Thu Hoạch Tuần 1

Nhóm BE

[I. Giới thiệu về API 2](#_Toc132684281)

[1. API là gì? 2](#_Toc132684282)

[2. RESTful API và GraphQL API 2](#_Toc132684283)

[a. RESTful API là gì? 2](#_Toc132684284)

[b. GraphQL API là gì 2](#_Toc132684285)

[3. Các công nghệ phổ biến cho việc phát triển API 3](#_Toc132684286)

[a. Các API miễn phí 3](#_Toc132684287)

[b. Các công cụ kiểm thử API 3](#_Toc132684288)

[4. Các thư viện hỗ trợ phát triển API trong Nodejs 3](#_Toc132684289)

[II. Xây dựng API đơn giản với Nodejs 5](#_Toc132684290)

[1. Tạo project mới với Nodejs và Expressjs 5](#_Toc132684291)

[2. Tạo các route đơn giản 5](#_Toc132684292)

[3. Sử dụng middleware trong Expressjs 5](#_Toc132684293)

[4. Kết nối với cơ sở dữ liệu 5](#_Toc132684294)

[III. Các thành phần của một RESTful API 5](#_Toc132684295)

[1. Resource 5](#_Toc132684296)

[2. Request và Response 5](#_Toc132684297)

[a. Request 5](#_Toc132684298)

[b. Response 6](#_Toc132684299)

[3. Parameter và Query 6](#_Toc132684300)

[4. Hateoas 6](#_Toc132684301)

1. Giới thiệu về API
2. API là gì?

API là một tập hợp các commands, functions, protocols, objects, giúp hai phần mềm (ứng dụng) có thể tương tác và trao đổi dữ liệu qua lại được với nhau.

Có bốn loại API chính thường được sử dụng trong các ứng dụng dựa trên web: Công khai, đối tác, riêng tư và hỗn hợp.

Vd1:

Ứng dụng 1: xây dựng hàm tính tổng ‘*function sum(a,b)*’

Ứng dụng 2: sử dụng hàm tính tổng của ứng dụng 1: ứng dụng 1 cung cấp API hàm tính tổng cho ứng dụng 2 sử dụng.

Vd2:

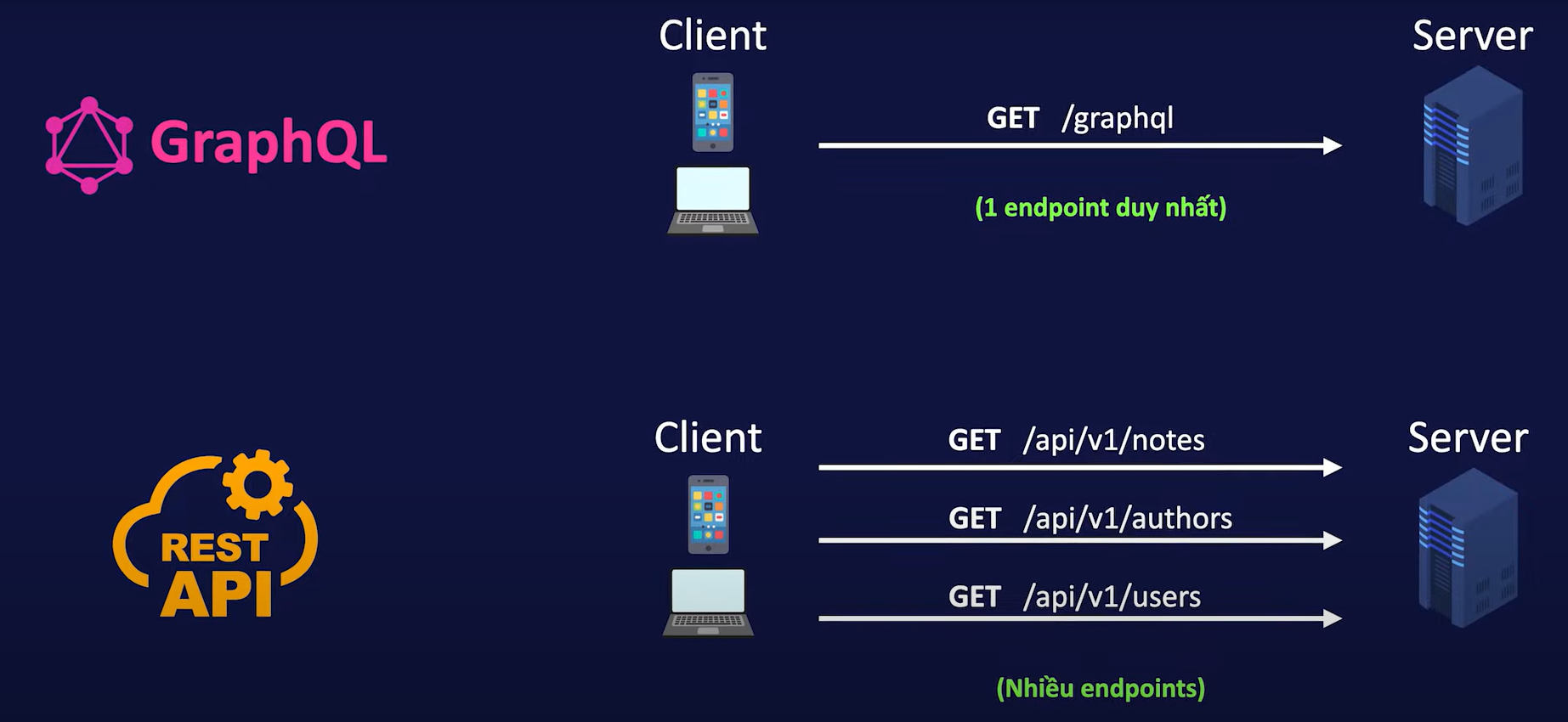
* React xây dựng hàm `*useState(initValue)*`
* Lập trình viên: Sử dụng hàm `*useState’* trong ứng dụng Javascript của họ

Vd3:

Back End: Xây dựng API lấy danh sách bài viết *http://api.com/posts?category=1*

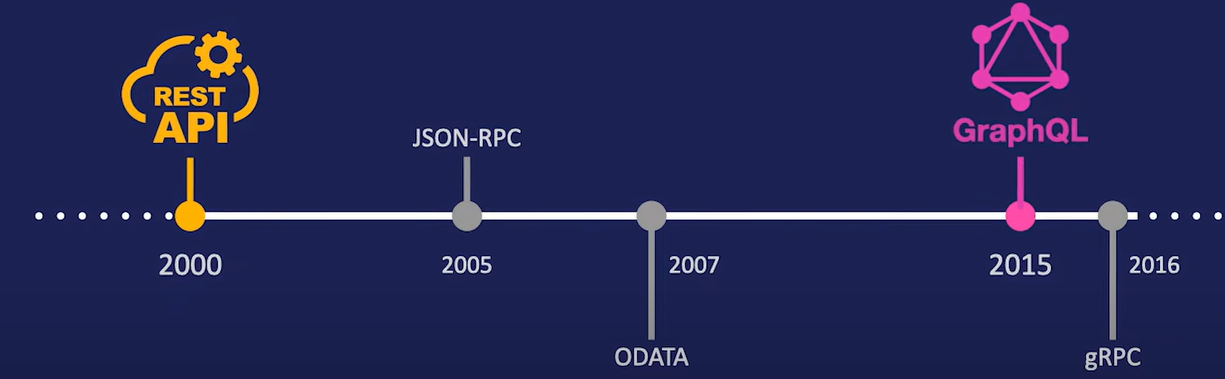
Front End: Sử dụng API của Back End để lấy dữ liệu về Front End

1. RESTful API và GraphQL API
2. RESTful API là gì?
   * REST API là một kiểu kiến trúc được sử dụng trong việc giao tiếp giữa các máy tính, định nghĩa 1 tập các ràng buộc, quy tắc mà tuân theo khi phát triển web server.
   * Một web server có các API tuân thủ theo REST định nghĩa -> RESTful API
     + Endpoint: *https://test.com/api/user*
     + HTTP methods: GET, POST, PUT, PATCH, DELETE
     + HEADERS: cung cấp thông tin client muốn gửi cho sever
     + BODY: chứa các thông tin client muốn gửi cho sever
3. GraphQL API là gì
   * GraphQL API là ngôn ngữ truy vấn cho API.
   * Cho phép clinet truy vấn chính xác dữ liệu mà clinet mong muốn.
   * GraphQL có 1 endpoint duy nhất thay vì nhiều endpoint như REST



* + GraphQL có 3 tùy chọn





1. Các công nghệ phổ biến cho việc phát triển API
2. Các API miễn phí

* API Dog
* shibe.online
* Anime News Network API
* Petfinder
* Jikan
* JSONPlaceholder
* REQ | RES

1. Các công cụ kiểm thử API

* Postman
* Jmeter
* Katalon Studio
* Tricentis Tosca
* Apigee

1. Các thư viện hỗ trợ phát triển API trong Nodejs

* **Express**

expressjs là bộ khung giúp dựng các ứng dụng web trên Nodejs, bạn có thể dễ dàng custom router, controller, … với express.

Làm ứng dụng MEAN stack (MongoDB, Express.js, Angular.js, Node.js)

Có thể xử lý dễ dàng nhiều loại yêu cầu như GET, PUT, POST và DELETE

Xây dựng các single page app, multi page app và hybrid web app.

* **Async**

Workflow của nodejs chạy theo hướng bất đồng bộ và callback, đây là một trong những điểm mạnh của nodejs. Nhưng đây lại là nỗi khốn khổ của phần lớn begginner bởi không thể kiểm soát được các luồng của async. Async là module giúp bạn khử bất đồng bộ, chạy các hàm 1 cách tuần tự.

* **lodash**

Lodash giúp bạn tăng sức mạnh cho ứng dụng javascript và nodejs với các hàm mở rộng để xử lý chuỗi, mảng, object, …

* **cheerio**

Cheerio phân tích cú pháp như HTML và cung cấp API để duyệt/thao tác cấu trúc dữ liệu kết quả

* **nodemailer**

Gửi các email từ Node.JS, dễ dàng gửi các email - sử dụng SMTP

* **faker**

Tạo một lượng lớn dữ liệu giả trong trình duyệt và node.js.

* **morgan**

Phần mềm trung gian logger yêu cầu HTTP cho node.js

* **http-errors**

Tạo erros HTTP cho Express, Koa, Connect, v.v. Dễ dàng gửi phản hồi lỗi, nhiều thuộc tính lỗi có sẵn

* **moment**

Cung cấp các hàm helper để xử lý, tính toán, xác thực, hiển thị, … thời gian trong javascript.

* **socket.io**

Socket.io là package giúp tạo các ứng dụng realtime bằng kết nối socket.

* **mongoose**

Mongoose có sẵn các hàm giúp tương tác đến CSDL MongoDB một cách dễ dàng.

* **mysql**

mysql.js giúp bạn dễ dàng kết nối đến cơ sở dữ liệu mysql bằng Nodejs.

1. Xây dựng API đơn giản với Nodejs
2. Tạo project mới với Nodejs và Expressjs

* Cài đặt NodeJS
* Tạo thư mục để chứa chương trình
* Tạo chương trình nodeJS
* Tạo index.js
* Cài thư viện express
* Code hello world
* Chạy

1. Tạo các route đơn giản
2. Sử dụng middleware trong Expressjs

* Phần mền trung gian.
  + Vd: kiểm tra dữ liệu khi client gửi về server
* Lưu ý:
  + Thứ tự
  + Để trống middleware dẫn đến treo

1. Kết nối với cơ sở dữ liệu

npm isntall mongoose@6.10.0

1. Các thành phần của một RESTful API
2. Resource

* Là các tài nguyên mà API có thể tác động đến.
* Restful API chú trọng đến các tài nguyên hệ thống gồm các tệp văn bản, video, ảnh, âm thành hay các dữ liệu di động,… là những mục tiêu mà chúng hướng tới, bao gồm các trạng thái tài nguyên được định dạng và truyền tải qua HTTP.

1. Request và Response
2. Request
   * URL

URL là địa chỉ duy nhất cho 1 request, thường là đường dẫn tới một hàm xử lí logic.

* + Method

HTTP request có tất cả 9 loại method , 2 loại được sử dụng phổ biến nhất là GET và POST

* GET: Sử dụng để lấy thông tin từ server 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 parameters 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 resource 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.
  + Headers

Là nơi chứa các thông tin cần thiết của 1 request nhưng end-users không biết có sự tồn tại của nó. Ví dụ: độ dài của request body, thời gian gửi request, loại thiết bị đang sử dụng, loại định dạng cái response mà client có đọc được…

* + Body

Là nơi chứa thông tin mà client sẽ điền.

1. Response

* Sau khi nhận được request từ phía client, server sẽ xử lý cái request đó và gửi ngược lại cho client 1 cái response. Cấu trúc của 1 response tương đối giống phần request nhưng Status code sẽ thay thế cho URL và Method. Tóm lại, nó có cầu trúc 3 phần:
  + Status code
  + Headers
  + Body
* Phần Header và body tương đối giống với request.
* Status code (Mã hóa trạng thái thường được gọi là mã trạng thái) là một số nguyên 3 ký tự, trong đó ký tự đầu tiên của Status-Code định nghĩa loại Response và hai ký tự cuối không có bất cứ vai trò phân loại nào. Có 5 giá trị của ký tự đầu tiên:
  + 1xx: Information (Thông tin): Khi nhận được những mã như vậy tức là request đã được server tiếp nhận và quá trình xử lý request đang được tiếp tục.
  + 2xx: Success (Thành công): Khi nhận được những mã như vậy tức là request đã được server tiếp nhận, hiểu và xử lý thành công
  + 3xx: Redirection (Chuyển hướng): Mã trạng thái này cho biết client cần có thêm action để hoàn thành request
  + 4xx: Client Error (Lỗi Client): Nó nghĩa là request chứa cú pháp không chính xác hoặc không được thực hiện.
  + 5xx: Server Error (Lỗi Server): Nó nghĩa là Server thất bại với việc thực hiện một request nhìn như có vẻ khả thi.

1. Parameter và Query

* Khi thực hiện một API ta có thể truyền thêm tham số vào câu query
* Vd: *GET /ccadmin/v1/orders?limit=5*

1. Hateoas

* Hateoas là gì?
  + Một Response sẽ tốt nhất khi có chứa Hypermedia trong giá trị trả về. Dễ hiểu là các liên kêt trong kết quả trả về được kết nối với các phương thức API khác để người dùng có nhiều lựa chọn tiếp theo.
  + Vd: trả về kết qảu trả về *links* để người dùng có nhiều lựa chọn cho bước tiếp theo

