**BEST PRATICES RESTFULL API DESIGN**

**Link**

* <https://viblo.asia/p/mot-vai-best-practice-khi-xay-dung-rest-api-3P0lP36PZox>

**Best practices RESTful api design**

* *Sử dụng snake\_case hoặc kebab-case cho URL*

VD, nếu muốn trả về danh sách system order

**BAD:** /systemOrders

**GOOD:** /system-orders hoặc /system\_orders

* *Sử dụng camelCase hoặc snake\_case cho parameter*

VD, muốn trả về thông tin của 1 order

**BAD:** /system-orders/{OrderId}

**GOOD:** /system\_orders/{orderId} hoặc /system\_orders/{order\_id}

* *Sử dụng số nhiều với Collection*

VD, muốn trả về danh sách user

**BAD:** GET /user or GET /User

**GOOD:** GET /users

* *Bỏ động từ ra khỏi URL của Resource:*Không sử dụng động từ để mô tả ý định trong URL. Thay vào đó, sử dụng HTTP method để thể hiện

**BAD:** POST /updateuser/{userId} or GET /getusers

**GOOD:** PUT /users/{userId} # cập nhật user

POST /users # tạo mới user

GET /users/{userId} # trả về thông tin user

* *Thống nhất sử dụng camelCase hoặc snake\_case cho thuộc tính JSON:*Khi chúng ta cần xây dựng API với body request hoặc response là JSON, thì thuộc tính cần được thống nhất chỉ là camelCase hoặc snake\_case

**GOOD:**

|  |
| --- |
| {  user\_name: "Mohammad Faisal"  user\_id: "1"  } |

hoặc

|  |
| --- |
| {  userName: "Mohammad Faisal"  userId: "1"  } |

* *Luôn gửi Parameters limit và offset:* Trong request GET danh sách resource, cần gửi thêm cả limit và offset, vì đây là các tham số hỗ trợ cho phân trang ở client

**GOOD**: GET /shops?offset=5&limit=5

* *Thêm tham số fields trong URL:* Thêm tham số fields để chỉ trả về các trường cần thiết từ API -> Nhờ vậy, sẽ giúp giảm bớt kích thước của response. VD, khi chỉ muốn trả về name, contact, address của shop: **GOOD:** GET /shops?fields=id,name,address,contact
* *Sử dụng HTTP Methods cho các hàm CRUD:* HTTP methods phục vụ cho mục đích giải thích các chức năng CRUD
  + - GET: để trả về thông tin của một resource
    - POST: để tạo một resource mới hoặc các sub resource
    - PUT: để cập nhật các resource có sẵn, nếu resource không tồn tại thì sẽ tạo mới
    - PATCH: để cập nhật resource có sắn, nó sẽ chỉ cập nhật các trường được cung cấp và bỏ qua các trường không được gửi
    - DELETE: xoá các resource đã có
* *Sử dụng Relation cho Nested Resources trong URL*
  + - GET /shops/2/products: trả về danh sách products của shop có id 2
    - GET /shops/2/products/31: trả về thông tin chi tiết của product có id 31 của shop id 2
    - DELETE /shops/2/products/31: xoá product id 31 và product này thuộc shop id 2
    - PUT /shops/2/products/31: cập nhật product id 31 và product này thuộc shop id 2