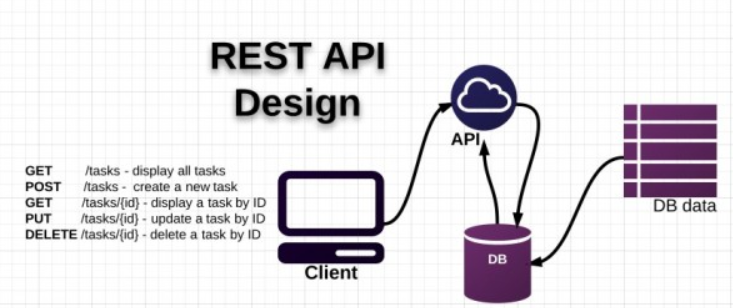
**Ex6\_Rest API:**  
- Viết báo cáo thu hoạch về phần tìm hiểu Rest API

- **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.



- **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** (**Representation State Transfer**) 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.

- Tìm hiểu **resource**:

**Resource** là tài nguyên. Và quản lý **tài nguyên** chính là nhiệm vụ quan trọng và chiếm phần lớn thời gian trong việc phát triển website.

**resource** của các website khác nhau thì có thể sẽ khác nhau, ví dụ:

+ **Facebook**: **resource** nó có thể là người dùng, fanpage....

+ **Tiki**: thì có thể là sản phẩm, người dùng....

+ **Blog của Sói**: Thì là mấy cái bài viết linh tinh của sói...

Và quản lý một resource của một website bao gồm 4 tác vụ chính: Thêm, sửa, xóa, lấy thông tin của resource.

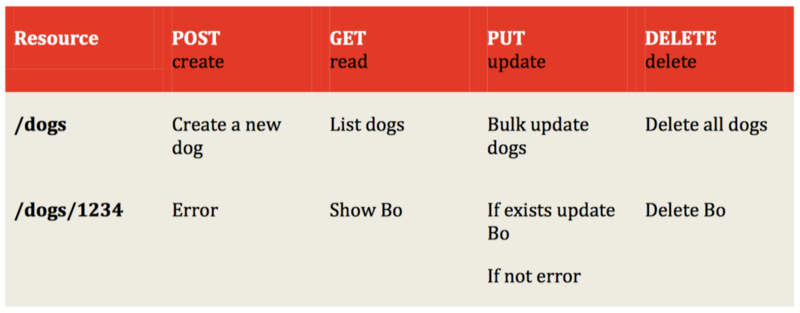
- Các phương thức HTTP:

* **GET**: được sử dụng để lấy thông tin từ sever theo URI đã cung cấp.
* **HEAD**: giống với GET nhưng response trả về không có body, chỉ có header.
* **POST**: gửi thông tin tới sever thông qua các biểu mẫu HTTP.
* **PUT**: ghi đè tất cả thông tin của đối tượng với những gì được gửi lên.
* **PATCH**: ghi đè các thông tin được thay đổi của đối tượng.
* **DELETE**: xóa tài nguyên trên server.
* **CONNECT**: thiết lập một kết nối tới server theo URI.
* **OPTIONS**: mô tả các tùy chọn giao tiếp cho resource.
* **TRACE**: thực hiện một bài test loop - back theo đường dẫn đến resource.

Nhưng chỉ sử dụng các phương thức GET, POST, PUT, DELETE, PATCH là chính.

* **Nguyên tắc thiết kế của REST API**

1. Dùng HTTP method rõ ràng.

Chúng ta có 4 HTTP method cơ bản bao gồm POST, GET, PUT, DELETE. Với mỗi method sẽ ứng với một chức năng tương ứng của API là tạo, đọc, sửa và xoá.

2. Sử dụng danh từ số nhiều và không sử dụng động từ.

Ví dụ như /dogs, /cats,... chứ không phải là /getAllDog,...

3. Chỉ sử dụng danh từ số nhiều.

Không vừa dùng số nhiều vừa dùng số ít.

4. Versioning

Versioning là một điều bắt buộc với tất cả resource, việc đánh version cho resource tuân thủ 2 nguyên tắc sau:

* Bắt đầu bằng “v” và kết thúc bằng một số nguyên dương , tránh dùng số thập phân (dùng v1 thay vì v1.5)
* Versioning sẽ được đặt ở vị trí đầu tiên của resource

5. HTTP status code và error message

* 200 — OK — Everything is working
* 304 — Not Modified — The client can use cached data
* 400 — Bad Request — The request was invalid or cannot be served. The exact error should be explained in the error payload. E.g. „The JSON is not valid“
* 401 — Unauthorized — The request requires an user authentication
* 403 — Forbidden — The server understood the request, but is refusing it or the access is not allowed.
* 404 — Not found — There is no resource behind the URI.
* 422 — Unprocessable Entity — Should be used if the server cannot process the enitity, e.g. if an image cannot be formatted or mandatory fields are missing in the payload.
* 500 — Internal Server Error — API developers should avoid this error. If an error occurs in the global catch blog, the stracktrace should be logged and not returned as response.