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](#_Toc183124615)

[Lời mở đầu 2](#_Toc183124616)

[1. Giới thiệu đề tài 4](#_Toc183124617)

[2. Lịch sử phát triển 6](#_Toc183124618)

[3. Giới thiệu về .NET Core 10](#_Toc183124619)

[3.1 Các thành phần chính 10](#_Toc183124620)

[3.2 Ưu điểm & nhược điểm 17](#_Toc183124621)

[4. Các dự án tiêu biểu trong .NET Core 19](#_Toc183124622)

[5. So sánh và đánh giá 21](#_Toc183124623)

[6. Demo so sánh 23](#_Toc183124624)

[7. Tài liệu tham khảo 28](#_Toc183124625)

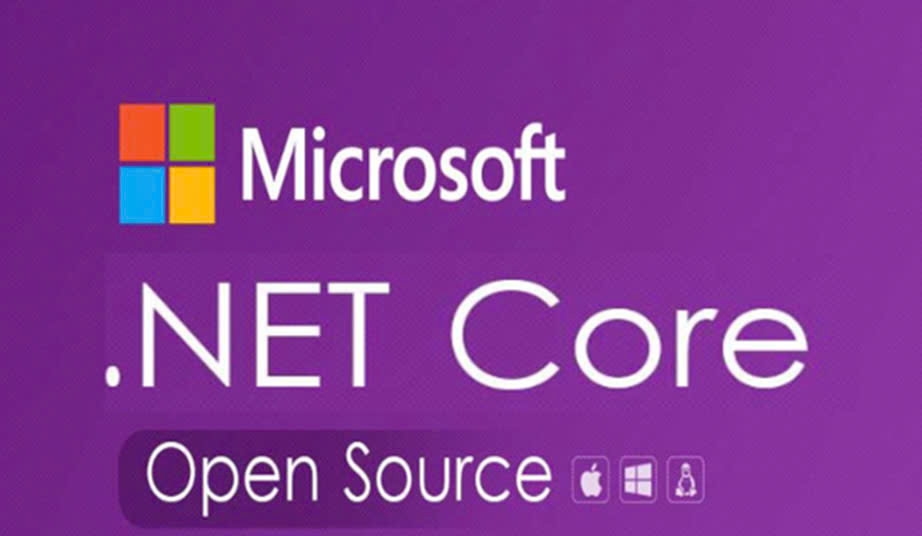
[8. Kết luận 29](#_Toc183124626)

# 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.

* 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.

* MIT License

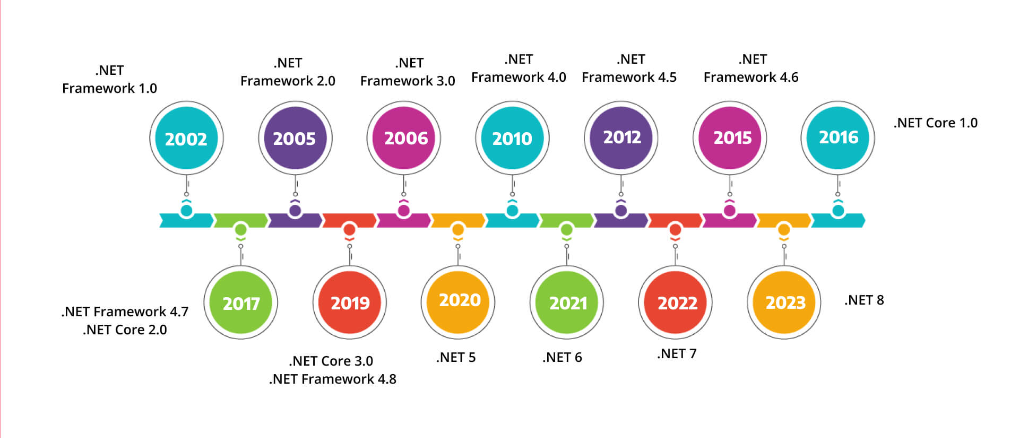
+ Nguồn : <https://github.com/dotnet/runtime/blob/main/LICENSE.TXT>

A screenshot of a computer

Description automatically generated

# 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 2.1**
* **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.
* **.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.
* **.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.
  + **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 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.
  + **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+.

# Giới thiệu về .NET Core

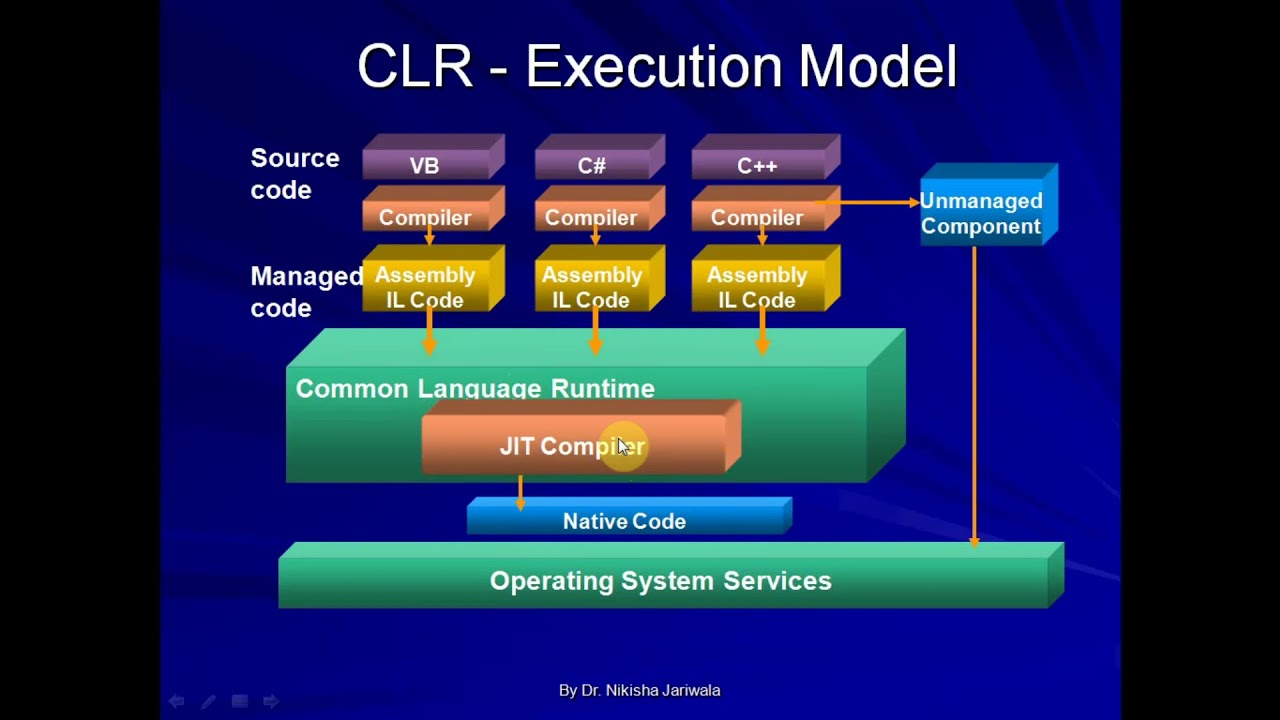
## Các thành phần chính

* **.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.



* **SignalR**

ASP.NET Core SignalR là một thư viện mã nguồn mở giúp đơn giản hóa việc thêm chức năng web thời gian thực vào ứng dụng. Chức năng web thời gian thực cho phép mã phía máy chủ đẩy nội dung đến máy khách ngay lập tức.

+ WebSockets: là giao thức cung cấp kết nối liên tục giúp gửi và nhận dữ liệu 2 chiều giữa clients và server. Các loại ứng dụng sử dụng websocket như chat trực tuyến, Bảng điều khiển thời gian thực, tích hợp thông báo hệ thống.

A diagram of a computer network

Description automatically generated

+ Server-Sent Events (SSE): SSE là một giao thức trong đó máy chủ có thể gửi sự kiện và dữ liệu mới đến máy khách một cách tự động. Nó là một phương tiện truyền tải đơn hướng, máy khách không thể gửi dữ liệu đến máy chủ. Phù hợp với các ứng dụng như: cập nhật sự kiện thể thao và Giải trí trực tuyến tới người dùng.

A diagram of a computer

Description automatically generated

+ Long Polling: Long Polling là một kỹ thuật trong đó máy khách gửi một yêu cầu đến máy chủ, và máy chủ giữ kết nối mở cho đến khi có dữ liệu mới hoặc hết thời gian chờ. Sau đó, máy khách và máy chủ sẽ thiết lập một kết nối mới. SignalR với phương tiện truyền tải Long Polling thường được sử dụng trong các tình huống mà việc duy trì kết nối liên tục không phải là ưu tiên, nhưng vẫn cần tính năng thời gian thực.

A diagram of a computer system

Description automatically generated

SignalR sử dụng WebSocket làm phương thức truyền tải cơ bản chính, đồng thời cung cấp các tính năng bổ sung như:

+ Kết nối lại tự động.

+ Chuyển sang phương tiện vận chuyển khác.

+ Một API để tạo các cuộc gọi thủ tục từ xa giữa máy chủ và máy khách (RPC).

+ Khả năng gửi tin nhắn đến tất cả các máy khách được kết nối cùng lúc hoặc đến các (nhóm) máy khách cụ thể.

SignalR sử dụng hub để giao tiếp giữa máy khách và máy chủ. Hub SignalR là một đường ống cấp cao cho phép máy chủ và máy khách được kết nối gọi các phương thức trên nhau.

|  |  |  |
| --- | --- | --- |
| **Giao thức** | **SignalR** | **WebSocket** |
| Khả năng tương thích (máy chủ) | Chủ yếu liên quan đến ASP.NET Core. Ngoài ra, còn sử dụng dịch vụ Azure SignalR. | Có thể được triển khai bằng nhiều công nghệ máy chủ khác nhau, bao gồm Node, Java, Python, v.v. |
| Khả năng tương thích (khách hàng) | Thư viện máy khách chính thức có sẵn cho JavaScript, C# và Java (Android), nhưng không có Swift (iOS). | Có thể triển khai bằng nhiều công nghệ giao diện khác nhau, bao gồm JavaScript, Java (Android) và Swift (iOS). |
| Dễ sử dụng | Dễ sử dụng hơn đối với các nhà phát triển vì nó tóm tắt một số sự phức tạp khi xử lý các cơ chế vận chuyển khác nhau. | Yêu cầu xử lý giao thức WebSocket ở cấp độ thấp hơn, điều này có thể gây khó khăn hơn cho một số nhà phát triển. |
| Tính linh hoạt | Ít linh hoạt và kiểm soát hơn | Linh hoạt và kiểm soát hơn |
| Hiệu suất | Đáng tin cậy, có thể mở rộng | Độ trễ thấp, thông lượng cao |

* **Ưu và nhược điểm của SignalR**

**\* Ưu Điểm**

+ Nhiều tùy chọn mặt phẳng nền để lựa chọn khi mở rộng quy mô: Redis, SQL Server hoặc Azure Service Bus (lưu ý rằng không cần phải cấu hình và quản lý mặt phẳng nền khi sử dụng phiên bản được quản lý hoàn toàn, Azure SignalR Service).

+ Một phần của ASP. NET Framework, giúp dễ dàng sử dụng SignalR kết hợp với các tính năng khác của ASP. NET như xác thực, ủy quyền và tiêm phụ thuộc.

+ SignalR sử dụng WebSockets bất cứ khi nào có thể, nhưng nó cũng hỗ trợ hai phương thức truyền tải dự phòng: [Sự kiện do máy chủ gửi](https://hubs.la/Q01w14sx0) và HTTP [long polling](https://hubs.la/Q01w14q00), cho các môi trường mà phương thức truyền tải WebSocket bị chặn/không được hỗ trợ.

**\* Nhược điểm**

+ SignalR cung cấp QoS nhắn tin khá yếu; việc sắp xếp và phân phối không được đảm bảo. Bạn sẽ phải phát triển cơ chế riêng của mình để đảm bảo nhắn tin mạnh mẽ (ví dụ: thêm thông tin sắp xếp vào chính tin nhắn).

+ SignalR cung cấp một số lượng hạn chế SDK cho khách hàng: C#, Java, Python và JavaScript. Không có SDK nào cho các nền tảng và ngôn ngữ như iOS, Android, Go, Ruby hoặc PHP.

+ SignalR được thiết kế để trở thành giải pháp vùng đơn, điều này có thể ảnh hưởng tiêu cực đến hiệu suất, độ tin cậy và tính khả dụng của giải pháp.

* **Các trường hợp sử dụng SignalR**

Vì SignalR là một triển khai WebSocket, bạn có thể sử dụng nó để thiết kế các loại tính năng thời gian thực giống như bạn xây dựng với WebSocket thô. Ví dụ bao gồm:

+ Bảng thông tin thời gian thực.

+ Trò chuyện trực tiếp.

+ Ứng dụng cộng tác.

+ Cập nhật vị trí theo thời gian thực.

+ Thông báo đẩy.

Demo nhỏ của SignalR , kịch bản khi 2 người dùng trong cùng 1 địa chỉ gửi lời thoại cho nhau , thì khi thành viên 2 của nhóm gửi đi lời thoại thì ngay sau đó thành viên 1 của nhóm nhận được lời nhắn từ thành viên 2 :

Trước khi gửi :

A screenshot of a computer

Description automatically generated

Sau khi gửi :

Screens screenshot of a computer

Description automatically generated

Screens screenshot of a chat

Description automatically generated

A black screen with white text

Description automatically generated

* Đăng ký SignalR trong Programs:

A screenshot of a computer program

Description automatically generated

* **gRPC**
* **gRPC là gì ?**

gRPC là một framework hiện đại, hiệu suất cao phát triển từ giao thức remote procedure call (RPC) truyền thống. Ở cấp độ ứng dụng, gRPC tối ưu hóa việc nhắn tin giữa các client và dịch vụ back-end. Được khởi nguồn từ Google, gRPC là mã nguồn mở và nằm trong hệ sinh thái của Cloud Native Computing Foundation (CNCF) về các giải pháp điện toán đám mây.

* **Các trường hợp sử dụng gRPC**

Hãy ưu tiên sử dụng gRPC trong các tình huống sau:

**+** Giao tiếp đồng bộ giữa các microservice ở backend khi cần phản hồi ngay lập tức để tiếp tục xử lý.

**+** Môi trường đa ngôn ngữ lập trình (polyglot) cần hỗ trợ nhiều nền tảng lập trình khác nhau.

**+** Giao tiếp với độ trễ thấp và thông lượng cao, nơi hiệu suất là yếu tố quan trọng.

**+** Giao tiếp điểm-điểm theo thời gian thực – gRPC có thể đẩy tin nhắn theo thời gian thực mà không cần polling và hỗ trợ rất tốt streaming hai chiều.

**+** Môi trường bị giới hạn băng thông mạng – các tin nhắn nhị phân của gRPC luôn nhỏ hơn so với tin nhắn JSON dựa trên văn bản tương đương.

+ gRPC chủ yếu được sử dụng với các dịch vụ backend.

Các trình duyệt hiện đại không thể cung cấp mức độ kiểm soát HTTP/2 cần thiết để hỗ trợ client gRPC ở phía front-end.Tuy nhiên, hiện nay đã có hỗ trợ gRPC-Web với .NET, cho phép giao tiếp gRPC từ các ứng dụng dựa trên trình duyệt được xây dựng bằng JavaScript hoặc công nghệ Blazor WebAssembly.

gRPC-Web giúp ứng dụng gRPC ASP.NET Core hỗ trợ các tính năng gRPC trong ứng dụng trình duyệt:

+ Client được tạo mã và có kiểu dữ liệu mạnh.

+ Tin nhắn Protobuf nhỏ gọn.

+ Server streaming.

* **Ưu điểm và nhược điểm gRPC**

**\* Ưu điểm gRPC**

**+ Hiệu suất cao:** gRPC nhanh hơn REST+JSON nhờ Protobuf (dữ liệu nhỏ gọn) và HTTP/2.**Khả năng tương thích đa ngôn ngữ**: Hỗ trợ nhiều ngôn ngữ lập trình, dễ dàng triển khai hệ thống microservices.

**+ Phát trực tuyến:** Hỗ trợ streaming phía client, server hoặc hai chiều, phù hợp xây dựng ứng dụng thời gian thực.

+ Tạo mã tự động: Tự động tạo mã client/server từ file .proto, tiết kiệm thời gian phát triển.

+ Khả năng tương tác: Hỗ trợ đa ngôn ngữ và nền tảng.

+ Bảo mật: Kết nối mã hóa TLS qua HTTP/2 đảm bảo API an toàn hơn.

+ Hiệu suất đa nền tảng: Là giải pháp RPC toàn diện, chạy trên nhiều hệ điều hành và ngôn ngữ.

**\* Nhược điểm gRPC**

**+Kiến thức**: Sử dụng Protobuf và HTTP/2 có thể phức tạp hơn đối với người mới.

**+ Không hỗ trợ tốt trên trình duyệt**: Do trình duyệt không hỗ trợ đầy đủ HTTP/2, việc sử dụng gRPC-Web là cần thiết.

**+ Khó debugging hơn REST**: Vì dữ liệu được nén và mã hóa, nên không thể dễ dàng kiểm tra bằng cách đọc như JSON hay XML.

* **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.

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.

## 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.

* **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.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.

* **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 thường xuyên 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.

+ 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.** 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 bảo 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.

* **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.

* **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.
* **IoT (Internet of Things) Applications**

**Mô tả:** .NET Core cung cấp khả năng phát triển các ứng dụng IoT chạy trên các thiết bị như Raspberry Pi, Arduino hay các thiết bị nhúng khác.

* Nhẹ và hiệu quả: .NET Core nhẹ và nhanh chóng, thích hợp cho các ứng dụng chạy trên các thiết bị có tài nguyên hạn chế.
* GPIO Access: Có thể truy cập trực tiếp vào các chân GPIO trên các thiết bị như Raspberry Pi.
* Cross-platform: Chạy trên Linux và Windows IoT, cho phép ứng dụng IoT linh hoạt trên nhiều nền tảng.
* **Microservices Architecture**

**Mô tả:** .NET Core là nền tảng lý tưởng để xây dựng kiến trúc microservices, nơi các dịch vụ nhỏ, độc lập giao tiếp với nhau qua các API.

* Tích hợp với Docker/Kubernetes: .NET Core rất phù hợp để container hóa và triển khai trên Docker và Kubernetes.
* Dễ quản lý và bảo trì: Các ứng dụng microservices dễ dàng mở rộng và triển khai độc lập từng phần.
* Khả năng mở rộng: Hỗ trợ scaling dễ dàng và hiệu quả, giúp các dịch vụ microservices có thể đáp ứng được các yêu cầu lớn.
* Gaming Applications

**Mô tả:** Mặc dù không phải là nền tảng phát triển game chính, nhưng .NET Core đã được sử dụng trong một số công cụ game, ví dụ như Unity (một trong những engine phát triển game phổ biến).

* Cross-platform: Có thể phát triển game chạy trên Windows, macOS, Linux, và thậm chí là thiết bị di động.
* Unity support: Hỗ trợ tốt cho Unity, cho phép các nhà phát triển sử dụng C# để lập trình logic game.
* Hiệu suất tốt: Các dự án game với yêu cầu cao về hiệu suất và tối ưu tài nguyên có thể tận dụng .NET Core để tối ưu hóa game.
* **Console Applications**

**Mô tả:** Console App là một loại ứng dụng đơn giản, chạy trên dòng lệnh (command line) và không có giao diện người dùng đồ họa (GUI). Ứng dụng Console được tạo bằng .NET Core chủ yếu dùng để thực hiện các tác vụ tự động hóa, các công cụ quản trị hệ thống, hoặc các thử nghiệm đơn giản.

* Đơn giản và dễ sử dụng:
* Hiệu suất cao:
* Tích hợp tốt với hệ thống:
* Dùng trong các tác vụ nền (Background Tasks)

# 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). | 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 |

=> .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.

# Demo so sánh

Kịch bản So sánh :

Tạo 1 api trả về 10000 dữ liệu gồm các dữ liệu là id , Email và Name , sau khi tạo xong api và test trên 2 local của .Net frame là 64147 và .Net core là 5000 .

* Môi trường thử nghiệm :

.Net core : Bản .Net 7.0

.Net framework: Bản .Net 4.8

Công cụ đo Performance giữa 2 bản : Post Man

* Đoạn code thực nghiệm : code 1 api tạo 10000 dữ liệu trả về đồng thời :

A screenshot of a computer program

Description automatically generated

Sau khi chạy chương trình đồng thời cả 2 bản .Net :

A screenshot of a computer

Description automatically generated

Tiến hành cho Api vào Post Man để test :

* Ở lần đo 1 :
* .Net Core

A screenshot of a computer

Description automatically generated

* .Net Frame

A screenshot of a computer

Description automatically generated

* Ở lần đầu tiên ta thấy tốc độ request dữ liệu trả về của Net Core là 40 ms còn Net Frame là 46 ms , để tăng độ chính xác cho phép đo so sánh thì nhóm em sẽ tự động lặp lại đo với 300 lần( có thể đo thêm rất nhiều lần ) trên Post Man như sau:

A black screen with a white line

Description automatically generated

Dựa vào số lần Test và do số lượng test lớn trên 300 lần Test , em xin phép lập 1 bảng cho kết quả 30 lần để tính trung bình thời gian request Api trả về data (ms) của .Net core và .Net Frame như sau:

|  |  |  |
| --- | --- | --- |
| Lần Test | .Net Core (ms) | .Net Frame (ms) |
| 1 | 40 | 46 |
| 2 | 43 | 71 |
| 3 | 36 | 58 |
| 4 | 34 | 53 |
| 5 | 43 | 45 |
| 6 | 31 | 44 |
| 7 | 33 | 44 |
| 8 | 37 | 69 |
| 9 | 38 | 48 |
| 10 | 50 | 49 |
| 11 | 36 | 45 |
| 12 | 39 | 68 |
| 13 | 98 | 69 |
| 14 | 37 | 51 |
| 15 | 41 | 48 |
| 16 | 30 | 48 |
| 17 | 38 | 47 |
| 18 | 68 | 60 |
| 19 | 40 | 48 |
| 20 | 36 | 70 |
| 21 | 39 | 48 |
| 22 | 102 | 70 |
| 23 | 38 | 65 |
| 24 | 52 | 69 |
| 25 | 35 | 53 |
| 26 | 49 | 53 |
| 27 | 30 | 49 |
| 28 | 43 | 50 |
| 29 | 59 | 71 |
| 30 | 35 | 55 |
| Avg(trung bình) | 44.33 ms | 55.47 ms |

Dựa vào bảng đánh giá trên thì thầy và các bạn thấy tốc độ trả về dữ liệu của .Net Core đang hơn nhanh hơn so với .Net Framework.

* **Nhận xét và đánh giá**
* **.NET Core:**
* Hiệu suất cao hơn: Xử lý API nhanh hơn, đặc biệt khi có nhiều yêu cầu đồng thời nhờ tối ưu hóa quản lý bộ nhớ và hỗ trợ tốt async/await.
* Ưu việt trong ứng dụng hiện đại: Thích hợp cho API yêu cầu cao về tốc độ và khả năng mở rộng.
* **.NET Framework:**
* Hiệu suất thấp hơn: Xử lý API chậm hơn trong các tình huống phức tạp, do thiết kế cũ không tối ưu cho hệ thống đa luồng hoặc async.
* Hạn chế trong ứng dụng hiện đại: Kém linh hoạt và không lý tưởng cho ứng dụng yêu cầu cao.

# 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://dev.to/ably/signalr-vs-websocket-which-is-best-for-your-realtime-app-3b4g?comments_sort=latest>

# 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.