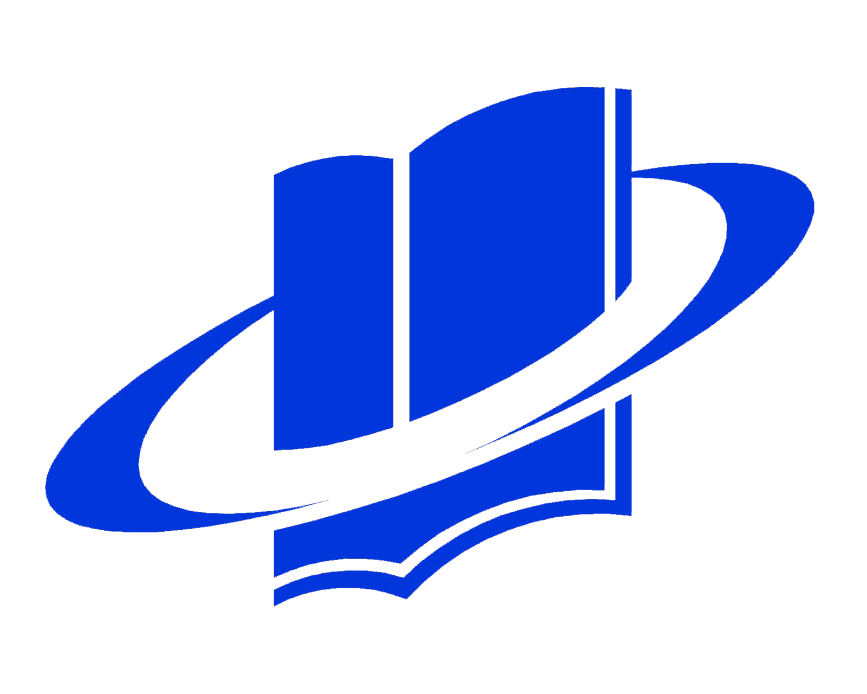
**TRƯỜNG ĐẠI HỌC MỞ TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙢🙡🕮🙣🙠**



**ĐỒ ÁN MÔN HỌC**

**Đề tài: Website bán hàng thời trang trực tuyến dựa trên nền tảng ASP.NET**

Sinh viên thực hiện: Trần Duy

MSSV: 1551010019

Giảng Viên Hướng Dẫn: ThS. Lê Viết Tuấn

Mục lục

[1. Tổng quát 5](#_Toc21964455)

[1.1. Đặt vấn đề 5](#_Toc21964456)

[1.2. Mục đích đề tài 5](#_Toc21964457)

[1.3. Đối tượng 5](#_Toc21964458)

[1.4. Mục đích nghiên cứu 6](#_Toc21964459)

[2. Cơ sở lý thuyết 6](#_Toc21964460)

[2.1. Tổng quan về ASP.NET 6](#_Toc21964461)

[2.2. Tổng quan về ASP.NET MVC 7](#_Toc21964462)

[2.2.1. Nền tảng MVC bao gồm các thành phần dưới đây: 7](#_Toc21964463)

[2.2.2. Nền tảng ASP.NET MVC mang lại những lợi ích sau: 9](#_Toc21964464)

[2.2.3. Các tính năng của nền tảng ASP.NET MVC 9](#_Toc21964465)

[2.3. Mô hình 3 Lớp 10](#_Toc21964466)

[2.3.1. Giới thiệu 10](#_Toc21964467)

[2.3.2. Mô hình 3 lớp gồm 3 thành phần chính được tách rời gồm: 10](#_Toc21964468)

[2.4. Repository Pattern 10](#_Toc21964469)

[2.5. Unit of Work trong Repository Pattern 11](#_Toc21964470)

[2.6. Service Layer 11](#_Toc21964471)

[2.7. Entity Framework 11](#_Toc21964472)

[2.7.1. Giới thiệu 11](#_Toc21964473)

[2.7.2. Entity Framework trong dự án Web ASP.NET MVC 12](#_Toc21964474)

[2.7.3. Cài đặt Entity Framework 12](#_Toc21964475)

[2.7.4. Code First Migrations 14](#_Toc21964476)

[2.7.5. Bật Migrations 14](#_Toc21964477)

[2.7.6. Tạo các migrations và áp dụng các thay đổi vào cơ sở dữ liệu. 15](#_Toc21964478)

[2.7.7. Phương thức Seed 15](#_Toc21964479)

[2.8. ASP.NET Identity 16](#_Toc21964480)

[2.8.1. Mục đích 16](#_Toc21964481)

[2.9. Dependency Injection 16](#_Toc21964482)

[2.10. Inversion of Controller container 16](#_Toc21964483)

[2.11. Autofac 17](#_Toc21964484)

[2.11.1. Đăng ký thanh phần, các lớp và phụ thuộc của nó 17](#_Toc21964485)

[2.11.2. Đăng ký các thanh phần của Autofac 18](#_Toc21964486)

[2.11.3. Phân giải các phụ thuộc của một lớp bằng phương thức khởi tạo 18](#_Toc21964487)

[3. Kết quả thực hiện 19](#_Toc21964488)

[3.1. Kết quả sử dụng ASP.NET MVC 5 19](#_Toc21964489)

[3.1.1. Cấu trúc sử dụng ASP.NET MVC 5 19](#_Toc21964490)

[3.1.2. Hiển thị sản phẩm theo danh sách 21](#_Toc21964491)

[3.1.3. Hiện thị sản phẩm theo collection 23](#_Toc21964492)

[3.1.4. Hiển thị chi tiết một sản phẩm 24](#_Toc21964493)

[3.1.5. Hiển thị giỏ hàng trên thanh menu 27](#_Toc21964494)

[3.1.6. Hiển thị trang giỏ hàng 28](#_Toc21964495)

[3.2. Kết quả sử dụng mô hình 3 lớp vào đề tài 28](#_Toc21964496)

[3.2.1. Cấu trúc được 3 lớp được sử dụng trong đề tài 29](#_Toc21964497)

[3.2.2. Áp dụng vào đề tài 29](#_Toc21964498)

[3.2.3. Đánh giá 30](#_Toc21964499)

[3.3. Kết quả sử dụng Reponsitory Pattern 30](#_Toc21964500)

[3.3.1. Áp dụng vào đề tài 30](#_Toc21964501)

[3.3.2. Đánh giá 31](#_Toc21964502)

[3.4. Kết quả sử dụng UnitOfWork 31](#_Toc21964503)

[3.4.1. Áp dụng vào đề tài 31](#_Toc21964504)

[3.5. Kết quả sử dụng Service Layer 32](#_Toc21964505)

[3.5.1. Áp dụng vào đề tài 32](#_Toc21964506)

[3.5.2. Đánh giá 32](#_Toc21964507)

[3.6. Kết quả sử dụng Entity Framework 33](#_Toc21964508)

[3.6.1. Áp dụng vào đề tài 33](#_Toc21964509)

[3.6.2. Đánh giá 34](#_Toc21964510)

[3.7. Kết quả sử dụng Identity 35](#_Toc21964511)

[3.7.1. Áp dụng vào đề tài 35](#_Toc21964512)

[3.7.2. Chức năng đăng ký 36](#_Toc21964513)

[3.7.3. Chức năng đăng nhập 38](#_Toc21964514)

[3.7.4. Chức năng thay đổi thông tin người dùng 40](#_Toc21964515)

[3.8. Autofac 41](#_Toc21964516)

[3.8.1. Áp dụng vào đề tài 41](#_Toc21964517)

[3.8.2. Đăng ký các lớp 42](#_Toc21964518)

[4. Kết luận 42](#_Toc21964519)

[5. Tài liệu tham khảo 43](#_Toc21964520)

Lời cảm ơn

Báo cáo được hoàn thanh vào tháng 10 năm 2019 với sự hướng dẫn của thầy ThS. Lê Viết Tuấn. Em xin bày tỏ lòng biết ơn với thầy, người đã hướng dẫn cho em thực hiện đề tài và báo cáo này. Cám ơn các thầy cô giáo Khoa Công nghệ thông tin trường Ðại học Mở Tp. Hồ Chí Minh tạo giảng dạy, tạo điều kiện để chúng em được học tập những kiến thức liên quan đến đề tài này.

*Em xin chân thành cám ơn*

# Tổng quát

## Đặt vấn đề

Khoa học càng ngày càng phát triển, tốc độ phát triển của mạng internet, sự phổ cập của các thiết bị thông minh. Mọi người dễ dàng hơn trong việc truy cập vào các tài nguyên trên mạng máy tính. Cùng với sự tiện lợi của việc kinh doanh trực tuyến, các website bán hàng trở nên phổ biết hơn, các dịch vụ hỗ trợ như dịch vụ giao hàng, dịch vụ quảng cáo theo từ khóa. Vậy nên nhu cầu xây dựng một website là rất cần thiết với một doanh nghiệp kinh doanh.

ASP.NET là một nền tảng ứng dụng web (web application framework) được phát triển và cung cấp bởi Microsoft, cho phép những người lập trình tạo ra những trang web động, những ứng dụng web và những dịch vụ web. ASP.NET được tích hợp nhiều công nghệ như Entity Framework, LINQ để tương tác với cơ sở dữ liệu.

Do được hỗ trợ bởi Microsoft và nó thân thiên hơn với người dùng Windows, cũng sự phổ biến của Windows ở thị trường Việt Nam nên em quyết định chọn để tài xây dựng website bán hàng thời trang trực tuyến dựa trên nền tảng ASP.NET.

## Mục đích đề tài

Xây dựng một website bán hàng để đơn giản hóa việc kinh doanh, nhiều doanh nghiệp lựa chọn giải pháp xậy dựng website bán hàng vì nó mang nhiều lợi ích như: giảm thiểu chi phí mặt bằng trưng bày, đơn giản hóa việc quản lý kho, quản lý thu nhập, quản lý việc kinh doanh và để khách hàng dễ dàng tiếp cận mặt hàng, dễ mua hàng mà không bị rào cản về khoản cách đia lý.

Khách hàng có thể xem, tìm kiếm sản phẩm, đặt hàng trực tuyến, đăng nhập, đăng ký tải khoản phục vụ cho việc đặt hàng, thanh toán.

Quản trị viên có thể quản lý sản phẩm theo thuộc tinh, có thể quản lý người dùng, quản lý đơn hàng.

## Đối tượng

Đối tượng có nhu cầu xây dựng một webiste phần lớn là doanh nghiệp nhỏ có nhu cầu phát triển theo hướng hiện đại hóa để đáp ứng cách mạng công nghiệp 4.0 sắp tới. Ngoài ra doanh nghiệp vừa và lớn đã có thời gian dài kinh doanh theo cách truyền thống cũng có nhu xậy dựng website để mở rộng kinh doanh, tăng hiệu xuất kinh doanh, tăng lợi nhuận.

## Mục đích nghiên cứu

Để tìm hiểu kiến thức để xậy dựng một trang web hoàn chỉnh

Học được nhiều kiến thức mới.

Tìm hiểu về công nghệ hỗ trợ như HTML, CSS, Jquery.

Tìm hiểu các tương tác với cơ sở dữ liệu thông qua những công nghệ hữu ích như Entity Framework, LINQ.

Tìm hiểu các xử lý dữ liệu, các luận lý trong việc xử lý

Phạm vi nghiên cứu

Phạm vị nghiên cứu là các công nghệ cần thiết để xây dựng một website bán hàng trực tuyến dựa trên nền tảng ASP.NET cũng như các biện pháp để tăng hiệu xuất hoạt động của website.

# Cơ sở lý thuyết

## Tổng quan về ASP.NET

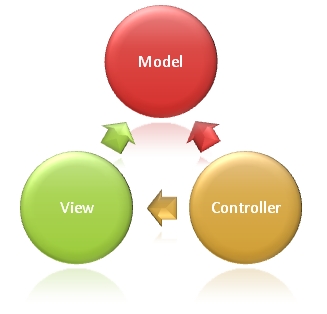
Đầu năm 2002, Microsoft giới thiệu một kỹ thuật lập trình Web khá mới mẻ với tên gọi ban đầu là ASP+, tên chính thức sau này là ASP.Net. Với ASP.Net, không những không cần đòi hỏi bạn phải biết các tag HTML, thiết kế web, mà nó còn hỗ trợ mạnh lập trình hướng đối tượng trong quá trình xây dựng và phát triển ứng dụng Web. ASP.Net là kỹ thuật lập trình và phát triển ứng dụng web ở phai Server (Server-side) dựa trên nền tảng của Microsoft .Net Framework. Hầu hết những người mới đến với lập trình web đều bắt đầu tìm hiểu những kỹ thuật ở phía Client (Client-side) như: HTML, Java Script, CSS (Cascading Style Sheets). Khi Web browser yêu cầu một trang web (trang web sử dụng kỹ thuật clientside), Web server tìm trang web mà Client yêu cầu, sau đó gởi về cho Client. Client nhận kết quả trả về từ Server và hiển thị lên màn hình. ASP.Net sử dụng kỹ thuật lập trình ở phía server thì hoàn toàn khác, mã lệnh ở phía server (ví dụ: mã lệnh trong trang ASP) sẽ được biên dịch và thi hành tại Web Server. Sau khi được Server đọc, biên dịch và thi hành, kết quả tự động được chuyển sang HTML/JavaScript/CSS và trả về cho Client. Tất cả các xử lý lệnh ASP.Net đều được thực hiện tại Server và do đó, gọi là kỹ thuật lập trình ở phía server. ASP.NET được Microsoft phát triển qua nhiều phiên bản từ ASP.NET 1.0, 1.1, 2.0 và gần đây nhất là phiên bản ASP.NET 5.

## Tổng quan về ASP.NET MVC

Mẫu kiến trúc Model – View – Controller được sử dụng nhằm chi ứng dụng thành ba thành phần chính: model, view và controller. Nền tảng ASP.NET MVC giúp cho chúng ta có thể tạo được các ứng dụng web áp dụng mô hình MVC thay vì tạo ứng dụng theo mẫu ASP.NET Web Forsm. Nền tảng ASP.NET MVC có đặc điểm nổi bật là nhẹ (lighweigt), dễ kiểm thử phần giao diện (so với ứng dụng Web Forms), tích hợp các tính năng có sẵn của ASP.NET. Nền tảng ASP.NET MVC được định nghĩa trong namespace System.Web.Mvc và là một phần của name space System.Web.

MVC là một mẫu thiết kế (design pattern) chuẩn mà nhiều lập trình viên đã quen thuộc. Một số loại ứng dụng web sẽ thích hợp với kiến trúc MVC. Một số khác vẫn thích hợp với ASP.NET Web Forms và cơ chế postbacks. Đôi khi có những ứng dụng kết hợp cả hai kiến trúc trên.

### Nền tảng MVC bao gồm các thành phần dưới đây:



#### Models: Các đối tượng Models là một phần của ứng dụng, các đối tượng này thiết lập logic của phần dữ liệu của ứng dụng. Thông thường, các đối tượng model lấy và lưu trạng thái của model trong CSDL. Ví dụ như, một đối tượng Product (sản phẩm) sẽ lấy dữ liệu từ CSDL, thao tác trên dữ liệu và sẽ cập nhật dữ liệu trở lại vào bảng Products ở SQL Server.

#### Views: Views là các thành phần dùng để hiển thị giao diện người dùng (UI). Thông thường, view được tạo dựa vào thông tin dữ liệu model. Ví dụ như, view dùng để cập nhật bảng Products sẽ hiển thị các hộp văn bản, drop-down list, và các check box dựa trên trạng thái hiện tại của một đối tượng Product.

#### Controllers: Controller là các thành phần dùng để quản lý tương tác người dùng, làm việc với model và chọn view để hiển thị giao diện người dùng. Trong một ứng dụng MVC, view chỉ được dùng để hiển thị thông tin, controller chịu trách nhiệm quản lý và đáp trả nội dung người dùng nhập và tương tác với người dùng. Ví dụ, controller sẽ quản lý các dữ liệu người dùng gởi lên (query-string values) và gởi các giá trị đó đến model, model sẽ lấy dữ liệu từ CSDL nhờ vào các giá trị này.

Mẫu MVC giúp tạo được các ứng dụng mà chúng tách rời các thành phần của ứng dụng (logic về nhập liệu, logic xử lý tác vụ và logic về giao diện). MVC chỉ ra mỗi loại thành phần kể trên nên được thiếp lập ở đâu trên ứng dụng. Logic giao diện (UI logic) thuộc về views. Logic nhập liệu (input logic) thuộc về controller. Và logic tác vụ (Business logic – là logic xử lý thông tin, mục đích chính của ứng dụng) thuộc về model. Sự phân chia này giúp bạn giảm bớt được sự phức tạp của ứng dụng và chỉ tập trung vào mỗi khía cạnh cần được cài đặt ở mỗi thời điểm. Ví dụ như bạn chỉ cần tập trung vào giao diện (views) mà không phải quan tâm đến logic xử lý thông tin của ứng dụng.

### Nền tảng ASP.NET MVC mang lại những lợi ích sau:

#### Dễ dàng quản lý sự phức tạp của ứng dụng bằng cách chia ứng dụng thành ba thành phần model, view, controller

#### Nó không sử dụng view state hoặc server-based form. Điều này tốt cho những lập trình viên muốn quản lý hết các khía cạnh của một ứng dụng.

#### Nó sử dụng mẫu Front Controller, mẫu này giúp quản lý các requests (yêu cầu) chỉ thông qua một Controller. Nhờ đó bạn có thể thiết kế một hạ tầng quản lý định tuyến. Để có nhiều thông tin hơn, bạn nên xem phần Front Controller trên web site MSDN

#### Hỗ trợ tốt hơn cho mô hình phát triển ứng dụng hướng kiểm thử (TDD)

#### Nó hỗ trợ tốt cho các ứng dụng được xây dựng bởi những đội có nhiều lập trình viên và thiết kế mà vẫn quản lý được tính năng của ứng dụng.

### Các tính năng của nền tảng ASP.NET MVC

Tách bạch các tác vụ của ứng dụng (logic nhập liệu, business logic, và logic giao diện), dễ dàng kiểm thử và mặc định áp dụng hướng phát triển TDD. Tất cả các tính năng chính của mô hình MVC được cài đặt dựa trên interface và được kiểm thử bằng cách sử dụng các đối tượng mocks, mock object là các đối tượng mô phỏng các tính năng của những đối tượng thực sự trong ứng dụng. Bạn có thể kiểm thử unit-test cho ứng dụng mà không cần chạy controller trong tiến trình ASP.NET, và điều đó giúp unit test được áp dụng nhanh chóng và tiện dụng. Bạn có thể sử dụng bất kỳ nền tảng unit-testing nào tương thích với nền tảng .NET.

MVC là một nền tảng khả mở rộng (extensible) & khả nhúng (pluggable). Các thành phần của ASP.NET MVC được thiết kể để chúng có thể được thay thế một cách dễ dàng hoặc dễ dàng tùy chỉnh. Bạn có thể nhúng thêm view engine, cơ chế định tuyến cho URL, cách kết xuất tham số của action-method và các thành phần khác. ASP.NET MVC cũng hỗ trợ việc sử dụng Dependency Injection (DI) và Inversion of Control (IoC). DI cho phép bạn gắn các đối tượng vào một lớp cho lớp đó sử dụng thay vì buộc lớp đó phải tự mình khởi tạo các đối tượng. IoC quy định rằng, nếu một đối tượng yêu cầu một đối tượng khác, đối tượng đầu sẽ lấy đối tượng thứ hai từ một nguồn bên ngoài, ví dụ như từ tập tin cấu hình. Và nhờ vậy, việc sử dụng DI và IoC sẽ giúp kiểm thử dễ dàng hơn.

ASP.NET MVC có thành phần ánh xạ URL mạnh mẽ cho phép bạn xây dựng những ứng dụng có các địa chỉ URL xúc tích và dễ tìm kiếm. Các địa chỉ URL không cần phải có phần mở rộng của tên tập tin và được thiết kế để hỗ trợ các mẫu định dạng tên phù hợp với việc tối ưu hóa tìm kiếm (URL) và phù hợp với lập địa chỉ theo kiểu REST.

Hỗ trợ sử dụng đặc tả (các thẻ) của các trang ASP.NET(.aspx), điều khiển người dùng (.ascx) và trang master page (.marter). Bạn có thể sử dụng các tính năng có sẵn của ASP.NET như là sử dụng lồng các trang master page, sử dụng in-line expression (<%= %>), sử dụng server controls, mẫu, data-binding, địa phương hóa (localization) và hơn thế nữa.

Hỗ trợ các tính năng có sẵn của ASP.NET như cơ chế xác thực người dùng, quản lý thành viên, quyền, output caching và data caching, seession và profile, quản lý tình trạng ứng dụng, hệ thống cấu hình…

ASP.NET MVC 3 còn bổ sung một view engine mới là Razor View Engine cho phép thiết lập các view nhanh chóng, dễ dàng và tốn ít công sức hơn so với việc sử dụng Web Forms view engine.

## Mô hình 3 Lớp

### Giới thiệu

Mô hình 3 lớp là một kiến trúc mà trong đó giao diện người dùng (User Interface), Các xử lý luận lý (Bussiness Login) và Dữ liệu (Data) được phát triển thành các thành phần độc lập. Mô hình 3 lớp được coi là một cấu trúc phần mềm và là một mẫu thiết kế.

### Mô hình 3 lớp gồm 3 thành phần chính được tách rời gồm:

#### Tầng Presentation: hiển thị các thành phần giao diện để tương tác với người dùng như tiếp nhận thông tin, thông báo lỗi, …

#### Tầng Business Logic: thực hiện các hành động nghiệp vụ của phần mềm như tính toán, đánh giá tính hợp lệ của thông tin, … Tầng này còn di chuyển, xử lí thông tin giữa 2 tầng trên dưới.

#### Tầng Data: nơi lưu trữ và trích xuất dữ liệu từ các hệ quản trị CSDL hay các file trong hệ thống. Cho phép tầng Business logic thực hiện các truy vấn dữ liệu.

## Repository Pattern

Repository là một mẫu dùng để tạo ra một lớp trừu tượng trung gian giữa lớp dữ liệu và lớp luận lý. Lớp này chứa đựng phương thức thao tác mà để giao tiếp với lớp dữ liệu để phục vụ cho lớp luận lý. Mục đích tạo ra lớp này để cách ly với việc tiếp cận dữ liệu sao cho những thay đổi không ảnh hưởng trực tiếp đến lớp luận lý.

## Unit of Work trong Repository Pattern

Unit of Work được sử dụng để đảm bảo nhiều hành động như insert, update, delete...được thực thi trong cùng một transaction thống nhất. Nói đơn giản hơn, nghĩa là khi một hành động của người dùng tác động vào hệ thống, tất cả các hành động như insert, update, delete...phải thực hiện xong thì mới gọi là một transaction thành công. Gói tất cả các hành động đơn lẻ vào một transaction để đảm bảo tính toàn vẹn dữ liệu.

## Service Layer

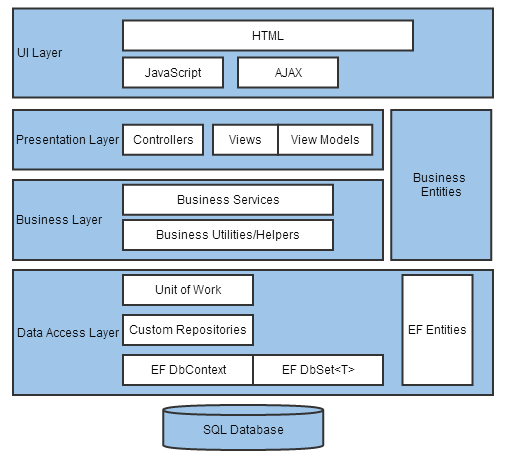
Service Layer là một mẫu thiết kế, được áp dụng trong mô hình thiết kế hướng dịch vụ, nhằm mục đích tổ chức các dịch vụ, trong kho dịch vụ, thành một tập hợp các lớp logic. Các dịch vụ được phân loại thành một chức năng chia sẻ lớp cụ thể.

## Entity Framework

### Giới thiệu

Entity Framework (EF) là một framework ánh xạ quan hệ đối tượng (ORM) dành cho ADO.NET, là 1 phần của .NET Framework. EF cho phép các nhà phát triển Web tương tác với dữ liệu quan hệ theo phương pháp hướng đối tượng đặc trưng. Lợi ích lớn nhất của EF là giúp lập trình viên giảm thiểu việc lập trình mã nguồn cần thiết để truy cập và tương tác với cơ sở dữ liệu. EF được Microsoft hỗ trợ phát triển lâu dài và bền vững, vì vậy EF là 1 framework mạnh nhất hiện nay để phát triển ứng dụng Web với sự hỗ trợ đông đảo của các nhà phát triển Web.

### Entity Framework trong dự án Web ASP.NET MVC



Trong mô hình trên, Entity Framework nằm gọn trong tầng Data Access Layer, Trong mô hình này Entity Framework đóng vai trò là nơi trung gian để tương tác với dữ liệu được lấy từ cơ sở dữ liệu.

### Cài đặt Entity Framework

Cài đặt bằng dòng lệnh

#### Trong menu chọn Tools –> NuGet Package Manager –> Package Manager Console

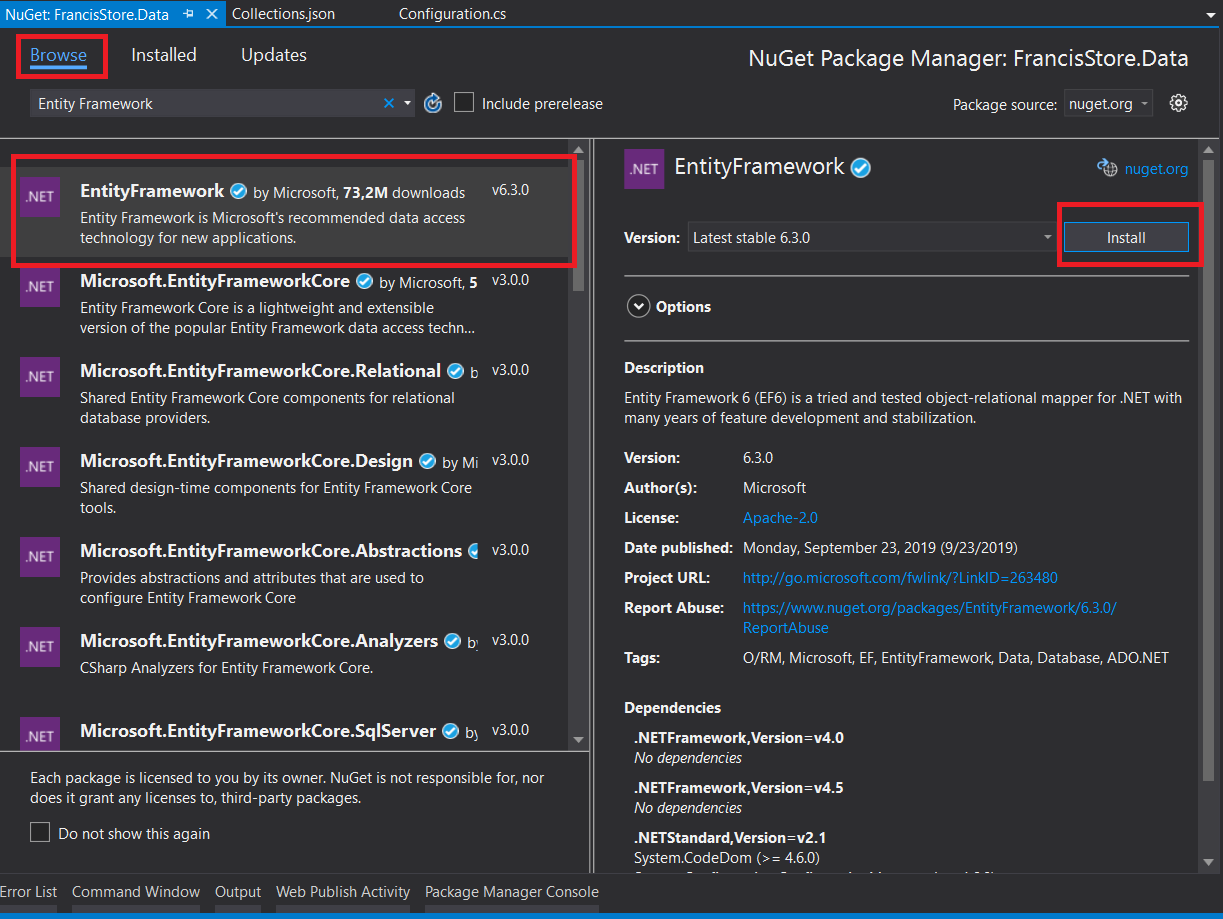
#### Run the Install-Package EntityFramework command

Cài đặt bằng NuGet

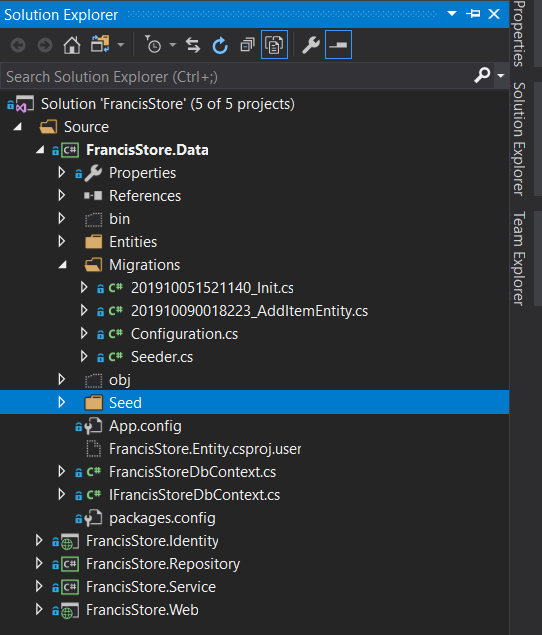
#### Trong giao diện Solution Explorer, nhấn chuột phải vào dự án cần cài đặt Manage Nuget Packages.

#### Mở tab Browse và tìm kiếm với tên Entity Framework.

#### Nhấn vào EntityFramework và nhấn Install



### Code First Migrations



Migrations là một công cụ dùng để phát triển cơ sở dữ liệu của ứng dụng nếu ứng dụng có sử dụng Entity Code First. Code First Migrations cung cấp một số công cụ cho phép bạn:

#### Khởi tạo cơ sở dữ liệu dựa vào các Entity Framework Model

#### Tạo ra những migrations giúp theo dõi những thay đổi trong cơ sở dữ liệu.

#### Giữ cho cơ sở dữ liệu được cập nhập theo những thay đổi đó.

### Bật Migrations

Để bật migrations bạn thực hiện theo các bước sau:

#### Vào menu Tools –> Library Package Manager –> Package Manager Console

#### Chạy câu lệnh Enable-Migrations trong Package Manager Console

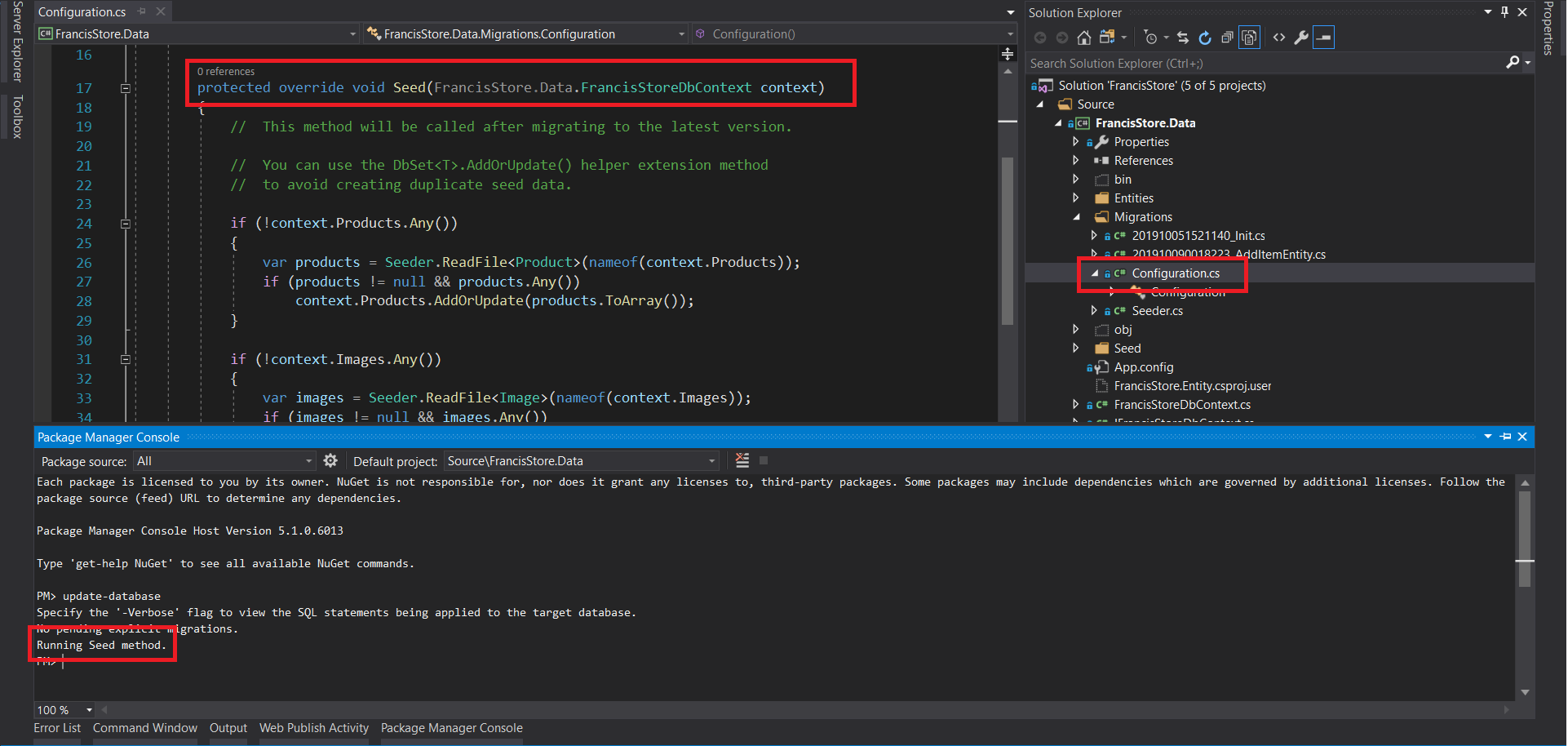
### Tạo các migrations và áp dụng các thay đổi vào cơ sở dữ liệu.

Code First Migrations có 2 câu lệnh chinh được dùng.

#### Câu lệnh Add-Migrations trong Package Manager Console để tạo ra những thay đổi trong cơ sở dữ liệu.

#### Câu lệnh Update-Database trong Package Manager Console để áp dụng các thay đổi đó vào cơ sở dữ liệu.

### Phương thức Seed



#### Phương thức Seed sẽ được gọi mỗi khi câu lệnh Update-Database được gọi

#### Phương thức sẽ tạo ra những dữ liệu ban đầu cho cơ sở dự liệu.

## ASP.NET Identity

ASP.NET Identity là một thanh phần của ASP.NET ra đời nhằm thay thế ASP.NET Membership. ASP.NET Identity cung cấp các tính năng đầy đủ và đa dạng về xác thực người dùng như đăng nhập, đăng ký, khôi phục mật khẩu, xác thực email, xác thực bằng SMS. Ngoài ra ASP.NET Identity còn hỗ trợ xác thực người dùng thông qua xác thực bên thứ ba như Facebook, Google, Twitter, Microsoft.

### Mục đích

#### One ASP.NET Identity system

ASP.Net Identity có thể sử dụng trên toàn bộ nền tảng ASP.NET bao gồm ASP.NET MVC, Web Forms, Web Pages, Web API, and SignalR.

#### Ease of plugging in profile data about the user

Bạn có thể dễ dàng kiểm soát để lưu trữ thông tin người dùng ví dụ như bạn có thể dễ dàng cho phép hệ thống lưu trữ thông tin về ngày sinh của người dùng khi người dùng đăng ký.

#### Persistence control

Mặc định, ASP.NET Identity sẽ lưu trữ thông tin người dùng trên cơ sở dữ liệu. ASP.NET Identity sử dụng Entity Framework Code First để thực hiện việc lưu trữ thông tin.

ASP.NET Identity cho phép bạn dễ dàng thay đổi khóa chinh của bảng người dùng trong cơ sở dữ liệu.

#### Unit testability

ASP.NET Identity cho phép bạn dễ dàng kiểm thử ứng dụng có sử dụng ASP.NET Identity.

#### Role provider

ASP.NET Identity giúp kiểm soát người dùng nhằm hạn chế truy cập vào các phần của ứng dụng theo vai trò của người dùng.

#### Social Login Providers

ASP.NET Identity hỗ trợ liên kết đăng nhập thông qua các tài khoản như Microsoft Account, Facebook, Twitter, Google…

#### OWIN Integration

ASP.NET Identity tích hợp OWIN.

#### NuGet package

ASP.NET Identity có thể được cài đặt thông qua NuGet

## Dependency Injection

Dependency injection (DI) là cung cấp cho một object những object nó phụ thuộc (dependencies) từ bên ngoài truyền vào mà không phải khởi tạo từ trong phương thức khởi tạo. Điều này giúp ứng dụng linh động hơn và dễ test hơn.

Dependency Injection là một design pattern hay một kiểu mẫu thiết kế phần mềm được áp dụng trong trường hợp mà ở đó một object A phụ thuộc vào object B.

Trường hợp một object này phụ thuộc vào object khác là khá phổ biến trong lập trình. Một ví dụ đơn giản có thể kể đến như method registerAccount() trong object User thực hiện việc đăng ký tài khoản sẽ cần tới object Mailer dùng để gửi email kích hoạt tài khoản.

Khi một object A phụ thuộc vào một Object B thì một hệ lụy có nhiều khả năng xảy ra đó là sự thay đổi trong B có thể khiến Object A cũng phải thay đổi theo. Sử dụng Dependency Injection chúng ta có thể giảm thiểu sự thay đổi hoặc giữ nguyên source code của Object A ngay cả khi Object B thay đổi mà vẫn đảm bảo ứng hoạt động như dự kiến.

## Inversion of Controller container

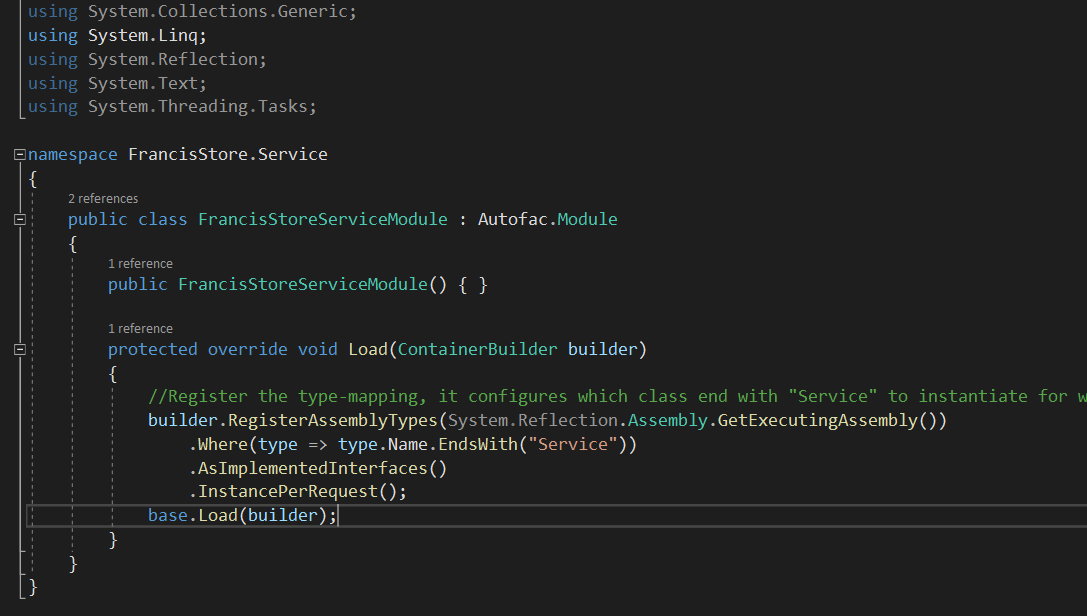
Khi áp dụng kỹ thuật DI này, thì một vấn đề khác lại nảy sinh, làm thế nào chúng ta biết được một lớp A này phụ thuộc vào những lớp nào để khởi tạo nó? Việc tạo ra một đối tượng của lớp A rất đơn giản nếu như nó chỉ phụ thuộc trực tiếp vào một lớp khác. Tuy nhiên có khả năng xảy ra trường hợp phụ thuộc lồng nhau, ví dụ như lớp A --> lớp B--> lớp C. Và nó gây rất nhiều khó khăn cho việc khởi tạo một object mà chúng ta cần, bởi vì danh sách các lớp phụ thuộc lồng nhau rất sâu.

Để giải quyết điều này, người ta nghĩ ra Dependency Injection Container hay còn gọi là Inversion of Control Container (IoC container). Về bản chất thì IoC Conainter là một tấm bản đồ, hay một dịch vụ tổng đài cuộc gọi. Nó cho ta biết một lớp phụ thuộc vào những lớp class nào khác và phân giải được những class đó bằng kỹ thuật Reflection, hoặc từ danh sách đã được lập trinh viên đăng ký trước.

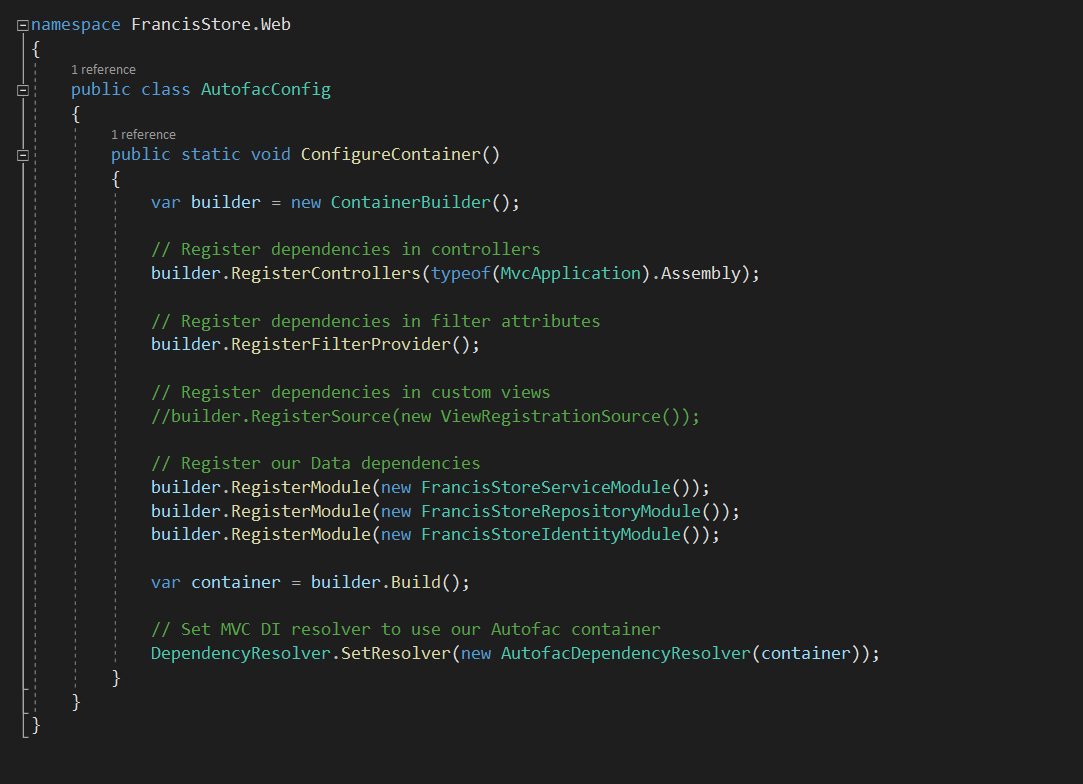
## Autofac

Autofac là một Inversion of Controller container cho .NET Core, ASP.NET Core, .NET 4.5.1+, ASP.NET, ứng dụng Universal…

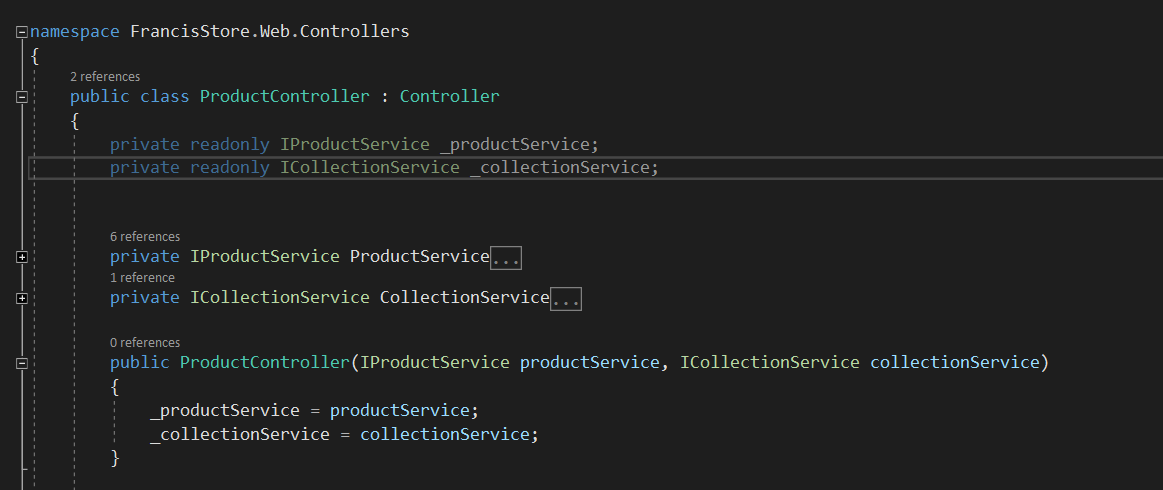
### Đăng ký thanh phần, các lớp và phụ thuộc của nó



### Đăng ký các thanh phần của Autofac



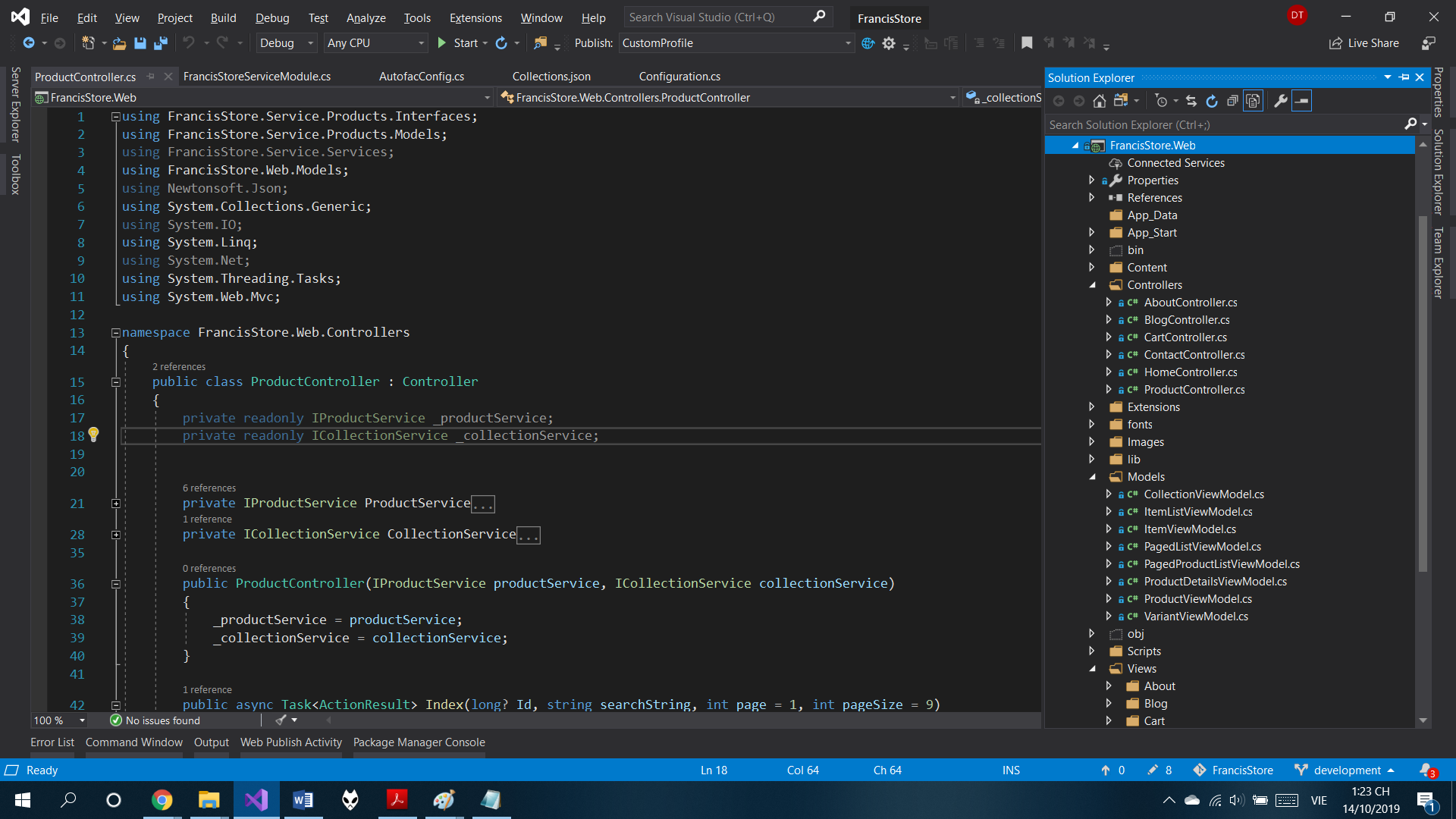
### Phân giải các phụ thuộc của một lớp bằng phương thức khởi tạo



# Kết quả thực hiện

## Kết quả sử dụng ASP.NET MVC 5

### Cấu trúc sử dụng ASP.NET MVC 5



Cấu trúc đề tài sử dụng ASP.NET MVC 5 bao gồm

#### Thư mục App\_Start chứa các lớp, phương thức được gọi khi ứng dụng web chạy.

#### Thư mục Content chứa các thư viện của style script có trên NuGet

#### Thư mục Controllers chứa các lớp Controller để xử lý những thông tin được lấy từ dự án FrancisStore.Service sau đó kết hợp với các View và Model để phản hổi lại các trang cho người dùng.

#### Thư mục Extensions chứa các phương thức mở rộng cho các lớp cụ thể.

#### Thư mục fonts chứa các fonts chữ và style của chúng.

#### Thư mục Images chứa hình ảnh bao gồm hình ảnh của website và hình ảnh sản phẩm.

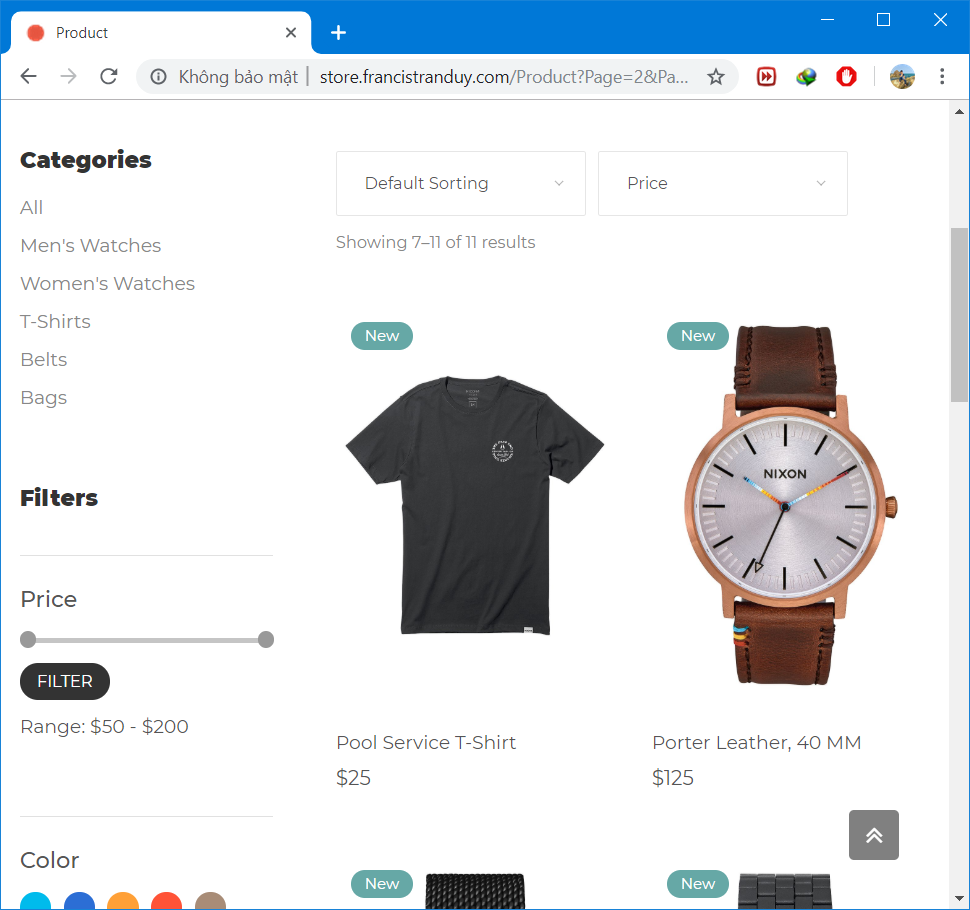
#### Thư mục lib chứa các thư viên của javascript và style script dược cài đặt bằng Libman.

#### Thư mục Models (ViewModel) chứa các lớp mô hình dữ liệu, thuộc tinh nào sẽ được xuất ra cho người dùng.

#### Thư mục Scripts chưa các thư viên của javascript có trong NuGet

#### Thư mục Views chứa các Razor Page để kết hợp cùng với Models để kết xuất ra các trang để hiển thị cho người dùng.

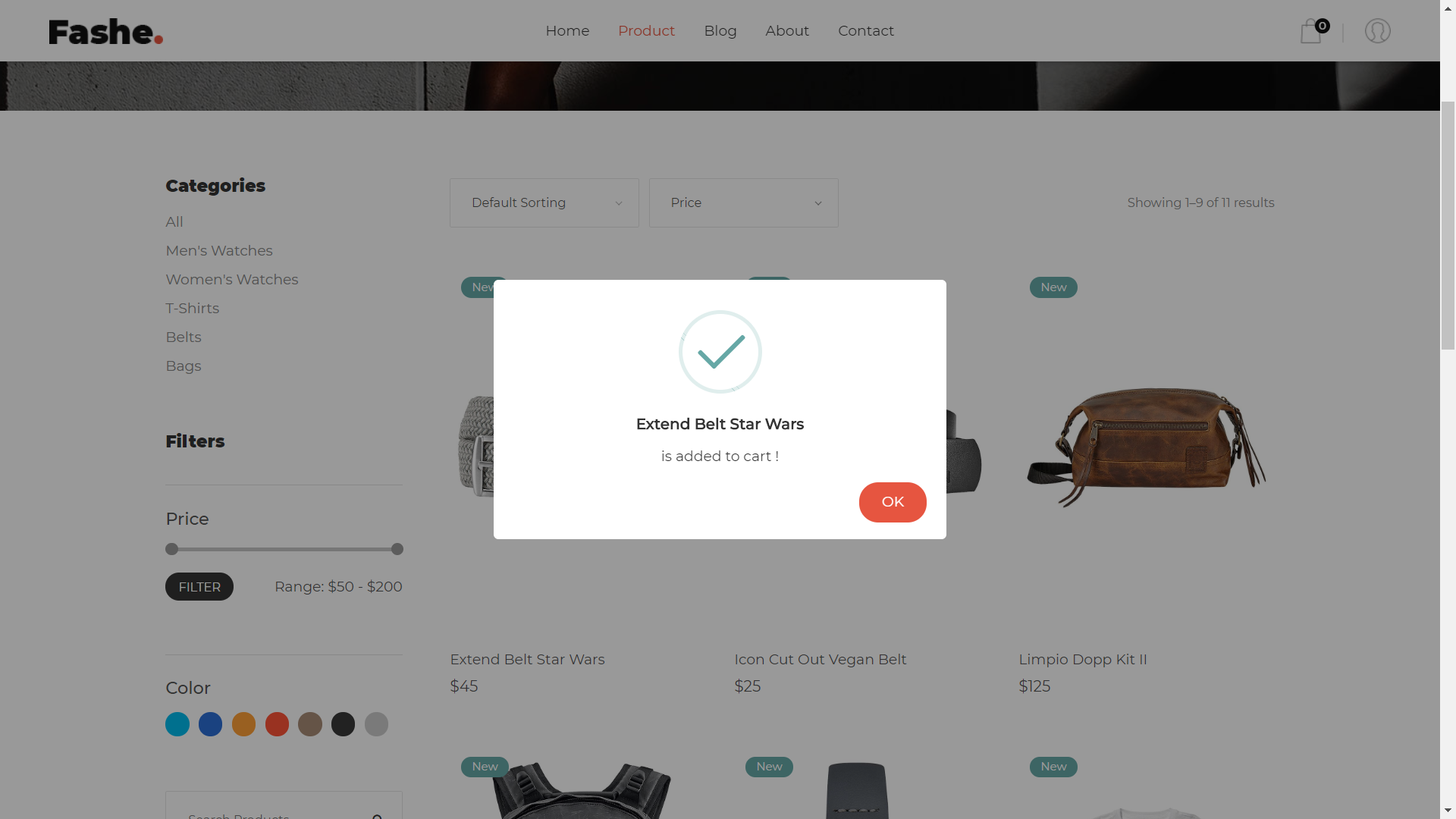
### Hiển thị sản phẩm theo danh sách



#### Các thông tin hiển thị bao gồm hình ảnh, tên sản phẩm, giá sản phẩm. Giá của sản phẩm là giá của biến thể mặc định.

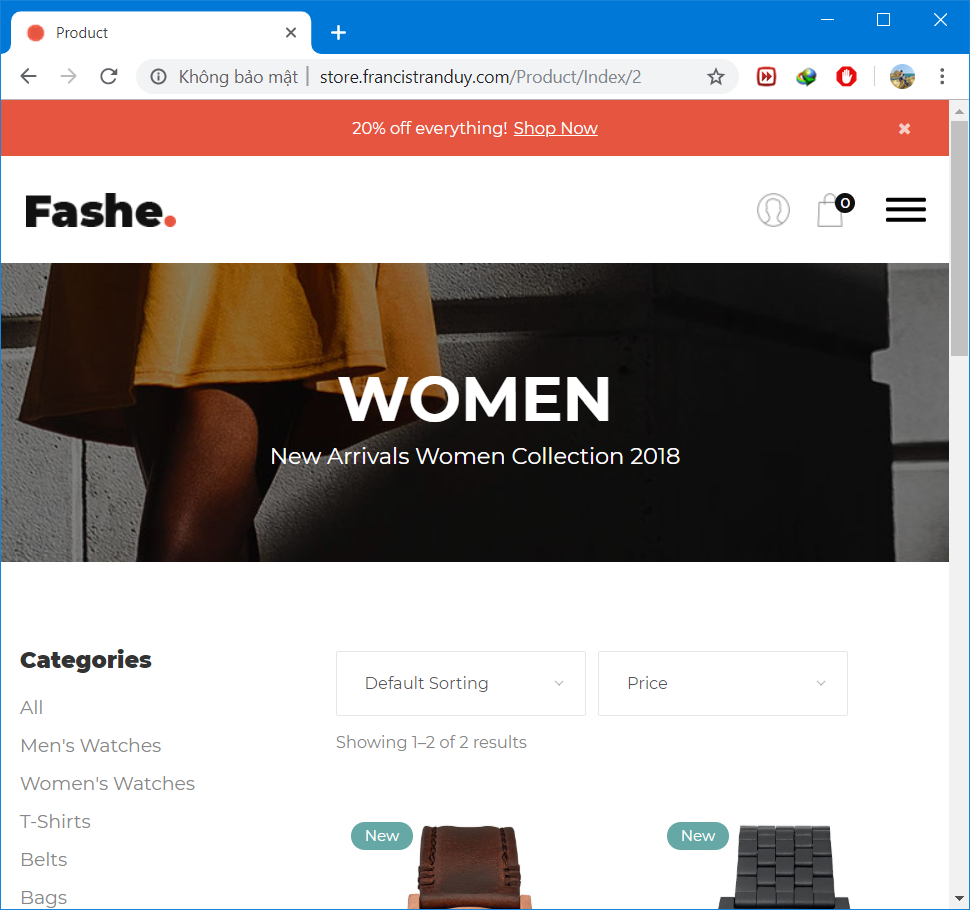
#### Sản phẩm được hiện thị theo danh sách có phân trang.

#### Tùy theo kích thước của cửa sổ trinh duyệt mà số lượng sản phẩm hiển thị trong một hàng là khác nhau.



#### Sản phẩm có thể được thêm vào giỏ hàng trong danh sách, biến thể được thêm vào giỏ hàng là biến thể đầu tiên.

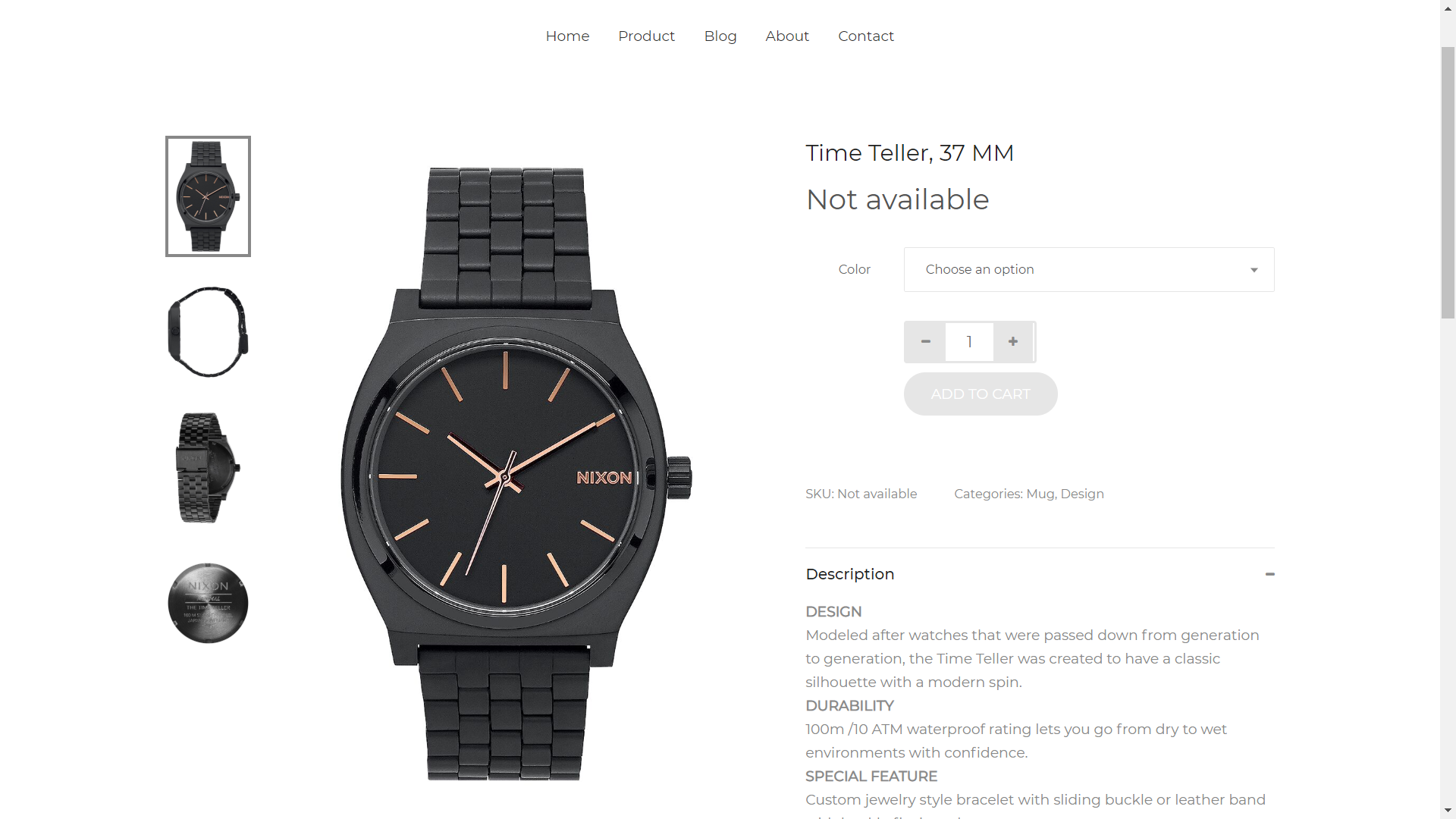
### Hiện thị sản phẩm theo collection



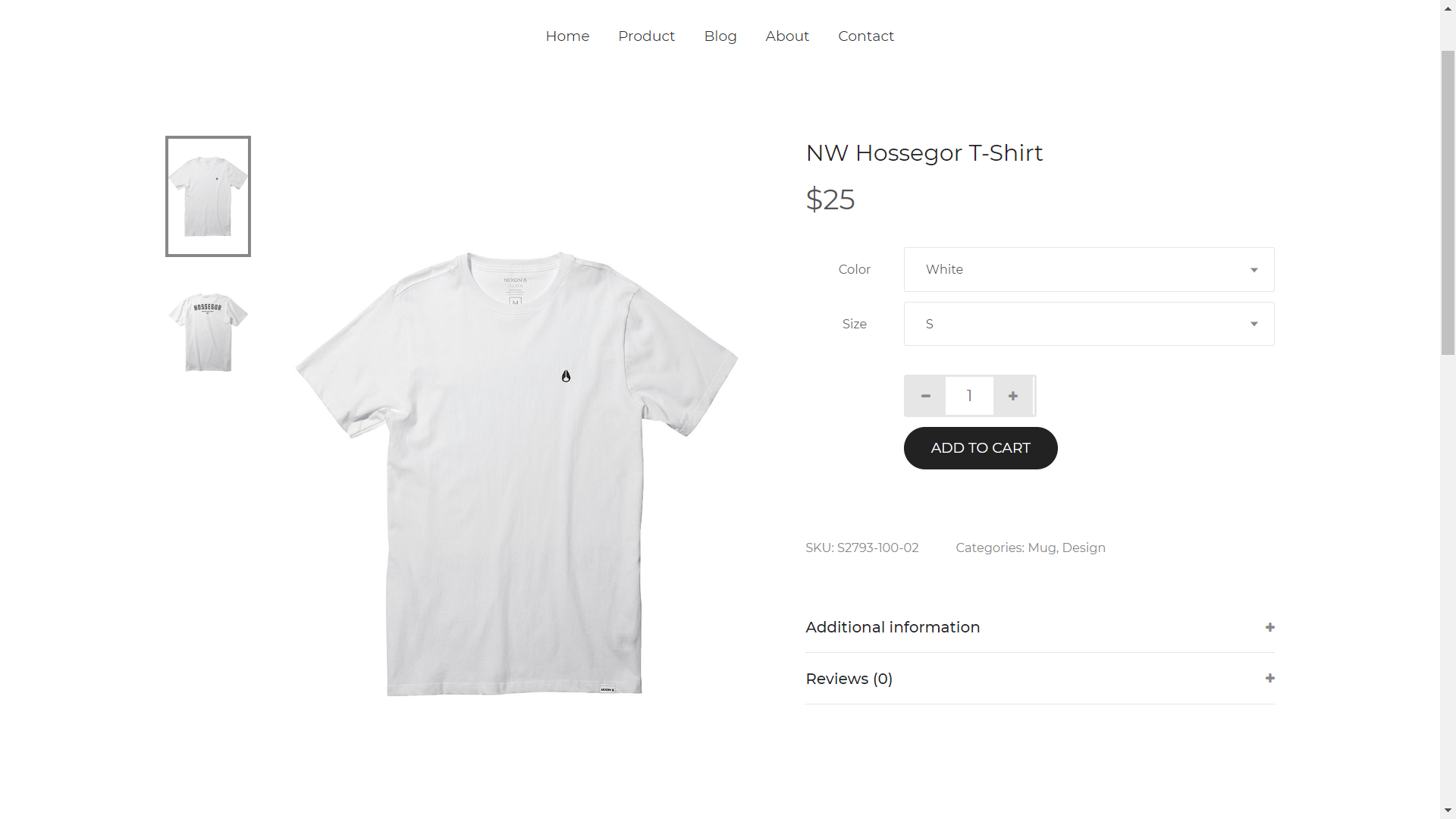
#### Các chức năng tương tự như hiển thị sản phẩm theo danh sách.

#### Mỗi sản phẩm thuộc nhiều collection, mỗi collection có nhiều sản phẩm. Website cho phép sản phẩm hiển thị theo collection.

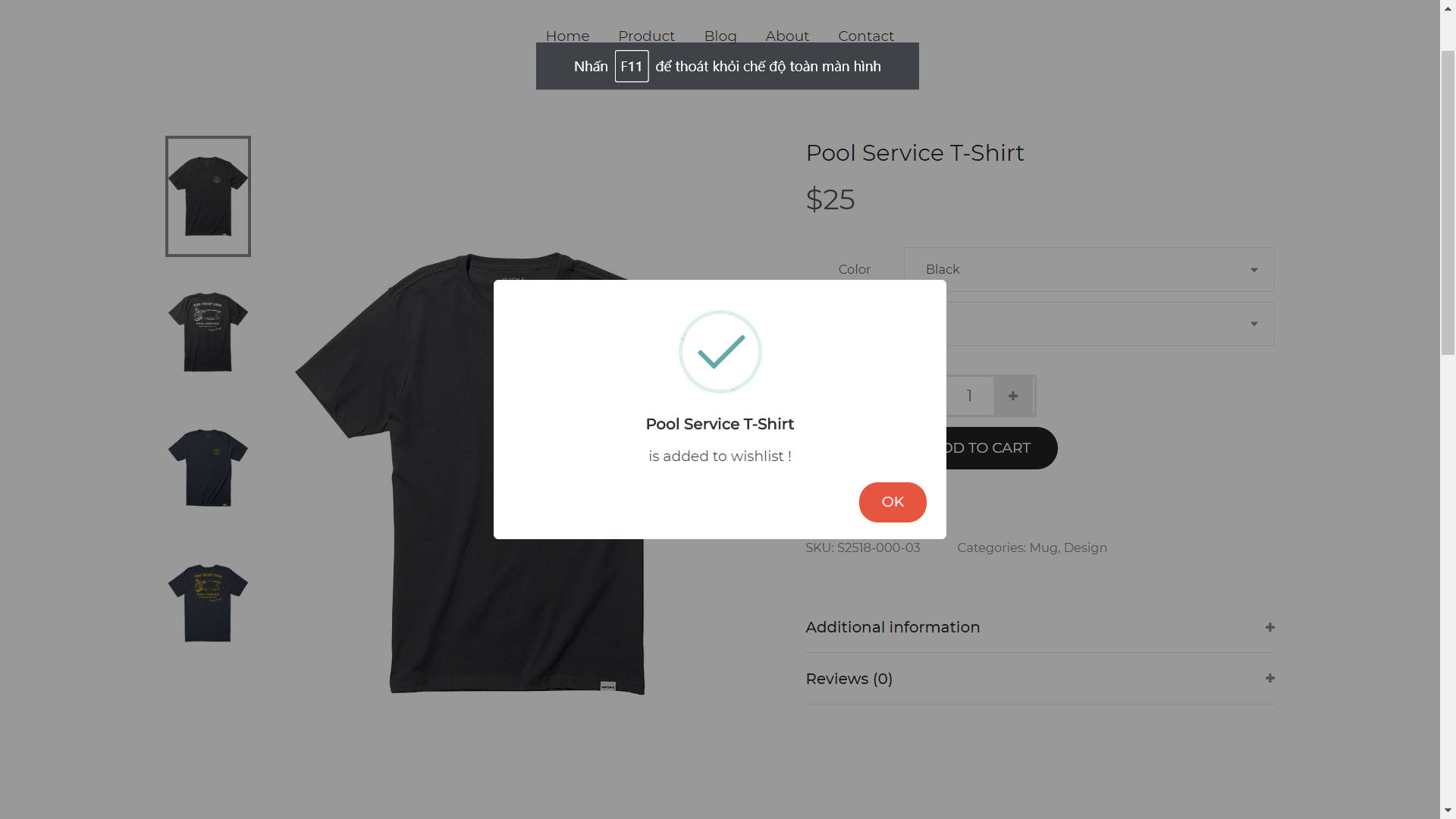
### Hiển thị chi tiết một sản phẩm



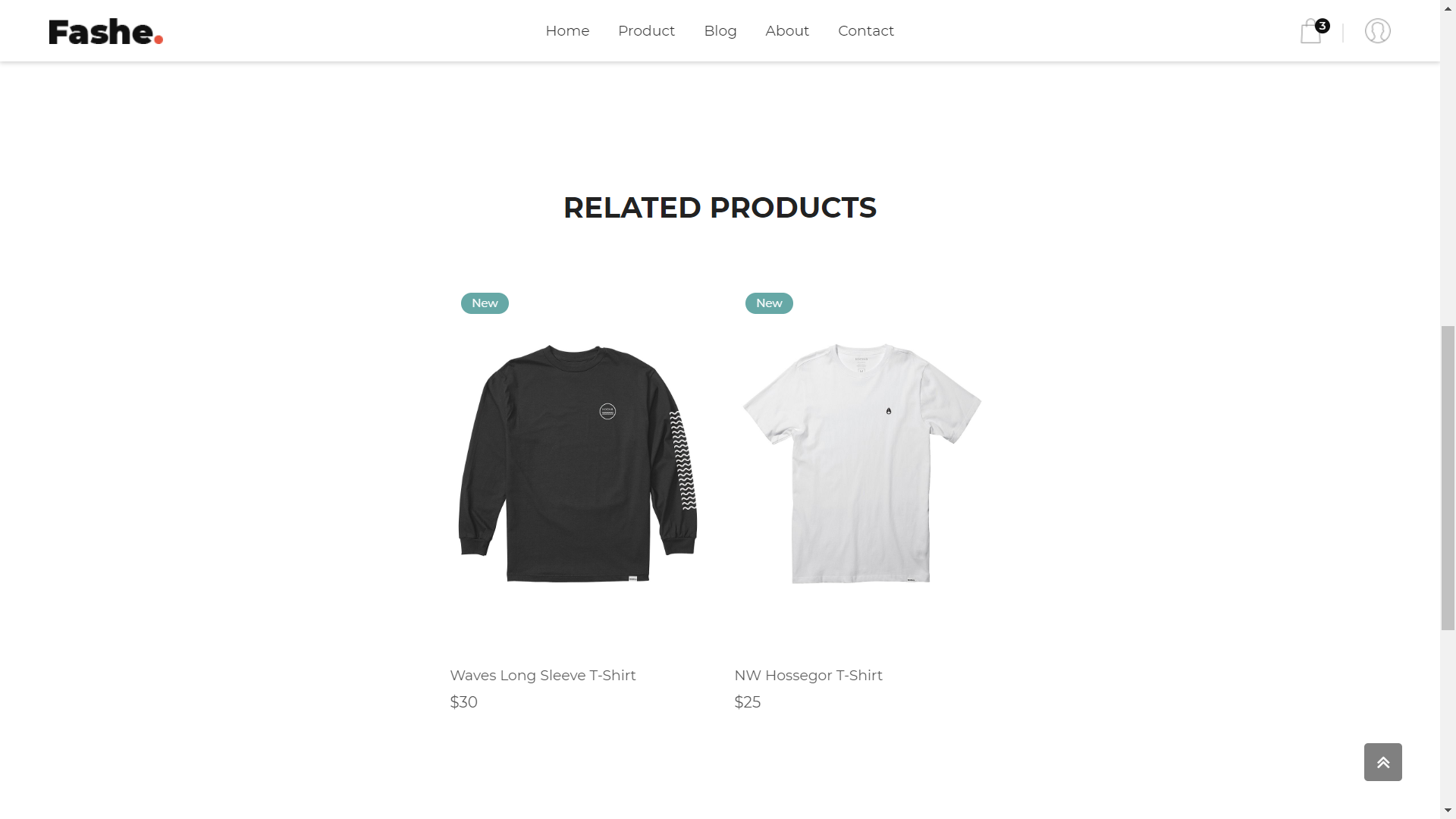
#### Các thông tin sản phẩm bao gồm những hình ảnh của sản phẩm, tên sản phẩm, mô tả, thông tin thêm.



#### Trang có thể hiển thị thông tin của từng biến thể của sản phẩm khi người dùng lựa chọn tất cả các thuộc tinh của sản phẩm bao gồm các thông tin như giá sản phẩm, các thuộc tinh và SKU

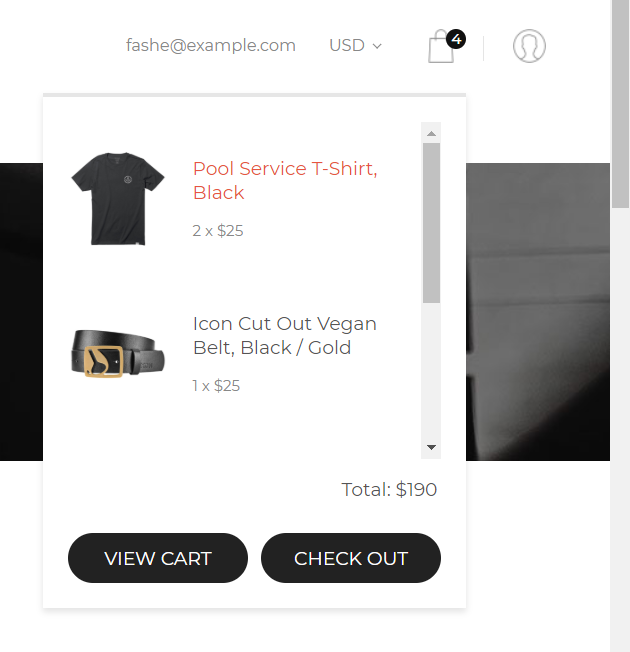


#### Khi người dùng lựa chọn đầy đủ các thuộc tinh, sản phẩm có thể được thêm vào giỏ hàng. Nếu các thuộc tinh không có trong bất kỳ biến thể nào của sản phẩm thì không thể thêm sản phẩm vào giỏ hàng.



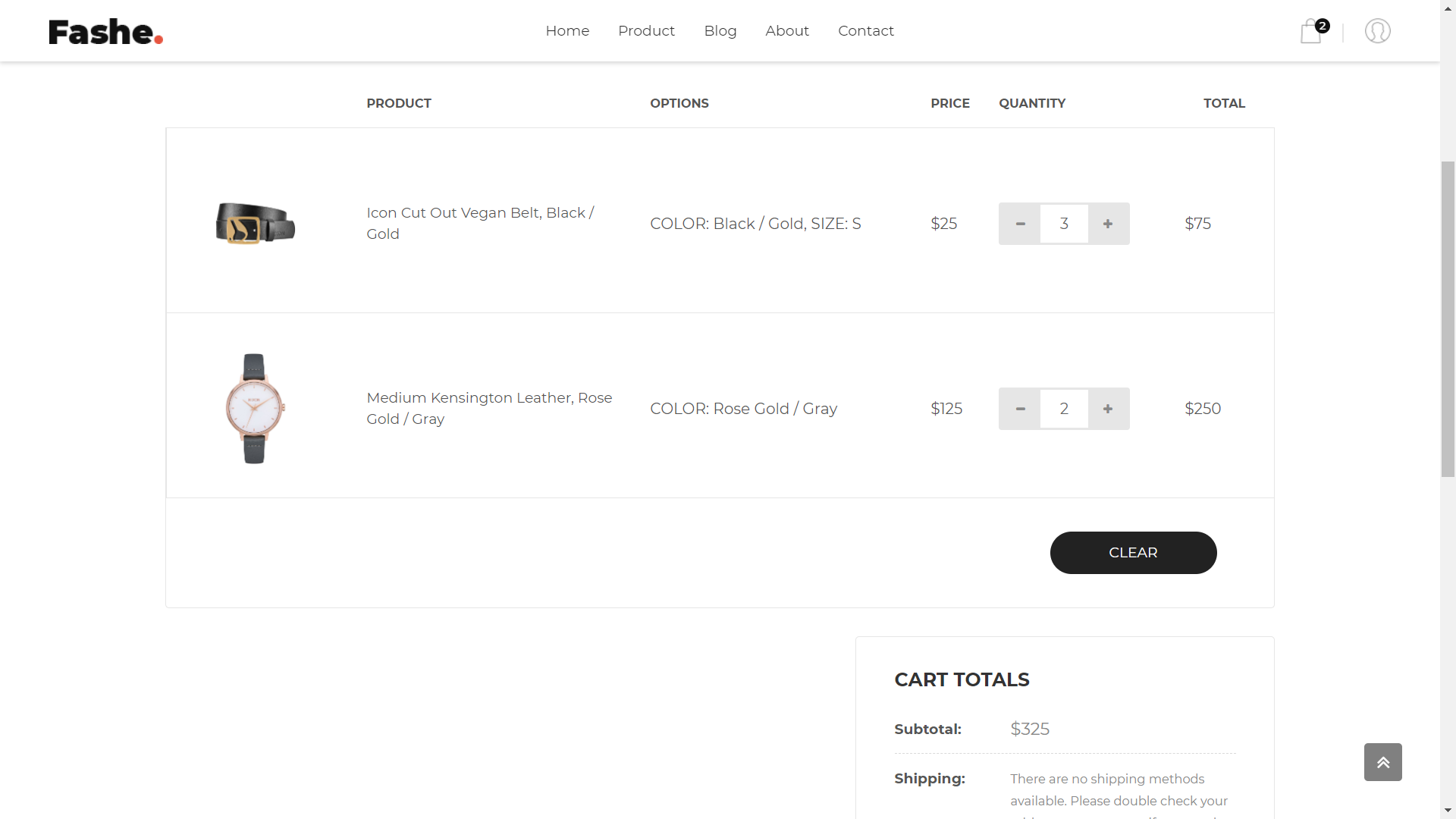
#### Website cũng hiển thị những sản phẩm liên quan được lấy từ collection được ưu tiên. Thông tin của sản phẩm liên quan bao gồm hình ảnh, tên, giá sản phẩm.

### Hiển thị giỏ hàng trên thanh menu



#### Trên thanh menu các sản phẩm có trong giỏ hàng sẽ được hiển thị bao gồm hình ảnh sản phẩm, tên sản phẩm, số lượng sản phẩm, tổng giá giỏ hàng và số sản phẩm.

### Hiển thị trang giỏ hàng



#### Giỏ hàng hiển thị các thông tin bao gồm danh sách sản phẩm, thông tin sản phẩm, tổng giá trị giỏ hàng.

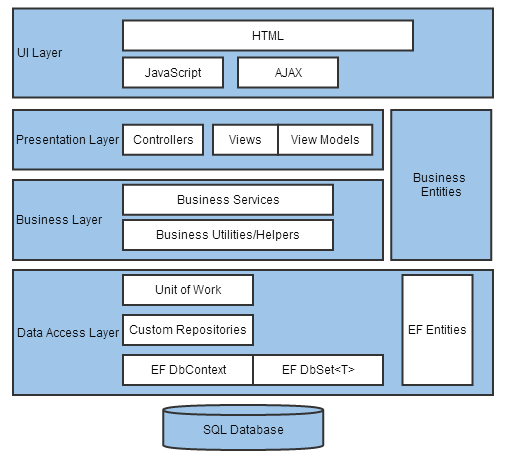
#### Thông tin sản phẩm bao gồm hình ảnh, tên sản phẩm, các thuộc tinh sản phẩm, giá sản phẩm, số tượng và tổng giá sản phẩm.

#### Người dùng có thể xóa giỏ hàng bằng cách nhấn vào nút xóa.

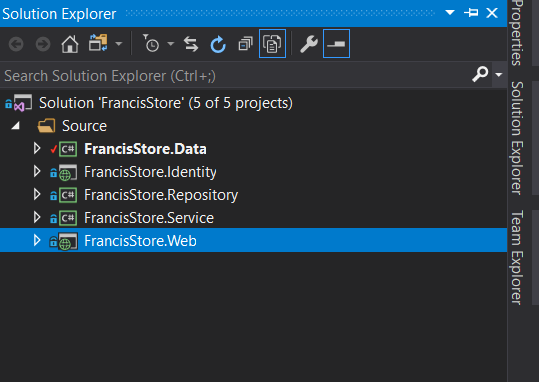
#### Khi người dùng tiến hành đăng nhập, tất cả sản phẩm có trong giỏ hàng sẽ được chuyển vào giỏ hàng của người dùng. Giỏ hàng của người dùng sẽ được duy trì đến khi người dùng xóa.

## Kết quả sử dụng mô hình 3 lớp vào đề tài

### Cấu trúc được 3 lớp được sử dụng trong đề tài



### Áp dụng vào đề tài



#### Lớp Presentation bao gồm dự án FrancisStore.Web, FrancisStore.Identity chứa các Controllers, Views và ViewModels.

#### Lớp Bussiness bao gồm dự án FrancisStore.Service chứa các lớp service phục vụ cho việc xử lý thông tin lấy dừ tầng Data và trả về các Bussiness Model để tầng lớp Presentation sử dụng.

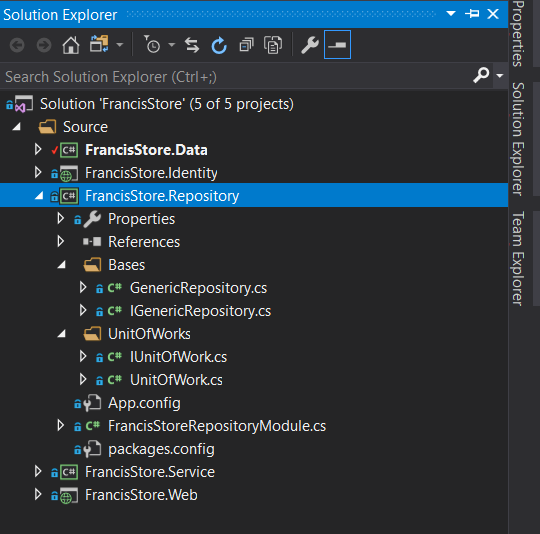
#### Lớp Data bao gồm dự án FrancisStore.Data, FrancisStore.Repository chứa các Entities Model, DbContext, Repository, UnitOfWork phục vụ cho việc lấy dữ liệu từ cơ sở dữ liệu.

### Đánh giá

Các dự án án trong đề tài được phân biệt rõ ràng thành các tính năng riêng biệt, dễ dàng kiểm thử, dễ dàng tái sử dụng nếu có phát sinh thêm các dự án khác như dự án liên quan đến Web API hay Website quản lý.

## Kết quả sử dụng Reponsitory Pattern

### Áp dụng vào đề tài



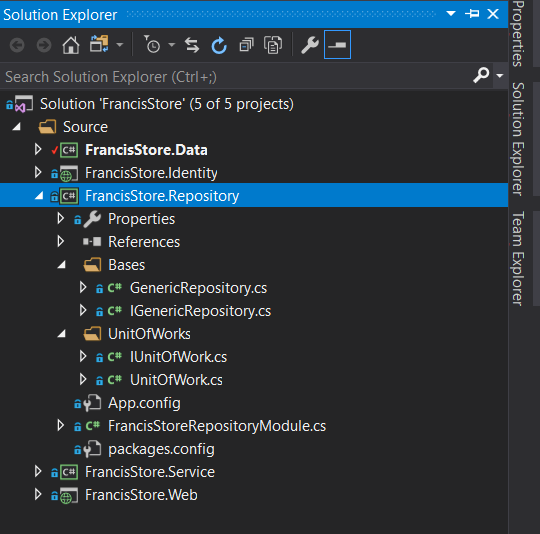
Repository Pattern nằm toàn bộ trong dự án FrancisStore.Repository phục vụ cho việc truy xuất dữ liệu bằng các phương thức cơ bản đươc định nghĩ trong các interface như GetAll, GetById, Insert, Delete, Save.

### Đánh giá

Repository Pattern mổ tả các hành vi để truy xuất dữ liệu, hỗ trợ cho việc quản lý các đổi tượng trong việc sử dụng dependency injection.

## Kết quả sử dụng UnitOfWork

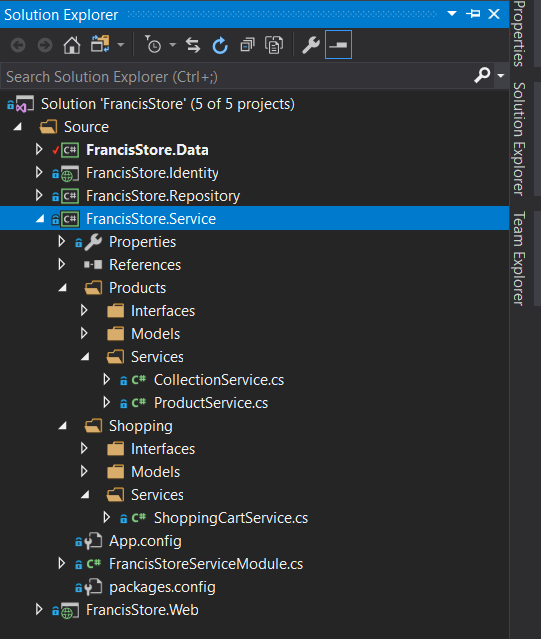
### Áp dụng vào đề tài



UnitOfWork nằm toàn bộ trong dự án FrancisStore.Repository phục vụ cho việc khai báo các Repository, DbContext cần thiết để thực hiện các công việc trong lớp FrancisStore.Service.

## Kết quả sử dụng Service Layer

### Áp dụng vào đề tài



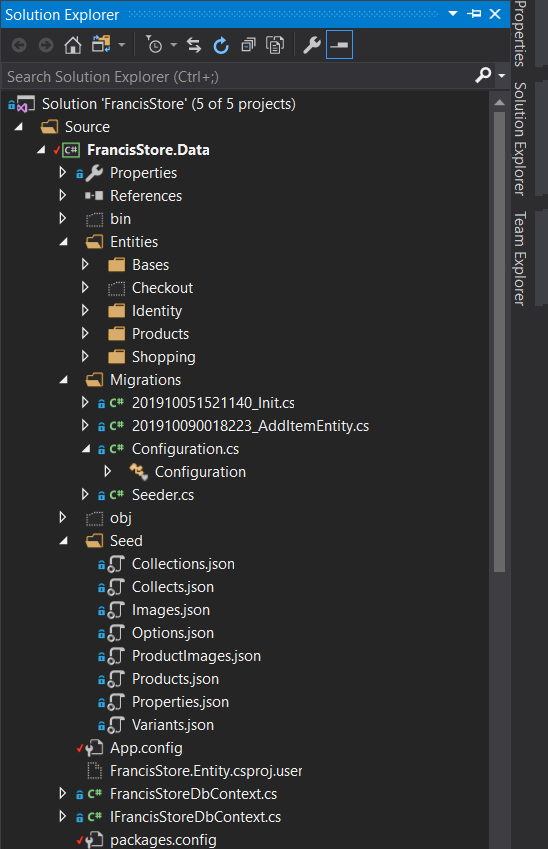
Service layer nằm toàn bộ trong dự án FrancisStore.Service phục vụ cho việc lấy yêu cầu từ lớp Presentation sau đó truy vấn từ dữ liệu lấy từ lớp Data rồi xử lý và trả kết quả về cho lớp Presetation bằng các Bussiness Model.

### Đánh giá

Service layer hỗ trợ dễ dàng cho việc quản lý các đối bằng Dependency Injection. Ngoài ra Service layer dễ dàng sử dụng bởi nhiều dự án thuộc lớp Presetation như dự án FrancisStore.Identity và FrancisStore.Web.

## Kết quả sử dụng Entity Framework

### Áp dụng vào đề tài



Entity Framework được sử dụng trong dự án FrancisStore.Data.

Thư mục Entities có chứa các lớp Entity Model ánh xạ đến các bảng trong cơ sở dự liệu.

Thư mục Migrations có chứa các migrations lưu lại các thay đổi trong cơ sở dữ liệu.

Thư mục Seed có chưa các têp tin JSON để phương thức Seed có thể đọc được và ghi dữ liệu vào cơ sở dữ liệu.

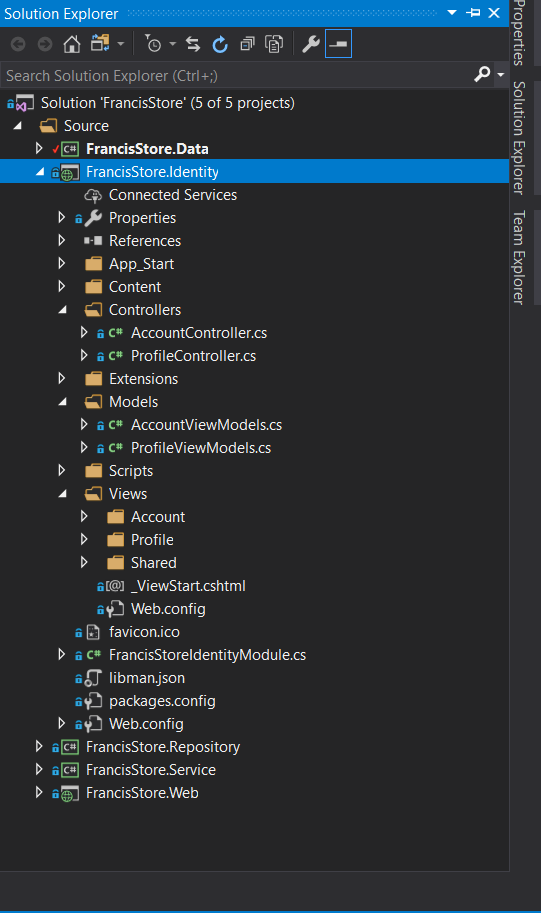
FrancisStoreDbContext mô tả những bảng có trong cơ sở dữ liệu.

### Đánh giá

Entity Framework giúp lập trinh viên dễ dàng hơn trong việc truy xuất dữ liệu trong các bảng, ngoài ra Code First Migrations giúp cho việc thay đổi cấu trúc cơ sở dữ liệu không làm ảnh hưởng đến cơ sở dữ liệu đã tồn tại. Phương thức Seed giúp khởi tạo dữ liệu mặc định trong cơ sở dữ liệu của đề tài.

## Kết quả sử dụng Identity

### Áp dụng vào đề tài



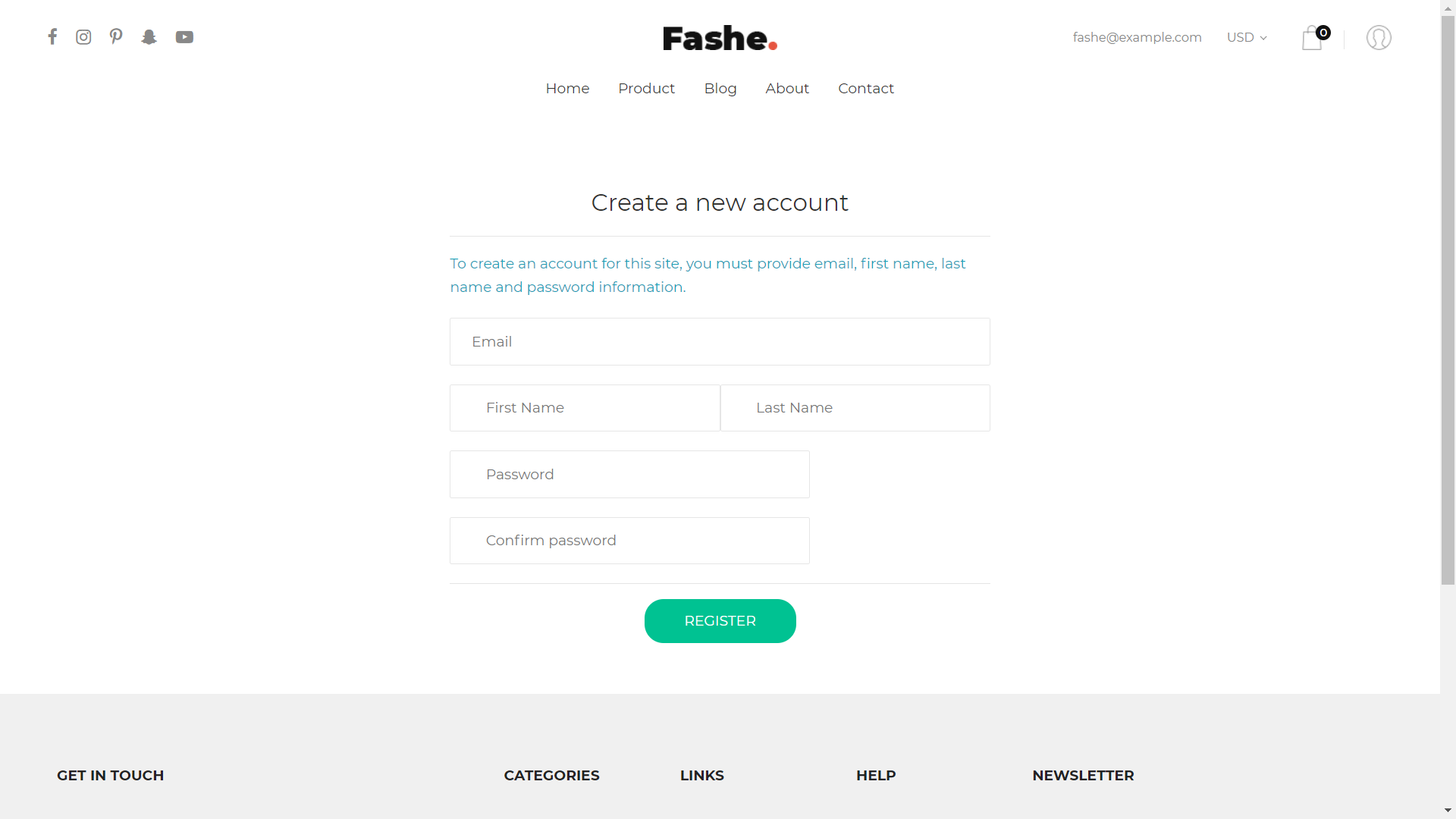
ASP.NET Identity được sử dụng trong dự án FrancisStore.Entity có cấu trúc tương tự như dự án FrancisStore.Web đã được trình bày phía trên.

Điểm khác biệt ở đây là FrancisStore.Identiy được tích hợp sẵn các Controllers, View, Model phục vụ cho việc xác thực người dùng.

FrancisStore.Identity được tạo bằng mẫu có sẵn trong hộp thoại tạo dự án và được tùy chỉnh theo FrancisStore.Web, nhưng được tách rời khỏi FrancisStore.Web để có thể tái sử cho dự án khác.

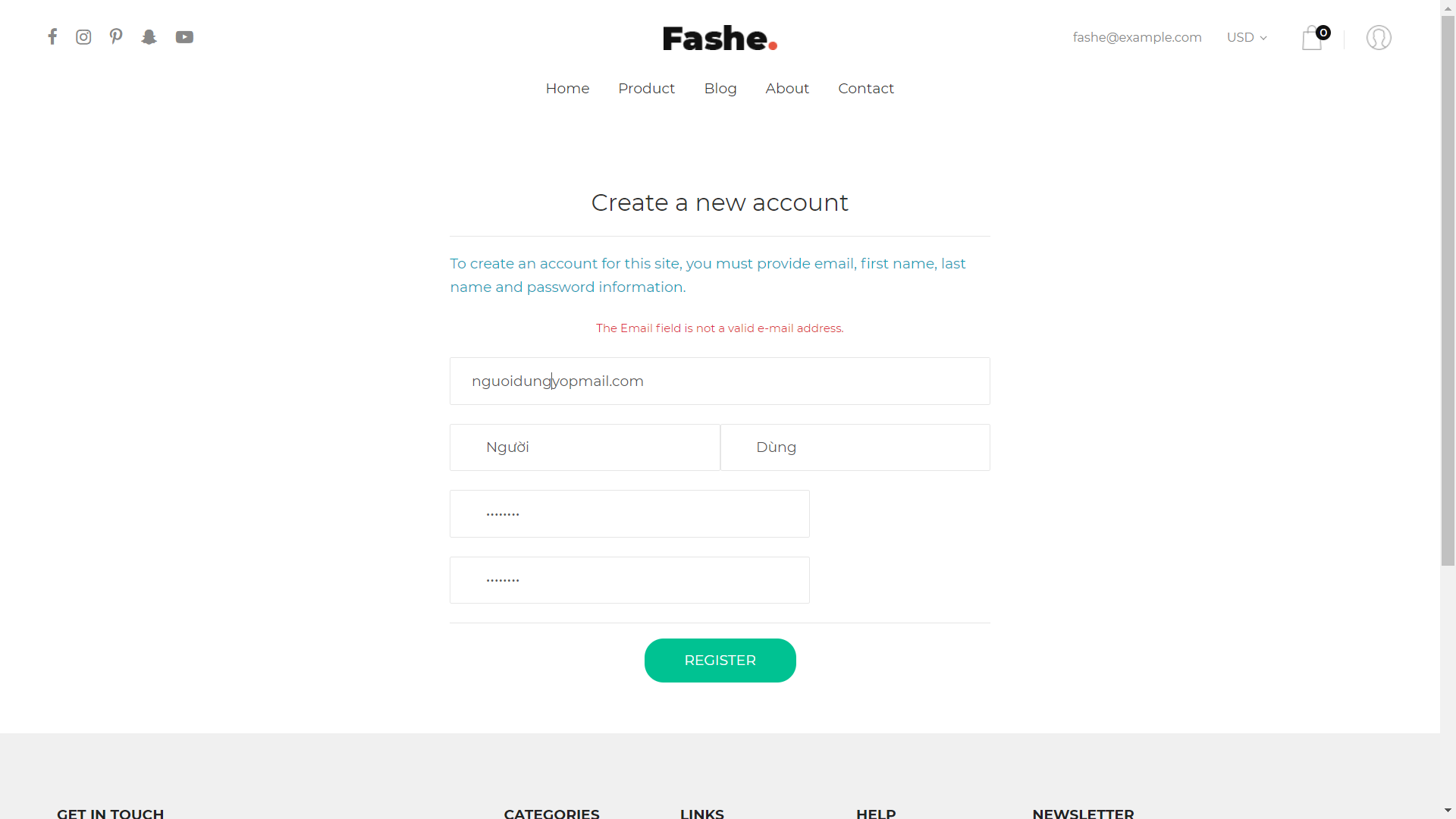
Để sử dụng được ASP.NET Identity. Bạn phải đăng tải (public) FrancisStore.Identity trước FrancisStore.Web trên cùng một web application.

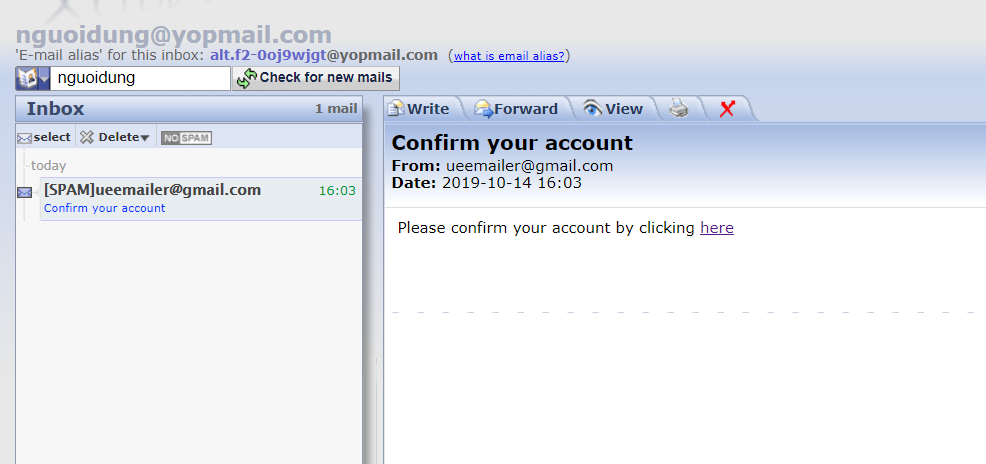
### Chức năng đăng ký



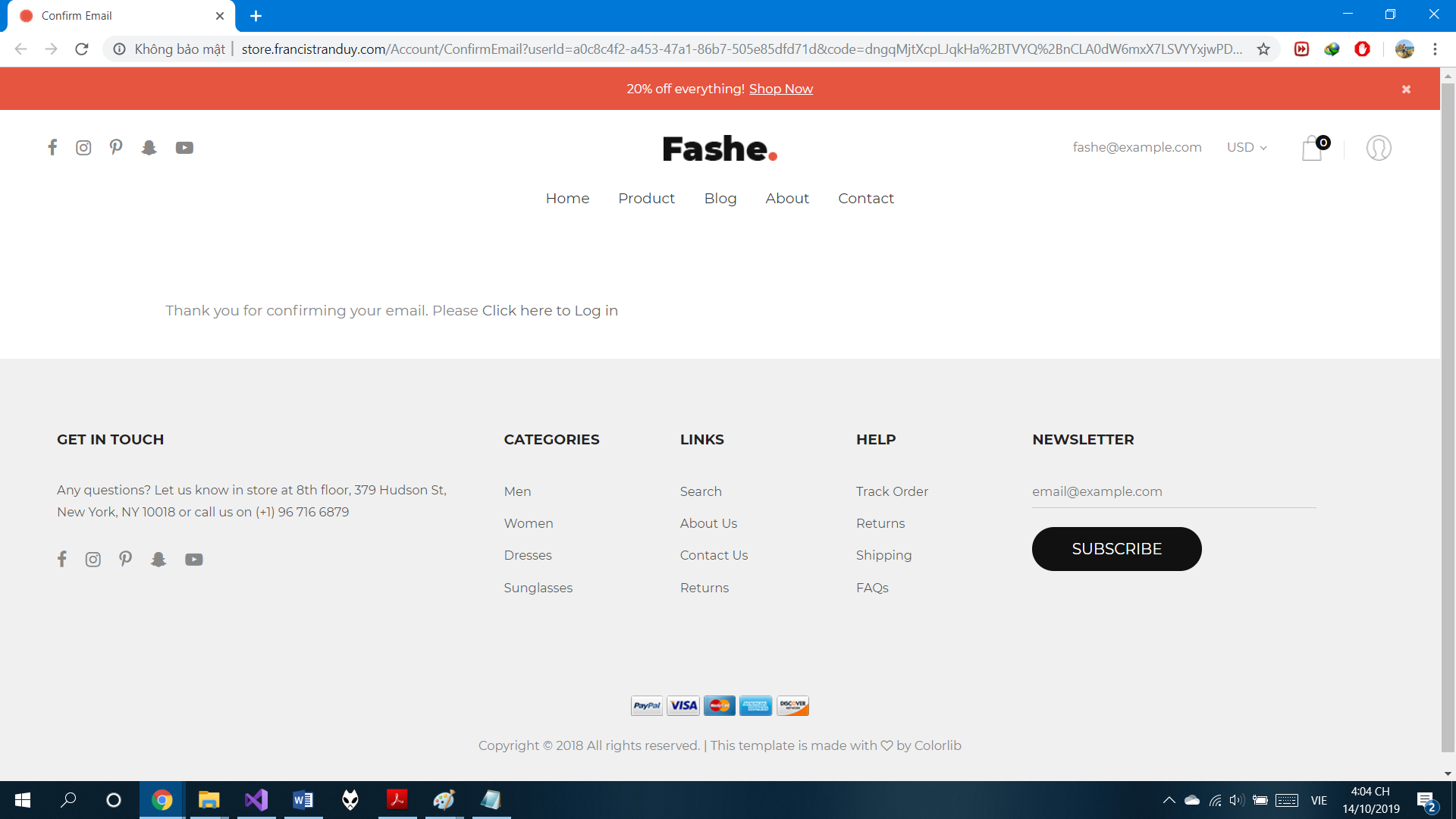
Để đăng ký tài khoản, website yêu cầu người dùng nhập thông tin bao gồm email, họ, tên và mật khẩu.

Website sẽ kiểm tra tính hợp lệ và gửi email xác thực

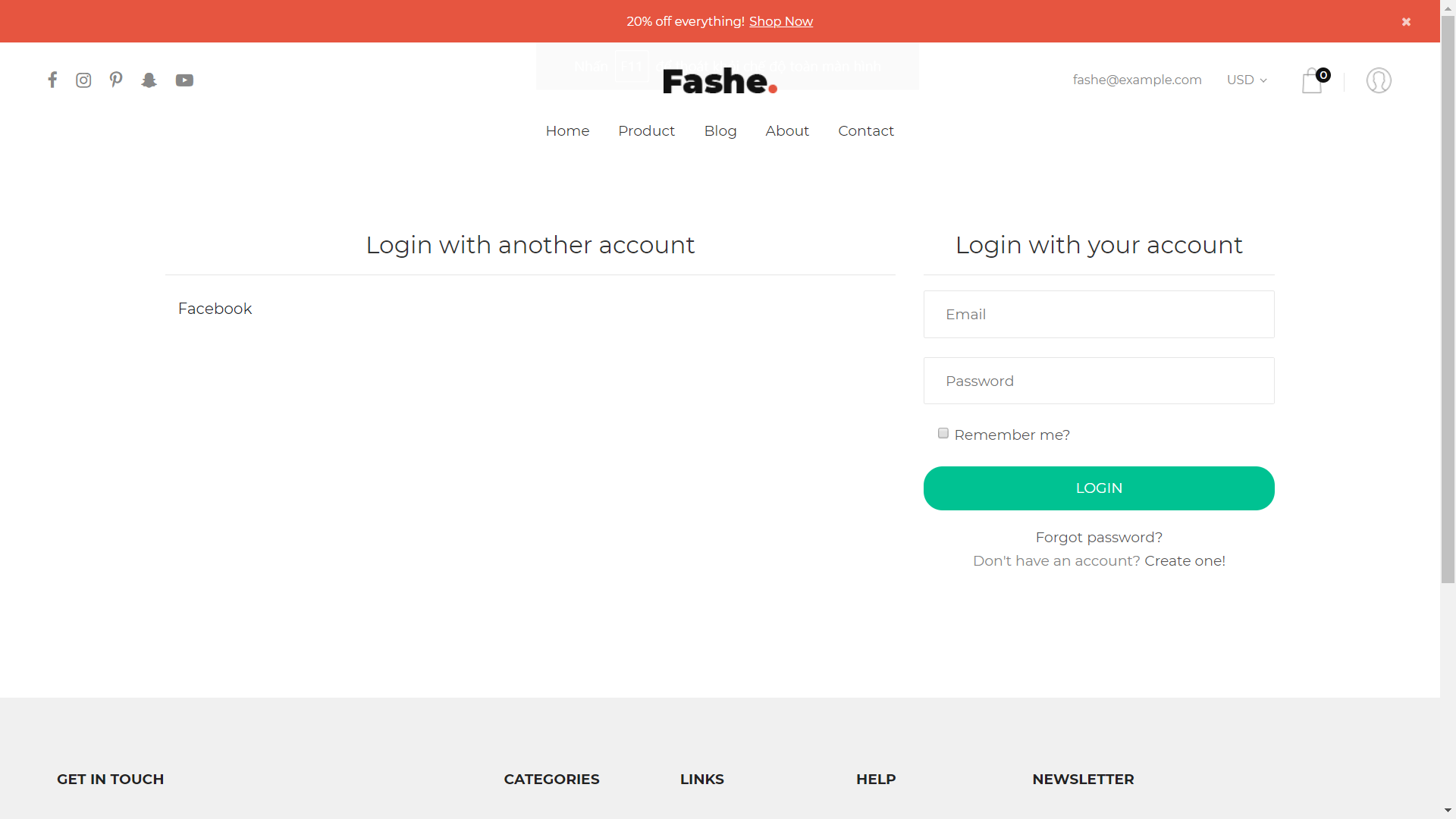




Xác thực email thành công

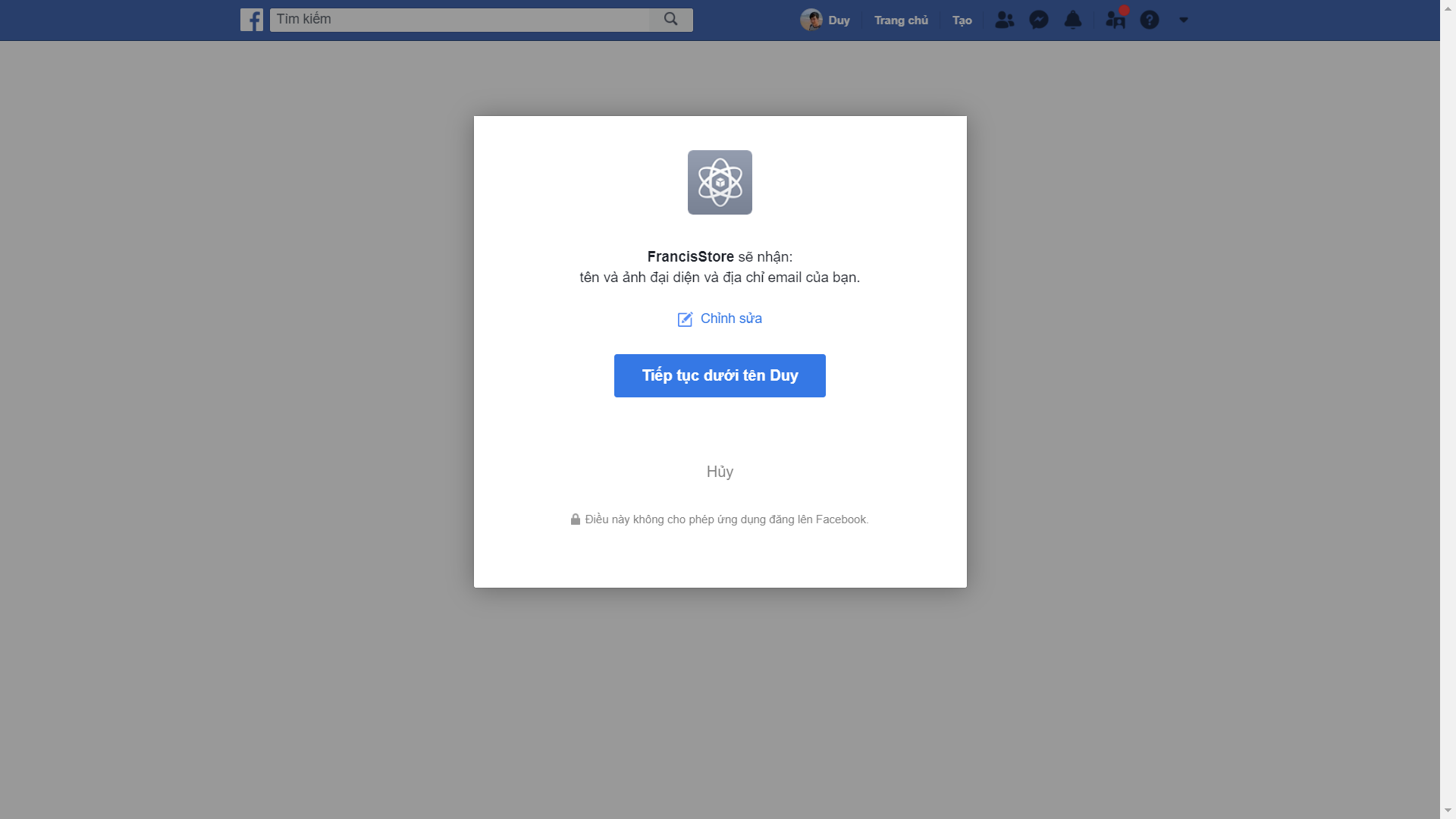


### Chức năng đăng nhập

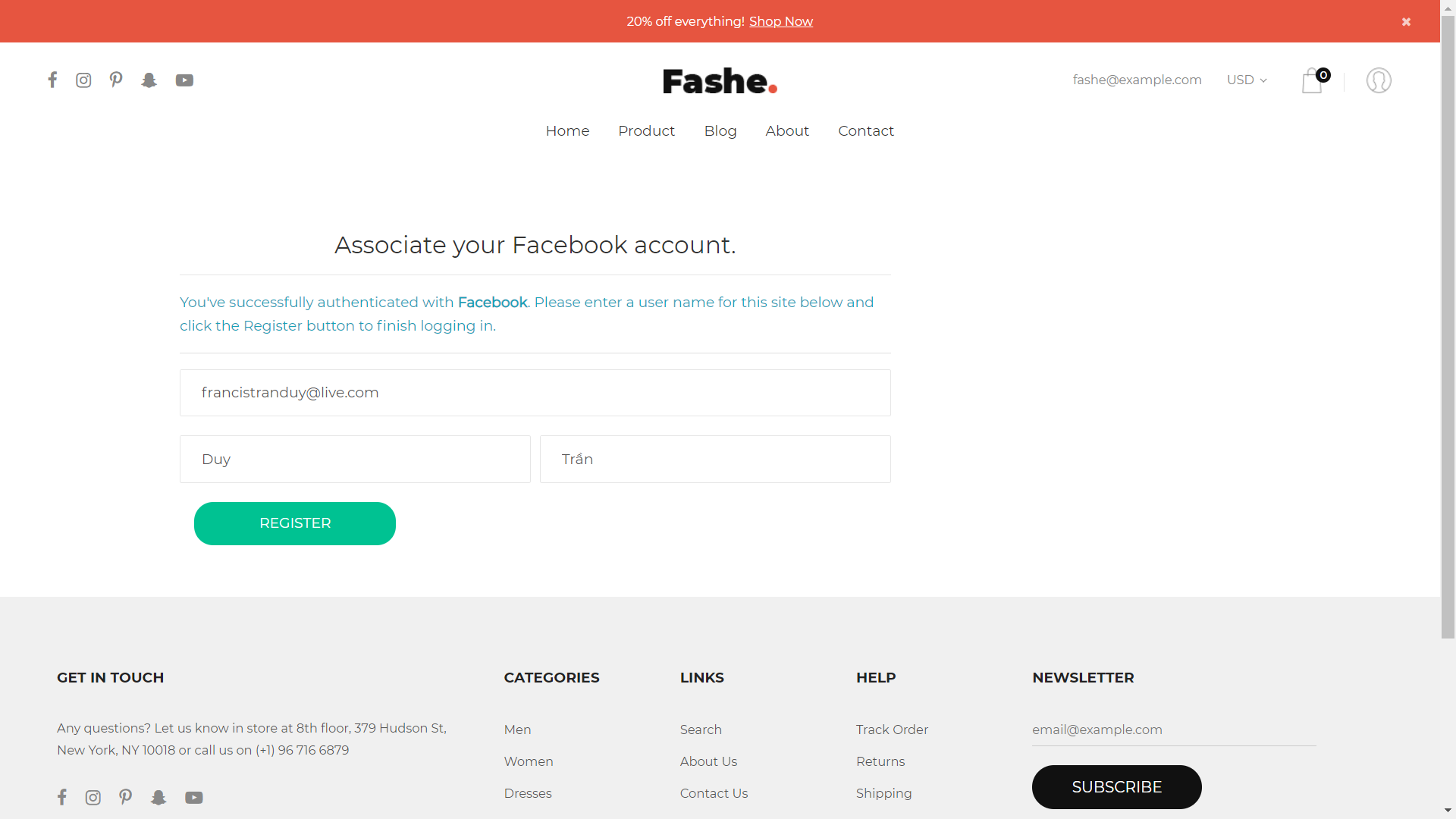


Người dùng có 2 lựa chọn đăng nhập là đăng nhập tài khoản của website và đăng nhập tài khoản facebook.

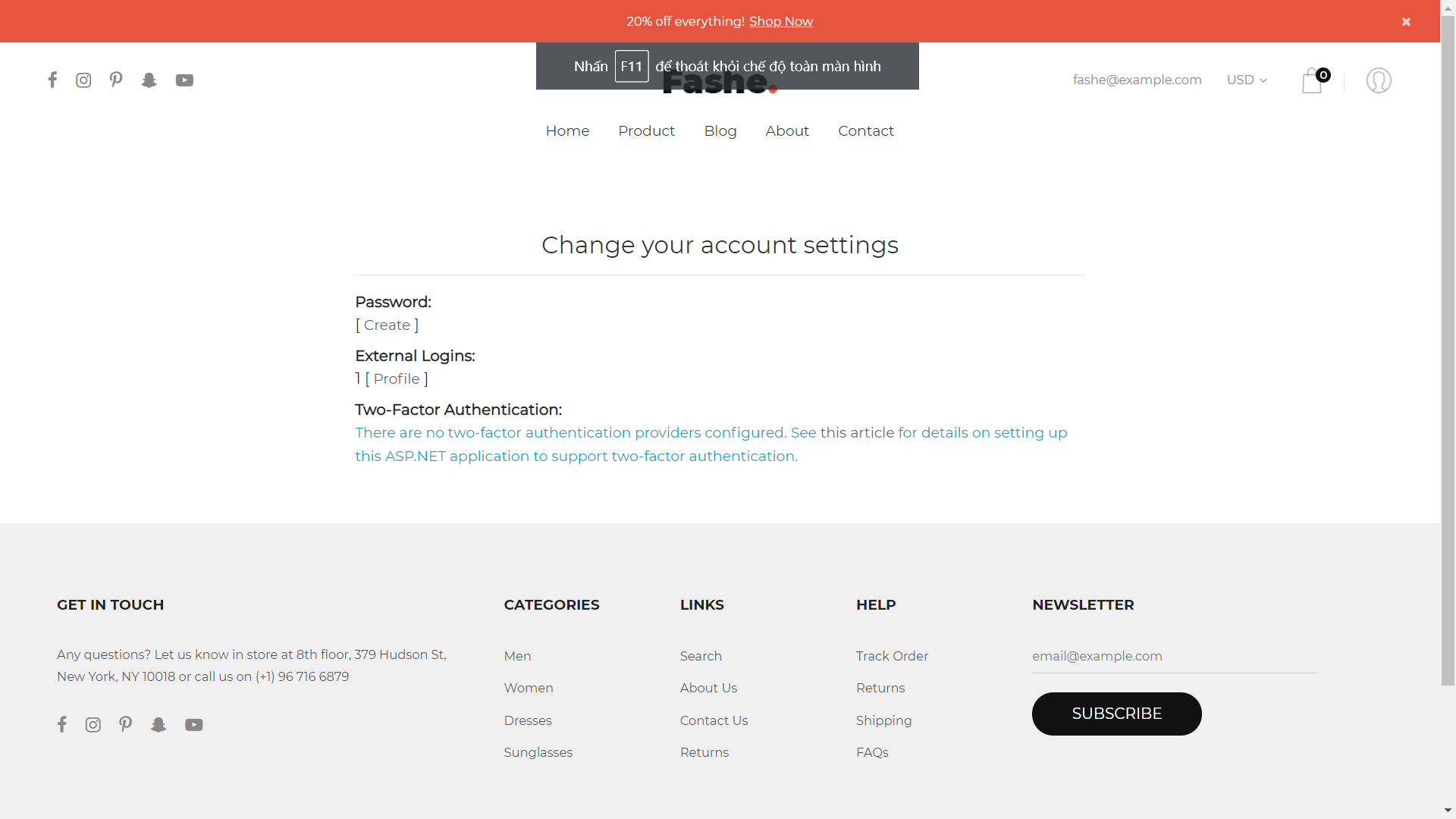
Khi đăng nhập bằng tài khoản facebook, facebook sẽ yêu cầu người dùng cho phép website được lấy thông tin của người dùng.



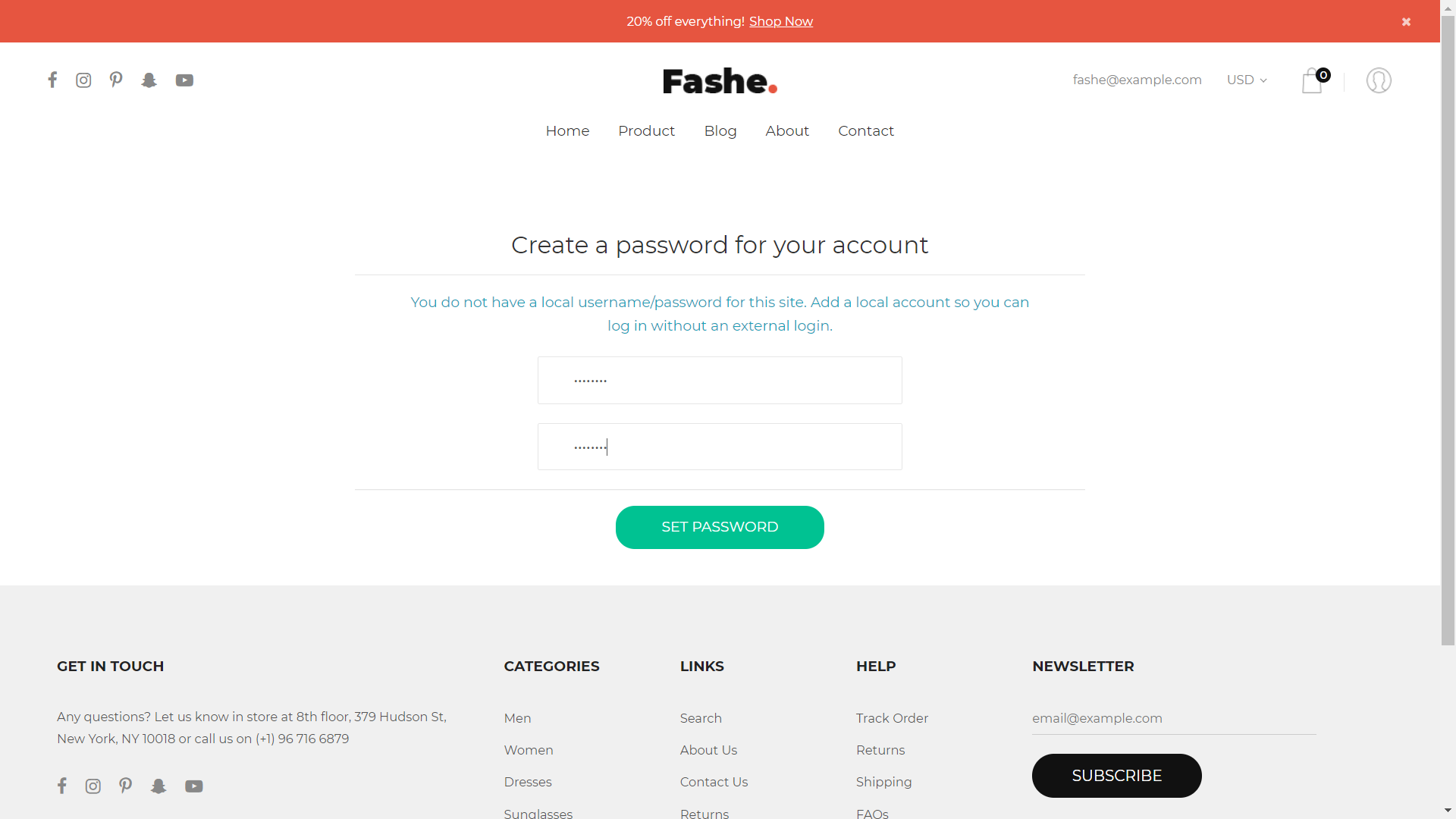
Website dựa vào thông tinh đó để tạo người dùng và lưu vào cơ sở dữ liệu.



### Chức năng thay đổi thông tin người dùng

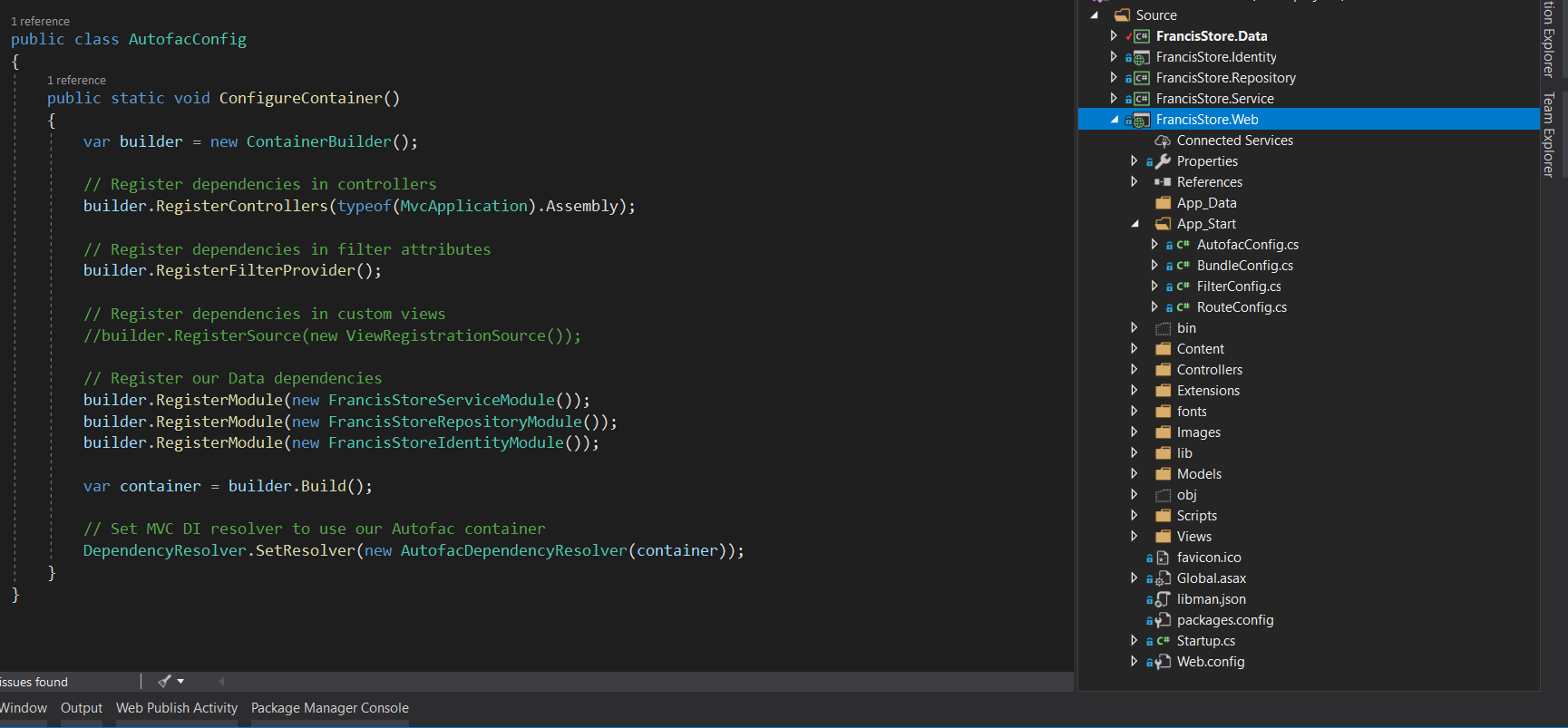


Người dùng có thể thay đổi mật khẩu hoặc đặt mật khẩu nếu người dùng đăng nhập bằng tài khoản facebook.



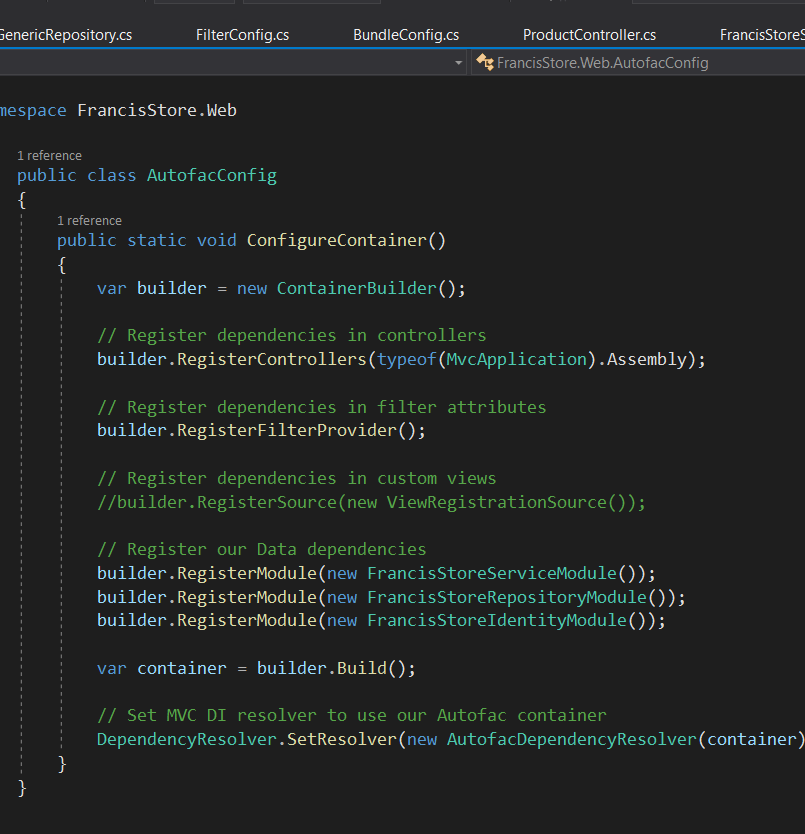
## Autofac

### Áp dụng vào đề tài



Autofac là một Inversion of Controller container được dùng để phân giải các service, Dbcontext nhằm đảm bảo rằng mỗi một yêu cầu HTTP của người dùng chỉ khởi tạo một loại service, một Dbcontext trong suốt quá trinh xứ lý yêu cầu. Vì mỗi yêu cầu đến website có thể có nhiều câu truy vấn và mỗi câu truy vấn này thông thường sẽ khởi tạo một DbContext, điều này làm cho website sẽ chậm. Việc đảm bảo trong suốt quá trinh xử lý yêu cầu HTTP chỉ có một DbContext, một loại service duy nhất, một UnitOfWork duy nhất sẽ tiết kiệm thời gian khởi tạo làm cho hệ thống phản hồi nhanh hơn.

### Đăng ký các lớp



Trong đề tài này các lớp được đăng ký trong IoC container là các lớp Controller, Service, Repository, UnitOfWork thuộc nhiều dự án và autoface module khác nhau để khi lớp phụ thuộc được khởi tạo thì các lớp đã đăng ký này sẽ được đưa vào từ IoC Container.

# Kết luận

Thông qua đề tài em được cũng cố nhiều kiến thức các kiến thức phục vụ cho việc phát triển một website như HTML, CSS, Jquery, hướng đối tượng. Được tìm hiểu kiến thức về nền tảng xây dựng trang web là ASP.NET. Được áp dụng những mô hình phát triển phần mềm như mô hình MVC, mô hình 3 lớp và các mẫu thiết kết. Ngoài ra em được có cái nhìn thực về quá trinh phát triển một website trên thực tế. Biết sử dụng các công cụ hỗ trợ phát triển dự á để có thể thực hiện các dự án tiếp theo.

# Tài liệu tham khảo

https://support.microsoft.com/vi-vn/help/2645095

https://www.dammio.com/2016/11/04/entity-framework-phan-1-gioi-thieu-entity-framework

https://www.codeproject.com/Articles/36847/Three-Layer-Architecture-in-C-NET-2

https://techtalk.vn/mo-hinh-3-lop-co-gi-hay.html

https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity

https://tedu.com.vn/lap-trinh-aspnet/ket-hop-unit-of-work-va-repository-pattern-trong-aspnet-mvc-37.html

https://en.wikipedia.org/wiki/Service\_layer\_pattern

https://techtalk.vn/giai-thich-ve-dependency-injection-pattern-su-dung-laravel-5.html