A black and white rectangular frame

Description automatically generated

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**A blue and yellow logo

Description automatically generated

**Báo Cáo Bài Tập Lớn**

**Đề tài: Tìm hiểu về .NET Core**

**Giáo viên hướng dẫn:** Nguyễn Trọng Phúc

**Lớp : CNTT3-K62**

**Nhóm sinh viên thực hiện:** Nhóm 12

1. Lã Hữu Duy – 211203862
2. Hoàng Văn Hùng – 211202497
3. Đàm Văn Đức – 211203481

**Hà Nội - 2024**

# Mục lục

[Mục lục 2](#_Toc178365572)

[Lời mở đầu 2](#_Toc178365573)

[1. Giới thiệu đề tài 4](#_Toc178365574)

[2. Lịch sử phát triển 5](#_Toc178365575)

[3. Giới thiệu về .NET Core 10](#_Toc178365576)

[3.1 Các thành phần chính 10](#_Toc178365577)

[3.2 Kiến trúc của .NET Core 15](#_Toc178365578)

[3.3 Ưu điểm & nhược điểm 17](#_Toc178365579)

[4. Các dự án tiêu biểu trong .NET Core 20](#_Toc178365580)

[5. So sánh và đánh giá 21](#_Toc178365581)

[6. Tài liệu tham khảo 24](#_Toc178365582)

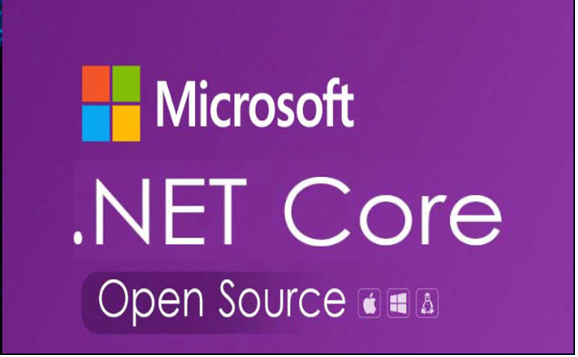
[7. Kết luận 25](#_Toc178365583)

# Lời mở đầu

Trong thời đại công nghệ số phát triển không ngừng, việc phát triển các ứng dụng đa nền tảng và hiệu quả trở nên ngày càng quan trọng. .NET Core, một nền tảng mã nguồn mở của Microsoft, đã trở thành một trong những công nghệ chủ đạo hỗ trợ việc phát triển các ứng dụng hiện đại. Với tính năng đa nền tảng, hiệu suất cao, và khả năng tích hợp mạnh mẽ với các công cụ hiện đại như Docker và Kubernetes, .NET Core cung cấp một môi trường phát triển linh hoạt và mạnh mẽ, phục vụ cho nhiều loại ứng dụng từ web, API đến các dịch vụ cloud-native.

Trong báo cáo này, chúng tôi sẽ đi sâu vào nghiên cứu và triển khai ứng dụng trên nền tảng .NET Core, khám phá các thành phần kiến trúc chính, đặc điểm nổi bật của nền tảng này, và các công nghệ đi kèm. Báo cáo không chỉ giới thiệu về .NET Core mà còn trình bày quá trình thực hiện một bài tập lớn liên quan, bao gồm phân tích, thiết kế, và triển khai một ứng dụng thực tế. Thông qua đó, chúng tôi hy vọng có thể minh chứng được sức mạnh và tiềm năng của .NET Core trong phát triển ứng dụng hiện đại.

# Giới thiệu đề tài



* **Đặt vấn đề**
* Giới thiệu về bối cảnh:

.NET Core là một nền tảng phát triển phần mềm mã nguồn mở, đa nền tảng được Microsoft giới thiệu như một sự kế thừa hiện đại của .NET Framework. Ra đời với mục tiêu cung cấp một nền tảng phát triển linh hoạt, có khả năng chạy trên nhiều hệ điều hành như Windows, macOS và Linux, .NET Core đã nhanh chóng trở thành sự lựa chọn hàng đầu cho các nhà phát triển muốn xây dựng các ứng dụng có hiệu năng cao và dễ dàng triển khai trên nhiều môi trường khác nhau.

* Xác định vấn đề:

Mặc dù .NET Core mang đến nhiều lợi ích, nhưung sự đa dạng và linh hoạt của nó cũng đặt ra không ít thách thức cho các nhà phát triển, đặc biệt là những người đã quen thuộc với .NET Framework. Nhiều người vẫn còn phân vân về cách thức tận dụng tối đa các tính năng của .NET Core, và liệu nó có phải sự lựa chọn đúng đăng cho các dự án của họ hay không. Vấn đề chính là làm sao để tận dụng hết tiềm năng của .NET Core mà không bị mất đi những lợi ích từ các nền tảng trước đây.

* Mục tiêu:

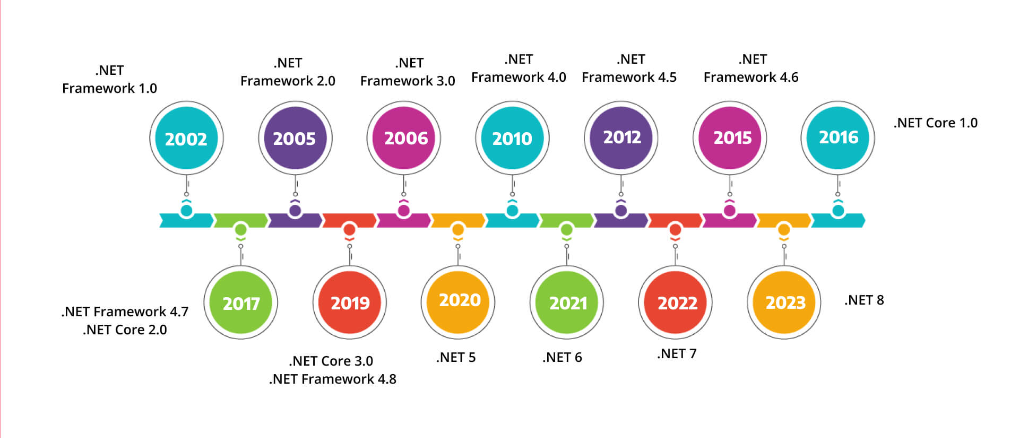
Bài báo cáo này nhằm mục đích cung cấp một cái nhìn tổng quan về .NET Core, làm rõ các tính năng nổi bật và những khác biệt so với .NET Framework. Ngoài ra, bài viết sẽ hướng dẫn các nhà phát triển cách tiếp cận và sử dụng .NET Core trong các dự án thực tế, từ đó giúp họ tối ưu hoá hiệu suất và khả năng mở rộng của ứng dụng.

* Ý nghĩa và tầm quan trọng:

Việc hiểu rõ và áp dụng hiệu quả .NET Core sẽ giúp các nhà phát triển không chỉ bắt kịp xu hướng công nghệ hiện đại, mà còn tăng cường khả năng cạnh tranh trong môi trường phát triển phần mềm. Điều này đặc biệt quan trọng trong bối cảnh các ứng dụng cần đáp ứng được yêu cầu về hiệu năng cao, khả năng triển khai linh hoạt và sự hỗ trợ lâu dài trên nhiều nền tảng khác nhau.

# Lịch sử phát triển

**.NET Core** là một nền tảng phát triển phần mềm mã nguồn mở và đa nền tảng do Microsoft phát triển. Nó được thiết kế để thay thế và mở rộng các tính năng của **.NET Framework** nhằm hỗ trợ việc xây dựng các ứng dụng hiện đại, bao gồm web, desktop, cloud, microservices, API, dịch vụ IoT, và các ứng dụng game. .NET Core ra mắt lần đầu vào năm 2016 và là sự phát triển vượt bậc so với .NET Framework trong việc tối ưu hóa tính linh hoạt, hiệu suất và tính đa nền tảng.

****

* **.NET Core 1.0**
* **Ra mắt**: Tháng 6 năm 2016, .NET Core 1.0 đánh dấu bước chuyển lớn khi Microsoft quyết định mở mã nguồn nền tảng .NET để hỗ trợ đa nền tảng (Windows, macOS, Linux).
* **Công nghệ đi kèm**:
  + **ASP.NET Core**: Phiên bản cải tiến từ ASP.NET để chạy trên mọi nền tảng.
  + **Entity Framework Core**: ORM nhẹ để truy cập cơ sở dữ liệu với hiệu năng cao.
  + **CLI (.NET Command-Line Interface)**: Giúp các nhà phát triển sử dụng dòng lệnh để tạo, biên dịch, và chạy ứng dụng.
* **.NET Core 2.0**
* **Ra mắt**: Tháng 8 năm 2017.
* **Công nghệ đi kèm**:
  + **.NET Standard 2.0**: Đưa ra một tập hợp các API chung cho tất cả các nền tảng .NET (bao gồm .NET Framework, .NET Core, Xamarin), giúp chia sẻ code dễ dàng hơn giữa các ứng dụng.
  + **ASP.NET Core 2.0**: Cải tiến cho framework web, bổ sung nhiều tính năng mới như Razor Pages để phát triển web nhanh chóng.
* **.NET Core 2.1**
* **Ra mắt**: Tháng 5 năm 2018 (phiên bản LTS - Long Term Support).
* **Công nghệ đi kèm**:
  + **SignalR**: Framework cho các ứng dụng thời gian thực, hỗ trợ WebSockets và các giao thức thời gian thực khác.
  + **HTTP Client Factory**: Tăng cường quản lý các kết nối HTTP, giảm chi phí khi tạo các yêu cầu HTTP.
  + **Span<T>**: Cải tiến hiệu suất và quản lý bộ nhớ tốt hơn cho các thao tác trên dữ liệu.
* **.NET Core 2.2**
* **Ra mắt**: Tháng 12 năm 2018.
* **Công nghệ đi kèm**:
  + **API gRPC**: Hỗ trợ gRPC, một giao thức cho việc phát triển dịch vụ API nhanh và hiệu quả, đặc biệt hữu ích trong các ứng dụng microservices.
  + **Entity Framework Core 2.2**: Cải thiện hiệu suất và hỗ trợ tốt hơn cho các cơ sở dữ liệu phi quan hệ.
* **.NET Core 3.0**
* **Ra mắt**: Tháng 9 năm 2019.
* **Công nghệ đi kèm**:
  + **Hỗ trợ Windows Desktop**: .NET Core 3.0 mang lại khả năng phát triển ứng dụng desktop cho **Windows Forms** và **WPF** trên nền tảng .NET Core.
  + **Blazor**: Cho phép phát triển ứng dụng web bằng C# thay vì JavaScript, hỗ trợ cả Blazor Server và Blazor WebAssembly.
  + **Cải thiện hiệu suất**: Nhiều cải tiến cho các ứng dụng web và desktop.
* **.NET 5.0**
* **Ra mắt**: Tháng 11 năm 2020.
* **Công nghệ đi kèm**:
  + **Hợp nhất .NET**: Hợp nhất **.NET Core** và **.NET Framework** thành một nền tảng duy nhất, loại bỏ tên "Core" và chỉ gọi là **.NET**.
  + **gRPC và HTTP/3**: Hỗ trợ giao thức HTTP/3 và các tính năng mới cho **gRPC**, giúp xây dựng API nhanh hơn và tối ưu hơn cho microservices.
  + **Cải tiến hiệu suất**: Các cải tiến trên JIT (Just-In-Time Compilation), AOT (Ahead-of-Time Compilation), và quản lý bộ nhớ giúp nâng cao hiệu suất cho ứng dụng.
* **.NET 6.0**
* **Ra mắt**: Tháng 11 năm 2021 (phiên bản LTS).
* **Công nghệ đi kèm**:
  + **.NET MAUI (Multi-platform App UI)**: Kế thừa từ Xamarin, cho phép phát triển ứng dụng đa nền tảng (iOS, Android, macOS, Windows) từ một codebase duy nhất.
  + **Blazor Hybrid**: Cho phép xây dựng ứng dụng desktop bằng công nghệ web với Blazor.
  + **Hot Reload**: Tính năng cho phép các nhà phát triển thay đổi code và thấy kết quả tức thì mà không cần khởi động lại ứng dụng.
* **.NET 7.0**
* **Ra mắt**: Tháng 11 năm 2022.
* **Công nghệ đi kèm**:
  + **Cloud-native và microservices**: Tăng cường khả năng phát triển các ứng dụng **cloud-native** và **containerization**.
  + **Cải tiến hiệu suất**: Các tính năng như **Native AOT** được cải thiện, giúp tăng tốc khởi chạy ứng dụng và giảm kích thước file thực thi.
  + **MAUI cải tiến**: MAUI tiếp tục được tối ưu cho phát triển ứng dụng di động và desktop đa nền tảng.
* **.NET 8.0**
* **Ra mắt**: Tháng 11 năm 2023 (phiên bản LTS).
* **Công nghệ đi kèm**:
  + **Blazor Full-stack**: Cải tiến mạnh mẽ để Blazor chạy hoàn toàn trên client (WebAssembly), cung cấp hiệu suất và trải nghiệm người dùng tốt hơn.
  + **Native AOT**: Tối ưu hóa khả năng biên dịch trước, làm cho các ứng dụng nhẹ hơn và khởi động nhanh hơn.
  + **Hỗ trợ API nâng cao**: Tăng cường hỗ trợ cho gRPC và HTTP/3, giúp xây dựng các dịch vụ API hiệu quả cho các hệ thống phân tán.
  + **Hiệu suất cải tiến**: Tập trung vào việc tối ưu hóa cho ứng dụng chạy trong môi trường cloud-native, microservices và container.

Cùng với .Net Core ra đời .Net Framework cũng có 1 số bản cập nhật để theo kịp thời đại.

* **.NET Framework 4.6**
* **Ra mắt:** 20/07/2015
* **Công nghệ đi kèm:**
* **HTTP/2**: Hỗ trợ giao thức HTTP/2 cho các ứng dụng web.
* **RyuJIT**: Trình biên dịch JIT mới cho x64, cải thiện hiệu suất.
* **Đặc điểm nổi bật**:
  + Hỗ trợ **HTTP/2**: Giảm độ trễ và tăng tốc độ tải trang cho ứng dụng web.
  + Cải tiến **JIT Compiler**: Tăng cường hiệu suất cho ứng dụng desktop và server.
* **.NET Framework 4.7**
* **Ra mắt:** 02/05/2017
* **Công nghệ đi kèm:**
  + **.NET Standard 2.0**: Tăng cường khả năng tương thích giữa .NET Framework, .NET Core và Xamarin.
  + **High DPI support**: Cải thiện hỗ trợ màn hình có độ phân giải cao (DPI).
* **Đặc điểm nổi bật**:
  + Hỗ trợ tốt hơn cho màn hình có độ phân giải cao, đặc biệt là trong ứng dụng desktop.
  + Cải thiện tương thích giữa các phiên bản .NET.
* **.NET Framework 4.8**
* **Ra mắt:** 18/04/2019
* **Công nghệ đi kèm:**
  + **Windows Forms/WPF**: Cải thiện hiệu suất và hỗ trợ tốt hơn cho Windows 10.
  + **.NET Core Integration**: Bắt đầu tích hợp một số tính năng từ .NET Core.
* **Đặc điểm nổi bật**:
  + Phiên bản cuối cùng của .NET Framework.
  + Hỗ trợ các tính năng hiện đại trên Windows, nhưng không còn phát triển lớn sau này do sự ra đời của **.NET Core** và **.NET 5+**.

.NET Framework đã đóng vai trò chủ đạo trong phát triển ứng dụng Windows từ 2002. Từ những ứng dụng desktop như WinForms, WPF cho đến ứng dụng web với ASP.NET, .NET Framework đã là một công cụ mạnh mẽ cho các nhà phát triển. Tuy nhiên, từ sau phiên bản 4.8, Microsoft chuyển trọng tâm sang **.NET Core** và **.NET 5+**, nhằm đáp ứng yêu cầu về đa nền tảng, hiệu suất cao hơn và tính mở rộng.

# Giới thiệu về .NET Core

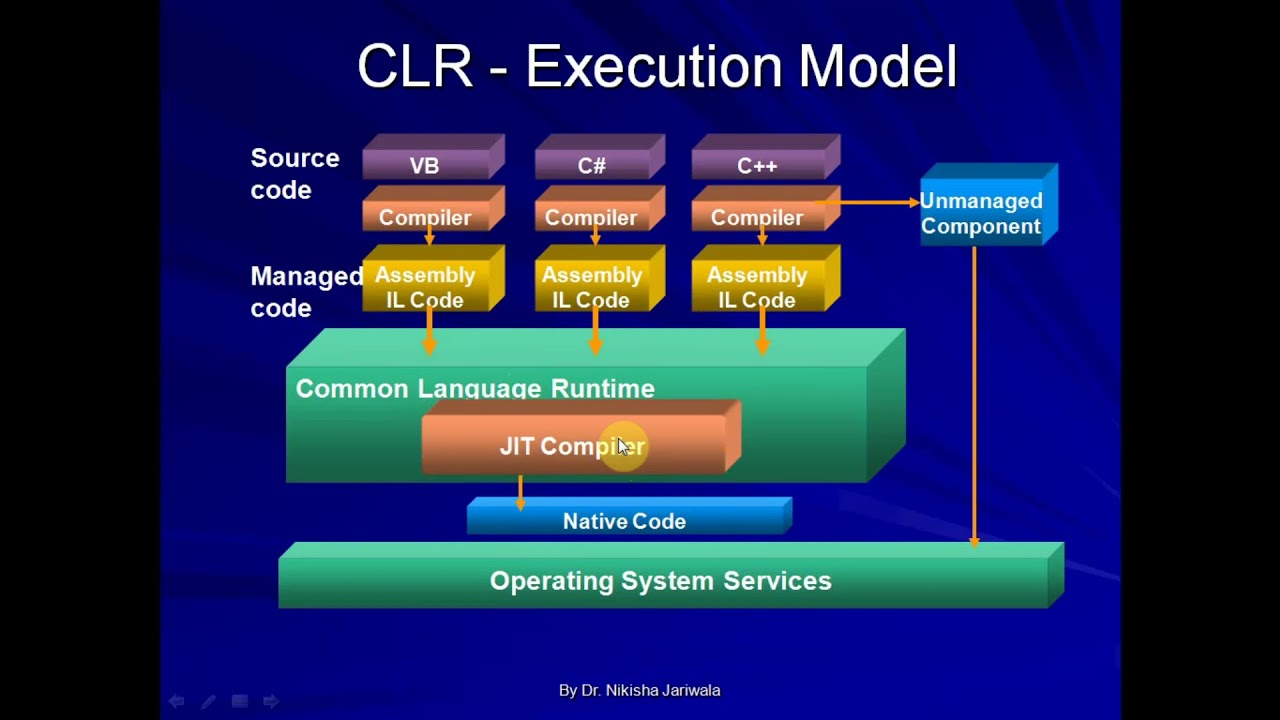
## Các thành phần chính

* **.NET Runtime:**

.NET Runtime là môi trường thực thi ứng dụng dành cho các ứng dụng được viết bằng các ngôn ngữ lập trình trên nền tảng .NET, chẳng hạn như C#, VB.NET, và F#.

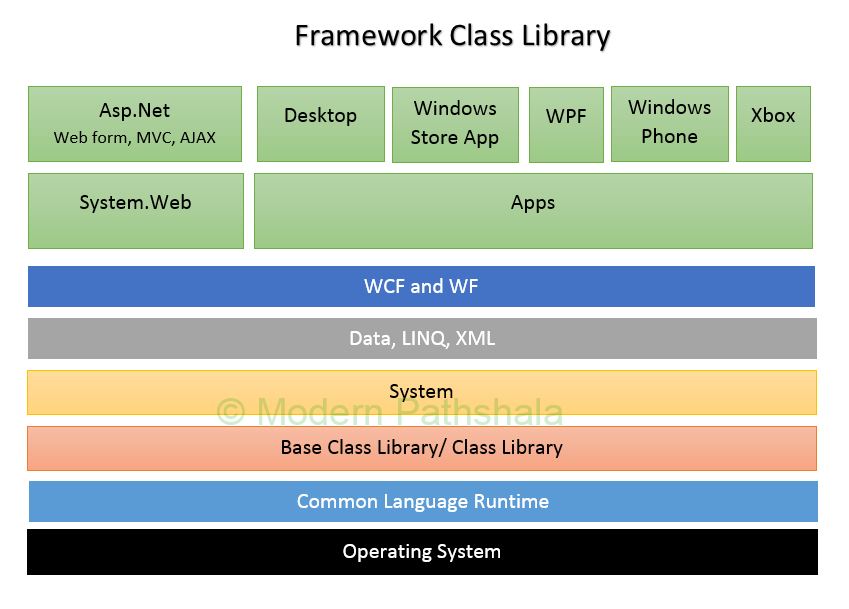
+ CLR (Common Language Runtime): Đây là thành phần cốt lõi của .NET Core, chịu trách nhiệm quản lí việc thực thi mã, bao gồm quản lí bộ nhớ, xử lí ngoại lệ, bảo mật, và nhiều chức năng khác. CLR đảm bảo rằng các ứng dụng được viết bằng các ngôn ngữ khác nhau có thể chạy trên cùng một nền tảng.

+ JIT (Just-In-Time Compiler): biên dịch mã Intermediate Language (IL) thành mã máy ngay tại thời điểm chạy, giúp tối ưu hoá hiệu suất của ứng dụng.



* **Base Class Library(BCL):**

BCL cung cấp một tập hợp các thư viện chuẩn chứa các lớp và phương thức cơ bản, hỗ trợ các tác vụ phổ biến như thao tác chuỗi, truy cập tệp, xử lí dữ liệu và làm việc với các kiểu dữ liệu cơ bản. BCL giúp giảm thiểu việc phải viết lại mã lệnh thông qua việc tái sử dụng các thành phần đã được phát triển sẵn.



* **ASP.NET Core:**

Là một framework được tối ưu hoá để xây dựng các ứng dụng web hiện đại. bao gồm ứng dụng Web, API, và dịch vụ thời gian thực. Nó hỗ trợ cả mô hình lập trình MVC và Razor Pages, đồng thời tích hợp tốt với các công nghệ hiện đại như Angular, React và Blazor.

* **Entity Framework Core (EF Core):**

EF Core là một ORM (Object-Relational Mapper) giúp các nhà phát triển tương tác với cơ sở dữ liệu bằng cách sử dụng các đối tượng trong code thay vì viết các câu lệnh SQL thủ công. EF Core hỗ trợ nhiều loại cơ sở dữ liệu khác nhau như SQL Server, SQLite, MySQL và PostgreSQL.

* **Command-Line Interface (CLI):**

Là công cụ dòng lệnh cung cấp khả năng tạo, xây dựng, chạy và xuất bản các ứng dụng .NET Core. CLI cho phép các nhà phát triển thực hiện các thao tác này mà không cần dùng đến IDE, giúp việc phát triển trở nên linh hoạt hơn, đặc biệt trên các môi trường phát triển khác nhau.

* **Cross-Platform Support:**

.NET Core được thiết kế để hỗ trợ đa nền tảng. cho phép các ứng dụng chạy trên Windows, macOS và Linux. Điều này mở rộng khả năng phát triển và triển khai của ứng dụng .NET Core trên nhiều môi trường khác nhau.

A diagram of a cross platform

Description automatically generated

* **NuGet Package Manager:**

NuGet là hệ thống quản lí gói tích hợp trong .NET Core, cho phép các nhà phát triển cài đặt, quản lí và cập nhật các thư viện và thành phần bên ngoài thông qua các gói NuGet. Điều này giúp quản lí phụ thuộc và chia sẻ mã trở nên dễ dàng hơn.

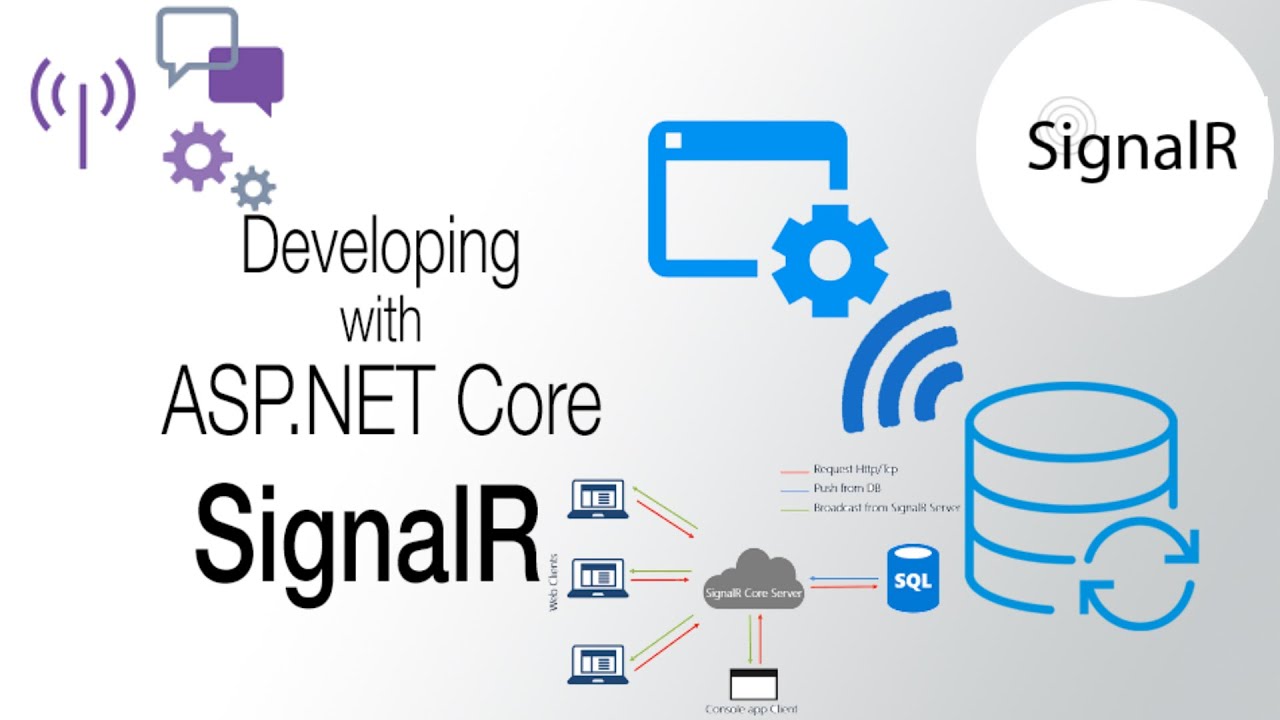
* **gRPC :**

Là framework hiện đại cho việc xây dựng các API hiệu suất cao, sử dụng giao thức RPC (Remote Procedure Call). gRPC hoạt động nhanh và nhẹ, phù hợp cho các ứng dụng microservices và cloud-native.



**SignalR**:

Hỗ trợ xây dựng các ứng dụng thời gian thực như chat, thông báo, hay các ứng dụng yêu cầu cập nhật dữ liệu liên tục giữa client và server.



* **.NET MAUI (Multi-platform App UI):**

**.NET MAUI** là framework phát triển ứng dụng đa nền tảng, kế thừa từ **Xamarin**. Nó cho phép lập trình viên tạo ra các ứng dụng di động (iOS, Android) và desktop (Windows, macOS) chỉ từ một codebase duy nhất. **MAUI** hỗ trợ phát triển giao diện người dùng linh hoạt, đồng thời hỗ trợ các thành phần gốc của mỗi hệ điều hành, giúp tối ưu hóa trải nghiệm người dùng trên mỗi nền tảng.

A screen shot of a black background

Description automatically generated

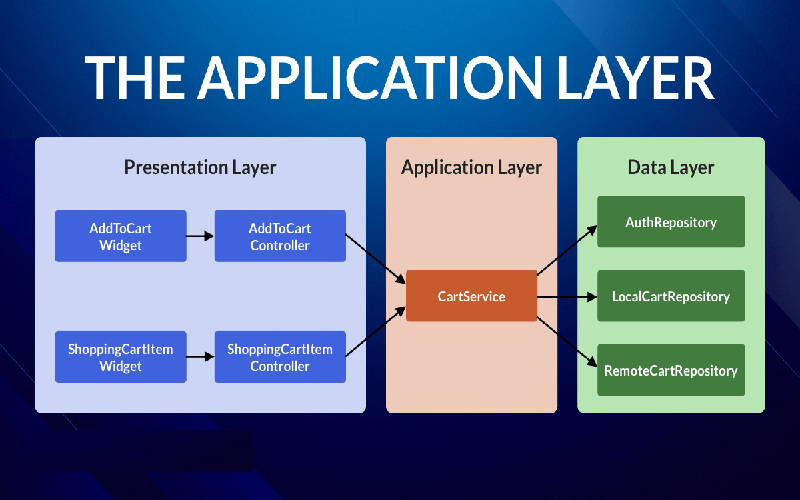
* **Languages:**

.NET Core hỗ trợ nhiều ngôn ngữ lập trình, với C# là ngôn ngữ chính. Các ngôn ngữ khác như F# và VB.NET cũng được hỗ trợ đầy đủ. Giúp các nhà phát triển có nhiều lựa chọn phù hợp với nhu cầu của họ.

**Những thành phần này kết hợp với nhau tạo nên một hệ sinh thái mạnh mẽ cho việc phát triển các ứng dụng hiện đại, từ các ứng dụng web, dịch vụ microservices, đến các ứng dụng desktop và cloud.**

## Kiến trúc của .NET Core

* **Tầng giao diện (Presentation Layer)**
* Đây là tầng đầu tiên, nơi mà người dùng cuối sẽ tương tác với ứng dụng. Mọi giao diện đồ họa, trang web hoặc ứng dụng di động đều thuộc về tầng này. Presentation Layer chịu trách nhiệm hiển thị thông tin từ hệ thống đến người dùng và thu thập dữ liệu nhập từ người dùng để chuyển xuống các tầng dưới. Công nghệ phổ biến trong .NET Core để xây dựng tầng này bao gồm ASP.NET Core MVC, Razor Pages, Blazor, và các frontend frameworks như Angular, React hoặc Vue.js.
* Tầng Presentation cần đảm bảo rằng mọi tương tác của người dùng đều được quản lý hiệu quả và cung cấp trải nghiệm người dùng (User Experience) tốt nhất. Tầng này cũng là nơi thực hiện các kiểm tra sơ bộ như xác thực dữ liệu người dùng trước khi gửi đến các tầng khác.
* **Tầng ứng dụng (Application Layer)**



* Tầng Application đóng vai trò là trung gian giữa tầng Presentation và tầng Business Logic. Nhiệm vụ chính của nó là điều phối các luồng công việc, quản lý các yêu cầu từ người dùng và đảm bảo chúng được xử lý bởi các dịch vụ thích hợp. Nó không trực tiếp xử lý logic nghiệp vụ nhưng có nhiệm vụ gọi các dịch vụ cần thiết để hoàn thành yêu cầu của người dùng.
* Trong kiến trúc .NET Core, Application Layer giúp đảm bảo rằng các tầng khác không phải lo lắng về cách dữ liệu được chuyển từ tầng này sang tầng khác. Tầng này thường bao gồm các lớp dịch vụ (service layer) và các pattern như Command hoặc Query để quản lý logic ứng dụng.
* **Tầng Logic nghiệp vụ (Business Logic Layer)**
* Đây là tầng cốt lõi của hệ thống, nơi xử lý các quy tắc nghiệp vụ, quy trình phức tạp và các phép tính quan trọng. Business Logic Layer chịu trách nhiệm thực hiện các quy tắc và logic nghiệp vụ của ứng dụng, đảm bảo rằng các hoạt động của ứng dụng tuân thủ đúng các yêu cầu kinh doanh đã đặt ra.
* Các thành phần chính trong tầng này bao gồm các dịch vụ nghiệp vụ, các đối tượng nghiệp vụ, và các bộ xử lý quy trình. Việc tách riêng logic nghiệp vụ khỏi các tầng khác giúp dễ dàng thay đổi, mở rộng và bảo trì ứng dụng mà không ảnh hưởng đến các tầng khác.
* **Tầng truy cập dữ liệu (Data Access Layer)**
* Tầng Data Access chịu trách nhiệm quản lý mọi hoạt động liên quan đến dữ liệu, bao gồm lưu trữ, truy xuất, cập nhật, và xóa dữ liệu từ cơ sở dữ liệu hoặc các nguồn dữ liệu khác. Tầng này đảm bảo rằng mọi thao tác với dữ liệu đều được thực hiện một cách an toàn và hiệu quả.
* Trong .NET Core, Data Access Layer thường được xây dựng bằng cách sử dụng Entity Framework Core (EF Core) hoặc các thư viện ORM (Object-Relational Mapping) khác như Dapper. Tầng này cũng chịu trách nhiệm kết nối với các dịch vụ dữ liệu khác, bao gồm các API, hệ thống tập tin, hoặc các nguồn dữ liệu bên ngoài.

## Ưu điểm & nhược điểm

* **Ưu điểm:**
* **Đa nền tảng**

.NET Core được thiết kế để có thể chạy trên nhiều hệ điều hành như **Windows, macOS, và Linux**. Điều này giúp các lập trình viên dễ dàng phát triển ứng dụng mà không bị giới hạn bởi hệ điều hành cụ thể. Hơn nữa, khả năng viết mã một lần và triển khai trên nhiều nền tảng giúp giảm thiểu thời gian và công sức trong việc quản lý môi trường phát triển đa dạng. Điều này đặc biệt hữu ích trong các môi trường đám mây, nơi mà ứng dụng cần tính linh hoạt cao.

* **Hiệu suất cao**

Một trong những ưu điểm lớn nhất của .NET Core là **hiệu suất** vượt trội. Theo các bài kiểm tra hiệu năng, .NET Core thường có tốc độ nhanh hơn so với .NET Framework trong nhiều trường hợp. Điều này được lý giải bởi các cải tiến về **runtime**, **bộ thu gom rác (garbage collector)** và sự tối ưu hóa cho các ứng dụng web và API. Nhờ đó, .NET Core được ưu ái cho các hệ thống yêu cầu xử lý khối lượng lớn, đặc biệt là các dịch vụ web và microservices.

* **Mã nguồn mở**

.NET Core được phát triển dưới hình thức mã nguồn mở, với sự hỗ trợ và đóng góp mạnh mẽ từ cộng đồng phát triển trên toàn thế giới. Dự án này được lưu trữ trên **GitHub**, cho phép các lập trình viên từ nhiều nơi tham gia vào việc cải tiến, sửa lỗi và đề xuất các tính năng mới. Sự minh bạch và khả năng tiếp cận mã nguồn mở giúp cho .NET Core phát triển nhanh chóng và nhận được sự hỗ trợ tốt từ cộng đồng.

* **Triển khai độc lập**

Khả năng triển khai độc lập của .NET Core cho phép các nhà phát triển đóng gói toàn bộ ứng dụng cùng với các thư viện cần thiết vào một tệp duy nhất mà không phụ thuộc vào phiên bản .NET nào đang được cài đặt trên máy chủ. Điều này giảm thiểu sự cố về môi trường và giúp triển khai dễ dàng hơn, đặc biệt là trong các hệ thống lớn hoặc hệ thống phân tán.

* **Hỗ trợ tốt cho container và cloud-native**

.NET Core tích hợp hoàn hảo với các công nghệ hiện đại như **Docker** và **Kubernetes**, giúp cho việc triển khai ứng dụng trong môi trường **container hóa** và **đám mây** trở nên dễ dàng và hiệu quả. Điều này làm cho .NET Core trở thành lựa chọn lý tưởng cho các kiến trúc **microservices** và hệ thống phân tán, giúp giảm thiểu chi phí và tăng cường tính linh hoạt khi mở rộng hệ thống.

* **Khả năng cập nhật nhanh chóng**

Do được phát triển theo quy trình mã nguồn mở và có sự tham gia của cộng đồng, .NET Core thường xuyên được cập nhật với các bản phát hành mới. Microsoft cung cấp các **bản cập nhật đều đặn** với các tính năng cải tiến và các bản sửa lỗi, đảm bảo rằng nền tảng này luôn duy trì trạng thái hiện đại và hỗ trợ các công nghệ mới.

* **Tích hợp với công nghệ hiện đại**

+ Containerization và Microservices: .NET Core hỗ trợ mạnh mẽ cho các **container** như **Docker**, giúp dễ dàng triển khai các ứng dụng theo mô hình **microservices**. Đây là xu hướng kiến trúc hiện đại giúp các hệ thống có khả năng mở rộng linh hoạt, dễ dàng quản lý và triển khai.

+ Tích hợp với Cloud-native: .NET Core tương thích tốt với các nền tảng đám mây như **Microsoft Azure**, **AWS**, và **Google Cloud Platform**. Điều này giúp các ứng dụng có thể được triển khai dễ dàng trong các môi trường đám mây, tận dụng được tính năng mở rộng và quản lý tài nguyên của các nền tảng đám mây.

+ Kết hợp với DevOps: .NET Core hỗ trợ tích hợp chặt chẽ với các công cụ **CI/CD** trong quá trình phát triển phần mềm, chẳng hạn như **Azure DevOps**, **GitHub Actions**, và **Jenkins**. Điều này làm tăng hiệu suất và tự động hóa quy trình triển khai ứng dụng.

+ Tối ưu hoá cho API và IoT: .NET Core được tối ưu hóa để phát triển các **RESTful API**, giúp giao tiếp giữa các dịch vụ khác nhau trở nên nhanh chóng và linh hoạt. Ngoài ra, nó còn hỗ trợ phát triển các ứng dụng **Internet of Things (IoT)**, một trong những xu hướng công nghệ hiện đại nhất hiện nay

* **Khả năng bao mật cao**

+ Tích hợp tiêu chuẩn bảo mật và công cụ bảo mật mạnh mẽ: .NET Core hỗ trợ các chuẩn bảo mật phổ biến như OAuth và OpenID Connect, đồng thời tích hợp sẵn các công cụ bảo mật như **ASP.NET Core Identity** để dễ dàng quản lý xác thực, phân quyền và bảo mật dữ liệu người dùng.

+ Bảo vễ ứng dụng khỏi tấn công: Với khả năng ngăn chặn các lỗ hổng bảo mật phổ biến như **SQL Injection**, **XSS**, và **CSRF**, .NET Core giúp tăng cường an ninh cho ứng dụng web, giảm thiểu nguy cơ bị tấn công từ bên ngoài.

* **Nhược điểm:**
* **Khả năng tương thích ngược hạn chế**

+ Một số **thư viện và tính năng cũ** từ .NET Framework không hoàn toàn tương thích với .NET Core. Điều này gây khó khăn cho những dự án lớn, lâu đời khi muốn chuyển đổi hoặc tích hợp mã nguồn cũ với .NET Core.

* **Thiếu tính năng cho ứng dụng desktop**

+ Một số **thư viện và tính năng cũ** từ .NET Framework không hoàn toàn tương thích với .NET Core. Điều này gây khó khăn cho những dự án lớn, lâu đời khi muốn chuyển đổi hoặc tích hợp mã nguồn cũ với .NET Core.

* **Độ ổn định cho một số ứng dụng lớn**

+ Mặc dù .NET Core có nhiều cải tiến, nhưng đối với những hệ thống lớn và phức tạp được xây dựng trên .NET Framework, việc chuyển đổi sang .NET Core đôi khi không phải là lựa chọn tốt. Sự thay đổi cấu trúc và các vấn đề tương thích có thể ảnh hưởng đến độ ổn định của ứng dụng.

* **Không hoàn toàn thay thế được .NET Framework**

+ .NET Core vẫn chưa hoàn toàn thay thế được .NET Framework cho một số ứng dụng **Windows đặc thù**, như ứng dụng sử dụng Windows Workflow Foundation (WF), WCF hay ASP.NET Web Forms. Các tính năng này chưa được hỗ trợ hoàn toàn trong .NET Core.

# Các dự án tiêu biểu trong .NET Core

* **ASP.NET Core Web App (Model-View-Controller - MVC):**

**Mô tả**: Dự án này dựa trên mô hình **MVC** (Model-View-Controller), một kiến trúc phổ biến cho việc phát triển các ứng dụng web. Nó phân chia ứng dụng thành ba phần chính:

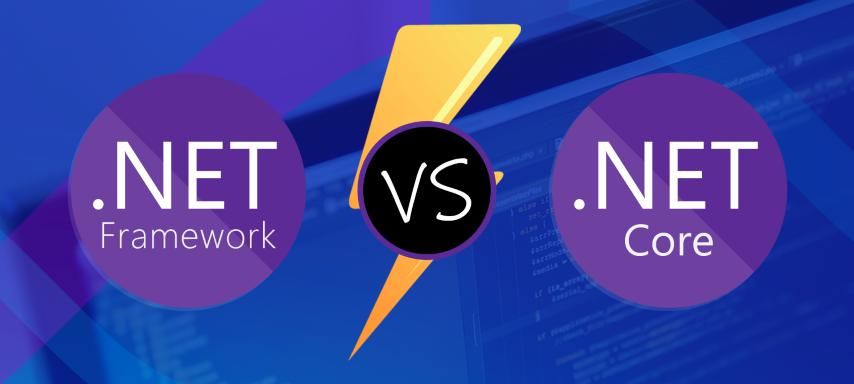
* + **Model**: Đại diện cho dữ liệu và logic nghiệp vụ.
  + **View**: Giao diện người dùng.
  + **Controller**: Điều khiển và xử lý các yêu cầu từ người dùng.
* **ASP.NET Core Web API:**

**Mô tả**: Đây là project dùng để xây dựng **RESTful API** hoặc các dịch vụ **HTTP** mà không có giao diện người dùng.

* **Trường hợp sử dụng**: Dùng để phát triển các ứng dụng back-end, nơi các thiết bị hoặc ứng dụng khác (mobile, web app) sẽ gọi API để lấy dữ liệu.
* **Điểm nổi bật**: Hỗ trợ **JSON** và **XML** để trao đổi dữ liệu, dễ dàng tích hợp với các ứng dụng phía client và các dịch vụ bên ngoài.
* **ASP.NET Core Blazor App:**
* **Mô tả**: Blazor là một framework mới cho phép bạn phát triển giao diện người dùng web bằng **C#** thay vì **JavaScript**.
  + **Blazor Server**: Chạy mã trên server và cập nhật giao diện người dùng qua **SignalR**.
  + **Blazor WebAssembly**: Chạy trực tiếp trên trình duyệt bằng công nghệ **WebAssembly**.
* **Trường hợp sử dụng**: Phù hợp cho các ứng dụng web với logic phức tạp hoặc những ai muốn tránh sử dụng JavaScript.
* **Điểm nổi bật**: Cho phép tái sử dụng mã C# từ server đến client.
* **ASP.NET Core Web App (Razor Pages)**
* **Mô tả**: Dự án Razor Pages đơn giản hóa mô hình phát triển web bằng cách kết hợp logic và giao diện trong cùng một trang. Không cần phải chia thành model, view và controller như MVC.
* **Trường hợp sử dụng**: Thích hợp cho các ứng dụng có giao diện đơn giản hoặc các trang web không yêu cầu quá nhiều logic phức tạp.
* **Điểm nổi bật**: Được xem là một cách tiếp cận dễ dàng và nhẹ nhàng hơn so với MVC, nhưng vẫn rất mạnh mẽ.

# So sánh và đánh giá

Công nghệ tương đồng nhất với .NET Core là .NET Framwork, vậy nên ta sẽ so sánh 2 công nghệ này với nhau dưới đây :



|  |  |  |
| --- | --- | --- |
|  | **.NET Core** | **.NET Framework** |
| **Năm phát hành** | 2016 | 2002 |
| **Mã nguồn mở** | Mã nguồn mở hoàn toàn trên GitHub | Một phần (ASP.NET MVC, ...) |
| **Ngôn ngữ được hỗ trợ** | C#, F#, VB.NET (giới hạn) | C#, VB.NET, F# |
| **Mẫu ứng dụng** | Mô hình ứng dụng của .Net Core bao gồm ASP.NET và Windows Universal Apps. | Mô hình ứng dụng của Net Framework bao gồm WinForms, ASP.NET và WPF. |
| **Cập nhật** | Không phụ thuộc vào hệ điều hành, thông qua NuGet hoặc tải xuống thủ công | Thông qua Windows Update |
| **Khả năng đa nền tảng** | Được thiết kế để phát triển các ứng dụng **đa nền tảng**. Điều này có nghĩa là các ứng dụng có thể chạy trên **Windows, Linux, và macOS**, giúp các nhà phát triển có nhiều lựa chọn triển khai hơn và tận dụng được môi trường đám mây hiện đại | Chỉ chạy trên **Windows**, phù hợp cho các ứng dụng doanh nghiệp lớn và truyền thống, nhưng giới hạn trong việc triển khai trên các hệ điều hành khác |
| **Hiệu suất** | Có hiệu suất cao hơn trong việc xử lý các ứng dụng hiện đại. Do được tối ưu hóa để chạy trên cloud-native, microservices, và containerization, nó mang lại tốc độ xử lý nhanh hơn và khả năng mở rộng tốt hơn, đặc biệt với các ứng dụng cần tương tác liên tục trên nền tảng đám mây | Dù vẫn mạnh mẽ nhưng thiếu các tính năng hiện đại hỗ trợ microservices hay tối ưu hóa cho container, dẫn đến hiệu suất thấp hơn khi so sánh với .NET Core trong các môi trường ứng dụng mới |
| **Khả năng triển khai** | Hỗ trợ triển khai các **self-contained executables** (các tệp thực thi chứa tất cả các phụ thuộc), giúp đơn giản hóa quá trình triển khai, đặc biệt trong môi trường cloud và container | Yêu cầu phải có phiên bản .NET Framework tương thích trên hệ thống máy chủ, làm phức tạp hóa quá trình triển khai, đặc biệt là trong các hệ thống có nhiều phụ thuộc khác nhau |
| **Công nghệ và tính năng hỗ trợ** | Hỗ trợ tốt hơn các **công nghệ hiện đại** như microservices, Docker, và Kubernetes, cũng như có CLI mạnh mẽ để hỗ trợ lập trình viên trong môi trường không cần GUI (Graphical User Interface). Đây là lựa chọn tốt cho các kiến trúc cloud-native và các ứng dụng yêu cầu khả năng mở rộng lớn | Tốt cho các hệ thống lớn, phức tạp đã tồn tại lâu đời. Tuy nhiên, nó không hỗ trợ tốt cho các công nghệ mới như microservices hay container, dẫn đến hạn chế khi phát triển các ứng dụng hiện đại |
| **Cộng đồng và hệ sinh thái** | Cộng đồng .NET Core đang phát triển nhanh chóng với sự hỗ trợ từ Microsoft và cộng đồng mã nguồn mở. Các bản cập nhật liên tục giúp .NET Core thích ứng với xu hướng công nghệ hiện đại | Có một cộng đồng lớn, nhưng do Microsoft đã chuyển trọng tâm sang .NET Core và .NET 5+, sự phát triển của .NET Framework chậm lại, với ít tính năng mới được bổ sung và chỉ có các bản cập nhật bảo mật |

=> .NET Core với khả năng đa nền tảng, hiệu suất cao, và tính linh hoạt trong triển khai đã trở thành lựa chọn tối ưu cho các ứng dụng hiện đại, đặc biệt là trong môi trường cloud-native và microservices. Trong khi đó, .NET Framework vẫn là lựa chọn vững chắc cho các ứng dụng Windows truyền thống, nhưng sự mở rộng và tích hợp của .NET Core đã giúp các nhà phát triển tiếp cận nhiều môi trường và hệ sinh thái hơn, hướng tới tương lai phát triển ứng dụng hiện đại.

# Tài liệu tham khảo

1. <https://learn.microsoft.com/en-us/dotnet/fundamentals/>

2. <https://github.com/dotnet/core>

3. <https://devblogs.microsoft.com/dotnet/>

4. <https://codegym.vn/blog/net-core-la-gi-tong-quan-ve-net-core/>

5. <https://stackoverflow.com/questions/38063837/whats-the-difference-between-net-core-net-framework-and-xamarin>

6. <https://stackoverflow.com/questions/44136118/net-core-vs-asp-net-core>

7. <https://tenten.vn/tin-tuc/net-core-la-gi/>

8. <https://dzone.com/articles/net-framework-vs-net-core-a-complete-quick-compari>

9. <https://chatgpt.com/>

# Kết luận

.NET Core đã chứng minh mình là một nền tảng mạnh mẽ và linh hoạt, phù hợp cho phát triển ứng dụng hiện đại với nhiều ưu điểm nổi bật. Khả năng đa nền tảng, hiệu suất cao và tính linh hoạt trong công việc triển khai là những yếu tố then chốt giúp .NET Core trở thành lựa chọn ưu tiên cho các doanh nghiệp và lập trình viên.

Nếu như .NET Framework dã đóng vai trò quan trọng trong công việc phát triển các ứng dụng doang nghiệp trên nền tảng Windows, thì .NET Core đã mở rộng cánh cửa ra thế giới cross-platform, giúp các ứng dụng có thể hoạt động trên nhiều hệ điều hành khác nhau, bao gồm Windows, macOS và Linux. Điều này cho phép các doanh nghiệp giảm chi phí triển khai và tăng tính hiệu quả khi xây dựng hệ thống ứng dụng phân tán.

Một điểm mạnh đáng chú ý là .NET Core liên tục nhận được các bản cập nhật từ Microsoft, tập trung vào cải thiện hiệu uất và hỗ trợ các xu hướng công nghệ mới như **containerization**, **microservices**, và **cloud-native development. Đây là nền tảng lí tưởng cho các dự án cần khả năng mở rộng và hiệu suất cao trong môi trường đa dạng.**

Với tất cả những tính năng ưu việt trên, .NET Core không chỉ đơn thuần là một lựa chọn công nghệ, mà còn trở thành công cụ quan trọng giúp các doanh nghiệp và nhà phát triển xây dựng các ứng dụng hiện đại, đáp ứng nhu cầu ngày càng cao trong lĩnh vực công nghệ thông tin ngày nay.