# **LÝ THUYẾT**

1. **Tìm hiểu vấn đề an toàn mật khẩu.**

Ngày nay, với sự phát triển không ngừng của kĩ thuật máy tính, nguy cơ bị tấn công vào hệ thống thông tin ngày càng gia tăng, vấn đề an toàn mật khẩu ngày càng trở nên phức tạp, giờ đây mật khẩu không chỉ đơn thuần là một chuỗi ký tự bí mật của riêng người dùng, mà nó luôn ở trong nguy cơ bị bẻ khóa cao độ, do đó đòi hỏi người dùng cần có những kiến thức cần thiết về việc bảo mật mật khẩu.

1. **Mật khẩu là gì?**

Mật khẩu (hay còn gọi là mã xác nhận) là một từ hoặc một chuỗi ký tự bí mật được sử dụng để xác thực, chứng minh hoặc nhận dạng người sử dụng truy cập vào một website, phần mềm hay một nguồn tài nguyên cần phải chứng thực.

1. **Tại sao mật khẩu lại cần thiết?**

Mật khẩu giúp chúng ta ngăn chặn việc xâm nhập trái phép vào hệ thống, bảo vệ thông tin và giúp chúng ta xác nhận duy nhất cá nhân đăng nhập hệ thống cũng như ghi lại dấu vết những hành động của họ trên dữ liệu.

Bất cứ hệ thống nào một vài người dùng nhất định sẽ có những đặc quyền mà những người khác không có. Bằng cách nhận dạng chính bạn trên chính máy tính của bạn hoặc các website, bạn được quyền tiếp cận môi trường làm việc của riêng bạn và các dữ liệu các nhân của bạn, những tài liệu này là các dữ liệu nhạy cảm và không muốn công khai.

1. **Những mối nguy hiểm đến từ mật khẩu.**

Trong khi phần lớn các tổ chức và 99% người dùng tại mỗi quốc gia vẫn phụ thuộc vào mật khẩu như là một hình thức nhận dạng cơ bản đối với các dữ liệu nhạy cảm và riêng tư, thì các hệ thống mạng có cơ chế bảo mật thấp vô hình dung tạo ra các lỗ hổng cho kẻ tấn công tiếp cận tài nguyên của công ty và tài sản người dùng.

Mặc dù mật khẩu là phương tiện cần thiết và thân thiện với người dùng nhất để nhận dạng người dùng khi tiếp cận mang hoặc cơ sở dữ liệu của họ, nhưng thật sự người dùng rất lơ là với những yêu cầu là họ cần thay đổi mật khẩu thường xuyên, cần tạo ra mật khẩu có tính an toàn cao và làm theo những chỉ dẫn để tạo và giữ mật khẩu một cách an toàn hơn. Kết quả là một số lượng lớn các mật khẩu có thể dò tìm ra được, những mật khẩu giống nhau trên nhiều hệ thống khác nhau và người dùng ghi chú đâu đó để truy cập một cách dễ dàng như trên bàn làm việc hay sổ ghi chép,…để rồi bị những kẻ tấn công phát hiện và lợi dụng thực hiện hành vi xấu.

Những nguy hiểm khi mật khẩu bị bẻ khóa.

* Identity theft(Trộm thông tin nhận dạng): xảy ra khi dữ liệu tài khoản của bạn bị một người nào đó sử dụng. Điều này đưa đến những tổn hại về tài chính cũng như tổn hại cá nhân (ví dụ dùng tài khoản để đi giả mạo nhờ chuyển tiền, phản động, tuyên truyển sai lệch,…).
* Sensitive data exposure(Lộ dữ liệu nhạy cảm): nội dung của thư điện tử, tài liệu dự án, ảnh cá nhân bị phơi bày trước các hacker, hay các cá nhân nhắm đến bạn với mục đính xấu.
* Company data exposure(Lộ dữ liệu công ty): các hoạt động gián điệp lấy các thông tin nhạy cảm, nội bộ thông qua dữ liệu tài khoản được duy trì và giữ gìn thiếu cẩn thận đến sự ảnh hưởng vô cùng to lớn đến công ty bạn đang làm việc và chính cả bản thân bạn.
* Sử dụng cho các hoạt động tội phạm: tài khoản của bạn sẽ bị người khác lợi dụng cho mục đính tội phạm nếu bị bẻ khoá. Đừng quên rằng dấu vết đều được lưu lại trên mọi hệ thống nên việc tìm ra người gây là chuyện rất đơn giản và người chịu trách nhiệm là chủ tài khoản dù đó không phải bạn làm.

1. **Tìm hiểu phương pháp đánh giá mật khẩu.**
2. **Thế nào là mật khẩu yếu?**

Một mật khẩu yếu là một mật khẩu ngắn, phổ biến, một mặt định của hệ thống cung cấp hoặc một thứ gì đó có thể bị đoán ra nhanh chóng bằng cách thực thi tấn công vét cạn sử dụng một tập con của tất cả các mật khẩu khả dĩ như các từ trong từ điển, tên riêng, những từ dựa trên tên người dùng hoặc những biến thể thông thường của các từ đó. Mật khẩu có thể dễ dàng đoán được dựa trên những hiểu biết về người đó như ngày tháng năm sinh và tên thú nuôi cũng bị xem là yếu.

Các ví dụ về mật khẩu yếu:

* Dễ đoán, dễ dò tìm: 1234, abc123, abc, qwerty,..
* Có trong từ điển, tên riêng, biệt danh: Admin, password, An, Minh,…
* Ngày sinh: 01011900, 01/01/1900,…

Theo thống kê sơ bộ 3,8% số lượng mật khẩu là những từ đơn tìm thấy trong từ điển và 12% khác là một từ cộng thêm một con số ở cuối của tên ( hơn 60% số đó là số 1).Nhiều người dùng không đổi mật khẩu mặc định đi kèm với nhiều hệ thống bảo mật máy tính. Danh sách cách mật khẩu mặc định được cung cấp một cách công khai và rất đầy đủ trên internet.

Một mật khẩu có thể trở nên dễ dàng đoán được nếu người dùng chọn mật khẩu có chứa thông tin cá nhân công khai mà nhiều người biết như mã số sinh viên, tên người yêu, sinh nhật, số điện thoại hoặc biển số xe. Dữ liệu cá nhân về một người nào đó hiện phổ biến ở nhiều nguồn, nhiều khi còn được đưa lên rao bán trực tuyến trên internet hoặc có thể lấy được bởi người khác bằng cách sử dụng các kỹ thuật lừa bịp như đưa ra một bản lấy ý kiến hoặc một bản kiểm tra việc quản lý an ninh.

Nguy cơ cao nhất của việc sử dụng mật khẩu ngắn hoặc dễ đoán đó là tiếp cận tấn công từ những bạn bè của người dùng. Trong khi tên không phổ biến lắm của một con vật nuôi hoặc một nhân vật ưa thích trong trò chơi điện tử rất khó đoán đối với một người hoàn toàn xa lạ và khó tìm thấy trong từ điển, thì một người bạn khi có điều gì bất bình rõ ràng sẽ có ít lựa chọn để đoán hơn hẳn và cũng chẳng cần đến sự giúp đỡ của máy tính để dò ra mật khẩu.

Một ví dụ của mật khẩu nghèo nàn chống lại những kẻ tấn công “biết mặt” này có thể là “22HUTECH2018”, lấy thông tin người này tốt nghiệp trường Đại Học Công Nghệ TP.HCM có tên viết tắt là HUTECH vào năm 2018. Trong khi với độ dài đến 12 ký tự và khả năng chống lại vét cạn rất tốt, nếu biết rõ nạn nhân thì việc lấy thông tin trên hoàn toàn khả thi. Do đó trong khi có thể khiến cho một máy tính mạnh chạy mất vài tháng hoặc cũng có thể là vài năm để đoán được ra mật khẩu này.

Một mật khẩu thường dễ bị tổn thương nếu nó bị tìm thấy trong danh sách từ điển. Từ điển ở dạng máy đọc được có ở rất nhiều ngôn ngữ khác nhau và tồn tại những danh sách các mật khẩu thường được chọn. Trong các thử nghiệm đối với hệ thống đang hoạt động, tấn công bằng từ điển dễ thành công tới mức hiện thực kiểu tấn công này hiện nay phổ biển với nhiều hệ thống.

Một mật khẩu quá ngắn có lẽ được chọn để dễ dàng cho việc đánh bàn phím, dễ nhớ và dễ bị bẻ khoá nếu kẻ tấn công có thể lấy được bảng mật mã của mật khẩu. Các máy tính hiện nay đủ nhanh để thử tất cả các mật khẩu toàn chữ cái ngắn hơn 7 ký tự. Những nhân viên, lập trình viên và người quản trị hệ thống khi nghỉ việc thường biết khá rõ những mật khẩu hệ thống đã sử dụng vì nó hiếm khi được đổi. Các mật khẩu dễ đoán như vật có thể dẫn đến tổn hại nặng nề đến kinh tế nếu người tấn công với mục đích chuộc lợi hoặc trả thù.

### Thế nào là mật khẩu mạnh.

Một mật khẩu mạnh là một mật khẩu đủ dài, mang tính ngẫu nhiên hoặc nếu không chỉ có người chọn nó mới nghĩ ra được sao cho việc đoán được ra nó sẽ phải cần nhiều thời gian hơn là thời gian mà một kẻ tấn công bẻ khoá mật khẩu sẵn sàng bỏ ra để đoán ra nó. Thời gian để được cho là quá dài sẽ thay đổi tuỳ thuộc vào kẻ tấn công, tài nguyên của kẻ tấn công, sự dễ dàng tiếp cận với những mật khẩu có thể thử và giá trị của mật khẩu đó đổi với kẻ tấn công. Một mật khẩu của sinh viên chẳng đáng để máy tính bỏ ra vài giây để đoán, trong khi mật khẩu của quản lý việc truy xuất đến hệ thống chuyển tiền điện tử của một ngân hàng lớn có thể đáng để bỏ ra nhiều tuần hoặc thậm chí nhiều tháng để đoán.

Các ví dụ về mật khẩu mạnh:

* p3WahT35S
* A@1bcE35

Một cách kỹ thuật thì những ví dụ trên đều có tính hỗn loạn thông tin (về bit) là lớn hơn 3 trong khi các ví dụ yếu có độ hỗn loạn thông tin dưới 3. Nhưng là một vấn đề kỹ thuật, độ mạnh của mật khẩu có thể thoả mãn một muc đích sức mạnh của nó nếu thời gian cần thiết để phá vỡ mật khẩu vượt quá thời gian có thể bỏ ra để phá vỡ nó và nếu thông tin được bảo vệ trước khi những nỗ lực bẻ khoá hoàn thành.

Mật khẩu càng dài và việc lựa chọn ký hiệu càng rộng thì nỗ lực để bẻ khoá một mật khẩu càng phải mạnh mẽ mới có thể đánh bại mật khẩu đó (giả thiết bảng băm mật khẩu và các phương pháp bảo vệ phù hợp nơi lưu trữ của nó). Hơn nữa, không sử dụng từ đơn sẽ khiến cho tấn công vét cạn vô cùng kém hiệu quả hơn.

Chú ý rằng một số hệ thống không cho phép dùng ký hiệu hoặc những ký tự đặc biệt trong mật khẩu và hơn nữa chúng có thể khó tìm trong những kiểu bàn phím khác nhau. Trong trường hợp đó chỉ cần thêm một số chữ và số cũng có thể tăng được độ bảo mật tương đương: vừa chữ vừa số với chỉ một kiểu thường hoặc hoa thì không gian tìm kiếm là 36 nhưng nếu viết cả hoa và thường, chữ số thì ta mở rộng thêm không gian ra là 62.

Ngoài ra, những ví dụ ở trên được coi là mật khẩu mạnh thì bây giờ nó đã trở thành mật khẩu yếu vì nó đã vi phạm yếu tố công khai nên không còn là lựa chọn tối ưu. Tương tự những mật khẩu được xem là an toàn trên các diễn đàn hay được đề xuất là những ứng viên không còn yếu tố an toàn, đôi khi còn được liệt kê vào từ điển.

Một dạng mật khẩu mạnh khác là một từ được ngẫu nhiên hoàn toàn hoặc một phần với chữ in thường khác nhau và một hoặc nhiều số hoặc ký hiệu được dùng thêm vào. Mật khẩu kiểu đóm trong khi hầu như toàn chữ và người dùng dễ nhớ, rất dài và cần phải có bộ sinh mật khẩu vét cạn để kiểm thử tất cả các ký tự ở tất cả các kiểu chữ cũng như tất cả các số và ký hiệu bàn phím ở mỗi ký số, vì ký hiệu và con số có thể nẳm ở bất cứ đâu trong một mật khẩu. Như đã nói ở đoạn dưới điều này sẽ đánh bại tấn công vét cạn với tài nguyên thực tế.

Mật khẩu có thể tìm thấy bằng cách sử dụng một bộ máy sinh mật khẩu vét cạn như thế. Trong trường hợp đơn giản nhất, chhúng là những chương trình nhỏ chỉ đơn gian là thử tất cả các tổ hợp có thể. Một bộ xử lý 3GHz có thể tạo sấp xỉ 3 triệu mật khẩu trong một giây. Một mật khẩu có độ dài là 10 như “123456Ab@” có 95 ký tự trong bảng mã mật khẩu nên có 95^10 khả năng và sẽ phải tốn 632,860 năm để tìm ra giả sử mật khẩu được sinh ra ngẫu nhiên. Một một khẩu chứa 15 chữ cái in hoa ngẫu nhiên sẽ chỉ an toàn tương đương (với điểu kiện hệ thống đang bàn tới có phân biệt hoa thường và cho phép dùng ký hiệu) và có thể dễ nhớ hơn đối với một số người.

Nếu chỉ sử dụng 26 ký tự chữ thường và mật khẩu dài 7 ký tự thì số tổ hợp khá nhỏ: 26^7 = 8.03 tỷ tố hợp. Điều này có vẻ lớn đối với con người nhưng đối với một máy tính có thể sinh ra 3 triệu mật khẩu trong một giây, nó chỉ mất có 45 phút để tìm ra mật khẩu.

## **III. Tìm hiểu thuật toán đánh giá mật khẩu**

### 1. KeePass:

**a. Xác định tập ký tự:**

KeePass chia các ký tự thành 7 tập:

* Tập ký tự thường: “**a**” -> ”**z**” (ascii 97 -> 122)
* Tập ký tự hoa: “**A**” -> “**Z**” (ascii 65 -> 90)
* Tập ký tự số: “**0**” - > “**9**” (ascii 48 -> 57)
* Tập ký tự đặc biệt đơn giản: ““ - > “/” (ascii 32 -> 47)
* Tập ký tự đặc biệt mở rộng: “**:**” -> “**@**” (ascii 58 -> 64) và “**[**“ ->”**’**” (ascii 91 -> 96) và “**{**“ -> “**~**” (ascii 123 -> 126)
* Tập ký tự lớn hơn còn lại: (ascii > 126)
* Tập ký tự nhỏ hơn còn lại: (ascii<32)

**b. Tính toán không gian ký tự cho password:**

KeePass quy ước như sau:

* Ký tự thường: 1
* Ký tự hoa: 2
* Ký tự số: 4
* Ký tự đặc biệt đơn giản: 8
* Ký tự đặc biệt khác: 16
* Ký tự cao hơn còn lại: 32

Mỗi ký tự trong chuỗi mật khẩu sẽ thuộc một trong các loại ký tự trong bảng. Thực hiện phép OR tất cá các ký tự trong chuỗi mật khẩu. Mục đích của việc này là bật bit tương ứng với ký tự có trong chuỗi.

Sau đó sử dụng phép AND bit để kiểm tra mật khẩu có chứa tập ký tự nào thì cộng thêm tập đó và không gian ký tự. Nếu kết quả của phép AND khác 0 thì cộng thêm vùng không gian ký tự đó.

Cuối cùng tính số bit trung bình của chuỗi mật khẩu.

Tính theo %(q) so với 128bit: nếu bitperchar lớn hơn 128 thì q=100.

Ngược lại, q =bitperchar\*100

**c. Ưu điểm**

Kết hợp được đánh giá the không gian ký tự (charspace) và tính entropy.

Không gian ký tự tương đổi chi tiết.

### 2. Mozilla.

Như đã biết, không gian kí tự có ảnh hưởng rất lớn đến độ mạnh của password. Không gian kí tự càng lớn thì password càng khó phá. Dựa vào đó Mozilla đánh giá password theo không gian kí tự. Đây là phương pháp đơn giản nhưng khá hiệu. Mozilla định nghĩa 4 tập không gian kí tự ảnh hưởng đến độ mạnh của password:

* Số kí tự hoa (upp).
* Số kí tự số (dig).
* Số kí tự đặc biệt (spe).
* Độ dài password (len).

Cách tính:

* Nếu len = 5 thì p=p+len\*10
* Nếu len >5 thì p=p+50
* Nếu dig = 3 thì p=p+dig\*10
* Nếu dig > 3 thì p=p+30
* Nếu spe =3 thì p=p+spe\*15
* Nếu spe > 3 thì p=p+45
* Nếu upp = 3 thì p=p+upp\*10
* Nếu upp > 3 thì p=p+30

**Kết quả:**p=p-20(%)  
**Nhận xét:**Khác với Keepass, với phương pháp này độ dài password không quyết định hoàn toàn tới độ mạnh của password. Ví dụ: password có 10 kí tự gồm 5 kí tự hoa, 5 kí tự thường có độ mạnh bằng một password dài 15 kí tự gồm 5 kí tự hoa và 10 kí tự thường.

**Ưu điểm:**

Ưu điểm của phương pháp này là đơn giản.

Tính toán nhanh

Không tính toán Entropy, nên không cần sử dụng các phép toán phức tạp như logarit.

**Nhược điểm:**

Số tập kí tự ít (4 tập)

Đánh giá tương đối thoáng

1. **Tìm hiểu ngôn ngữ lập trình ASP.NET để xây dựng website.**
2. **APS .NET là gì?**

ASP.NET là nền tảng phát triển web (web application frameword), cung cấp một mô hình lập trình, cơ sở hạ tầng phần mềm toàn diện và các dịch vụ cần thiết để xây dụng các ứng dụng web động mạnh mẽ cho máy tính cũng như trên các thiết bị di động.

ASP.NET là một phần của nền tảng Microsoft.NET. Ứng dụng ASP.NET được biên dịch mã, được viết bằng cách sử dụng mở rộng và tái sử dụng các thành phần hoặc đối tượng trong nền tảng NET. Các mã này được sử dụng cho toàn bộ hệ thống phân cấp của các class trong .NET

Các ứng dụng ASP.NET có thể được viết bằng bất kỳ ngôn ngữ nào sao đây:

* C#
* Visual Basic.Net
* Jscript
* J#

ASP.NET được sử dụng để tạo ra các tương tác, dữ liệu điều khiển các ứng dụng web trên internet. ASP.NET bao gồm một số lượng lớn các controls như là các text box, button và labels cho assembling, và các thao tác mã để tạo ra các trang HTML.

1. [**ASP .NET**](http://asp.net/)**Core là gì?**

[ASP.NET](http://asp.net/) Core là một open-source mới và framework đa nền tảng (cross-platform) cho việc xây dựng những ứng dụng hiện tại dựa trên kết nối đám mây, giống như web apps, IoT và backend cho mobile. Ứng dụng [ASP.NET](http://asp.net/) Core có thể chạy trên .NET Core hoặc trên phiên bản đầy đủ của .NET Framework. Nó được thiết kế để cung cấp và tối ưu development framework cho những dụng cái mà được triển khai trên đám mây (clound) hoặc chạy on-promise. Nó bao gồm các thành phần theo hướng module nhằm tối thiểu tài nguyên và chi phí phát triển, như vậy bạn giữ lại được sự mềm giẻo trong việc xây dựng giải pháp của bạn. Bạn có thể phát triển và chạy những ứng dụng [ASP.NET](http://asp.net/) Core đa nền tảng trên Windows, Mac và Linux. Đồng thời nó đã trở thành một mã nguồn mở. Đây là một thay đổi rất lớn và theo mình là quan trọng nhất của [ASP.NET](http://asp.net/) Core. Điều mà trước đây khó có một lập trình viên nào có thể nghĩ đến. Có lẽ đó cũng là một xu thế mà các ngôn ngữ lập trình hiện nay đang hướng tới.

1. [**ASP.NET**](http://asp.net/)**Core ra đời trong hoàn cảnh nào?**

Bản phát hành đầu tiên của [ASP.NET](http://asp.net/) đã xuất hiện cách đây 15 năm trước, nó là một phần của .NET Framework. Từ đó, hàng triệu lập trình viên đã sử dụng nó để xây dựng những ứng dụng web tuyệt vời, và trên những năm đó Microsoft đã phát triển thêm nhiều tính năng mới.

[ASP.NET](http://asp.net/) Core có một số thay đổi kiến trúc lớn, đó là kết quả của việc học hỏi rất nhiều từ các framework module hóa khác. [ASP.NET](http://asp.net/) Core không còn dựa trên System.Web.dll nữa. Nó được dựa trên một tập hợp các gói, các module hay cũng được gọi là các Nuget packages. Điều này cho phép bạn tối ưu ứng dụng của bạn để chỉ bao gồm những packages nào cần thiết. Lợi ích của nó là giúp cho ứng dụng nhỏ hơn, bảo mật chặt chẽ hơn, giảm sự phức tạp, tối ưu hiệu suất hoạt động và giảm chi phí, thời gian cho việc phát triển.

Với [ASP.NET](http://asp.net/) Core bạn đạt được những nền tảng cải tiến dưới đây:

* Hợp nhất việc xây dựng web UI và web APIs
* Tích hợp những client-side frameworks hiện đại và những luồng phát triển
* Hệ thống cấu hình dựa trên môi trường đám mây thật sự
* Dependency injection được xây dựng sẵn
* HTTP request được tối ưu nhẹ hơn
* Có thể host trên IIS hoặc self-host trong process của riêng bạn
* Được xây dựng trên .NET Core, hỗ trợ thực sự app versioning
* Chuyển các thực thể, thành phần, module như những NuGet packages
* Những công cụ mới để đơn giản hóa quá trình phát triển web hiện đại
* Xây dựng và chạy đa nền tảng(Windows, Mac và Linux)
* Mã nguồn mở và tập trung vào cộng đồng lớn.

1. **Xây dựng web UI và web API sử dụng**[**ASP.NET**](http://asp.net/)**Core MVC**

Bạn có thể tạo ra các ứng dụng web có thể testing theo mô hình MVC(Model-View-Controller).

Bạn có thể xây dựng HTTP services cái mà hỗ trợ nhiều định dạng và có đầy đủ hỗ trợ cho nội dung của dữ liệu trả về.

Razor cung cấp một ngôn ngữ hiệu quả để tạo Views.

Tag Helper cho phép code server side tham gia vào việc tạo và render phần tử HTML.

Model Binding tự động ánh xạ dữ liệu từ HTTP request tới tham số của method action.

Model Validation tự động thực hiện validate client và server

### ****Đặc điểm riêng biệt của** **ASP.NET Core:****

1. **Hỗ trợ đa nền tảng.**

Ngày nay bạn có thể phát triển và chạy ASP.NET trên cả Windows, Mac, và Linux. Và nếu trên Windows bạn có thể thể sử dụng công cụ tốt nhất Visual Studio 2015 để tạo, quản lý và gỡ lỗi các ứng dụng ASP.NET Core của bạn, thì nay trên bất kỳ nền tảng nào bạn có thể sử dụng Visual Studio Code. Visual Studio Code là một trình soạn thoải với các plugin có hỗ trợ để chỉnh sửa các ứng dụng ASP.NET Core của bạn.

1. **Nguồn mở.**

Ngày nay mã nguồn và tài liệu ASP.NET đã được Microsoft mở tất cả. Các mã nguồn giờ đã có sẵn trong Github bạn giờ có thể tải hay thay đổi bất kỳ mã nào mà bạn thích. Nếu bạn có bất kỳ một góp ý cải tiến gì đó, bạn có thể gửi một yêu cầu đến cho microsoft để xem xét và kết hợp.

Tương tự như vậy, tất cả các tài liệu cũng là mã nguồn mở và có sẵn trong bài viết docs.asp.net. Mỗi trang trên đó điều có chức năng “chỉnh sửa trang này” ở phía trên và bạn có thể chỉnh sửa các tài liệu từ Microsoft.

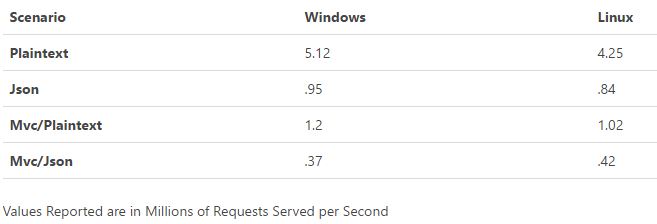
1. **Hỗ trợ đầy đủ cho framework.**

Một thông tin hữu ích là asp.net core dẫu trở thành mã nguồn mở nhưng Microsoft vẫn sẽ cung cấp hỗ trợ trong 3 năm cho mỗi bản phát hành lớn nhỏ của họ.

1. **Hiệu suất.**

Microsoft giới thiệu máy chủ mới web Kestrel chạy trong host IIS của bạn hoặc chạy sau một host process khác. Kestrel hiện tại là máy chủ .NET chạy nhanh nhất hiện nay.

Dưới đây là bảng hiệu xuất đánh giá từ Microsoft



1. **Hỗ trợ xây dựng bằng Dependency Injection**

Nếu mà giải thích thì chỉ vài dòng thì sẽ hơi khó hiểu ASP .NET Core đã được xây dựng trong Dependency Injection. Dependency Injection là một mẫu thiết kế cho phép các phụ thuộc của một class được injected như các đối tượng được yêu cầu trong ứng dụng của bạn. Với ASP.NET Core, Microsoft đã cung cấp một Dependency Injection mà bạn có thể sử dụng để xác định sự phụ thuộc được đưa vào Controller, View của bạn, hoặc bất kỳ lớp học khác mà framework sẽ tạo ra cho bạn. Bạn có thể bắt đầu việc cấu hình thông qua phương thức ConfigureServices trong tập tin Startup.cs.

1. **Một Framework duy nhất.**

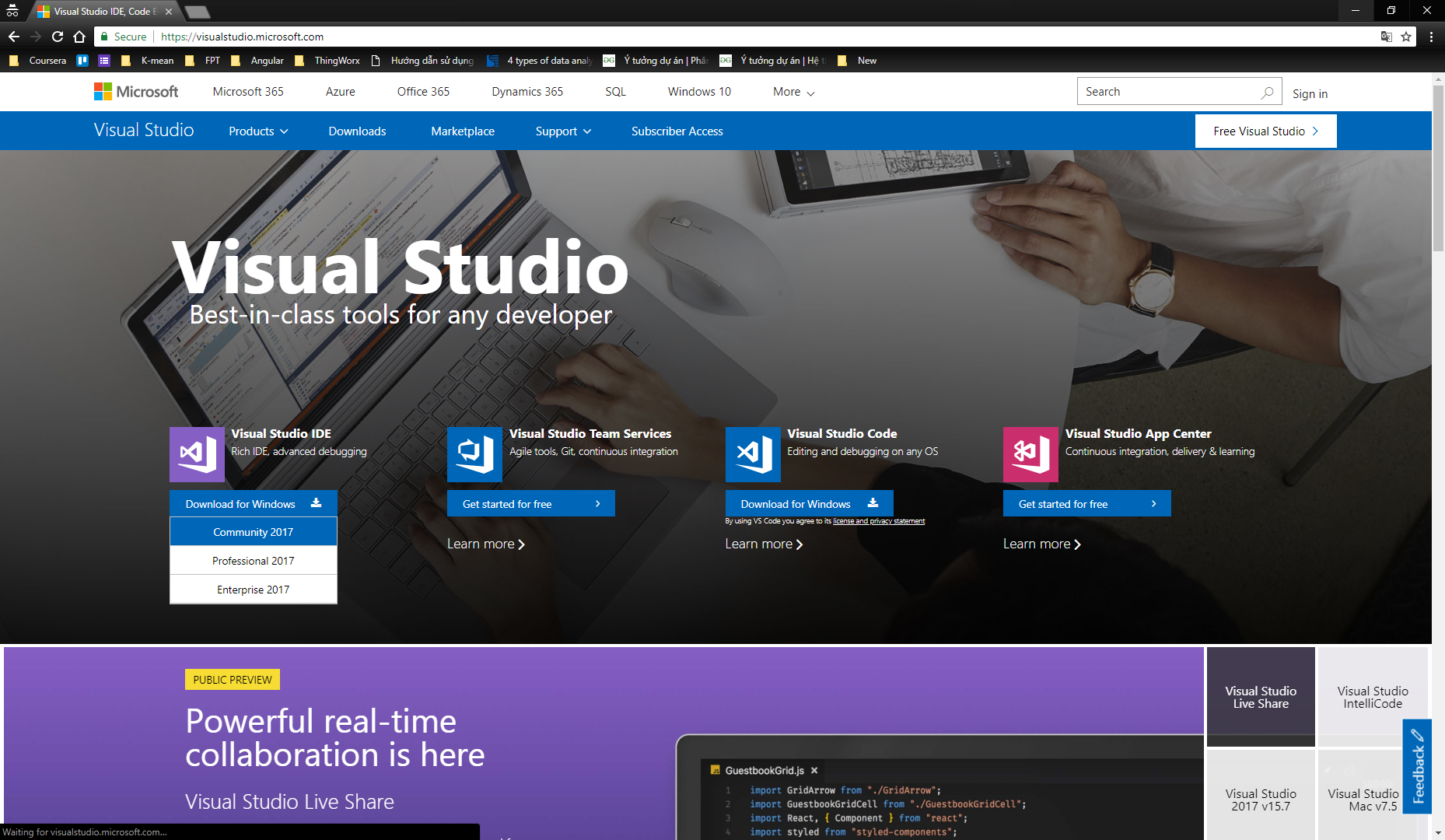
Trong ASP.NET Core, Microsoft đã đưa tất cả các framework vào một framework duy nhất vừa nhẹ hơn và vừa có những tính năng của MVC và WebAPI. Với việc sáp nhập này của MVC và Web API, mọi thứ đơn giản hơn nhiều khi bạn không cần phải cân nhắc định tuyến khác nhau, an toàn, hoặc các bộ lọc cho một ApiController so với MVC Controller. Tất cả các lớp Controller hiện giờ có thể xử lý các yêu cầu sử dụng API Web hoặc cách tiếp cận MVC.

1. **MVC Helpers Tag:**

Với ASP.NET Core, Microsoft đã giới thiệu tag helpers để tạo ra mã phía client từ .NET và làm cho nó dễ dàng hơn để tái sử dụng trong Razor markup. Nó được tham chiếu trong đánh dấu phía máy chủ của bạn như thể họ là một tag HTML mà bạn được sử dụng. Công cụ Razor sẽ nhận ra thẻ và thực thi các mã .NET có liên quan tương ứng với nó.

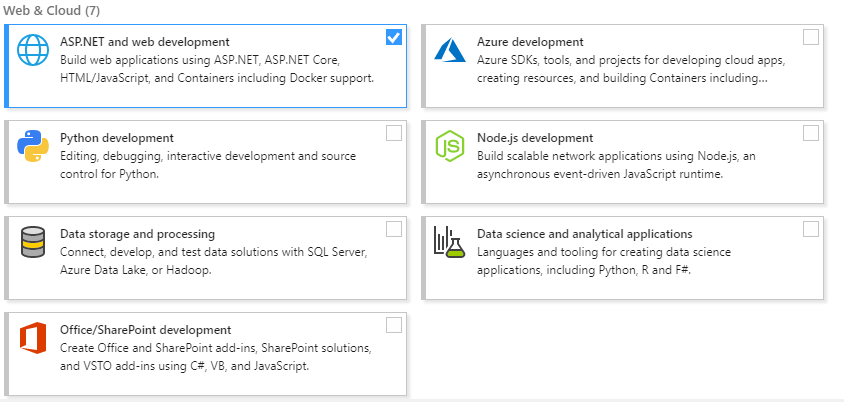
1. **Xây dựng ứng dụng đầu tiên**
2. **Cài đặt Visual Studio and .NET Core**

Cài đặt Visual Studio Community 2017. Truy cập vào trang chủ theo địa chỉ: <https://visualstudio.microsoft.com/>. Chọn Visual Studio IDE và phiên bản Community 2017 để dowload.

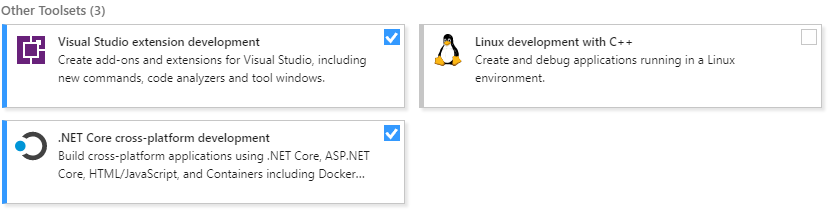


Chạy bản cài đặt và chọn những workloads bên dưới:

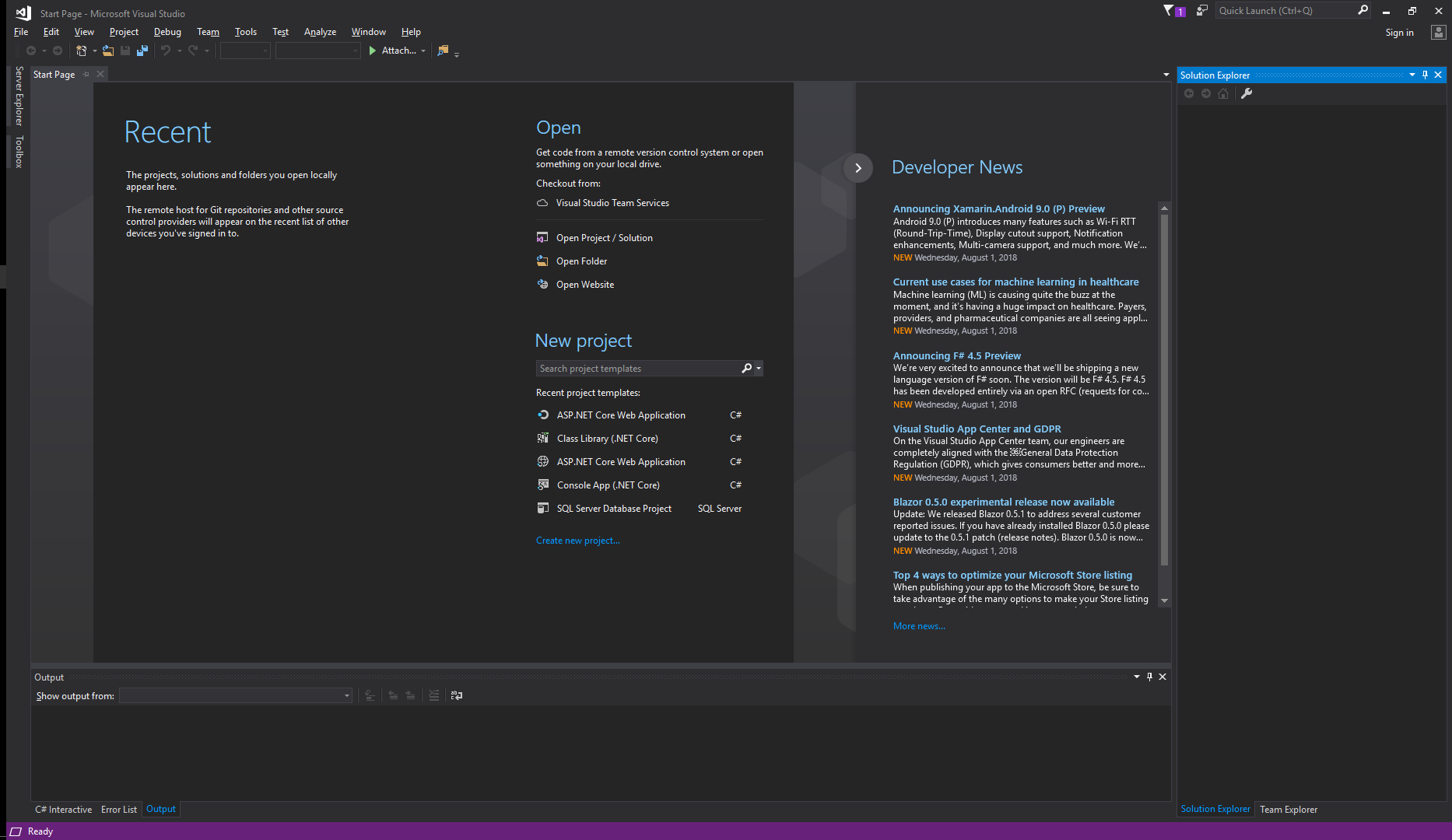
* [ASP.NET](http://asp.net/) và web development.



* .NET Core cross-platform development.



Nhấn Install để bắt đầu dowload và cài đặt các gói thư viện. Quá trình này có thể mất vài giờ. Giao diện sau khi cài đặt xong và chạy Visual Studio.



1. **Tạo một web app**

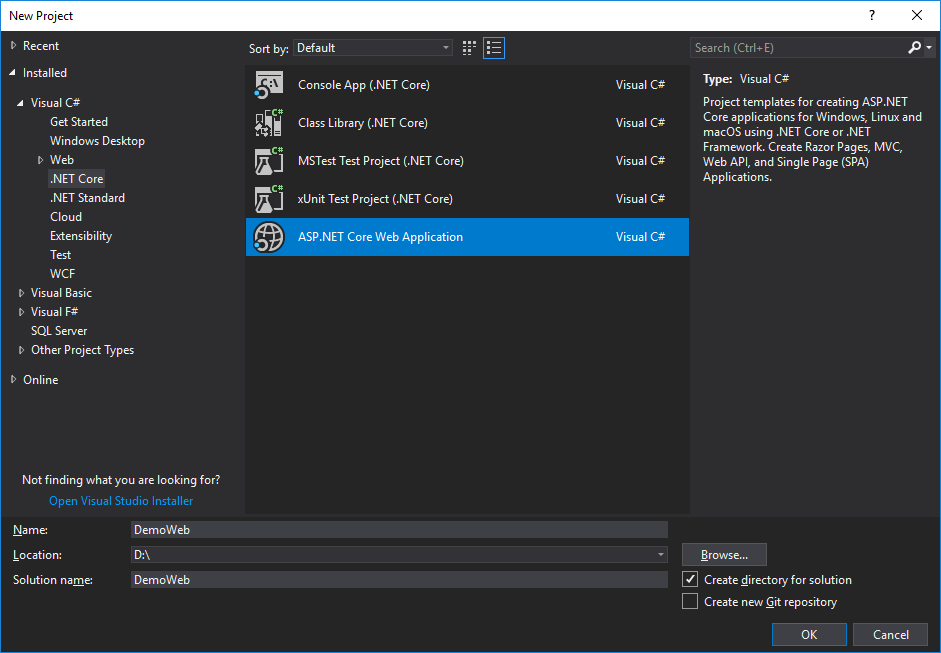
Từ Visual Studio, chọn File > New > Project. Hoàn thành hộp thoại New Project:

- Trong phần bên trái, chọn .NET Core.

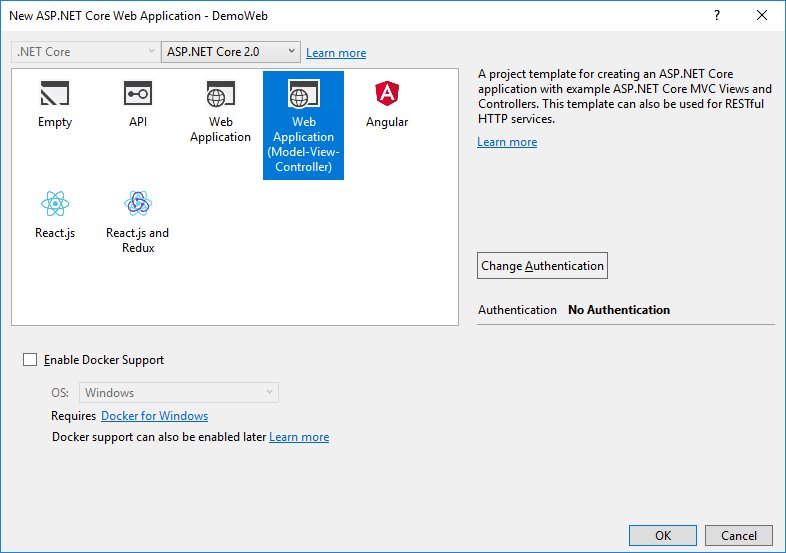
- Trong phần giữa, chọn [ASP.NET](http://asp.net/) Core Web Application(.NET Core).

- Tên của project ví dụ là ‘DemoWeb".

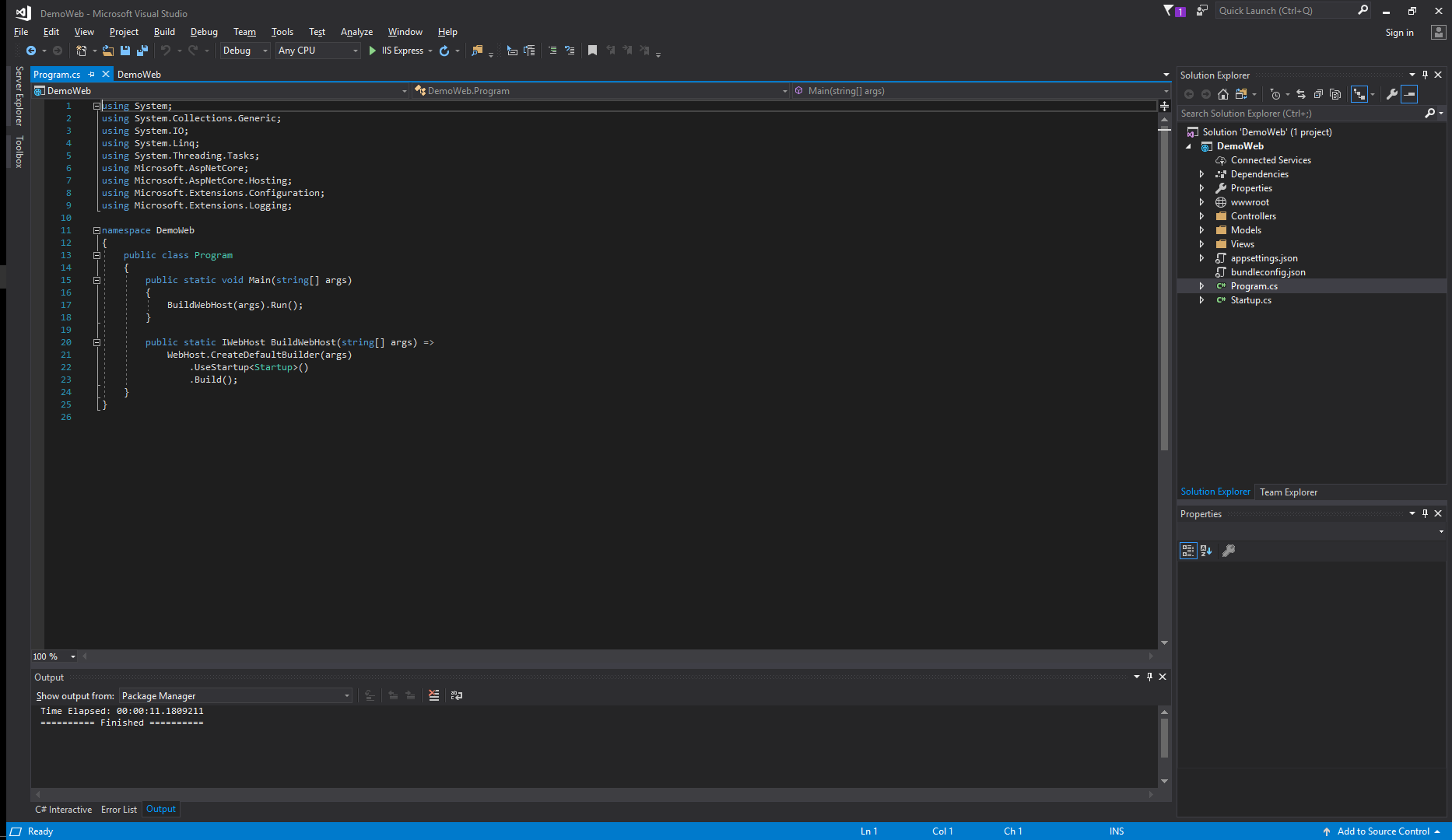
- Nhấn OK.



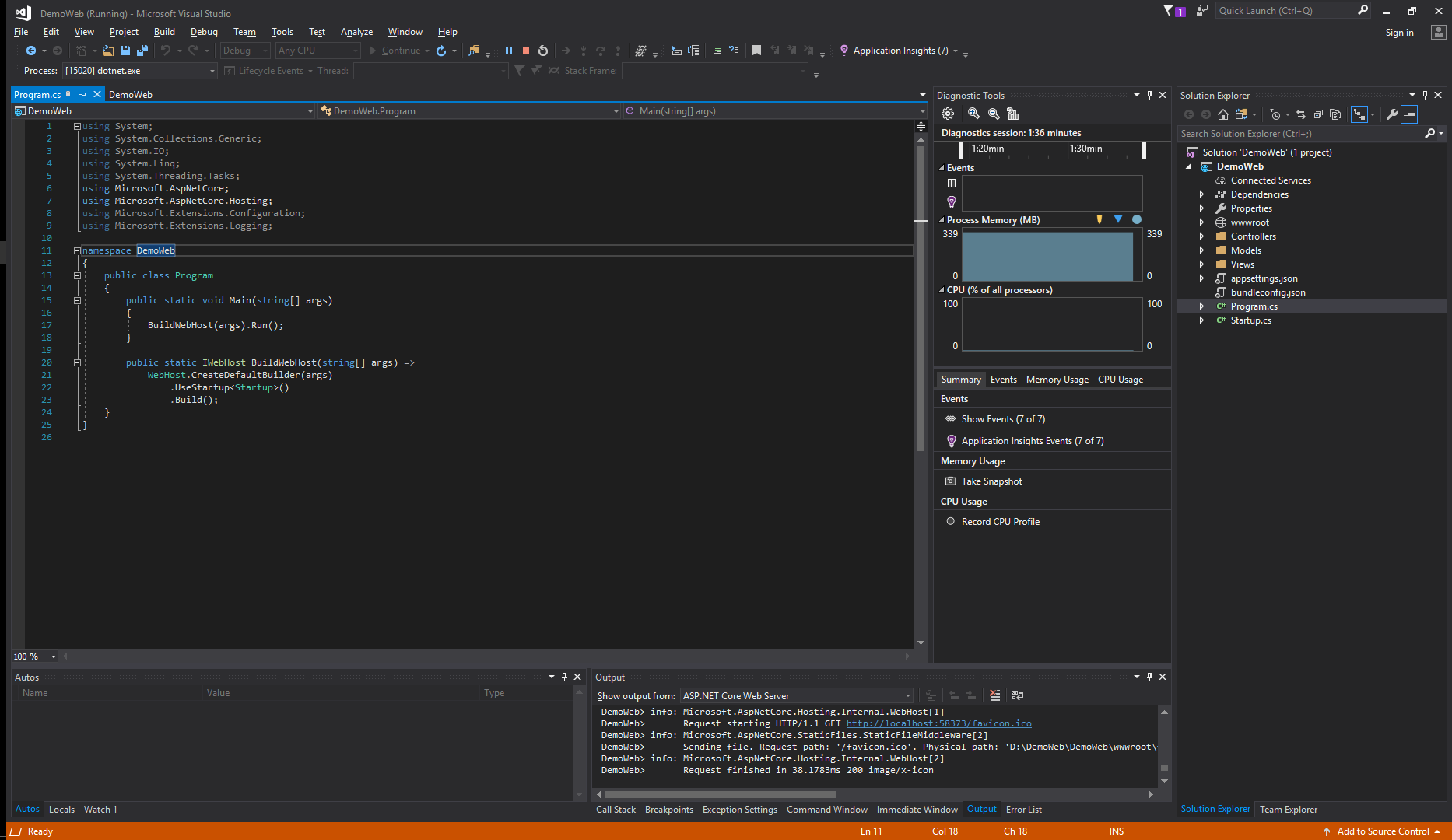
Trong hộp thoại New ASP>NET Core Web Application, chọn Web Application (Model-View-Controller), nhấn OK.



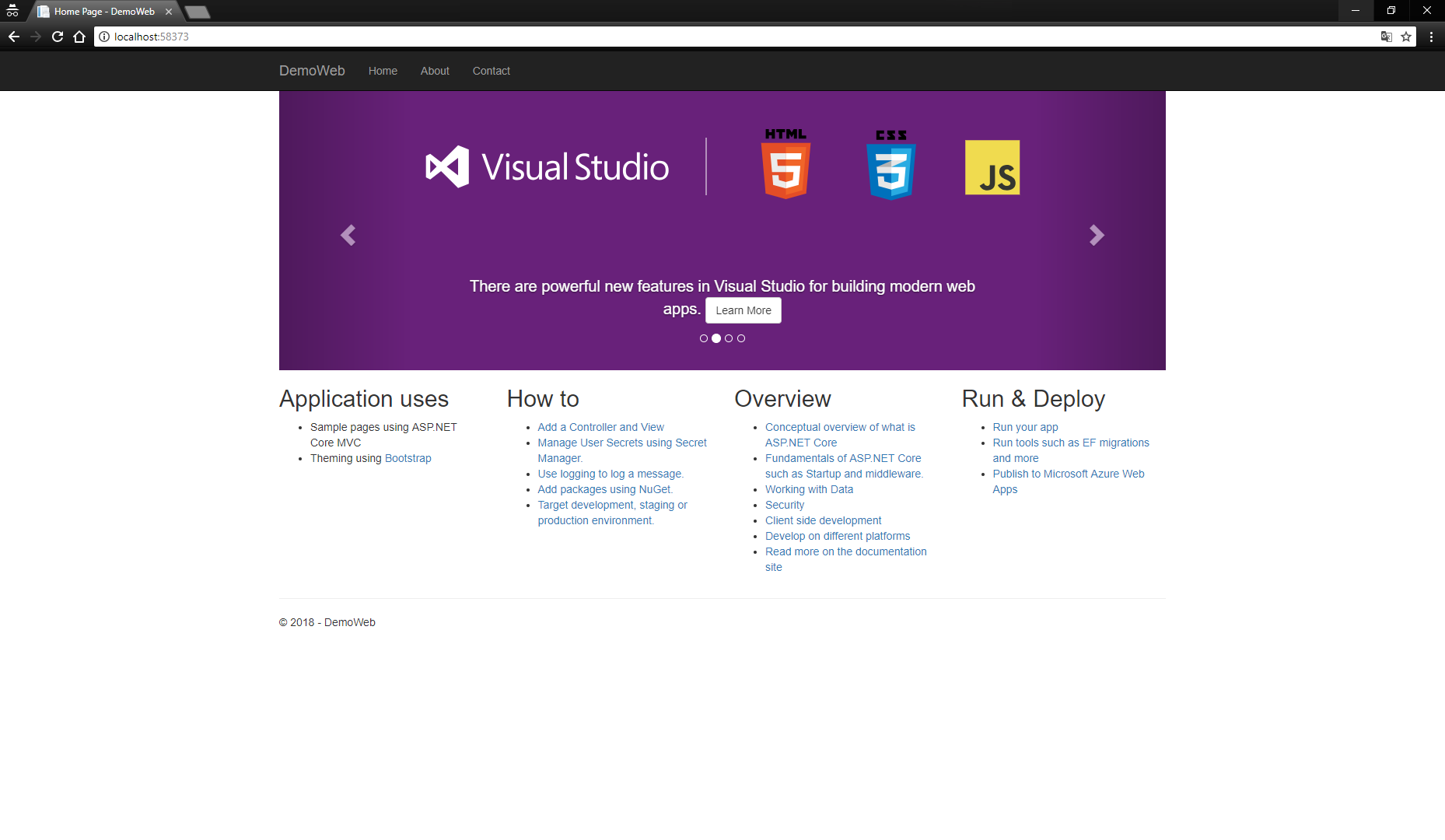
Visual Studio đã sử dụng một template mặc định cho dự án MVC bạn vừa tạo. Bạn có một ứng dụng làm việc bây giờ bằng việc điền tên dự án và một vài tùy chọn. Điều này là một dự án bắt đầu đơn giản.



Nhấn F5 để chạy ứng dụng trong chế độ debug hoặc Ctrl-F5 trong chế độ non-debug.



Các bạn có thể quan sát kết quả của trang chủ.



Visual studio bắt đầu với IIS Express và chạy ứng dụng của bạn. Chú ý rằng thanh địa chỉ hiển thị dạng localhost:port# và không phải kiểu như [example.com](http://example.com/). Đó là bởi vì localhost là hostname chuẩn cho máy tính local của bạn. Khi Visual Studio tạo một dự án web, một ramdom port được sử dụng cho web server. Trong hình trên, số cổng là 58373. Khi bạn chạy ứng dụng, bạn sẽ thấy một số cổng khác.

Việc chạy ứng dụng với Ctrl-F5(non-debug) cho phép bạn làm code thay đổi, lưu file, refresh trình duyệt và thấy ngay những thay đổi của code. Nhiều lập trình viên thích sử dụng chế độ non-debug để nhanh chóng chạy ứng dụng và xem những thay đổi. Điều này là tương tự với ngôn ngữ thông dịch như php, python hay ruby,..

Chạy với chế độ self-host. Một điểm mới trong [ASP.NET](http://asp.net/) Core là các bạn có thể chạy server như một ứng dụng console thay vì host trên IIS. Để dùng chế độ này, tại điểm bắt đầu với IIS chúng ta thay bằng bắt đầu với self-host(Tên chính là tên ứng dụng).

**THỰC HÀNH.**

Xây dựng Website đánh giá độ an toàn của mật khẩu.

**Setup**

Các bước setup visual studio phía trên. Nếu chạy console di chuyển đến thư mục chứ file Program.cs chạy lệnh theo thứ tự:

**$dotnet build**

**$dotnet run**

Nếu găp lỗi hãy chạy lệnh $dotnet –help để kiểm tra phiên bản của .NET

**Thuật toán đánh giá**

***Thuật toán đánh giá dựa trên các yêu cầu về cách đặt mật khẩu của các hệ thống hiện tại trên internet và các website đánh giá mật khẩu. Tình toán thời gian độ phức tạp và đưa ra gọi ý tạo mật khẩu.***

**TT1: Đánh giá bằng không gian tìm kiểm của chuỗi mật khẩu.**

Sử dụng bảng mã của thuật toán Mozilla gồm 4 tập ký tự:

* Chữ thường
* Chữ in hoa
* Số
* Ký tự đặc biệt

Mỗi tập ký tự có không gian tìm kiếm riêng:

* Chữ thường: 26
* Chữ in hoa: 26
* Số: 10
* Ký tự: 33

*(Tham khảo trong Contants.Alphabet.SpaceDeth và Contants.Alphabet.Regular)*

Dựa vào tập ký tự ta chia mức độ mạnh yếu của mật khẩu. Quy định mức độ mật khẩu dựa trên quy tắc của dãy fibonaci.

* Chữ thường: 1
* Chữ in hoa: 2
* Số: 4
* Ký tự: 8

Mỗi tập ký tự kết hợp với nhau thì sẽ tạo ra một mức độ mới bằng cách công mức độ của các tập đó lại. Tổng cộng ta được 15 mức độ. Từ các mức độ đó ta tính được tương ứng với mỗi mức độ là một không gian tìm kiếm, Tuy nhiên sẽ có không gian tìm kiếm giống nhau chỉ khác các tập ký tự ví dụ như mức độ 1 và 2,...

*(Tham khảo trong Contants.Alphabet.Level và Contants.Alphabet.LevelNumber và Contants.Alphabet.LevelSpace )*

Quy định mức độ mạnh yếu dựa vào không gian tìm kiếm:

* Rất yếu: 1, 2, 4, 8 (gồm 1 tập ký tự)
* Yếu: 3, 5, 6, 9, 10, 12 (gồm 2 tập ký tự)
* Trung bình: 7, 11, 13, 14 (gồm 3 tập ký tự)
* Mạnh:15 (gồm 4 tập ký tự)
* Rất mạnh: đặt mạnh và có độ dài ký tự không nhỏ hơn 10.

*(Tham khảo hàm LevelPassword() trong Web.Services.AnalysisPasswordService)*

**TT2: Đánh giá dựa trên từ điển.**

Sử dụng bảng từ điển mật khẩu để kiểm tra xem nó thuộc top bao nhiêu nếu đã thuộc top mật khẩu thì mặc định là Rất yếu.

*Tham khảo:*

*- TopPassword:* [*https://en.wikipedia.org/wiki/Wikipedia:10,000\_most\_common\_passwords*](https://en.wikipedia.org/wiki/Wikipedia:10,000_most_common_passwords)

*- ListTopPassword: Web.Resources.TopPassword.txt*

*- Service: Web.Services.CheckTopPasswordService*

**Đánh giá độ dài của chuỗi mật khẩu**: Giả sử độ dài tối đa của mật khẩu là 30.

* Quá ngắn: len<6
* Ngắn: 6<len<8
* Trugn bình: 8<len<15
* Dài: 15<len<20
* Quá dài: 20<len<30

*(Tham khảo Contants.Lenght.Number và Web.Services.CheckLengthPasswordService)*

**Đánh giá thời gian:** Giả sử hệ thống đánh giá sử dụng một máy tính có tốc độ tính toán cao và có khả năng sinh ra ngẫu nhiên 10 tỷ (1e10) mật khẩu mỗi giây. (Có thể thay đổi trong file Contants, filed BILLON)

Kết quả từ việc đánh giá không gian tìm kiểm sẽ được sử dụng để tính toán thời gian. Vì hệ thống sinh ngẫu nhiên nên ta sẽ có được thời gian tối đa để bẻ khóa mật khẩu đó (hay còn gọi là vét cạn (BruteForce)).

**Công thức tính theo xác xuất thống kê:**

T = S^1 +S^2+S^3+…+S^L

- T: tổng thời gian bẻ khóa

- S: Không gian tìm kiếm.

- L: độ dài mật khẩu.

**Ví dụ :** mật khẩu là 1a

- Ta có không gian tìm kiếm là 10 + 26 = 36

- Độ dài là 2.

Tính thời gian:

Với độ dài 1: có 36 mật khẩu hay có 36 lần thử

Với độ dài 2: tương tự có 36^2 lần thử = 1296 lần thử

Tổng ta có 1332 lần thử, với máy tính trên thì ta mất 1332/1e10s để bẻ khóa.

**Thời gian tính toán chỉ mang tính tương đối:**

Ví dụ: 1 năm 2 tháng 10 ngày thì chỉ in ~ 1 năm (Lấy thời gian lớn nhất, khoảng thời gian phía sau không hiển thị vì chỉ mang tính tương đổi không cần chính xác.)

Ví dụ: Nếu ngẫu nhiên sinh trúng mật khẩu thì những phép thử còn lại không cần thiết nên thời gian sẽ ngẵn hơn. Có thể trong lần thử đầu tiên đã chính xác.

*(Tham khảo Web.Service.AnalysisPasswordService)*

**Lưu ý:**

- Giới hạn độ dài mật khẩu không quá 30 ký tự. (Thời gian tính toán sẽ không đảm bảo vì giới hạn lưu trữ của kiểu dữ liệu). *Tham khảo kiểu dữ liệu ulong.*

- Thời gian là đơn vị số nguyên dương: nếu dưới 1s thì trả về Ngay Lập Tức, nếu năm quá nhiều thì sẽ hiển thị theo định dạng phân cách bằng dấu “,”.

- Các thuộc tính đánh giá chỉ mang tính chất riêng cho thuộc tính đó.  
 *Ví dụ: “123456” giả sử mất 100 năm để bẻ khóa nhưng nó thuộc top mật khẩu thì vẫn đánh giá là yếu.*

**Luồng chạy của hệ thống**

- Khởi tạo các service, render html. *(Tham khảo dependency của .NET Core)*

**B1. Kiểm tra trong từ điển:**

**-** Đọc file lấy list password

- So sánh password nếu có thì lấy số top trả về

**B2. Kiểm tra độ phức tạp**

**-** Tạo model PasswordDetailModel để lưu trữ các giá trị. Tham khảo các giá trị trong Web.Model

**B3. Kiểm tra dộ dài**

**B4. Tính toán thời gian.**

**B5. Đưa ra gợi ý tạo mật khẩu phù hợp**

**-** Dựa trênPasswordDetailModel để hiển thị các gợi ý.

**Một số câu hỏi**

1. **Thời gian bẻ khóa chỉ mang tính tương đối thì việc đánh giá đúng hay sai?**

**-** Đánh giá cũng chỉ mang tính tương đối vì có nhiều yếu tố khác ví dụ như sử dụng siêu máy tính có thể tạo ra nhiều mật khẩu hơn thì có thể thời gian nhanh hơn hoặc sử dụng nhiều siêu máy tính 1 lúc thì thời gian sẽ khác.

1. **Không gian tìm kiếm là gì.**

**-** Là khoảng giới hạn tìm kiếm mà ta cần.

1. **Tại sao lại sử dụng thuật toán BurteForce để tính thời gian.**

**-** Vì có rất nhiều các để dò tìm mật khẩu, cách hiệu quả nhất là dùng BurteForce.

Ví dụ có mật khẩu là số thì ta chỉ cần dò từ 0 đến 9 nhưng có thêm chữ thì cần từ 0-**>**9 và a->z

Ví dụ mật khẩu gồm 1 ký tự thì ta chỉ cần chạy qua không gian tìm kiếm 1 lần nhưng 2 ký tự thì ta phải đi qua nhiều hơn.

Các ví dụ là trường hợp ta biết mật khẩu có không gian và độ dài nhưng nếu ko biết thì ta chỉ có cách thử hết tất cả.

1. **Ngoài cách dò tìm thì còn nhiều cách khác, vậy làm sao để mật khẩu an toàn.**

**-** Thay đổi mật khẩu thường xuên

**-** Sử dụng các tập ký tự khác nhau và ngẫu nhiên

- Sử dụng phương pháp xác thực khác.