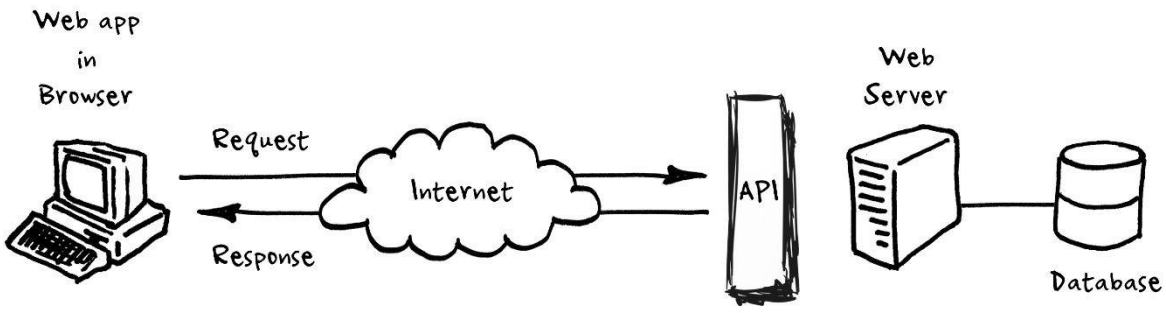
API TESTING VỚI POSTMAN

I. API là gì? Và vì sao phải test API?

1. API là gì?



API là viết tắt của **Application Programming Interface** - giao diện lập trình ứng dụng. Là phương tiện để các ứng dụng hay phần mềm có thể giao tiếp với nhau.

Nói đơn giản, API là cái cầu nối giữa client và server. **Client** ở đây có thể là máy tính, điện thoại sử dụng hệ điều hành khác nhau và được viết bằng những ngôn ngữ khác nhau, ví dụ như Swift, Objective-C, Java.

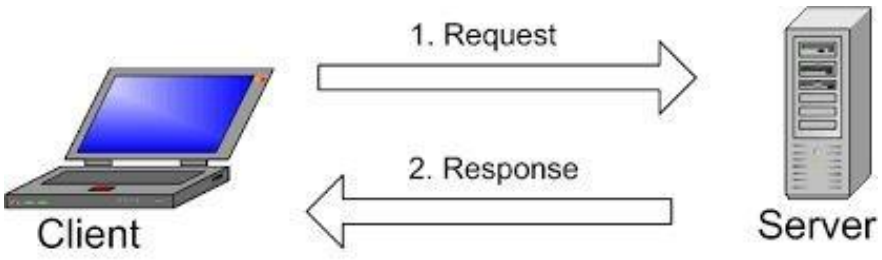
Tương tự, **Server** back-end cũng được viết bằng các ngôn ngữ khác nhau. Để **Client và Server** này có thể nói chuyện được với nhau chúng phải nói cùng 1 ngôn ngữ. Ngôn ngữ ấy chính là **API**.

Ví dụ:

Giả sử bạn là 1 người hướng dẫn viên du lịch, và quản lý 1 nhóm du lịch hợp chủng quốc. Trong nhóm có người Nga, Mỹ, Nhật, Thụy Điển, Đức, Việt Nam. Để có thể làm mọi việc một cách suôn sẻ, tất cả cái nhóm này phải cùng nói 1 ngôn ngữ, có thể là tiếng anh hoặc tiếng Việt. Ở đây người **hướng dẫn viên** sẽ đóng vai trò là **Server**, **người du lịch** sẽ đóng vai trò là **Client**.

Khi đi trên đường hoặc đến thăm địa danh du lịch, những người khách có thể hỏi hướng dẫn viên rất nhiều câu hỏi khác nhau. “Cái kia là cái gì?”, “Quả này ăn như thế nào?”, “Bạn ơi, nhà hàng nào ăn ngon nhỉ?”… Với mỗi một hành động hỏi như vậy, tương ứng với việc gửi 1 **request** được gửi lên **server** với những tham số đầu vào như “Cái kia” hay “quả này”. (**Gửi request còn được gọi là Call API**).

Với mỗi câu hỏi, người hướng dẫn viên sẽ trả lời 1 cách khác nhau – gọi là **Response**.



Định dạng trong việc hỏi và trả lời ở trên có thể thông qua trò chuyện trực tiếp hoặc viết giấy. Ở trong API thì có 2 định dạng chính là XML và JSON.

2. Vì sao phải test API?

Trong quá trình triển khai dự án, phần server và client làm độc lập với nhau nên có nhiều chỗ client chưa làm xong, chúng ta không thể chờ client làm xong để test được dữ liệu mà phải test API bằng công cụ khác –> Lúc này việc test hoàn toàn không phụ thuộc gì vào client.

Kể cả khi client làm xong rồi, nếu test trên client mà thấy lỗi liên quan đến logic và dữ liệu thì cũng cần test thêm API để biết chính xác là server sai hay client sai –> fix lỗi sẽ nhanh hơn.

Khi làm hệ thống web services, dự án chỉ viết API cho bên khác dùng, như vậy sẽ không có client để test giống như các dự án khác –> phải test API hoàn toàn.

II. Protocol dùng trong Restful API:

Phần I đã giải thích vai trò của API đối với client và server, phần này sẽ nói về cách mà Client và Server nói chuyện với nhau.

1. Protocol là gì?

Giả sử: Có 2 người A và B nói chuyện với nhau qua điện thoại, nếu người A hỏi 1 câu rồi im lặng, người B sẽ biết rằng người A đang chờ đợi câu trả lời và đến lượt người B nói. Hai chiếc máy tính cũng giao tiếp 1 cách lịch sự như vậy và được mô tả với thuật ngữ "Protocol" – giao thức.

Giao thức chính là những luật lệ được chấp thuận để 2 máy tính có thể nói chuyện với nhau.

Tuy nhiên, luật lệ này chặt chẽ hơn rất nhiều so với giao tiếp giữa người với người. Máy tính sẽ không thông minh để có thể nhận biết 2 câu "A là chồng B" hay "B là vợ A" có cùng ý nghĩa. Để 2 máy tính giao tiếp hiệu quả, server phải biết chính xác cách mà client sắp xếp message gửi lên như thế nào.
Chúng ta đã từng nghe đến những Protocol cho những mục đích khác nhau, ví dụ như Mail có POP hay IMAP, message có XMPP, Kết nối thiết bị: Bluetooth. Trong web thì Protocol chính là HTTP – HyperText Transfer Protocol, vì sự phổ biến của nó mà hầu hết các công ty chọn nó là giao thức cho các API.

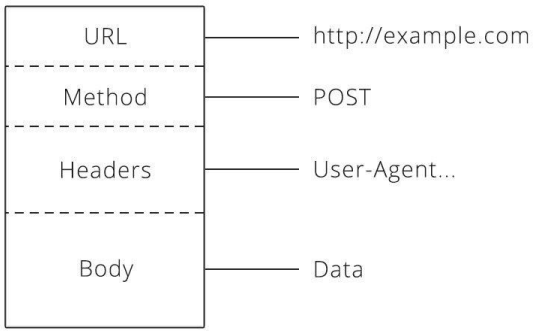
Lưu ý: API có thể viết trên nền Protocol khác, ví dụ như SOAP.

2. HTTP hoạt động như thế nào?

Vòng đời của HTTP xoay quanh vòng luẩn quẩn: **Request và Response**.

Client gửi request, server gửi lại response là liệu server có thể làm được cái client muốn hay không. Và API được xây dựng trên chính 2 thành phần: Request và Reponse. Trước tiên, ta phải hiểu cấu trúc của mỗi thành phần.

Request



Request

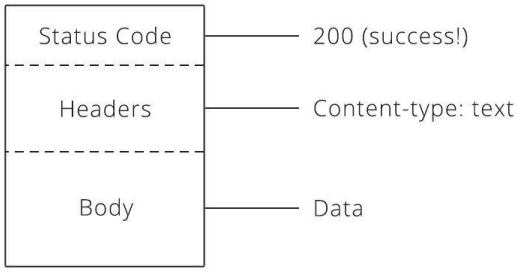
Một REQUEST đúng chuẩn cần có 4 thành phần:
1. URL
2. Method
3. Header
4. Body

Trong đó:

- **URL** là địa chỉ của một tài nguyên duy nhất trên Web. Mỗi URL hợp lệ sẽ trỏ đến một tài nguyên duy nhất, tài nguyên đó có thể là trang HTML, tài liệu, hình ảnh, video, file... Và client sẽ dễ dàng cho server biết cái nó muốn là cái gì, những cái này còn được gọi chung là "resources" – nguồn lực.
- **Method**: là hành động client muốn tác động lên "resources". Có 4 loại Method hay được dùng:
  - GET (SELECT): Yêu cầu server gửi thông tin resource. Ví dụ: bạn vào fb và lướt new feeds.
  - POST (CREATE): Yêu cầu server tạo 1 resource mới. Ví dụ: đăng ký 1 chuyến đi ở GrabBike.
  - PUT (UPDATE): Yêu cầu server sửa/thêm vào resource đã có trên hệ thống. Ví dụ: Edit 1 post trên fb.
  - DELETE (DELETE): Yêu cầu server xóa 1 resourse. Ví dụ: xóa 1 bình luận trên fb.
- **Header**: nơi chứa các thông tin cần thiết của 1 request nhưng end-users không biết được 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 mà response mà client có thể đọc được…
- **Body**: nơi chứa thông tin mà client sẽ điền. Giả sử bạn đặt 1 cái bánh pizza, thì thông tin ở phần body sẽ là: Loại bánh pizza, kích cỡ, số lượng đặt.

Response:

Sau khi nhận được request từ phía client, server sẽ xử lý request đó và gửi ngược lại cho client 1 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 cấu trúc 3 phần:



Response

1. Status code
2. Header
3. Body

- **Status code** là những con số có 3 chữ số và có duy nhất 1 ý nghĩa. Ví dụ: Error "404 Not Found" hoặc "503 Service Unavailable". Full list ở đây. https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
- **Header** và **Body** tương đối giống với request.

Ví dụ API:

- Khi bạn sử dụng một ứng dụng trên điện thoại di động, ứng dụng kết nối Internet và gửi dữ liệu tới máy chủ.
- Máy chủ sau đó lấy ra dữ liệu đó, diễn giải nó, thực hiện các hành động cần thiết và gửi nó trở lại điện thoại của bạn.
- Ứng dụng sau đó giải thích dữ liệu đó và trình bày cho bạn thông tin bạn muốn theo cách có thể đọc được.

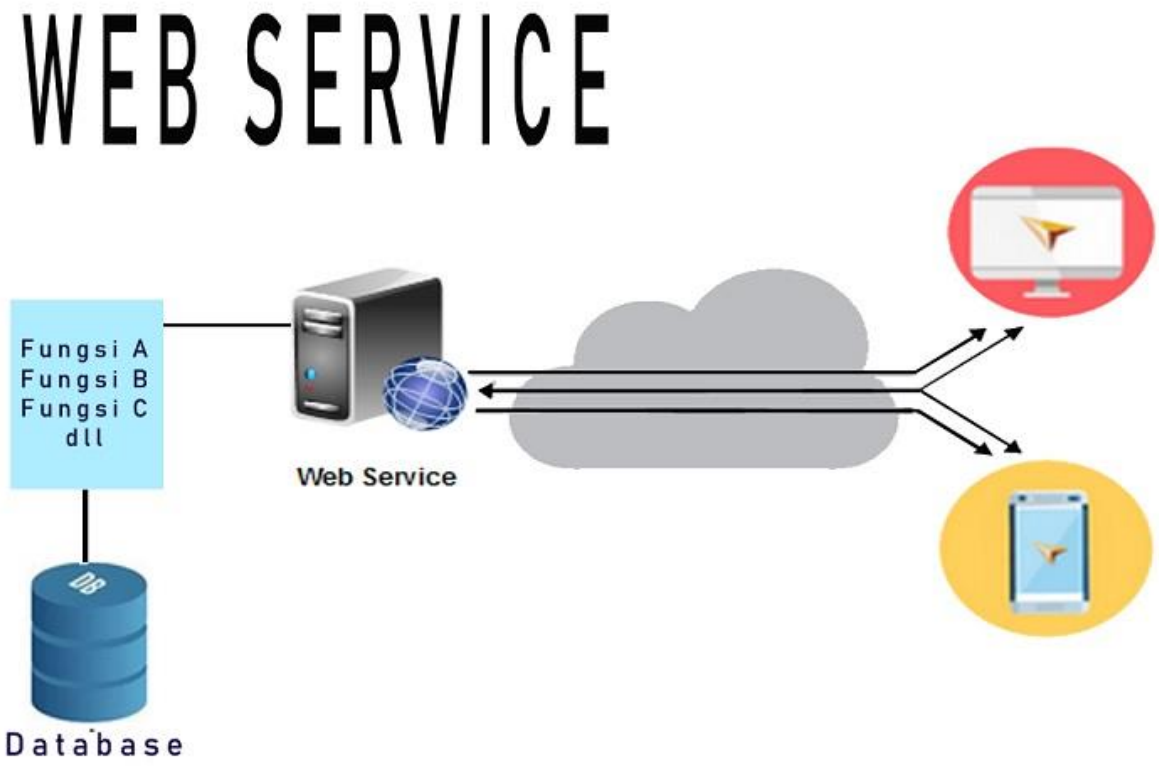
➔ Đây là những gì một API làm – tất cả điều này xảy ra thông qua API.

III. Web services:

1. Định nghĩa:

Web service là một lớp (framework) giữa hai máy tính, giúp hai máy tính có thể tương tác với nhau qua mạng.

Web service bao gồm toàn bộ các giao thức, tiêu chuẩn mở được dùng trong việc liên kết dữ liệu giữa các phần mềm ứng dụng khác nhau, là cầu nối TRAO ĐỔI DỮ LIỆU giữa các phần mềm ứng dụng được xây dựng từ các ngôn ngữ lập trình khác nhau.



Mô hình thể hiện sự kết nối: client (người dùng – máy tính 1) gửi tin nhắn đến server (máy chủ – máy tính 2) và server hồi âm lại tin nhắn đó nhờ có web service. Web service hiện nay đa số giao tiếp qua cơ chế HTTP, nhưng format dữ liệu khi gửi và nhận thì hoàn toàn khác nhau.

2. Sự khác nhau giữa Web service và API:

1. Tất cả Web services là APIs nhưng không phải API nào cũng là web service.

2. Web services không thể thực hiện được tất cả các thao tác mà API sẽ thực hiện.

3. Một Web service sử dụng 3 chuẩn chính: SOAP, REST và XML-RPC trong quá trình giao tiếp, ngược lại API có thể sử dụng bất kỳ chuẩn nào để giao tiếp.

4. Một Web service đòi hỏi luôn luôn phải có mạng để nó hoạt động nhưng API thì không cần.

5. API tạo điều kiện liên kết trực tiếp với một ứng dụng trong khi Web service thì không.

IV.Định dạng dữ liệu JSON và XML:

Định dạng data trong API thường dùng 2 loại chính là JSON (JavaScript Object Notation) và XML (Extensible Markup Language).

1. Định dạng JSON:

JSON được sử dụng nhiều trong Restful API. Nó được xây dựng từ ngôn ngữ Javascript, tương thích với cả front-end và back-end của cả web app và web service. JSON là 1 định dạng đơn giản với 2 thành phần: keys và values.

– **Key** thể hiện thuộc tính của Object

– **Value** thể hiện giá trị của từng Key

Ví dụ:

```
{
    "id": "5",
    "title": "KUNIKO METHOD",
    "created": "2017-04-24 13:48:56"
}
```

Trong ví dụ trên, keys nằm bên trái, values nằm bên phải.

Có nhiều trường hợp, 1 Key sẽ có Value là 1 dãy key + value. Ví dụ như hình:

```
{
    "error_code": 0,
    "error_msg": "",
    "data": {
        "id": "26",
        "file": "files/59017ab52cb10.epub"
    }
}
```

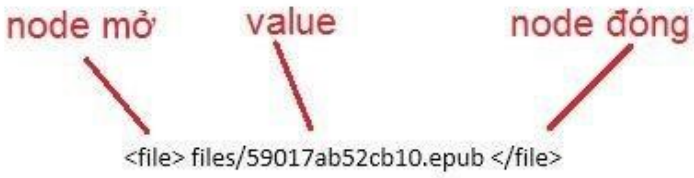
Trong hình trên Key có tên là Data có Value là 2 cặp Key + value.

2. Định dạng XML

Trong JSON dùng { } và [ ] để dánh dấu dữ liệu. XML thì tương tự như HMTL, dùng thẻ để đánh dấu và được gọi là nodes.

Lấy luôn ví dụ ở trên nhưng viết bằng xml, nó sẽ như thế này:

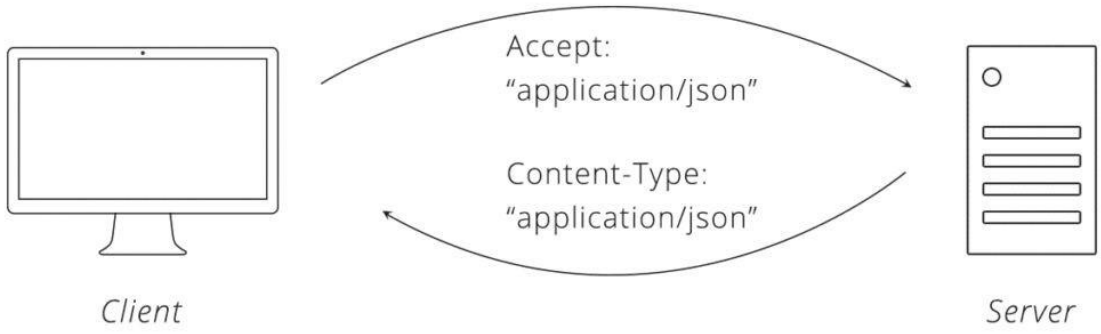
```
<error_code>0</error_code>
<error_msg></ error_msg >
<data>
        <id>26</id>
        <file> files/59017ab52cb10.epub </file>
</data>
```



3. Định dạng dữ liệu sử dụng như thế nào trong HTTP:

Quay lại phần II, phần header có chức năng lưu những thông tin mà người dùng không biết, trong đó có 1 thành phần xác định format của data là: *Content-Type*

Khi client gửi *Content-Type* trong header của request, nó đang nói với server rằng dữ liệu trong phần body của request là được định dạng theo kiểu đó. Khi client muốn gửi JSON nó sẽ đặt *Content-Type* là "application/json". Khi bắt đầu nhận request, server sẽ check cái Content-Type đầu tiên và như thế nó biết cách đọc dữ liệu trong body. Ngược lại, khi server gửi lại client 1 response, nó cũng gửi lại *Content-Type* để cho client biết cách đọc body của response.



Đôi khi client chỉ đọc được 1 loại định dạng, ví dụ là JSON mà server lại trả về XML thì client sẽ bị lỗi. Do đó, 1 thành phần khác ở trong header là *Accept* sẽ giúp client xử lý vấn đề trên bằng cách nói với server loại nó có thể đọc được. Ví dụ: Accept : "application/json" . Chốt lại: dựa vào 2 thành phần CONTENT-TYPE và ACCEPT, client và server có thể hiểu và làm việc một cách chính xác.

V. Giới thiệu chung về Postman:

1. Ưu, nhược điểm của Postman:

Postman là 1 công cụ để test API của cty Postdot Technologies được bắt đầu phát triển từ năm 2012.

Ưu điểm:

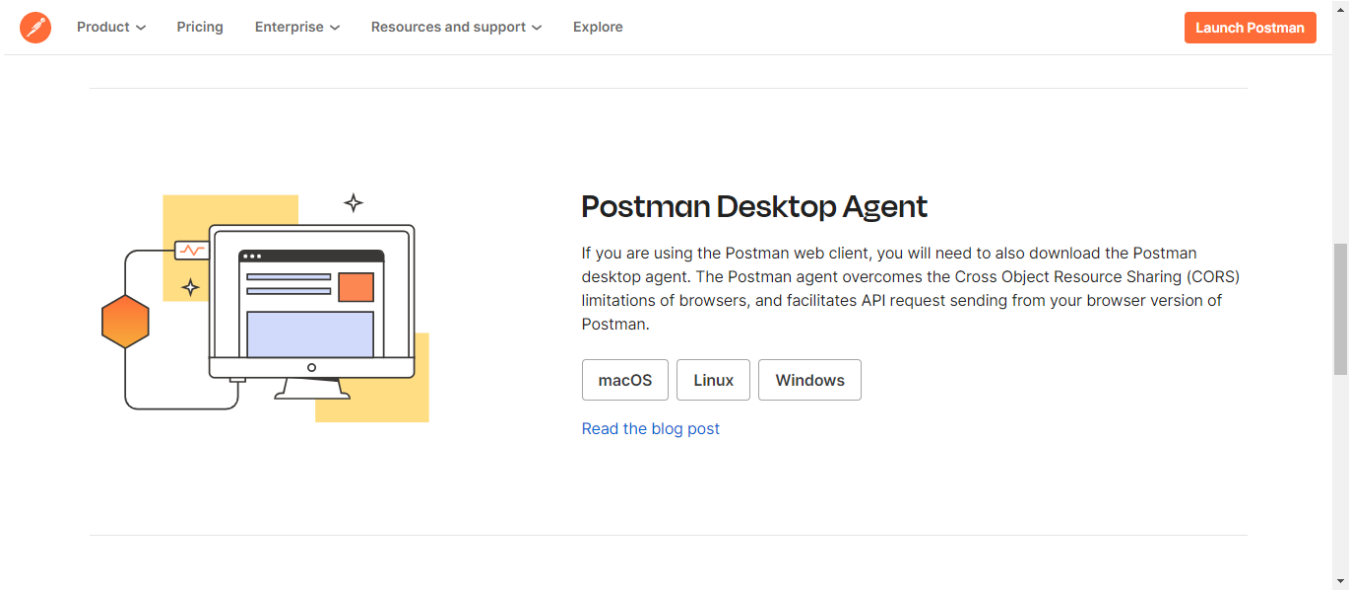
– Dễ sử dụng, hỗ trợ cả chạy bằng UI và non-UI.
– Hỗ trợ viết code cho assert tự động bằng Javascript.
– Hỗ trợ cả RESTful services và SOAP services.
– Có chức năng tạo API document.

Nhược điểm:

– Những bản tính phí mới hỗ trợ những tính năng advance: Làm việc theo team, support trực tiếp…

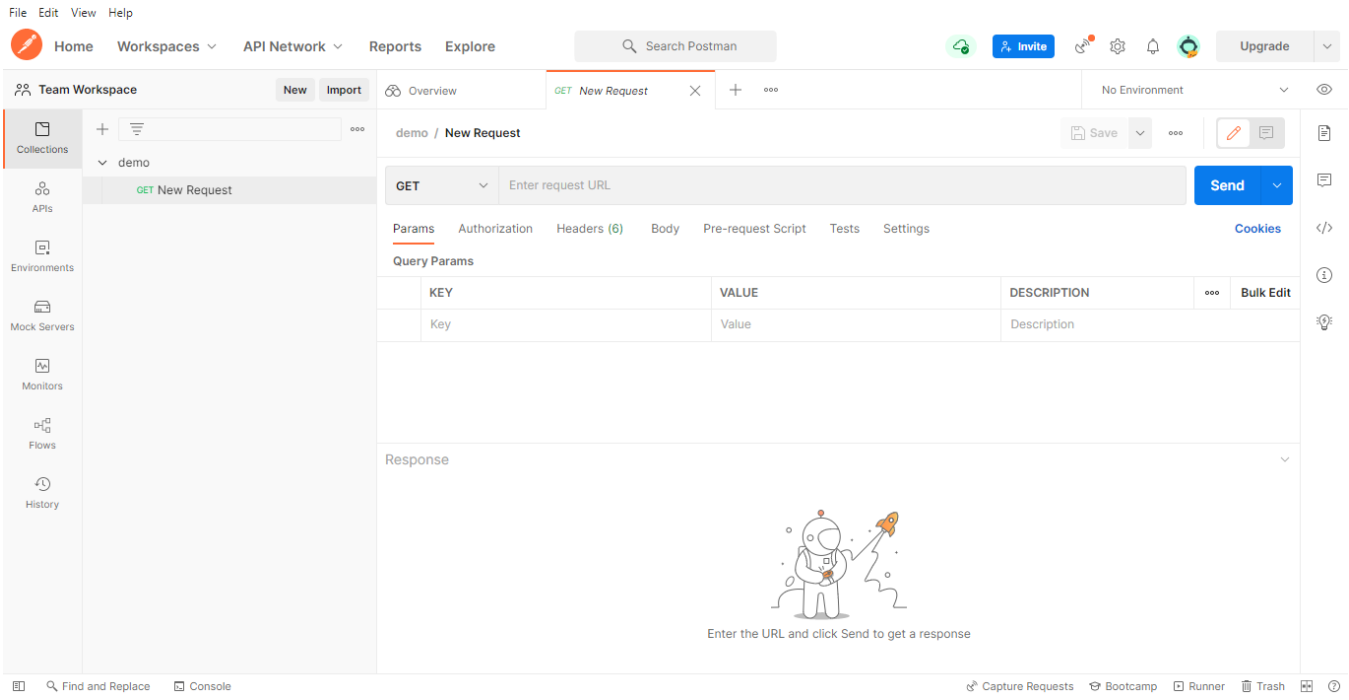
2. Cài đặt Postman:

Download tại địa chỉ: https://www.postman.com/downloads/

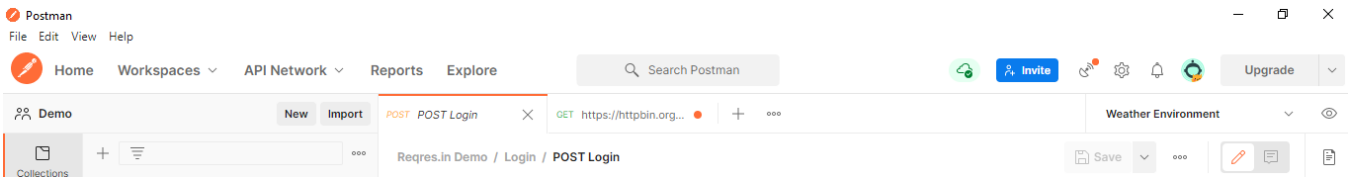


Sau đó, cài đặt như 1 phần mềm bình thường.

3. Giao diện của Postman:

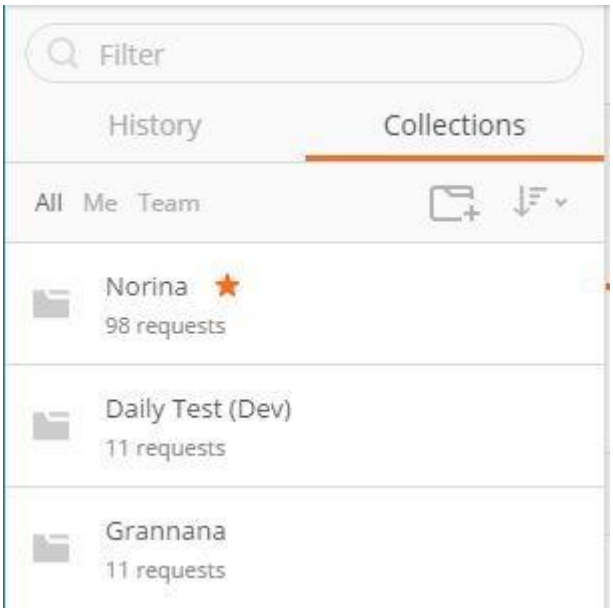


- **Settings**: chứa các thông tin về cài đặt chung.

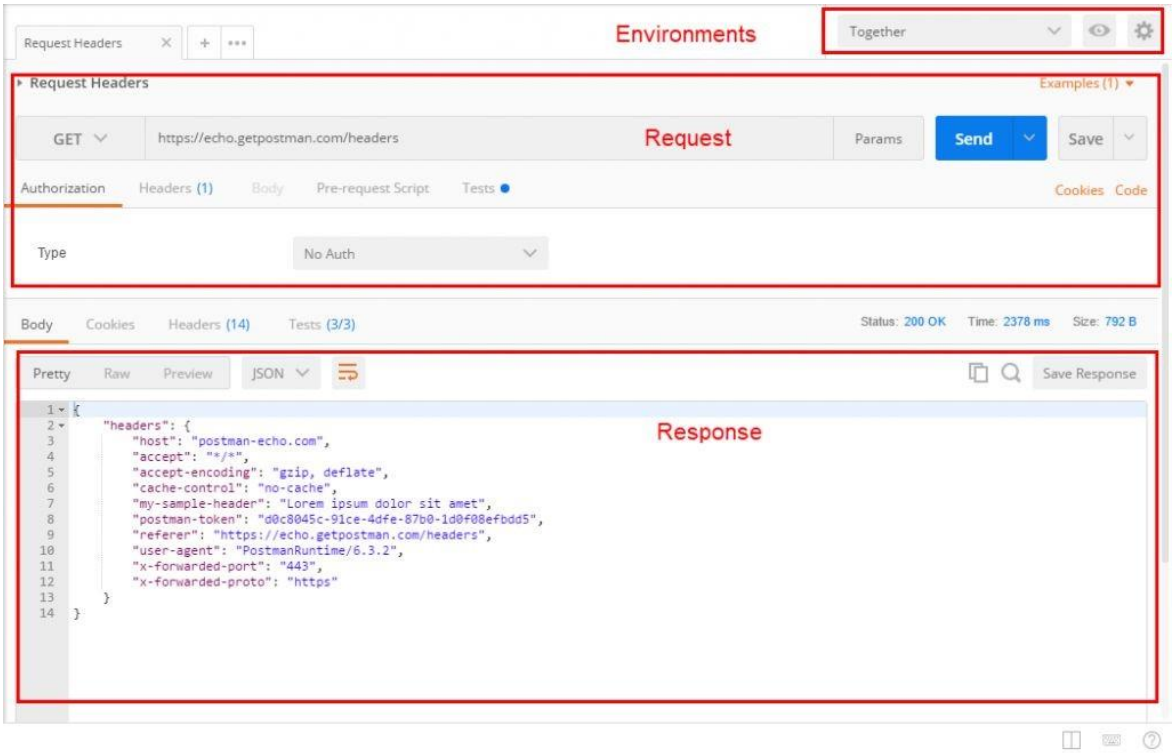


▪ Thông tin Account: dùng để Login, logout và sync data.
▪ Settings tùy chỉnh: themes, shortcut, format…

- **Collections**: lưu trữ thông tin của các API theo folder hoặc theo thời gian.



- **API content**: hiển thị nội dung chi tiết API và các phần hỗ trợ giúp thực hiện test API. Đây là phần mà tester phải làm việc nhiều nhất.



Trong phần này gồm có 3 thành phần chính:

- ***Enviroments*:** Chứa các thông tin môi trường. Ví dụ: cần làm 1 dự án nhưng có 3 môi trường khác nhau: dev, staging và product. Có phần này, chúng ta có thể nhanh chóng đổi sang môi trường cần test mà không phải mất công đổi URL của từng request. (Sẽ được nói rõ hơn ở những chương sau)

- ***Request*:** Phần chứa các thông tin chính của API, có thể đọc lại phần II.
- ***Reponse:*** Chứa các thông tin trả về sau khi send Request.

VI. Cách tạo Request:

Khi làm việc với API, chúng ta chỉ làm việc với 2 dạng API chính là GET và POST.

– *GET: Yêu cầu server đưa lại resource: Ví dụ: vào fb, vuốt newfeeds.*

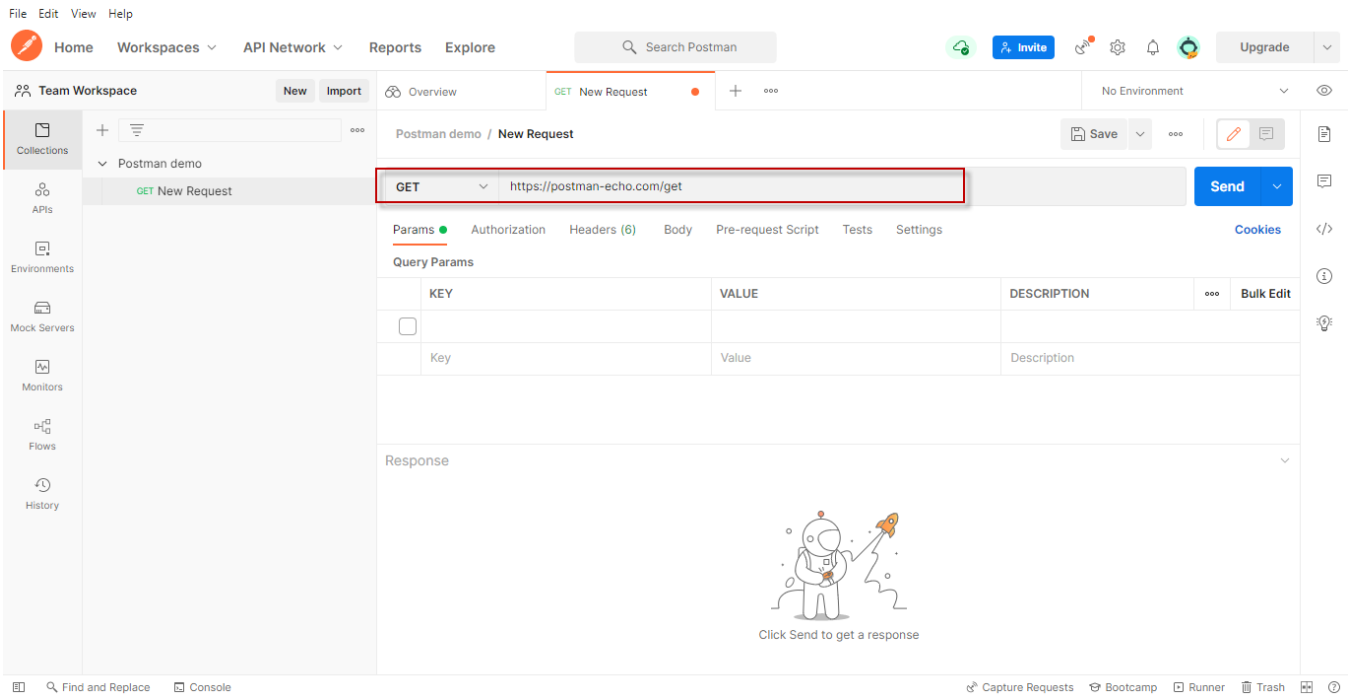
– *POST: Yêu cầu server cho tạo ra 1 resource mới. Ví dụ: đăng ký 1 chuyến đi ở GrabBike.* Và một request gồm có 4 thành phần:

1. URL
2. Method
3. Headers
4. Body

Khi vào dự án, những thông tin trên sẽ lấy từ developer. Muốn test được API thì phải có API documents. Cái này tùy công ty sẽ có chuẩn và mẫu riêng, nhưng mà nhìn chung thì phải cung cấp đủ các thông tin sau: Tên API, mục đích sử dụng, Method, URL, Params, Sample Request, Sample Response.

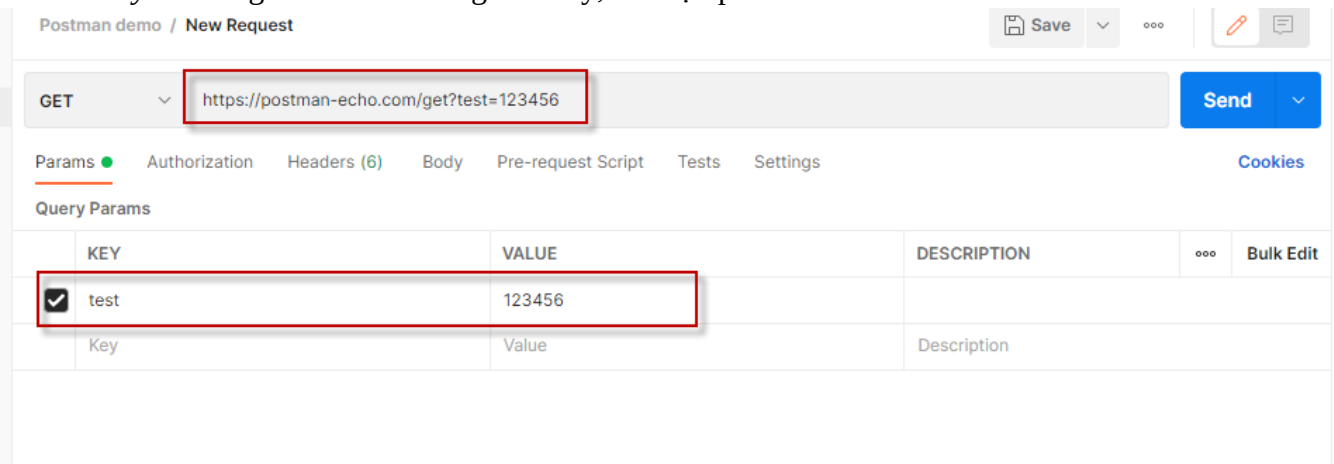
1. Tạo request GET:

Ví dụ: dùng API mẫu của Postman cung cấp..



1. *URL*: https://postman-echo.com/get

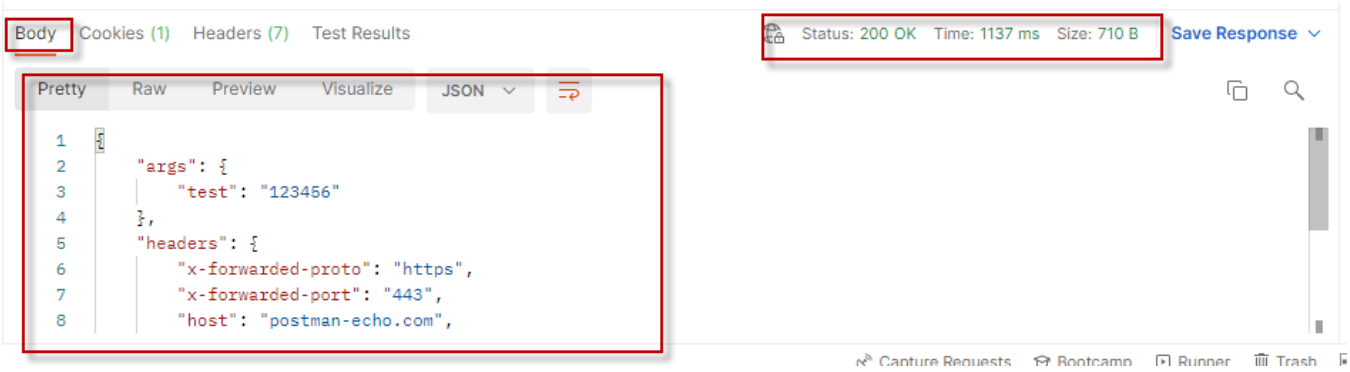
2. *Method*: GET
3. *Headers*: Không cần điền gì cả
4. *Body*: Phương thức GET không có body, các bạn phải điền tham số vào **Params**



Lưu ý:

- Tất cả các Params truyền vào phải chính xác, không được để thừa 1 khoảng trống hay xuống dòng.
- Tham khảo link JSON Editor để convert data sang JSON cho dễ review: https://jsonformatter.org/json-editor

Sau khi điền đầy đủ thông tin thì ấn SEND để gửi request và chờ response trả về.



Thông tin trả về sẽ có mấy điểm cần quan tâm:

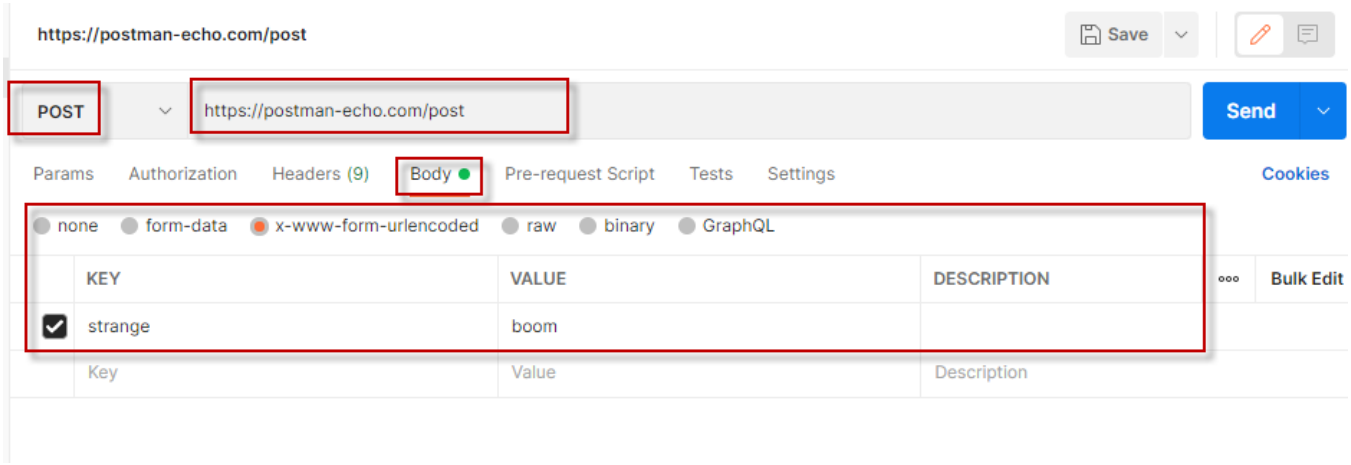
1. Định dạng dữ liệu trả về: thông thường là JSON và nên để chế độ Pretty để cho dễ nhìn.

2. Nội dung dữ liệu: Đây là phần bạn phải kiểm tra.
– Bạn so sánh với cái Sample Response ở API docs để xem *cấu trúc* trả về đã đúng hay chưa.
– Value của từng key đã đúng chưa, so sánh với nội dung trong DB. (không có DB là không làm được API testing).

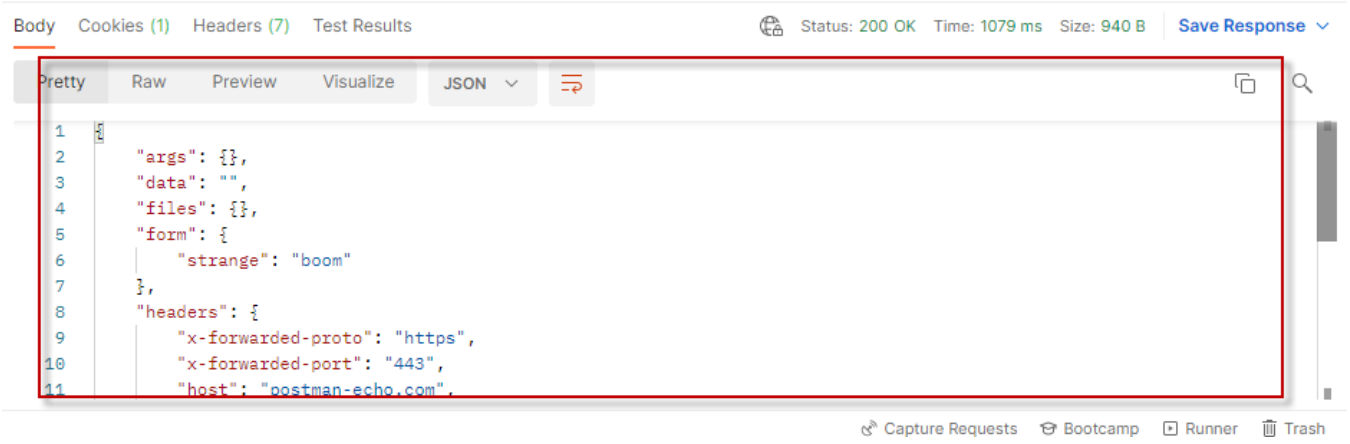
3. Trạng thái của API (status) và thời gian trả về.
Lưu ý: thời gian chạy API bằng Postman luôn ngắn hơn thời gian test trên giao diện Mobile vì nhiều lý do: đường truyền internet ở máy tính ổn định hơn wifi, và sau khi nhận response thì Mobile phải chạy code khởi tạo giao diện để hiển thị.

2. Tạo request POST:

Tương tự như trên, chỉ khác là điền tham số vào trong Body:



Và phần response cũng như vậy



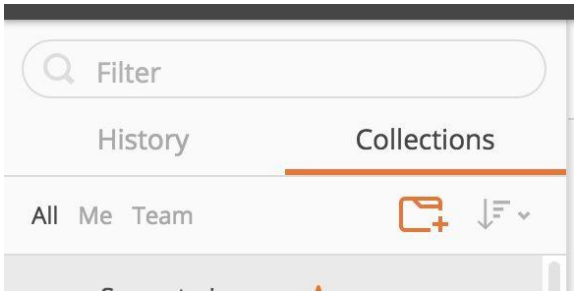
VI. Collections trong Postman:

Hiểu nôm na, nó chính là Folder, giúp đóng gói những request vào chung 1 chỗ.
Ưu điểm của việc sử dụng Collection trong Postman:

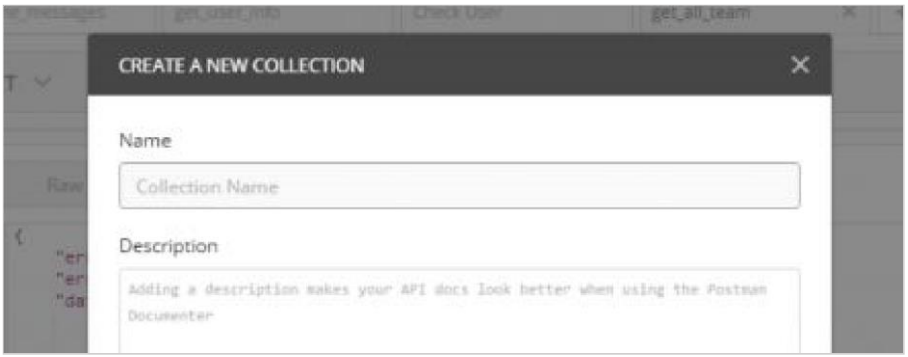
- Không cần dùng History để tìm lại những request đã tạo.
- Sử dụng được chức năng tạo API documents tự động mà Postman cung cấp
- Sử dụng được chức năng Runner, giúp chạy liên tục các Request.

1. Cách tạo Collection.

- Click vào button [tạo collection] bên sidebar

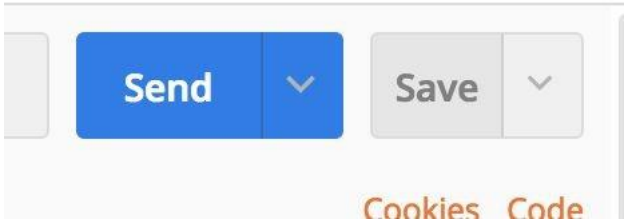


- Điền tên và mô tả (không bắt buộc) collection đó.



- Lưu request vào Collection.

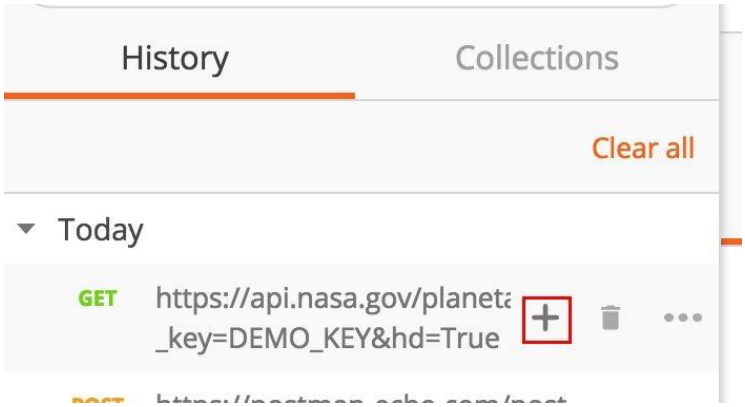
Bước 1 . Tạo ra 1 new Request (Như phần trước)
Bước 2 . Ấn nút Save



Bước 3 . Chọn Collection cần lưu và Save tiếp.

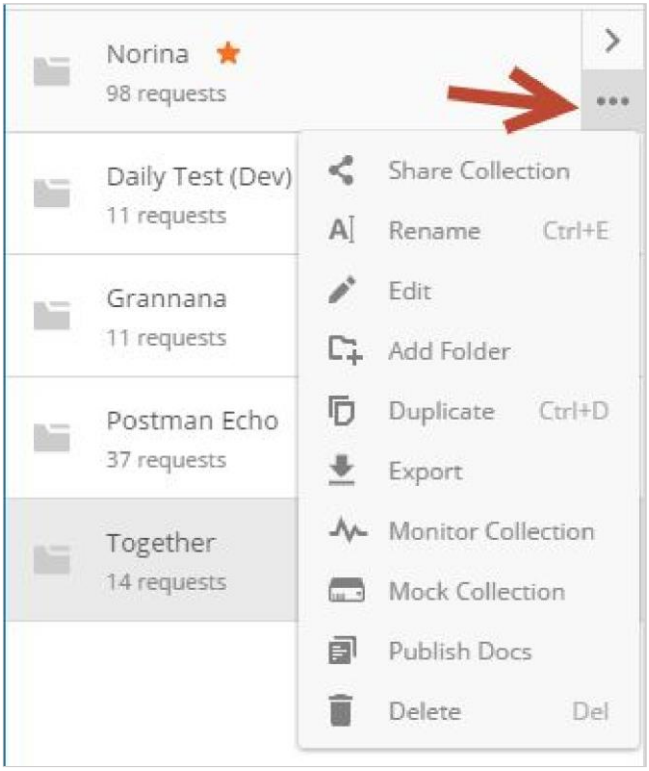
Note: Với những trường TH muốn add request từ History vào Collection.

Bước 1 . Click vào icon **(+)**



Bước 2 . Chọn Collection cần lưu và Save.

Các setting chính của Collection:



- ***Share collections****:* tạo ra link để share với người khác collection (bị hạn chế bởi kiểu account).
- ***Rename***: Đổi tên của collection.
- ***Edit***: Sửa tên và mô tả của collection.
- ***Add Folder***: tạo thêm collection mới bên trong Collection đó.
- ***Duplicate***: nhân đôi collection đang có.
- ***Export***: Xuất collection ra dạng file .json
- ***Monitor Collection***: Dùng để test hiệu năng (bị hạn chế bởi kiểu account).
- ***Mock Collection***: giúp giả lập các API sử dụng chức năng Example mà postman hỗ trợ. (bị hạn chế bởi kiểu account).
- ***Publish Docs***: Tạo ra API Docs định dạng HTML.
- ***Delete***: Xóa Collection.

Thực hành: Sử dụng các link này: https://any-api.com VÀ https://reqres.in/ để thực hành các thao tác GET, POST, PUT, DELETE dữ liệu thông qua API Postman.