TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

****

BÁO CÁO THỰC TẬP TỐT NGHIỆP

KHOA HỌC MÁY TÍNH

Giảng viên hướng dẫn: Ths. Hồ Đắc Quán

Sinh viên thực hiện : Nguyễn Vũ Khánh Huy

Mã sinh viên: 16025591

Lớp: DHKHMT12A

TP. Hồ Chí Minh 2020

LỜI CẢM ƠN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập – Tự do – Hạnh phúc

NHẬN XÉT CỦA ĐƠN VỊ THỰC TẬP

Họ và tên sinh viên : Nguyễn Vũ Khánh Huy

**Mã sinh viên : ** 16025591

1. Nhận xét chung :

.....

.....

.....

.....

.....

.....

.....

.....

.....

Ngày ..... tháng ..... năm .....

NGƯỜI HƯỚNG DẪN

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Ngày ..... tháng ..... năm .....

GIẢNG VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

Table of content

1. Chương 1: GIỚI THIỆU CHUNG ĐƠN VỊ THỰC TẬP

1.1 Thông tin về đơn vị thực tập

- Tên công ty: Công ty cổ phần VNG, Zalo.Ads

- VNG là công ty công nghệ Việt Nam thành lập năm 2004 với 4 mảng sản phẩm chính: Trò chơi trực tuyến, nền tảng kết nối, thanh toán điện tử và dịch vụ đám mây.
- Sau 15 năm hoạt động, VNG là doanh nghiệp start-up kỳ lân (unicorn) duy nhất tại Việt Nam khi được định giá trên 1 tỷ USD (World start-up report).

- Zalo là ứng dụng tin nhắn và gọi điện miễn phí hoạt động trên nền tảng di động và máy tính, được sử dụng tại các nước Việt Nam, Hoa Kỳ, Myanmar, Nhật Bản, Đài Loan, Hàn Quốc, Cộng hòa Séc, Nga.

- Website: vng.com.vn.
- Appstore: Zalo.

- Địa chỉ:

Trụ sở chính: 182 Lê Đại Hành - Phường 15 - Quận 11 TP.HCM
VNG Campus: VNG Campus Tân Thuận Đông Quận 7 TP.HCM.

1.2 Thông tin về vị trí sinh viên thực tập

Sinh viên năm 3,4 nghành khoa học máy tính, kỹ thuật phần mềm.

- ✓ Tư duy về ngôn ngữ lập trình Java tốt.
- ✓ Có kiến thức về OOP và xu hướng thiết kế.
- ✓ Kiến thức cơ bản HTML, CSS, Javascript.
- ✓ Khả năng giải quyết vấn đề tốt.
- ✓ Sử dụng thành thạo cấu trúc dữ liệu và giải thuật.

Tham gia vào phát triển các dự án của team cho các sản phẩm của Adtima và Zalo media.

2. Chương 2: Vấn đề sinh viên tham gia giải quyết tại đơn vị thực tập.

2.1 Tóm tắt vấn đề mà sinh viên tham gia giải quyết/thực hiện tại Đơn vị/Doanh nghiệp thực tập

- Sinh viên được hướng dẫn về quá trình làm việc của team, review lại các kiến thức cơ bản về lập trình hướng đối tượng, Java, lập trình hướng hàm, Scala, hệ cơ sở dữ liệu MySQL, RabbitMQ, Restful API, PostMan cho backend và công nghệ Angular và các công cụ, kiến liên quan để phục vụ cho phát triển phần mềm.
- Tham gia cuộc thi ZA challenge làm các bài test IQ.
- Tham gia dự án của team, service,tool quảng cáo của Zalo.

2.2 Tiến độ công việc thực hiện

Thời gian	Nội dung công việc theo tuần
Tuần 1	Nhận và cài đặt cho thiết bị cá nhân được cấp. 1. Cài đặt Ubuntu cho pc. 2. Set mạng ảo của công ty cho pc. 3. Tạo và sử dụng tài khoản zalo gitlab của công ty. Tìm hiểu quá trình hình thành và hoạt động của team. 1. Lợi và hại của Java, Scala. 2. Nguyên do sử dụng MySQL. 3. Sự khác biệt của SQL và NoSQL. 4. Mục đích và hướng đi các sản phẩm của team.
Tuần 2	Cài đặt môi trường 1. MySQL. 2. RabbitMQ. 3. Java jdk 8 4. Scala 2.12.10 5. Sbt 1.3.6 Sử dụng PostMan kiểm thử trên hệ thống BackEnd của dự án. 1. OAuth, Secuirty. 2. Restful Api
Tuần 3	Tìm hiểu và làm quen công nghệ Angular 1. Angular cơ bản. 2. Giới thiệu về module và component angular. 3. Databinding. 4. Angular cli.

	5. Webstorm. Tổng kết và review assignment cuối tuần.
Tuần 4	Tiếp tục tìm hiểu về Angular 1. Component and Databinding deep dive. 2. Property & event binding. 3. Life cycle hook and template access 4. Thực hành trên life cyle hook của angular Tổng kết và review demo cuối tuần.
Tuần 5	Tiếp tục tìm hiểu về Angular và dự án ARC(Ad Review Central) của team. 1. Course Project - Component & Data binding. 2. Directives Deep Dives. 3. Test ARC api trên Postman 4. Thao tác code Angular trên api của dự án ARC Tổng kết và review demo cuối tuần và trao đổi với team về dự án.
Tuần 6	Tìm hiểu sâu hơn về Angular và dự án ARC. 1. Services & Dependency injection. 2. Changing Pages with routing. 3. Observables. 4. Áp dụng công nghệ đã học vào arc. Tổng kết và review demo cuối tuần và trao đổi với team về dự án.
Tuần 7	1. Handling forms in Angular Apps. 2. Using Pipes to transform output. 3. Making http requests. 4. Authentication & route protection in angular. Tổng kết và review demo cuối tuần và trao đổi với team về dự án.
Tuần 8	Tìm hiểu hệ thống gắn tag quảng cáo của arc, ôn tập các kiến thức toán học. 1. Ma trận. 2. Xác suất. 3. Hoán vị. 4. Tổ hợp. 5. Biến số ngẫu nhiên và hàm phân phối Tiếp tục tham gia vào dự án arc, trao đổi với team và review demo cuối tuần.
Tuần 9	Tìm hiểu các hàm phân phối xác suất. Viết phần mềm mô phỏng cho từng hàm phân phối. 1. Hàm phân phối nhị phân. 2. Hàm mật độ nhị phân. 3. Hàm nhị phân tích lũy. 4. Hàm phân phối Poision. Tiếp tục tham gia vào dự án arc, trao đổi với team và review demo cuối tuần.
Tuần 10	Tìm hiểu các hàm phân phối xác suất. Viết phần mềm mô phỏng cho từng hàm phân phối. 1. Hàm phân phối chuẩn.

	2. Hàm mật độ phân phối chuẩn. 3. Hàm xác suất chuẩn tích lũy. 4. Hàm phân phối chuẩn hóa. Tiếp tục tham gia vào dự án arc, trao đổi với team và review demo cuối tuần.
Tuần 11	Hoàn thành bản demo gắn tag tự động dựa vào Naive Bayes Classifier. Tìm hiểu và sử dụng công nghệ Django và Angular cho backend và front end của bản demo.
Tuần 12	Review lại quá trình làm việc làm tại công ty, đánh giá các điểm yếu và mạnh của bản thân cũng như các phần công việc chưa hoàn thành.

2.3 Tóm tắt vấn đề mà sinh viên tham gia giải quyết/thực hiện tại Đơn vị/Doanh nghiệp thực tập

1.Tuần 01

1.1 Giới thiệu công nghệ.

1.1.1 Hệ điều hành Ubutu.

- Ubuntu là một hệ điều hành mã nguồn mở dựa trên kiến trúc của hệ điều hành Debian cũng chính là một nhánh khác của các phiên bản hệ điều hành của Linux. Ubuntu có mặt trên desktop, server, và trong các thiết bị iot, robot.
- Ubuntu trong tiếng Zulu có nghĩa là “tình người” mô tả triết lý của ubuntu “tôi được là chính mình là nhờ mọi người”. Logo ubuntu mô tả ba người nắm tay nhau và tạo thành một vòng tròn.
- Ubuntu có được thành công hơn các phiên bản Linux khác là vì được hỗ trợ bởi công ty Canocial. Khoảng những năm 2004, các phiên bản của Linux không hề thân thiện người dùng và được đánh giá là rất tệ khi không có được chăm sóc chuyên nghiệp từ cộng đồng. Tầm nhìn của Canocical về tương lai khi linux có được những người dùng trung thành cũng như là những người dùng mới khi và chỉ khi có một hệ điều hành được tạo ra và thân thiện với người dùng. Ubuntu được đầu tư về mặt quảng cáo, và công ty Canocial thu được các khoản lợi nhuận từ những công ty về dịch vụ điện toán và không thu phí người dùng. Điều đó tạo nên sự thành công của Ubuntu.

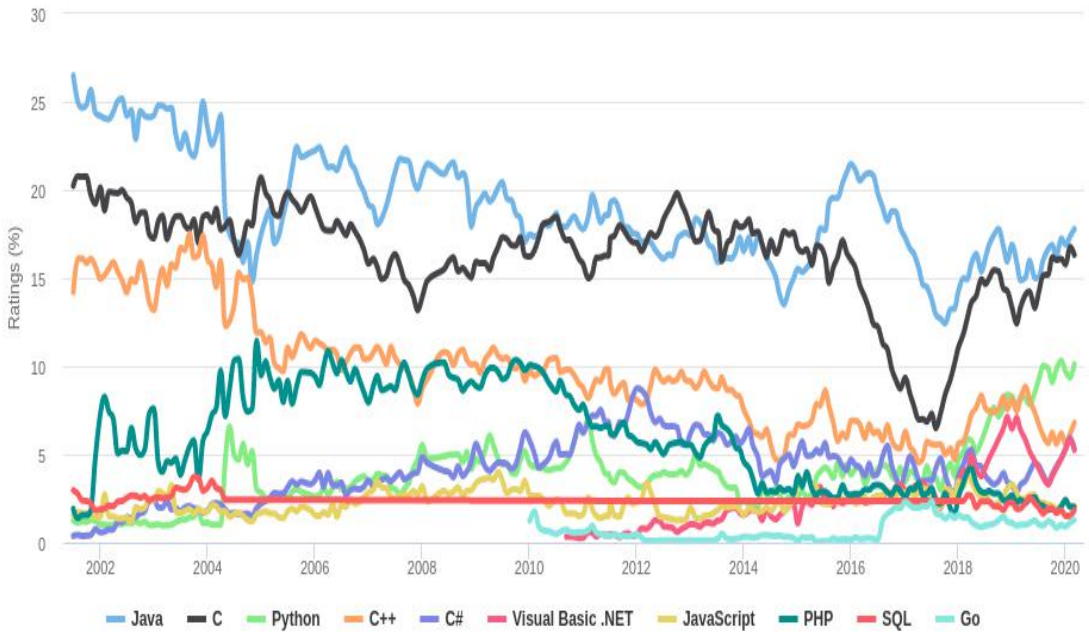
1.1.2 Ngôn ngữ Java và Object Oriented Programing.

- Ngôn ngữ Java được phát triển bởi James Gosling. Ông được biết đến là cha đẻ của Java vào năm 1995 nhưng sự thật là ông cùng các đồng nghiệp đã bắt đầu dự án từ đầu những năm 90. Java được tạo ra dựa trên các nền tảng cần thiết của ngôn ngữ lập trình là đơn giản, mạnh mẽ, dễ thay đổi, độc lập, bảo mật, hiệu năng cao, đa luồng, không phụ thuộc hệ điều hành, hướng đối tượng, linh hoạt. Java được sử dụng ở hầu hết các lĩnh vực trong nghành công nghệ thông tin.

Version	Release date	End of Free Public Updates[6][7]	Extended Support Until
JDK Beta	1995	?	?
JDK 1.0	January 1996	?	?
JDK 1.1	February 1997	?	?
J2SE 1.2	December 1998	?	?
J2SE 1.3	May 2000	?	?
J2SE 1.4	February 2002	October 2008	February 2013
J2SE 5.0	September 2004	November 2009	April 2015
Java SE 6	December 2006	April 2013	December 2018
Java SE 7	July 2011	April 2015	July 2022
Java SE 8 (LTS)	March 2014	**January 2019 for Oracle (commercial)** December 2020 for Oracle (personal use) At least September 2023 for AdoptOpenJDK At least June 2023[8] for Amazon Corretto	December 2030
Java SE 9	September 2017	March 2018 for OpenJDK	N/A
Java SE 10	March 2018	September 2018 for OpenJDK	N/A
Java SE 11 (LTS)	September 2018	At least August 2024[8] for Amazon Corretto September 2022 for AdoptOpenJDK	September 2026
Java SE 12	March 2019	September 2019 for OpenJDK	N/A
Java SE 13	September 2019	March 2020 for OpenJDK	N/A
Java SE 14	March 2020	September 2020 for OpenJDK	N/A
Java SE 15	September 2020	March 2021 for OpenJDK	N/A
Java SE 16	March 2021	September 2021 for OpenJDK	N/A
Java SE 17 (LTS)	September 2021	TBA	TBA

Legend: Old version | Older version, still maintained | **Latest version** | Latest preview version | Future release

Phiên bản JDK theo từng năm.



Bảng xếp hạng các ngôn ngữ lập trình đến năm 2020

Object Oriented Programing(OOP) lập trình hướng đối tượng được tạo ra để đáp ứng các nhu cầu về mô tả dữ liệu thông qua các trường(field). Một Object có nhiều field thì mỗi field sẽ là mỗi thuộc tính riêng biệt. Hướng lập trình của OOP rất đa dạng nhưng nổi tiếng nhất là hướng lập trình hướng đối tượng dựa trên lớp(Class based).

Các ngôn ngữ có hỗ trợ lập trình hướng đối tượng là C++, Java, Python đồng thời ba ngôn ngữ này cũng hỗ trơ các dạng lập trình như imperative language, procedural programingocedural. Bên cạnh đó còn rất nhiều các loại ngôn ngữ khác hỗ trợ lập trình hướng đối tượng như ngôn ngữ C#, PHP, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, MATLAB, and Smalltalk.

Ưu điểm	Nhược điểm
- Một team làm việc đồng thời và độc lập cho các tính năng cùng dự án dễ dàng. - Cấu trúc của class trong OOP giúp nó có thể tái sử dụng. - Ngôn ngữ được một cộng đồng rất lớn hỗ trợ cũng như là có bề dày lịch sử nên sẽ dễ bảo trì nếu có vấn đề xảy ra.	- Hướng đối tượng do cần tạo ra nhiều class nên sử dụng nhiều tài nguyên hơn các dạng lập trình khác. - Trong trường hợp đã mở rộng các tính năng quá nhiều và quản lý không tốt thì một lượng lớn code không được tái sử dụng ảnh hượng đến hiệu năng và gây ra phí tổn thất là lớn. - OOP dễ thiết kế hơn là hiện thực, trong quá trình hiện thực quá trình các tiến trình bị lặp lại mà không được tái sử dụng có khả năng xảy ra là cao.

1.1.3 Ngôn ngữ Scala và Functional Programing.

- Scala được tạo ra để phục vụ cho nhiều hướng phát triển của ngôn ngữ lập trình. Martin Odersky là người phát triển cũng như là cha đẻ của ngôn ngữ scala. Scala không phải là phần mở rộng của ngon ngữ java nhưng giữa hai ngôn ngữ này hoàn toàn trao đổi dữ liệu trong quá trình hoạt động. Trong quá trình complie dữ liệu, scala truyền file về Java bytecode và chạy trên JVM(Java virtual machine).

Scala được thiết kế để phục vụ **lập trình hướng đối tượng (OOP)** và cả **lập trình hướng hàm (FP)**, tất cả mọi giá trị trong scala đều có kiểu object và tất cả mọi hàm trong scala đều trả về một giá trị. Scala được lấy cảm hứng từ chữ scalable nghĩa là có thể mở rộng dễ dàng dựa trên nhu cầu của người dùng.

Martin Ordesky bắt đầu dự án scala từ năm 2001 tại viện Ecole Polytechnique Federale de Lausanne (EPFL). Chính thức công bố ngôn ngữ là vào năm 2004.

Ngôn ngữ scala dần nổi tiếng và được sử dụng trong các phần quan trọng của các công ty công nghệ lớn.

- Twitter thông báo chuyển toàn bộ hệ thống backend từ Ruby về Scala.
- Apple sử dụng scala ở một số team cùng với Java Play Framework.
- Năm 2014 New York Times tuyên bố sử dụng scala, akka, play framework để xây dựng hệ thống quản lý nội dung Blackbeard.
- Google sử dụng scala để xây dựng Firebase và Nest vì cần scale lượng dữ liệu khổng lồ.
- Walmart sử dụng scala cho backend.

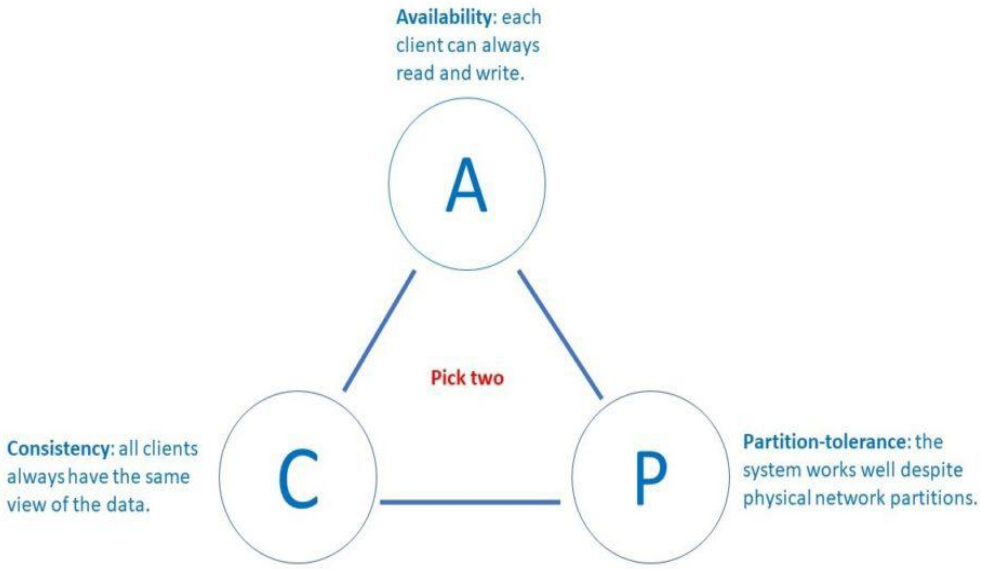
1.1.4 SQL và NoSQL.

Ưu điểm	Nhược điểm
➢ Ngôn ngữ truy vấn mạnh. ➢ Tối ưu hoá cho việc xử lý trên nhiều bảng cùng lúc. ➢ Xử lý một truy vấn số lượng lớn thao tác tốt. ➢ Có nhiều sự lựa chọn, được hỗ trợ bởi trên quy mô doanh nghiệp. ➢ Xử lý nhanh tốc độ query và tìm kiếm dữ liệu. ➢ Đáp ứng nhu cầu về tính sẵn có và tính ổn định của dữ liệu (CP).	➢ Mô hình dữ liệu được định nghĩa trước gây ra sự không linh hoạt. ➢ Một số ngôn ngữ lập trình không hỗ trợ việc chuyển đổi Object thành bảng trong SQL. ➢ Scale theo chiều dọc nghĩa là database chỉ chạy được trên một server, khi scale sẽ chi thêm tiền cho phần cứng. ➢ Tính phân vùng bị hạn chết do database chỉ chạy trên một server, trong trường hợp có sự cố thì việc truy cập database là không thể.

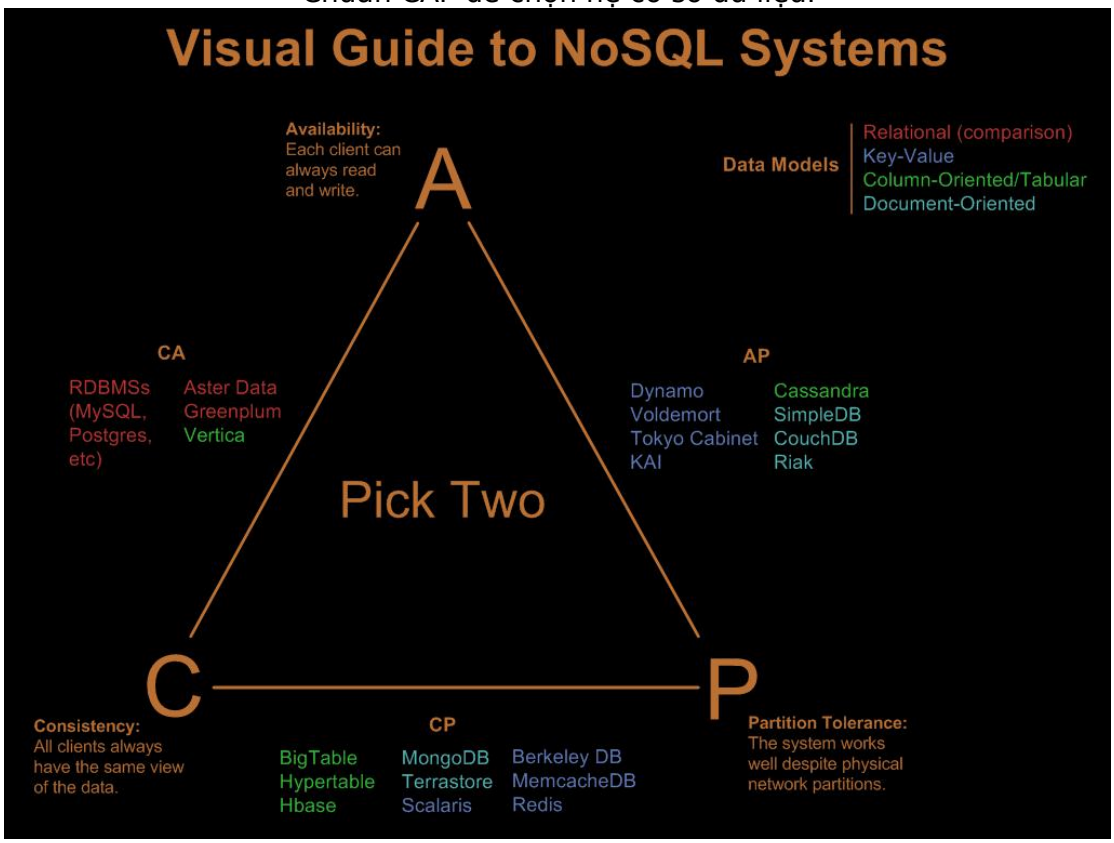
Ưu điểm và Nhược điểm của SQL

Ưu điểm	Nhược điểm
➢ Linh hoạt khi thao tác với mô hình dữ liệu, khi chương trình chạy thì việc thay đổi cấu trúc mô hình sẽ không để lại hậu quả. ➢ Scale theo chiều ngang, nghĩa là có thể triển khai trên nhiều máy tính khác nhau giúp giảm chi phí. ➢ Phục vụ tốt cho lưu trữ số lượng lớn datasets/ objects. ➢ Tính sẵn có và tính phân vùng cao (AP).	➢ Không xác thực được tính toàn vẹn của dữ liệu. ➢ Tìm kiếm số lượng nhiều trên các collection khác nhau chậm. ➢ Thiếu sự hỗ trợ trên quy mô doanh nghiệp. ➢ Hệ thống truy vấn quá đa dạng nhưng không quá mạnh về chất lượng. ➢ Dữ liệu trả về không phải lúc nào cũng là mới nhất.

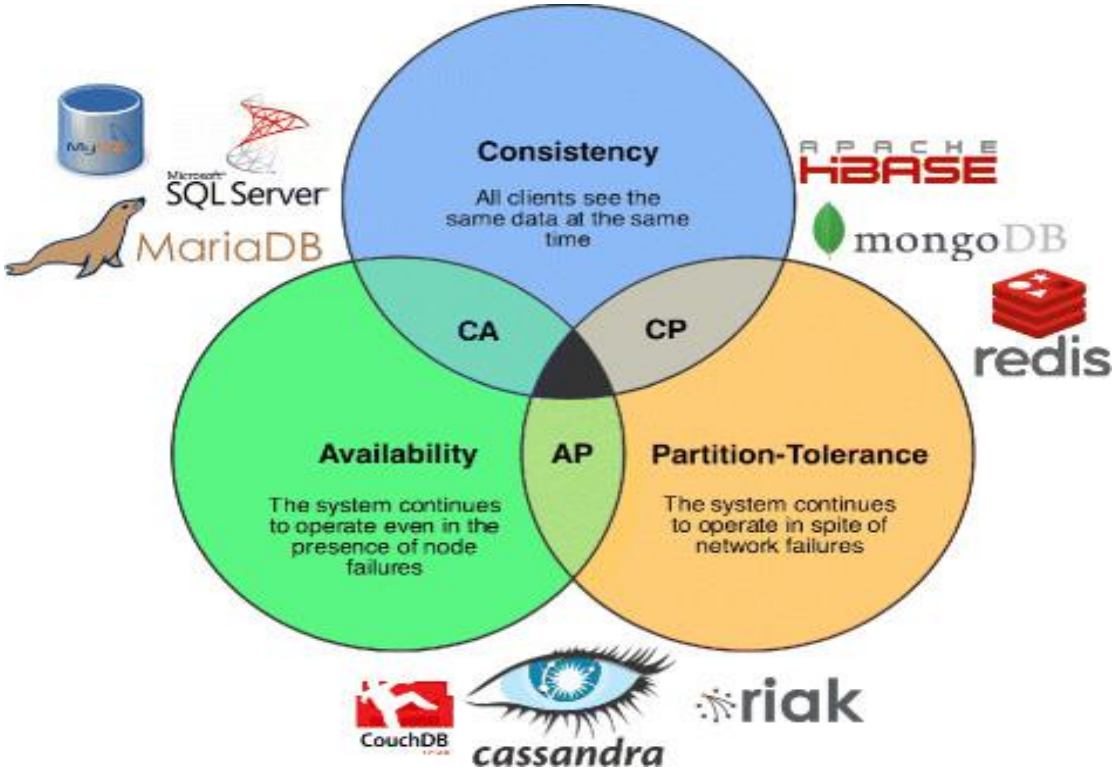
Ưu điểm và nhược điểm của NoSQL.



Chuẩn CAP để chọn hệ cơ sỡ dữ liệu.



Dựa vào **CAP** để chọn hệ cơ sở dữ liệu **NoSQL** phù hợp.



Dựa vào **CAP** để chọn hệ cơ sở dữ liệu **SQL** phù hợp.

1.1.5 Hệ cơ sở dữ liệu MySQL.

Cho đến hiện nay thì MySQL là hệ cơ sở dữ liệu nổi tiếng nhì thế giới chỉ sau Oracle.

Rank Mar 2020	Rank Feb 2020	Rank Mar 2019	DBMS	Database Model	Score Mar 2020	Score Feb 2020
1.	1.	1.	Oracle	Relational, Multi-model	1340.64	-4.11
2.	2.	2.	MySQL	Relational, Multi-model	1259.73	-7.92
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1097.86	+4.11
4.	4.	4.	PostgreSQL	Relational, Multi-model	513.92	+6.98
5.	5.	5.	MongoDB	Document, Multi-model	437.61	+4.28
6.	6.	6.	IBM Db2	Relational, Multi-model	162.56	-2.99
7.	7.	↑ 9.	Elasticsearch	Search engine, Multi-model	149.17	-2.98
8.	8.	8.	Redis	Key-value, Multi-model	147.58	-3.84
9.	9.	↓ 7.	Microsoft Access	Relational	125.14	-2.92
10.	10.	10.	SQLite	Relational	121.95	-1.41
11.	11.	11.	Cassandra	Wide column	120.95	+0.60
12.	12.	↑ 13.	Splunk	Search engine	88.52	-0.26
13.	13.	↓ 12.	MariaDB	Relational, Multi-model	88.35	+1.01
14.	14.	↑ 15.	Hive	Relational	85.38	+1.85
15.	15.	↓ 14.	Teradata	Relational, Multi-model	77.84	+1.03
16.	16.	↑ 21.	Amazon DynamoDB	Multi-model	62.51	+0.38
17.	17.	↓ 16.	Solr	Search engine	55.09	-1.07

Sun Microsystem mua **MySQL** lúc đó chỉ là một mã nguồn mở từ năm 2008, năm 2009 thì **Oracle** mua lại **Sun Microsystem** (bao gồm cả MySQL).

Ưu điểm	Nhược điểm
➢ MySQL là mã nguồn mở. ➢ Các sản phẩm của Oracle đều được	➢ MySQL không có sự trưởng thành như các hệ cơ sở dữ liệu khác.

hỗ trợ tốt trên quy mô cộng đồng hay doanh nghiệp, MySQL cũng là một trong số đó. ➢ MySQL được thiết kế cho web, điện toán đám mây, dữ liệu lớn. ➢ Điểm quan trọng nhất của MySQL là có một đồng lớn những người sử dụng sẽ là nguồn tài nguyên cho việc phát triển hệ cơ sở dữ liệu này.	➢ MySQL là mã nguồn mở (đã từng vì dưới sự điều hành của Oracle thì có nhiều mã nguồn của MySQL không được public). ➢ MySQL phải cạnh tranh với cùng một loại database khác của Oracle là OracleDB. ➢ Cạnh tranh giữa các hệ điều hành khác là lớn khi sự lựa chọn là đến từ Red Hat Enterprise, Fedora, Slackware Linux, OpenSUSE.

2. Tuần 2

2.1 Cài đặt môi trường.

- ✓ MySQL 8.0
- ✓ RabbitMQ 3.8,3
- ✓ Java JDK 8.
- ✓ Scala 2.12.10
- ✓ SBT 1.3.6

2.2 Postman.

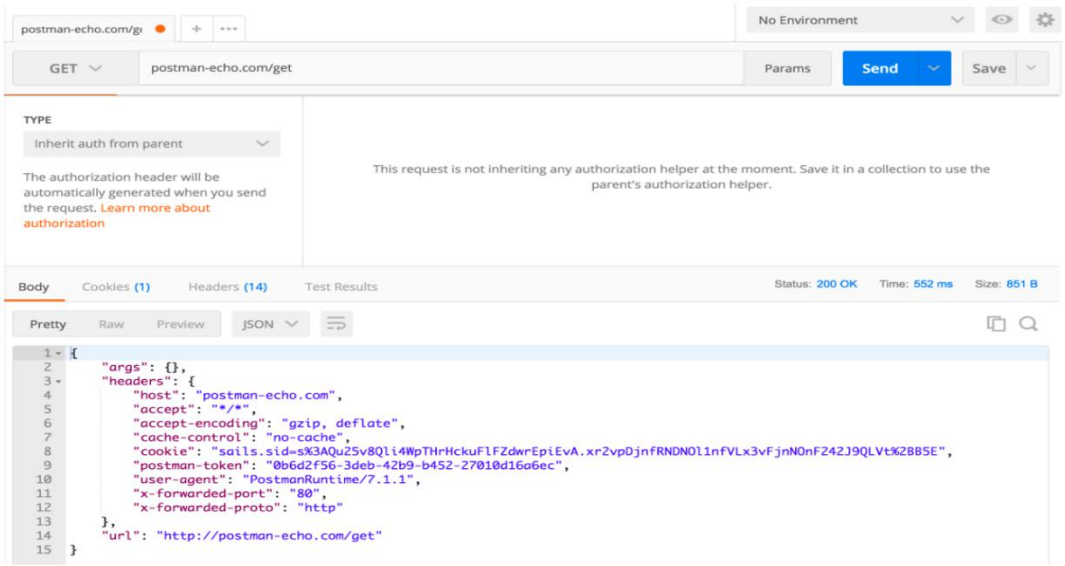
Postman là nền tảng phát triển API. Người dùng sử dụng postman cho ba mục đích chính là **thiết kê, xây dựng, kiểm thử** các API trong quá trình phát triển, hỗ với các thành viên trong team.

API viết tắt cho **Application Interface Programing**, đây chính là phương thức kết nối giữa hệ thống lưu trữ dữ liệu và giao diện sử dụng.

Mỗi API request sử dụng một phương thức HTTP. Những phương thức HTTP được sử dụng nhiều là GET, POST, PATCH, PUT, DELETE.

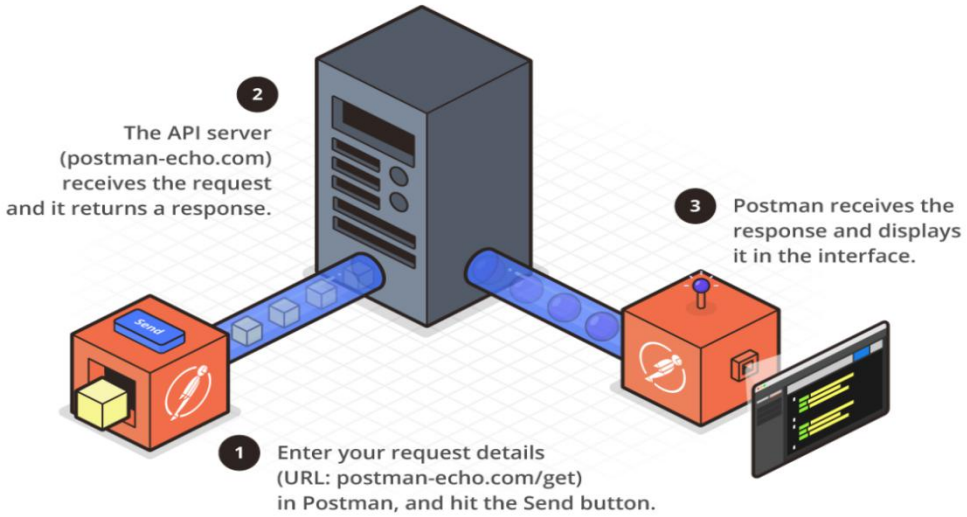
GET: phương thức lấy dữ liệu từ API.
POST: gửi dữ liệu đến API.
PATCH và PUT là hai phương thức cập nhật liệu đã có.
DELETE sẽ xoá dữ liệu đã có trên API.

Nhờ có Postman mà người dùng có thể kiểm thử API ngay lập tức mà không cần đến câu lệnh trên **terminal** hay là phải viết code. Toàn bộ quá trình kiểm thử đều có sẵn trên giao diện của Postman.



Tiến trình được **Postman** xử lý khi người dùng request api trong giao diện của Postman.

Bước 1: người dùng yêu cầu API từ URL được kích hoạt bằng nút SEND.
Bước 2: server khi nhận được request thì trả về một response.
Bước 3: Postman nhận response và hiện ra trên giao diện hiển thị.

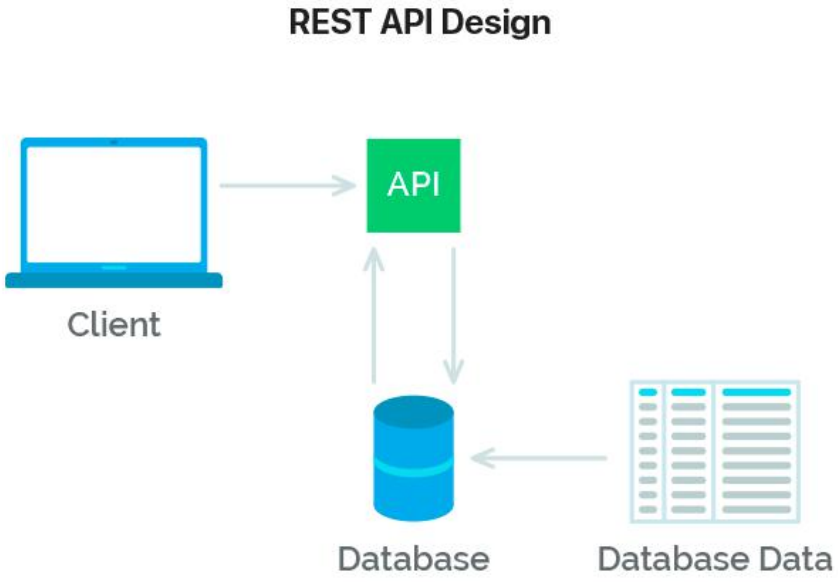


2.3 RESTful API.

REST viết tắt cho **RE**presentational **S**tate **T**ransfer là một kiểu xây dựng cho hệ thống phân phối dữ liệu phân tán được trình bày lần đầu tiên vào năm 2000 dưới cái tên của Roy Fielding.

Một **interface** sẽ được công nhận là **RESTful** khi đảm bảo **6** quy định sau:

- ✓ **Client - Server**: phân tách giữa giao diện người dùng và nơi lưu trữ dữ liệu, điều này làm tăng khả năng mở rộng trên nhiều nền tảng khác nhau.
- ✓ **Stateless**: mỗi request từ máy client đến server phải tồn tại đủ các thông tin cần thiết để xác định được request, không thể lợi dụng các context được lưu trên server. Session state được lưu hoàn toàn trên máy client.
- ✓ **Cacheable**: dữ liệu trả về từ response hoàn toàn có thể được dùng lại ngay trên máy client.
- ✓ **Uniform interface**: chuẩn hoá các interface dựa trên bốn ràng buộc, một là về định danh các tài nguyên, thao tác với tài nguyên thông qua giao diện, thông điệp có khả năng tự mô tả, đa phương tiện.
- ✓ **Layered system**: kiến trúc theo kiểu hệ thống phân lớp cho phép xây dựng ứng dụng theo mô hình có thứ bậc nhiều lớp cùng các ràng buộc. Nghĩa là một tầng sẽ không truy cập nhiều hơn các tầng mà nó tương tác.
- ✓ **Code on demand** (không bắt buộc) : REST cho phép các chức năng của người dùng được mở rộng để tải và chạy code trên applets hoặc là script. Điều này làm đơn giản công việc trên máy client bằng cách giảm các chức năng buộc phải có.

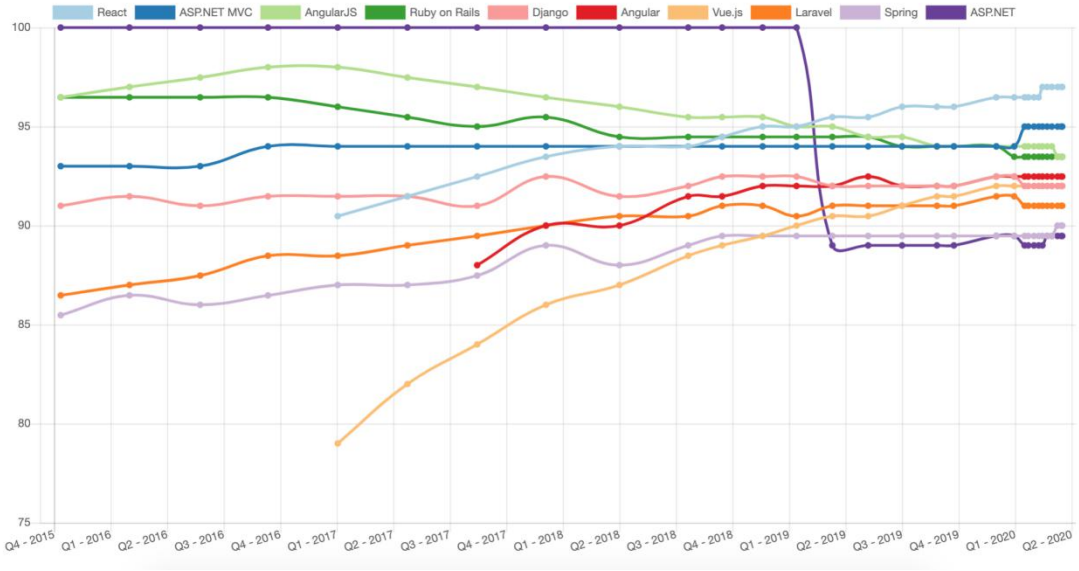


3. Tuần 3.

3.1 Giới thiệu Angular

Angular là một Javascript framework dùng để viết giao diện web (Front-end) được phát triển bởi Google. Hiện nay, Angular được sử dụng bởi nhiều công ty lớn Forbes, General Motors, Upwork, Zalo, ...
Angular nổi tiếng là giải pháp hữu hiệu cho các web app **SPA** (Single Page Application) cùng với các framework nổi tiếng không kém khác là **React** và **Vue.js**. Đã tồn tại từ hơn 10 năm trước, trải qua rất nhiều cái phiên bản thay đổi và cải tiến

Angular giành được sự tin tưởng từ cộng đồng các nhà phát triển web về khả năng hỗ trợ và phát triển công nghệ.



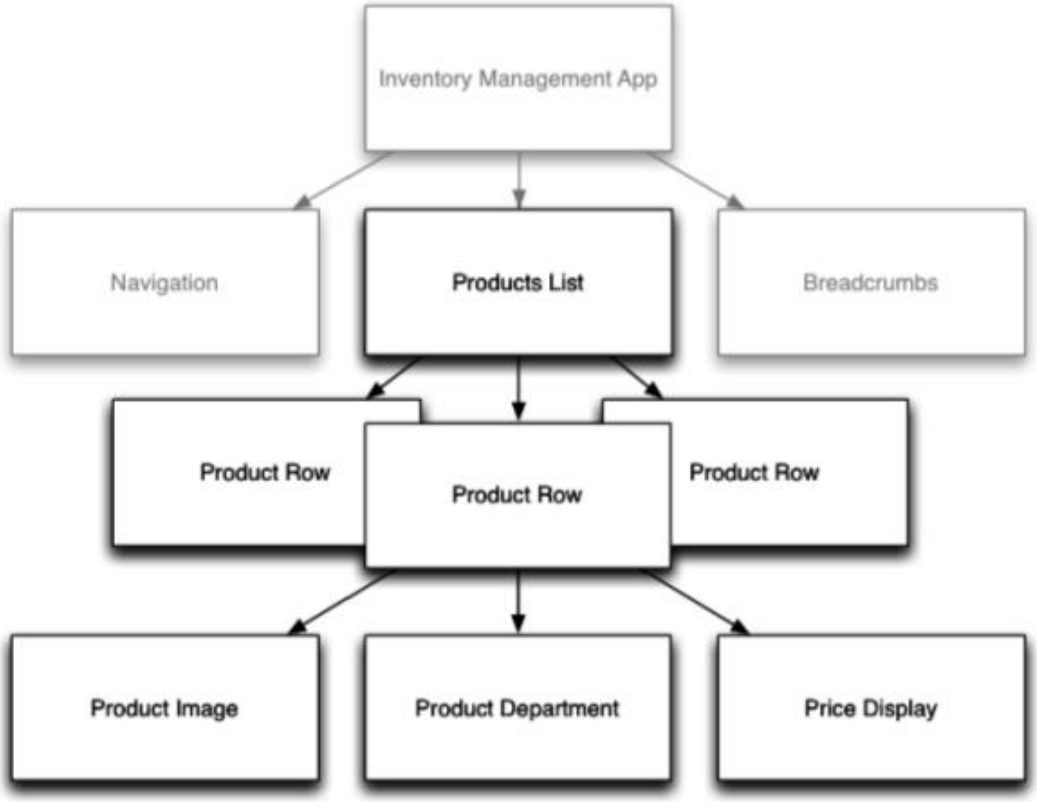
Xếp hạng các loại framework được sử dụng nhiều bởi cộng đồng, nếu chỉ so về framework front-end thì Angular chỉ đứng sau React.

4. Tuần 4.

4.1. Khái niệm về component.

Khi nói đến Ng2 nhiều lập trình viên thường tỏ ra ái ngại hoặc cảm thấy khó khăn để bắt đầu. Nhiều quan điểm được đưa ra, rằng Ng2 là một phiên bản mới của Ng1 nhưng cú pháp lại hoàn toàn khác nhau, gây khó khăn trong quá trình tiếp cận. Một trong những sự khác biệt lớn nhất đó là khái niệm Component, với việc kiến trúc lại hướng component đã làm thay đổi Ng2 hoàn toàn. Nếu đã từng viết Java, C# hoặc một vài ngôn ngữ mạnh mẽ với thiết kế hướng đối tượng(OOP) thì việc hiểu về component trong Ng2 sẽ đơn giản hơn nhiều. Vậy Component trong Ng2 là gì?

- Trong Ng2 mỗi ứng dụng được cấu trúc như một cây component. Mỗi nút bao gồm các thành phần chính là file component và file template.
- Mỗi component có thể chạy như một ứng dụng độc lập, có tính module hóa rất cao, dễ dàng sử dụng mọi nơi.
- Root Component là thành phần cao nhất chứa các thành phần (component) còn lại.
- Các component muốn có thể sử dụng trong ứng dụng phải được khai báo trong NgModule. Ví dụ minh họa component:



Trong ví dụ trên: Products List là Root-component, các component anh em liền kề Product-row và cuối cùng là các child component của Product-row là: Product-image, Product-department, Price-display.

4.2 Vòng đời component.

- Một component có một vòng đời được quản lí bởi chính Angular. Angular tạo ra, render nó, tạo và render các component con của nó, kiểm tra nó khi thuộc tính của giá trị ràng buộc thay đổi, và hủy bỏ nó trước khi xóa trên phần tử DOM.
- Angular đặt ra component lifecycle hooks giúp chúng ta có cái nhìn tổng quan về các thời điểm then chốt và có hành động khi chúng xuất hiện. Bằng việc thêm các hành động vào các component lifecycle hook, ta có thể thực hiện, lặp lại các hành động khi các sự kiện tương ứng trong vòng đời của component xảy ra.

5. Tuần 5.

5.1. Khái niệm directives.

Directives là một đối tượng giúp chúng ta dễ dàng thay đổi một đối tượng khác và cách áp dụng rất đơn giản và linh hoạt. Có 3 loại Directives trong angular: 1.

Component-Directives với template 2. Structural directives-thay đổi cấu trúc DOM bằng việc thêm bớt các phần tử trong DOM 3. Attribute directives-thay đổi giao diện và tương tác của các đối tượng hoặc thay đổi directive khác

- Directive cho component được sử dụng rất phổ biến , có xem tại đây QuickStart
- Structural Directives thay đổi cấu trúc view trong template, ví dụ đơn giản là NgFor và NgIF, có thể xem tại đây https://angular.io/guide/structural-directives

Nó được dùng như một thuộc tính của đối tượng, cho nên khi build thì directive và các thuộc tính thông thường khác được build cùng một lúc dẫn đến dự thay đổi của directive là đồng thời khi render đối tượng đó.

6. Tuần 6

6.1 Khái niệm dependency injection.

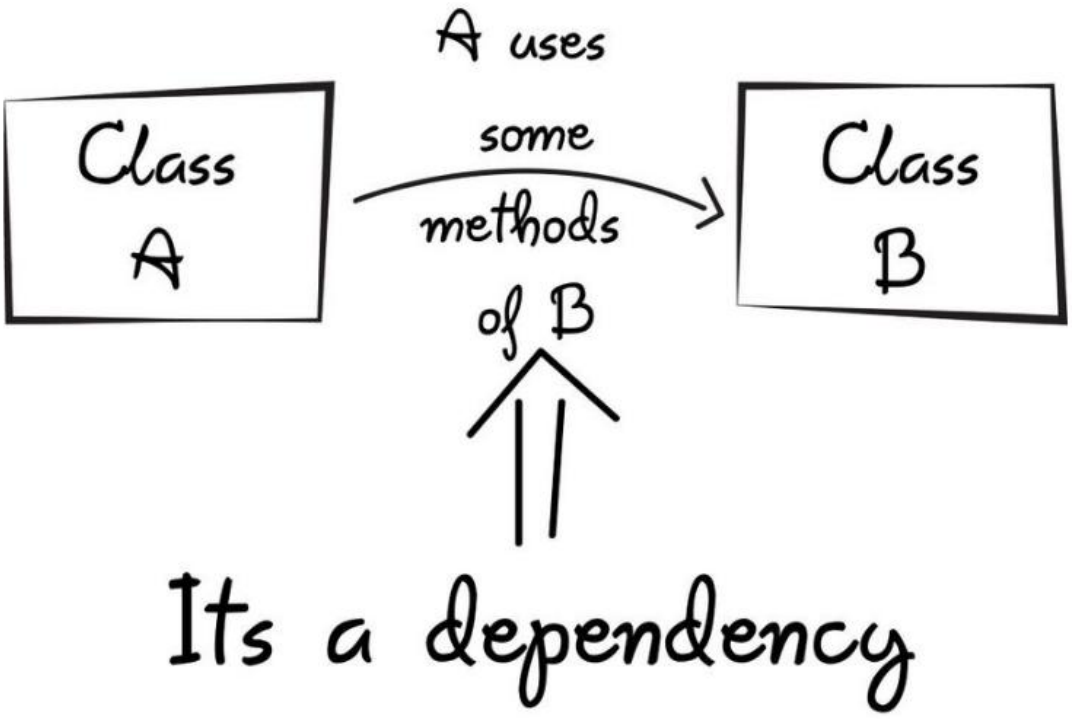
Theo như định nghĩa của Wikipedia

> Dependency injection là một kĩ thuật trong đó một object (hoặc một static method) cung cấp các dependencies của một object khác. Một dependency là một object mà có thể sử dụng (một service). Tuy nhiên định nghĩa trên vẫn khá là khó hiểu, vậy nên hãy cùng tìm hiểu để hiểu rõ hơn về nó nào.

Dependency hay dependent nghĩa là phụ thuộc vào hỗ trợ của một cái gì, việc gì đó. Ví dụ như nếu chúng ta phụ thuộc quá nhiều vào smartphone, thì có thể hiểu là chúng ta đã dependent lên smartphone.

Trước khi nói về dependency injection, hãy hiểu xem dependency trong lập trình nghĩa là gì trước.

Khi mà class A sử dụng một số chức năng của class class B, thì có thể nói là class A có quan hệ phụ thuộc với class B.



Trong java, trước khi ta có thể sử dụng method của class khác, ta phải khởi tạo một object của class đấy (hay A cần phải tạo 1 thực thể của B).

Vậy ta có thể hiểu, việc chuyển giao nhiệm vụ khởi tạo object đó cho một ai khác và trực tiếp sử dụng các dependency đó được gọi là dependency injection.

Về cơ bản có 3 loại dependency injection:

1. **Constructor injection**: các dependency đc cung cấp thông qua constructor của class.
2. **Setter injection**: client tạo ra một setter method để các class khác có thể sử dụng chúng để cấp dependency.
3. **Interface injection**: dependency sẽ cung cấp một hàm injector để inject nó vào bất kì client nào đc truyền vào. Các client phải implement một interface mà có một setter method dành cho việc nhận dependency.

Vậy trách nhiệm của dependency injection là:

1. Tạo ra các object.
2. Biết được class nào cần những object đấy.
3. Cung cấp cho những class đó những object chúng cần.

Bằng cách này, nếu trong tương lai object đó có sự thay đổi thì dependency injection có nhiệm vụ cấp lại những object cần thiết cho class.

Lợi ích của dependency injection.

1. Giúp viết Unit test dễ dàng hơn.
2. Giảm thiểu đc boilerplate code vì việc khởi tạo dependency đc làm bởi một component khác.
3. Mở dụng dự án dễ dàng hơn.
4. Giúp ích trong việc liên kết lỏng (loose coupling) giữa các thành phần trong dự án.

Bất lợi khi dùng dependency injection.

1. Nó khá là phức tạp để học, và nếu dùng quá đà thì có thể dẫn tới một số vấn đề khác.
2. Rất nhiều các lỗi ở compile time có thể bị đẩy sang runtime.
3. Có thể làm ảnh hưởng tới chức năng auto-complete hay Find references của một số IDE

7. Tuần 7.

7.1 Khái niệm OAuth.

Thiết kế một ứng dụng, service hoặc API và tự hỏi nên làm gì để hỗ trợ xác thực người dùng. Ngày nay, chúng ta với tư cách là các nhà phát triển luôn cảnh giác khi bảo vệ các nền tảng và ứng dụng của mình. Sẽ không qua khó khăn nếu chỉ cần kiểm tra username và mật khẩu, tuy nhiên cũng cần xác định những tính năng mà người dùng được phép truy cập tùy vào cấp độ của họ. Vậy thì có thể cần quan tâm đến giao thức authorization (ủy quyền) mà người dùng sẽ tương tác và khả năng "truy cập ủy quyền an toàn" phía máy chủ. Chúng ta sẽ bắt đầu bằng cách định nghĩa một vài thuật ngữ:

- **Authentication (xác thực)**: quá trình hoặc hành hành động xác minh danh tính của một người dùng hoặc tiến trình.
- **Authorization (ủy quyền)**: chức năng chỉ định quyền truy cập hoặc đặc quyền đối với tài nguyên.
- **Resource (tài nguyên)**: một bản ghi, tài liệu, tập tin hoặc thực thể kỹ thuật số khác.

Cần hiểu rõ sự khác biệt giữa authentication và authorization là rất quan trọng.

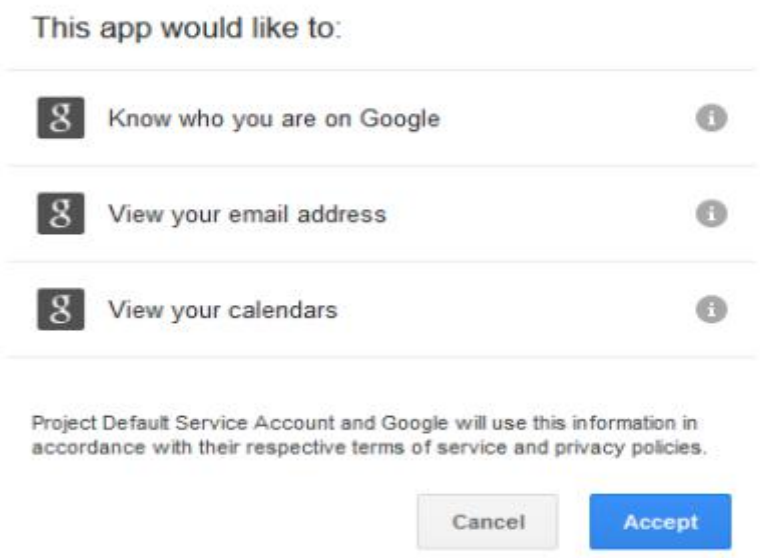
Authentication chỉ đơn giản là cung cấp một bộ thông tin đăng nhập để máy chủ có thể kiểm tra để xác nhận là người nói và thực sự có thông tin xác thực hợp lệ.

Authorization tuân theo quy trình authentication và đảm bảo sau khi xác định thông tin xác thực của người dùng là chính xác thì có thể xác định và thực thi quyền kiểm soát truy cập và các quyền khác đối với resource được lưu.

OAuth là một token-based authorization framework (khung ủy quyền dựa trên token), được thiết kế đặc biệt để hoạt động với HTTP. Bản thân nó không phải là API, service hay package. OAuth thường được sử dụng bởi các công ty như Google, Facebook, Twitter, Amazon, Microsoft, v.v. như một cách để cho phép người dùng của họ cấp cho các trang web hoặc dịch vụ khác quyền truy cập vào thông tin của họ mà không cần cung cấp mật khẩu. Ngoài ra, OAuth cho phép các nhà phát triển bên thứ 3 chỉ định thông tin nào người dùng có thể được yêu cầu chia sẻ để sử dụng dịch vụ và thông tin nào có thể là tùy chọn để chia sẻ. Ví dụ, sau đó người dùng có thể tùy chọn có hay không chia sẻ ngày sinh của họ được liên kết với tài khoản Twitter. Nếu sử dụng bất kỳ trang web hoặc dịch vụ nào trong số này, có lẽ đã thấy một dấu nhắc xác nhận tương tự như những gì tôi đề cập đến.

Có hai phiên bản OAuth: OAuth 1.0a và OAuth 2.0, nhưng những thông số kỹ thuật và cách thực hiện của chúng hoàn toàn khác nhau. May mắn là việc lựa chọn lại rất dễ dàng, OAuth 2.0 là hình thức sử dụng rộng rãi nhất của OAuth. OAuth 2.0 rõ ràng được xây dựng dựa trên những vấn đề gặp phải với 1.0.

Dưới đây là ví dụ từ Google:



Trong ảnh chụp màn hình ở trên, có thể thấy hộp thoại xác nhận với Google sau khi được chuyển hướng từ ứng dụng bên thứ 3 (có thể là một số ứng dụng lịch trong trường hợp này) đến các máy chủ của Google, yêu cầu quyền truy cập vào tài khoản Google của người dùng (tên, email , lịch, vv) thay mặt cho người dùng. Sau khi được chuyển hướng đến Google (hoặc bất kỳ máy chủ OAuth nào) nếu

chưa đăng nhập, sẽ được nhắc làm như vậy, sau khi chấp nhận quyền truy cập được yêu cầu sẽ được chuyển hướng trở lại ứng dụng của bên thứ 3. Sau đó ứng dụng này có thể truy cập các tài nguyên được phép từ tài khoản Google của bằng cách sử dụng *access token*.

7.2 Diving in Deeper

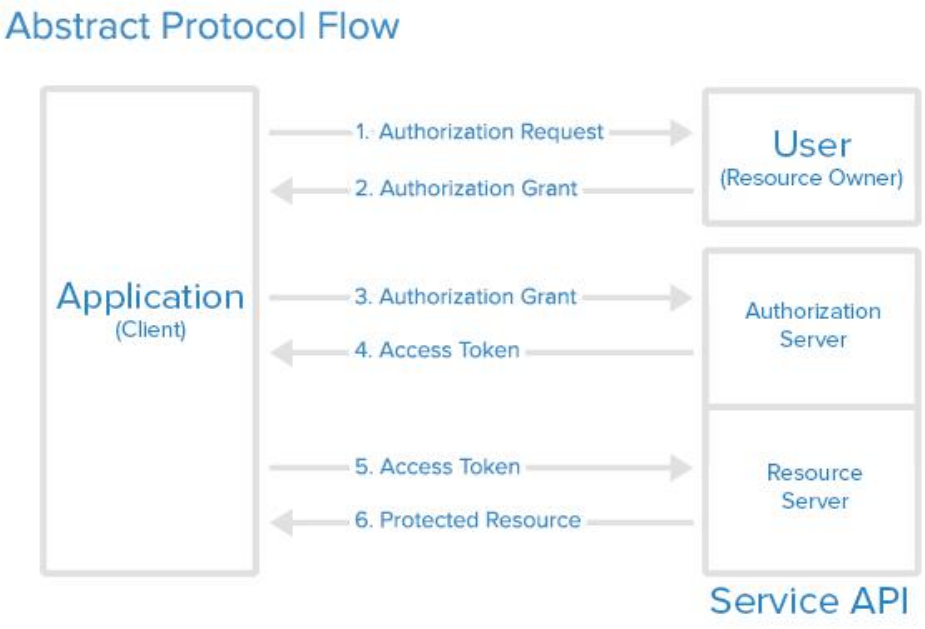
OAuth bao gồm bốn vai trò khác nhau:

Resource Server: REST API là một ví dụ, một máy chủ HTTP nơi người dùng có thể tạo, sửa đổi hoặc xóa các bản ghi, tài liệu hoặc tệp.

Resource Owner: Duy trì quyền sở hữu tài nguyên mà người dùng đã tạo hoặc sửa đổi trên máy chủ và người ủy quyền cho ứng dụng bên thứ 3 truy cập vào tài khoản của họ. Ứng dụng của bên thứ 3 có quyền truy cập hạn chế vào tài khoản của người dùng, dựa trên phạm vi của phạm vi của ủy quyền được cấp.

Client: Ứng dụng bên thứ 3 muốn truy cập vào tài khoản người dùng. Trước khi nó có thể làm như vậy, máy chủ resource/authorization và chủ sở hữu tài nguyên phải ủy quyền cho yêu cầu đó. Mọi khách hàng phải được đăng ký với máy chủ authorization và sẽ được cung cấp cho nó thông tin xác thực duy nhất của riêng mình (client_id và client_secret) để xác thực riêng.

Authorization Server (thường là giống Resource Server): Đôi khi, bạn có thể muốn rút ra khỏi máy chủ authorization từ máy chủ resource và triển khai nó như một phiên bản chuyên dụng, đặc biệt là trong các môi trường phân tán.



1. Ứng dụng bên thứ 3 yêu cầu ủy quyền từ service và người dùng để truy cập tài nguyên do người dùng sở hữu hoặc có khả năng truy cập.
2. Giả sử người dùng cho phép yêu cầu, ứng dụng bên thứ 3 nhận được một authorization grant (chứng nhận được ủy quyền).

3. Sau khi ứng dụng của bên thứ 3 đã được cấp phép, nó có thể yêu cầu access token từ máy chủ authorization bằng cách xuất trình authorization grant hiện có được cung cấp bởi người dùng, cũng như thông tin xác thực của chính họ với tư cách là client.
4. Nếu ứng dụng của bên thứ 3 có thể xác thực và authorization grant là hợp lệ, máy chủ ủy quyền sẽ tạo một access token duy nhất, được liên kết với quyền authorization grant và chủ sở hữu tài nguyên để sử dụng và xác thực trong tương lai.
5. Bây giờ, ứng dụng có thể yêu cầu tài nguyên từ máy chủ tài nguyên sử dụng access token trong các yêu cầu.
6. Với access token hợp lệ, máy chủ tài nguyên sẽ cung cấp các tài nguyên được bảo vệ, các tài nguyên này hiện đã được người dùng và máy chủ ủy quyền hoàn toàn để truy cập.

Ngoài authorization, có một số loại grant khác có thể được sử dụng tùy thuộc vào các trường hợp sử dụng. Nhìn chung, authorization grant là loại OAuth grant được sử dụng phổ biến nhất, đặc biệt là cho public services và API.

8. Tuần 8.

8.1 Các phép toán trong ma trận.

Các phép toán trên ma trận là những tính toán cơ bản khi làm việc với học máy. Trong phần này, tôi không đề cập đầy đủ tất cả các phép toán của nó, mà chỉ đề cập tới các phép toán cơ bản để có thể sài được với học máy cơ bản.

8.2. Nhân ma trận với một vô hướng

Nhân ma trận với một số (vô hướng) là phép nhân số đó với từng phần tử của ma trận.

$$\alpha[A_{ij}]_{mn} = [\alpha.A_{ij}]_{mn}$$

Ví dụ:

$$5\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} = \begin{bmatrix}5 & 10 & 15\\20 & 25 & 30\end{bmatrix}$$

Các tính chất

- Tính giao hoán: $\alpha A = A\alpha$
- Tính kết hợp: $\alpha(\beta A) = (\alpha\beta)A$
- Tính phân phối: $(\alpha + \beta)A = \alpha A + \beta A$

8.3. Cộng 2 ma trận.

Là phép cộng từng phần tử tương ứng của 2 ma trận cùng cấp với nhau.

$$[A_{ij}]_{mn} + [B_{ij}]_{mn} = [A_{ij} + B_{ij}]_{mn}$$

Ví dụ:

$$\begin{bmatrix} 5 & 10 & 15 \\ 20 & 25 & 30 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 6 & 12 & 18 \\ 24 & 29 & 36 \end{bmatrix}$$

Các tính chất

- Tính giao hoán: $A + B = B + A$
- Tính kết hợp: $A + (B + C) = (A + B) + C$
- Tính phân phối: $\alpha(A + B) = \alpha A + \alpha B$

8.4. Nhân 2 ma trận.

Nhân 2 ma trận là phép lấy tổng của tích từng phần tử của hàng tương ứng với cột tương ứng. Phép nhân này chỉ khả thi khi số cột của ma trận bên trái bằng với số hàng của ma trận bên phải. Cho 2 ma trận [A]mp[*A*]*mp* và [B]pn[*B*]*pn*, tích chúng theo thứ tự đó sẽ là một ma trận có số hàng bằng với số hàng của A*A* và số cột bằng với số cột của B*B*: [AB]mn[*AB*]*mn*.

$$C_{ij} = AB_{ij} = \sum_{k=1}^{p} A_{ik}B_{kj} \quad , \forall i = \overline{1,m}; j = \overline{1,n}$$

Ví dụ:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 22 & 28 \\ 49 & 64 \end{bmatrix} \qquad (1)$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \qquad (2)$$

Các tính chất:

- Tính kết hợp: $A(BC) = (AB)C$
- Tính phân phối: $A(B + C) = AB + AC, (A + B)C = AC + BC.$

9. Tuần 9

Tiếp tục tham gia vào dự án , trao đổi với team và review công việc vào đầu tuần.
Sử dụng công nghệ angular cho front end.

10. Tuần 10

Tiếp tục tham gia vào dự án , trao đổi với team và review công việc vào đầu tuần.
Sử dụng công nghệ angular cho front end.

11. Tuần 11

Tiếp tục tham gia vào dự án , trao đổi với team và review công việc vào đầu tuần.
Sử dụng công nghệ angular cho front end.

12. Tuần 12.

Review lại quá trình làm việc làm tại công ty, đánh giá các điểm yếu và mạnh của bản thân cũng như các phần công việc chưa hoàn thành.

3. Chương 3: Nhận xét, đánh giá quá trình thực tập.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....