

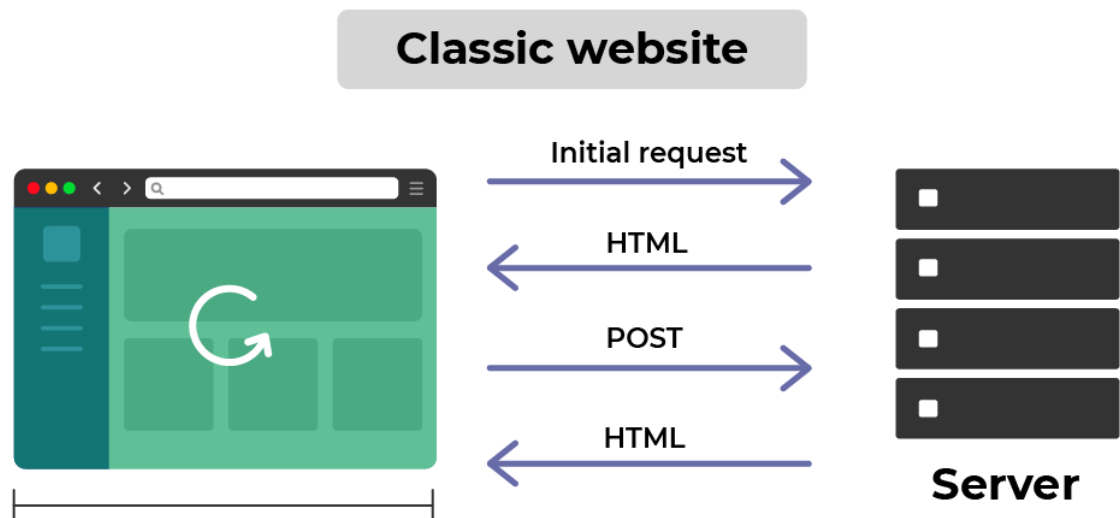
Xây dựng website tra cứu nguồn gốc nông sản khu vực chợ Đầu Mối

Họ tên: Võ Sơn Nam

Mssv: 1751061002

I. Mutili page applicaton, Single page application:

1) Mutili page applicaton



Ứng dụng nhiều trang, còn được gọi là MPA, là những ứng dụng web hoạt động theo cách truyền thống: mỗi khi người dùng yêu cầu (hoặc gửi) dữ liệu đến máy chủ, họ sẽ hiển thị một trang mới được gửi quay lại trình duyệt. Đây là cách tất cả các trang web từng hoạt động trong 20 năm đầu tiên của World Wide Web và vẫn là pháp được sử dụng rộng rãi nhất hiện nay do một số ưu điểm mà MPAs vẫn có thể mang lại:

- **Tốt hơn cho SEO:** Google Analytics cũng hoạt động tốt hơn với MPA, cho phép bạn nhanh chóng biết phần nào trong ứng dụng của bạn được người dùng phản hồi mạnh mẽ nhất - và một lần nữa, giúp trang web của bạn có thứ hạng cao trong các tìm kiếm;

- **Tương thích với các trình duyệt cũ hơn:** MPA là một cách cổ điển để xây dựng các trang web, do đó nó tương thích với hầu hết các trình duyệt cũ và các hệ thống cũ. Khả năng tương thích này rất quan trọng trong các tình huống mà người dùng có thể truy cập ứng dụng từ nhiều thiết bị hoặc nền tảng khác nhau;
- **Thời gian tải ban đầu nhanh hơn:** Thời gian tải đầu tiên của MPA thường nhanh hơn vì trình duyệt chỉ cần tải nội dung cho trang hiện tại thay vì toàn bộ nội dung cho toàn bộ ứng dụng.

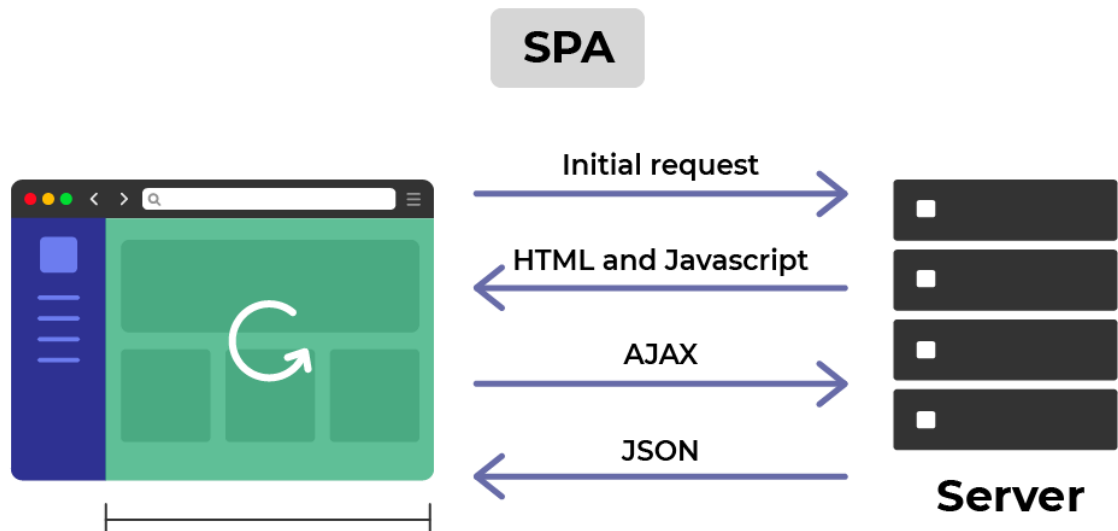
Tuy nhiên, các MPA cũng có một số nhược điểm đáng kể:

Hiệu suất: Vì MPA yêu cầu tải lại toàn bộ trang khi người dùng tương tác với ứng dụng nên chúng có hiệu suất kém, ảnh hưởng tiêu cực đến trải nghiệm của người dùng.

Phức tạp để phát triển và duy trì: Vì frontend và backend của chúng được kết hợp chặt chẽ hơn. Các ứng dụng như vậy cũng khó bảo trì vì mỗi trang phải được sửa đổi riêng. Trong số những vấn đề này đã được giảm thiểu trong suốt nhiều năm nhờ vào các tính năng và trình duyệt khác nhau của trình duyệt.

Tài nguyên máy chủ : MPA yêu cầu máy chủ xử lý nhiều yêu cầu hơn vì mỗi trang cần một yêu cầu riêng để tải. Điều này có thể dẫn đến tải máy chủ cao hơn và thời gian phản hồi chậm hơn, đặc biệt là đối với các ứng dụng có nhiều người dùng.

2) Single page application



Nói một cách ngắn gọn, SPA là một ứng dụng dựa trên web cố gắng cung cấp trải nghiệm người dùng giống như một ứng dụng máy tính để bàn. Nếu xem xét thực tế là tất cả các SPA vẫn được phục vụ thông qua web server và do đó được truy cập bởi các trình duyệt web, giống như bất kỳ trang web tiêu chuẩn nào khác, chúng ta có thể dễ dàng hiểu cách kết quả mong muốn đó chỉ có thể đạt được bằng cách thay đổi một số mẫu mặc định thường được sử dụng trong phát triển web, chẳng hạn như tải tài nguyên, quản lý DOM và điều hướng UI. Trong một SPA tốt, cả nội dung và tài nguyên – HTML, JavaScript, CSS, v.v. – đều được truy xuất trong một tải trang hoặc được tải nạp động khi cần. Điều này cũng có nghĩa là trang không cần tải lại hoặc refresh; nó chỉ thay đổi và điều chỉnh để đáp ứng với hành động của người dùng, thực hiện các yêu cầu server-side.

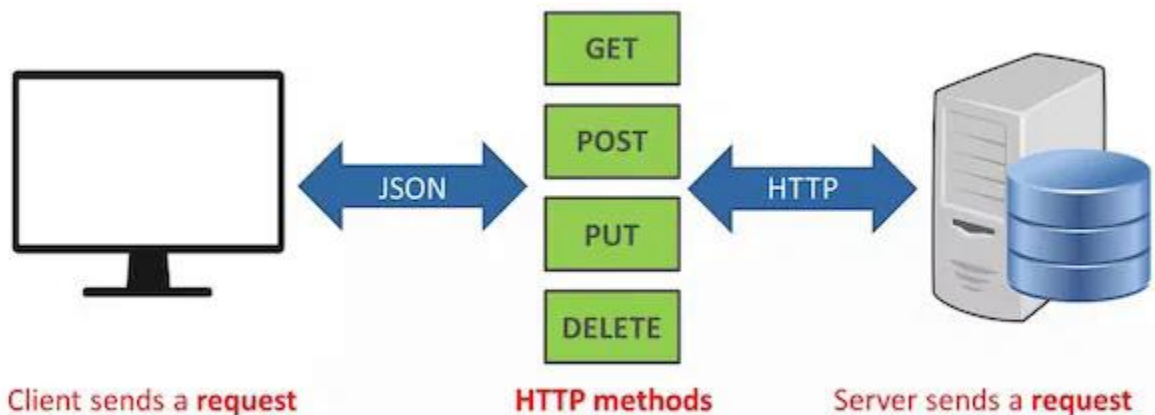
Đây là một số tính năng chính được cung cấp bởi SPA cạnh tranh hiện nay:

- **Không có chuyển đi khứ hồi phía máy chủ:** SPA có thể hiển thị lại bất kỳ phần nào của client UI mà không yêu cầu chuyển đi khứ hồi đầy đủ phía máy chủ để truy xuất trang HTML đầy đủ. Điều này chủ yếu đạt được bằng cách

triển khai nguyên tắc thiết kế separation of concerns (SOC), có nghĩa là nguồn dữ liệu, logic nghiệp vụ và lớp trình bày sẽ được tách biệt.

- **Định tuyến hiệu quả:** SPA có thể theo dõi trạng thái và vị trí hiện tại của người dùng trong toàn bộ trải nghiệm điều hướng bằng cách sử dụng các bộ định tuyến dựa trên JavaScript, có tổ chức.
- **Hiệu suất và tính linh hoạt:** SPA thường chuyển tất cả UI của mình sang máy khách nhờ vào SDK JavaScript được lựa chọn (Angular, jQuery, Bootstrap, v.v.). Điều này thường tốt cho hiệu suất mạng vì việc tăng khả năng hiển thị client-side và xử lý ngoại tuyến sẽ giảm tác động của UI qua mạng. Tuy nhiên, lợi ích thực sự mà phương pháp này mang lại là tính linh hoạt được cấp cho UI vì nhà phát triển sẽ có thể viết lại hoàn toàn UI của ứng dụng mà không hoặc ít tác động đến máy chủ, ngoại trừ một số tệp tài nguyên tĩnh.

II. RESTful API



RESTful API (Representational State Transfer API) là một kiểu kiến trúc cho các API (Application Programming Interface) được sử dụng để truyền tải và trao đổi dữ liệu giữa các ứng dụng web. RESTful API sử dụng giao thức HTTP để truyền tải dữ liệu giữa máy chủ và máy khách, và sử dụng các phương thức HTTP như GET, POST, PUT và DELETE để thực hiện các thao tác trên tài nguyên.

RESTful API sử dụng các URL dễ đọc và dễ hiểu, và sử dụng các định dạng dữ liệu như JSON hoặc XML để trao đổi thông tin giữa máy chủ và máy khách. RESTful API cũng có tính khả di động cao, cho phép các ứng dụng khác nhau có thể truy cập và sử dụng các tài nguyên một cách dễ dàng.

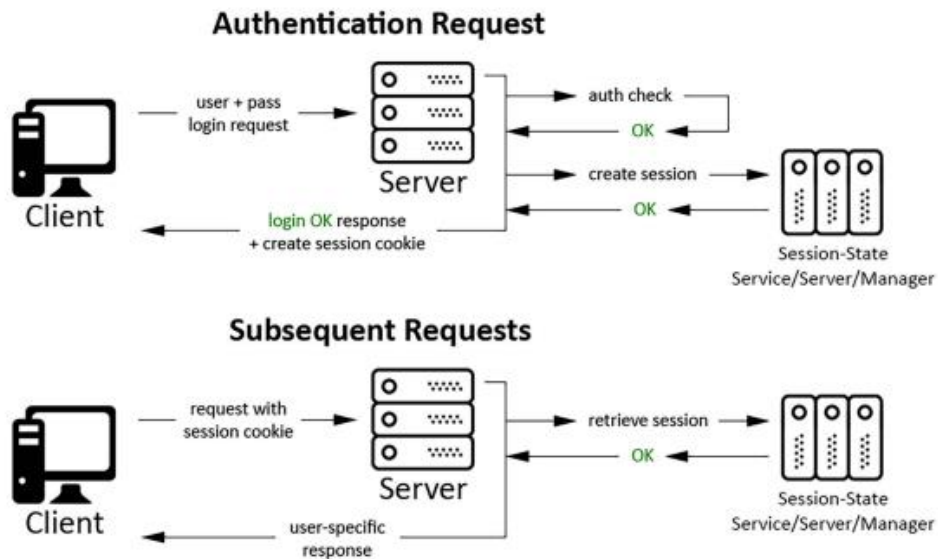
Restful API là một công nghệ quan trọng trong phát triển ứng dụng web hiện nay. Việc sử dụng Restful API giúp cho việc tương tác giữa các ứng dụng trở nên dễ dàng và thuận tiện hơn. Dưới đây là một số lý do vì sao Restful API là cần thiết trong phát triển ứng dụng web:

Đơn giản hóa việc phát triển: Restful API sử dụng các phương thức HTTP tiêu chuẩn, giúp cho việc phát triển ứng dụng trở nên đơn giản hơn. Nó cũng giúp cho các lập trình viên có thể tập trung vào phát triển các chức năng chính của ứng dụng thay vì phải quản lý việc tương tác giữa các thành phần khác nhau của ứng dụng.

- **Tính mở rộng:** Restful API cho phép các ứng dụng có thể kết nối với nhau và trao đổi dữ liệu một cách dễ dàng, từ đó giúp cho việc mở rộng ứng dụng trở nên thuận tiện hơn. Các ứng dụng cũng có thể thêm các tính năng mới bằng cách tương tác với các API của ứng dụng khác.
- **Tính độc lập của ứng dụng:** Restful API cho phép các ứng dụng hoạt động độc lập với nhau, từ đó giúp cho việc bảo trì và nâng cấp ứng dụng trở nên thuận tiện hơn.
- **Khả năng tương thích:** Restful API có khả năng tương thích với các ứng dụng khác nhau, từ đó giúp cho việc tích hợp các ứng dụng với nhau trở nên dễ dàng hơn. Các ứng dụng có thể sử dụng cùng một API để tương tác với các dữ liệu khác nhau mà không cần phải tạo ra các API riêng biệt.

III. Authentication user: Session-Based Authentication, Token-Based Authentication

1) Session-Based Authentication



Cho đến vài năm trước, phương pháp truyền thống và phổ biến nhất để thực hiện việc này là lưu trữ dữ liệu trên máy chủ sử dụng trình quản lý phiên dựa trên bộ nhớ, dựa trên đĩa hoặc bên ngoài. Mỗi phiên có thể được truy xuất bằng ID duy nhất mà khách hàng nhận được cùng với phản hồi xác thực, thường là bên trong một session cookie, được truyền đến máy chủ theo mỗi yêu cầu tiếp theo.

Đây vẫn là một kỹ thuật rất phổ biến được hầu hết các ứng dụng web sử dụng. Không có gì sai với áp dụng phương pháp này, miễn là chúng ta đồng ý với những nhược điểm được thừa nhận rộng rãi của nó, chẳng hạn như tiếp theo:

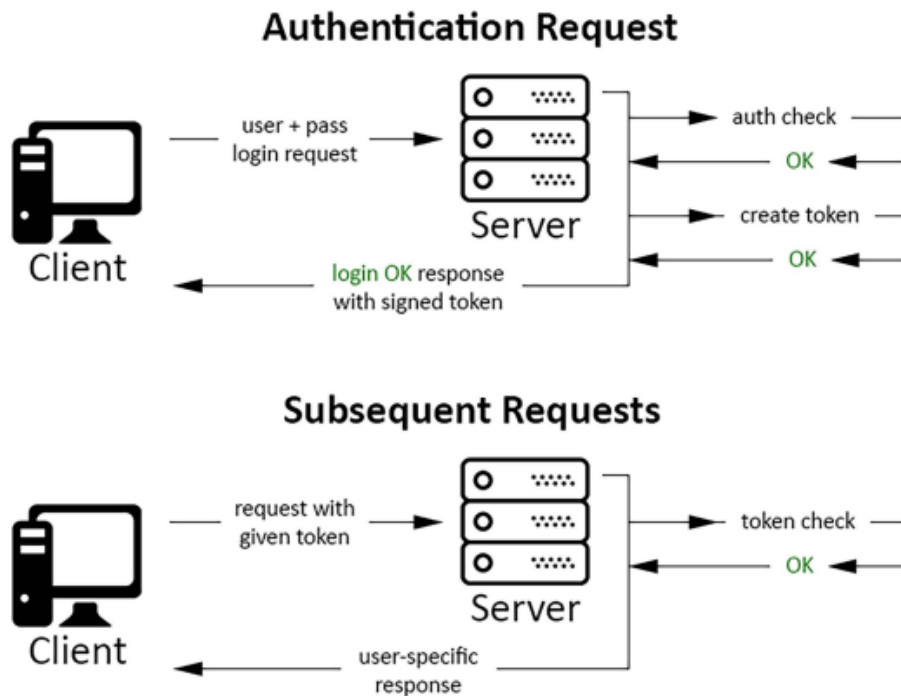
- **Vấn đề về bộ nhớ:** Bất cứ khi nào có nhiều người dùng được xác thực, máy chủ web sẽ tiêu thụ bộ nhớ ngày càng nhiều. Ngay cả khi sử dụng nhà cung cấp phiên dựa trên tệp hoặc bên ngoài, sẽ có tuy nhiên vẫn là một chi phí I/O, TCP hoặc socket overhead.

- **Các vấn đề về khả năng mở rộng:** Sao chép nhà cung cấp phiên trong kiến trúc có thể mở rộng (IIS web farm, load-balanced cluster và tương tự) có thể không phải là một nhiệm vụ dễ dàng và thường dẫn đến tắc nghẽn hoặc lãng phí tài nguyên.
- **Vấn đề về Cross-domain:** Cookie phiên hoạt động giống như cookie tiêu chuẩn, vì vậy chúng không thể được dễ dàng chia sẻ giữa các nguồn/tên miền khác nhau. Những loại vấn đề này thường có thể được giải quyết với một số cách giải quyết, tuy nhiên chúng thường dẫn đến những tình huống không an toàn để khiến mọi việc diễn ra suôn sẻ.
- **Các vấn đề bảo mật:** Có rất nhiều tài liệu chi tiết về các vấn đề liên quan đến bảo mật liên quan đến phiên và cookie phiên: ví dụ: Cross-Site Request Forgery (CSRF) và một số mối đe dọa khác sẽ không được đề cập ở đây vì mục đích đơn giản. Hầu hết trong số họ có thể được giảm thiểu bằng một số biện pháp đối phó, tuy nhiên chúng có thể khó xử lý đối với cấp dưới hoặc các nhà phát triển mới làm quen.

Vì những vấn đề này đã phát sinh trong nhiều năm nên chắc chắn rằng hầu hết các nhà phân tích và nhà phát triển đều đã nỗ lực rất nhiều để tìm ra các cách tiếp cận khác nhau cũng như giảm thiểu chúng.

Một cải tiến quan trọng liên quan đến việc giảm nhẹ đã đạt được vào năm 2016 với dự thảo cookie SameSite, trong đó đã đề xuất chính sách bảo mật HTTP sau đó được cải thiện bởi Cookies HTTP State Management Mechanism (tháng 4 năm 2019) và dự thảo Incrementally Better Cookies (tháng 5 năm 2019).

2) Token-Based Authentication







Token-based authentication ngày càng được các web SPA và ứng dụng di động áp dụng ngày càng nhiều trong vài năm gần đây, vì một số lý do chính đáng không thể phủ nhận được tóm tắt ngắn gọn ở đây.

Sự khác biệt quan trọng nhất giữa session-based authentication và Token-based authentication là cái sau không có trạng thái, nghĩa là sẽ không lưu trữ bất kỳ thông tin cụ thể nào của người dùng trên bộ nhớ máy chủ, cơ sở dữ liệu, nhà cung cấp phiên hoặc các loại chứa dữ liệu khác.

Khía cạnh duy nhất này giải quyết hầu hết các nhược điểm mà đã chỉ ra trước đó đối với xác thực dựa trên phiên. sẽ không có phiên hợp nên sẽ không có chi phí tăng thêm; chúng ta sẽ không cần một phiên nhà cung cấp, vì vậy việc mở rộng quy mô sẽ dễ dàng hơn nhiều. Ngoài ra, đối với các trình duyệt hỗ trợ LocalStorage, thậm chí sẽ không đang sử dụng cookie để không bị chặn bởi các chính sách hạn chế giữa nhiều nguồn gốc và hy vọng rằng sẽ nhận được xung quanh hầu hết các vấn đề bảo mật.

Về mặt tương tác giữa máy khách và máy chủ, các bước này dường như không khác mấy so với sơ đồ luồng session-based authentication; rõ ràng, sự khác biệt duy nhất là sẽ phát hành và kiểm tra mã thông báo thay vì tạo và truy xuất phiên. Tuy nhiên, sự việc thực sự đang diễn ra (hoặc không xảy ra) vào phía máy chủ. Chúng ta có thể thấy ngay rằng luồng Token-based authentication không dựa vào máy chủ, dịch vụ hoặc trình quản lý trạng thái phiên có trạng thái. Điều này sẽ dễ dàng chuyển thành một sự thúc đẩy đáng kể về mặt hiệu suất và khả năng mở rộng.

IV. Net core

ASP.Net 4.x Desktop		ASP.Net Core 1.x
Web Form MVC WebAPI	Windows Form WPF	MVC Core WebAPI Entity Framework Core
C#/VB Compiler		C#/VB Compiler(Roslyn)
.Net Framework		.Net Core
FCL/BCL (<i>System.Web, System.Data, System.Xml</i>) Net Framework Class Library		CoreFX (<i>System.Collections, System.IO, System.Xml</i>) Net Core Class Library, NuGetPackages
Common Language Runtime(CLR) <i>JIT Compiler/GC/CAS</i>		Common Language Runtime(CoreCLR) <i>RyuJIT Compiler(for 64-bit systems)/GC/SIMD</i>
Operating System 		Operating System   

C#

.NET 8 là một khuôn khổ lập trình miễn phí, đa nền tảng và mã nguồn mở cho phép phát triển nhiều ứng dụng điện toán đám mây, di động và web. Nền tảng phát triển mã nguồn mở mạnh mẽ này đã thúc đẩy ngành công nghiệp phần mềm trong nhiều năm. Với mỗi lần phát hành phiên bản mới, các chuyên gia kỹ trị được trao quyền để xây dựng các ứng dụng mạnh mẽ, hiệu quả và an toàn hơn.

Kể từ khi ra đời, .NET framework đã công bố bảy bản phát hành, mỗi bản có một bộ cải tiến mới. Gần đây, phiên bản kế nhiệm của .NET 7, .NET 8, đã được phát hành. .NET 8 không chỉ là một phiên bản framework mà còn hơn thế nữa. Nó định nghĩa lại cách các ứng dụng phần mềm được xây dựng và triển khai, cho phép các nhà phát triển đáp ứng các nhu cầu đang thay đổi của điện toán hiện đại. Nó cung cấp hiệu suất được cải thiện, khả năng chẩn đoán và quan sát được cải thiện, hỗ trợ đa nền tảng được mở rộng, công cụ và tích hợp tiên tiến, hỗ trợ dài hạn (LTS) và nhiều hơn nữa.

Sau đây là một số đặc điểm nổi bật cũng như là tính ưu trội của NET Core:

- Đa nền tảng: Chạy trên các hệ điều hành Windows, macOS và Linux.
- Nhất quán trên các kiến trúc: có thể chạy mã nguồn của bạn với cùng một hành vi trên nhiều kiến trúc hệ thống, bao gồm x64, x86 và ARM.
- Các công cụ dòng lệnh: Bao gồm các công cụ dòng lệnh để sử dụng, có thể được sử dụng để phát triển cục bộ và trong các tình huống tích hợp liên tục.
- Triển khai linh hoạt: có thể cài đặt song song (cài đặt toàn người dùng hoặc toàn hệ thống). Có thể được sử dụng với các container Docker
- Tương thích: tương thích với .NET Framework, Xamarin và Mono, thông qua .NET Standard.
- Nguồn mở: Nền tảng là nguồn mở, sử dụng giấy phép MIT và Apache 2. NET Core là một dự án .NET Foundation.
- Được hỗ trợ bởi Microsoft: được Microsoft hỗ trợ

Bên cạnh đó là những tính năng khá hay ho và tiện ích giúp đỡ người dùng, với NET Core bạn có thể tối ưu được rất nhiều hoạt động như:

- HTTP request được tối ưu nhẹ hơn.
- Hợp nhất xây dựng web UI và web APIs.
- Tích hợp những client-side frameworks hiện đại và có những luồng phát triển.
- Hệ thống cấu hình dựa trên môi trường đám mây thật sự.

- Dependency injection được xây dựng sẵn.
- Có thể host trên IIS hoặc self-host trong process của riêng bạn.
- Những công cụ mới để đơn giản hóa quá trình phát triển web tối ưu.
- Xây dựng và chạy đa nền tảng(Windows, Mac và Linux).
- Mã nguồn mở và tập trung vào cộng đồng.

V. SQL

SQL (Structured Query Language) là ngôn ngữ truy vấn có cấu trúc. Nó là một ngôn ngữ, là tập hợp các lệnh để tương tác với cơ sở dữ liệu. Dùng để lưu trữ, thao tác và truy xuất dữ liệu được lưu trữ trong một cơ sở dữ liệu quan hệ. Trong thực tế, SQL là ngôn ngữ chuẩn được sử dụng hầu hết cho hệ cơ sở dữ liệu quan hệ. Tất cả các hệ thống quản lý cơ sở dữ liệu quan hệ (RDMS) như MySQL, MS Access, Oracle, Postgres và SQL Server... đều sử dụng SQL làm ngôn ngữ cơ sở dữ liệu chuẩn.

Hầu như công ty nào lớn cũng cần xây dựng một hệ thống để lưu trữ cơ sở dữ liệu. Mọi thứ trong cơ sở dữ liệu này sẽ được diễn tả ra thành nhiều bảng, có mối quan hệ với nhau. Để truy vấn và lấy dữ liệu từ các bảng này nhằm tổng hợp thành thông tin nào đó, người ta dùng đến SQL thông qua các câu query.

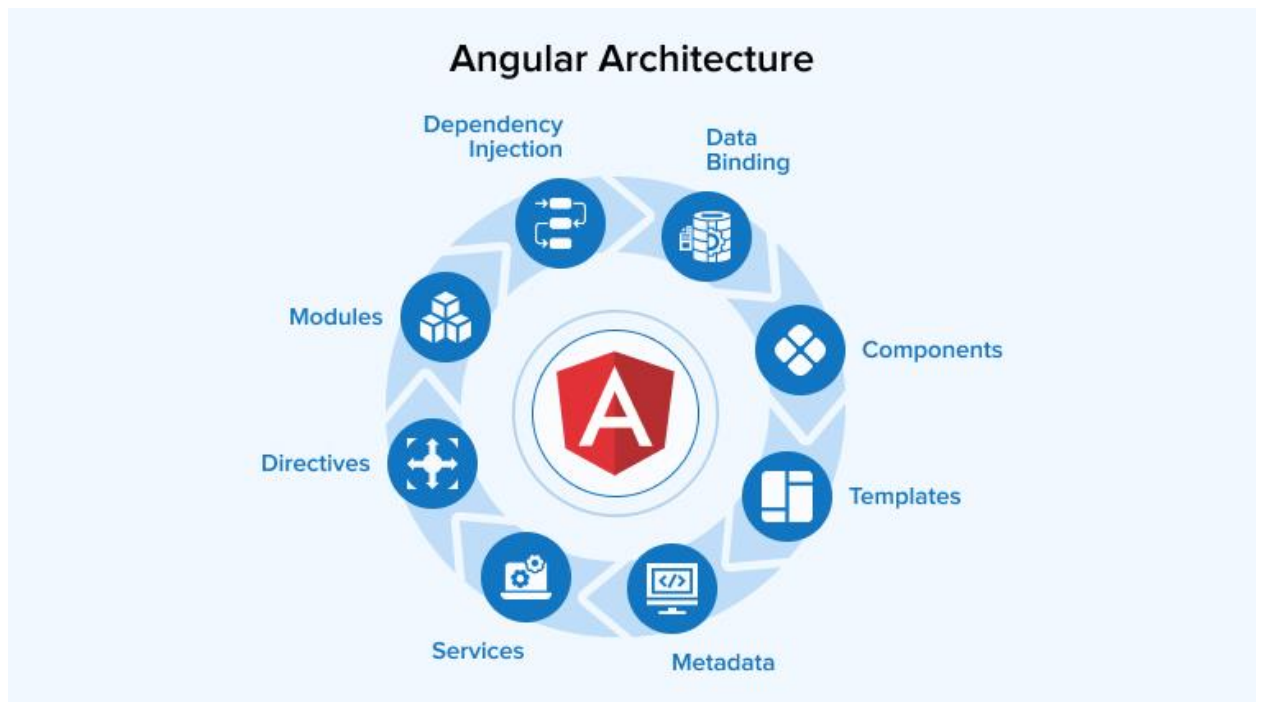
SQL được sử dụng phổ biến vì nó có các ưu điểm sau:

- Cho phép truy cập dữ liệu trong các hệ thống quản lý cơ sở dữ liệu quan hệ.
- Cho phép mô tả dữ liệu.
- Cho phép xác định dữ liệu trong cơ sở dữ liệu và thao tác dữ liệu đó.
- Cho phép nhúng trong các ngôn ngữ khác sử dụng mô-đun SQL, thư viện và trình biên dịch trước.
- Cho phép tạo và thả các cơ sở dữ liệu và bảng.
- Cho phép tạo chế độ view, thủ tục lưu trữ, chức năng trong cơ sở dữ liệu.
- Cho phép thiết lập quyền trên các bảng, thủ tục và view.

SQL sẽ giúp quản lý hiệu quả và truy vấn thông tin nhanh hơn, giúp bảo trì, bảo mật thông tin dễ dàng hơn.

Ví dụ: trước đây, trường đại học thường lưu trữ thông tin sinh viên bằng hồ sơ giấy. Sau đó, cất giữ hồ sơ trong kho. Khi cần tìm kiếm hoặc thêm/xóa/sửa thông tin nào đó, họ phải mất rất nhiều thời gian để lục tìm lại hồ sơ. Trong khi, nếu lưu trữ thông tin vào một hệ thống cơ sở dữ liệu, họ chỉ cần gõ một câu lệnh SQL ngắn là đã có thể trích xuất được thông tin cần. Việc thêm/xóa/sửa cũng được thực hiện một cách dễ dàng, nhanh chóng.

VI. Angular



Angular là một mã nguồn mở viết bằng TypeScript và được sử dụng để thiết kế giao diện web (front – end). Angular được xây dựng, phát triển từ những năm 2009 và đang duy trì cho đến nay bởi Google. Đây được xem là framework front end mạnh mẽ và chuyên dụng dành cho các lập trình viên sử dụng HTML cao cấp.

Angular được ứng dụng rộng rãi để xây dựng các project Single Page Application (ứng dụng trang đơn). Hiện tại, version stable của Angular là Angular 9 (released on February 7, 2020) với TypeScript 3.6 và 3.7.

Hiện nay, Angular được các công ty lớn lựa chọn sử dụng như: Upwork, Forbes, General Motors,... Đây sẽ là cơ hội việc làm rất lớn nếu bạn sử dụng thành thạo Angular. Tuy nhiên, trước đó bạn cần nắm vững các kiến thức nền tảng về JavaScript, CSS và HTML, cách làm việc với kiến trúc Model-View-Controller (MVC).

Angular mang đến nhiều ưu điểm nổi bật cho nhiều lập trình viên, cụ thể như:

- Angular được các chuyên gia đánh giá cao, mã nguồn này giúp các Single Page Application làm việc dễ dàng, nhanh chóng.
- Nhờ khả năng Binding data lên trên các nền tảng HTML nên code front-end thường rất thân thiện với người dùng.
- Bạn có thể thuận tiện Unit Test.
- Component có thể tái sử dụng dễ dàng hơn.
- Angular có khả năng hỗ trợ cho các lập trình viên có thể viết code được ít hơn cùng với nhiều chức năng hơn. Từ đó giúp tiết kiệm thời gian lập trình và tăng hiệu suất công việc.
- AngularJS tương thích với nhiều nền tảng khác nhau. Bạn có thể dùng được trên nhiều loại trình duyệt khác nhau cả trong máy tính và thiết bị điện thoại di động.

Ngoài những ưu điểm nổi bật đã nêu ở trên, Angular còn tồn tại một vài nhược điểm cần được khắc phục, cụ thể như:

- Tính bảo mật: Bản chất của Angular là một framework front-end. Thông thường, tính bảo mật của front-end thường không cao bằng back-end. Chính vì thế, bạn cần xây dựng một hệ thống kiểm tra dữ liệu sao cho việc trả về được tốt nhất khi sử dụng API.
- Khả năng an toàn: Website có thể trở nên không an toàn nếu bạn sử dụng một số trình duyệt sở hữu tính năng Disable JavaScript.

VII. Tài liệu tham khảo

ASP.NET Core 8 and Angular Sixth Edition Full-stack web development with ASP.NET Core 8 and Angular.

<https://www.tatvasoft.com/outsourcing/2023/08/angular-architecture.html>

<https://www.codeproject.com/Articles/1158377/NET-Core-Startup>

<https://www.tma.vn/Hoi-dap/cam-nang-nghe-nghiep/REST-API-la-gi-Cach-thiet-ke-REST-API-co-the-ban-chua-biet/60206>