Công cụ:

Chạy ASP.Net

Restful Web Api

Postman (để test api)

Docker(chọn database [mysql])

Thấy còn thiếu gì có thể thêm vào

* Giới thiệu tổng quan

**REST** (REpresentational State Transfer) là một kiểu kiến trúc sử dụng trong việc giao tiếp giữa máy khách và máy chủ web. Kiến trúc REST rất phổ biến nhờ sở hữu tính năng đơn giản, được xây dựng dựa trên các hệ thống và tính năng hiện có của HTTP, giao tiếp thông qua mạng internet để đạt được các mục tiêu của nó, thay vì tạo ra các tiêu chuẩn, framework và công nghệ mới.

https://bizflycloud.vn/tin-tuc/tin-hieu-rest-representational-state-transfer-va-soap-nen-tang-phat-trien-cua-apis-restful-20190331233216107.htm

Rest là gì?

REST là một kiểu kiến ​​trúc cung cấp các standard giữa các hệ thống máy tính trên web, giúp các hệ thống giao tiếp với nhau dễ dàng hơn, trong đó client-server giao tiếp bằng phương thức make request và send response.

<https://www.codecademy.com/article/what-is-rest>

Restful API là gì?

RESTful API (Representational State Transfer API) là kiến trúc REST cho APIs được sử dụng giao thức HTTP để truyền tải dữ liệu giữa server - client hoặc các ứng dụng web.

<https://stringee.com/vi/blog/post/restful-api-la-gi>

So sánh với ai đây? Cho xin cái ý nào

* Kiến trúc và các thành phần

A diagram of a flowchart

Description automatically generated

<https://makingloops.com/restful-api-visual/>

**+ API Global Handlers and Filters**

Các handler và filter là một dạng middleware mà có thể sử dụng để áp dụng logic vào việc log request và gửi response. Các handler và filter có thể được áp dụng toàn cục (áp dụng cho mọi yêu cầu API) hoặc cụ thể cho một bộ điều khiển hoặc phương thức.

+ **Log Request and Response Handler**

Là nhật ký ghi lại các thông tin quan trọng vè request và response. Bằng cách sử dụng DelegatingHandler, có thể kiểm tra và ghi lại mọi thông điệp gửi đến hoặc đi qua hệ thống. Điều này rất hữu ích cho việc theo dõi, kiểm tra và gỡ lỗi hệ thống, đảm bảo mọi thứ đang hoạt động như mong đợi.

+ **Filters**

Các filter toàn cục là một tập hợp chức năng cho việc xác thực và ủy quyền của các API Client. Đảm bảo khách hàng được phép sử dụng API không lãng phí dữ liệu xử lý

+ **Controller & Endpoint-Specific Handlers & Filters**

**Controller** **Specific Handlers & Filters** là các trình xử lý áp dụng chỉ cho một hoặc một số controller cụ thể. Mục đích áp dụng logic hoặc xử lý cho các yêu cầu đối với một controller nhất định mà không ảnh hưởng đến các controller khác.

**Endpoint-Specific Handlers & Filters** làcác trình xử lý áp dụng cho các hành động hoặc phương thức cụ thể. Chúng giúp trong việc các kiểm tra xác thực, kiểm tra tính hợp lệ dữ liệu hoặc các tác vụ khác chỉ đối với một endpoint cụ thể mà không ảnh hưởng đến các endpoint khác trong hệ thống.

**+ Controller Method Body** làquá trình xử lý yêu cầu thêm một đối tượng vào cơ sở dữ liệu một cách an toàn, hiệu quả và đảm bảo tính hợp lệ của dữ liệu.

**api Rest sử dụng HTTP method, gồm:**

**Get(read):** **Trả về một Resource hoặc một danh sách Resource.**

**Post(create): Tạo mới một Resource**

**Put(update): Cập nhật thông tin cho Resource(toàn bộ resource)**

**Patch(update): Cập nhật thông tin cho Resource(một phần resource)**

**delete(delete):Xoá một Resource.**

* Ưu, nhược điểm

**Ưu điểm:**

1. **Đơn giản và dễ hiểu**:
   * REST sử dụng các phương thức HTTP chuẩn như GET, POST, PUT và DELETE, giúp đơn giản hóa giao tiếp và dễ dàng hiểu được cách hoạt động của API.
2. **Không trạng thái (Stateless)**:
   * Mỗi yêu cầu từ khách hàng đến máy chủ phải chứa tất cả thông tin cần thiết để hiểu và xử lý yêu cầu đó. Điều này giúp API dễ dàng mở rộng và bảo trì.
3. **Hiệu suất cao**:
   * Vì REST sử dụng định dạng JSON hoặc XML, dữ liệu được truyền tải nhẹ nhàng hơn, giúp tăng tốc độ xử lý và phản hồi.
4. **Khả năng tương thích rộng rãi**:
   * RESTful API có thể được sử dụng với bất kỳ ngôn ngữ lập trình nào, điều này làm cho nó trở nên rất linh hoạt và dễ tích hợp.
5. **Khả năng mở rộng (Scalability)**:
   * Kiến trúc không trạng thái và sử dụng HTTP làm giao thức giao tiếp giúp RESTful API dễ dàng mở rộng để phục vụ nhiều người dùng hơn.

**Nhược điểm:**

1. **Bảo mật**:
   * RESTful API yêu cầu bảo mật phải được quản lý cẩn thận, vì thông tin nhạy cảm có thể bị lộ nếu không được mã hóa và xác thực đúng cách.
2. **Không phù hợp cho giao tiếp thời gian thực**:
   * REST không hỗ trợ tốt cho các ứng dụng yêu cầu giao tiếp thời gian thực, như chat ứng dụng hoặc trò chơi trực tuyến.
3. **Sự phức tạp khi quản lý các yêu cầu phức tạp**:
   * Khi ứng dụng yêu cầu các thao tác phức tạp và nhiều bước, việc sử dụng REST có thể trở nên phức tạp và khó quản lý.
4. **Thiếu chuẩn hóa**:
   * Mặc dù REST là một kiến trúc phổ biến, nhưng không có tiêu chuẩn chính thức nào, do đó có thể có sự khác biệt trong cách các nhà phát triển xây dựng và sử dụng RESTful API.

* Demo ứng dụng minh họa