãi được thực hiện. 4. Hoàn tất quá trình thêm, xóa và sửa khuyến mãi. |
| Alternative flows | Nếu việc thêm, xóa và sửa khuyến mãi không được thực hiện thì hệ thống sẽ thông báo cho người quản trị. |
| Pre-condition | Người quản trị phải đăng nhập vào hệ thống quản lý trước khi thực hiện chức năng quản lý khuyến mãi. |
| Post-condition | Thực hiện thành công việc thêm, xóa và sửa khuyến mãi. |

* **Quản lý nhà sản xuất**

|  |  |
| --- | --- |
| Usecase | Quản lý nhà sản xuất |
| Actor | Người quản trị nhà sản xuất |
| Description | Usecase cho phép người quản trị quản lý nhà sản xuất với các chức năng như xem chi tiết, thêm mới, sửa đổi và xóa nhà sản xuất. |
| Main flows | 1. Người quản trị đăng nhập vào hệ thống bằng tài khoản của người quản trị. 2. Người quản trị thực hiện chức năng quản lý nhà sản xuất thông qua việc thêm, xóa và sửa nhà sản xuất. 3. Người quản trị xác nhận lại thông tin thay đổi trước khi quá trình thêm, xóa, và sửa nhà sản xuất được thực hiện. 4. Hoàn tất quá trình thêm, xóa và sửa nhà sản xuất. |
| Alternative flows | Nếu việc thêm, xóa và sửa nhà sản xuất không được thực hiện thì hệ thống sẽ thông báo cho người quản trị. |
| Pre-condition | Người quản trị phải đăng nhập vào hệ thống quản lý trước khi thực hiện chức năng quản lý nhà sản xuất. |
| Post-condition | Thực hiện thành công việc thêm, xóa và sửa nhà sản xuất. |

* **Quản lý đơn hàng (dành cho admin thường)**

|  |  |
| --- | --- |
| Usecase | Quản lý đơn hàng |
| Actor | Người quản trị đơn hàng |
| Description | Usecase cho phép người quản trị quản lý đơn hàng thực hiện chức năng xem thông tin chi tiết đơn hàng. |
| Main flows | 1. Người quản trị đăng nhập vào hệ thống bằng tài khoản của người quản trị. 2. Người quản trị thực hiện chức năng quản lý đơn hàng thông qua việc xem đơn hàng. |
| Alternative flows |  |
| Pre-condition | Người quản trị phải đăng nhập vào hệ thống quản lý trước khi thực hiện chức năng quản lý đơn hàng. |
| Post-condition | Thực hiện thành công việc xem đơn hàng. |

* **Quản lý đơn hàng (dành cho người quản lý là root)**

|  |  |
| --- | --- |
| Usecase | Quản lý đơn hàng |
| Actor | Root |
| Description | Usecase cho phép người quản trị quản lý đơn hàng với các chức năng như xem chi tiết, thêm mới, sửa đổi và xóa đơn hàng. |
| Main flows | 1. Người quản trị đăng nhập vào hệ thống bằng tài khoản của người quản trị. 2. Người quản trị thực hiện chức năng quản lý đơn hàng thông qua việc thêm, xóa và sửa đơn hàng. 3. Người quản trị xác nhận lại thông tin thay đổi trước khi quá trình thêm, xóa, và sửa đơn hàng được thực hiện. 4. Hoàn tất quá trình thêm, xóa và sửa đơn hàng. |
| Alternative flows | Nếu việc thêm, xóa và sửa đơn hàng không được thực hiện thì hệ thống sẽ thông báo cho người quản trị. |
| Pre-condition | Người quản trị phải đăng nhập vào hệ thống quản lý trước khi thực hiện chức năng quản lý đơn hàng. |
| Post-condition | Thực hiện thành công việc thêm, xóa và sửa đơn hàng. |

Phụ lục 2: Mô tả chi tiết class diagram

* **Bảng Group**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Định danh |
| nameUS | String | Tên nhóm trang phục bằng tiếng Anh |
| nameVN | String | Tên nhóm trang phục bằng tiếng Việt |
| descriptionUS | String | Mô tả chi tiết nhóm trang phục bằng tiếng Anh |
| descriptionVN | String | Mô tả chi tiết nhóm trang phục bằng tiếng Việt |
| createdDate | LocalDateTime | Ngày tạo nhóm trang phục |
| enabled | Boolean | True là hiển thị, False là không hiển thị |

* **Bảng Type**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Định danh |
| nameUS | String | Tên loại trang phục bằng tiếng Anh |
| nameVN | String | Tên loại trang phục bằng tiếng Việt |
| descriptionUS | String | Mô tả chi tiết loại trang phục bằng tiếng Anh |
| descriptionVN | String | Mô tả chi tiết loại trang phục bằng tiếng Việt |
| createdDate | LocalDateTime | Ngày tạo loại trang phục |
| enabled | Boolean | True là hiển thị, False là không hiển thị |

* **Bảng Producer**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Định danh |
| nameUS | String | Tên nhà sản xuất bằng tiếng Anh |
| nameVN | String | Tên nhà sản xuất bằng tiếng Việt |
| Cooperation | Boolean | True là còn hợp tác, False là đã dừng hợp tác |
| descriptionUS | String | Mô tả về nhà sản xuất bằng tiếng Anh |
| descriptionVN | String | Mô tả về nhà sản xuất bằng tiếng Việt |
| createdDate | LocalDateTime | Ngày tạo thông tin nhà sản xuất |
| enabled | Boolean | True là hiển thị False là không hiển thị |

* **Bảng Promotion**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Định danh |
| nameUS | String | Tên chương trình khuyến mãi bằng tiếng Anh |
| nameVN | String | Tên chương trình khuyến mãi bằng tiếng Việt |
| startDate | LocalDateTime | Ngày bắt đầu khuyến mãi |
| endDate | LocalDateTime | Ngày kết thúc khuyến mãi |
| descriptionUS | String | Mô tả chương trình khuyến mãi bằng tiếng Anh |
| descriptionVN | String | Mô tả chương trình khuyến mãi bằng tiếng Việt |
| createdDate | LocalDateTime | Ngày tạo thông tin chương trình khuyến mãi |
| enabled | Boolean | True là hiển thị False là không hiển thị |

* **Bảng Authority**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Định danh |
| name | String | Tên quyền người dùng |
| descriptionUS | String | Mô tả quyền người dùng bằng tiếng Anh |
| descriptionVN | String | Mô tả quyền người dùng bằng tiếng Việt |
| createdDate | LocalDateTime | Ngày tạo quyền người dùng |
| enabled | Boolean | True là hiển thị False là không hiển thị |

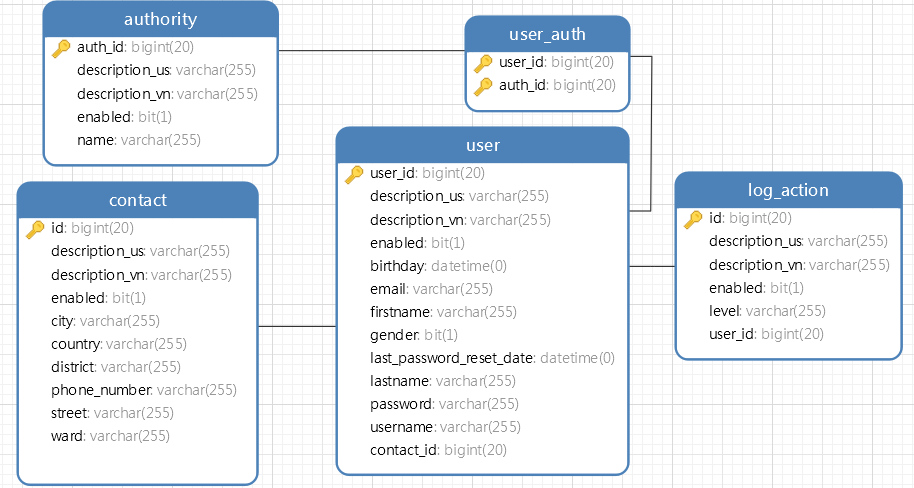
* **Bảng User**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Định danh |
| birthday | LocalDateTime | Ngày sinh người dùng |
| email | String | Địa chỉ email |
| username | String | Tên tài khoản |
| password | String | Mật khẩu |
| firstname | String | Họ và tên lót người dùng |
| lastname | String | Tên người dùng |
| contact | Contact | Địa chỉ liên lạc |
| descriptionVN | String | Mô tả quyền người dùng bằng tiếng Việt |
| createdDate | LocalDateTime | Ngày tạo quyền người dùng |
| enabled | Boolean | True là hiển thị False là không hiển thị |

* **Bảng Order**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Định danh |
| delivery | Boolean | 0 là chưa giao hàng, 1 đã giao hàng |
| lastUpdate | LocalDateTime | Thời gian cập nhật đơn hàng lần cuối |
| shippingAddress | Address | Địa chỉ giao hàng |
| status | String | Trạng thái đơn hàng |
| total | Long | Giá tổng cộng của đơn hàng |
| user | User | Người đặt hàng |
| descriptionUS | String | Mô tả đơn hàng bằng tiếng Anh |
| descriptionVN | String | Mô tả đơn hàng bằng tiếng Việt |
| createdDate | LocalDateTime | Ngày tạo đơn hàng |
| enabled | Boolean | True là hiển thị False là không hiển thị |

Phụ lục 3: Mô tả chi tiết cơ sở dữ liệu

* **Chức năng dành cho người dung**
* **Bảng user**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| birthday | datetime | Ngày tháng năm sinh |
| email | varchar | Địa chỉ email |
| firstname | varchar | Họ và tên lót |
| Gender | bit | 1 là nữ, 0 là nam |
| last\_password\_reset\_date | datetime | Thời gian reset password lần cuối |
| lastname | varchar | Tên |
| password | varchar | Mật khẩu |
| username | varchar | Tên tài khoản |
| contact\_id | bigint | Đinh danh contact |

* **Bảng authority**

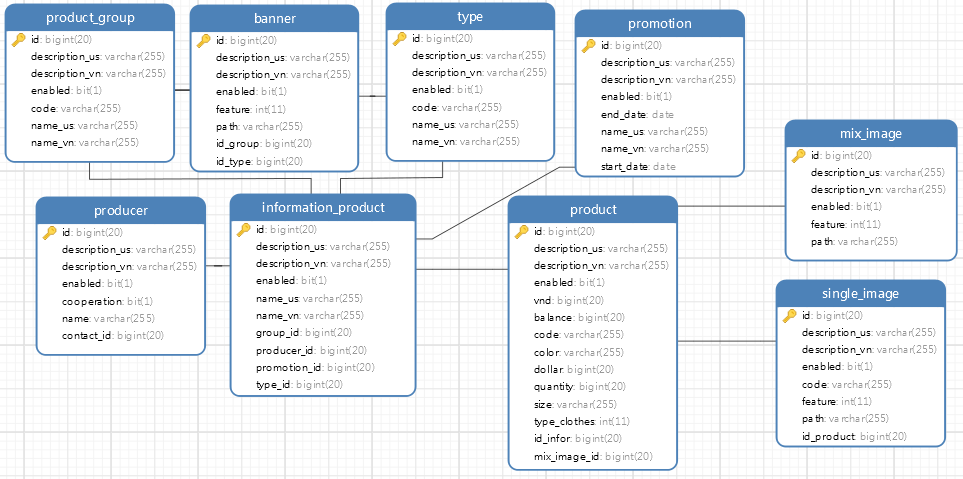
|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| Name | Varchar | Tên |

* **Bảng log\_action**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| level | varchar |  |
| user\_id | bigint | Định danh user |

* **Bảng contact**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| city | varchar | Tê thành phố |
| country | varchar | Tên quốc gia |
| district | varchar | Tên quận |
| phone\_number | varchar | Số điện thoại |
| street | varchar | Tên đường |
| ward | varchar | Tên phường |

* **Chức năng hiển thị trang phục**
* **Bảng product\_group**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| code | varchar | Mã cho từng nhóm sản phẩm |
| name\_us | varchar | Tên sản phẩm bằng tiếng Anh |
| name\_vn | varchar | Tên sản phẩm bằng tiếng Việt |

* **Bảng type**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| code | varchar | Mã cho từng nhóm sản phẩm |
| name\_us | varchar | Tên sản phẩm bằng tiếng Anh |
| name\_vn | varchar | Tên sản phẩm bằng tiếng Việt |

* **Bảng banner**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| feature | int | Vị trí hiện thỉ của banner |
| path | varchar | Đường dẫn tới banner |
| id\_group | bigint | Định danh của group |
| id\_type | bigint | Định danh của type |

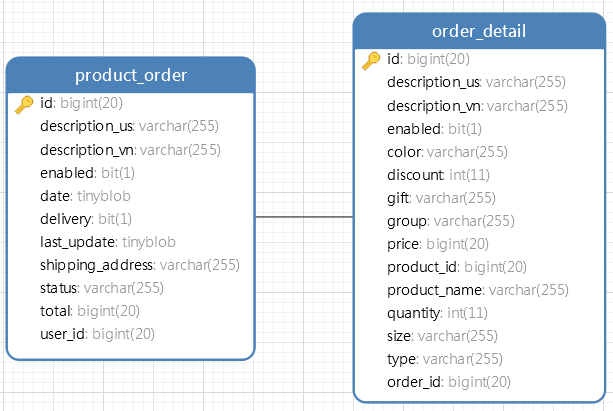
* **Bảng promotion**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| end\_date | date | Ngày kết thúc khuyến mãi |
| name\_us | varchar | Tên sản phẩm bằng tiếng Anh |
| name\_vn | varchar | Tên sản phẩm bằng tiếng Việt |
| start\_date | date | Ngày bắt đầu khuyến mãi |

* **Bảng producer**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| cooperation | bit | 0 là hết hợp tác, 1 là còn hợp tác |
| name | varchar | Tên nhà sản xuất |
| contact\_id | bigint | Định danh contact |

* **Chức năng đặt hang**



* **Bảng order**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| date | tinyblob | Thời gian đặt hàng |
| delivery | bit | 0 là chưa giao hàng, 1 là đã giao hàng |
| last\_update | tinyblob | Lần cuối cập nhật đơn hàng |
| shipping\_address | varchar | Địa chỉ giao hàng |
| status | varchar | Trạng thái đơn hàng |
| total | bigint | Tổng giá đơn hàng |
| user\_id | bigint | Định danh user |

* **Bảng order\_detail**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | bigint | Định danh |
| description\_us | varchar | Nội dung thông tin bằng tiếng Anh |
| description\_vn | varchar | Nội dung thông tin bằng tiếng Việt |
| enabled | bit | 0 là ẩn, 1 là hiện |
| color | varchar | Màu sắc sản phẩm |
| gift | varchar | Quà tặng |
| group | varchar | Nhóm sản phẩm |
| price | bigint | Giá sản phẩm |
| product\_id | bigint | Định danh sản phẩm |
| product\_name | varchar | Tên sản phẩm |
| quantity | int | Số lượng |
| size | varchar | Size |
| type | varchar | Loại sản phẩm |
| order\_id | bigint | Định danh product\_order |