Networks

1. What is the Web?

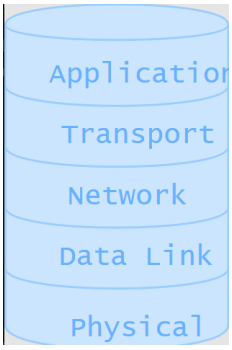
Web, đơn giản là một mạng lưới (network) lan tỏa trên toàn cầu kết nối nhiều thiết bị với nhau và cho phép chúng giao tiếp với nhau. Các trang web (website) trên Internet được lưu trữ trên các thiết bị được gọi là máy chủ (server), và khi bạn tương tác với một trang web trên Internet, thực chất bạn đang trao đổi dữ liệu với máy chủ (server) mà trang web được lưu trữ trên đó. Thiết bị mà bạn sử dụng để truy cập trang web được gọi là khách hàng (client) trong ngữ cảnh của web. Nói một cách đơn giản, web cho phép trao đổi dữ liệu giữa các client và server thông qua một số cơ chế phức tạp.

Network layers

Vì web là một mạng lưới phức tạp và lan rộng, các máy trong mạng thường được chia thành các tầng trừu tượng, mỗi tầng thực hiện một nhiệm vụ cụ thể hỗ trợ quá trình giao tiếp tổng thể. Các tầng được liệt kê dưới đây:

1. Tầng ứng dụng (Application Layer)
2. Tầng vận chuyển (Transport Layer)
3. Tầng mạng (Network Layer)
4. Tầng liên kết dữ liệu (Data Link Layer)
5. Tầng vật lý (Physical Layer)

Mỗi tầng được xây dựng trên tầng trước đó và có các giao thức thực hiện các chức năng cụ thể liên quan đến quá trình trao đổi dữ liệu.



Physical Layer

Tầng vật lý (Physical Layer) của một máy tính đề cập đến việc dây cáp và mạch vật lý được sử dụng để kết nối máy tính đó vào mạng.

Data Link Layer

Tầng liên kết (Data Link Layer) dữ liệu chịu trách nhiệm truyền dữ liệu từ một máy tính cụ thể đến một thiết bị hoặc máy tính chỉ cách đó một liên kết.

Network Layer

Tầng mạng (Network Layer) chịu trách nhiệm kết nối bất kỳ hai máy tính nào trên Internet. Nó cung cấp kết nối toàn cầu và cho phép các hệ thống cuối có thể giao tiếp với nhau trên quy mô lớn, vượt ra ngoài những gì tầng liên kết dữ liệu có thể cung cấp.

Transport Layer

Tầng vận chuyển (Transport Layer) chịu trách nhiệm kết nối các ứng dụng trên Internet. Nó phân mảnh dữ liệu đến từ một nguồn duy nhất và truyền nó đến ứng dụng mà nó được dự định. Mục đích cơ bản của tầng vận chuyển là cung cấp giao tiếp từ quá trình này sang quá trình khác; nó cho phép hai quá trình riêng biệt trên cùng một máy hoặc máy khác nhau gửi tin nhắn cho nhau. Để làm điều này, nó sử dụng socket, đó là cổng thông tin cho một quá trình (gateway to a process). Nói cách khác, socket là phương tiện thông qua đó các tin nhắn được nhận và gửi đi bởi một quá trình.

Application Layer

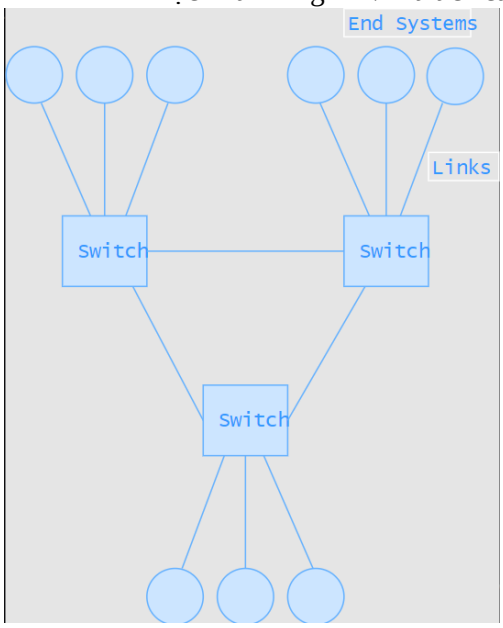
Tầng ứng dụng (Application Layer) chịu trách nhiệm cho giao tiếp từ quá trình này sang quá trình khác trên Internet. Đây là tầng cao nhất trong cấu trúc phân cấp và ứng dụng được xây dựng trên tầng này. Tầng ứng dụng cung cấp giao diện giao tiếp và các dịch vụ cho người dùng cuối để ứng dụng có thể giao tiếp với các quá trình riêng lẻ.

2. How Does It Work

Switches to connect devices

Về cơ bản, việc truyền thông là khá trực quan. Các client gửi tin nhắn đến server yêu cầu dữ liệu, và server phản hồi với dữ liệu cần thiết. Nhưng làm thế nào dữ liệu này được truyền đi? Câu trả lời đòi hỏi định nghĩa cấu trúc của Internet trước.

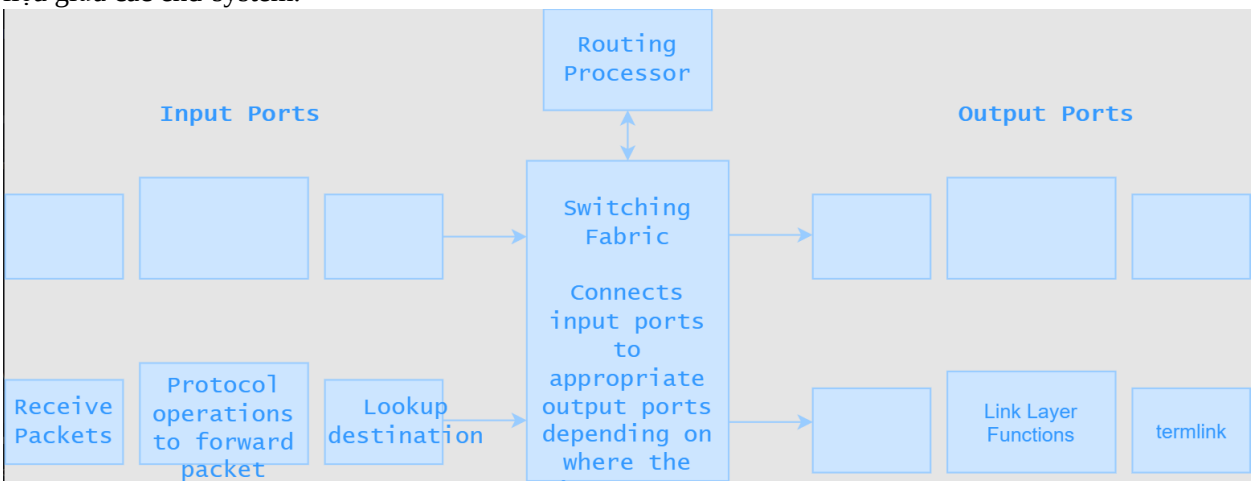
Internet bao gồm các thiết bị được gọi là switch (chuyển mạch) giúp kết nối từng thiết bị với tất cả các thiết bị khác trên mạng. Các thiết bị này được gọi là end-systems (hệ thống cuối), mà thực chất chỉ là một thuật ngữ phức tạp để chỉ máy tính bạn đang sử dụng để truy cập trang web này!
Các end-system được kết nối với switch thông qua các liên kết, và tất cả các switch đều kết nối với nhau, đảm bảo rằng mọi end-system trên Internet được kết nối ngầm với tất cả các end-system khác.



Ngoài việc kết nối các end-system với nhau, switch còn giúp giao tiếp giữa bất kỳ hai end-system nào bằng cách chuyển tiếp các gói tin (packet) theo đường dẫn (path) mà chúng biết tồn tại giữa nguồn và đích của gói tin. Về cơ bản, switch lưu trữ các đường dẫn được xác định trước giữa các end-system và chuyển tiếp các gói tin giữa chúng.

Routers

Một thuật ngữ phổ biến khi nói về mạng là "router" (bộ định tuyến). Router có chức năng tương tự như switch trong việc kết nối các end-system với phần còn lại của mạng. Tuy nhiên, router thực sự rất khác biệt so với switch vì nó có khả năng cho phép tra cứu địa chỉ đích và xác định đường dẫn ngắn nhất hoặc ít tải nhất từ nguồn của một gói tin đến đích của nó. Router, do đó, là một phiên bản mạnh mẽ hơn của switch và Internet bao gồm một sự kết hợp của cả router và switch để tạo điều kiện cho việc truyền dữ liệu giữa các end-system.



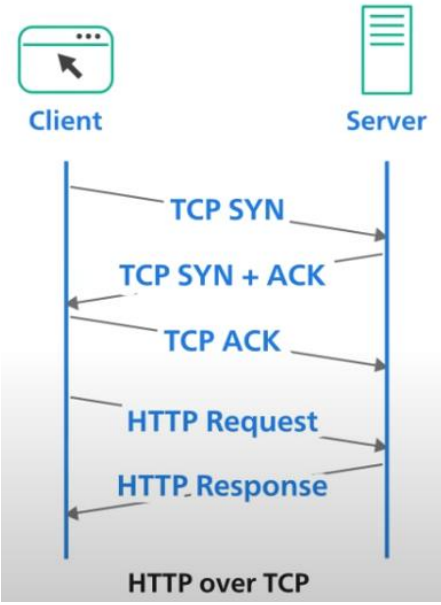
Data Packets

Sau khi đã xác định cách Internet được cấu trúc để đảm bảo kết nối, bây giờ chúng ta hãy nói về cách dữ liệu thực sự được vận chuyển qua mạng. Điều này được thực hiện bằng cách chia dữ liệu, chỉ là một tập hợp các bit, cần được truyền thành nhiều mảnh nhỏ hơn của các bit được gọi là gói tin (packet), sau đó gửi mỗi gói tin (packet) đến đích của nó một cách độc lập. Lý do cho việc này là bạn không thể luôn luôn gửi một lượng lớn dữ liệu trong một gói tin (packet) duy nhất vì rất có thể dữ liệu trong các gói tin (packet) lớn bị hỏng trên đường từ nguồn đến đích do một bit bị đảo ngược trong quá trình truyền. Do đó, việc gửi nhiều gói tin (packet) nhỏ hơn là hiệu quả và đáng tin cậy hơn.

TCP

Vì quan trọng là các client và server trên mạng có thể hiểu cùng một ngôn ngữ, có một số giao thức (protocol) quy định việc giao tiếp giữa các thiết bị trên Internet. Giao thức chủ yếu được sử dụng cho giao tiếp giữa ứng dụng web và trình duyệt được gọi là Transmission Control Protocol (TCP). TCP là một giao thức tầng vận chuyển đảm nhận trách nhiệm truyền dữ liệu và đảm bảo việc truyền dữ liệu đáng tin cậy giữa các client và server trên web. TCP thực hiện điều này bằng cách thêm thông tin bổ sung vào các gói tin dữ liệu cho phép xác thực gói tin và cho phép trao đổi thông điệp xác nhận giữa client và server để xác nhận việc truyền dữ liệu.

Giao thức TCP bắt đầu bằng một bước giao tiếp 3 chiều (3-way handshake). Bước giao tiếp này cho phép cả hai đầu (server và client) khởi tạo và duy trì nhiều kết nối TCP cùng một lúc. Hãy xem sơ đồ đơn giản của bước giao tiếp 3 chiều của TCP dưới đây.



HTTP & HTTPS

Chúng ta vừa mới tìm hiểu rằng client và server giao tiếp với nhau bằng cách khởi tạo các kết nối TCP và sau đó gửi các tin nhắn cho nhau. Bây giờ, chúng ta sẽ xem xét cách chính xác các tin nhắn này được cấu trúc như thế nào.

HyperText Transfer Protocol, thường được gọi là HTTP, là một giao thức tầng ứng dụng (application layer protocol) quy định cách các tin nhắn mà một client và server trao đổi trên web được cấu trúc và trao đổi như thế nào. Chương trình client và chương trình server trò chuyện với nhau bằng cách trao đổi các tin nhắn HTTP, và lợi ích của HTTP là đảm bảo các tin nhắn được gửi một cách toàn vẹn và đúng thời gian.

Điều này có thể có vẻ mơ hồ, nhưng ý tưởng cấp cao là HTTP được xây dựng trên nền tảng TCP và tạo ra một server HTTP cho ứng dụng web của bạn thực chất chỉ đơn giản là tạo ra một server mà các client tạo kết nối TCP nhằm truyền dữ liệu đáng tin cậy. Nói một cách đơn giản, HTTP là phương tiện để đảm bảo ứng dụng của bạn sử dụng TCP để truyền tin nhắn từ client tới server và ngược lại. Vì vậy, khi bạn nhập một URL trong trình duyệt của bạn, thực sự điều gì xảy ra là một lệnh HTTP được gửi đến server lưu trữ ứng dụng để lấy và truyền trang web được yêu cầu thông qua TCP.

Cấu trúc thông thường của URL là https://www.google.com.vn/?hl=vi/. Vậy, HTTPS viết tắt của cái gì? HTTPS là viết tắt của HyperText Transfer Protocol Secure, và nó đơn giản chỉ là phiên bản bảo mật của HTTP. Điều này có nghĩa là việc truyền thông giữa trình duyệt và máy chủ (hosting server) lưu trữ được mã hóa để không có bên thứ ba nào trên mạng có thể truy cập thông tin không được chia sẻ.

Ports

Cho đến nay, chúng ta đã thảo luận về cả giao thức tầng vận chuyển lẫn giao thức tầng ứng dụng đảm bảo việc giao tiếp hiệu quả giữa các end-system trên web. Nhưng, chính xác là tin nhắn mà các giao thức này cho phép end-system di chuyển tới đâu? Cổng (port) là điểm cuối của giao tiếp giữa client và server. Nghĩa là, các tin nhắn từ mạng đến trên một end-system thông qua port. Chúng ta đã tóm tắt về sockets trước đây và nói về cách chúng là cổng thông tin cho các quy trình (gateway to process); sockets được mở trên port để cho phép quy trình gửi và nhận tin nhắn. Cổng được định danh bằng các số, và tất cả các cổng dưới 1024 mặc định được liên kết với một giao thức cụ thể. Ví dụ, port number cho HTTP là 80, điều này có nghĩa là bất kỳ tin nhắn nào bạn gửi hoặc nhận trên web sẽ vào và ra khỏi máy tính của bạn thông qua một socket tại port 80. Các port trên 1024 là các cổng mở sẵn (open port available) để các lập trình viên sử dụng cho bất kỳ quy trình nào họ muốn giao tiếp với mạng. Họ có thể xây dựng sockets trên các cổng

này và xác định cấu trúc và loại tin nhắn mà socket này có thể đáp ứng thông qua việc lập trình socket. Lập trình socket là một khía cạnh của Mạng máy tính vượt quá phạm vi thảo luận này, nhưng đó là một kỹ năng rất hữu ích nếu bạn quan tâm đến lĩnh vực này và nên tìm hiểu thêm.

3. How Data Finds its Way?

Internet protocol & IP address

Internet Protocol, hay gọi là IP, là một giao thức tầng mạng (network layer protocol) có trách nhiệm gán địa chỉ cho các thiết bị để cung cấp cho chúng các định danh duy nhất giúp chúng có thể được kết nối và tìm thấy. Mỗi thiết bị trên Internet có một địa chỉ IP duy nhất mà các thiết bị khác sử dụng để kết nối với nó.

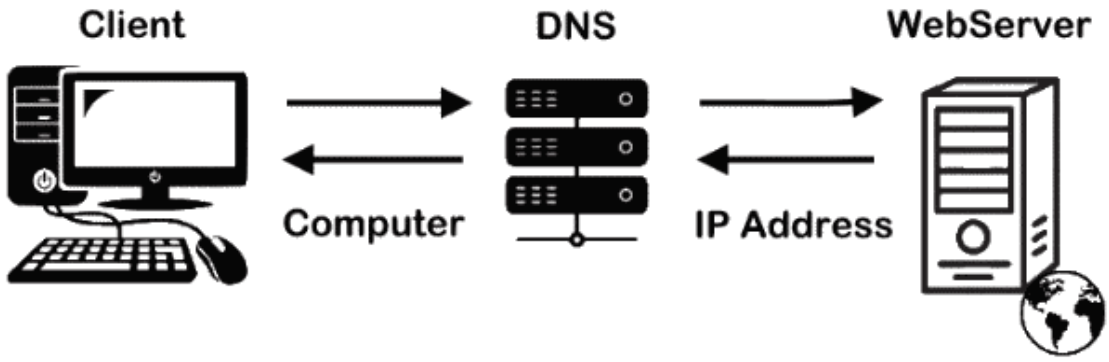
Điều này có nghĩa là khi bạn có trang web của mình đang hoạt động trên một server, người dùng sẽ có thể kết nối với nó thông qua địa chỉ IP của server và nhận dữ liệu. Bạn có thể tự hỏi bây giờ địa chỉ IP từ đâu nếu bạn truy cập vào các trang web qua URL. Đó là do DNS. Các router trên web không lưu trữ URL cho mỗi trang web được lưu trữ trên Internet; nó chỉ biết địa chỉ IP và chuyển tiếp các gói tin dựa trên IP. Tuy nhiên, người dùng chỉ biết URL hoặc tên miền (domain name) của các trang web mà họ đang truy cập. Để giải quyết điều này, Internet sử dụng các DNS servers, đóng vai trò như các trung gian dịch thuật giữa end-system và các router để chuyển tiếp các gói tin dữ liệu (data packet) đến đích của chúng. Quá trình tra cứu tên miền trên máy chủ DNS và nhận địa chỉ IP tương ứng được gọi là quá trình giải quyết DNS (DNS resolution).

DNS lookups

Điều cuối cùng cần được trình bày là cách tra cứu trong máy chủ DNS hoạt động. Hãy xem xét các bước trong quá trình tương tác giữa bất kỳ end-system nào trên web, trong ngữ cảnh này được gọi là client, và một máy chủ DNS:

1. Client muốn truy cập một trang web thông qua tên miền của nó (ví dụ: www.example.com), nhưng cần lấy địa chỉ IP tương ứng.
2. Client gửi một yêu cầu tra cứu DNS đến máy chủ DNS gần nhất trong hệ thống. Yêu cầu này chứa tên miền cần tra cứu.
3. Máy chủ DNS nhận yêu cầu và kiểm tra trong cơ sở dữ liệu của nó xem có thông tin tương ứng cho tên miền đó hay không. Nếu có, máy chủ DNS sẽ trả về địa chỉ IP tương ứng cho client.
4. Trong trường hợp máy chủ DNS không có thông tin tương ứng trong cơ sở dữ liệu của mình, nó sẽ tiếp tục gửi yêu cầu tra cứu tới máy chủ DNS khác, tiếp tục quá trình này cho đến khi tìm được địa chỉ IP tương ứng hoặc đạt tới máy chủ DNS cuối cùng mà không có thông tin phù hợp.
5. Khi máy chủ DNS tìm thấy địa chỉ IP tương ứng, nó sẽ trả về cho client.
6. Client nhận được địa chỉ IP và sử dụng nó để thiết lập kết nối đến trang web tương ứng thông qua giao thức HTTP hoặc HTTPS.

Quá trình này diễn ra trong vòng chưa đến một giây, và các máy chủ DNS trên toàn thế giới làm việc cùng nhau để đảm bảo việc tra cứu nhanh chóng và hiệu quả.



4. Life of a Packet

Life of a Packet đề cập đến hành trình và quy trình mà một gói tin dữ liệu trải qua khi di chuyển qua mạng. Dưới đây là tổng quan chung về cuộc sống của một gói tin:

1. Tạo gói tin: Gói tin được tạo ra tại thiết bị nguồn, chẳng hạn như một client hoặc server, khi dữ liệu cần được truyền. Gói tin bao gồm dữ liệu và thông tin tiêu đề cần thiết.
2. Đóng gói: Gói tin được đóng gói trong các tiêu đề giao thức khác nhau khi di chuyển xuống ngăn xếp mạng. Mỗi tầng thêm tiêu đề riêng của nó, bao gồm tiêu đề tầng liên kết dữ liệu, tầng mạng và tầng vận chuyển.
3. Định tuyến: Gói tin được chuyển tiếp từ một thiết bị mạng sang thiết bị mạng khác dựa trên các mục bảng định tuyến và địa chỉ IP đích. Các router xem xét thông tin tiêu đề của gói tin để xác định điểm tiếp theo trên đường đi.
4. Truyền: Gói tin được truyền qua phương tiện vật lý, chẳng hạn như dây đồng, cáp quang hoặc kết nối không dây. Tầng vật lý xử lý việc truyền thực tế của gói tin dưới dạng tín hiệu điện hoặc quang.
5. Đến nơi và giải gói: Gói tin đến thiết bị mạng đích, nơi các tiêu đề đóng gói được loại bỏ tuần tự tại mỗi tầng của ngăn xếp mạng. Dữ liệu được trích xuất và gửi đến các giao thức tầng cao hơn phù hợp.
6. Giao nhận: Dữ liệu trong gói tin được gửi đến ứng dụng hoặc quy trình mục tiêu trên thiết bị đích. Các giao thức tầng cao hơn, chẳng hạn như TCP hoặc UDP, tiếp tục xử lý và chuyển giao dữ liệu đến cổng hoặc socket chính xác.
7. Xác nhận: Trong các giao thức như TCP, thiết bị nhận gửi lại một xác nhận đến thiết bị gửi để xác nhận việc nhận thành công của gói tin. Điều này đảm bảo việc chuyển giao đáng tin cậy và cho phép gửi lại trong trường hợp mất gói tin hoặc lỗi.
8. Loại bỏ: Sau khi dữ liệu đã được xử lý và sử dụng bởi thiết bị đích, gói tin được loại bỏ. Các tài nguyên mạng trước đây được cấp cho gói tin được giải phóng để sử dụng cho các truyền dữ liệu khác.

Cần lưu ý rằng các bước này có thể thay đổi dựa trên kiến trúc mạng, giao thức và thiết bị liên quan. Tuy nhiên, khung tóm tắt chung này cung cấp một hiểu biết về hành trình và vòng đời thông thường của một gói tin trong một mạng.

Databases

1. Databases in Web Development

Static vs dynamic web pages

Trang web tĩnh chỉ hiển thị thông tin như văn bản hoặc hình ảnh trên trang web mà người dùng không thể tương tác.

Trái lại, trang web động cho phép tương tác của người dùng, và đó là lúc cơ sở dữ liệu xuất hiện. Hãy nghĩ về bất kỳ trang web nào bạn đã truy cập vào hôm nay. Bao nhiêu trong số đó yêu cầu bạn đăng nhập hoặc cho phép bạn nhấp vào một nút để truy xuất thông tin nào đó, thậm chí có thể là một trang web mới? Thông tin mà các ứng dụng web có thể hiển thị hoặc xử lý để cho phép bạn truy cập thông tin khác cần được lưu trữ ở một nơi nào đó. Nơi đó chính là cơ sở dữ liệu liên kết với ứng dụng web.

DBMS in web applications

DBMS, viết tắt của Hệ thống Quản lý Cơ sở dữ liệu (Database Management System), là một sự bổ sung quan trọng trong phát triển web, cho phép dữ liệu được tách biệt khỏi logic của ứng dụng và được lưu trữ riêng biệt để được truy xuất và xử lý khi cần thiết. Điều này có nghĩa là dữ liệu được lưu trữ trong một cơ sở dữ liệu có hệ thống hoàn toàn tự động quản lý. Ứng dụng sau đó gọi cơ sở dữ liệu này để lấy bất kỳ dữ liệu cần thiết nào mà không cần tích hợp một lượng lớn mã xử lý trong mã ứng dụng chính. Hệ thống quản lý cơ sở dữ liệu rất quan trọng đối với các nhà phát triển vì nó cung cấp một phương pháp hiệu quả cao để xử lý nhiều loại dữ liệu mà không làm gây xáo trộn mã ứng dụng.

Types of DBMS

Có 2 loại: SQL và NoSQL

SQL

Các cơ sở dữ liệu SQL, còn được gọi là cơ sở dữ liệu quan hệ, sử dụng ngôn ngữ truy vấn cấu trúc (SQL) để định nghĩa và thao tác dữ liệu. Một mặt, điều này rất hữu ích vì SQL là một trong những lựa chọn linh hoạt và phổ biến nhất, làm cho nó trở thành một lựa chọn an toàn và đặc biệt tuyệt vời cho các truy vấn phức tạp. Tuy nhiên, nó cũng có thể gây hạn chế, vì SQL yêu cầu bạn sử dụng các schema được định trước để xác định cấu trúc dữ liệu trước khi làm việc với nó. Hơn nữa, dữ liệu luôn được lưu trữ dưới dạng bảng trong cơ sở dữ liệu SQL, dẫn đến việc truy xuất dữ liệu không hiệu quả và phức tạp. Ngoài ra, tất cả dữ liệu của bạn phải tuân thủ cùng một cấu trúc. Điều này có thể đòi hỏi nỗ lực đáng kể và có thể dẫn đến mã xử lý dữ liệu phức tạp, ảnh hưởng đến chất lượng của ứng dụng tổng thể.

NoSQL

Các cơ sở dữ liệu NoSQL, còn được gọi là cơ sở dữ liệu phi quan hệ, có một schema linh hoạt cho dữ liệu không có cấu trúc và dữ liệu được lưu trữ theo nhiều cách, bao gồm cặp khóa-giá trị, tài liệu và thậm chí đồ thị. Điều linh hoạt này có nghĩa là bạn không cần phải định nghĩa cấu trúc dữ liệu một cách rõ ràng và mỗi tập dữ liệu có thể có cấu trúc riêng biệt mà không cần phải lo lắng về việc viết mã không cần thiết để xử lý dữ liệu này để phù hợp với một cấu trúc đã định sẵn cụ thể. Các cơ sở dữ liệu này có thể phát triển một cách linh hoạt và cấu trúc dữ liệu có thể thay đổi từ cơ sở dữ liệu này sang cơ sở dữ liệu khác.

2. SQL Databases

MySQL

MySQL là một trong những hệ thống quản lý cơ sở dữ liệu phổ biến nhất, và nó đã trở thành một phần không thể thiếu của hầu hết các ứng dụng web trong quá khứ. MySQL được hỗ trợ bởi Oracle và sử dụng Ngôn ngữ Truy vấn Chuẩn (SQL) để lưu trữ dữ liệu dưới dạng bảng và để truy xuất dữ liệu cần tạo các truy vấn bằng SQL. MySQL, khác với SQL chính nó, hoạt động trên tất cả các nền tảng, bao gồm Linux, iOS và Windows và nó có đơn giản như SQL vì nó không đòi hỏi phải học cú pháp mới nếu bạn đã biết SQL. Hãy xem một ví dụ về truy vấn MySQL tạo một bảng mới:

```
CREATE TABLE [IF NOT EXISTS] table_name(
   column1 DATATYPE,
   column2 DATATYPE,
   column3 DATATYPE,
   column4 DATATYPE,
   PRIMARY KEY (column1)
) ENGINE=storage_engine
```

Như đã rõ từ ví dụ trên, MySQL sử dụng các truy vấn SQL cơ bản để tạo bảng và xử lý dữ liệu. Điều này là một trong những ưu điểm chính của MySQL vì SQL không khó để học và được sử dụng rộng rãi, điều này có nghĩa là MySQL tương đối dễ sử dụng đối với hầu hết các người mới bắt đầu đã biết SQL.

PostgreSQL

PostgreSQL, còn được gọi là Postgres, là một hệ thống quản lý cơ sở dữ liệu đối tượng-quan hệ mã nguồn mở có thể xử lý lượng lớn dữ liệu và cung cấp hỗ trợ liền mạch cho các ứng dụng web xử lý nhiều người dùng đồng thời. Nó là cơ sở dữ liệu mặc định cho macOS và cung cấp quản lý dữ liệu hiệu quả trên các nền tảng khác nhau. Hơn nữa, nó cũng hỗ trợ truy cập cơ sở dữ liệu đồng thời và cho phép người dùng thêm các chức năng tùy chỉnh được phát triển bằng nhiều ngôn ngữ lập trình, bao gồm C, C++ và Java. Ngoài ra, lợi ích chính khác của PostgreSQL là nó được thiết kế để có thể mở rộng. Điều này có nghĩa là người dùng có thể định nghĩa các loại dữ liệu riêng của họ và thậm chí cài đặt các plugin tùy chỉnh để thay đổi các phần của hệ thống theo yêu cầu của họ.

MariaDB

MariaDB là một phiên bản phát triển độc lập của MySQL, được tạo ra bởi các nhà phát triển gốc của MySQL sau khi MySQL bị mua lại. Nó được thiết kế để thay thế MySQL và bao gồm một loạt các bộ đệm lưu trữ để dễ dàng làm việc với dữ liệu từ các hệ thống quản lý dữ liệu quan hệ khác. MariaDB, giống như MySQL, sử dụng ngôn ngữ truy vấn tiêu chuẩn, điều này khiến nó không khác gì MySQL trong việc sử dụng. Hơn nữa, MariaDB chạy trên nhiều hệ điều hành và hỗ trợ nhiều ngôn ngữ lập trình khác nhau. Đặc biệt, nó hỗ trợ PHP, một trong những ngôn ngữ phát triển web phổ biến nhất.
Ngoài những lợi ích mà MySQL mang lại, MariaDB còn cung cấp nhiều hoạt động và lệnh không có sẵn trong MySQL và thay thế các tính năng có xu hướng ảnh hưởng đến hiệu suất một cách tiêu cực.

The transition to NoSQL

Tốc độ là một trong những yếu tố quan trọng nhất ảnh hưởng đến trải nghiệm người dùng trên ứng dụng web. Phụ thuộc vào một số lượng không cần thiết của các cuộc gọi tới cơ sở dữ liệu thường dẫn đến làm chậm các hoạt động cho người dùng, và vì vậy, việc quan trọng đối với các nhà phát triển web là lưu trữ một số phần dữ liệu được yêu cầu thường xuyên trong bộ nhớ để có thể truy xuất nhanh chóng, mà không gây ra độ trễ lớn trong việc hiển thị dữ liệu. Để làm điều này, các nhà phát triển web cần nghĩ về cách lấy được nhiều dữ liệu trong bộ nhớ càng hợp lý càng tốt, sau đó là cách lưu trữ các phần dữ liệu trong bộ nhớ đệm ở cấp độ hệ thống tệp để tránh việc tạo ra các cuộc gọi tới cơ sở dữ liệu hoàn toàn. Lý do cho điều này là việc truy xuất dữ liệu từ cơ sở dữ liệu là mắc cảnh trở ngại đối với hầu hết các ứng dụng web, và giảm số lượng các thao tác này có thể cải thiện đáng kể sự phản hồi của ứng dụng. Quá trình lựa chọn các thành phần dữ liệu được yêu cầu thường xuyên và lưu trữ chúng trong bộ nhớ để tránh các cuộc gọi tới cơ sở dữ liệu không cần thiết được gọi là việc đặt vào bộ nhớ đệm (caching), và các cơ sở dữ liệu

NoSQL đã trở nên phổ biến trong thời gian gần đây nhờ khả năng tự động đặt dữ liệu vào bộ nhớ đệm để cải thiện hiệu suất.
Ngoài ra, các cuộc gọi tới cơ sở dữ liệu chính mình cũng không hiệu quả hơn trong cơ sở dữ liệu SQL so với cơ sở dữ liệu NoSQL. Điều này là do cơ sở dữ liệu SQL lưu trữ dữ liệu dưới dạng bảng, và các truy vấn yêu cầu các lần lặp chi tiết của bảng. Trong khi đó, cơ sở dữ liệu NoSQL sử dụng cấu trúc dữ liệu để lưu trữ các mục, điều này có nghĩa rằng việc tra cứu luôn được tối ưu hóa và các cuộc gọi tới cơ sở dữ liệu hiệu quả hơn nhiều. Do đó, với các ứng dụng web hiện đại ngày càng yêu cầu thời gian phản hồi nhanh để đạt được chức năng dự định, sự chuyển đổi sang sử dụng cơ sở dữ liệu NoSQL đã rõ ràng trong thời gian gần đây.

3. NoSQL Databases

MongoDB

MongoDB, which is perhaps the most popular NoSQL DBMS, is an open-source non-relational database management system that has come to be known as the leading option when it comes to developing modern web applications. The reason for this is that MongoDB uses a document-based storage system that stores key-value pairs and allows for highly efficient lookups, making data retrieval much faster and easier than any typical SQL DBMS. To add to this, the document model ensures that data can be mapped directly to objects within the application code, and is, therefore, makes data handling significantly easy by eliminating the need for adding code to process queried data. In addition to this, MongoDB stores data in a highly flexible manner, thus allowing for fields to vary from document to document and the structure of data to be open to change over time. Data can also be indexed and queried according to specific user requirements and then updated in real-time. This makes MongoDB an exceptionally powerful system for data analysis, and that shows in its popularity. However, the most important aspect of quality that MongoDB covers is reliability; MongoDB is a distributed database at its core which means that it is available, scalable, and easily distributable across locations, thus making it well equipped for modern applications that require quick access to data at all times.

Apache CouchDB

CouchDB là một hệ thống quản lý cơ sở dữ liệu NoSQL mã nguồn mở, với mục tiêu chính là cung cấp sự dễ sử dụng. CouchDB kết hợp một mô hình lưu trữ tài liệu trực quan với một trình truy vấn mạnh mẽ, cho phép người dùng lưu trữ dữ liệu của họ an toàn trên máy chủ cá nhân hoặc với bất kỳ nhà cung cấp đám mây hàng đầu nào. Ngoài ra, CouchDB không chỉ có khả năng lưu trữ mọi loại dữ liệu, mà còn cho phép ứng dụng web trích xuất dữ liệu này một cách thuận tiện mà không cần thêm các lệnh dịch ngôn ngữ, vì CouchDB hỗ trợ định dạng mà các ứng dụng web thông thường sử dụng.

Redis

Redis là một cơ sở dữ liệu cấu trúc dữ liệu lưu trữ trong bộ nhớ (in-memory) mã nguồn mở thường được sử dụng như một cơ sở dữ liệu. Nó hỗ trợ tất cả các loại cấu trúc dữ liệu từ chuỗi, băm (hashes), danh sách (lists) và tập hợp (sets) đến các tập hợp đã sắp xếp (sorted sets) với truy vấn phạm vi, bitmaps, hyperloglogs và chỉ mục địa lý với truy vấn bán kính và streams. Điều này có nghĩa là Redis cung cấp một loạt các cấu trúc dữ liệu có thể được sử dụng để lưu trữ dữ liệu ứng dụng của bạn một cách tối ưu nhất dựa trên cách dữ liệu được cấu trúc. Ngoài ra, do Redis lưu trữ dữ liệu trong bộ nhớ, nó cho phép truy xuất dữ liệu nhanh chóng và do đó, đáng kể tăng tốc quá trình phản hồi các yêu cầu từ người dùng.

4. Web Caching

Introduction to web caching

Caching trên web là một tính năng thiết kế của giao thức HTTP nhằm giảm thiểu lưu lượng mạng mà một ứng dụng xử lý tại bất kỳ thời điểm nào nhằm cải thiện tính phản hồi của ứng dụng web, như được người dùng cảm nhận, tổng thể. Để làm điều này, các bộ nhớ đệm được sử dụng ở mỗi cấp độ, bắt đầu từ máy chủ chính nó cho đến trình duyệt của người dùng, và mỗi cấp độ này được thiết kế để lưu trữ dữ liệu mà người dùng rất có thể yêu cầu.

Cơ bản, caching trên web hoạt động bằng cách lưu trữ các phản hồi HTTP cho các yêu cầu cụ thể dựa trên phân tích xác suất của các yêu cầu thường xuyên được quan sát trên máy chủ. Các yêu cầu tiếp theo cho nội dung được lưu trữ trong bộ nhớ đệm có thể được đáp ứng từ một bộ nhớ đệm gần người dùng thay vì phải gửi yêu cầu ngược trở lại máy chủ web, sau đó máy chủ thực hiện cuộc gọi tới cơ sở dữ liệu để lấy dữ liệu cần thiết.

Benefits of caching

Có nhiều lợi ích của việc lưu trữ đệm trên web, mỗi lợi ích được liệt kê dưới đây:

1. Dữ liệu có thể được lưu trữ đệm ở nhiều điểm khác nhau trên đường đi giữa máy khách và máy chủ. Khi dữ liệu cần thiết được lưu trữ đệm gần máy khách, các yêu cầu không tăng lưu lượng mạng quá nhiều vì chúng được giải quyết sớm hơn trên đường đi.
2. Mở rộng từ điểm đầu tiên, vì yêu cầu được giải quyết sớm hơn trên đường đi, các phản hồi cũng được gửi trở lại nhanh hơn, do đó cải thiện tính phản hồi của ứng dụng web.
3. Lưu trữ đệm quyết liệt trên mạng cũng cho phép ứng dụng chịu được tải cao hơn về dữ liệu vì một phần đáng kể của dữ liệu có thể được lưu trữ trong các bộ nhớ đệm.
4. Trong trường hợp máy chủ gặp vấn đề khi truy cập vào cơ sở dữ liệu vì một số lý do nào đó, dữ liệu đã được lưu trữ trong các bộ nhớ đệm vẫn có thể được phục vụ cho người dùng cuối.

MongoDB

MongoDB có các cơ chế tích hợp để xử lý việc lưu trữ đệm và giữ dữ liệu được sử dụng gần đây nhất trong bộ nhớ RAM. Nếu người dùng đã tạo chỉ mục (index) cho các truy vấn của họ và tập dữ liệu hoạt động vừa với bộ nhớ RAM, MongoDB sẽ phục vụ tất cả các truy vấn từ bộ nhớ. Tuy nhiên, MongoDB không lưu trữ kết quả truy vấn để trả về kết quả từ bộ nhớ đệm cho tất cả các truy vấn trùng lặp trong tương lai.

Redis

Redis, như đã học, mặc định lưu trữ tất cả dữ liệu trong bộ nhớ và thường được sử dụng như một hệ thống đệm. Vì vậy, nó là sự lựa chọn tối ưu để sử dụng như một hệ thống cơ sở dữ liệu trong các ứng dụng web hiện đại.

CouchDB

CouchDB, giống như hai cơ sở dữ liệu NoSQL khác chúng ta đã thảo luận, cố gắng lưu trữ đệm tất cả dữ liệu có thể. Kích thước tệp nhỏ hơn, càng nhiều phần tệp có thể được lưu trữ đệm bởi CouchDB. Do đó, ý tưởng tốt là nghĩ về dữ liệu bạn muốn lưu trữ khi sử dụng các cơ sở dữ liệu NoSQL để ứng dụng web của bạn có thể chạy mượt mà và không gánh nặng bởi lượng dữ liệu không cần thiết.

Memcached

Memcached là một hệ thống đệm đối tượng trong bộ nhớ phân tán, hiệu năng cao và mã nguồn mở, được thiết kế nhằm tăng tốc ứng dụng web động bằng cách giảm tải cho cơ sở dữ liệu. Memcached lưu trữ các cặp khóa-giá trị từ các cuộc gọi cơ sở dữ liệu trong bộ nhớ để tăng tốc quá trình tra cứu cơ sở dữ liệu. Ý

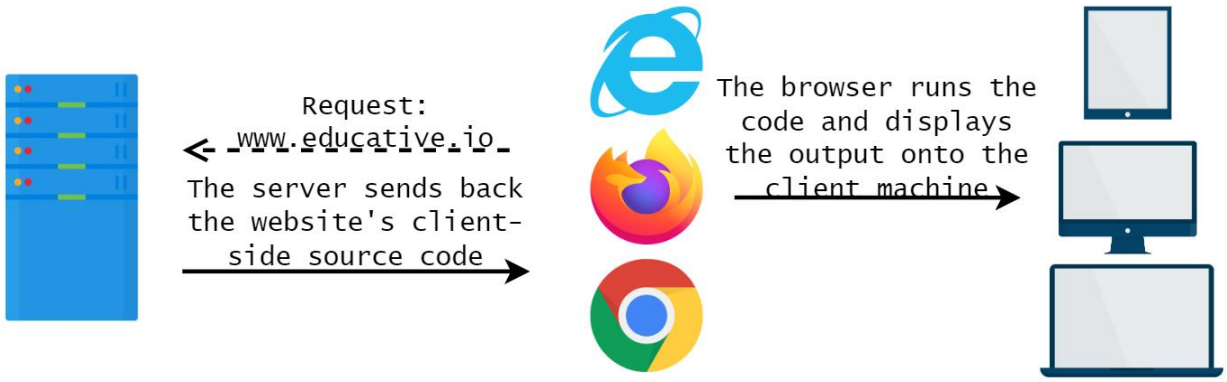
tưởng cốt lõi của Memcached là cho phép bạn sử dụng bộ nhớ từ các phần của hệ thống nơi bạn có nhiều hơn là cần thiết và làm cho nó truy cập được cho các vùng bạn có ít hơn là cần thiết. Do đó, Memcached là một công cụ rất hữu ích.

The Front End

1. The Server-side and The Client-side

Client-side

Mã nguồn của một trang web có thể được chia thành hai phần: phía máy khách (client-side) và phía máy chủ (server-side). Mã trên phía "máy khách" chạy trên trình duyệt của máy tính người dùng và xử lý các yếu tố về giao diện của trang web, cách trang web yêu cầu dữ liệu từ máy chủ và cách nó tương tác với bộ nhớ tạm thời và bộ nhớ cục bộ trên máy tính người dùng. Điều này bao gồm nhưng không giới hạn ở việc lựa chọn và thiết kế các thành phần giao diện người dùng, tạo bố cục, điều hướng, xác thực biểu mẫu và xử lý bộ nhớ đệm. Ngôn ngữ phía máy khách bao gồm HTML, CSS và JavaScript.



Server-side

Server-side programming được sử dụng để cung cấp thông tin được truy vấn từ các trang web cho các máy khách.

Các công ty như Amazon sử dụng lập trình phía máy chủ để xây dựng kết quả tìm kiếm cho các sản phẩm và đưa ra gợi ý sản phẩm được đích danh dựa trên sở thích và thói quen mua hàng trước đó của khách hàng. Ngân hàng sử dụng lập trình phía máy chủ để lưu trữ thông tin tài khoản và hạn chế truy cập từ các tài khoản không được ủy quyền. Các trang mạng xã hội như Facebook, Twitter và Instagram sử dụng lập trình phía máy chủ để tạo nên, chia sẻ và kiểm soát truy cập đến nội dung phù hợp với từng người dùng. Lập trình phía máy chủ bao gồm nhưng không giới hạn ở:

1. Cung cấp và lưu trữ thông tin một cách hiệu quả
2. Cung cấp trải nghiệm tùy chỉnh cho từng người dùng
3. Kiểm soát truy cập vào nội dung
4. Lưu trữ thông tin phiên/ trạng thái
5. Thông báo và giao tiếp
6. Phân tích dữ liệu

Các ngôn ngữ phía máy chủ bao gồm JavaScript, Python, PHP, Ruby và C#.

The Backend

1. What is the Back End?

What are web servers?

Một máy chủ web là hệ thống cung cấp nội dung và dịch vụ cho người dùng qua Internet. Máy chủ web hoàn toàn giống như các máy tính thông thường, chỉ khác là nó mạnh mẽ và mạnh mẽ hơn. Hơn nữa, hầu hết các máy chủ không có màn hình giống như màn hình máy tính cá nhân thông thường, nhưng bạn có thể kết nối với chúng bằng SSH. Để đơn giản, SSH là một cách để lấy một máy khác's terminal trên máy bạn. Một máy chủ web có thể 'lưu trữ' một trang web, tức là máy chủ web sẽ có toàn bộ mã nguồn cho trang web trên đó: phía trước và phía sau, và bất kỳ ai có kết nối Internet đều có thể truy cập vào nó bằng máy tính của họ để xem trang web.

Về mặt kỹ thuật, bạn có thể mở cổng 80 trên máy tính cá nhân của bạn và lưu trữ trang web của mình ở đó. Tuy nhiên, điều này thường không được thực hiện vì máy tính cá nhân thông thường sẽ không thể xử lý được tải của hàng ngàn kết nối khách hàng có thể xảy ra, điều này sẽ đặt dữ liệu cá nhân của bạn vào nguy cơ và bạn sẽ phải giữ máy tính hoạt động liên tục với máy chủ hoạt động 24/7. Vì vậy, dịch vụ lưu trữ web giải quyết vấn đề này.

Web hosting services

Các công ty lưu trữ web cung cấp không gian trên một máy chủ thuê hoặc sở hữu để sử dụng cho khách hàng, cùng với việc cung cấp kết nối Internet, thường là trong một trung tâm dữ liệu. Họ thường thu phí hàng tháng cho dịch vụ này. Một trung tâm dữ liệu là một tòa nhà chứa nhiều máy chủ. Trung tâm dữ liệu thường yêu cầu hệ thống làm mát và bảo trì đầy đủ để giữ nhiệt độ máy chúng ở mức thấp, điều này giảm tỷ lệ sự cố và tăng lợi nhuận. Ví dụ, Facebook sở hữu một số trung tâm dữ liệu lớn nhất trên thế giới!

Web server operating systems

Tương tự như máy tính cá nhân của bạn có một hệ điều hành như Windows, Mac OS hoặc Linux, máy chủ cũng yêu cầu một hệ điều hành để hoạt động. Tuy nhiên, hầu hết các hệ điều hành máy chủ được chuyên biệt hóa cho máy chủ và khác một chút so với hệ điều hành máy tính để bàn.

Linux

Linux không phải là một hệ điều hành duy nhất. Đó là một nhóm các hệ điều hành mã nguồn mở và miễn phí. Thông thường, Linux được đóng gói dưới dạng một phiên bản gọi là bản phân phối Linux (hoặc distro ngắn gọn) để sử dụng trên cả máy tính để bàn và máy chủ. Có nhiều phiên bản hoặc bản phân phối của Linux dành cho việc sử dụng máy chủ. Ví dụ:

CentOS

Lợi ích của việc sử dụng CentOS là nó ổn định và hiếm khi gặp sự cố. Nó cũng là bản phân phối phổ biến nhất cho máy chủ đến thời điểm hiện tại, do đó cũng có lợi ích từ sự hỗ trợ tuyệt vời từ cộng đồng web, điều này có nghĩa là bạn sẽ không gặp khó khăn trong việc giải quyết bất kỳ vấn đề nào.

Hơn nữa, hầu hết các bản phân phối Linux đều được cập nhật thường xuyên, nhưng CentOS không. Lợi ích của việc này là một khi bạn đã cài đặt và chạy máy chủ của bạn, nó sẽ hoạt động tốt trong một thời gian dài mà không cần nâng cấp thường xuyên lên phiên bản mới hơn.

Đối với nhược điểm, bạn sẽ gặp khó khăn khi chạy các gói phần mềm mới hơn và sử dụng các công nghệ mới nhất. Nếu bạn muốn sử dụng công nghệ mới nhất, CentOS có thể không phù hợp với bạn.

Debian

Giống CentOS, Debian cũng không cập nhật thường xuyên. Debian cũng là một bản phân phối cộng đồng, điều này có nghĩa là các phiên bản mới của Debian chỉ được phát hành khi cộng đồng đạt được sự nhất trí. Vì vậy, mỗi phiên bản Debian được thử nghiệm kỹ lưỡng và đáng tin cậy. Chỉ có một loại Debian duy nhất để bạn tải xuống và cài đặt, nó hoạt động trên cả máy chủ và máy tính để bàn.

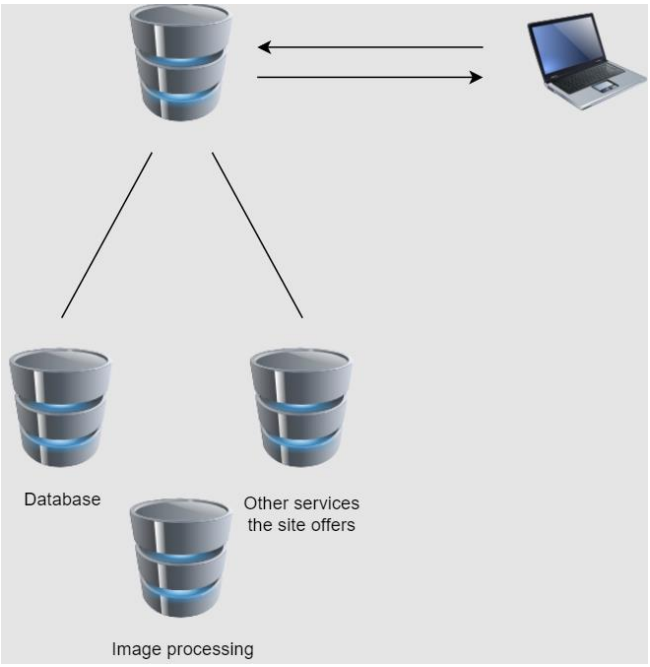
Ubuntu

Ubuntu là bản phân phối Linux phổ biến nhất cho máy tính để bàn. Nếu bạn quen thuộc với Ubuntu, bạn có thể sử dụng Ubuntu trên máy chủ để có lợi thế là quen thuộc với giao diện và quy trình làm việc của nó!

2. Microservice Architecture

Microservices

Tuy nhiên, hầu hết các trang web hiện đại không phụ thuộc vào kiểu lưu trữ web như vậy, tức là mã của tất cả các chức năng của toàn bộ trang web không tồn tại trên một máy chủ duy nhất. Thay vào đó, các trang web được lưu trữ trên các máy ảo (hãy tưởng tượng chúng như là máy tính trong máy tính). Các máy ảo cho phép chúng ta có nhiều 'máy' trên một máy chủ duy nhất, mỗi máy có hệ điều hành khác nhau. Vì vậy, trang web được lưu trữ trên một máy chủ trong một máy chủ và máy chủ đó giao nhiệm vụ cho các máy chủ khác.



Trong kiến trúc Microservices, nhiều máy ảo tồn tại trên một máy chủ vật lý, và mỗi máy ảo có một công việc riêng. Ví dụ, một trong số chúng có thể là 'giao diện' mà người dùng tương tác, và nó có thể kết nối với một máy khác để truy cập vào cơ sở dữ liệu.

Theo trang web Microservices,

"Microservices - còn được gọi là kiến trúc microservice - là một kiến trúc sắp xếp ứng dụng như một tập hợp các dịch vụ có các đặc điểm sau:

- Dễ bảo trì và kiểm thử
- Kết nối lỏng lẻo
- Có thể triển khai độc lập
- Tổ chức xung quanh các khả năng kinh doanh.

Kiến trúc microservice cho phép việc triển khai/cung cấp liên tục các ứng dụng lớn, phức tạp. Nó cũng cho phép một tổ chức tiến hóa công nghệ của mình."

Back-end programming

What do backend engineers do?

Các kỹ sư backend là những người lập trình các máy chủ để xử lý yêu cầu của người dùng và phản hồi với tài nguyên được yêu cầu một cách chính xác. Họ cũng viết mã để xử lý và lưu trữ dữ liệu người dùng.

Efficient storage and delivery of information

Dữ liệu liên quan đến một trang web phải được lưu trữ trong cơ sở dữ liệu và được cung cấp khi có yêu cầu. Hãy xem xét cơ sở dữ liệu của Amazon về các mặt hàng; nếu hệ thống truy vấn cơ sở dữ liệu của họ trở nên không hiệu quả tại bất kỳ thời điểm nào, họ sẽ mất khách hàng.

Customized user experience

Các máy chủ thường lưu trữ và sử dụng thông tin về khách hàng để cung cấp một trải nghiệm người dùng tùy chỉnh. Ví dụ, nhiều trang web lưu trữ thông tin thẻ tín dụng để không cần nhập lại thông tin đó.

Controlled access to content

Lập trình backend bao gồm việc hạn chế truy cập vào thông tin một cách phù hợp. Ví dụ, trong ứng dụng đặt xe đi chung như Uber, một người dùng không nên có thể xem lịch sử di chuyển bằng ô tô của người dùng khác.

Store session/state information

Các kỹ sư backend cũng thực hiện việc xử lý phiên người dùng, tức là một chuỗi được liên kết với mỗi người dùng truy cập trang web và dữ liệu liên quan đến chuỗi đó như địa chỉ email hoặc lịch sử đặt hàng được lưu trữ và hiển thị khi người dùng truy cập lại. Một ví dụ khác là lưu trạng thái của một trò chơi đơn giản để người dùng có thể truy cập lại trang web và tiếp tục từ nơi họ rời khỏi.

Notifications

Các máy chủ có thể được lập trình để gửi thông báo chung hoặc dành riêng cho người dùng thông qua trang web chính hoặc qua email, tin nhắn SMS, hội thoại tức thì, trò chuyện video hoặc các dịch vụ giao tiếp khác.

Một số ví dụ bao gồm:

- Facebook và Twitter gửi email và tin nhắn SMS để thông báo cho bạn về các thông tin mới.
- Amazon thường gửi email sản phẩm để gợi ý các sản phẩm tương tự với những gì bạn đã mua hoặc xem trước đó mà bạn có thể quan tâm.
- Một máy chủ web có thể gửi thông báo cảnh báo cho các quản trị viên trang web, cảnh báo về bộ nhớ thấp trên máy chủ hoặc hoạt động đáng ngờ của người dùng.

Data analysis

Một trang web có thể thu thập rất nhiều dữ liệu về người dùng: những gì họ tìm kiếm, những gì họ mua, những gì họ đề xuất, thời gian họ ở trên mỗi trang. Lập trình phía máy chủ có thể được sử dụng để tinh chỉnh các phản hồi dựa trên phân tích dữ liệu này.

Ví dụ, Amazon và Google đều quảng cáo các sản phẩm dựa trên những tìm kiếm (và mua hàng) trước đó.

Principles of Software Engineering

Software Process Models

Software process

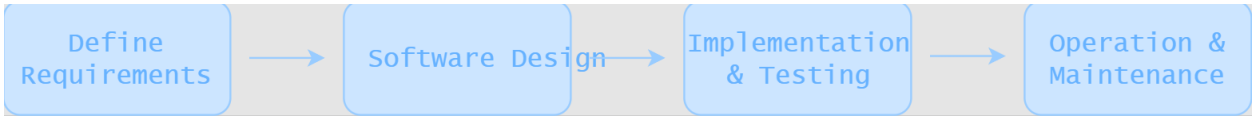
Bây giờ khi chúng ta đã có một cái nhìn cơ bản về quá trình phát triển một trang web và các thành phần khác nhau của nó, chúng ta sẽ tiếp tục thảo luận về các cách khác nhau để cấu trúc quá trình phát triển.

Một quy trình phần mềm được xác định là một tập hợp các hoạt động liên quan dẫn đến việc sản xuất một sản phẩm phần mềm, chẳng hạn như một trang web. Các quy trình phần mềm mà chúng tôi sẽ thảo luận

trong chương này đều thuộc về kỹ thuật phần mềm chuyên nghiệp, bao gồm xây dựng các ứng dụng web quy mô lớn. Tuy nhiên, đây là các khái niệm quan trọng mà bạn nên quen thuộc nếu bạn đang muốn làm web phát triển chuyên nghiệp hoặc dự định triển khai trang web của mình sau này vì chúng cung cấp một phương pháp học thuyết cho quá trình cho phép bạn tổ chức công việc của mình thành các giai đoạn logic.

Dưới đây là bốn hoạt động cơ bản trong kỹ thuật phần mềm mà tất cả các quy trình đều phải bao gồm:

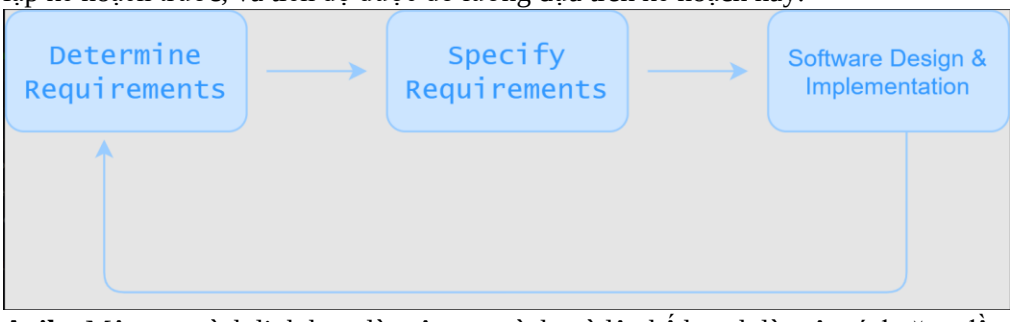
1. Xác định phần mềm: Định nghĩa chức năng dự kiến của phần mềm và các ràng buộc tiềm năng về hoạt động của nó. Trong trường hợp của một ứng dụng web cơ bản, đây là giai đoạn mà bạn xác định các tính năng mà bạn muốn triển khai trên trang web của mình.
2. Thiết kế và triển khai phần mềm: Sản xuất phần mềm dựa trên các thông số kỹ thuật đã xác định. Đây là giai đoạn mà ứng dụng thực tế được phát triển.
3. Xác nhận phần mềm: Đảm bảo phần mềm hoạt động như những gì người dùng muốn. Trong giai đoạn này, bạn đảm bảo rằng các tính năng bạn muốn triển khai đã được thực hiện đúng cách.
4. Tiến hóa phần mềm: Khả năng thích ứng với các yêu cầu người dùng đang phát triển. Đây là giai đoạn liên tục, trong đó bạn liên tục đưa ra các tính năng bạn cho rằng sẽ làm cho ứng dụng của bạn tốt hơn và triển khai chúng.



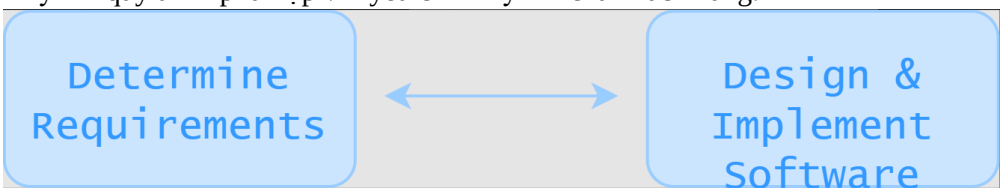
Những khái niệm trên là quan trọng trong quá trình phát triển một ứng dụng web thành công.

Phân loại

1. **Plan-driven:** Một quy trình dựa vào kế hoạch là quy trình mà tất cả các hoạt động quy trình được lập kế hoạch trước, và tiến độ được đo lường dựa trên kế hoạch này.



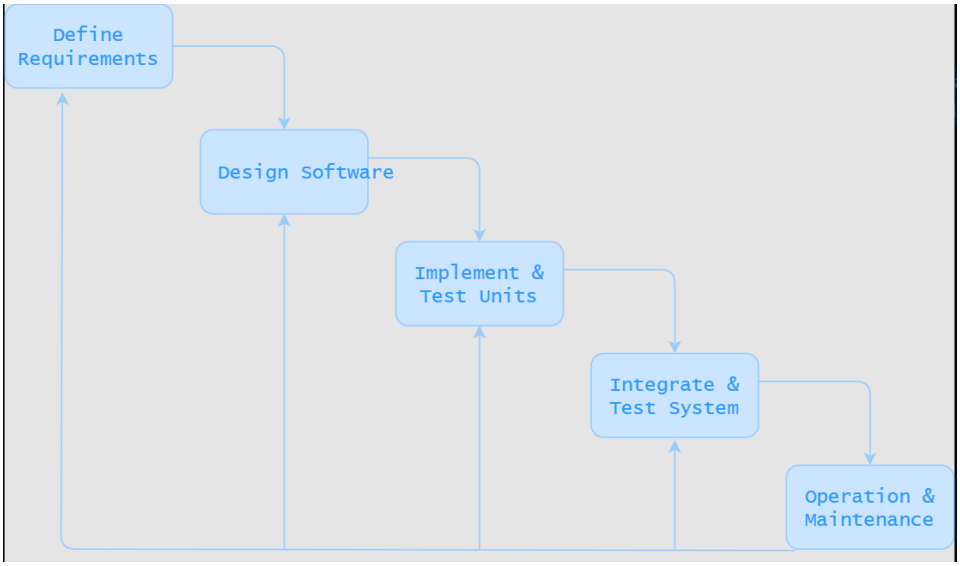
2. **Agile:** Một quy trình linh hoạt là một quy trình mà lập kế hoạch là một cách tăng dần, và dễ dàng thay đổi quy trình phù hợp với yêu cầu thay đổi của khách hàng.



The Waterfall Model

Introduction

Mô hình thác nước (Waterfall model) là mô hình đầu tiên được công bố về quy trình phát triển phần mềm, và nó bao gồm việc biểu diễn các hoạt động cơ bản của xác định, phát triển, xác nhận và tiến hóa như là các giai đoạn quy trình riêng biệt, tuần tự.



Principle stages

Mô hình thác nước có nhiều giai đoạn tham gia, và lý thuyết là mỗi giai đoạn phải hoàn thành trước khi giai đoạn tiếp theo có thể "cascaded" vào, đó là lý do tại sao mô hình này được gọi là mô hình thác nước.

Mỗi trong số năm giai đoạn được miêu tả chi tiết hơn trong phần còn lại của bài học.

1. **Analyse and define requirements**

Các dịch vụ dự kiến, ràng buộc tiềm năng và mục tiêu của hệ thống được xác định và sau đó được định rõ chi tiết. Những yêu cầu này sau đó được sử dụng như một thông số kỹ thuật sản phẩm, dựa theo đó trang web được phát triển.

2. **Design software**

Quá trình thiết kế bao gồm phân bổ các yêu cầu đã thực hiện ở giai đoạn trước cho các thành phần phần mềm khác nhau bằng cách xác định kiến trúc hệ thống tổng thể. Thiết kế phần mềm bao gồm xác định và mô tả các trừu tượng cơ bản của hệ thống phần mềm và các mối quan hệ giữa chúng mà trang web của bạn sẽ cần.

3. **Implement and test units**

Trong giai đoạn này, thiết kế phần mềm được dịch thành một tập hợp các chương trình hoặc nhiều đơn vị chương trình. Kiểm thử đơn vị (Unit testing) bao gồm xác minh rằng mỗi đơn vị đáp ứng các thông số kỹ thuật của nó.

4. **Integrate units and test system**

Các đơn vị chương trình riêng lẻ được tích hợp và kiểm tra như một hệ thống hoàn chỉnh để đảm bảo các yêu cầu phần mềm đã được đáp ứng. Sau khi kiểm thử, hệ thống phần mềm được giao cho khách hàng. Trong trường hợp trang web của bạn, bạn sẽ triển khai nó vào thời điểm này.

5. Operation and maintenance

Thường (mặc dù không nhất thiết), đây là giai đoạn trong vòng đời của phần mềm kéo dài nhất. Hệ thống được cài đặt và đưa vào sử dụng thực tế. Bảo trì bao gồm sửa chữa các lỗi không được phát hiện trong các giai đoạn trước của vòng đời, cải thiện việc thực hiện các đơn vị hệ thống và nâng cao dịch vụ của hệ thống khi có yêu cầu mới được phát hiện.

Conclusion

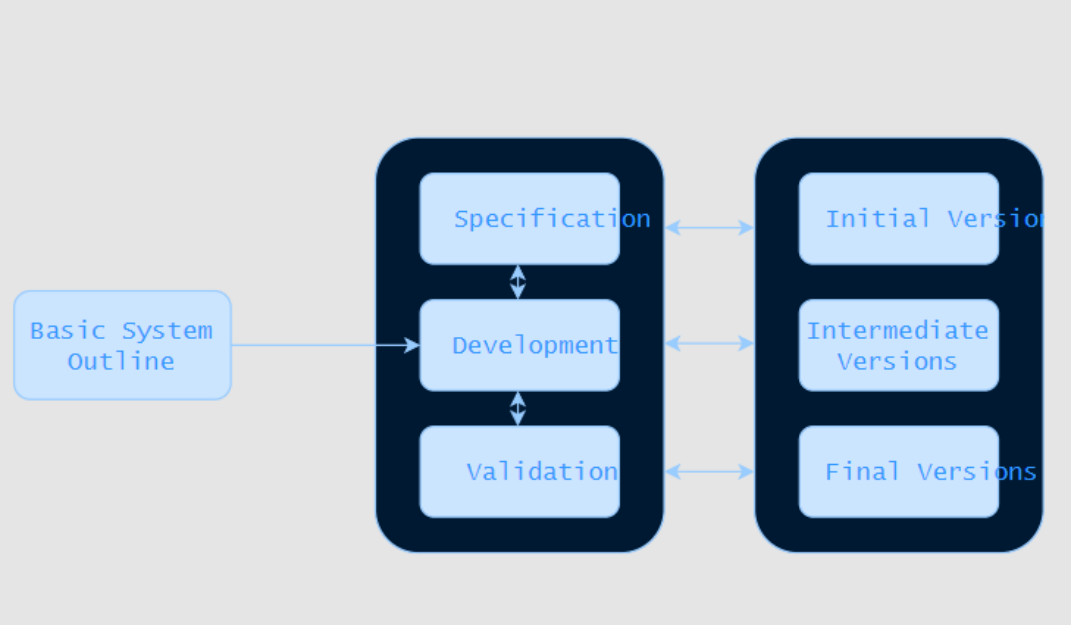
Mô hình thác nước được phân loại là một quy trình định hướng kế hoạch (plan-driven) vì nó yêu cầu một kế hoạch chi tiết phải được thiết lập trước khi thực hiện mỗi giai đoạn. Mô hình này hoạt động hiệu quả nhất trong các tổ chức lớn có nhiều người làm việc trên các dự án quy mô lớn cần có tài liệu và hồ sơ chi tiết để phối hợp với nhau. Tuy nhiên, trong trường hợp các trang web cá nhân hoặc các dự án nhỏ, phương pháp này có thể thấy là quá chi tiết và dư thừa vì không cần phải giữ hồ sơ phức tạp trong tình huống đó.

Incremental Model

Introduction

Phương pháp phát triển theo từng bước (incremental development) là một quy trình linh hoạt cơ bản, bao gồm xen kẽ các hoạt động xác định, phát triển và xác nhận. Hệ thống sau đó được phát triển dưới dạng một loạt các phiên bản hoặc bước tăng dần, với mỗi phiên bản bổ sung chức năng so với phiên bản trước.

Ý tưởng cơ bản trong quy trình phát triển theo từng bước là phát triển một phiên bản ban đầu của trang web của bạn, mở nó để nhận phản hồi và sau đó phát triển các phiên bản mới tăng dần dựa trên phản hồi này. Trong trường hợp này, các giai đoạn xác định, phát triển và xác nhận đều diễn ra đồng thời và tương tác lẫn nhau.



Phương pháp phát triển theo từng bước (incremental software development) phù hợp hơn phương pháp thác nước đối với hầu hết các hệ thống doanh nghiệp và cá nhân. Điều này bởi vì phát triển theo từng bước là một cách giải quyết vấn đề tự nhiên hơn. Nó tương đương với tiến tới một giải pháp thông qua

một loạt các bước, quay lại để sửa sai lầm và tiếp tục cải tiến đến khi đạt được giải pháp cuối cùng. Hơn nữa, việc phát triển phần mềm một cách tăng dần có nghĩa là giá thành rẻ hơn và dễ dàng thêm các thay đổi vào nó trong quá trình phát triển thay vì phải chờ đợi để phát hiện vấn đề sau khi đã trải qua toàn bộ quá trình phát triển.

Incremental development vs waterfall model

Như chúng ta đã thấy, mô hình phát triển theo từng bước (incremental development) thường hiệu quả hơn mô hình thác nước (waterfall model) đối với phát triển các trường hợp sử dụng phổ biến nhất của trang web. Ngoài ra, còn có một số lợi ích khác mà phát triển theo từng bước mang lại so với mô hình thác nước, và chúng được liệt kê như sau:

1. Chi phí điều chỉnh yêu cầu của khách hàng giảm đáng kể trong trường hợp này vì lượng phân tích và tài liệu phải làm lại ít hơn rất nhiều so với mô hình thác nước.
2. Dễ dàng nhận phản hồi từ khách hàng về công việc phát triển đã thực hiện vì họ có thể bình luận trực tiếp trên các phiên bản thực tế của phần mềm và nhìn thấy mức độ triển khai thay vì chỉ đánh giá dựa trên tài liệu thiết kế phần mềm.
3. Có thể giao hàng và triển khai phần mềm hữu ích cho khách hàng nhanh hơn, ngay cả khi chưa bao gồm toàn bộ chức năng. Khách hàng có thể sử dụng và thu được giá trị từ phần mềm sớm hơn là có thể với quy trình thác nước.

Conclusion

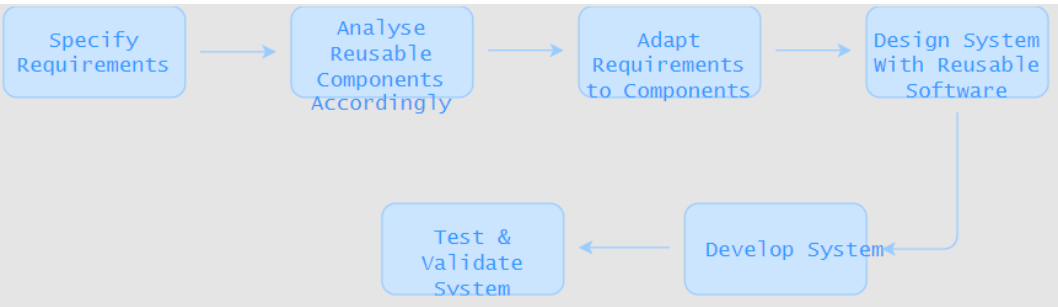
Mô hình phát triển theo từng bước (incremental development) là một phương pháp đơn giản hơn rất nhiều và dễ triển khai, cho phép phát triển phần mềm hiệu quả trên quy mô nhỏ. Đối với các trang web không phức tạp và không đòi hỏi nhiều nhóm người cùng làm việc, phương pháp này là phù hợp nhất vì nó cho phép thực hiện một sự đơn giản và hiệu quả hệ thống đồng thời.

Reuse-oriented Software Engineering

Introduction

Như chúng ta đã thấy trong chương về các khung thức (frameworks), hầu hết các dự án phần mềm đòi hỏi việc sử dụng lại phần mềm khi những người làm việc trên dự án biết về các thiết kế hoặc mã nguồn đã tồn tại có tính tương đồng với những gì được yêu cầu. Họ tìm kiếm những phần này, chỉnh sửa chúng theo cần thiết và tích hợp chúng vào hệ thống của họ theo yêu cầu cụ thể.

Tuy nhiên, trong thời gian gần đây, các quy trình phát triển phần mềm tập trung vào việc tái sử dụng phần mềm hiện có đã trở nên rộng rãi sử dụng. Các phương pháp hướng tới việc tái sử dụng dựa trên các thành phần phần mềm có thể tái sử dụng và một khung chương trình tích hợp để kết hợp các thành phần này. Đôi khi, các thành phần này là những hệ thống độc lập có thể cung cấp các chức năng cụ thể, tương tự như các khung mà chúng ta đã học trước đó.



Advantages & Disadvantages

Kỹ thuật phần mềm hướng tái sử dụng có lợi thế rõ ràng là giảm đáng kể việc phát triển thực tế phải làm từ đầu, điều này đảm bảo việc giao hàng sản phẩm nhanh hơn.

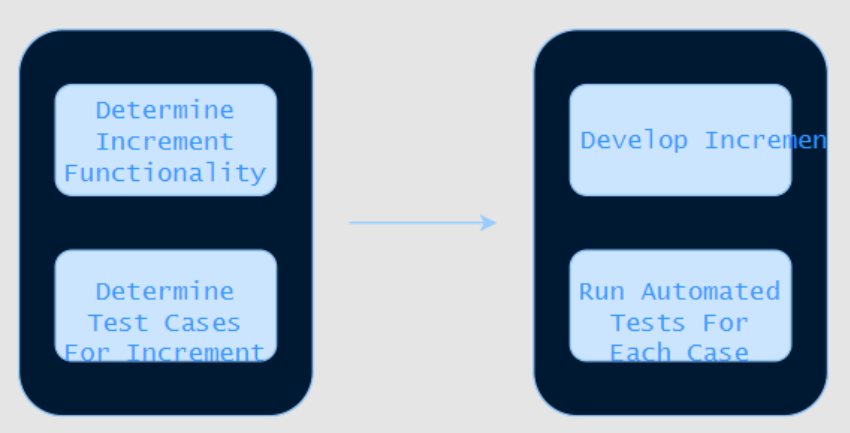
Tuy nhiên, việc tái sử dụng phần mềm khác cũng đồng nghĩa với việc yêu cầu sẽ không tránh khỏi việc thay đổi theo từng trường hợp cụ thể của phần mềm được sử dụng, có thể không hoàn toàn đáp ứng đủ các yêu cầu của phần mềm đang phát triển. Điều này cũng đồng nghĩa với việc có sự mất kiểm soát đáng kể về cách phần mềm phát triển vì tiến hóa của nó sau đó phụ thuộc vào cách các thành phần của bên thứ ba tiến hành phát triển.

Conclusion

Kỹ thuật phần mềm hướng tái sử dụng là phương pháp phổ biến nhất trong việc phát triển phần mềm hiện nay. Với sự ra đời của các khung thư như Express và Angular, phát triển web đã trở nên đơn giản đáng kể, và các yếu tố tinh tế được cung cấp cho mã phức tạp được yêu cầu trong hầu hết các trang web. Do đó, việc sử dụng những yếu tố tinh tế này khi phát triển một trang web ban đầu là một ý tưởng tốt để tạo ra các chương trình tinh vi cho phép bạn hiểu rõ chức năng bạn đang phát triển thay vì bị rối trong những chi tiết phụ của mã sản xuất chúng.

Test Driven Development

Test-driven development (TDD) là một phương pháp phát triển chương trình mà bạn xen kẽ cả hai công việc kiểm thử và phát triển mã. Cơ bản, bạn phát triển mã của mình một cách tăng dần và đồng thời phát triển một bài kiểm tra cho từng bước tăng dần. Bạn không tiến hành bước tăng dần tiếp theo cho đến khi mã mà bạn đã phát triển vượt qua bài kiểm tra của nó.



Automated testing

1. Selenium

Selenium là một trong những công cụ thường được sử dụng nhất để kiểm thử ứng dụng web. Mục đích chính của nó là tự động hóa trình duyệt, điều này có nghĩa là Selenium cho phép thực hiện các bài kiểm tra có thể tự động truy cập vào ứng dụng đang được phát triển và kiểm tra xem các chức năng dự kiến đã được triển khai đúng cách hay chưa. Ý tưởng chính của Selenium là cho phép người dùng xác định một tập hợp các hoạt động để trình duyệt web thực hiện, để ứng dụng có thể tự động mở và kiểm tra thông qua công cụ này, giống như việc con người kiểm tra chức năng một cách thủ công.

2. Jest

Jest là một công cụ kiểm thử JavaScript tích hợp và "không cần cấu hình" thường được Facebook sử dụng để kiểm thử toàn bộ mã JavaScript của mình, bao gồm cả các ứng dụng React. Jest hoạt động với mọi ngôn ngữ biên dịch thành JavaScript và tích hợp một cách mượt mà với Babel, điều này có nghĩa là bạn có thể viết React, TypeScript, và nhiều ngôn ngữ khác mà không cần cấu hình phức tạp.

3. PyUnit

PyUnit là một framework kiểm thử đơn vị của Python. Một bài kiểm tra đơn vị nhắm đến một phần nhỏ của mã, chẳng hạn như một phương thức thực hiện một chức năng cụ thể. Còn được gọi là unittest, PyUnit hỗ trợ tự động hóa kiểm thử, khả năng chia sẻ mã thiết lập và dừng cho các bài kiểm tra, tổng hợp các bài kiểm tra thành các bộ sưu tập, cũng như khả năng giữ cho các bài kiểm tra độc lập với hệ thống báo cáo. PyUnit là một công cụ cần thiết trong việc kiểm thử chức năng dự kiến của một ứng dụng web cụ thể.

4. Junit

JUnit là phiên bản tương đương của PyUnit dành cho ngôn ngữ lập trình Java và cho phép người dùng viết các bài kiểm tra đơn vị trong ngôn ngữ lập trình Java.

JUnit, giống như PyUnit, đã đóng vai trò quan trọng trong phát triển của test-driven development (TDD) và là một trong những framework kiểm thử đơn vị trong gia đình xUnit.

Tổng kết

- Internet chỉ là một mạng kết nối các thiết bị với nhau. Một số trong số các thiết bị này là máy chủ web và máy khách web.
- Máy khách web là các thiết bị dùng để truy cập và xem các trang web. Máy chủ web "đăng trữ" những trang web này, điều này có nghĩa là nó chứa mã của các trang web. Mã của trang web có thể được chia thành hai phần: front-end và back-end.
- Mã front-end bao gồm HTML, CSS và JavaScript. Hầu hết các trang web hiện đại không sử dụng các ngôn ngữ này theo dạng gốc (plain/vanilla); thay vào đó, thường sử dụng các khung thư viện (front-end frameworks).
- Khung thư viện giúp việc lập trình front-end đơn giản hơn vì hầu hết các chức năng cấp thấp và chung đã được cài đặt sẵn. Phần back-end bao gồm một máy chủ, phần mềm máy chủ và một cơ sở dữ liệu.
- Cơ sở dữ liệu lưu trữ toàn bộ dữ liệu của trang web mà người dùng có thể truy vấn thông qua các chức năng được cung cấp tại phần front-end.
- Người có khả năng về cả lập trình front-end và back-end được gọi là lập trình viên full-stack.
- Git là một hệ thống quản lý phiên bản mà bạn có thể sử dụng để theo dõi các thay đổi bạn thực hiện trên mã code của mình. Nó cũng có thể giúp bạn cộng tác và hoàn tác bất kỳ thay đổi nào mà bạn thực hiện nếu gặp phải lỗi sau này. Điều này rất hữu ích vì bạn không cần phải theo dõi dự án của mình bằng cách đặt tên các tập tin như 'code-1.cpp', 'code-2.cpp', 'code-with-jeffs-changes.cpp'. Nó cũng giúp giữ cho thư mục của bạn gọn gàng và dễ quản lý.
- Phát triển phần mềm là một nghệ thuật và có nhiều phương pháp tiếp cận trong việc phát triển phần mềm, mỗi phương pháp phù hợp với các tình huống khác nhau.