KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH**

**HỌC KỲ I, NĂM HỌC 2024-2025**

**ÁP DỤNG FRAMEWORK ASP.NET CORE**

**XÂY DỰNG WEBSITE BÁN MÁY TÍNH**

Giảng viên hướng dẫn:

Ngô Thanh Huy

Sinh viên thực hiện:

Trần Bá Hiếu – 110121024

***Trà Vinh, .. tháng .. năm 2024***

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH**

**HỌC KỲ I, NĂM HỌC 2024-2025**

**ÁP DỤNG FRAMEWORK ASP.NET CORE**

**XÂY DỰNG WEBSITE BÁN MÁY TÍNH**

Giảng viên hướng dẫn:

Ngô Thanh Huy

Sinh viên thực hiện:

Trần Bá Hiếu – 110121024

***Trà Vinh, .. tháng .. năm 2024***

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

*Trà Vinh, ngày …. . tháng …… năm ……*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

**NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG**

*Trà Vinh, ngày …. . tháng …… năm ……*

**Thành viên hội đồng**

*(Ký tên và ghi rõ họ tên)*

**LỜI CẢM ƠN**

Lời nói đầu, nhóm chúng em xin cảm ơn thầy Ngô Thanh Huy đã hỗ trợ, hướng dẫn tận tình chúng em trong thời gian xây dựng đồ án chuyên ngành, những kiến thức mà thầy đã dạy và hướng dẫn em sẽ là hành trang quý báo trên con đường học vấn và phát triển sự nghiệp tương lai rộng mở của em. Thầy đã luôn kiên nhẫn, nhiệt tình trong việc truyền đạt kiến thức và kinh nghiệm quý báo, giúp chúng em vượt qua những khó khăn và thử thách trong quá trình học tập và nghiên cứu.

Những lời khuyên, góp ý của thầy không chỉ là kim chỉ nam cho sự phát triển của đồ án chuyên ngành này mà còn là nguồn động viên, khích lệ tinh thần lớn lao cho em.

Em xin hứa sẽ tiếp tục nỗ lực không ngừng để không phụ lòng thầy đã dành cho em.

Xin chân thành cảm ơn thầy.

Trà Vinh, .. Tháng .. năm 2024

**Trần Bá Hiếu**

**MỤC LỤC**

**DANH MỤC HÌNH ẢNH**

# TÓM TẮT NIÊN LUẬN

**Vấn đề nghiên cứu:**

**-** Áp dụng framework asp.net core xây dựng website bán máy tính.

**Các hướng tiếp cận:**

+ Tìm hiểu về ngôn ngữ C# và Asp.net core framework.

+ Tìm hiểu mô hình MVC.

+ Tiến hành xây dựng View, Model và Controller.

+ Kiểm tra hiệu năng của Website.

**Một số kết quả đạt được:**

+ Website hoạt động ổn định đảm bảo các chức năng cần thiết.

+ Kiểm thử và đánh giá website khi hoạt động.

**Kết cấu của bài báo cáo:**

* **Chương 1:** Cơ sở lý thuyết.
* **Chương 2:** Thực nghiệm.
* **Chương 3:** Kết luận.

# TỔNG QUAN

# Tổng quan về vấn đề

Trong thời đại công nghệ hiện nay, việc phát triển các website thương mại điện tử trở nên rất quan trọng, đặc biệt là đối với các ngành hàng như bán máy tính, nơi người tiêu dùng ngày càng có nhu cầu mua sắm trực tuyến. Đồ án này tập trung vào việc xây dựng một website bán máy tính sử dụng framework ASP.NET Core, một công nghệ mạnh mẽ và phổ biến để phát triển các ứng dụng web hiện đại.

Mục tiêu của đề tài là thiết kế và phát triển một website bán máy tính, giúp khách hàng dễ dàng tìm kiếm và mua sản phẩm qua các tính năng như lọc sản phẩm, giỏ hàng, thanh toán trực tuyến, và quản lý tài khoản người dùng. Website sẽ được xây dựng với giao diện thân thiện và dễ sử dụng, đồng thời đảm bảo tính bảo mật và hiệu suất cao.

Công nghệ ASP.NET Core sẽ được áp dụng để phát triển website, vì đây là một framework mã nguồn mở, dễ dàng triển khai và có khả năng mở rộng linh hoạt. ASP.NET Core cung cấp các công cụ mạnh mẽ như Entity Framework để tương tác với cơ sở dữ liệu, Razor Pages để tạo giao diện động và các API RESTful để hỗ trợ kết nối giữa frontend và backend.

Trong quá trình phát triển, một số vấn đề quan trọng cần được giải quyết, bao gồm:

* **Quản lý sản phẩm**: Cần xây dựng hệ thống quản lý danh mục sản phẩm, bao gồm thông tin chi tiết về máy tính, giá cả, và các lựa chọn cấu hình.
* **Quản lý đơn hàng**: Hệ thống phải đảm bảo quá trình đặt hàng và thanh toán diễn ra suôn sẻ và an toàn.
* **Tối ưu hóa trải nghiệm người dùng**: Website cần được tối ưu hóa để nhanh chóng và dễ sử dụng trên cả desktop và mobile.
* **Bảo mật và xác thực**: Cần có hệ thống bảo mật để bảo vệ thông tin người dùng và giao dịch.

Từ đó, đề tài này không chỉ giúp nâng cao kỹ năng lập trình web với ASP.NET Core mà còn mang lại một giải pháp thực tiễn cho việc phát triển các website thương mại điện tử trong lĩnh vực bán máy tính.

# Hướng giải quyết

Để giải quyết các vấn đề đã đề cập trong tổng quan, mô hình MVC (Model-View-Controller) của ASP.NET Core sẽ được áp dụng một cách linh hoạt. Mô hình MVC giúp phân chia rõ ràng các thành phần của ứng dụng, làm cho việc phát triển và bảo trì dễ dàng hơn, đồng thời tối ưu hóa trải nghiệm người dùng và khả năng mở rộng của hệ thống.

*Một vài lớp ví dụ để có thể có cái nhìn tổng quan hơn về hướng giải quyết cho việc xây dựng Website:*

**Mô hình (Model)**: Mô hình đại diện cho dữ liệu của ứng dụng và các quy tắc nghiệp vụ. Trong dự án website bán máy tính, chúng ta sẽ xây dựng các lớp Model để quản lý thông tin về sản phẩm (máy tính), khách hàng, đơn hàng và các chức năng khác.

* Sản phẩm: Một lớp Product sẽ chứa các thuộc tính như tên, mô tả, giá, ảnh, và các đặc tính kỹ thuật khác của máy tính.
* Đơn hàng: Lớp Order sẽ lưu trữ thông tin về các đơn hàng, bao gồm các sản phẩm trong giỏ hàng, trạng thái thanh toán và thông tin vận chuyển.
* Khách hàng: Lớp User sẽ lưu trữ thông tin người dùng, bao gồm tên, email, mật khẩu, và địa chỉ giao hàng.

**Giao diện người dùng (View)**: Giao diện người dùng sẽ được xây dựng bằng Razor Views trong ASP.NET Core, giúp tách biệt logic hiển thị khỏi logic xử lý. Các trang chính của website sẽ bao gồm:

* Trang chủ: Hiển thị các sản phẩm nổi bật và các danh mục sản phẩm.
* Trang chi tiết sản phẩm: Cung cấp thông tin chi tiết về một sản phẩm, bao gồm các đặc tính kỹ thuật, hình ảnh, và nút "Thêm vào giỏ hàng".
* Giỏ hàng và thanh toán: Người dùng có thể xem và chỉnh sửa các sản phẩm trong giỏ hàng, sau đó thanh toán trực tuyến.
* Trang quản lý đơn hàng: Người quản trị có thể xem và quản lý các đơn hàng đã được đặt.

**Bộ điều khiển (Controller)**: Các Controller sẽ xử lý logic và tương tác giữa Model và View. Ví dụ:

* ProductController: Xử lý các yêu cầu về danh sách sản phẩm, chi tiết sản phẩm, và tìm kiếm sản phẩm.
* CartController: Xử lý các chức năng liên quan đến giỏ hàng, như thêm sản phẩm, xóa sản phẩm, tính tổng tiền.
* OrderController: Xử lý quá trình thanh toán và quản lý đơn hàng.
* AccountController: Quản lý đăng nhập, đăng ký và thông tin tài khoản người dùng.

**Tính năng quản lý và bảo mật**: Để đảm bảo bảo mật cho website, các tính năng như xác thực người dùng, phân quyền và bảo vệ thông tin khách hàng sẽ được triển khai. ASP.NET Core cung cấp các tính năng như ASP.NET Identity để xử lý việc đăng nhập, đăng ký, và phân quyền người dùng.

**Xác thực và phân quyền**: Sử dụng ASP.NET Core Identity để xử lý việc đăng ký và đăng nhập người dùng. Phân quyền sẽ đảm bảo rằng chỉ người quản trị mới có thể truy cập vào các tính năng quản lý sản phẩm và đơn hàng.

**Bảo mật thanh toán**: Để bảo vệ các giao dịch thanh toán, việc tích hợp với các cổng thanh toán như PayPal hoặc Stripe sẽ được thực hiện để đảm bảo tính bảo mật cao.

**Tối ưu hóa hiệu suất**: Để đảm bảo website hoạt động mượt mà, đặc biệt khi có lượng người dùng lớn, các kỹ thuật tối ưu hóa như caching, lazy loading, và phân trang sẽ được áp dụng.

# NGHIÊN CỨU LÝ THUYẾT

# Ngôn ngữ lập trình C# ( C Sharp )

C Sharp hay được biết đến C# là một ngôn ngữ lập trình do Microsoft phát triển, thuộc dòng ngôn ngữ lập trình C. Nó được sử dụng chủ yếu trong môi trường .NET, giúp xây dựng các ứng dụng trên Windows, web, và di động. C# dễ học và có cú pháp rõ ràng, giống như các ngôn ngữ lập trình khác như Java và C++ nhưng lại đơn giản hơn, dễ hiểu hơn.

C# hỗ trợ các đặc điểm như lập trình hướng đối tượng, giúp người lập trình dễ dàng quản lý và tái sử dụng mã nguồn. Nó cũng có tính năng quản lý bộ nhớ tự động thông qua garbage collection, giảm bớt những lo ngại về việc giải phóng bộ nhớ thủ công.

Với tính ổn định và khả năng tích hợp tốt với các công nghệ khác của Microsoft, C# được sử dụng rộng rãi trong việc phát triển các ứng dụng doanh nghiệp, phần mềm desktop, ứng dụng web và game.

Tóm lại, C# là một ngôn ngữ mạnh mẽ nhưng dễ tiếp cận, phù hợp với nhiều loại ứng dụng và là công cụ lý tưởng cho các dự án phần mềm hiện đại.

**Ví dụ khai báo Model cho thực thể User:**

public class UserModel

{

public int Id { get; set; }

[Required(ErrorMessage = "Yêu cầu nhập Tên đăng nhập")]

public string UserName { get; set; }

[DataType(DataType.Password), Required(ErrorMessage = "Yêu cầu nhập Mật khẩu")]

public string Password { get; set; }

[Required(ErrorMessage = "Yêu cầu nhập Email"), EmailAddress]

public string Email { get; set; }

}

# ASP.NET Framework và sự khác nhau giữa ASP.NET Core và ASP.NET

ASP.NET Core là một framework phát triển ứng dụng web mã nguồn mở và đa nền tảng (cross-platform) được Microsoft phát triển. Nó là phiên bản cải tiến của ASP.NET và được thiết kế lại để chạy trên nhiều hệ điều hành khác nhau như Windows, Linux và macOS, thay vì chỉ giới hạn trên Windows như trước đây.

Dưới đây là sự khác biệt chính giữa ASP.NET Core và ASP.NET để có cái nhìn tổng quát hơn về Framework này:

1. **Đa nền tảng (Cross-Platform):**

+ ASP.NET: Chỉ chạy trên hệ điều hành Windows.

+ ASP.NET Core: Được thiết kế để chạy trên nhiều nền tảng khác nhau, bao gồm Windows, Linux và macOS. Điều này giúp các nhà phát triển có thể xây dựng và triển khai ứng dụng web của mình trên bất kỳ nền tảng nào.

1. **Hiệu suất:**

+ ASP.NET: Mặc dù có hiệu suất tốt nhưng không được tối ưu như ASP.NET Core.

+ ASP.NET Core: Cải thiện rất nhiều về hiệu suất so với ASP.NET. Nó sử dụng một mô hình xử lý nhẹ và hiện đại hơn, giúp các ứng dụng chạy nhanh hơn và sử dụng tài nguyên hệ thống hiệu quả hơn.

1. **Kiến trúc:**

+ ASP.NET: Được xây dựng dựa trên mô hình Web Forms, MVC, và Web API, có một số phần không linh hoạt và dễ bị lỗi khi phát triển ứng dụng lớn.

+ ASP.NET Core: Hỗ trợ một kiến trúc sạch hơn và linh hoạt hơn, chủ yếu tập trung vào MVC và API, loại bỏ những phần thừa thãi như Web Forms, và giúp cấu trúc ứng dụng dễ duy trì và mở rộng hơn.

1. **Tích hợp Dependency Injection:**

*Dependency Injection là một kỹ thuật mà trong đó một đối tượng (class) không tự tạo ra các phụ thuộc của mình (dependencies), mà thay vào đó, các phụ thuộc này sẽ được "chuyển vào" đối tượng từ bên ngoài thông qua một cơ chế nào đó (chẳng hạn như qua constructor, setter hoặc interface).*

+ ASP.NET: Dependency Injection (DI) không phải là phần cốt lõi và phải cài đặt thêm để sử dụng.

+ ASP.NET Core: DI được tích hợp sẵn trong framework, giúp quản lý các phụ thuộc giữa các lớp dễ dàng hơn, và giảm sự phụ thuộc chặt chẽ giữa các thành phần trong ứng dụng.

1. **Cấu hình và Tùy biến:**

+ ASP.NET: Cấu hình chủ yếu thông qua các tệp cấu hình như web.config.

+ ASP.NET Core: Sử dụng một hệ thống cấu hình linh hoạt, có thể đọc cấu hình từ nhiều nguồn như tệp JSON, tệp môi trường, biến môi trường, hoặc thậm chí từ các dịch vụ trong đám mây.

1. **Khởi động ứng dụng:**

+ ASP.NET: Khởi động và cấu hình ứng dụng khá phức tạp, nhất là khi cần thực hiện các thay đổi lớn.

+ ASP.NET Core: Quá trình khởi động ứng dụng đơn giản hơn và linh hoạt hơn, nhờ vào việc sử dụng các tệp cấu hình và cách quản lý dịch vụ dễ dàng hơn.

1. **Tối ưu hóa cho Microservices:**

+ ASP.NET: Không được thiết kế tối ưu cho mô hình microservices.

+ ASP.NET Core: Được thiết kế đặc biệt để hỗ trợ phát triển các ứng dụng microservices, với các tính năng như hỗ trợ Docker và Kubernetes, giúp dễ dàng triển khai các dịch vụ độc lập.

1. **Quản lý bộ nhớ và Garbage Collection:**

*Garbage Collection (GC) là một cơ chế tự động trong lập trình nhằm quản lý bộ nhớ, đảm nhận việc giải phóng bộ nhớ mà chương trình không còn sử dụng nữa.*

+ ASP.NET: Hệ thống garbage collection trong ASP.NET có thể kém tối ưu hơn khi phải xử lý các ứng dụng phức tạp.

+ ASP.NET Core: Tối ưu hóa hơn về quản lý bộ nhớ và garbage collection, cải thiện hiệu suất và độ ổn định trong các ứng dụng quy mô lớn.

1. **Cộng đồng và Mã nguồn mở:**

+ ASP.NET: Là sản phẩm đóng của Microsoft, mặc dù có cộng đồng đóng góp nhưng không mã nguồn mở hoàn toàn.

+ ASP.NET Core: Là mã nguồn mở hoàn toàn và phát triển với sự tham gia của cộng đồng. Điều này giúp ASP.NET Core phát triển nhanh chóng và liên tục nhận được các cải tiến từ cộng đồng lập trình viên.

1. **Tính năng Web API và SignalR:**

+ ASP.NET: Các tính năng như Web API và SignalR có thể được sử dụng nhưng không được tối ưu hóa cho các ứng dụng hiện đại.

+ASP.NET Core: Cung cấp các cải tiến và tính năng mới cho Web API và SignalR, giúp phát triển các ứng dụng realtime và API RESTful hiệu quả hơn.

🡺 Nhìn chung có thể thấy ASP.NET Core là phiên bản hiện đại hơn, nhanh hơn, và linh hoạt hơn của ASP.NET Nó giúp bạn phát triển các ứng dụng web với hiệu suất tối ưu, khả năng tương thích đa nền tảng, và dễ dàng bảo trì, mở rộng trong tương lai. Nếu bạn bắt đầu phát triển các ứng dụng web mới, ASP.NET Core là sự lựa chọn tốt nhất, còn ASP.NET vẫn có thể hữu ích trong những hệ thống cũ hoặc những ứng dụng không yêu cầu tính năng đa nền tảng.

# Mô hình MVC (Model-View-Controller)

Mô hình MVC (Model-View-Controller) là một kiểu kiến trúc phần mềm phổ biến trong phát triển ứng dụng web, giúp tách biệt các thành phần của ứng dụng để dễ dàng quản lý, bảo trì và mở rộng. Mô hình này phân chia ứng dụng thành ba phần riêng biệt: Model, View, và Controller, mỗi phần đảm nhiệm một chức năng cụ thể.

### Khái quát về cấu trúc mô hình MVC

Mô hình MVC giúp chia nhỏ ứng dụng thành ba phần cơ bản, nhằm tăng tính mô-đun và dễ dàng mở rộng. Cách phân chia này cũng giúp tăng tính tái sử dụng mã nguồn và dễ dàng kiểm thử.

**Model (M):** Là phần mô tả dữ liệu và logic nghiệp vụ của ứng dụng. Model quản lý dữ liệu, các quy tắc, và trạng thái của ứng dụng.

**View (V):** Là phần giao diện người dùng, giúp hiển thị dữ liệu từ Model ra màn hình. View chịu trách nhiệm về việc trình bày và giao diện của ứng dụng.

**Controller (C):** Là phần trung gian giữa Model và View. Controller nhận các yêu cầu từ người dùng (như nhấn nút, nhập liệu), xử lý các yêu cầu đó và thay đổi Model hoặc cập nhật View.

### Cách hoạt động của mô hình MVC

Khi một người dùng tương tác với ứng dụng (ví dụ: nhấn nút, điền biểu mẫu), các bước sau đây sẽ xảy ra:

1. Controller nhận yêu cầu từ người dùng (ví dụ: nhấn nút, chọn tùy chọn).
2. Controller xử lý yêu cầu và yêu cầu Model lấy hoặc cập nhật dữ liệu (có thể là từ cơ sở dữ liệu hoặc từ các nguồn khác).
3. Model thực hiện logic nghiệp vụ, cập nhật hoặc trả về dữ liệu cho Controller.
4. Controller nhận dữ liệu từ Model, quyết định loại giao diện hoặc hành động cần thực hiện, và yêu cầu View cập nhật giao diện người dùng.
5. View hiển thị dữ liệu cho người dùng, hoàn tất quá trình.

### Cấu trúc mô hình MVC

Model

Mô tả: Là đại diện cho dữ liệu và trạng thái của ứng dụng. Model không quan tâm đến cách hiển thị dữ liệu hoặc cách thức tương tác của người dùng với dữ liệu đó.

**Chức năng:**

* Quản lý dữ liệu, kết nối cơ sở dữ liệu.
* Thực thi các thao tác nghiệp vụ (chẳng hạn như tính toán, xác thực dữ liệu).
* Thông báo cho Controller khi dữ liệu đã thay đổi.

Ví dụ: Một lớp Product có các thuộc tính như tên sản phẩm, giá cả, số lượng trong kho.

View

Mô tả: Là giao diện người dùng, nơi dữ liệu từ Model được hiển thị cho người dùng. View chỉ chịu trách nhiệm trình bày dữ liệu, không chứa logic nghiệp vụ.

**Chức năng:**

* Hiển thị thông tin mà Controller cung cấp.
* Cung cấp các thành phần giao diện người dùng (button, form, text field) để người dùng tương tác.

Ví dụ: Một trang web hiển thị danh sách sản phẩm dưới dạng bảng hoặc thẻ.

Controller

Mô tả: Là cầu nối giữa Model và View. Controller nhận các yêu cầu từ người dùng và xử lý chúng.

**Chức năng:**

* Xử lý yêu cầu người dùng (thường là các sự kiện từ giao diện).
* Cập nhật Model (thêm, sửa, xóa dữ liệu).
* Lựa chọn View và chuyển dữ liệu tới View.

**Đoạn mã ví dụ về Controller thêm sản phẩm:**

[HttpPost]

public IActionResult Create(Product product)

{

if (ModelState.IsValid)

{

\_context.Add(product);

\_context.SaveChanges();

return RedirectToAction(nameof(Index));

}

return View(product);

}

### So sánh mô hình MVC và các mô hình khác

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mô hình** | **Cấu trúc chính** | **Ưu điểm** | **Nhược điểm** | **Phù hợp với ứng dụng** |
| MVC | Model, View, Controller | Tách biệt rõ ràng, dễ bảo trì, dễ kiểm thử | Phức tạp hơn cho ứng dụng nhỏ, cần mã nguồn kết nối | Ứng dụng web, giao diện phức tạp |
| MVVM | Model, View, ViewModel | Tách biệt logic UI và nghiệp vụ, dễ kiểm thử, data binding | Phức tạp hơn, cần hiểu về ViewModel | Ứng dụng desktop, mobile, WPF, Xamarin |
| MVP | Model, View, Presenter | Dễ kiểm thử, logic nghiệp vụ tách biệt, giúp kiểm soát giao diện | Cần kết nối giữa Presenter và View | Ứng dụng desktop, mobile |
| SOA | Services, Clients | Tái sử dụng, phân tán tốt, mở rộng dễ dàng | Phức tạp trong quản lý, yêu cầu giao tiếp giữa các dịch vụ | Các hệ thống phân tán, Microservices |
| Clean Architecture | Entities, Use Cases, Interface Adapters, Frameworks | Tách biệt các tầng rõ ràng, dễ bảo trì và mở rộng, kiểm thử độc lập, linh hoạt trong việc thay đổi công nghệ | Phức tạp và dư thừa cho ứng dụng đơn giản | Các ứng dụng lớn, enterprise, microservices, hệ thống phức tạp |

# Một số phương thức, khái niệm, công nghệ/công cụ... khác được sử dụng

+ Phương thức **HTTP (HTTP Methods)** là các hành động được sử dụng trong giao thức HTTP (HyperText Transfer Protocol) để tương tác với tài nguyên trên một server. Mỗi phương thức HTTP xác định kiểu yêu cầu mà client gửi tới server để thực hiện một tác vụ cụ thể trên tài nguyên (như thêm, sửa, xóa, hoặc lấy dữ liệu).

# THỰC NGHIỆM

# KẾT LUẬN

# TÀI LIỆU THAM KHẢO