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.

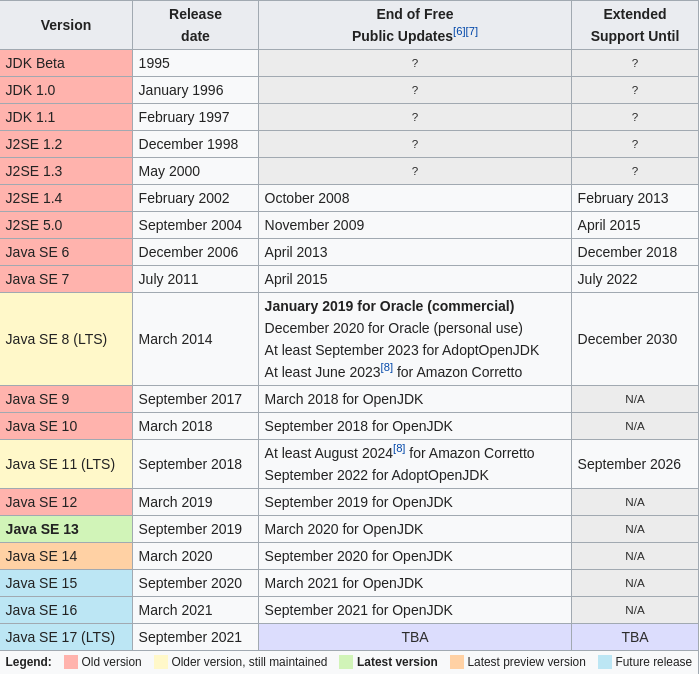
- Ubuntu có được thành công hơn các phiên bản Linux khác là vì được hỗ trợ bởi công ty Canocial. Khoảng những năm 2004, các phiên bản của Linux không hề thân thiện người dùng và được đánh giá là rất tệ khi không có được chăm sóc chuyên nghiệp từ cộng đồng. Tầm nhìn của Canocical về tương lai khi linux 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.

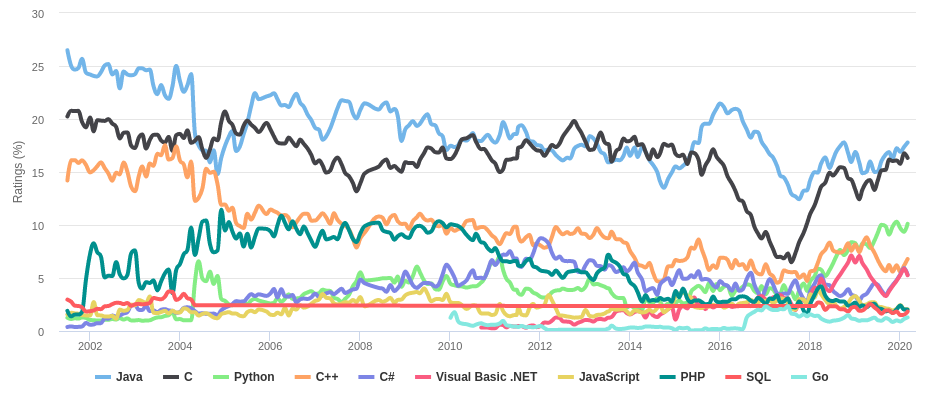


## 1.1.2 Ngôn ngữ Java và Object Oriented Programing.

- 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.



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). Mộ Object có nhiều field 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#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)" \o "C Sharp (programming language)), [PHP](https://en.wikipedia.org/wiki/PHP" \o "PHP), [JavaScript](https://en.wikipedia.org/wiki/JavaScript" \o "JavaScript), [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)" \o "Ruby (programming language)), [Perl](https://en.wikipedia.org/wiki/Perl" \o "Perl), [Object Pascal](https://en.wikipedia.org/wiki/Object_Pascal" \o "Object Pascal),  [Objective-C](https://en.wikipedia.org/wiki/Objective-C" \o "Objective-C), [Dart](https://en.wikipedia.org/wiki/Dart_(programming_language)" \o "Dart (programming language)), [Swift](https://en.wikipedia.org/wiki/Swift_(programming_language)" \o "Swift (programming language)), [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)" \o "Scala (programming language)), [Common Lisp](https://en.wikipedia.org/wiki/Common_Lisp" \o "Common Lisp), [MATLAB](https://en.wikipedia.org/wiki/MATLAB" \o "MATLAB), and [Smalltalk](https://en.wikipedia.org/wiki/Smalltalk" \o "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 lập trình hướng đối tượng và cả lập trình hướng hàm, tất cả mọi giá trị trong scala đều có kiểu object và tất cả mọi hàm trong scala đều trả 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.



## 

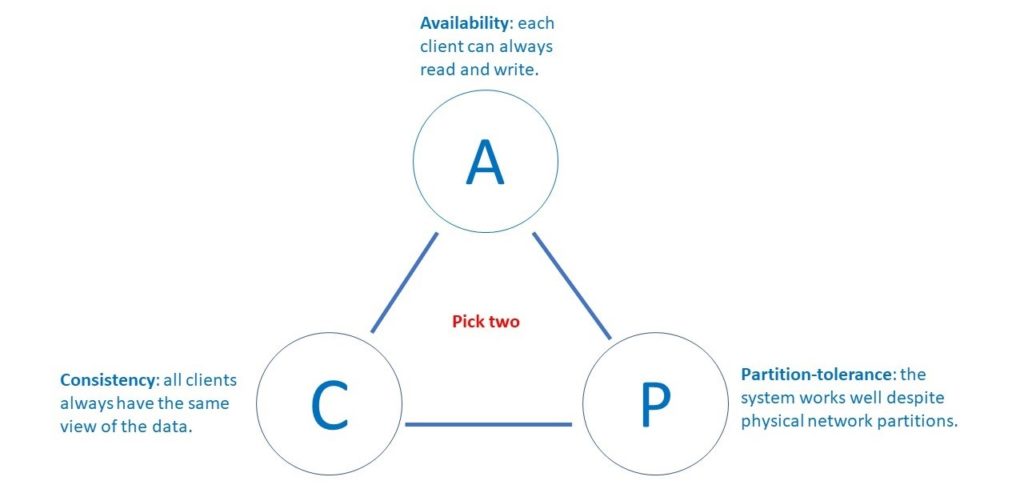
## 1.1.4. SQL và NoSQL.

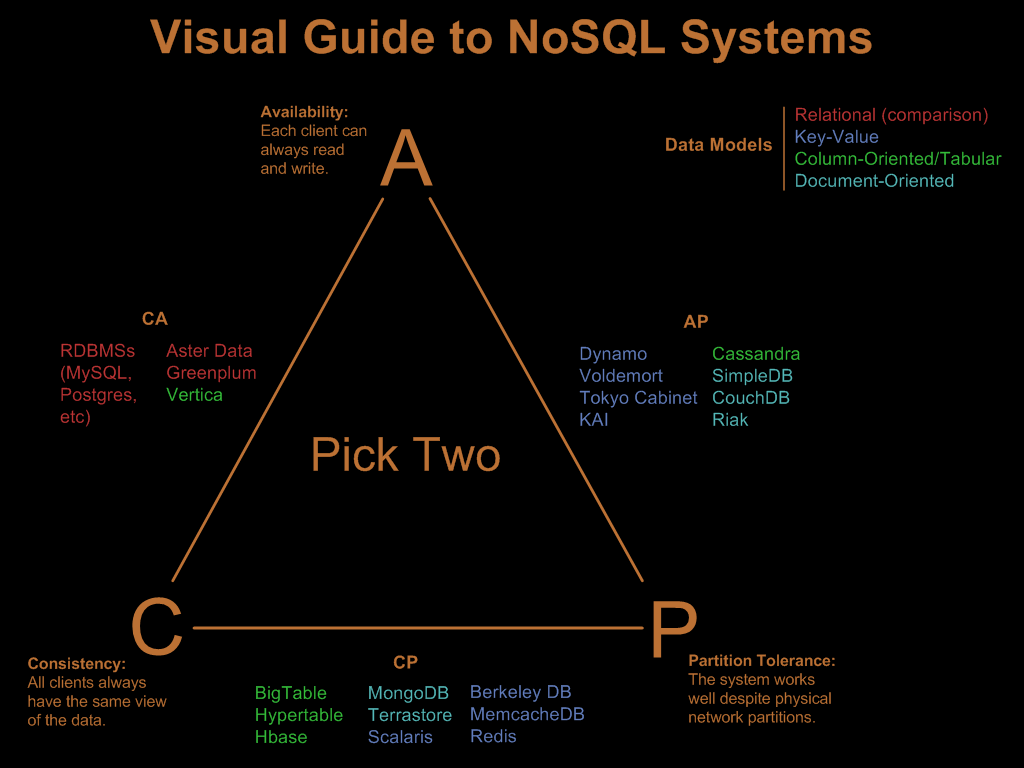
|  |  |
| --- | --- |
| Ưu điểm | Nhược điểm |
| * Ngôn ngữ truy vấn mạnh. * Tối ưu hoá cho việc xử lý trên nhiều bảng cùng lúc. * Xử lý một truy vấn số lượng lớn thao tác tốt. * Có nhiều sự lựa chọn, được hỗ trợ bởi trên quy mô doanh nghiệp. * Xử lý nhanh tốc độ query và tìm kiếm dữ liệu. * Đáp ứng 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

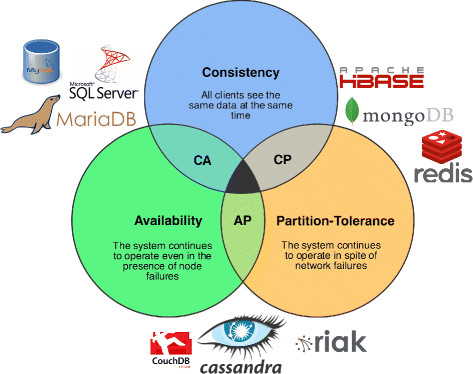
|  |  |
| --- | --- |
| Ưu điểm | Nhược điểm |
| * Linh hoạt khi thao tác với mô hình dữ liệu, khi chương trình chạy thay đổi cấu trúc mô hình sẽ không để lại hậu quả. * Scale theo chiều ngang, nghĩa là có thể triển khai trên nhiều máy tính khác nhau giúp giảm chi phí. * Phục vụ tốt cho lưu trữ số lượng 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.



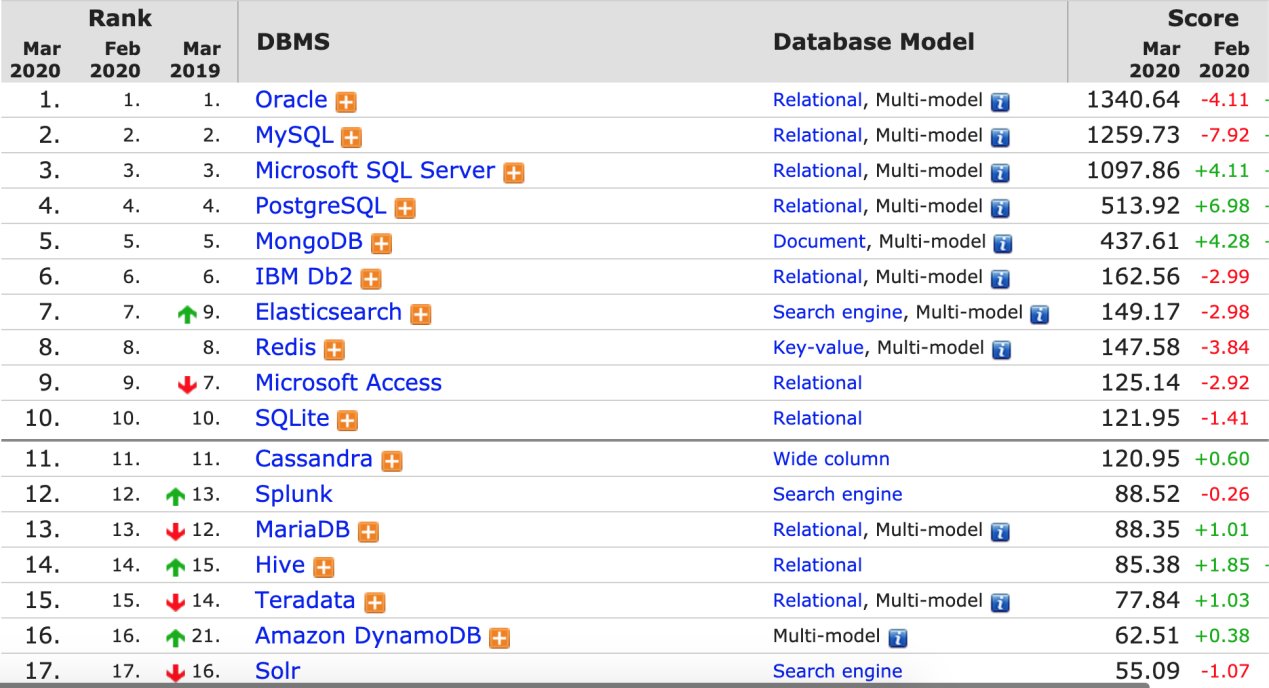
Dựa vào **CAP** để chọn hệ cơ sở dữ liệu **NoSQL** phù hợp.



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.



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 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 không có sự trưởng thành như các hệ cơ sở dữ liệu khác. * 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. |

# 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.

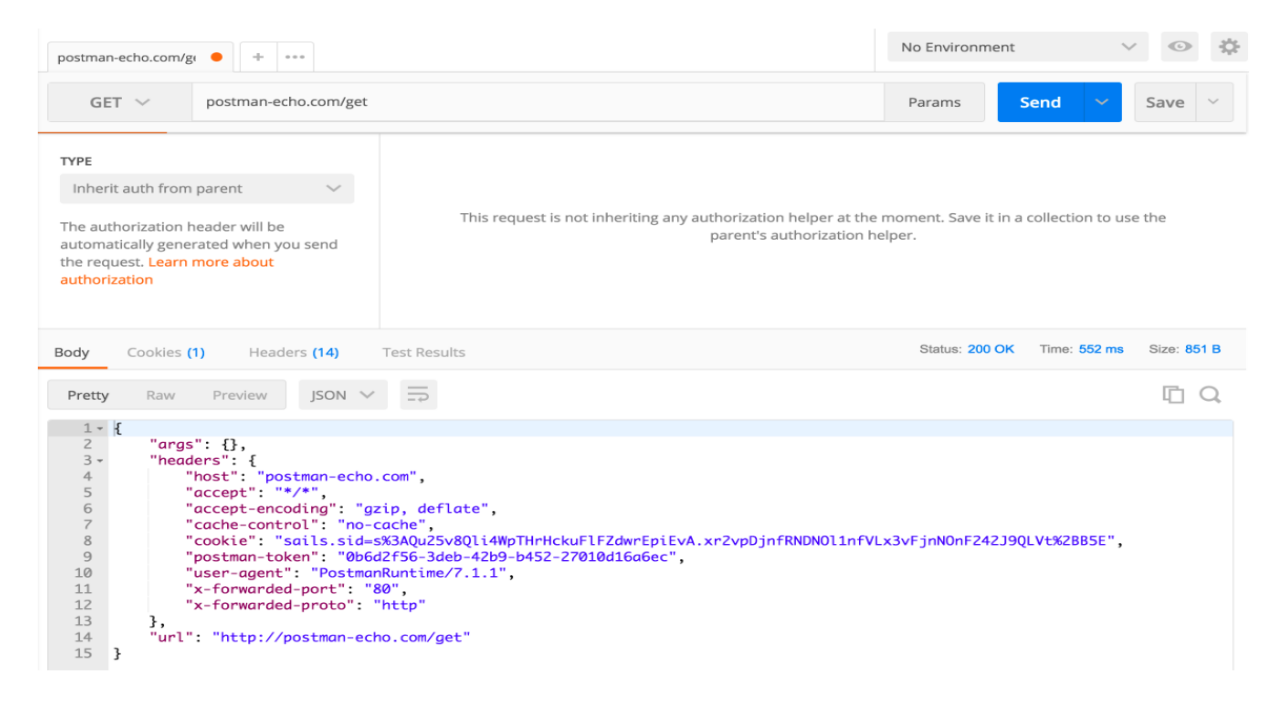
GET: phương thức lấy dữ liệu từ API.

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.

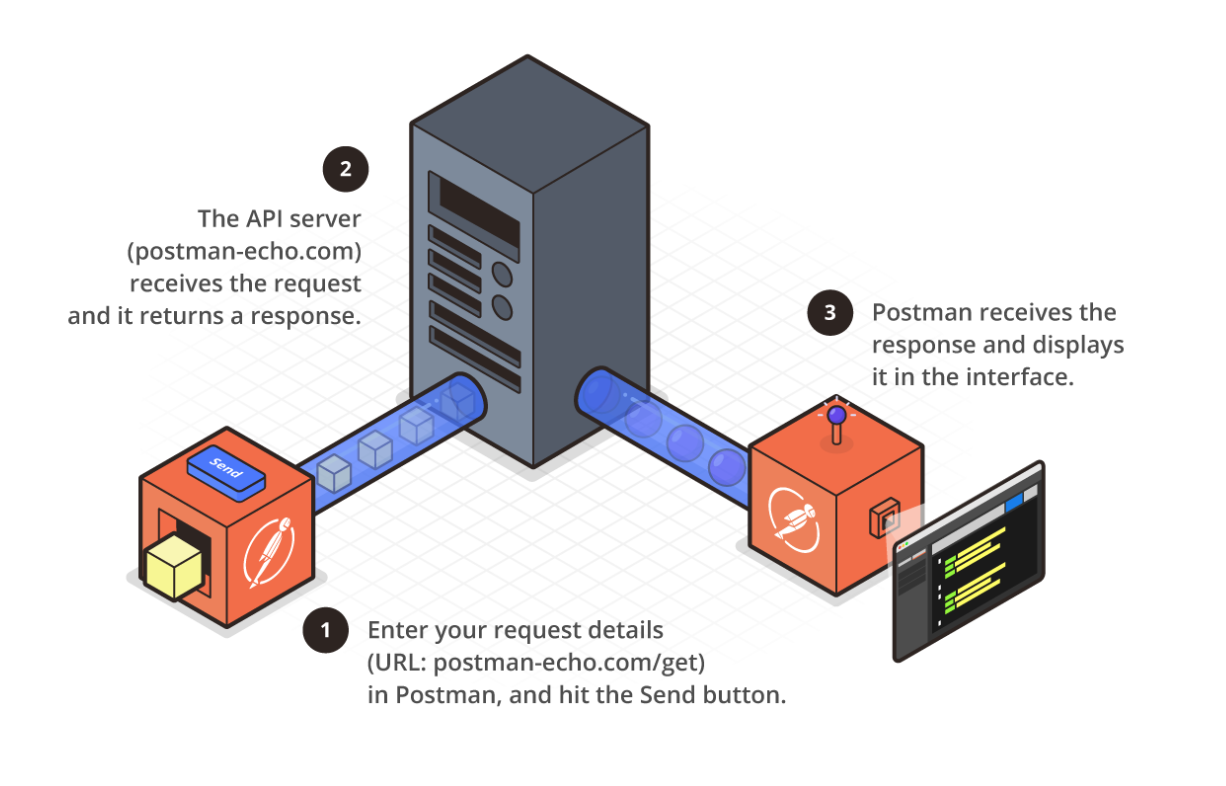


Tiến trình được **Postman** xử lý khi người dùng request api trong giao diện của Postman.

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.

