**Nodejs API**

**Chương 1: Giới thiệu về API**

* API là gì?

API – Application Programming Interface, là một tổ hợp function, object,... dùng để giao tiếp giữa 2 phần mềm. API được tạo ra để user có thể dùng một cách dễ dàng mà không cần quan tâm đến các yếu tố khác.

Hiện nay có hai kiểu thiết kế API phổ biến nhất đó là RESTful API và GrapQL API.

* RESTful API và GraphQL API
  + RESTful API là một kiểu thiết kế API dựa trên cấu trúc REST(Rpresentational State Transfer), RESTful API dùng các phương thức HTTP như GET, POST, PUT, PATCH, DELETE để thực hiện các resource được định danh bằng URL và được truy xuất thông qua yêu cầu của HTTP.
  + GraphQL API là kiểu thiết kế API mới được ra đời bởi Facebook nhằm khắc phục vấn đề yêu cầu dữ liệu của khách hàng mà RESTful API đang mắc lỗi nhằm đưa ra các dữ liệu khác hàng mong muốn và tối ưu hóa hiệu suất. Khác với RESTful API, GraphQL API cho phép không giới hạn yêu cầu dữ liệu bằng các resource nhất định mà nhiều resource khác nhau trong cùng 1 yêu cầu.
  + So sánh
* Các công nghệ phổ biến cho việc phát triển API

RESTful API:

GraphQL API:

OpenAPI:

Docker:

* Các thư viện hỗ trợ phát triển API trong Nodejs

Express: Cung cấp các tính năng xử lí yêu cầu HTTP, middleware, routing.

Nodemon: Là thư viện hỗ trợ trong quá trình phát triển, Nodemon sẽ theo dõi, và khi có sự thay đổi sẽ tự khởi động lại, và cho phép thấy sự thay đổi của ứng dụng.

Handlebars: thư viện tạo các mẫu HTML sử dụng trong môi trường Nodejs.

Restify: tập trung vào việc phát triển cac RESTful API, cung cấp các tính năng như routing, middleware, xử lí lỗi.

Ngoài ra còn rất nhiều các thư viện khác hỗ trợ trong việc phát triển API

**Chương 2: Xây dựng API đơn giản với Nodejs**

* Tạo project mới với Nodejs và Expressjs

B1: tạo đối tượng sử dụng Expressjs, và thiết lập cổng.

B2: thiết lập path

B3: tạo route.

* Tạo các route đơn giản

Cần tạo 1 đối tượng sử dụng Express bằng cách sử dụng hàm ‘express()’

Các route dựa trên các HTTP Method như:

// tạo route với phương thức Get

app.get(‘/’, function(res, req)){

res.send(‘hello world’);})

có thể tạo các route với các HTTP method khác.

* Sử dụng middleware trong Expressjs

Middleware là một lớp trung gian thực hiện các xử lý trên các yêu cầu của HTTP trước khi được request hay respone, có chức năng kiểm tra, xác thực thông tin, thay đổi hay chỉnh sửa,..

Cấu trúc của middleware bao gồm:

app\_object(res, req, next)

{

//các câu lệnh điều kiện

next() //Call back hay chuyển đến một middleware hay routing khác.

}

\*nếu không middleware không có next, tiến trình sẽ mãi load cho đến khi time out.

* Kết nối với cơ sở dữ liệu

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

* Resource
* HTTP Methods
  + là các phương thức của HTTP nhằm liên lạc giữa server với client
  + các phương thức bao gồm:

GET: lấy thông tin từ server

POST: gửi thông tin lên server

HEAD: tương tự GET, HEAD truyền tải các trạng thái vào khu vực header

PUT: ghi đè hay tạo mới nhưng sẽ không báo lỗi

DELETE: xóa các tài nguyên trên server

CONECT: thiết lập các kết nối trên server

OPTIONS: mô tả các tùy chọn giao tiếp trên server

TRACE: thực hiện test loop-back theo đường dẫn

* Request và Response

Request (Req): gửi các yêu cầu đến server

Respone (Res): các thông tin được trả về từ server

* Parameter và Query

Querry Parameter là một các truyền thông tin từ ứng dụng web hay RESTful API.

Nêu rõ đích đến tới các đối tượng hay định hướng đường đi cụ thể đến các đối tượng, chức năng, thông tin cụ thể

Querry parameter xuất hiện ở cuối các dòng URL được ngăn cách bởi dấu ?

Vd: “<https://google.com/api>?brand=nike&category=sneaker”

* 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.

Ví dụ:

POST /api/products

Content-Type: application/json

Authorization: Long

{

“name”: “Air Force 1”,

“description”: “Product of NIKE”,

“price”: 2000000,

“quantity”: 10

}

* 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

**Chương 4: Bảo mật trong API**

* Authentication
* Authorization
* Encryption
* Token-based Authentication
* JSON Web Token (JWT)
* OAuth 2.0

**Chương 5: Testing và Document API**

* Unit testing và Integration testing
* Test automation và Continuous Integration (CI)
* API documentation
* OpenAPI (Swagger)

**Chương 6: Các chủ đề nâng cao trong API**

* Caching và Throttling
* Realtime API với WebSocket và Socket.IO
* GraphQL API với Apollo Server
* Microservices và API Gateway
* Security Best Practices
* Scaling và Monitoring

Version 4

Roadmap theo Workshop

* Cài đặt và tạo dự án NodeJS trên Windows 11
* Cài đặt và tạo dự án NodeJS trên MacOS Ventura
* Tạo project mới trong NodeJS
* Hàm và module trong Javascript
* Làm việc với thư viện ExpressJS
* Biến môi trường với file .env
* Xử lý Request/Response trong Express
* Sử dụng nodemon để không phải chạy lại code mỗi lần thay đổi
* Tùy biến Router trong Express
* Test các Request với Postman
* Debug các request và kiểm tra các tham số truyền đến NodeJS App
* Viết request Login, gửi tham số qua Request Body
* Validate các thông tin đầu vào với Express Validator
* Controllers và cách tổ chức Controllers trong dự án NodeJS
* Trả về dữ liệu dạng JSON kèm HTTP Status Code
* Tổ chức các Repositories để lấy dữ liệu dưới Database
* Sử dụng Event Emitter trong Nodejs
* MongoDB và kết nối với NodeJS
* 2:48:51 - Xử lý error và tạo class Exception riêng
* Giới thiệu về ORM, Model và validate Model trong Mongoose
* Viết request đăng ký người dùng - Register
* Mã hóa mật khẩu với bcrypt
* Xây dựng Request đăng nhập User, lưu dữ liệu vào MongoDb
* Giới thiệu về JWT - Json Web Token
* Cài đặt Json Web Token và tích hợp vào dự án Nodejs
* Thêm Authentication Middleware có JWT
* Thêm JWT vào header, validate JWT cho các Request
* Hoàn thiện chức năng insert Student
* Sử dụng thư viện Faker để seeding data
* Chức năng Paging(phân trang), kèm Filter(lọc) danh sách sinh viên dài
* Chức năng lấy thông tin chi tiết 1 Student
* Chức năng cập nhật thông tin student