|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **HỌC VIỆN KỸ THUẬT QUÂN SỰ**  **KHOA CÔNG NGHỆ THÔNG TIN**  \*\*\*\*\*\* Description: bs00975_ \*\*\*\*\*\*  Description: bieu tuong  **ĐỒ ÁN TỐT NGHIỆP**  **XÂY DỰNG HỆ THỐNG HỌC TRỰC TUYẾN, KIỂM TRA, ĐÁNH GIÁ CHẤT LƯỢNG ONLINE**   |  |  | | --- | --- | | Giáo viên hướng dẫn: | Nguyễn Thị Hiền | | Sinh viên thực hiện: | Nguyễn Quang Dự | |  | Bùi Ngọc Toàn |     Hà Nội, 2020 |

MỤC LỤC

[LỜI CẢM ƠN 4](#_Toc99565721)

[1 CHƯƠNG 1. GIỚI THIỆU 5](#_Toc99565722)

[1.1 Tính cấp thiết, tình hình nghiên cứu liên quan đến đề tài 5](#_Toc99565723)

[1.2 Mục tiêu, nhiệm vụ của đề tài 5](#_Toc99565724)

[1.3 Phương pháp nghiên cứu 5](#_Toc99565725)

[1.4 Đối tượng và phạm vi nghiên cứu 5](#_Toc99565726)

[2 CHƯƠNG 2. KIẾN THỨC CƠ SỞ 5](#_Toc99565727)

[2.1 FastAPI 5](#_Toc99565728)

[2.1.1 Khái niệm 5](#_Toc99565729)

[2.1.2 Lý do chọn FastAPI 6](#_Toc99565730)

[2.2 MongoDB 8](#_Toc99565731)

[*2.2.1* *NoSql -* cơ sở dữ liệu phi quan hệ: 8](#_Toc99565732)

[2.2.2 MongoDB là gì? 9](#_Toc99565733)

[2.2.3 Một số câu lệnh cơ bản trên MongoDB 10](#_Toc99565734)

[2.3 Redis 13](#_Toc99565735)

[2.3.1 Redis là gì? 13](#_Toc99565736)

[2.3.2 Các ứng dụng của Redis 13](#_Toc99565737)

[2.3.3 Các kiểu dữ liệu trong Redis 14](#_Toc99565738)

[2.3.4 Persistent redis là gì 14](#_Toc99565739)

[2.4 Websocket 16](#_Toc99565740)

[2.4.1 Socket là gì? 16](#_Toc99565741)

[2.4.2 Socket sử dụng như thế nào? 16](#_Toc99565742)

[2.4.3 Socket hoạt động như thế nào? 16](#_Toc99565743)

[2.4.4 Phân loại Socket 16](#_Toc99565744)

[2.4.5 Web socket là gì? 18](#_Toc99565745)

[2.5 Docker 20](#_Toc99565746)

[2.5.1 Container trong Docker là gì? 21](#_Toc99565747)

[2.5.2 Các khái niệm liên quan 22](#_Toc99565748)

[2.5.3 Quy trình thực thi của một hệ thống sử dụng Docker 23](#_Toc99565749)

[2.6 Angular 24](#_Toc99565750)

[2.6.1 Khái niệm Angular là gì? 24](#_Toc99565751)

[2.6.2 Đôi nét về Lịch sử phát triển Angular 25](#_Toc99565752)

[2.6.3 Ưu điểm và nhược điểm của AngularJS : 25](#_Toc99565753)

[2.6.4 Sử dụng Angular sẽ làm được gì? 26](#_Toc99565754)

[2.6.5 Các đặc trưng cơ bản của Angular là gì? 27](#_Toc99565755)

[2.6.6 Cách thức hoạt động của Angular 27](#_Toc99565756)

[2.6.7 Giới thiệu tính năng cơ bản của AngularJS 27](#_Toc99565757)

# LỜI CẢM ƠN

Để hoàn thành đồ án này, em xin chân thành gửi lời cảm ơn đến Cô giáo Nguyễn Thị Hiền, giảng viên trực tiếp phụ trách hướng dẫn đề tài đồ án. Sự hướng dẫn tận tình của cô đã giúp em có thêm nhiều kiến thức, khắc phục được những lỗi em thường mắc phải để đồ án được hoàn thiện tốt hơn. Qua những hướng dẫn tận tâm của cô đã giúp chúng em hiểu sâu hơn về những kiến thức đã học, và trau dồi thêm những kiến thức mới bổ ích.

Em xin chân thành gửi lời cảm ơn quý thầy cô trong bộ môn cũng như tất cả các thầy cô trong khoa Công nghệ thông tin của Học Viện Kỹ Thuật Quân Sự đã tận tình giúp đỡ và trang bị cho em những kiến thức cơ bản, cần thiết để em có thể hoàn thành tốt đồ án này.

Ngoài ra, em xin gửi lời cảm ơn tới Ban Giám hiệu Học Viện Kỹ Thuật Quân Sự đã tạo điều kiện để em thực hiện tốt đồ án này.

Mặc dù có nhiều cố gắng để thực hiện đồ án tốt nghiệp một cách hoàn chỉnh nhất. Song do năng lực còn nhiều hạn chế nên không thể tránh khỏi những thiếu sót nhất định.

Em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các quý thầy cô để em có điều kiện bổ sung, nâng cao ý thức của mình, phục vụ tốt hơn công tác thực tế sau này.

*Em xin chân thành cảm ơn!*

# CHƯƠNG 1. GIỚI THIỆU

## Tính cấp thiết, tình hình nghiên cứu liên quan đến đề tài

Với sự phát triển không ngừng của khoa học và công nghệ trong thời đại 4.0, việc sử dụng internet trở nên phổ biến và dễ tiếp cận hơn rất nhiều so với trước đó. Theo đó, chúng ta đang hướng đến việc sử dụng internet để tối ưu hóa mọi công việc trong cuộc sống. Bây giờ, chúng ta có thể “gặp mặt”, nói chuyện với nhau thông qua video call, không còn vấn đề về khoảng cách địa lý, chỉ cần có internet, mọi người có thể kết nối với nhau mọi lúc.

Với tình hình Covid-19 đang lây lan trên toàn cầu, việc đi lại trở nên khó khăn hơn. Đặc biệt là việc học sinh đến trường để học tập sẽ cần thời gian để chúng ta có thể kiểm soát được dịnh bệnh, đảm bảo an toàn cho giáo viên và học sinh khi đến trường. Trong tình hình đó, việc dạy và học online là lựa chọn phù hợp nhất. Và từ đó, chúng ta sẽ hướng tới việc dạy và học online có thể trở thành hình thức dạy học chính thức trong thời đại mới.

## Mục tiêu, nhiệm vụ của đề tài

* Nghiên cứu công nghệ Web Angular, Fast API, MongoDB
* Xây dựng giao diện website
* Tổ chức quản trị hệ thống: phân quyền sử dụng hệ thống đối với các đối tượng khác nhau
* Tổ chức quản lý khóa học: tạo mới, quản lý khóa học
* Build ứng dụng lên server

## Phương pháp nghiên cứu

* Nghiên cứu công nghệ Web Angular, Fast API, MongoDB

## Đối tượng và phạm vi nghiên cứu

* Đối tượng nghiên cứu: sinh viên
* Phạm vi nghiên cứu: Trường Học viện Kỹ thuật Quân sự

# CHƯƠNG 2. KIẾN THỨC CƠ SỞ

## FastAPI

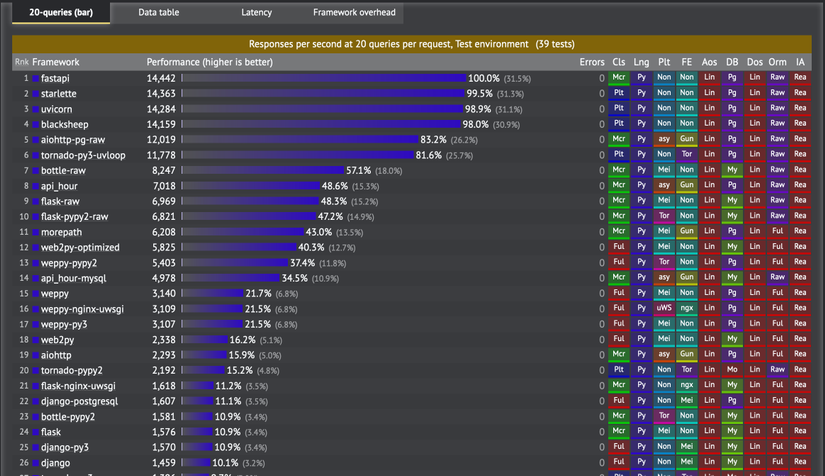
### Khái niệm

FastAPI là nền tảng thiết kế, lập trình xây dựng API cực kỳ nhanh trên cả 2 phương diện phát triển và thực thi trên Python 3.6+.Từ Python 3.6+ thì bạn đã có thể sử dụng cú pháp await/async để chạy code bất đồng bộ, vì lý do này các framework trên Python sẽ đạt được hiệu năng cao, FastAPI là một trong số Python framework nhanh nhất hiện nay.

### Lý do chọn FastAPI

FastAPI là một micro framework khá mới, chỉ vừa được release năm 2018. Github của framework này hiện tính đến tháng 1/2021 đang đạt 25,4k star, tuy nhiên do được áp dụng khá nhiều công nghệ mới nên có FastAPI có vài điểm mạnh mà mình cảm thấy khá phù hợp để sử dụng phát triển project:

* High performance Do được base trên 2 lib khá mạnh ở thời điểm hiện tại của python là Pydantic và Starlette nên FastAPI sở hữu hiệu suất cao nhất trong tất cả các framework Python hiện nay, bạn có thể tham khảo so sánh hiệu năng giữa các framework tại [https://www.techempower.com/](https://www.techempower.com/benchmarks/#section=test&runid=7464e520-0dc2-473d-bd34-dbdfd7e85911&hw=ph&test=query&l=zijzen-7) .



* Development Speed Được hỗ trợ tích hợp sẵn giao diện Swagger – OpenAPI kèm theo cách code khá đơn giản nên lập trình có thể release function rất nhanh mà vẫn có document đầy đủ, đây là lợi thế có thể nói là quan trọng nhất của FastAPI so với các Framework khác. Dưới đây là một đoạn code in ra dòng text healthcheck.
* Bất đồng bộ Hiện tại bất đồng bộ đã được hỗ trợ từ phiên bản Django 3.x nhưng ngay từ khi release, FastAPI mặc định đã hỗ trợ developer Async, cũng vì vậy mà FastAPI chỉ có thể sử dụng với python3.6 trở lên.

*So sánh 3 Framework Python là Django, Flask and FastAPI*

Dưới đây là bảng so sánh giữa 3 Framework Django, Flask và FastAPI:

|  | **Django** | **Flask** | **FastAPI** |
| --- | --- | --- | --- |
| Community | Cộng đồng của Django hiện tại khá lớn và lâu đời, vì vậy số lượng lib hỗ trợ cho Django có thể nói là nhiều nhất ở thời điểm hiện tại | Cộng đồng của Flask hiện tại cũng lớn nhưng không bằng Django, bởi vậy số lượng lib cũng hạn chế hơn Django | Cộng đồng của FastAPI là khá mới, do vậy số lượng lib hỗ trợ khá ít |
| Python version | All version: 2x, 3x | All version: 2x, 3x | Only python3.x |
| Performance | Do hỗ trợ nhiều feature nên perfomance của Django không được cao | Performance khá cao | Là framework có hiệu năng cao nhất trong các framework python hiện tại |
| ORM | Django ORM | Flask-sqlachemy | Sqlachemy |
| Async | Hỗ trợ từ Django 3.x | Chưa hỗ trợ | Tương thích hoàn toàn |
| Builtin Admin UI | Có | Không | Không |
| Project Frame | Có, chỉ cần run python manage startapp yourapp | Không | Không |
| Python shell | Có | Có | Không – đây là hạn chế của một microframework so với các framework thực thụ |
| OpenAPI document | Phổ biến nhất là sử dụng Django-rest-framework, với Swagger thì dùng django-rest-swagger với python2.x, dùng drf-yasg với python3.x | Sử dụng Flask-restplus | Mặc định đã tích hợp sẵn Swagger và Redoc |

## MongoDB

### *NoSql -* cơ sở dữ liệu phi quan hệ:

* NoSQL là 1 dạng CSDL mã nguồn mở và được viết tắt bởi: None-Relational SQL hay có nơi thường gọi là Not-Only SQL.
* NoSQL được phát triển trên Javascript Framework với kiểu dữ liệu là JSON và dạng dữ liệu theo kiểu key và value.
* NoSQL ra đời như là 1 mảnh vá cho những khuyết điểm và thiếu xót cũng như hạn chế của mô hình dữ liệu quan hệ RDBMS (Relational Database Management System - Hệ quản trị cơ sở dữ liệu quan hệ) về tốc độ, tính năng, khả năng mở rộng,...
* Với NoSQL bạn có thể mở rộng dữ liệu mà không lo tới những việc như tạo khóa ngoại, khóa chính, kiểm tra ràng buộc .v.v ...
* NoSQL bỏ qua tính toàn vẹn của dữ liệu và transaction để đổi lấy hiệu suất nhanh và khả năng mở rộng.
* NoSQL được sử dụng ở rất nhiều công ty, tập đoàn lớn, ví dụ như FaceBook sử dụng Cassandra do FaceBook phát triển, Google phát triển và sử dụng BigTable,...

Đến đây chắc các bạn cũng phần nào hiểu về NoSql rồi phải không, giờ chúng ta đi đến khái niệm MongoDB nhé.

### MongoDB là gì?

* MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở, là CSDL thuộc NoSql và được hàng triệu người sử dụng.
* MongoDB là một database hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON thay vì dạng bảng như CSDL quan hệ nên truy vấn sẽ rất nhanh.
* Với CSDL quan hệ chúng ta có khái niệm bảng, các cơ sở dữ liệu quan hệ (như MySQL hay SQL Server...) sử dụng các bảng để lưu dữ liệu thì với MongoDB chúng ta sẽ dùng khái niệm là **collection** thay vì bảng
* So với RDBMS thì trong MongoDB **collection** ứng với **table**, còn **document** sẽ ứng với **row** , MongoDB sẽ dùng các document thay cho row trong RDBMS.
* Các collection trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ không cần tuân theo một cấu trúc nhất định.
* Thông tin liên quan được lưu trữ cùng nhau để truy cập truy vấn nhanh thông qua ngôn ngữ truy vấn MongoDB

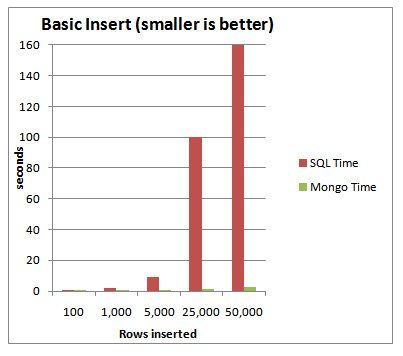
Vậy là mình đã giới thiệu xong khái niệm và một số điểm khác cơ bản của MongoDB so với CSDL quan hệ rồi, tiếp theo sẽ là một vài thao tác với MongoDB để các bạn có thể hình dung rõ hơn nhé

### Một số câu lệnh cơ bản trên MongoDB

| **CSDL** | **MySQL** | **MongoDB** |
| --- | --- | --- |
| Tạo csdl | CREATE DATABASE test; | use test; |
| Tạo bảng | CREATE TABLE students (ten\_cot - kieu\_du\_lieu); | db.createCollection('students'); |
| Tạo bản ghi | INSERT INTO studetns ('name', 'gender') VALUES('thanh', 'male'); | db.students.insert({ name:'thanh', gender: 'male'}); |
| Cập nhật | UPDATE students SET name = 'thanh update' WHERE id = 1; | db.students.update({ \_id: 1 },{$set:{ name: 'thanh update' }}); |
| Xóa bản ghi | DELETE FROM students Where id = 1; | db.students.remove({ \_id: 1}); |
| Tìm kiếm all | SELECT \* FROM students; | db.students.find({}); |
| Tìm kiếm | SELECT \* FROM students WHERE name = 'thanh'; | db.students.find({ name: 'thanh' }); |

*Ưu điểm của mongoDB.*

* Do MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ có các kích cỡ và các document khác nhau, linh hoạt trong việc lưu trữ dữ liệu, nên bạn muốn gì thì cứ insert vào thoải mái.
* Dữ liệu trong MongoDB không có sự ràng buộc lẫn nhau, không có join như trong RDBMS nên khi insert, xóa hay update nó không cần phải mất thời gian kiểm tra xem có thỏa mãn các ràng buộc dữ liệu như trong RDBMS.
* MongoDB rất dễ mở rộng (Horizontal Scalability). Trong MongoDB có một khái niệm cluster là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node với vào cluster:
* Trường dữ liệu “\_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.
* Khi có một truy vấn dữ liệu, bản ghi được cached lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.
* Hiệu năng cao: Tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Với một lượng dữ liệu đủ lớn thì thử nghiệm cho thấy tốc độ insert của MongoDB có thể nhanh tới gấp 100 lần so với MySQL.



*Nhược điểm của mongoDB.*

* Một ưu điểm của MongoDB cũng chính là nhược điểm của nó. MongoDB không có các tính chất ràng buộc như trong RDBMS nên khi thao tác với mongoDB thì phải hết sức cẩn thận.
* Tốn bộ nhớ do dữ liệu lưu dưới dạng key-value, các collection chỉ khác về value do đó key sẽ bị lặp lại. Không hỗ trợ join nên dễ bị dữ thừa dữ liệu.
* Khi insert/update/remove bản ghi, MongoDB sẽ chưa cập nhật ngay xuống ổ cứng, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng điêù này sẽ là nhược điểm vì sẽ có nguy cơ bị mất dữ liệu khi xảy ra các tình huống như mất điện...

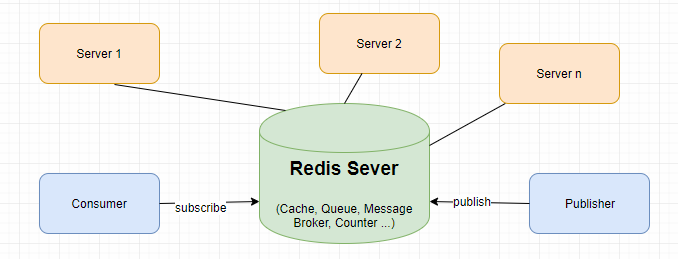
*Khi nào nên dùng MongoDB*

Ví dụ như các hệ thống realtime (thời gian thực) yêu cầu phản hồi nhanh, Các hệ thống bigdata với yêu cầu truy vấn nhanh hay các hệ thống có lượng request lớn thì MongoDB sẽ là sự lựa chọn ưu tiên hơn CSDL quan hệ. Tùy theo dự án và trường hợp cụ thể để sử dụng CSDL quan hệ hay sử dụng MongoDB đem lại hiệu quả cao.

## Redis

### Redis là gì?

**Redis (REmote DIctionary Server)** là một mã nguồn mở được dùng để lưu trữ dữ liệu có cấu trúc, có thể sử dụng như một database, bộ nhớ cache hay một message broker.



### **Các ứng dụng của Redis**

Ngoài tính năng lưu trữ KEY-VALUE trên RAM thì Redis còn hỗ trợ tính năng xắp xếp, query, backup dữ liệu trên đĩa cứng cho phép bạn có thể phục hồi dữ liệu khi hệ thống gặp sự cố…và có thể nhân bản (Chạy nhiều Server Redis cùng lúc).

* **Caching:**Sử dụng làm bộ nhớ đệm. Chính tốc độ đọc ghi nhanh mà Redis có thể làm bộ nhớ đệm, nơi chia sẻ dữ liệu giữa các ứng dụng hoặc làm database tạm thời. Ngoài ra Redis có thể sử dụng để làm Full Page Cache cho website. Cũng vì tính nhất quán của Redis, cho dù restart Redis thì người dùng cũng không có cảm nhận chậm khi tải trang.
* **Counter:** Sử dụng làm bộ đếm. Với thuộc tính tăng giảm thông số rất nhanh trong khi dữ liệu được lưu trên RAM, sets và sorted sets được sử dụng thực hiện đếm lượt view của một website, các bảng xếp hạng trong game chẳng hạng. Redis hỗ trợ thread safe do đó nó có thể đồng bộ dữ liệu giữa các request.
* **Publish/Suscribe (Pub/Sub):** Tạo kênh chia sẻ dữ liệu. Redis hỗ trợ tạo các channel để trao đổi dữ liệu giữa publisher và subscriber giống như channel trong **Socket Cluster** hay topic trong **Apache Kafka. Ví dụ:**Pub/Sub được sử dụng theo dõi các kết nối trong mạng xã hội hoặc các hệ thống chat.
* **Queues:** Tạo hàng đợi để xử lý lần lượt các request. Redis cho phép lưu trữ theo list và cung cấp rất nhiều thao tác với các phần tử trong list, vì vậy nó còn được sử dụng như một message queue.

### **Các kiểu dữ liệu trong Redis**

Khác với RDMS như MySQL, hay PostgreSQL, Redis không có table (bảng). Redis lưu trữ data dưới dạng key-value. Thực tế thì memcache cũng làm vậy, nhưng kiểu dữ liệu của memcache bị hạn chế, không đa dạng được như Redis, do đó không hỗ trợ được nhiều thao tác từ phía người dùng. Dưới đây là sơ lược về các kiểu dữ liệu Redis dùng để lưu value.

**– STRING**: string, integer hoặc float. Redis có thể làm việc với cả string, từng phần của string, cũng như tăng/giảm giá trị của integer, float.

**– LIST**: List là một danh sách của strings, sắp xếp theo thứ tự insert. Redis có thể thêm một phần tử vào đầu hoặc cuối list. List phù hợp cho các bài toán cần thao tác với các phần tử gần đầu và cuối vì việc truy xuất này là cực nhanh, cho dù insert cả triệu phần tử. Tuy nhiên nhược điểm là việc truy cập vào các phần tử ở giữa list rất chậm.

**– SET**: tập hợp các string (không được sắp xếp). Redis hỗ trợ các thao tác thêm, đọc, xóa từng phần tử, kiểm tra sự xuất hiện của phần tử trong tập hợp. Ngoài ra Redis còn hỗ trợ các phép toán tập hợp, gồm intersect/union/difference.

**– HASH**: lưu trữ hash table của các cặp key-value, trong đó key được sắp xếp ngẫu nhiên, không theo thứ tự nào cả. Redis hỗ trợ các thao tác thêm, đọc, xóa từng phần tử, cũng như đọc tất cả giá trị.

**– SORTED SET (ZSET)**: là 1 danh sách, trong đó mỗi phần tử là map của 1 string (member) và 1 floating-point number (score), danh sách được sắp xếp theo score này. Các phần tử của zset được sắp xếp theo thứ tự từ score nhỏ tới lớn.

Ngoài ra, Redis còn hỗ trợ các data types khác như: Bit arrays, HyperLogLogs, Streams.

### **Persistent redis là gì**

Bên cạnh việc lưu key-value trên bộ nhớ RAM, Redis có 2 background threads chuyên làm nhiệm vụ định kỳ ghi dữ liệu lên đĩa cứng.

Có 2 loại file được ghi xuống đĩa cứng:

* **RDB (Redis DataBase file)**

RDB thực hiện tạo và sao lưu snapshot của DB vào ổ cứng sau mỗi khoảng thời gian nhất định.

* **Ưu điểm**

RDB cho phép người dùng lưu các version khác nhau của DB, rất thuận tiện khi có sự cố xảy ra.

Bằng việc lưu trữ data vào 1 file cố định, người dùng có thể dễ dàng chuyển data đến các data centers, máy chủ khác nhau.

Khi restart server, dùng RDB làm việc với lượng data lớn sẽ có tốc độ cao hơn là dùng AOF.

* **Nhược điểm**

RDB không phải là lựa chọn tốt nếu muốn giảm thiểu tối đa nguy cơ mất mát dữ liệu.

Thông thường người dùng sẽ set up để tạo RDB snapshot 5 phút 1 lần (hoặc nhiều hơn). Do vậy, trong trường hợp có sự cố, Redis không thể hoạt động, dữ liệu trong những phút cuối sẽ bị mất.

RDB cần dùng fork() để tạo tiến trình con phục vụ cho thao tác disk I/O. Trong trường hợp dữ liệu quá lớn, quá trình fork() có thể tốn thời gian và server sẽ không thể đáp ứng được request từ client trong vài milisecond hoặc thậm chí là 1 second tùy thuộc vào lượng data và hiệu năng CPU.

* **AOF (Append Only File)**

AOF lưu lại tất cả các thao tác write mà server nhận được, các thao tác này sẽ được chạy lại khi restart server hoặc tái thiết lập dataset ban đầu.

#### **Ưu điểm**

Sử dụng AOF sẽ giúp đảm bảo dataset được bền vững hơn so với dùng RDB. Người dùng có thể config để Redis ghi log theo từng câu query hoặc mỗi giây 1 lần.

Redis ghi log AOF theo kiểu thêm vào cuối file sẵn có, do đó tiến trình seek trên file có sẵn là không cần thiết. Ngoài ra, kể cả khi chỉ 1 nửa câu lệnh được ghi trong file log (có thể do ổ đĩa bị full), Redis vẫn có cơ chế quản lý và sửa chữa lối đó (redis-check-aof).

Redis cung cấp tiến trình chạy nền, cho phép ghi lại file AOF khi dung lượng file quá lớn.

#### **Nhược điểm**

File AOF thường lớn hơn file RDB với cùng 1 dataset.

AOF có thể chậm hơn RDB tùy theo cách thức thiết lập khoảng thời gian cho việc sao lưu vào ổ cứng. Tuy nhiên, nếu thiết lập log 1 giây 1 lần có thể đạt hiệu năng tương đương với RDB.

Developer của Redis đã từng gặp phải bug với AOF (mặc dù là rất hiếm), đó là lỗi AOF không thể tái tạo lại chính xác dataset khi restart Redis. Lỗi này chưa gặp phải khi làm việc với RDB bao giờ.

## Websocket

### Socket là gì?

Socket là gì? Socket là một điểm cuối (end-point) của liên kết giao tiếp hai chiều (two-way communication) giữa hai chương trình chạy trên mạng. Nghĩa là một socket được sử dụng để cho phép 1 process nói chuyện với 1 process khác.

Các lớp Socket được sử dụng để tiến hành kết nối giữa client và server. Nó được ràng buộc với một cổng port (thể hiện là một con số cụ thể) để các tầng TCP (TCP Layer) có thể định danh ứng dụng mà dữ liệu sẽ được gửi tới.

### Socket sử dụng như thế nào?

Sau khái niệm socket là gì? Thì mình sẽ giới thiệu đến chức năng của socket, nó được áp dụng trong quá trình làm việc, nếu có thể chạy nhiều socket cùng một lúc nên công việc của bạn sẽ nhanh hơn, nâng cao hiệu suất làm việc.

### Socket hoạt động như thế nào?

Socket giúp lập trình viên kết nối các ứng dụng để truyền và nhận dữ liệu trong môi trường có kết nối Internet bằng cách sử dụng phương thức TCPIP và UDP.

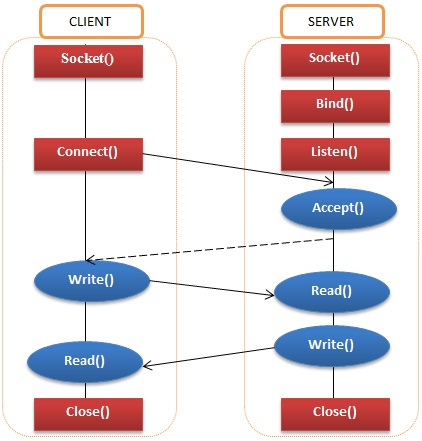
Khi cần trao đổi dữ liệu cho nhau thì 2 ứng dụng cần phải biết thông tin IP và port bao nhiêu của ứng dụng kia.

Có rất nhiều dạng socket khác nhau phụ thuộc vào sự khác biệt giữa cách truyền dữ liệu (protocol). Dạng phổ biến nhất là TCP và UDP.

### Phân loại Socket

#### Stream Socket là gì?

Dựa trên giao thức TCP( Tranmission Control Protocol), stream socket thiết lập giao tiếp 2 chiều theo mô hình client và server. Được gọi là socket hướng kết nối.



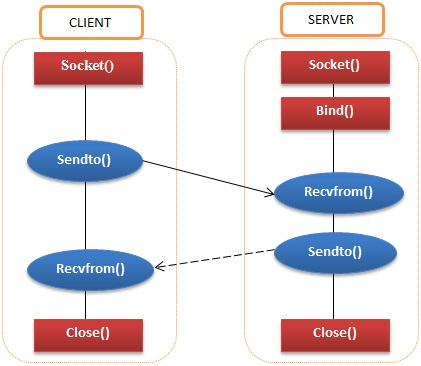
Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng tuần tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn.

**Đặc điểm tóm gọn:**

* Có một đường kết nối (địa chỉ IP) giữa 2 tiến trình.
* Một trong hai tiến trình kia phải đợi tiến trình này yêu cầu kết nối.
* Mô hình client /sever thì sever lắng nghe và chấp nhận từ client.
* Mỗi thông điệp gửi phải có xác nhận trả về.
* Các gói tin chuyển đi tuần tự.

#### Datagram Socket

Dựa trên giao thức UDP( User Datagram Protocol) việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 process. Tức là nó cung cấp connection-less point cho việc gửi và nhận packets. Gọi là socket không hướng kết nối



Do không yêu cầu thiết lập kết nối, không phải có những cơ chế phức tạp. Nên tốc độ giao thức khá nhanh, thuận tiện cho các ứng dụng truyền dữ liệu nhanh như chat, game online…

**Đặc điểm tóm gọn:**

* Hai tiến trình liên lạc với nhau không kết nối trực tiếp
* Thông điệp gửi đi phải kèm theo thông điệp người nhận
* Thông điệp có thể gửi nhiều lần
* Người gửi không chắc chắn thông điệp đến tay người nhận.
* Thông điệp gửi sau có thể đến trước và ngược lại.
* Để có thể thực hiện các cuộc giao tiếp, một trong 2 quá trình phải công bố port của socket mà mình đang sử dụng.

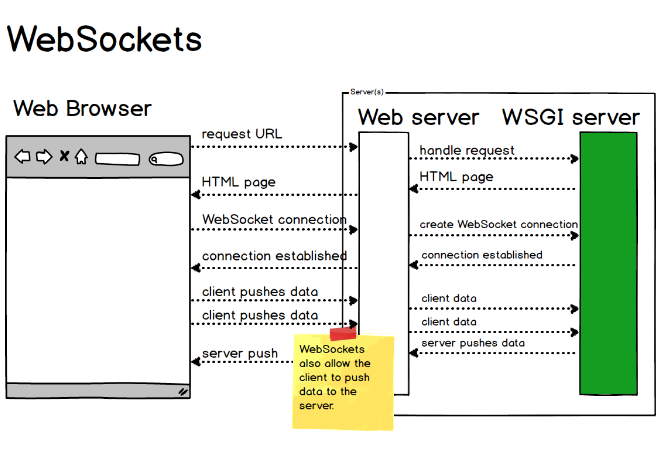
### Web socket là gì?

**Websocket** là giao thức hỗ trợ giao tiếp hai chiều giữa client và server để tạo một kết nối trao đổi dữ liệu. Giao thức này không sử dụng HTTP mà thực hiện nó qua TCP. Mặc dù được thiết kế để chuyên sử dụng cho các ứng dụng web, lập trình viên vẫn có thể đưa chúng vào bất kì loại ứng dụng nào.

#### Ưu điểm

WebSocket cung cấp giao thức giao tiếp hai chiều mạnh mẽ. No có độ trễ thấp và dễ xử lý lỗi. Websocket thường được sử dụng cho những trường hợp yêu cầu real time như chat, hiển thị biểu đồ hay thông tin chứng khoán.

#### Cấu trúc WebSocket



Giao thức chuẩn thông thường của WebSocket là ws:// , giao thức secure là wss:// . Chuẩn giao tiếp là String và hỗ trợ buffered arrays và blobs.

#### Các thuộc tính của WebSocket

|  |  |
| --- | --- |
| **THUỘC TÍNH** | **MÔ TẢ** |
| readyState | Diễn tả trạng thái kết nối. Nó có các giá trị sau:   * Giá trị 0: kết nối vẫn chưa được thiết lập (WebSocket.CONNECTING) * Giá trị 1: kết nối đã thiết lập và có thể giao tiếp (WebSocket.OPEN) * Giá trị 2: kết nối đang qua handshake đóng (WebSocket.CLOSING) * Giá trị 3: kết nối đã được đóng (WebSocket.CLOSED) |
| bufferedAmount | Biểu diễn số byte của UTF-8 mà đã được xếp hàng bởi sử dụng phương thức send() |

#### Các sự kiện WebSocket

|  |  |  |
| --- | --- | --- |
| **SỰ KIỆN** | **EVENT HANDLER** | **MÔ TẢ** |
| open | onopen | Khi một WebSocket chuyển sang trạng thái mở, “onopen” sẽ được gọi. |
| message | onmessage | Khi WebSocket nhận dữ liệu từ Server. |
| error | onerror | Có bất kỳ lỗi nào trong giao tiếp. |
| close | onclose | Kết nối được đóng. Những sự kiện được truyền cho “onclose” có ba tham số là “code”, “reason”, và “wasClean”. |

Event handlers có thể được tạo ra bằng cách sử dụng phương thức **addEventListener()**.

#### Các phương thức của WebSocket

|  |  |
| --- | --- |
| **PHƯƠNG THỨC** | **MÔ TẢ** |
| send() | send(data) gửi dữ liệu tới server. Message data là string, ArrayBuffer, blob. |
| close() | Đóng kết nối đang tồn tại. |

## Docker

**Docker** là một nền tảng để cung cấp cách để building, deploying và running ứng dụng dễ dàng hơn bằng cách sử dụng các containers (trên nền tảng ảo hóa). Ban đầu viết bằng Python, hiện tại đã chuyển sang Golang.

### Container trong Docker là gì?

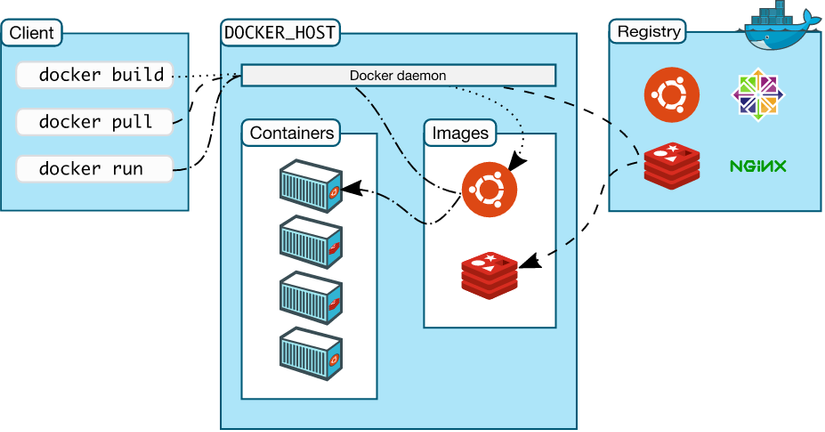
Các containers cho phép lập trình viên đóng gói một ứng dụng với tất cả các phần cần thiết, chẳng hạn như thư viện và các phụ thuộc khác, và gói tất cả ra dưới dạng một package.

Bằng cách đó, nhờ vào container, ứng dụng sẽ chạy trên mọi máy Linux khác bất kể mọi cài đặt tùy chỉnh mà máy có thể có khác với máy được sử dụng để viết code.

Theo một cách nào đó, Docker khá giống virtual machine. Nhưng tại sao Docker lại phát triển, phổ biến nhanh chóng? Đây là những nguyên nhân:

* **Tính dễ ứng dụng:** Docker rất dễ cho mọi người sử dụng từ lập trình viên, sys admin… nó tận dụng lợi thế của container để build, test nhanh chóng. Có thể đóng gói ứng dụng trên laptop của họ và chạy trên public cloud, private cloud… Câu thần chú là “Build once, run anywhere”.
* **Tốc độ:** Docker container rất nhẹ và nhanh, bạn có thể tạo và chạy docker container trong vài giây.
* **Môi trường chạy và khả năng mở rộng:** Bạn có thể chia nhỏ những chức năng của ứng dụng thành các container riêng lẻ. Ví dụng Database chạy trên một container và Redis cache có thể chạy trên một container khác trong khi ứng dụng Node.js lại chạy trên một cái khác nữa. Với Docker, rất dễ để liên kết các container với nhau để tạo thành một ứng dụng, làm cho nó dễ dàng scale, update các thành phần độc lập với nhau.

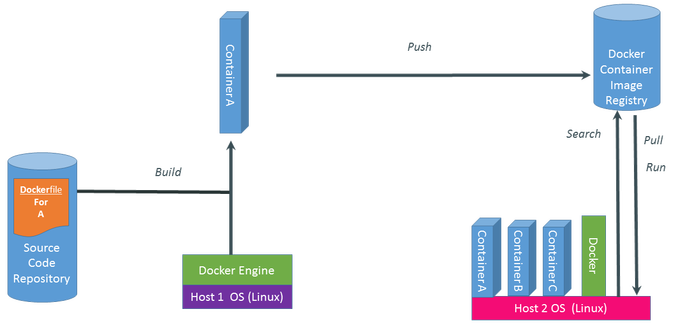
### Các khái niệm liên quan



* **Docker Engine** : là thành phần chính của Docker, như một công cụ để đóng gói ứng dụng
* **Docker Hub** : là một “github for docker images”. Trên DockerHub có hàng ngàn public images được tạo bởi cộng đồng cho phép bạn dễ dàng tìm thấy những image mà bạn cần. Và chỉ cần pull về và sử dụng với một số config mà bạn mong muốn.
* **Images**: là một khuôn mẫu để tạo một container. Thường thì image sẽ dựa trên 1 image có sẵn với những tùy chỉnh thêm. Ví dụ bạn build 1 image dựa trên image Centos mẫu có sẵn để chạy Nginx và những tùy chỉnh, cấu hình để ứng dụng web của bạn có thể chạy được. Bạn có thể tự build một image riêng cho mình hoặc sử dụng những image được chia sẽ từ cộng đồng Docker Hub. Một image sẽ được build dựa trên những chỉ dẫn của Dockerfile.
* **Container**: là một instance của một image. Bạn có thể create, start, stop, move or delete container dựa trên Docker API hoặc Docker CLI.
* **Docker Client**: là một công cụ giúp người dùng giao tiếp với Docker host.
* **Docker Daemon**: lắng nghe các yêu cầu từ Docker Client để quản lý các đối tượng như Container, Image, Network và Volumes thông qua REST API. Các Docker Daemon cũng giao tiếp với nhau để quản lý các Docker Service.
* **Dockerfile**: là một tập tin bao gồm các chỉ dẫn để build một image .
* **Volumes**: là phần dữ liệu được tạo ra khi container được khởi tạo.

Trên đây là những khái niệm cơ bản nhất về Docker. Ngoài ra còn nhiều khái niệm nữa như swarm, compose…

### Quy trình thực thi của một hệ thống sử dụng Docker



Như trong hình vẽ, một hệ thống Docker được thực thi với 3 bước chính :

*Build -> Push -> Pull,Run*

* Build: Đầu tiên tạo một dockerfile, trong dockerfile này chính là code của chúng ta. Dockerfile này sẽ được Build tại một máy tính đã cài đặt Docker Engine. Sau khi build ta sẽ có được Container, trong Container này chứa ứng dụng kèm bộ thư viện của chúng ta.
* **Push:** Sau khi có được Container, chúng ta thực hiện push Container này lên cloud và lưu tại đó.
* **Pull, Run:** Nếu một máy tính khác muốn sử dụng Container chúng ta thì bắt buộc máy phải thực hiện việc Pull container này về máy, tất nhiên máy này cũng phải cài Docker Engine. Sau đó thực hiện Run Container này.

*Khi nào sử dụng Docker?*

* Triển khai kiến trúc [Microservices](https://topdev.vn/blog/microserices-la-gi/).
* Khi xây dựng ứng dụng và cần scale một cách linh hoạt.
* Khi bạn muốn không tốn khá nhiều thời gian để config máy local và server cùng một môi trường để chạy được ứng dụng. Bạn chỉ cần build 1 lần chạy ở nhiều nơi mà thôi.
* Sản phẩm của công ty bạn cần một cách tiếp cận mới về xây dựng, đẩy lên server, thực thi ứng dụng một cách nhanh chóng dễ dàng.

## Angular

### Khái niệm Angular là gì?

Angular được xe là một open source (mã nguồn mở) hay frameworks miễn phí chuyên dụng cho công việc thiết kế web. Angular được phát triển từ những năm 2009 và được duy trì bởi Google. Frameworks này được xem là frameworks front end mạnh mẽ nhất chuyên dụng bởi các lập trình viên cắt HTML cao cấp.

Angular được ứng dụng rộng rãi với mục đích xây dựng project Single Page Application (SPA). Hiện tại, Version stable của Angular là Angular 9 (released on February 7, 2020) với TypeScript 3.6 và 3.7.

Trên thực tế, Angular sở hữu hai phiên bản hoàn toàn khác nhau như sau:

**Phiên bản 1:** Phiên bản này có tên chính thức là AngularJS đến hiện nay nó vẫn đang được phát triển hoàn thiện bởi các chuyên gia. Vào năm 2018, thì phiên bản 1.7.3 đã được nhà sản xuất phát hành. Đây là phiên bản được hoàn thành thực sự dựa trên Javascript bởi những lập trình viên viết mã theo ngôn ngữ Javascript thì nó thường dễ làm và dễ tìm kiếm Tuy nhiên, hiệu năng khi so sánh với phiên bản ReactJS thì vẫn còn nhiều công ty xây dựng phải cân nhắc sử dụng Angular với phiên bản khác 1.

**Phiên bản 2:**Phiên bản này được gọi với cái tên chính thức là Angular. Nó sở hữu tư duy thay đổi hoàn toàn với phiên bản thứ 1 và mang lại tham vọng giúp cho Angular có thể đánh bại ReactJS. AngularJS được xem là một trong những framework sở hữu cấu  trúc ứng dụng dạng động và cho phép bạn có thể sử dụng HTML như là ngôn ngữ mẫu cũng như mở việc mở rộng cú pháp của HTML. Nhờ vậy, việc diễn đạt các thành phần ứng dụng của người lập trình sẽ trở nên súc tích và rõ ràng hơn rất nhiều.

### Đôi nét về Lịch sử phát triển Angular

AngularJS được phát triển từ năm 2009 bởi Misko Hevery cùng 1 người bạn khác là Adam Abrons. Nó được xem là một dự án riêng cho đến khi Misko Hevery tham gia vào dự án Google Feedback dưới tư cách lập trình viên bán thời gian. Khi đó, Misko Hevery và 2 lập trình viên khác đã viết lên 17.000 các dòng mã khác dành cho dự án Google Feedback trong khoảng 6 tháng. Số lượng mã ngày càng nhiều và việc cần phải sửa lỗi kiểm soát phát sinh.

Ông đã mạnh dạn đánh cược với quản lý là có thể viết lại toàn bộ mã nhờ sử dụng dự án GetAngular của mình trong 2 tuần. Mặc dù đã thua cuộc nhưng mã dự án đã giảm nhanh từ 17.000 dòng xuống còn 1.500 dòng. Nhờ  sự thành công này mà AngularJS đã được phát triển nhân rộng và tăng tốc nhanh từ đó.

### Ưu điểm và nhược điểm của AngularJS :

* AngularJS được đánh giá là giải pháp dành cho các Single Page Application làm việc dễ dàng.
* Code Front end thường rất thân thiện nhờ khả năng Binding data lên trên các nền tảng HTML để có thể tạo ra mọi thao tác rất tuyệt.
* Bạn có thể dễ dàng Unit test.
* Có thể tái sử dụng component dễ dàng hơn.
* Hỗ trợ cho các lập trình viên có thể viết code được ít hơn với nhiều chức năng hơn.
* Bạn có thể chạy AngularJS trên nhiều loại trình duyệt khác nhau lẫn PC hoặc mobile.
* Không an toàn: Thông thường, bản chất của AngularJS là một trong những các Front End, mà front end này thường vốn không thể bảo mật bằng [Back End](https://blog.itnavi.com.vn/nhung-ky-nang-quan-trong-de-tro-thanh-mot-backend-developer/). Chính vì vậy, khi sử dụng API thì bạn cần xây dựng cho một hệ thống kiểm tra dữ liệu sao cho việc trả về được tốt nhất.
* Với một số trình duyệt sở hữu tính năng Disable Javascript nên có nghĩa là website sẽ không hoàn toàn có thể sử dụng được dựa trên những trình duyệt đó nữa.

### Sử dụng Angular sẽ làm được gì?

Sau khi hiểu rõ định nghĩa **Angular là gì** thì dưới đây là những điều bạn có thể áp dụng Angular để có thể dần làm quen với chúng.

* CRUD Web Apps: đây là một trong những tính năng mà chính người sử dụng Angular ưu tiên sử dụng nhiều nhất.
* Mobile Apps: Lập trình viên có thể sử dụng Angular kết hợp cùng Phonegap để tạo ra một loại Mobile App  bạn có thể tạo một Mobile Web App.
* CSS3 Animations.
* Chrome Extensions: Đây là cách đơn giản nhất giúp lập trình viên tạo ra Chrome Extension lả việc sử dụng Yeoman- một Chrome Extension Generator.
* Testable JS Apps.
* Firebase Powered Apps.

### Các đặc trưng cơ bản của Angular là gì?

* Được sử dụng để có thể phát triển dựa trên JavaScript.
* Angular có khả năng tạo các ứng dụng client-side dựa trên mô hình MVC.
* Angular sở hữu khả năng tương thích cao có thể tự động xử lý dễ dàng các mã Javascript sao cho phù hợp với các trình duyệt nhất.
* Khi có mã nguồn mở và miễn phí thì nó sẽ được sử dụng rộng rãi hơn.

Thông thường, kiến trúc của một ứng dụng Angular là dựa trên những ý tưởng liên quan đến Components. Mỗi một ứng dụng Angular thường bắt đầu với những  level trên cùng tên gọi là Root Component.

### Cách thức hoạt động của Angular

Sau khi AngularJS được nhúng vào trang thì nó sẽ hiển thị lên cho việc phân tích các mã lệnh [HTML](https://blog.itnavi.com.vn/html-la-gi/). Mã lệnh HTML này sẽ có thẻ  với thuộc tính ng-app=””. Khi đó, thuộc tính này sẽ được sử dụng để bắt đầu cho việc khởi tạo ứng dụng AngularJS. Thẻ tiếp theo <input> có thuộc tính ng-model=”name”  giúp tạo ra biến name bên trong ứng dụng AngularJS trên.

Từ đó, giá trị của biến luôn bằng với giá trị của trường <input>. cuối cùng của thẻ <p> thứ 2 với thuộc tính là <ng-bind=”name”>. Nó sẽ được sử dụng mỗi khi ứng dụng có thể phát hiện ra được những thay đổi của giá trị bên trong biến name và nó sẽ gắn giá trị này trở thành nội dung HTML rồi đặt bên trong thẻ <p> thứ 2 này.

### Giới thiệu tính năng cơ bản của AngularJS

* Controller : xử lý dữ liệu cho đối tượng $scope, từ đây bên views sẽ sử dụng các dữ liệu trong scope để hiển thị ra tương ứng.
* Data-binding : tự động đồng bộ dữ liệu giữa model và view
* Service : Nó được xem là singleton object có khả năng khởi tạo 1 lần duy nhất dành riêng cho mỗi ứng dụng và nó cung cấp các phương án dữ liệu có sẵn như: ($http,  $sce,  $rootElement, $controller, $document,  $httpBackend, $compile, $parse, $rootScope …..)
* Scope : là một trong những đối tượng có nhiệm vụ giao tiếp giữa controller và view của các ứng dụng.
* Filter : Việc lọc các tập hợp con từ bên trong  item ở các mảng và trả nhanh về các mảng mới.
* Directive : được sử dụng để tạo ra các thẻ HTML riêng nhằm mục đích phục vụ một số mục đích  riêng. AngularJS thường có những directive  sở hữu sẵn như ngBind, ngModel…
* Temple : Là một thành phần của view có khả năng hiển thị thông tin từ controller.
* Routing : là sự chuyển đổi giữa các action trong controller, qua lại ngay giữa các view.
* MVC & MVVM : Là mô hình thiết kế nhằm mục đích phân chia các ứng dụng có nhiều thành nhiều phần khác nhau (nó được gọi là Model, View và Controller) và mỗi phần thường sẽ sở hữu một nhiệm vụ nhất định. AngularJS thường không triển khai MVC dựa theo cách truyền thống mà chủ yếu gắn liền với Model-View-ViewModel hơn.
* Deep link : Liên kết sâu này cho phép lập trình viên mã hóa các trạng thái của ứng dụng bên trong URL để có thể bookmark với nhiều công cụ tìm kiếm khác. Hầu hết, các ứng dụng này đều có thể được phục hồi lại từ những địa chỉ URL với cùng trạng thái.
* Dependency Injection: AngularJS có sẵn dependency injection hỗ trợ  bạn tạo ra các ứng dụng có tiềm năng phát triển, dễ hiểu và kiểm tra.