CHƯƠNG 1:GIỚI THIỆU VỀ API

**API là gì?**

API là viết tắt của Application Programming Interface – phương thức trung gian kết nối các ứng dụng và thư viện khác nhau.

Nó cung cấp khả năng truy xuất đến một tập các hàm hay dùng, từ đó có thể trao đổi dữ liệu giữa các ứng dụng.

Nói cách khác, nếu muốn tương tác với một máy tính hay hệ thống để truy xuất thông tin thì API sẽ giúp người dùng giao tiếp với hệ thống để nó hiểu và đáp ứng các yêu cầu.

**RESTful API**

**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**

**Ưu điểm và nhược điểm của RESTful API**

**Ưu điểm:**

* Dễ hiểu, dễ học, đơn giản.
* Cho phép tổ chức các ứng dụng phức tạp, dễ dàng sử dụng tài nguyên.
* Quản lý tải cao nhờ HTTP proxy server và cache.
* Các client mới có thể dễ dàng làm việc trên những ứng dụng khác.
* Cho phép sử dụng các lệnh gọi thủ tục HTTP tiêu chuẩn để truy xuất dữ liệu và request.
* RESTful API dựa trên code và có thể sử dụng nó để đồng bộ hoá dữ liệu bằng website.
* Cung cấp các định dạng linh hoạt bằng cách tuần tự hoá (serialize) dữ liệu ở dạng XML hay JSON.
* Cho phép sử dụng các giao thức OAuth để xác thực request REST.

**Nhược điểm:**

Việc duy trì trạng thái này là nhiệm vụ của phía client, do đó ứng dụng client có thể cồng kềnh và khó bảo trì hơn.

Bảo mật:REST có thể phù hợp với các URL public, nhưng không phải là một lựa chọn tốt nếu cần truyền dữ liệu nhạy cảm giữa client và server.

**GraphQL là gì?**

**GraphQL là ngôn ngữ thao tác và truy vấn dữ liệu nguồn mở cho API**, cung cấp cho client 1 cách thức dễ dàng để request chính xác những gì họ cần, giúp việc phát triển API dễ dàng hơn theo thời gian

GraphQL bao gồm 3 điểm đặc trưng bao gồm:

* Cho phép client xác định chính xác những dữ liệu gì họ cần
* GraphQL làm cho việc tổng hợp dữ liệu từ nhiều nguồn dễ dàng hơn
* Sử dụng một type system để khai báo dữ liệu.

Các công nghệ phổ biến phát triển API

1. **Method (phương thức)**: là các phương thức HTTP được sử dụng để truy cập và thực hiện hành động trên tài nguyên. Các phương thức chính là GET (lấy thông tin), POST (tạo mới tài nguyên), PUT (cập nhật thông tin tài nguyên) và DELETE (xóa tài nguyên).
2. **Representation (biểu diễn)**: là cách mà tài nguyên được đại diện, thông thường là JSON, XML hoặc HTML. Client và server đều sử dụng cùng một định dạng biểu diễn để trao đổi dữ liệu.
3. **URI (Uniform Resource Identifier)**: là định danh duy nhất cho tài nguyên trên mạng, bao gồm schema (http, https), domain và path. URI phải được thiết kế sao cho dễ hiểu và dễ sử dụng cho client.
4. **Request (yêu cầu)**: là yêu cầu được gửi từ client đến server, bao gồm các phương thức HTTP, địa chỉ URI, thông tin header và body (nếu có). Thông tin trong yêu cầu sẽ giúp server xác định được tài nguyên và hành động cần thực hiện.
5. **Response (phản hồi)**: là phản hồi được gửi từ server về client, bao gồm các thông tin header, status code và body (nếu có). Phản hồi sẽ chứa thông tin về tài nguyên được yêu cầu hoặc kết quả của hành động được thực hiện.
6. **Hypermedia (siêu văn bản)**: là các liên kết tới các tài nguyên khác, giúp client truy cập và tương tác với các tài nguyên liên quan. Hypermedia giúp tăng tính linh hoạt của API và cho phép client khám phá các tài nguyên mới.
7. **Stateless (không lưu trạng thái)**: là kiểu kiến trúc của RESTful API, trong đó mỗi yêu cầu đều chứa đầy đủ thông tin để server thực hiện hành động, không lưu trạng thái giữa các yêu cầu. Điều này giúp tăng tính mở rộng của API và giảm tải cho server.

**Các thư viện hỗ trợ phát triển API trong NodeJS**

1. **Express**: là một framework web phổ biến cho Node.js, giúp xây dựng các ứng dụng web nhanh chóng và dễ dàng.
2. **Axios**: là một thư viện HTTP client phổ biến, giúp gửi các yêu cầu HTTP và xử lý phản hồi HTTP một cách dễ dàng.
3. **Nodemailer**: cung cấp các công cụ để gửi email trong Node.js.
4. **Debug**: cung cấp các công cụ để xử lý và ghi log trong Node.js.
5. **Winston**: là một thư viện ghi log phổ biến trong Node.js, giúp quản lý và ghi log các hoạt động của ứng dụng.
6. **Joi**: cung cấp các công cụ để xác thực và kiểm tra dữ liệu trong Node.js.
7. **Underscore.js**: cung cấp các phương thức tiện ích để xử lý và thao tác các đối tượng JavaScript, tương tự như Lodash.
8. **Cheerio**: giúp phân tích và xử lý HTML trong Node.js, giúp lấy dữ liệu từ các trang web.
9. **Bluebird**: cung cấp các công cụ để xử lý và quản lý các hoạt động bất đồng bộ trong Node.js.
10. **Mongoose**: là một thư viện MongoDB ODM (Object Document Mapping), giúp quản lý và truy xuất dữ liệu MongoDB một cách dễ dàng.
11. **Request**: cung cấp các công cụ để gửi các yêu cầu HTTP và xử lý các phản hồi HTTP.
12. **Async**: giúp quản lý các hoạt động bất đồng bộ và xử lý các lỗi trong Node.js.
13. **Passport**: cung cấp các công cụ để xác thực và quản lý đăng nhập người dùng.
14. **Node-sass**: là một thư viện Sass cho Node.js, giúp biên dịch các file Sass thành CSS.
15. **Imagemin**: cung cấp các công cụ để tối ưu hóa hình ảnh trong Node.js, giúp giảm kích thước tệp ảnh và tăng tốc độ tải trang web.
16. **Morgan**: giúp ghi log các yêu cầu HTTP vào console hoặc file log trong Node.js.

**Chương 3: Các thành phần của một RESful API**

**Resource (tài nguyên):** là đối tượng hoặc tập hợp các đối tượng mà client muốn truy cập thông tin hoặc thực hiện hành động. Mỗi tài nguyên được đại diện bằng một URI

ví dụ: /users hoặc /products/123.

HTTP Method

HTTP Request Method:

Bắt đầu của HTTP Request sẽ là dòng Request-Line bao gồm 3 thông tin đó là:

* Method: là phương thức mà HTTP Request này sử dụng, thường là GET, POST, ngoài ra còn một số phương thức khác như HEAD, PUT, DELETE, OPTION, CONNECT. Trong ví dụ trên là GET
* URI: là địa chỉ định danh của tài nguyên. Trong tường hợp này URI là / - tức request cho tài nguyên gốc, nếu request không yêu cầu một tài nguyên cụ thể, URI có thể là dấu \*.
* HTTP version: là phiên bản HTTP đang sử dụng, ở đây là HTTP 1.1

HTTP Response

Cấu trúc HTTP response gần giống với HTTP request, chỉ khác nhau là thay vì Request-Line, thì HTTP có response có Status-Line. Và giống như Request-Line, Status-Line cũng có ba phần như sau:

* HTTP-version: phiên bản HTTP cao nhất mà server hỗ trợ.
* Status-Code: mã kết quả trả về.
* Reason-Phrase: mô tả về Status-Code.

HTTP Status Code: là câu trả lời của máy chủ (server) cho yêu cầu đến từ máy khách (client)

Response và Request:

* **Request**: Là client gửi yêu cầu lên server
* **Response**: Là kết quả từ server trả về cho client

**Parameter and Query:**

Parameter :là các giá trị được truyền vào một API request như một phần của URL hoặc trong phần body của request. Các tham số này thường được sử dụng để truyền dữ liệu hoặc thông tin về hành động cần thực hiện.

Ví dụ, trong một API để tạo mới một người dùng, các tham số có thể bao gồm tên, địa chỉ email, mật khẩu và các thông tin khác liên quan đến người dùng.

Query: là một phần của URL và được sử dụng để yêu cầu dữ liệu từ server. Các tham số truy vấn (query parameters) được sử dụng để chỉ định các điều kiện tìm kiếm hoặc lọc dữ liệu trả về từ API response.

Ví dụ, trong một API để tìm kiếm các địa điểm du lịch, các tham số truy vấn có thể bao gồm kinh độ, vĩ độ, bán kính tìm kiếm và các điều kiện khác để giới hạn các kết quả trả về.

**Header và Body:**

Header là phần đầu của request hay respone chứa các thông tin về mô tả nội dung của message. Bao gồm các trường như Contetent-Type, Accept, Authorization, Cookie,...

Một số trường của Header:

* Content-Type: Xác định của nội dung được gửi đi hay trả về.
* Authorization: Xác thực người dùng để truy cập.
* Cookie: Thông tin được lưu trữ trên trình duyệt đi kèm với mỗi lần Request và Respone.
* User-Agent: Xác định thông tin của trình duyệt hay ứng dụng thực hiện request, respone.
* Body là phần chứa dữ liệu được gửi đi hay trả về trong request hay respone. Các dữ liệu trong body có định dạng xml hay JSON.

HATEOAS

Hypermedia liên kết đến các phần khác nhau của API, cho phép các resource của API chứa các liên kết và thông tin có thể truy cập tăng cường sự tương tác giữa client và server.

Các liên kể giúp cho client có thể tìm kiếm các tài nguyên và thực hiênh các hành động khác trên các tài nguyên đó.

HATEOAS là phương pháp thiết kế RESTful API. Bằng cách sử dụng các hypermedia, HATEOAS cho phép client có thể tự động khám phá các tài nguyên mới và hành động được thực hiện trên các tài nguyên đó mà không cần sự can thiệp bởi ltv.

HATEOAS là phương pháp thiết kế API là một phần của kiến trúc RESTful, tuy nhiên không phải các RESTful API đều dùng HATEOAS và HATEOAS đều chưa chắc đúng với các quy tắc của RESTful API