BÀI THU HOẠCH DAY 7

1. **API là gì:**

API là viết tắt của Application and Programming Interface. API cho phép các phần của một phầm mềm giao tiếp với nhau.

Có rất nhiều kiểu API, ví dụ như facebook api, twitter api, vv.. Nhưng khi nói về API, thường chúng ta sẽ đề cập đến REST API.

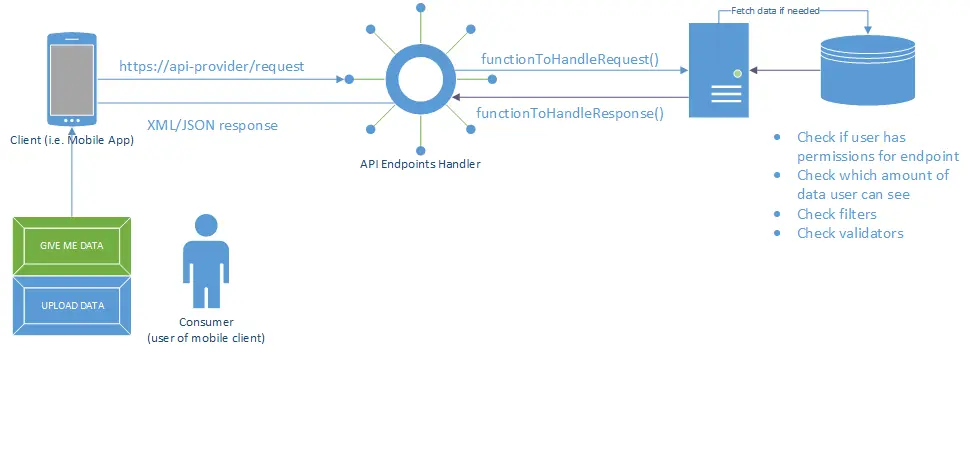
1. **REST là gì:**

REST là viết tắt của **Re**presentation **S**tate **T**ransfer, vậy ý nghĩa của nó là gì ?

Cho 1 ví dụ như là Twitter. Trước khi Twitter API được tạo ra thì tất cả những chức năng như tạo tweets, like, share đều phải thực hiện trên web site của Twitter.

Khi Twitter tạo ra API, họ đã khiến ta có khả năng dùng những chắc năng trên bằng những nguồn bên ngoài (external applications). Vì vậy mà khi phát triển phần mềm (Application hoặc Mobile), developer có thể tạo những phần mềm mà có thể làm những chức năng twitter cung cấp.

Ví dụ theo biểu đồ sau:



Theo biểu đồ trên, Cosumer (Client) sẽ là 1 người. Người này có thế ấn vào 1 trong 2 nút trên một ứng dụng điện thoại: Nút Give me date , Nút Upload Data. Khi người dùng ấn nút give me data, ứng dụng sẽ thực thi 1 request đến nơi cung cấp dịch vụ (service provider). Đầu tiền, API endpoints sẽ xử lý request này và gọi những hàm tương ứng. Sau đó các logic để check quyền, check lượng data khách hàng được truy cập.v.v.. sẽ diễn ra. Sau đó, data sẽ được sẵn sàng để trả về và REST API sẽ trả về data dưới dạng Json hoặc XML đến ứng dụng điện thoại của người dùng. Và cuối cùng ứng dụng của người dùng sẽ dùng những data đó để hiển thị lên.

**Những điều quan trọng cần lưu ý:**

* Client luôn phải được xác thực trước khi nhận hoặc gửi data
* Client có thể thực thi nhiều REPT API endpoints để lấy hoặc gửi các data khác nhau.
* Client có thể gọi API nhiều lần.
* API được dùng nhiều nhất là REST API.

Những từ vựng cần nhớ khi làm việc với REST API:

* API: Application Programming Language.
* REST: Representation State Tranfer
* Client: Thực thể sử dụng API, thông qua phương thức HTTP(s).
* Server: Thực thể chứa và thực thi API
* HTTP: Hyper Text Tranfer Protocol, định nghĩa cách thức truyền các gói tin cũng như định dạng của các gói tin đó
* Endpoints: Điểm cuối, là links của các API mà Client có thể sử dụng.
* Parameters: Tham số của request, có thể ở header hoặc body của request.
* Permission: Quyền sử dụng API của Client. Trong nhiều trường hợp liên quan đến TOKEN.
* TOKEN: Cơ chế xác thực định danh, tăng tính bảo mật.
* JSON: định dạng dữ liệu.
* HTTP Verb: GET, PUT, POST, DELETE…
* PostMan: ứng dụng miễn phí để thực thi API.

Vậy tóm lại REST là gì, REST là 1 phòng cách kiến trúc của phần mềm. Hiểu sâu hơn thì nó là 1 tập các quy luật để tạo nên 1 phần mềm REST. Và các phần mềm đảm bảo được những quy luật này sẽ được gọi là RESTful.

1. **Cụ thể hơn về REST API:**
   1. **Statelessness:**

Những hệ thống theo chuẩn REST sẽ là stateless (vô trạng thái), nghĩa là nó không quan tâm đến trạng thái của người dùng và ngược lại. Theo cách này, client và server có thể giao tiếp với nhau mà không cần biết những message trước đó là gì.

Vì hệ thống REST qua những phương thức tiêu chuẩn trên resources nên chúng không phụ thuộc vào việc cài đặt interfaces. Vì điều này nên những hệ thống REST hoạt động nhanh, đáng tin cậy, dễ dàng mở rộng, nâng cấp mà không ảnh hưởng đến toàn bộ hệ thống

* 1. **Giao tiếp giữa Client và Server:**
     1. Tạo Request:

1 Request đến hệ thống REST cần có :

* Một HTTP Verb (POST, PUT, GET, DELETE) để hệ thống biết được cần thực hiện phương thức nào.
* Một HEADER cho phép client truyền thông tin về request
* Một đường dẫn đến resource.
* Một Body message (Optional) chứa dữ liệu.
  + 1. HTTP VERB:
* GET: lấy về mật resource cụ thể ( theo ID) hoặc một danh sách các resource.
* POST: tạo mới một resource.
* PUT: update một resource (theo ID).
* DELETE: xóa một resource (theo ID).
  + 1. HEADER và Accept Parameter:

Trong HEADER của một request, sẽ chứa loại dữ liệu mà Client có thể nhận từ server. Gọi là là trường ‘Accept’ và nó đảm bảo server sẽ không gửi dữ liệu mà Client không thể đọc. Loại dữ liệu này gọi là MIME ( Mutipurepose Internet Mail Extension).

Ví dụ:

* Image : image/png, image/jepg, image/gif.
* Audio: audio/wav, audio/mpeg.
* Video: video/mp4, video/ogg.
* Application: application/json, application/xml, application/pdf ..
  + 1. PATHS:

Request cần chứa path (đường dẫn) đến resource mà operation cần thực thi trên đó. Trong REST APIs, paths cần được thiết kế để Client có thể biết được điều gì đang diễn ra.

Thông thường, path nên theo dạng cụ thể của một resource, để khiến cho những path nối theo dễ đọc và dễ hiểu.

Ví dụ: fashionboutique.com/customer/223/order/12

Đường dẫn trên chỉ rất rõ ràng nó thực hiện điều gì. Chúng ta có thể thấy nó lấy order với ID là 12 của customer có id là 223.

Path cần chứa những thông tin cần thiết để truy cập một resource với góc độ nhất định. Nhưng khi nói đến list hay collection thì path không phải lúc nào cũng cần đến id.

Nếu chúng ta lấy muốn lấy một resource thì cần thêm id vào cuối path

* + 1. Response codes.

Phản hồi từ server sẽ chứa status code (mã trạng thái) để báo với Client về kết quả của request. Ví dụ một số code:

* 400 (BAD REQUEST) : Request không thể thực hiện vì lỗi
* 403 (FORBIDDEN) : Request không thể thực hiện vì không có quyền.
* 404 (NOT FOUND) : Request đến một resource không tồn tại.
* 500 (INTERNAL SERVER ERROR): Response chung chung cho một Request thất bại với lỗi không cụ thể.
* 200 (OK) : Request thành công.
* 201 (CREATED) : Tạo resource thành công.