**REACT 13: NETWORKING (AXIOS MODULE)**

A – LÝ THUYẾT

I – HTTP REQUEST & RESPONSE

**1. Giao thức HTTP**

*HTTP – là viết tắt của Hyper Text Transfer Protocol, là một giao thức được thiết kế và hoạt động theo kiểu Client (máy khách) và Server (máy chủ), với cơ chế hoạt động sẽ là Client gửi yêu cầu (Request) tới Server, sau đó Server sẽ phản hồi (Response) ngược lại*

**2. HTTP Request**

HTTP Request được gọi là một yêu cầu gửi từ Client lên Server, yêu cầu Server thực thi một công việc nào đó.

VD: Click vào menu iPhone để xem tất cả sản phẩm thuộc danh mục iPhone, hay bấm vào nút Đăng nhập để hoàn thành việc gửi thông tin mà bạn vừa nhập vào Form trước đó,…

Để một Client gửi được Request lên Server thì nó cần phải được thực thi thông qua các phương thức truyền dữ liệu, về mặt truyền dữ liệu một cách truyền thống thì có 2 dạng truyền dữ liệu chính, đó là truyền dữ liệu thông qua việc click vào các liên kết (sử dụng phương thức truyền dữ liệu GET) và truyền dữ liệu thông qua các Form (sử dụng phương thức truyền dữ liệu POST). Tuy nhiên để đi sâu hơn và phức tạp hơn nữa thì chúng ta sẽ phải hiểu chính xác và làm việc với đầy đủ các phương thức truyền dữ liệu một cách chuẩn nhất bao gồm POST, GET, PUT, DELETE,… và chúng ta sẽ gặp lại trong phần RESTful API

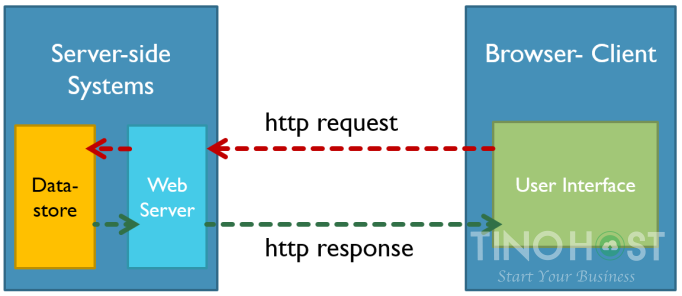
**3. HTTP Response**

HTTP Response được gọi là một phản hồi từ Server về cho Client

Ví dụ: Click vào menu iPhone thì bạn thấy được tất cả các sản phẩm thuộc iPhone, kết quả bạn nhìn thấy được đó chính là do Server đã Response dữ liệu về để hiển thị trên Web Browser sau hành động click vào menu để gửi một Request

**4. Mô hình và luồng dữ liệu trong HTTP Request và Response**

Để có thể hiểu rõ hơn về luồng dữ liệu và các xử lý trong hoạt động kiểu HTTP Request và Response, các bạn có thể tham khảo một hình ảnh trực quan dưới đây



II – REST API

**1. Rest API là gì**

*REST là viết tắt của Representational State Transfer. Nó là một kiểu kiến ​​trúc cho việc xây dựng các ứng dụng có kết nối dữ liệu. Nó sử dụng giao thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE,… đến một URL để xử lý dữ liệu.*

**Ví dụ:**

|  |
| --- |
| // Gửi yêu cầu GET thông thường  deleteuser?Id=10  // Gửi yêu cầu GET theo REST  DELETE/user/10 |

**2. Rest API bên thứ 3**

Trong Backend chúng ta thường phải viết API cho các ứng dụng web và mobile (Rest API). Tuy nhiên không phải lúc nào chúng ta cũng có sẵn các Backend API này, giải pháp nhanh và đơn giản đó là sử dụng một Rest API mẫu làm Demo từ một bên khác.

<https://jsonplaceholder.typicode.com/>

***Chú ý:***

*Để sử dụng và phân tích một Rest API (cho dù là thật hay giả) thì chúng ta có rất nhiều cách, có thể sử dụng Javascript thuần, Package của React,… tuy nhiên phần tiếp theo chúng ta sẽ tìm hiểu một công cụ test Rest API cực kỳ tuyệt vời đó là POSTMAN*

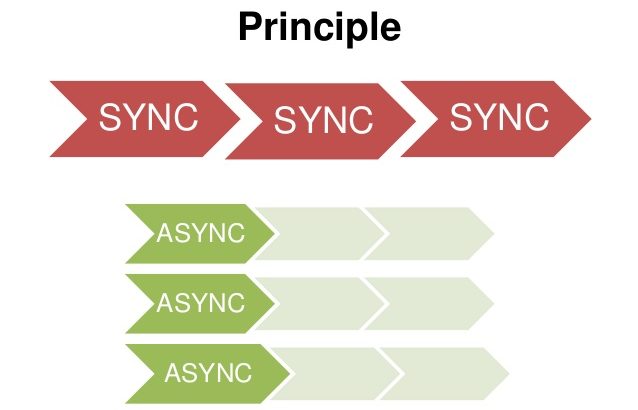
III – ĐỒNG BỘ & BẤT ĐỒNG BỘ TRONG JAVASCRIPT

**1. Xử lý đồng bộ (Synchronous)**

***Synchronous*** *(đồng bộ / tuần tự) tức là code chương trình sẽ chạy tuần tự từ trên xuống dưới. Khi nào lệnh trên hoàn thành thì lệnh dưới mới được chạy.  Đây là cách viết code rất thường dùng.*

**2. Xử lý bất đồng bộ (Asynchronous)**

**Asynchronous** (bất đồng bộ/bất tuần tự): tức là code chương trình không hẳn tuần tự nữa, nhiều lệnh có thể thực hiện cùng lúc. Có khi lệnh dưới cho kết thúc và cho kết quả trước cả lệnh phía trên.



Xử lý đồng bộ & bất đồng bộ

IV – AXIOS MODULE

**1. Tổng quan về Axios**

**a. Axios là gì**

*Axios là một HTTP client được viết dựa trên Promises được dùng để hỗ trợ cho việc xây dựng các ứng dụng API từ đơn giản đến phức tạp*

**b. Cài đặt**

Bật command chạy câu lệnh sau

|  |
| --- |
| npm install axios --save |

**c. Sử dụng**

Để sử dụng axios chúng ta đơn giản chỉ cần import vào là xong

**Code:**

|  |
| --- |
| import axios from "axios" |

**2. Các thuộc tính và phương thức hỗ trợ trong Axios**

**a. Cú pháp sử dụng axios**

**Code:**

|  |
| --- |
| axios({  method: "GET",  url: "http://localhost:3000/courses",  data: {}  }) |

**Trong đó:**

* method: phương thức truyền dữ liệu: POST / GET / PUT / DELETE / REQUEST / OPTION / HEAD/ PATCH
* url: đường dẫn tới Rest API
* data: thường lưu trữ dữ liệu trả về từ một Rest API

**b. Response Axios Async**

**Response Axios Async** là quá trình sử dụng Axios lấy dữ liệu từ API và trả về theo cách thức xử lý bất đồng bộ

Khi chúng ta tạo một Request thì Axios sẽ trả về một Promise cho dù két quả của Request đó là Responsive hay Error

**Code:**

|  |
| --- |
| import React from "react"  import axios from "axios"  const App = () => {  const fetchData = () => {  axios({  method: "GET",  url: "https://jsonplaceholder.typicode.com/users",  data: {}  }).then((response) => {  console.log(response.data)  }).catch((error) => {  console.log(error)  })  }  fetchData()  return (  <h1>Axios</h1>  )  }  export default App |

**Trong đó:**

* **then**: phương thức nhận data trả về từ Promise thông qua tham số response truyền vào
* **catch**: phương thức nhận báo lỗi trả về thông qua tham số error truyền vào
* **response**: tham số truyền vào để nhận dữ liệu trả về từ Server
* **data:** thuộc tính của đối tượng response lưu trữ tất cả dữ liệu trả về từ Server ở dạng mảng
* **error**: tham số truyền vào để nhận cảnh báo lỗi trả về từ Server

***Chú ý:*** *Chúng ta còn rất nhiều các phương thức xử lý cho cả Request và Response, nhưng sẽ tìm hiểu sâu và thực hành với nó khi đến phần làm dự án*

**c. Response Axios Sync**

**Response Axios Sync** là quá trình sử dụng Axios lấy dữ liệu từ API và trả về theo cách thức xử lý đồng bộ

**Code:**

|  |
| --- |
| import React from "react"  import axios from "axios"  const App = ()=>{  const fetchData = async ()=>{  const response = await axios("https://jsonplaceholder.typicode.com/users")  console.log(response.data)  }  fetchData()  return (  <h1>Axios</h1>  )  }  export default App |

**Trong đó:**

* **async**: là từ khóa được khai báo bắt buộc ngay hàm cha
* **await**: là từ khóa được khai báo ngay trước hàm mà chúng ta muốn dữ liệu của hàm đó được return trực tiếp cho hàm theo kiểu xử lý đồng bộ
* **response**: tham số truyền vào để nhận dữ liệu trả về từ Server
* **data:** thuộc tính của đối tượng response lưu trữ tất cả dữ liệu trả về từ Server ở dạng mảng

V – TEST API BẰNG CÔNG CỤ POSTMAN

**1. Giới thiệu**

*Postman là một công cụ được sử dụng để test API, với rất nhiều ưu điểm như: cực kỳ phổ biến, dễ sử dụng, hỗ trợ chạy cả UI và non-UI, hỗ trợ cả Restful Services và SOAP Services. Đặc biệt có chức năng tạo API Document rất hay*

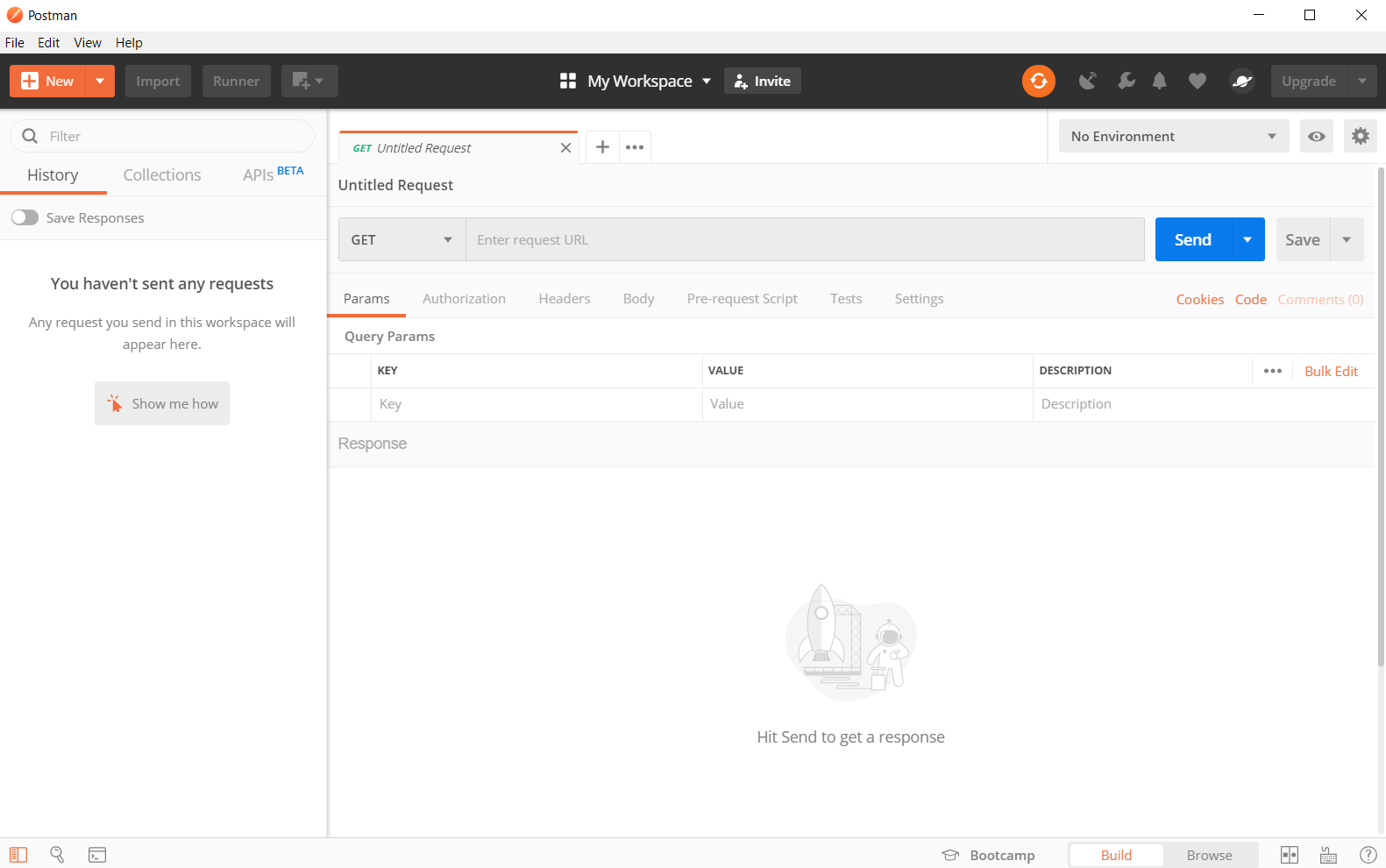
**2. Cài đặt**

B1. Lên trang chủ của postman để download: <https://www.getpostman.com/>

* Chú ý chọn đúng hệ điều hành (Win, Mac, Linux)
* Chú ý chọn đúng phiên bản 32 hoặc 64 cho Win

B2. Cài đặt bình thường

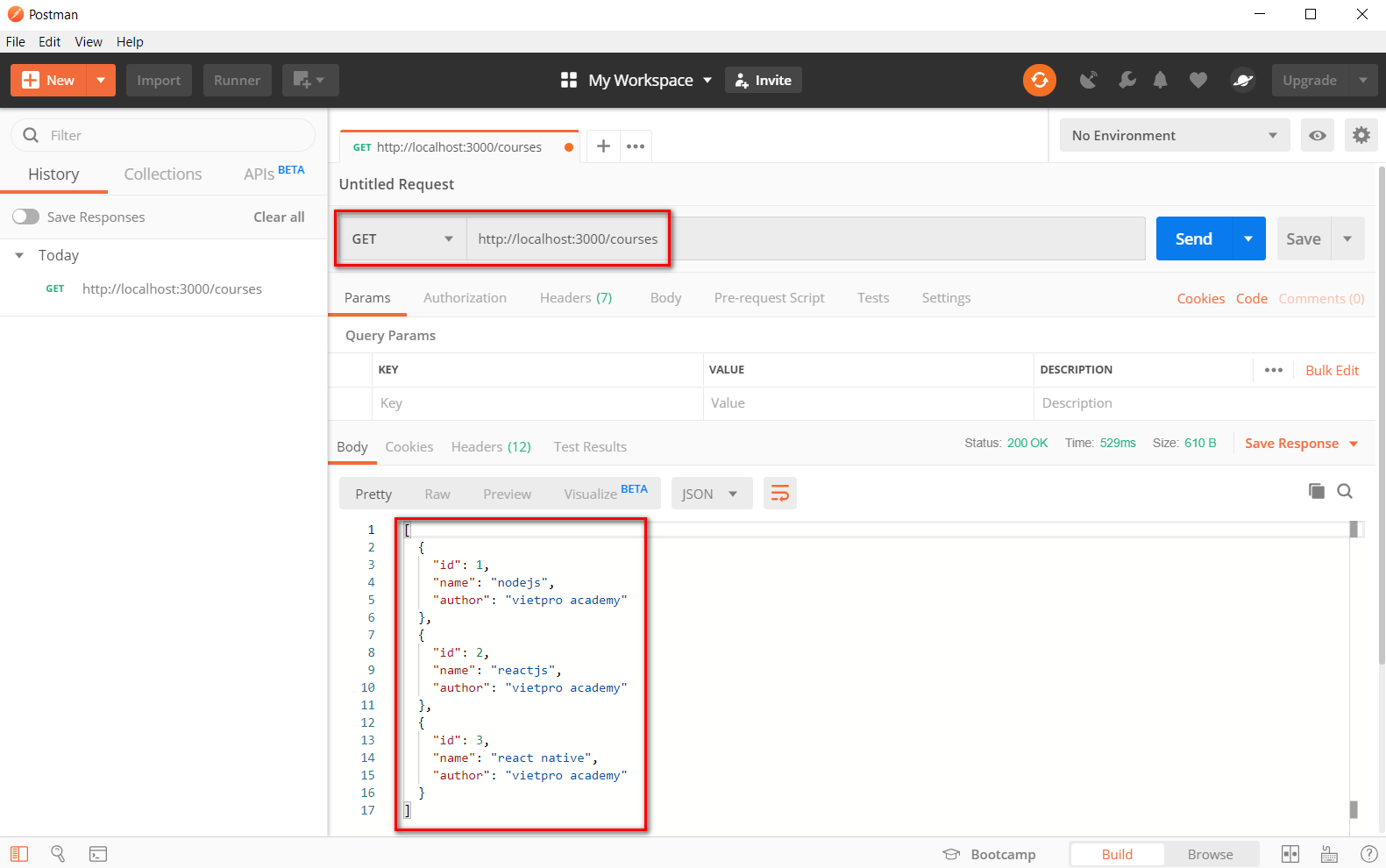
* Phải tạo tài khoản mới cho cài đặt (nên sử dụng hình thức đăng nhập để tiến hành cài đặt ngay bằng tài khoản Gmail)
* Trong quá trình cài đặt, công cụ sẽ yêu cầu khởi tạo và khai báo khá nhiều thông số, nếu có thời gian, tốt nhất là các bạn nên hoàn thành hết các bước này
* Các bạn cũng có thể chọn bỏ qua hoặc khai báo sau để tiến hành hoàn thành cài đặt nhanh (chỉ mất khoảng chưa đến 10s)

**

*Màn hình cài đặt thành công*

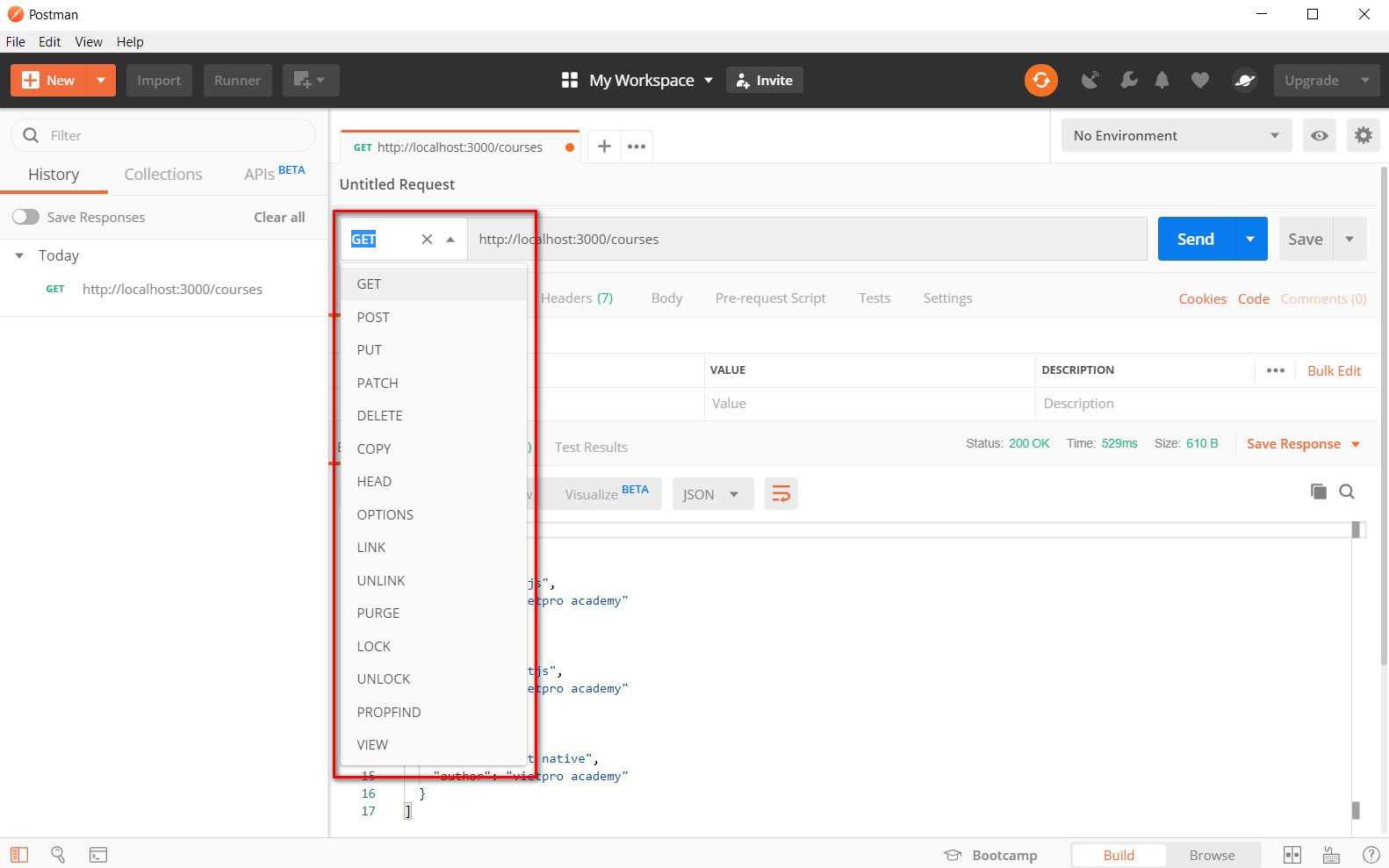
**3. Hướng dẫn test Rest API trên Postman**

Chọn phương thức GET, sau đó điền đường dẫn, và cuối cùng bấm send để xem kết quả



***Chú ý:***

* *Ngoài phương thức GET, chúng ta còn rất nhiều phương thức xử lý Rest API khác nữa, tôi sẽ hướng dẫn đầy đủ và chi tiết về Postman khi chúng ta đi sâu vào các dự án của ReactJS và React Native*

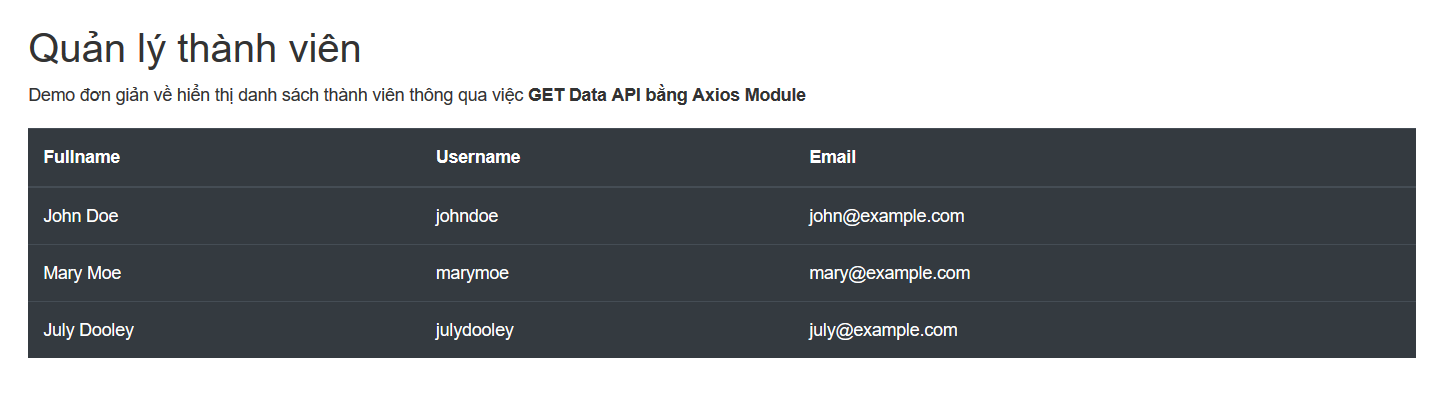


B – THỰC HÀNH

I – BÀI TẬP THỰC HÀNH

**Bài 1:**

Hiển thị danh sách thành viên từ API và giao diện có sẵn theo Demo dưới đây. Data thành viên được lấy từ API Users trên trang sau <https://jsonplaceholder.typicode.com/>



II – BÀI TẬP VỀ NHÀ

**Bài 2:**

Nâng cấp bài tập số 1 phần thực hành ở trên với việc tách ra thành các Component và sử dụng React Hook (nếu có thể)

**Bài 3:**

Nâng cấp bài tập số 2 ở trên với việc sử dụng Redux