Có thể nói nguyên lí REST và cấu trúc dữ liệu RESTful được biết đến rộng rãi trong giới lập trình web nói chung và lập trình ứng dụng nói riêng.

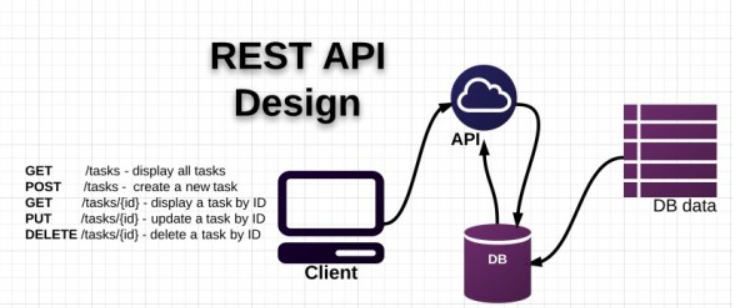
Có thể nói bản thân REST không phải là một loại công nghệ. Nó là phương thức tạo API với nguyên lý tổ chức nhất định. Những nguyên lý này nhằm hướng dẫn lập trình viên tạo môi trường xử lý API request được toàn diện.

Để hiểu rõ hơn về RESTful API ta sẽ đi lần lượt giải thích các khái niệm nhở API, REST hay RESTful.

**RESTful API là gì?**

Các lập trình viên web thường nhắc đến nguyên lý REST và cấu trúc dữ liệu RESTFUL bởi nó là một phần rất quan trọng trong sự phát triển của các ứng dụng web. Vậy RESTFUL API là gì ? Để hiểu rõ hơn chúng ta cùng nhau tìm hiểu nhé.

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.



**Các thành phần của nó**

**API** (**A**pplication **P**rogramming **I**nterface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML.

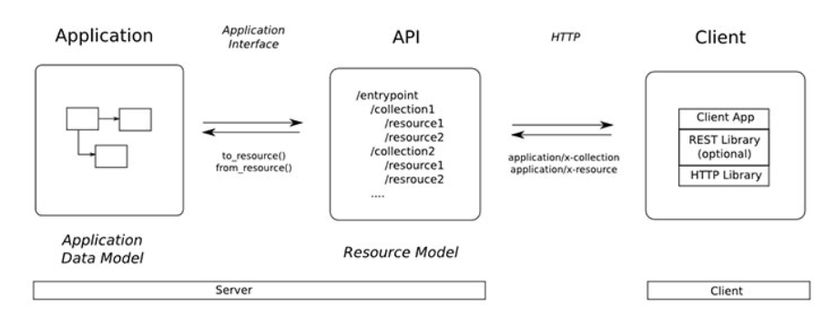
**REST** (**RE**presentational **S**tate\*\* T\*\*ransfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu.

**RESTful API** là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau.

Chức năng quan trọng nhất của **REST** là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE…) và cách định dạng các URL cho ứng dụng web để quản các resource. **RESTful** không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một **RESTful API.**

**RESTful API hoạt động như thế nào?**

Sau khi chúng ta biết được RESTful API là gì thì trong phần này chúng ta cùng tìm hiểu nguyên lý hoạt động của nó nhé. Giống như các giao thức truyền thông hay cấu trúc dữ liệu khác. Để hiểu được bản chất vấn đề thì trước hết cần phải hiểu nguyên lý hoạt động của nó.



REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

* GET (SELECT): Trả về một Resource hoặc một danh sách Resource.
* POST (CREATE): Tạo mới một Resource.
* PUT (UPDATE): Cập nhật thông tin cho Resource.
* DELETE (DELETE): Xoá một Resource.

Những phương thức hay hoạt động này thường được gọi là CRUD tương ứng với Create, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa.

Hiện tại đa số lập trình viên viết RESTful API giờ đây đều chọn JSON là format chính thức nhưng cũng có nhiều người chọn XML làm format, nói chung dùng thế nào cũng được miễn tiện và nhanh.

**Authentication request và cấu trúc dữ liệu trả về**

RESTful API không sử dụng session và cookie, nó sử dụng một access\_token với mỗi request. Bạn có thể tìm hiểu JWT (JsonWebToken) để biết rõ hơn. Mình sẽ làm một bài về JWT trong phần sau nữa ) Dữ liệu trả về thường có cấu trúc như sau:

{

"status\_code": 200,

"data": [

{

"name": "ManhLD",

"email": "manhld@example.com",

"ny": "not found"

},

{

"name": "Ahri",

"email": "ahriKDA@lmht.com",

"ny": "Ezreal"

}

],

error\_messages: ""

}

Ở trên là ví dụ về cấu trúc trả về của api get một list users trong hệ thống.

**Status code**

Khi chúng ta request một API nào đó thường thì sẽ có vài status code để nhận biết sau:

* 200 OK – Trả về thành công cho những phương thức GET, PUT, PATCH hoặc DELETE.
* 201 Created – Trả về khi một Resouce vừa được tạo thành công.
* 204 No Content – Trả về khi Resource xoá thành công.
* 304 Not Modified – Client có thể sử dụng dữ liệu cache.
* 400 Bad Request – Request không hợp lệ
* 401 Unauthorized – Request cần có auth.
* 403 Forbidden – bị từ chối không cho phép.
* 404 Not Found – Không tìm thấy resource từ URI
* 405 Method Not Allowed – Phương thức không cho phép với user hiện tại.
* 410 Gone – Resource không còn tồn tại, Version cũ đã không còn hỗ trợ.
* 415 Unsupported Media Type – Không hỗ trợ kiểu Resource này.
* 422 Unprocessable Entity – Dữ liệu không được xác thực
* 429 Too Many Requests – Request bị từ chối do bị giới hạn

Trong Ruby on Rails có thể sử dụng symbol status code hoặc 3 chữ số integer

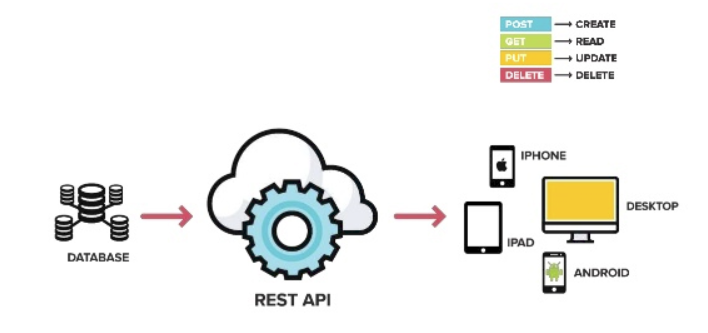
**Quản lí version của api**

Khi thiết api cho app ios hay client side, chúng ta nên đặt version cho các api. Ví dụ như endpoint sau: api/v1/users

Điều này sẽ giúp hệ thống sau khi nâng cấp lên version mới vẫn hộ trợ các api của version cũ, cũng như giúp việc bảo trì, sửa chữa dễ dàng hơn.

**Ưu điểm của RESTFUL API là gì ?**

Như trình bày ở trên, việc sử dụng RESTFUL API mang lại những hiệu quả nhất định cho các lập trình viên. Vậy những lợi ích nó mang lại là gì ? So với các phương pháp khác nó sẽ có điểm gì vượt trội



Một số ưu điểm chính khi sử dụng RESTFUL API là:

* Giúp cho ứng dụng rõ ràng hơn
* REST URL đại diện cho resource chứ không phải hành động
* Dữ liệu được trả về với nhiều định dạng khác nhau như: xml, html, json….
* Code đơn giản và ngắn gọn
* REST chú trọng vào tài nguyên của hệ thống

Những trang web ngày nay thường sử dụng REST API để cho phép kết nối đến dữ liệu của họ. Trong đó, facebook cũng cung cấp các REST API để giúp các ứng dụng bên ngoài kết nối đến dữ liệu của họ

Từ vựng quan trọng cho API REST

**API** - Giao diện ứng dụng và lập trình

**REST** - Chuyển giao trạng thái đại diện

**Client** - Thực thể đang sử dụng API, bằng cách thực hiện (các) lệnh gọi HTTP đối với "điểm cuối" API

**Server** - Thực thể cung cấp API, bằng cách cung cấp phản hồi cho (các) lệnh gọi HTTP nhận được trên "điểm cuối"

**HTTP - HyperText Transfer Protocol** là giao thức web xác định cách thức thông điệp được định dạng và truyền đi. Hơn nữa, HTTP mô tả các hành động Máy chủ web và trình duyệt phải thực hiện để đáp ứng các yêu cầu khác nhau.

Điểm cuối - các liên kết đại diện cho các chức năng của API mà khách hàng có thể sử dụng.

**HTTP** **Verbs**- các hành động khác nhau có thể được thực thi đối với các điểm cuối. Ví dụ: ĐỌC, ĐĂNG, CẬP NHẬT hoặc XÓA.

**Parameters** - Các tham số của yêu cầu. Các tham số này có thể là tham số tiêu đề hoặc nội dung.

Quyền - Tập hợp các quyền mà ứng dụng khách có liên quan đến một số API. Trong hầu hết các trường hợp, chúng liên quan đến mã thông báo.

Hạn chế - Tập hợp các giới hạn mà ứng dụng khách có liên quan đến một số API.

Token - Cơ chế ủy quyền, phiên bản kết hợp tên người dùng / mật khẩu an toàn hơn .

Loại mã thông báo - Các cách khác nhau để nhận mã thông báo từ một số API và sử dụng nó. Đây là một chủ đề rất phức tạp.

OAuth - Giao thức ủy quyền mở

JSON - Định dạng trao đổi dữ liệu nhẹ

cURL - Thư viện để thực thi các giao thức khác nhau từ dòng lệnh. Nó có thể được sử dụng để thực thi các yêu cầu HTTP từ dòng lệnh.

Postman -Phần mềm miễn phí để thực hiện các yêu cầu API (từ bảng điều khiển đồ họa). Rất hữu ích và phải có công cụ cho các nhà phát triển API.