KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH**

**HỌC KỲ I, NĂM HỌC 2023 – 2024**

**NGHIÊN CỨU CÁCH SỬ DỤNG POSTMAN ĐỂ KIỂM TRA RESTFUL API**

*Giáo viên hướng dẫn:*ThS. Ngô Thanh Huy

*Sinh viên thực hiện:*

Họ tên: Trần Văn Sang

MSSV: 110121094

Lớp: DA21TTB

***Trà Vinh, tháng 12 năm 2023.***

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH**

**HỌC KỲ I, NĂM HỌC 2023 – 2024**

**NGHIÊN CỨU CÁCH SỬ DỤNG POSTMAN ĐỂ KIỂM TRA RESTFUL API**

*Giáo viên hướng dẫn:*ThS. Ngô Thanh Huy

*Sinh viên thực hiện:*

Họ tên: Trần Văn Sang

MSSV: 110121094

Lớp: DA21TTB

***Trà Vinh, tháng 12 năm 2023.***

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

*Trà Vinh, ngày ….. tháng …… năm ……*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

*Trà Vinh, ngày ….. tháng …… năm ……*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

**NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG**

*Trà Vinh, ngày ….. tháng …… năm ……*

**Thành viên hội đồng**

*(Ký tên và ghi rõ họ tên)*

**LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành nhất đến quý Thầy/Cô Trường Đại Học Trà Vinh, những người đã dìu dắt em tận tình, đã truyền đạt cho em những kiến thức và bài học quý báu khi em theo học tại trường.

Em xin trân trọng gửi lời cảm ơn đến tất cả các thầy cô trong khoa Kỹ thuật – Công nghệ, đặc biệt là Thầy giáo ThS. Ngô Thanh Huy, thầy đã tận tình hướng dẫn và giúp đỡ em trong suốt quá trình làm đồ án cơ sở ngành. Với sự chỉ bảo của thầy, em đã có những định hướng tốt trong việc triển khai và thực hiện các yêu cầu trong quá trình làm đồ án.

Em xin cảm ơn những người thân và gia đình đã quan tâm, động viên và luôn tạo cho em những điều kiện tốt nhất trong suốt quá trình học tập và làm đồ án cơ sở ngành.

Ngoài ra, em cũng xin gửi lời cảm ơn tới tất cả bạn bè, đặc biệt là các bạn trong lớp DA21TTB đã luôn gắn bó, cùng học tập và giúp đỡ em trong quá trình thực hiện đồ án này.

Em xin chân thành cảm ơn!

**MỤC LỤC**

[CHƯƠNG 1 TỔNG QUAN 1](#_Toc155599847)

[CHƯƠNG 2 NGHIÊN CỨU LÝ THUYẾT 2](#_Toc155599848)

[2.1 Giới thiệu về API 2](#_Toc155599849)

[2.1.1 Khái niệm 2](#_Toc155599850)

[2.1.2 Các loại API 2](#_Toc155599851)

[2.1.3 Chức năng của API 3](#_Toc155599852)

[2.1.4 Định dạng dữ liệu JSON và XML 4](#_Toc155599853)

[2.1.5 Thách thức trong quản lí API 5](#_Toc155599854)

[2.1.6 Các công cụ và dịch vụ hỗ trợ quản lí API 5](#_Toc155599855)

[2.2 Giới thiệu về RESTful API 7](#_Toc155599856)

[2.2.1 Khái niệm 7](#_Toc155599857)

[2.2.2 Lợi ích của RESTful API 7](#_Toc155599858)

[2.2.3 Cách hoạt động của RESTful API 8](#_Toc155599859)

[2.2.4 Thành phần của Request client RESTful API 8](#_Toc155599860)

[2.2.5 Thành phần của Respone server RESTful API 9](#_Toc155599861)

[2.2.6 Các phương thức xác thực RESTful API 9](#_Toc155599862)

[2.3 Postman 10](#_Toc155599863)

[2.3.1 Lịch sử phát triển của Postman 10](#_Toc155599864)

[2.3.2 Các tính năng chính của Postman 10](#_Toc155599865)

[2.3.3 Đặc điểm của Postman 11](#_Toc155599866)

[2.3.4 Lí do sử dụng Postman 12](#_Toc155599867)

[CHƯƠNG 3 HIỆN THỰC HÓA NGHIÊN CỨU 13](#_Toc155599868)

[3.1 Cài đặt Postman 13](#_Toc155599869)

[3.1.1 Tải và cài đặt trên các hệ điều hành khác nhau 13](#_Toc155599870)

[3.1.2 Chức năng cơ bản của Postman 14](#_Toc155599871)

[3.2 Làm việc với Request GET 15](#_Toc155599872)

[3.2.1 Lấy tất cả người dùng 15](#_Toc155599873)

[3.2.2 Lấy một người dùng 16](#_Toc155599874)

[3.2.3 Lấy giới hạn người dùng 18](#_Toc155599875)

[3.2.4 Sắp xếp 19](#_Toc155599876)

[3.3 Làm việc với Request POST 20](#_Toc155599877)

[3.4 Làm việc với Request PUT 22](#_Toc155599878)

[3.5 Làm việc với Request PATCH 23](#_Toc155599879)

[3.6 Làm việc với Request DELETE 24](#_Toc155599880)

[3.7 Tests 25](#_Toc155599882)

[3.8 Đặt biến 28](#_Toc155599883)

[3.8.1 Biến toàn cục (Global Variables) 28](#_Toc155599884)

[3.8.2 Biến môi trường (Environment Variables) 29](#_Toc155599885)

[3.9 Tạo Collection 31](#_Toc155599886)

[3.10 Chạy Collection sử dụng Runner 33](#_Toc155599887)

[CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU 36](#_Toc155599888)

[4.1 Kết quả kiểm thử của các Request với Book API 36](#_Toc155599889)

[4.1.1 Authentication với Book API 36](#_Toc155599890)

[4.1.2 Kết quả thực hiện với Request GET 37](#_Toc155599891)

[4.1.3 Kết quả thực hiện với Request POST 38](#_Toc155599892)

[4.2 Kết quả triển khai ứng dụng Backend trên Docker 40](#_Toc155599893)

[4.2.1 Triển khai SaleOr trên Docker 40](#_Toc155599894)

[4.2.2 Kiểm thử với GraphQL 41](#_Toc155599895)

[CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 44](#_Toc155599896)

[5.1 Kết luận 44](#_Toc155599897)

[5.2 Đề xuất mới 44](#_Toc155599898)

[DANH MỤC TÀI LIỆU THAM KHẢO 45](#_Toc155599899)

**DANH MỤC HÌNH ẢNH – BẢNG BIỂU**

[*Hình 1. Download Postman* 23](#_Toc155567023)

[*Hình 2. Giao diện Postman* 24](#_Toc155567024)

[*Hình 3. Lấy tất cả người dùng* 26](#_Toc155567025)

[*Hình 4. Lấy một người dùng* 27](#_Toc155567026)

[*Hình 5. Lấy giới hạn người dùng* 28](#_Toc155567027)

[*Hình 6. Sắp xếp người dùng* 29](#_Toc155567028)

[*Hình 7. Thông tin của một người dùng mới* 31](#_Toc155567029)

[*Hình 8. Thêm mới thông tin người dùng* 31](#_Toc155567030)

[*Hình 9. Thông tin người dùng cần chỉnh sửa* 32](#_Toc155567031)

[*Hình 10. Chỉnh sửa thông tin người dùng* 33](#_Toc155567032)

[*Hình 11. Xóa người dùng* 34](#_Toc155567033)

[*Hình 12. Giao diện Tests* 35](#_Toc155567034)

[*Hình 13. Kiểm tra Status code* 36](#_Toc155567035)

[*Hình 14. Kết quả Test* 36](#_Toc155567036)

[*Hình 15. Một số Test case đơn giản* 37](#_Toc155567037)

[*Hình 16. Global Variables* 38](#_Toc155567038)

[*Hình 17. Request sử dụng Global Variable*s 39](#_Toc155567039)

[*Hình 18. Environment Variables* 40](#_Toc155567040)

[*Hình 19. Request sử dụng Environment Variables* 41](#_Toc155567041)

[*Hình 20. Tạo mới Collection* 42](#_Toc155567042)

[*Hình 21. Thiết lập tên và mô tả cho Collecction* 42](#_Toc155567043)

[*Hình 22. Thiết lập Collection hoàn tất* 43](#_Toc155567044)

[*Hình 23. Runner Tab* 44](#_Toc155567045)

[*Hình 24. Thiết lập Iterations và Delay* 45](#_Toc155567046)

[*Hình 25. Đăng kí tài khoản* 46](#_Toc155567047)

[*Hình 26. Lưu Access Token* 47](#_Toc155567048)

[*Hình 27. Lấy danh sách các quyển sách* 47](#_Toc155567049)

[*Hình 28. Lấy một quyểm sách theo ID* 48](#_Toc155567050)

[*Hình 29. Cài đặt Token* 48](#_Toc155567051)

[*Hình 30. Request POST sử dụng Bearer Token* 49](#_Toc155567052)

[*Hình 31. Gửi đơn đặt hàng* 49](#_Toc155567053)

[*Hình 32. Danh sách đơn hàng đã đặt* 50](#_Toc155567054)

[*Hình 33. Triển khai SaleOr trên Docker Deskstop* 51](#_Toc155567055)

[*Hình 34. Kết quả triển khai SaleOr* 51](#_Toc155567056)

[*Hình 35. Lấy danh sách sản phẩm* 52](#_Toc155567057)

[*Hình 36. Lọc sản phẩm theo tên* 52](#_Toc155567058)

[*Hình 37. Sắp xêp các sản phẩm theo giá* 53](#_Toc155567059)

[*Hình 38. Lấy tất cả các thuộc tính* 53](#_Toc155567060)

**TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH**

Nghiên cứu tập trung vào việc phân tích, kiểm tra, và cải thiện hiệu suất của RESTful API thông qua sử dụng công cụ Postman. Vấn đề này phát sinh do sự ngày càng tăng của ứng dụng và dịch vụ web, đặt ra nhu cầu quản lý và đảm bảo chất lượng của các API một cách hiệu quả.

Tiến hành phân tích chi tiết về các tính năng của Postman, xác định rõ những khả năng mà công cụ này mang lại cho quá trình kiểm thử API. Sau đó, so sánh Postman với các công cụ khác để hiểu rõ về ưu điểm và nhược điểm của mỗi lựa chọn. Cuối cùng, phát triển hướng dẫn sử dụng chi tiết để hỗ trợ việc tích hợp Postman vào quy trình kiểm thử một cách mạch lạc.

Đơn giản hóa việc tích hợp Postman vào công cụ phát triển và môi trường kiểm thử, tạo ra một quy trình linh hoạt và dễ triển khai. Sử dụng tính năng tự động hóa mạnh mẽ của Postman để giảm công việc lặp lại và tăng khả năng phát hiện lỗi, bằng cách xác định và triển khai các kịch bản kiểm thử quan trọng.

Áp dụng thành công việc tích hợp Postman vào quy trình kiểm thử một cách hiệu quả, giúp giảm thiểu lỗi và tăng cường độ tin cậy của hệ thống. Hướng dẫn sử dụng chi tiết cung cấp một cơ sở chất lượng cho cộng đồng phát triển phần mềm, đặc biệt là người mới bắt đầu trong lĩnh vực này. Đồng thời, việc đào sâu vào tính năng kiểm thử của Postman đã mở ra cánh cửa cho sự phát triển tiếp theo của công cụ này, hứa hẹn những cải tiến đáng kể trong tương lai.

**MỞ ĐẦU**

Với sự phổ biến ngày càng tăng của ứng dụng và dịch vụ sử dụng API, đặt ra thách thức đối với quá trình phát triển khi cần đảm bảo tính ổn định và hiệu suất của các API này. Postman, một công cụ mạnh mẽ cho kiểm thử API, đã thu hút sự quan tâm của cộng đồng phát triển, và việc nghiên cứu về cách sử dụng nó có thể mang lại giải pháp hiệu quả cho bài toán trên.

Mục đích chính của nghiên cứu là tìm hiểu và áp dụng các kỹ thuật kiểm thử RESTful API sử dụng Postman, xác định khả năng của công cụ trong việc kiểm thử tự động, kiểm thử bảo mật và đánh giá hiệu suất của các RESTful API. Nghiên cứu cũng hướng đến việc so sánh hiệu quả của Postman so với các công cụ khác có sẵn trên thị trường để đưa ra nhận định về sự linh hoạt và hiệu suất của nó.

Nghiên cứu giúp các nhà phát triển và tester hiểu rõ hơn về công cụ Postman và cách sử dụng công cụ này để kiểm thử API hiệu quả. Ngoài ra, nghiên cứu cũng có thể được sử dụng làm tài liệu tham khảo cho các bạn sinh viên, học viên đang theo học các ngành liên quan đến công nghệ thông tin.

2. TỔNG QUAN

Trong thời đại mà sự kết nối thông tin ngày càng trở nên phổ biến, API đóng vai trò lớn trong việc đơn giản hóa và tối ưu hóa giao tiếp giữa các ứng dụng và hệ thống khác nhau. Tuy nhiên, sự phổ biến này đi kèm với những thách thức liên quan đến độ tin cậy, hiệu suất và an ninh của các API. Vấn đề quan trọng là làm thế nào để có thể đảm bảo rằng các API không chỉ đáp ứng đúng chức năng mà còn đối mặt và vượt qua những thách thức ngày càng phức tạp trong môi trường số hóa. Với sự gia tăng về quy mô và độ phức tạp của hệ thống, việc kiểm thử và đảm bảo chất lượng của API trở nên rất quan trọng.

API có nhiều loại khác nhau, bao gồm API web, API RESTful, API SOAP, và nhiều hơn nữa. Trong đó, API RESTful được xem là một kiểu thiết kế API web tiêu biểu, tập trung vào tài nguyên và thao tác trên chúng thông qua các phương thức HTTP.

Trong các dự án, phần server và client làm độc lập với nhau nên nhiều chỗ back-end chưa hoàn thành nên không thể chờ xong rồi mới thực hiện kiểm tra. Lúc này việc kiểm tra cần phải thực hiện qua công cụ khác để kiểm tra dữ liệu trao đổi giữa client và server có chính xác hay không. Chính vì vậy, viêc sử dụng Postman để thực hiện các kiểm thử đầy đủ và hiệu quả trở thành một hướng tiếp cận thiết yếu để giải quyết những thách thức này.

Nghiên cứu cung cấp thông tin hữu ích và hướng dẫn chi tiết, không chỉ để giúp người quản lý và phát triển RESTful API hiểu rõ hơn về cách sử dụng Postman mà còn để góp phần vào sự nâng cao của quá trình duy trì và phát triển API. Xác định các vấn đề thường gặp, đảm bảo tính ổn định của API, tất cả nhằm cung cấp một cơ sở vững chắc cho việc triển khai và sử dụng các API một cách hiệu quả.

1. NGHIÊN CỨU LÝ THUYẾT
   1. **Giới thiệu về API**
      1. **Khái niệm**

API là từ viết tắt của Application Programming Interface (giao diện lập trình ứng dụng), là một tập hợp các giao thức cho phép các thành phần phần mềm khác nhau giao tiếp và truyền dữ liệu. Các nhà phát triển sử dụng API để thu hẹp khoảng cách giữa các đoạn mã nhỏ, rời rạc nhằm tạo ra các ứng dụng mạnh mẽ, linh hoạt, an toàn và có thể đáp ứng nhu cầu của người dùng. Mặc dù không thể nhìn thấy chúng nhưng API có ở khắp mọi nơi - hoạt động liên tục ở chế độ nền để hỗ trợ những trải nghiệm kỹ thuật số cần thiết cho cuộc sống hiện đại. [1]

API là cầu nối giữa client và server. Client ở đây có thể là web trên máy tính, app trên điện thoại sử dụng hệ điều hành khác nhau và được viết bằng những ngôn ngữ khác nhau. Tương tự, server back-end cũng được viết bằng các ngôn ngữ khác nhau. Để 2 đối tượng này có thể “nói chuyện” được với nhau chúng phải nói cùng 1 ngôn ngữ đó chính là API.

* + 1. **Các loại API**
       1. **Phân loại API theo người có quyền truy cập:**

Private APIs (API riêng tư): còn được gọi là API nội bộ, được sử dụng để kết nối các thành phần phần mềm khác nhau trong một tổ chức và chúng không có sẵn cho bên thứ ba sử dụng.

Public APIs (API công khai): API công khai cung cấp quyền truy cập công khai vào dữ liệu, chức năng hoặc dịch vụ của tổ chức mà các nhà phát triển bên thứ ba có thể tích hợp vào ứng dụng của riêng họ. Một số API công khai được cung cấp miễn phí, trong khi những API khác được cung cấp dưới dạng sản phẩm phải thanh toán.

Partner APIs (API đối tác): API đối tác cho phép hai hoặc nhiều công ty chia sẻ dữ liệu hoặc chức năng để cộng tác trong một dự án. Chúng không được cung cấp rộng rãi và do đó tận dụng các cơ chế xác thực để đảm bảo chúng chỉ được sử dụng bởi các đối tác được ủy quyền.

* + - 1. **Phân loại API theo phong cách kiến ​​trúc:**

REST: REST là kiến ​​trúc API phổ biến nhất để truyền dữ liệu qua internet. Trong ngữ cảnh RESTful, tài nguyên có thể truy cập được thông qua endpoint (điểm cuối) và các hoạt động được thực hiện trên các tài nguyên (resouce) bằng các phương thức HTTP tiêu chuẩn như GET, POST, PUT và DELETE. [2]

SOAP: SOAP là từ viết tắt của  Simple Object Access Protocol (Giao thức truy cập đối tượng đơn giản) sử dụng XML để truyền các thông báo có cấu trúc cao giữa máy khách và máy chủ. SOAP thường được sử dụng trong môi trường doanh nghiệp hoặc hệ thống cũ và mặc dù bao gồm các tính năng bảo mật nâng cao nhưng nó có thể chậm hơn các kiến ​​trúc API khác. [2]

GraphQL: GraphQL là ngôn ngữ truy vấn nguồn mở cho phép khách hàng tương tác với một điểm cuối API duy nhất để truy xuất dữ liệu chính xác mà họ cần mà không cần xâu chuỗi nhiều yêu cầu lại với nhau. Cách tiếp cận này làm giảm số lượng các chuyến đi khứ hồi giữa máy khách và máy chủ, điều này có thể hữu ích cho các ứng dụng có thể chạy trên kết nối mạng chậm hoặc không đáng tin cậy. [2]

* + 1. **Chức năng của API**

Tích hợp nhiều hệ thống: Một trong những lý do phổ biến nhất khiến các nhà phát triển chuyển sang sử dụng API là để tích hợp hệ thống này với hệ thống khác.

Thêm hoặc nâng cao chức năng: API cho phép kết hợp chức năng bổ sung vào ứng dụng của mình, điều này có thể cải thiện trải nghiệm của khách hàng.

Kết nối thiết bị IoT: API rất cần thiết cho Internet of Things (IoT), bao gồm các thiết bị như đồng hồ thông minh, máy theo dõi thể dục, chuông cửa và thiết bị gia dụng.

Tạo ra hệ thống có khả năng mở rộng: API được sử dụng để triển khai kiến ​​trúc dựa trên vi dịch vụ, trong đó các ứng dụng được xây dựng dưới dạng tập hợp các dịch vụ nhỏ giao tiếp với nhau thông qua các API riêng tư. Các vi dịch vụ được quản lý, triển khai và cung cấp độc lập với nhau, cho phép các nhóm mở rộng quy mô hệ thống một cách đáng tin cậy nhưng tiết kiệm chi phí.

* + 1. **Định dạng dữ liệu JSON và XML**
       1. **JSON**

Khái niệm: JSON là cụm từ viết tắt của JavaScript Object Notation (Ký pháp đối tượng của JavaScript). Đây là một dạng cấu trúc dữ liệu dạng văn bản thường được dùng để lưu trữ và trao đổi thông tin trực tiếp giữa các hệ thống.

Cấu trúc: JSON là một chuỗi văn bản nằm giữa cặp dấu { và }. Phần tử cơ bản nhất trong chuỗi JSON là một cặp thông tin dạng "key":"value" (hay "khóa": "giá trị"). Có thể ghép nhiều cặp thông tin với nhau và phân tách chúng bởi dấu phẩy.

Ví dụ : {

“id”: 1175,

“name”: “Nguyen Thanh Nam”,

“email”: “[nguyenthanhnam0703@gmail.com](mailto:nguyenvana@gmail.com)”

}

Key (khóa) trong JSON có thể là một chuỗi, một số..., nhưng để đảm bảo tính đồng nhất giữa các ngôn ngữ lập trình, key nên được đặt ở dạng chuỗi (bao quanh bởi cặp dấu nháy kép) và tuân thủ theo quy tắc đặt tên biến của các ngôn ngữ lập trình.

Value (giá trị) trong JSON có thể nhận một trong các giá trị sau: chuỗi (string), số (number), mảng (array), chuỗi JSON khác, boolean, null.

* + - 1. **XML**

Khái niệm: XML là từ viết tắt của Extensible Markup Language là ngôn ngữ đánh dấu mở rộng. XML thường được sử dụng để cấu trúc, lưu trữ và trong trao đổi dữ liệu giữa các hệ thống.

Cú pháp XML cơ bản cho một phần tử là: <tên thuộc\_tính=”giá trị”>nội dung</tên >

Ví dụ:

<id>1175</id>

<name gender=”male”>Nguyen Thanh Nam</name>

<email>[nguyenthanhnam0703@gmail.com](mailto:nguyenvana@gmail.com)<email/>

* + 1. **Thách thức trong quản lí API**

Mặc dù API mang lại nhiều lợi ích trong quá trình phát triển phần mềm và kết nối dịch vụ, nhưng cũng đối mặt với một số thách thức và rủi ro. Dưới đây là một số thách thức phổ biến của API:

Rủi ro an ninh: API thường là mục tiêu của các tấn công mạng, bao gồm tấn công SQL injection, cross-site scripting (XSS), và cross-site request forgery (CSRF).

Xác thực và ủy quyền: Quản lý xác thực người dùng và quyền truy cập đối với các API có thể là một thách thức, đặc biệt là khi cần tương tác với nhiều dịch vụ khác nhau.

Quản lý phiên bản: Khi API phát triển và thay đổi, việc quản lý sự tương thích với phiên bản cũ có thể là một thách thức. Các phiên bản mới của API phải duy trì sự tương thích với ứng dụng hiện tại.

Quản lý thời gian phản hồi (Latency): Latency trong việc truy cập và sử dụng API có thể ảnh hưởng đến hiệu suất của ứng dụng. Điều này đặc biệt quan trọng đối với các ứng dụng yêu cầu phản hồi nhanh chóng.

Tài nguyên hệ thống: API tiêu tốn tài nguyên hệ thống. Việc quản lý và mở rộng tài nguyên một cách hiệu quả là một thách thức, đặc biệt khi có nhiều API hoạt động cùng một lúc.

Quản lý tài liệu (Documentation): Hiểu rõ cách sử dụng API đôi khi phụ thuộc vào chất lượng của tài liệu hướng dẫn. Tài liệu không đầy đủ hoặc không chính xác có thể tạo ra sự nhầm lẫn và khó khăn trong việc tích hợp.

* + 1. **Các công cụ và dịch vụ hỗ trợ quản lí API**
       1. **Postman**

Postman là một ứng dụng được thiết kế để thử nghiệm, kiểm thử và quản lý API. Nó cung cấp một giao diện đồ họa dễ sử dụng và khả năng tự động hóa các bước kiểm thử.

Chức năng quan trọng:

Thử nghiệm API: Cho phép gửi các yêu cầu HTTP đến API và xem phản hồi.

Kiểm thử tự động: Hỗ trợ việc tự động hóa các bước kiểm thử, giúp đơn giản hóa quy trình kiểm thử.

Quản lý Environment: Cho phép quản lý các môi trường khác nhau để dễ dàng chuyển đổi giữa các cấu hình kiểm thử.

* + - 1. **Swagger/OpenAPI Editor:**

Swagger là một công cụ giúp mô tả và tài liệu hóa API. OpenAPI là một tiêu chuẩn mở dựa trên Swagger, giúp mô tả API một cách chi tiết và đồng nhất.

Chức năng quan trọng:

Mô tả API: Cho phép viết và mô tả API thông qua JSON hoặc YAML, bao gồm các endpoint, phương thức, và các tham số.

Tạo Tài liệu: Tự động tạo tài liệu API dựa trên mô tả, giúp nhà phát triển và người dùng hiểu rõ cách sử dụng API.

Hỗ trợ Code Generation: Cung cấp mã nguồn mẫu dựa trên mô tả, giúp người phát triển tích hợp API dễ dàng hơn.

* + - 1. **Apiary**

Apiary là một dịch vụ giúp xây dựng, thiết kế và tài liệu hóa API. Nó hỗ trợ quản lý chuỗi cung ứng API.

Chức năng quan trọng:

Tạo và mô tả API: Cho phép xây dựng API và mô tả chúng thông qua một giao diện trực quan.

Tài liệu hóa API: Tự động tạo tài liệu API từ mã nguồn và cung cấp một cách hiệu quả để tương tác với API.

Thử nghiệm API: Cho phép kiểm thử API trực tuyến và tạo kịch bản kiểm thử.

* + - 1. **AWS API Gateway**

AWS API Gateway là một dịch vụ của Amazon Web Services giúp quản lý và triển khai API trên nền tảng AWS.

Chức năng quan trọng:

Quản lý API: Cho phép tạo, quản lý và theo dõi API dễ dàng từ bảng điều khiển AWS.

Bảo mật API: Cung cấp các tùy chọn xác thực và ủy quyền để bảo vệ API.

Thống kê và Giám sát: Cung cấp thông tin thống kê và giám sát hiệu suất của API.

* 1. **Giới thiệu về RESTful API**
     1. **Khái niệm**

RESTful API (Representational State Transfer API) là một tiêu chuẩn được sử dụng trong việc thiết kế API cho các phần mềm ứng dụng và dịch vụ web để tạo sự thuận tiện quản lí các resouce. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP. [3]

Các nhà phát triển API có thể thiết kế các API bằng cách sử dụng nhiều kiến trúc khác nhau. Các API tuân theo kiểu kiến trúc REST được gọi là API REST. Các dịch vụ web triển khai kiến trúc REST được gọi là dịch vụ web RESTful. Thuật ngữ API RESTful thường là chỉ các API web RESTful.

* + 1. **Lợi ích của RESTful API**

Khả năng thay đổi quy mô: Các hệ thống triển khai RESTful API có thể thay đổi quy mô một cách hiệu quả vì REST tối ưu hóa các tương tác giữa client và server. Tình trạng phi trạng thái (stateless) loại bỏ tải nguyên của máy chủ vì máy chủ không phải giữ lại thông tin yêu cầu của client trong quá khứ. Việc lưu bộ nhớ đệm được quản lý tốt sẽ loại bỏ một phần hoặc hoàn toàn một số tương tác giữa client và server. Tất cả các tính năng này hỗ trợ khả năng thay đổi quy mô mà không gây ra tắc nghẽn giao tiếp làm giảm hiệu suất.

Sự linh hoạt: Các dịch vụ web RESTful hỗ trợ phân tách hoàn toàn giữa client và sever. Các dịch vụ này đơn giản hóa và tách riêng các thành phần máy chủ khác nhau để mỗi phần có thể phát triển độc lập. Các thay đổi ở nền tảng hoặc công nghệ tại ứng dụng máy chủ không ảnh hưởng đến ứng dụng client.

Sự độc lập: Các RESTful API không phụ thuộc vào công nghệ được sử dụng. Có thể viết cả ứng dụng client và server bằng nhiều ngôn ngữ lập trình khác nhau mà không ảnh hưởng đến thiết kế API.

* + 1. **Cách hoạt động của RESTful API**

Chức năng cơ bản của RESTful API cũng giống như việc duyệt Internet. Client liên hệ với máy chủ bằng cách sử dụng API khi yêu cầu tài nguyên. Các nhà phát triển API giải thích cách client nên sử dụng RESTful API trong tài liệu về API ứng dụng máy chủ. Đây là các bước chung cho bất kỳ lệnh gọi RESTful API nào:

1. Client gửi một yêu cầu đến server. Client làm theo tài liệu API để định dạng yêu cầu theo cách mà server hiểu được.
2. Server xác thực và xác nhận máy khách có quyền đưa ra yêu cầu đó.
3. Server nhận yêu cầu và xử lý trong nội bộ.
4. Server trả về một phản hồi đến client. Phản hồi chứa thông tin cho client biết liệu yêu cầu có thành công hay không. Phản hồi cũng bao gồm bất kỳ thông tin nào mà client yêu cầu.

Chi tiết về phản hồi và yêu cầu RESTful API sẽ khác nhau đôi chút tùy thuộc vào cách các nhà phát triển API thiết kế API.

* + 1. **Thành phần của Request client RESTful API**

URL (Uniform Resouce Indentifier): Đối với các dịch vụ REST, máy chủ thường thực hiện việc xác định tài nguyên bằng cách sử dụng Hệ thống định vị tài nguyên thống nhất (URL). URL chỉ định đường dẫn đến tài nguyên, còn được gọi là end point (điểm cuối) yêu cầu và chỉ định rõ cho máy chủ rằng client yêu cầu điều gì.

HTTP Method (Phương thức HTTP): Các phương thức HTTP được sử dụng để thực hiện các thao tác khác nhau trên dữ liệu. Các phương thức chính bao gồm:

1. GET: Trả về một tài nguyên hoặc danh sách các tài nguyên.
2. POST: Tạo mới một tài nguyên.
3. PUT: Cập nhật một tài nguyên hoặc tạo mới nếu không tồn tại.
4. DELETE: Xoá một tài nguyên.

Header**:** Header là phần đầu của một gói tin HTTP, chứa các thông tin bổ sung về request hoặc response. Header được gửi cùng với mỗi yêu cầu và phản hồi để cung cấp các thông tin liên quan đến việc xử lý yêu cầu hoặc phản hồi.

Body**:** Body là phần chứa dữ liệu của yêu cầu hoặc phản hồi. Trong một request, body thường chứa dữ liệu được gửi từ client đến server, trong khi trong một response, body thường chứa dữ liệu trả về từ server đến client.

* + 1. **Thành phần của Respone server RESTful API**

Status Code (Mã trạng thái): Mã trạng thái HTTP định nghĩa kết quả của yêu cầu HTTP mà client đã gửi tới server. Dưới đây là một số mã trạng thái phổ biến:

1. 100 Continue: Server đang chờ client tiếp tục với yêu cầu.
2. 200 OK: Yêu cầu thành công, server trả về dữ liệu theo yêu cầu.
3. 201 Created: Yêu cầu đã tạo thành công một tài nguyên mới.
4. 204 No Content: Yêu cầu thành công, nhưng không có dữ liệu để trả về.
5. 301 Moved Permanently: Tài nguyên đã được chuyển vĩnh viễn tới một địa chỉ mới.
6. 304 Not Modified: Dữ liệu không được trả về vì không có sự thay đổi kể từ lần truy cập trước đó.
7. 400 Bad Request: Yêu cầu không hợp lệ từ phía client.
8. 401 Unauthorized: Yêu cầu yêu cầu xác thực.
9. 403 Forbidden: Client không có quyền truy cập vào tài nguyên.
10. 404 Not Found: Tài nguyên yêu cầu không tồn tại trên server.
11. 500 Internal Server Error: Lỗi nội bộ của máy chủ.
12. 502 Bad Gateway: Máy chủ, nhưng không thể nhận dữ liệu từ máy chủ khác (ví dụ, máy chủ lớn chuyển hướng dữ liệu).
13. 503 Service Unavailable: Máy chủ không thể xử lý yêu cầu do tạm thời quá tải hoặc bảo trì.

Header/Body: tương tự như Request client

* + 1. **Các phương thức xác thực RESTful API**

API Key (Khóa API): Khóa API là một định danh duy nhất được chuyển trong các tiêu đề hoặc tham số truy vấn của yêu cầu API. Máy chủ kiểm tra khóa để xác định liệu yêu cầu có được ủy quyền hay không. Khóa API đơn giản và được sử dụng rộng rãi, nhưng có thể thiếu bảo mật đối với một số trường hợp sử dụng. [4]

OAuth (Open Authorization): OAuth là một giao thức xác thực mạnh mẽ và an toàn hơn, cho phép ứng dụng bên thứ ba truy cập vào tài nguyên của người dùng mà không tiết lộ thông tin đăng nhập của họ. OAuth 2.0 là phiên bản phổ biến nhất, cung cấp các quy trình ủy quyền cho các tình huống khác nhau, bao gồm xác thực người dùng và ủy quyền cho ứng dụng bên thứ ba. [4]

JWT (JSON Web Tokens): JWT là một cách tiện lợi và an toàn để biểu diễn các yêu cầu sẽ được chuyển giữa hai bên. JWT có thể được sử dụng như một loại Bearer Token để xác thực. Chúng thường được sử dụng trong các hệ thống xác thực dựa trên token và chứa thông tin về người dùng và tính hợp lệ của token. [4]

Bearer Token: Xác thực Bearer Token liên quan đến việc bao gồm một token trong các tiêu đề của yêu cầu (thường là tiêu đề "Authorization"). Máy chủ kiểm tra token để cấp quyền truy cập. OAuth 2.0 thường sử dụng Bearer Tokens, và chúng thường được sử dụng với HTTPS để tăng cường bảo mật.

* 1. **Postman**
     1. **Lịch sử phát triển của Postman**

Postman là một ứng dụng cho phép người dùng kiểm thử, quản lí và tương tác với API, giúp cho thao tác người dùng với API trở nên dễ dàng hơn. Postman tương thích với hầu hết các giao thức bao gồm: GET, POST, PUT, DELETE, … [5]

Postman được phát triển bởi công ty Mỹ Postdot Technologies và ra mắt lần đầu vào năm 2012. Ban đầu, Postman chỉ là một extension của Chrome nhằm hỗ trợ kiểm thử API. Tuy nhiên, với sự gia tăng đáng kể về quy mô và tính năng, Postman đã trở thành một ứng dụng riêng biệt, hỗ trợ nhiều nền tảng, từ Windows, macOS đến Linux. [6]

* + 1. **Các tính năng chính của Postman**

Gửi yêu cầu HTTP: Postman cho phép tạo và gửi các yêu cầu HTTP như GET, POST, PUT, DELETE với các tham số khác nhau.

Tạo và quản lí môi trường: Môi trường trong Postman là bộ các biến mà bạn có thể thay đổi dễ dàng giúp chuyển đổi giữa cấu hình môi trường (ví dụ: Development, Testing, Production) một cách nhanh chóng mà không cần phải thay đổi thủ công từng tham số.

Kiểm thử tự động: Postman có tích hợp một bộ kiểm thử mạnh mẽ cho kiểm thử tự động, giúp tự động hóa các yêu cầu và kiểm thử API.

Xác thực và quyền truy cập: Hỗ trợ nhiều phương pháp xác thực, tích hợp quản lí quyền truy cập để kiểm soát quyền truy cập vào các yêu cầu.

Chia sẻ và hợp tác: Chia sẻ bộ kiểm thử và môi trường thông qua liên kết hoặc tích hợp với Postman Cloud. Postman hỗ trợ làm viêc nhóm, cho phép nhiều người cùng thực hiện trên một bộ kiểm thử.

* + 1. **Đặc điểm của Postman**
       1. **Ưu điểm**

Giao diện của Postman rất thân thân thiện với người dùng, giúp dễ dàng tạo và gửi các yêu cầu mà không cần kiến thức sâu về cú pháp HTTP.

Postman sở hữu Collection nên người dùng có thể thoải mái tạo các request cho API. Đây chính là một điểm mạnh lớn mà Postman sở hữu. Các hoạt động thử nghiệm được diễn ra dễ dàng hơn nhờ sự bảo trợ của Postman.

Postman có tính linh động rất cao. Thay vì phải tạo một collection hoặc một environment mới, có thể import dự án đang dang dở từ những nền tảng khác. Postman chấp nhận đủ các loại file, link.

Ngoài ra, Postman còn được trang bị thêm bộ gỡ lỗi giúp kiểm tra các dữ liệu người dùng đã xuất. Từ đó đảm bảo tính vẹn toàn của kết quả sau khi sử dụng Postman hơn.

Cung cấp ứng dụng trên nhiều nền tảng khác nhau như Windows, macOS và Linux. Tích hợp với phiên bản trực tuyến và ứng dụng di động.

* + - 1. **Nhược điểm**

Cũng giống như các ứng dụng công nghệ khác, Postman có những nhược điểm riêng. Hạn chế của Postman là không phải cái gì cũng miễn phí. Rất nhiều tính năng quan trọng bị giới hạn và chỉ có thể sử dụng khi mua các gói trả phí. Tuy nhiên đây cũng không phải là hạn chế quá lớn. Thường các gói này không quá đắt. Hơn nữa, một dự án phát triển phần mềm sẽ không thiếu chút chi phí này để tối ưu hóa quá trình xây dựng.

* + 1. **Lí do sử dụng Postman**

Kiểm tra tự động: Postman cung cấp công cụ cho phép chúng ta viết các script test và thực thi chúng một cách tự động theo thời gian đặt trước, hoặc có thể tích hợp những script test đó vào trong CI/CD pipeline.

Đa dạng loại test: Postman hỗ trợ nhiều loại test như integration tests, regression tests, functional tests, mock tests, end-to-end tests.

Ứng dụng đa nền tảng: Postman online để có thể dùng trên bất cứ trình duyệt nào và cả ứng dụng cài đặt trên máy cho Windows, MacOS, Linux.

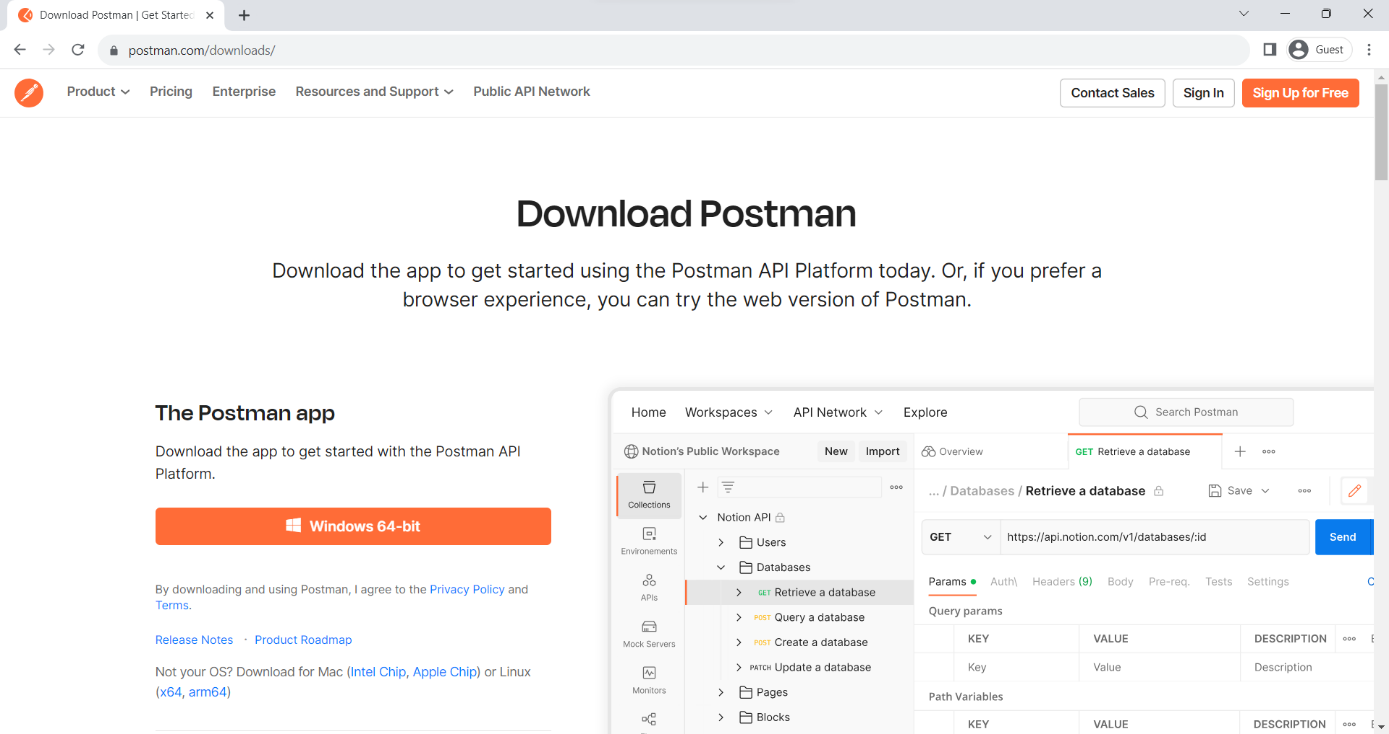
Báo cáo: Postman tích hợp sẵn các loại báo cáo về các API đang sử dụng như: Thống kê số lượng API, số lượng workspace, đo lường thời gian cũng như tần suất sử dụng các API theo thời gian, tổng hợp thông tin về thời gian phản hồi và kích thước các response, số lượng test thành công, thất bại theo thời gian ...

Phù hợp với tester ở mọi level : Người mới trong API testing có thể sử dụng cấu hình trực quan của Postman để sử dụng và test, với những người đã có kinh nghiệm thì có thể sử dụng script test để làm việc.

1. HIỆN THỰC HÓA NGHIÊN CỨU
   1. **Cài đặt Postman**
      1. **Tải và cài đặt trên các hệ điều hành khác nhau**

Postman là loại công cụ mã nguồn mở, vì vậy người dùng có thể tải về một cách rất dễ dàng. Cụ thể, để download và cài đặt Postman, người dùng sẽ căn cứ vào các bước sau:

Bước 1: Truy cập vào websitehttps://www.postman.com/downloads/. Sau đó, hãy lựa chọn đúng nền tảng cần tải về (có thể là Windows, Linux hoặc Mac) và click chuột vào nút Download.



*Hình 1. Download Postman*

Bước 2: Sau khi tải về thành công, file sẽ hiển thị ở phần cửa sổ dưới của trình duyệt. Trong quá trình tải về, chỉ cần thao tác nhấn nút hoàn tất và kích nút Run.

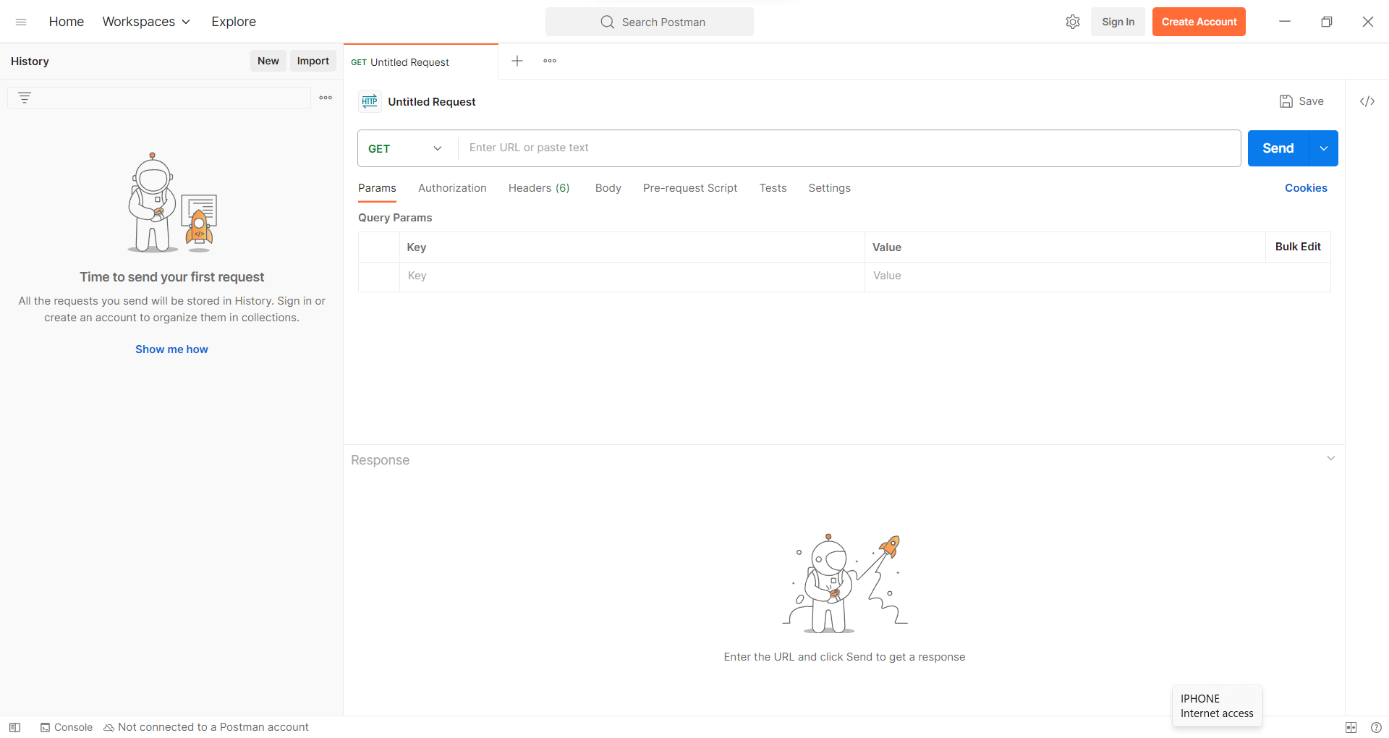
Bước 3: Tiến hành cài đặt.

Bước 4: Lựa chọn và đăng nhập tài khoản Postman. Có thể đăng nhập vào Postman bằng 2 cách là sử dụng tài khoản Google hoặc tạo một tài khoản mới trên hệ thống. Ngoài ra, Postman cũng cho phép người dùng sử dụng công cụ mà không cần phải đăng nhập. Tuy nhiên, việc đăng ký tạo tài khoản mới sẽ đảm bảo tính lưu trữ cũng như truy cập vào bộ sưu tập trong việc sử dụng về sau.

Bước 5: Lựa chọn công cụ dành cho workspace, sau đó click chuột vào Save My Preference.

Bước 6: Màn hình Startup được xuất hiện.

* + 1. **Chức năng cơ bản của Postman**



*Hình 2. Giao diện Postman*

Bên trên là cửa sổ làm việc của Postman:

1. **New**: Đây là nơi tạo request, collection hoặc enviroment mới.
2. **Import**: Được sử dụng để import collection hoặc environment. Có các tuỳ chọn để import từ file, folder, link hoặc paste từ text thuần.
3. **My Workspace**: Tạo sổ làm việc riêng hoặc cho một nhóm.
4. **History**: Các request đã thực hiện sẽ được hiển thị trong History. Giúp bạn có thể lần theo các hành động đã làm.
5. **Collections**: Tổ chức bộ thử nghiệm bằng cách tạo Collection. Mỗi collection có thể có các thư mục con và nhiều yêu cầu. Request hoặc thư mục cũng có thể được trùng lặp.
6. **Tab Request**: Hiển thị tiêu đề của requet đang làm việc. Mặc định “Untitled Request” sẽ được hiển thị cho các request không có tiêu đề.
7. **HTTP Request**: Khi click vào đây sẽ hiển thị danh sách thả xuống với các request khác nhau như GET, POST, COPY, DELETE, v.v.
8. **Request URL**: Còn được gọi là điểm cuối (endpoint), đây là nơi xác định liên kết đến nơi API sẽ giao tiếp.
9. **Save**: Nếu có thay đổi đối với request, nhấp vào Save là bắt buộc để những thay đổi mới sẽ không bị mất hoặc bị ghi đè.
10. **Params**: Đây là nơi viết các tham số cần thiết cho một request, ví dụ như các cặp key – value.
11. **Authorization**: Để truy cập API, cần được cấp quyền. Nó có thể ở dạng tên người dùng và mật khẩu, bearer token, v.v.
12. **Headers**: Người dùng có thể thiết lập các header như nội dung kiểu JSON tùy theo cách tổ chức.
13. **Body**: Đây là nơi tùy chỉnh chi tiết trong request thường được sử dụng trong request POST.
14. **Pre-request Script**: Đây là các tập lệnh sẽ được thực thi trước request. Thông thường, script tiền request (pre-request) cho cài đặt môi trường được sử dụng để đảm bảo các kiểm tra sẽ được chạy trong môi trường chính xác.
15. **Tests**: Đây là các script được thực thi khi request. Điều quan trọng là phải có các thử nghiệm như thiết lập các điểm checkpoint để kiểm tra trạng thái, dữ liêu nhận được có như mong đợi không và các thử nghiệm khác.
    1. **Làm việc với Request GET**

Phương thức GET thường được sử dụng khi muốn truy xuất dữ liệu từ một hoặc nhiều đối tượng.

Chúng ta sẽ sử dụng API có sẵn tại: **https://fakestoreapi.com**

* + 1. **Lấy tất cả người dùng**

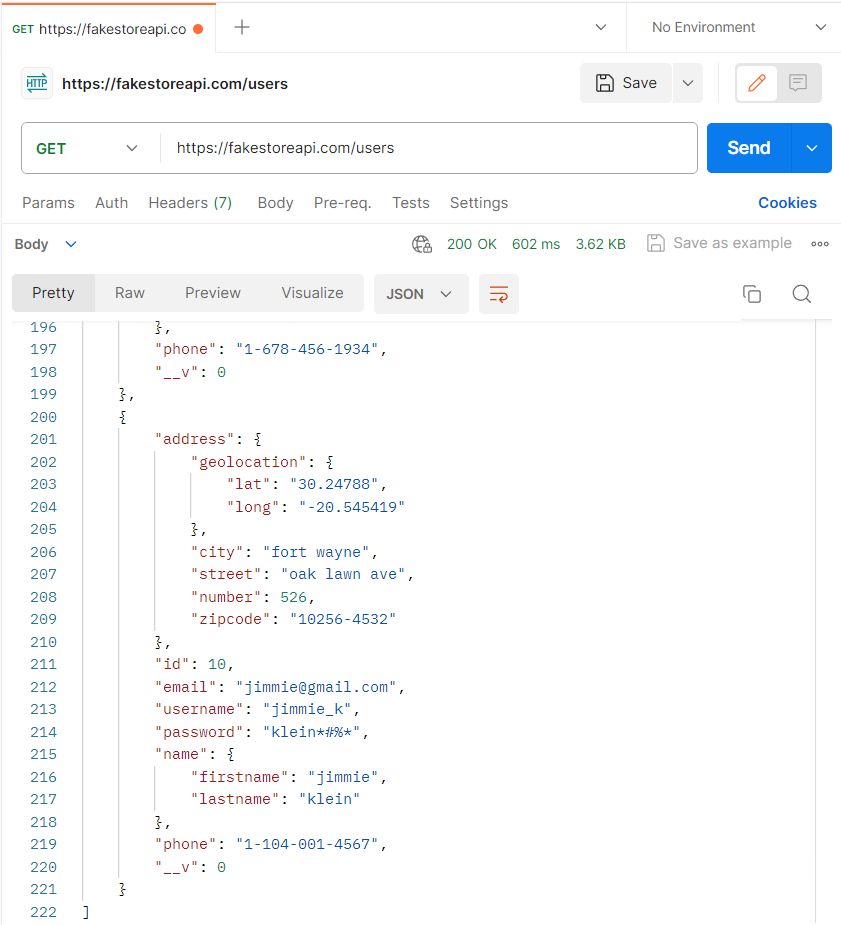
Yêu cầu trả về danh sách hoặc tập hợp tất cả thông tin người dùng trong hệ thống. Yêu cầu này được sử dụng khi một trang web hoặc ứng dụng di động có chức năng hiển thị danh sách tất cả người dùng hiện có, hay kiểm tra và quản lý quyền truy cập cho mỗi người dùng.

Trong workspace:

Bước 1: Thiết lập request HTTP là **GET**.

Bước 2: Nhập vào thanh URL với URL là **https://fakestoreapi.com/users** để lấy toàn bộ người dùng.

Bước 3: Kích nút **Send** để gửi yêu cầu GET lên server.



*Hình 3. Lấy tất cả người dùng*

Status hiển thị 200 OK. Kết quả hiển thị 10 người dùng trong phần Body.

* + 1. **Lấy một người dùng**

Yêu cầu trả về thông tin của một người dùng trong hệ thống, endpoint sẽ chứa ID của người dùng cần truy xuất thông tin. Yêu cầu này được sử dụng khi ứng dụng hoặc trang web cần lấy thông tin của người dùng để hiển thị chi tiết như tên, hình ảnh, địa chỉ, email.

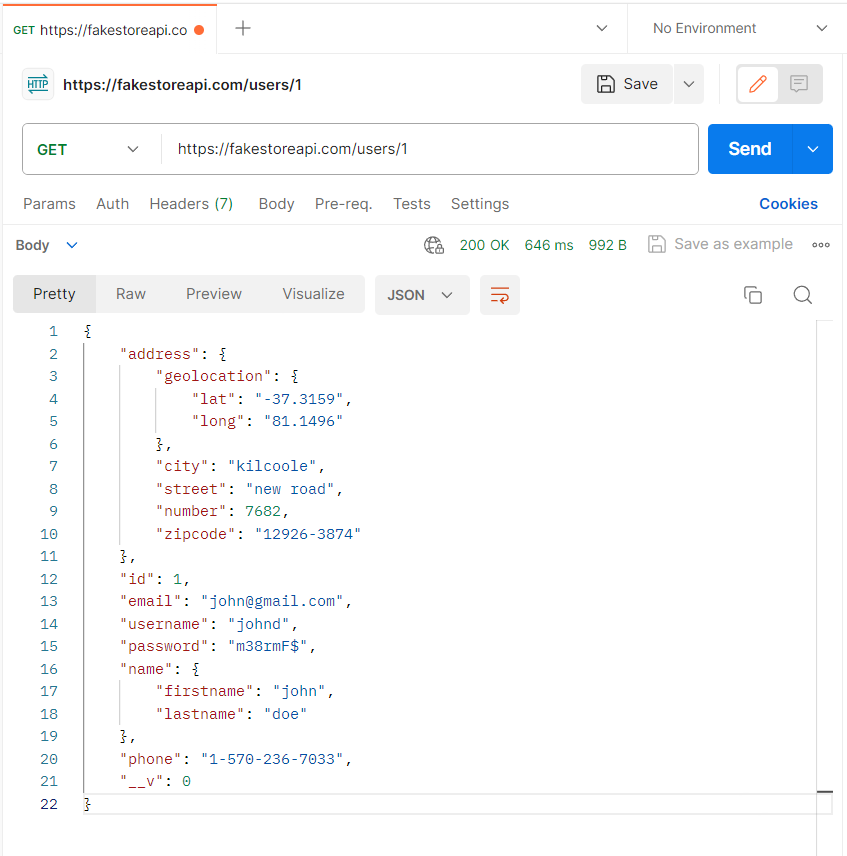
Trong workspace:

Bước 1: Thiết lập request HTTP là **GET**.

Bước 2: Nhập vào thanh URL với URL là

[**https://fakestoreapi.com/users/:usersID**](https://fakestoreapi.com/users/:usersID) để lấy người dùng theo ID.

Bước 3: Kích nút **Send**.



*Hình 4. Lấy một người dùng*

Status hiển thị 200 OK. Kết quả hiển thị người dùng có ID là 1 trong phần Body.

* + 1. **Lấy giới hạn người dùng**

Khi số lượng người dùng trong hệ thống lớn, việc lấy toàn bộ danh sách có thể làm cho yêu cầu trở nên lớn và tốn kém. Trong trường hợp này, việc lấy giới hạn số lượng người dùng và thực hiện phân trang, giúp giảm tải cho máy chủ và tối ưu hóa hiệu suất.

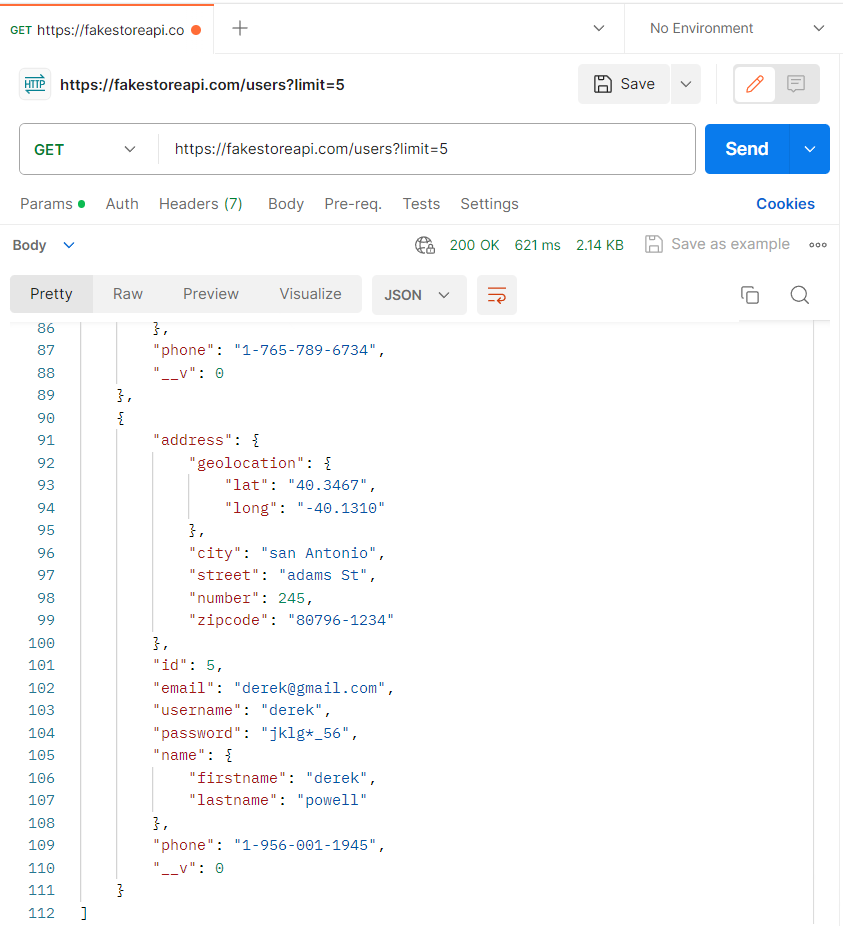
Trong workspace:

Bước 1: Thiết lập request HTTP là **GET**.

Bước 2: Nhập vào thanh URL với URL là

[**https://fakestoreapi.com/users?limit=userLimit**](https://fakestoreapi.com/users?limit=5) để lấy giới hạn người dùng.

Bước 3: Kích nút **Send**.



*Hình 5. Lấy giới hạn người dùng*

Status hiển thị 200 OK. Kết quả hiển thị 5 người dùng đầu tiên trong phần Body.

Lưu ý: Có thể thay đổi số lượng khác tùy theo mục đích sử dụng.

* + 1. **Sắp xếp**

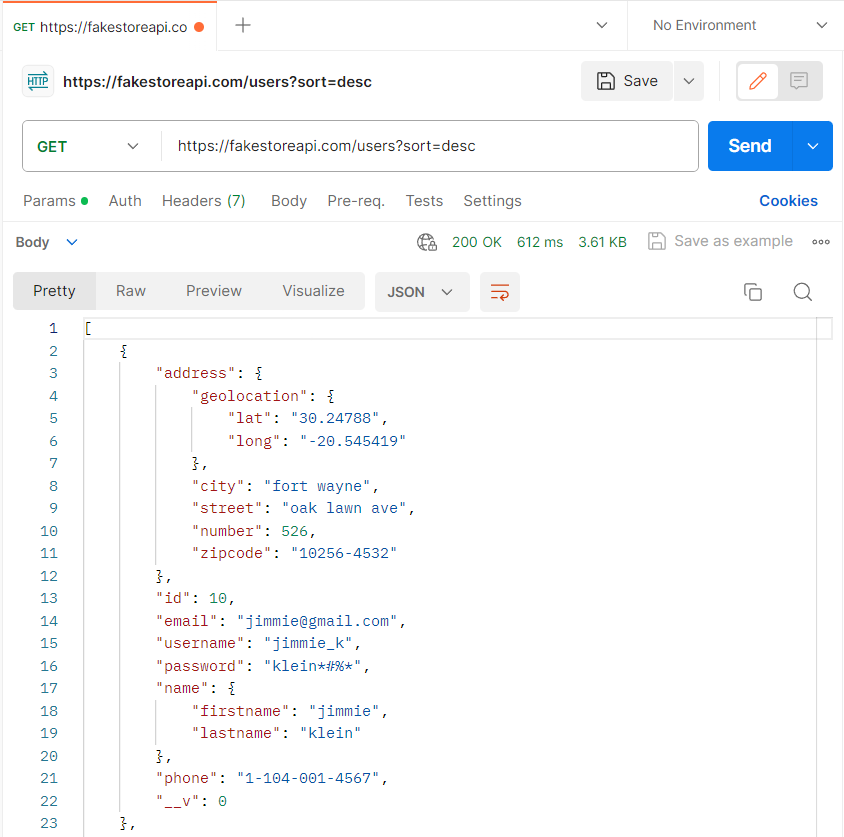
Yêu cầu dữ liệu được trả về theo một thứ tự cụ thể dựa trên một hoặc nhiều thuộc tính. Yêu cầu này được sử dụng khi muốn hiển thị danh sách người dùng trên giao diện người dùng theo một thứ tự cụ thể, chẳng hạn như theo tên, ngày đăng ký, hoặc mức độ hoạt động gần đây.

Bước 1: Thiết lập request HTTP là **GET**.

Bước 2: Nhập vào thanh URL với URL là

[**https://fakestoreapi.com/users?sort**](https://fakestoreapi.com/users?sort)**=name** để sắp xếp người dùng. Có 2 cách sắp xếp là desc (giảm dần) và asc (tăng dần).

Bước 3: Kích nút **Send**.



*Hình 6. Sắp xếp người dùng*

Status hiển thị 200 OK. Kết quả hiển thị tất cả người dùng được sắp xếp giảm dần theo ID.

* 1. **Làm việc với Request POST**

Request POST được sử dụng khi muốn tạo mới một tài nguyên trong hệ thống, chẳng hạn như tạo mới một người dùng, một bài viết, hoặc một sản phẩm. Dữ liệu của tài nguyên mới thường được đặt trong phần thân của yêu cầu (request body).

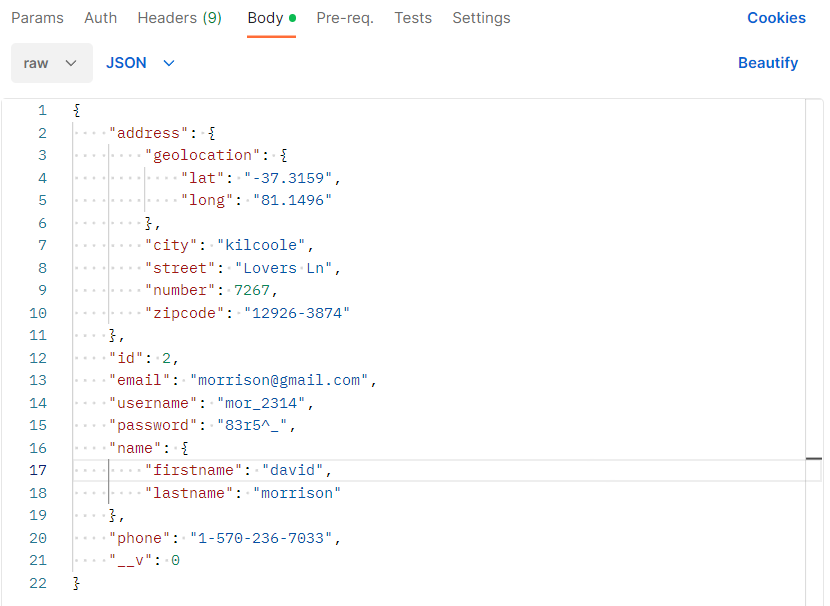
Các bước thực hiện:

Bước 1: Kích dấu **+** để thêm mới một tab cho request mới. Hoặc vào **File** 🡪 **New** chọn **HTTP**.

Bước 2: Trong tab mới, thiết lập request HTTP là **POST**.

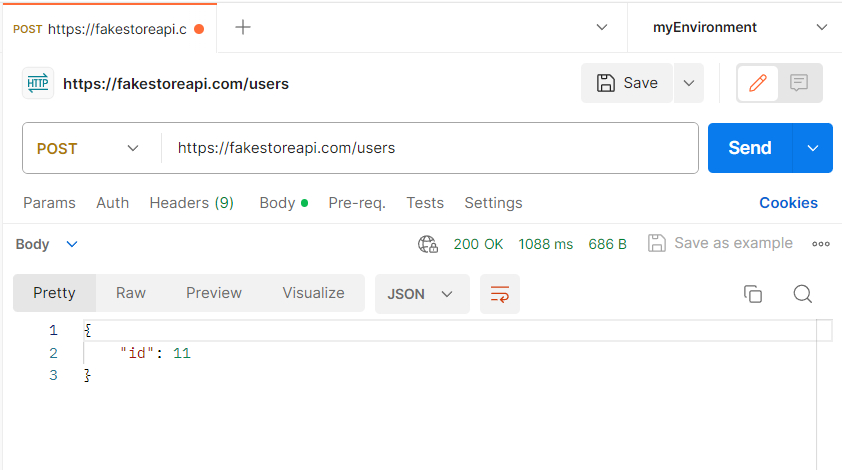
Bước 3: Nhập vào thanh URL với URL là: **https://fakestoreapi.com/users** để thêm mới một người dùng. Nếu yêu cầu đòi hỏi các tiêu đề (headers), có thể thêm chúng vào mục Headers bằng cách nhấp vào tab **Headers** và thêm các khóa và giá trị tương ứng. Sau đó, chuyển đến tab Body.

Bước 4: Trong tab Body, kích chọn **raw** để nhập dữ liệu trực tiếp. Chọn kiểu dữ liệu muốn gửi (ví dụ: JSON, XML). Sau đó, nhập dữ liệu người dùng cần thêm vào khung văn bản.



*Hình 7. Thông tin của một người dùng mới*

Bước 5: Kích nút **Send** để gửi yêu cầu POST lên server.



*Hình 8. Thêm mới thông tin người dùng*

Status hiển thị 200 OK (hoặc 201 Created). Một người dùng mới đã được tạo.

* 1. **Làm việc với Request PUT**

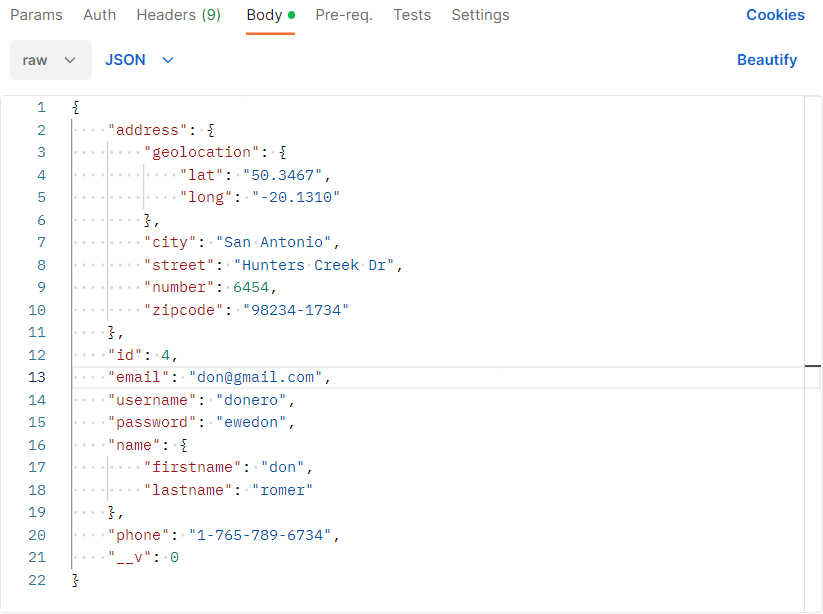
Request PUT được sử dụng khi muốn cập nhật thông tin của một tài nguyên đã tồn tại hoặc tạo mới một tài nguyên với một ID cụ thể, chẳng hạn như cập nhật thông tin của một người dùng, một bài viết, hoặc một sản phẩm. Dữ liệu cần cập nhật thường được đặt trong phần thân của yêu cầu (request body).

Các bước thực hiện:

Bước 1: Thiết lập request HTTP là **PUT**.

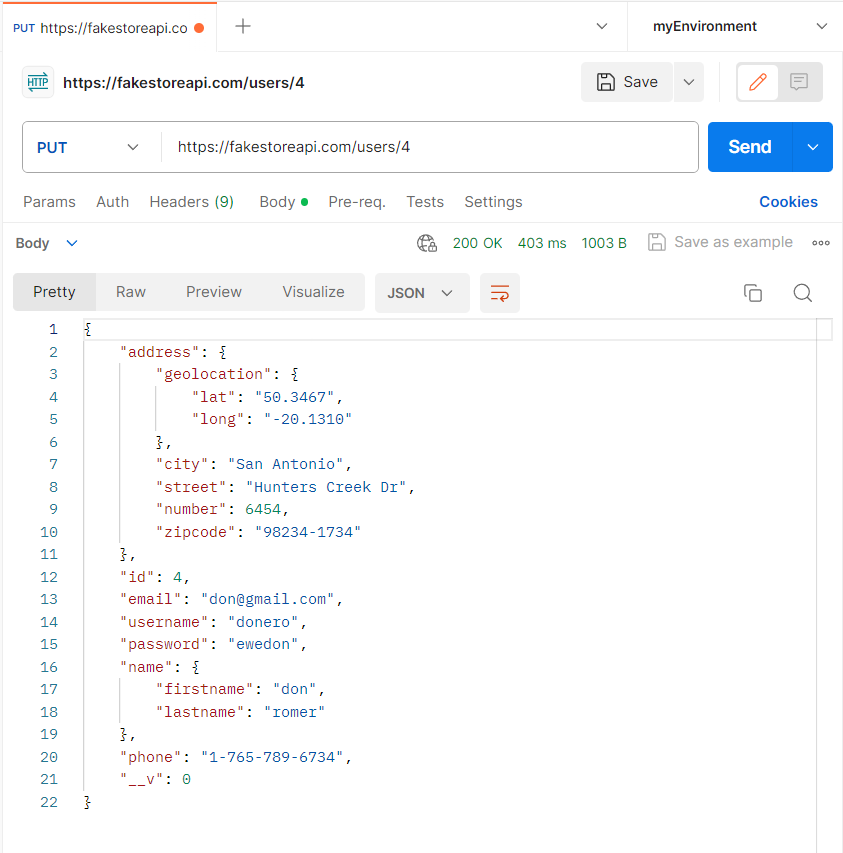
Bước 2: Nhập vào URL với URL: **https://fakestoreapi.com/users/userID** để sửa người dùng theo ID. Nếu yêu cầu đòi hỏi các tiêu đề (headers), có thể thêm chúng vào mục Headers bằng cách nhấp vào tab **Headers** và thêm các **Key** (khóa) và **Value** (giá trị) tương ứng. Sau đó, chuyển đến tab Body.

Bước 3: Trong tab Body, kích chọn **raw** để nhập dữ liệu trực tiếp. Chọn kiểu dữ liệu muốn gửi (ví dụ: JSON, XML). Sau đó, nhập dữ liệu người dùng cần chỉnh sửa vào khung văn bản.



*Hình 9. Thông tin người dùng cần chỉnh sửa*

Bước 4: Kích nút **Send** để gửi yêu cầu PUT lên server.



*Hình 10. Chỉnh sửa thông tin người dùng*

Status hiển thị 200 OK. Người dùng có ID là 4 đã được chỉnh sửa thông tin.

Lưu ý: Request PUT sử dụng để cập nhật hoặc thay thế toàn bộ tài nguyên hoặc tạo mới tài nguyên nếu nó không tồn tại vì vậy dữ liệu gửi đi chứa toàn bộ nội dung của một tài nguyên.

* 1. **Làm việc với Request PATCH**

Request PATCH được sử dụng khi muốn cập nhật một phần nhỏ của thông tin của một tài nguyên đã tồn tại. Điều này giúp giảm lượng dữ liệu cần gửi qua mạng.

Thiết lập request HTTP là **PATCH.** Các bước còn lại thực hiện tương tự như Request PUT.

Lưu ý: Request PATCH sử dụng để cập nhập một phần nhỏ của tài nguyên mà không cần gửi lại toàn bộ nội dung vì vậy dữ liệu gửi đi chỉ chứa các trường cần được cập nhật, không nhất thiết là toàn bộ nội dung.

* 1. **Làm việc với Request DELETE**

Request DELETE được sử dụng khi muốn xóa một tài nguyên cụ thể khỏi hệ thống, chẳng hạn như xóa một người dùng, một bài viết, hoặc một sản phẩm, endpoint sẽ chứa ID của người dùng cần xóa.

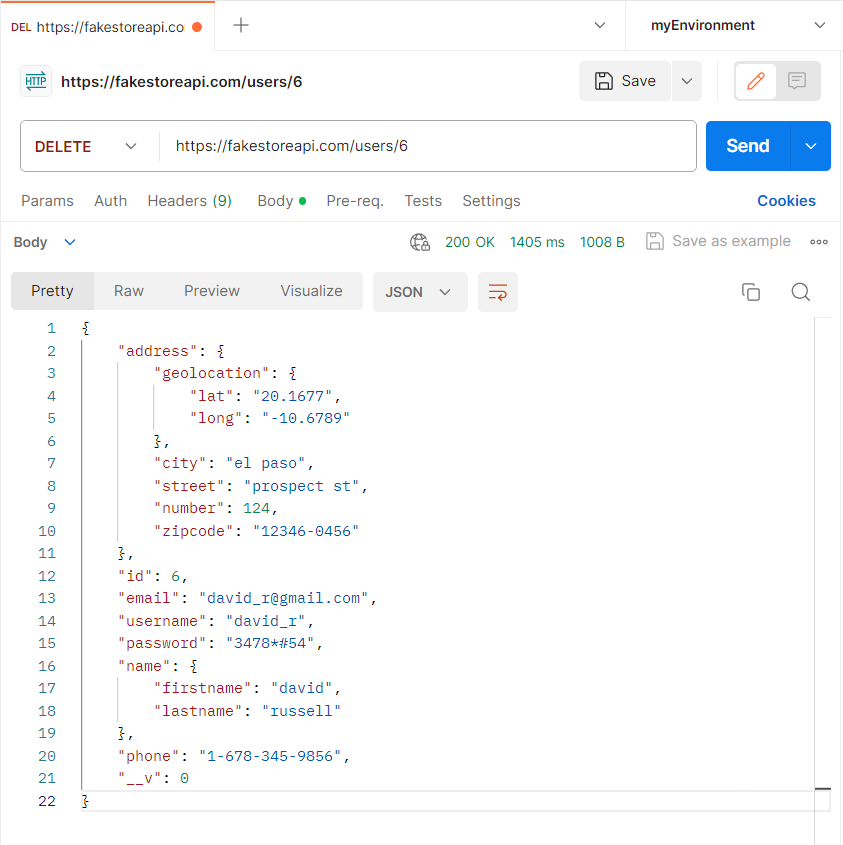
Các bước thực hiện:

Bước 1: Thiết lập request HTTP là **DELETE**.

Bước 2: Nhập vào thanh URL với URL là:

**https://fakestoreapi.com/users/userID** để xóa người dùng theo ID.

Bước 3: Kích nút **Send** để gửi yêu cầu DELETE lên server.



*Hình 11. Xóa người dùng*

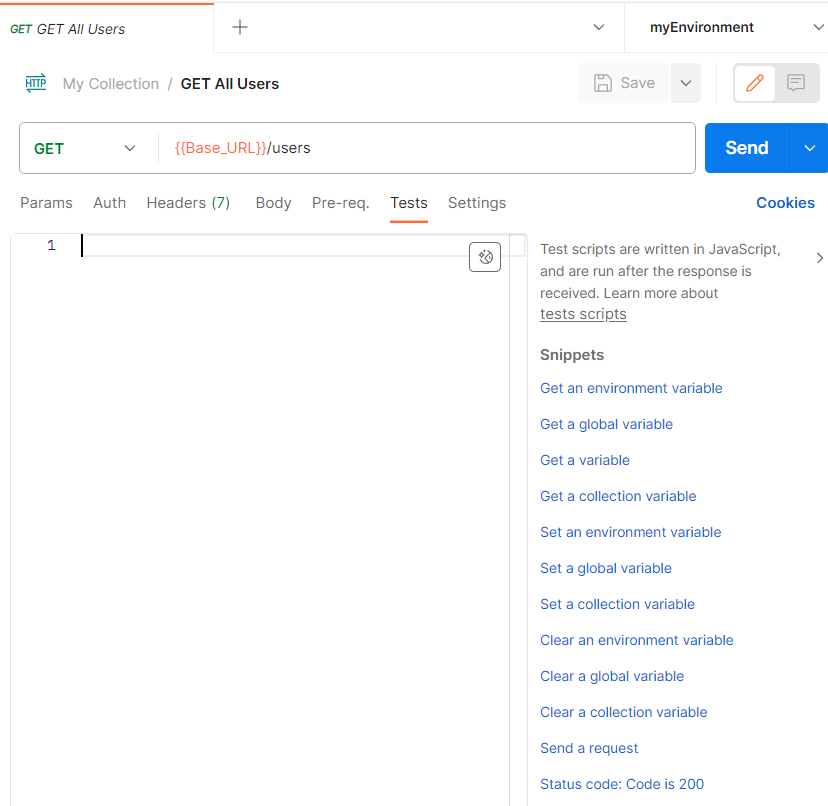
Status hiển thị 200 OK (hoặc 204 No Content). Người dùng có ID là 6 đã bị xóa.

* 1. **Tests**

Trong Postman, Tests là một phần quan trọng của mỗi Request và là nơi xác minh API có hoạt động đúng như mong đợi. Tests thường được viết bằng JavaScript và sử dụng một số phương thức và assertions được cung cấp bởi Postman.

Tạo một vài test đơn giản với request đã tạo ra ở phần trước.

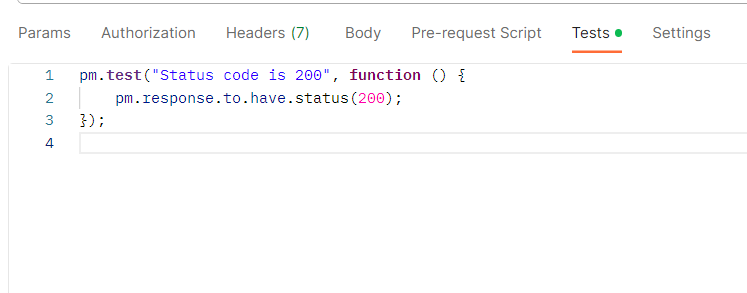
Bước 1: Chuyển đến Request GET ở phần trước, sau đó chuyển đến tab **Tests.**



*Hình 12. Giao diện Tests*

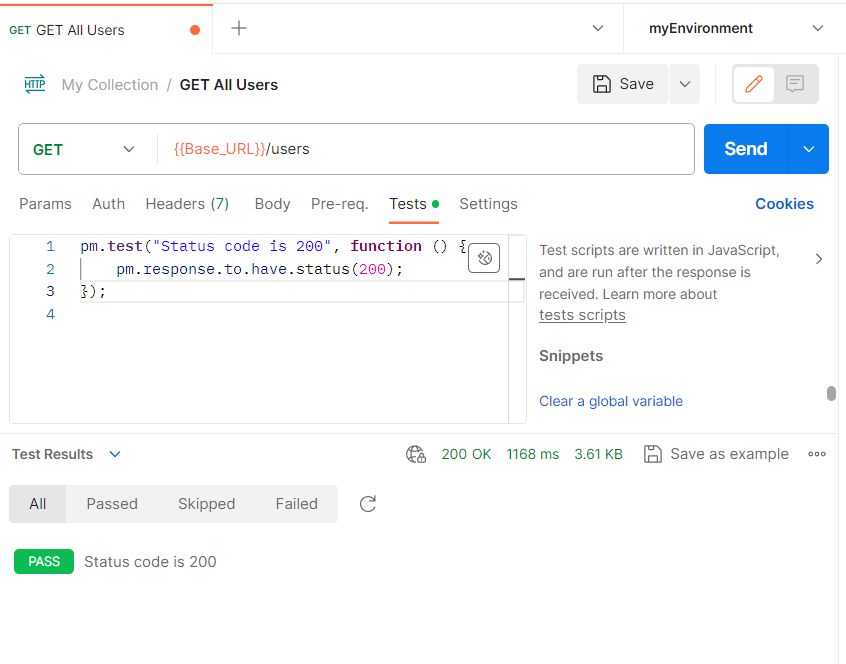
Trong phần Snippets, Postman đã cung cấp các đoạn mã mẫu (snippets) cho các Test Scripts, giúp nhanh chóng viết các kiểm thử phổ biến mà không cần ghi lại mã từ đầu.

Bước 2: Trong phần **Snippets**, chọn **Status code: Code is 200** đểkiểm tra xem mã trạng thái của phản hồi có đúng là một giá trị nhất định không.



*Hình 13. Kiểm tra Status code*

Bước 3: Kích nút **Send**. Sau đó chuyển qua tab **Test Result** trong phần Response để xem kết quả.

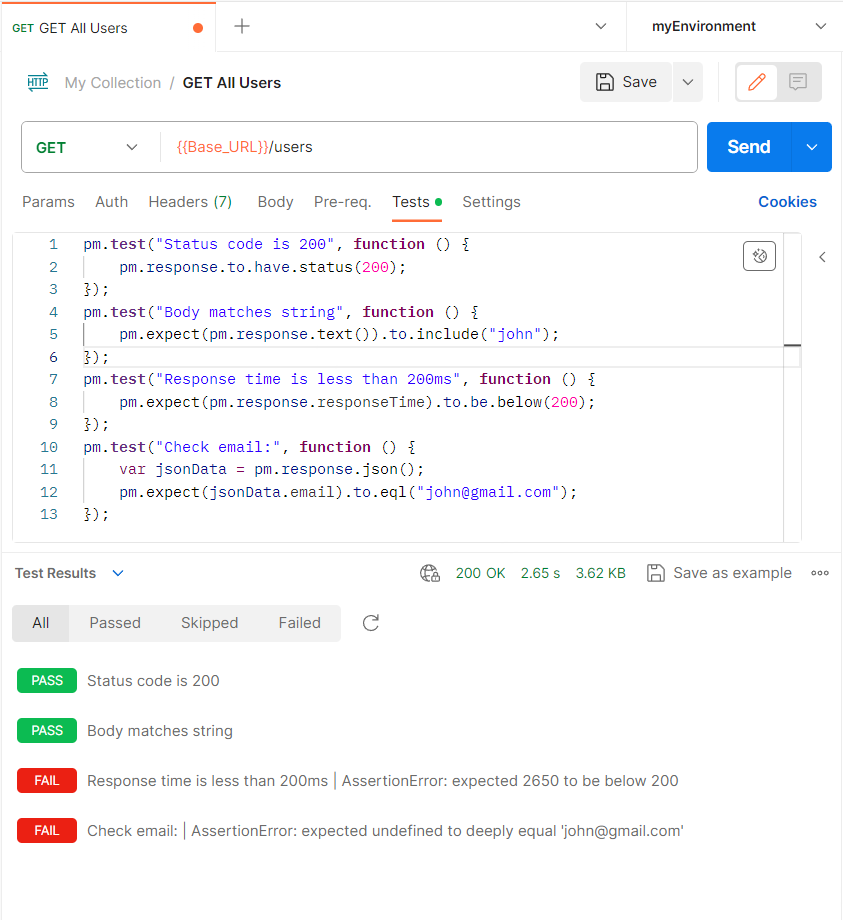


*Hình 14. Kết quả Test*

Kết quả kiểm thử sẽ hiện thị Pass (thành công) hoặc Fail (thất bại) cho từng Test Case.

Bước 4: Trở lại tab **Tests** và thêm một vài mẫu Snippets khác như: **Response body: Contain a string** (kiểm tra phản hồi có chứa một đoạn văn bản nhất định không), **Response time less than 200ms** (kiểm tra thời gian phản hồi có dưới một ngưỡng nhất định không), hay **Response body: JSON value check** (kiểm tra giá trị cụ thể của một trường trong phản hồi JSON), …

Bước 5: Kích nút **Send** và chuyển sang tab **Test Results** để xem kết quả.



*Hình 15. Một số Test case đơn giản*

Kết quả hiển thị 2 Test case Pass (thành công) và 2 Test case Fail (thất bại).

* 1. **Đặt biến**

Đặt biến là một trong những tính năng hữu ích nhất của Postman. Sử dụng biến để lưu trữ và tái sử dụng giá trị nhất định trong quá trình làm việc với request và response. Các biến này có thể là biến toàn cục (global variables) hoặc biến của một môi trường cụ thể (environment variables). Việc sử dụng biến giúp làm cho các request và các bộ kiểm thử linh hoạt, tránh lặp lại các request tương tự.

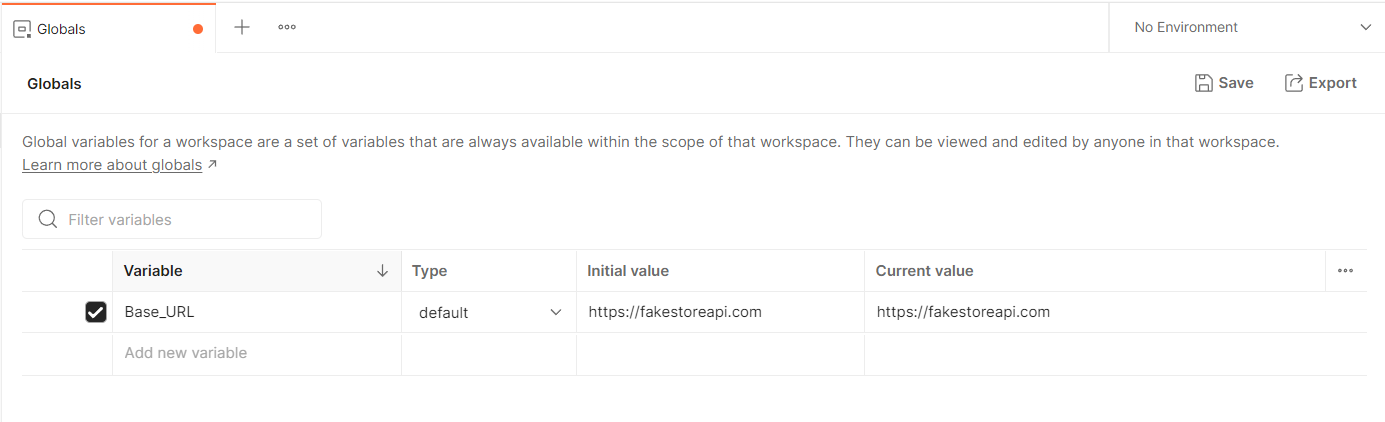
* + 1. **Biến toàn cục (Global Variables)**

Nếu có nhiều request trong một bộ kiểm thử và cần chia sẻ dữ liệu giữa chúng, có thể sử dụng biến toàn cục để lưu trữ giá trị và sử dụng lại nó trong các request khác. Biến toàn cục có thể được sử dụng trong toàn bộ môi trường làm việc.

Các bước thực hiện:

Bước 1: Trong My Workspaces, chọn **Environments** 🡪 **Global**.

Bước 2: Đặt tên biến trong trường **Variable**, giá trị của biến trong trường **Initial** **value.**

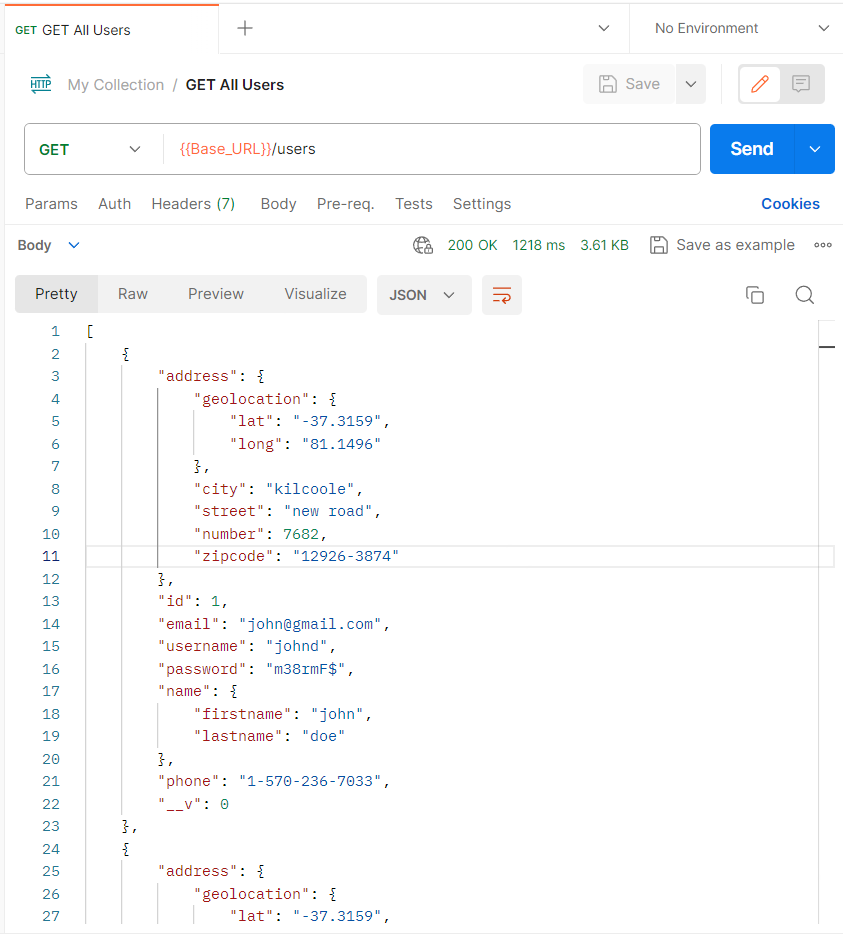


*Hình 16. Global Variables*

Bước 3: Kích nút **Save**.

Để sử dụng biến, sử dụng cú pháp **{{tên\_biến}}**.

Ví dụ: Thực hiện Request GET sử dụng biến toàn cục vừa tạo.



*Hình 17. Request sử dụng Global Variable*s

Kết quả của Request GET vẫn không thay đổi.

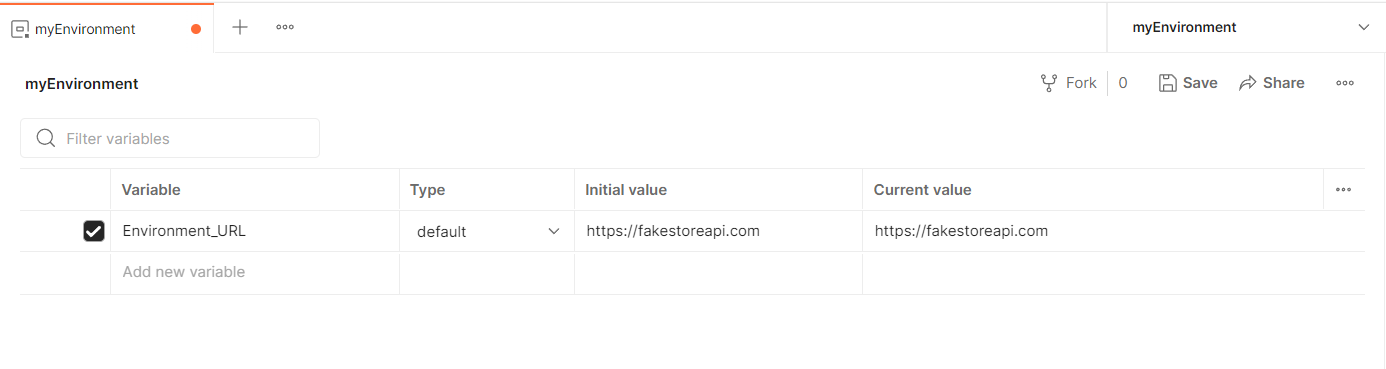
* + 1. **Biến môi trường (Environment Variables)**

Biến môi trường được sử dụng khi phát triển ứng dụng trong môi trường cục bộ (local) và cần chuyển đổi giữa các môi trường như phát triển (development), thử nghiệm (testing), và sản xuất (production), có thể sử dụng biến môi trường để lưu trữ các giá trị cấu hình phù hợp với từng môi trường. Biến môi trường chỉ có thể được sử dụng trong một môi trường cụ thể và không thể truy cập từ các môi trường khác.

Các bước thực hiện:

Bước 1: Trong My Workspaces, chọn **Environments** 🡪 **New Environment**. Có thể đặt tên cho Envionment bằng cách nhấn vào nút ••• bên phải hoặc nhấp phải chuột chọn **Rename**.

Bước 2: Đặt tên biến trong trường **Variable**, giá trị của biến trong trường **Initial** **value.**

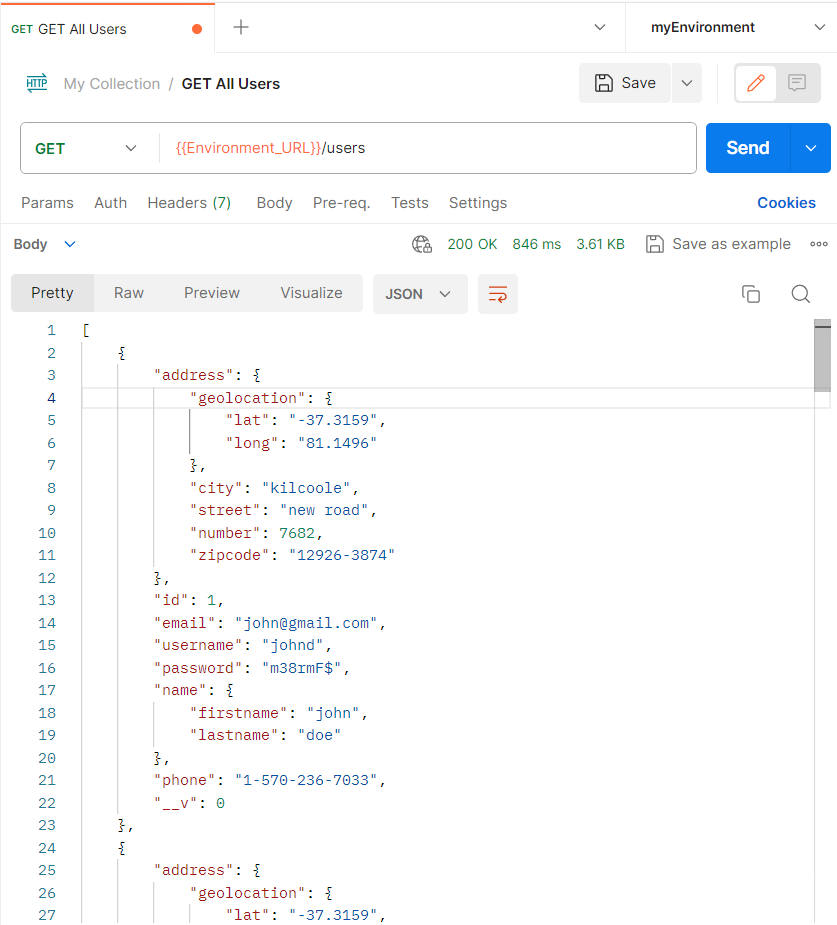
****

*Hình 18. Environment Variables*

Bước 3: Kích nút **Save**.

Cách sử dụng biến môi trường tương tự như cách sử dụng biến toàn cục.

Ví dụ: Sử dụng Request POST với biến môi trường vừa tạo.



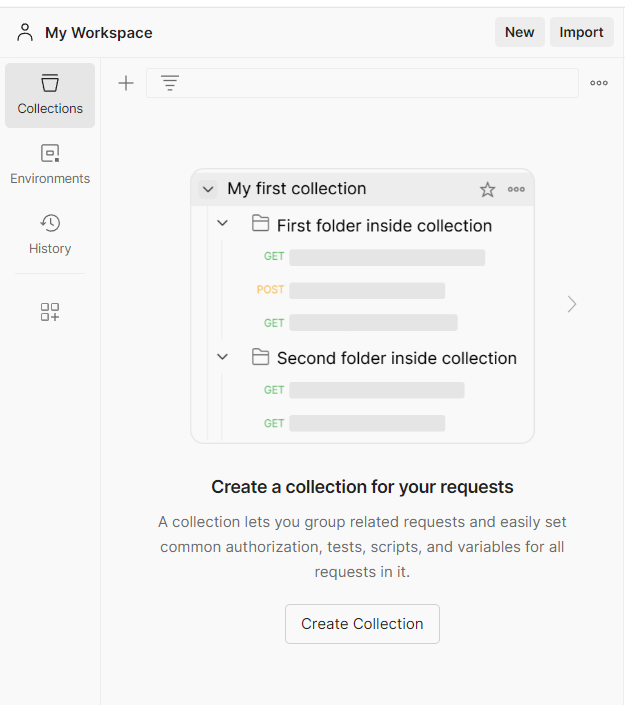
*Hình 19. Request sử dụng Environment Variables*

Kết quả của Request GET vẫn không thay đổi.

* 1. **Tạo Collection**

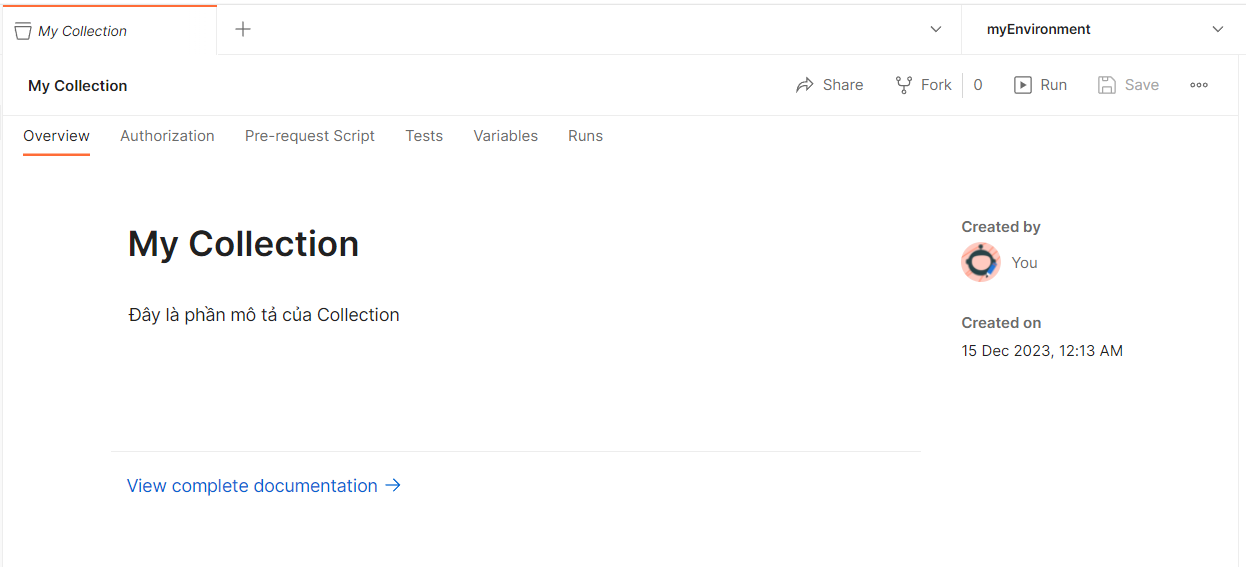
Collection đóng một vai trò quan trọng trong việc tổ chức các bộ thử nghiệm. Nó có thể được import và export để dễ dàng chia sẻ các collection giữa các nhóm. Để tạo Collection, thực hiện các bước:

Bước 1: Chọn vào **Collections** ở bên góc trái của ứng dụng. Sau đó chọn nút **Create** **Collection**.



*Hình 20. Tạo mới Collection*

Bước 2: Nhập tên Collection. Có thể nhập mô tả cho Collection ở phần khung tin nhắn.



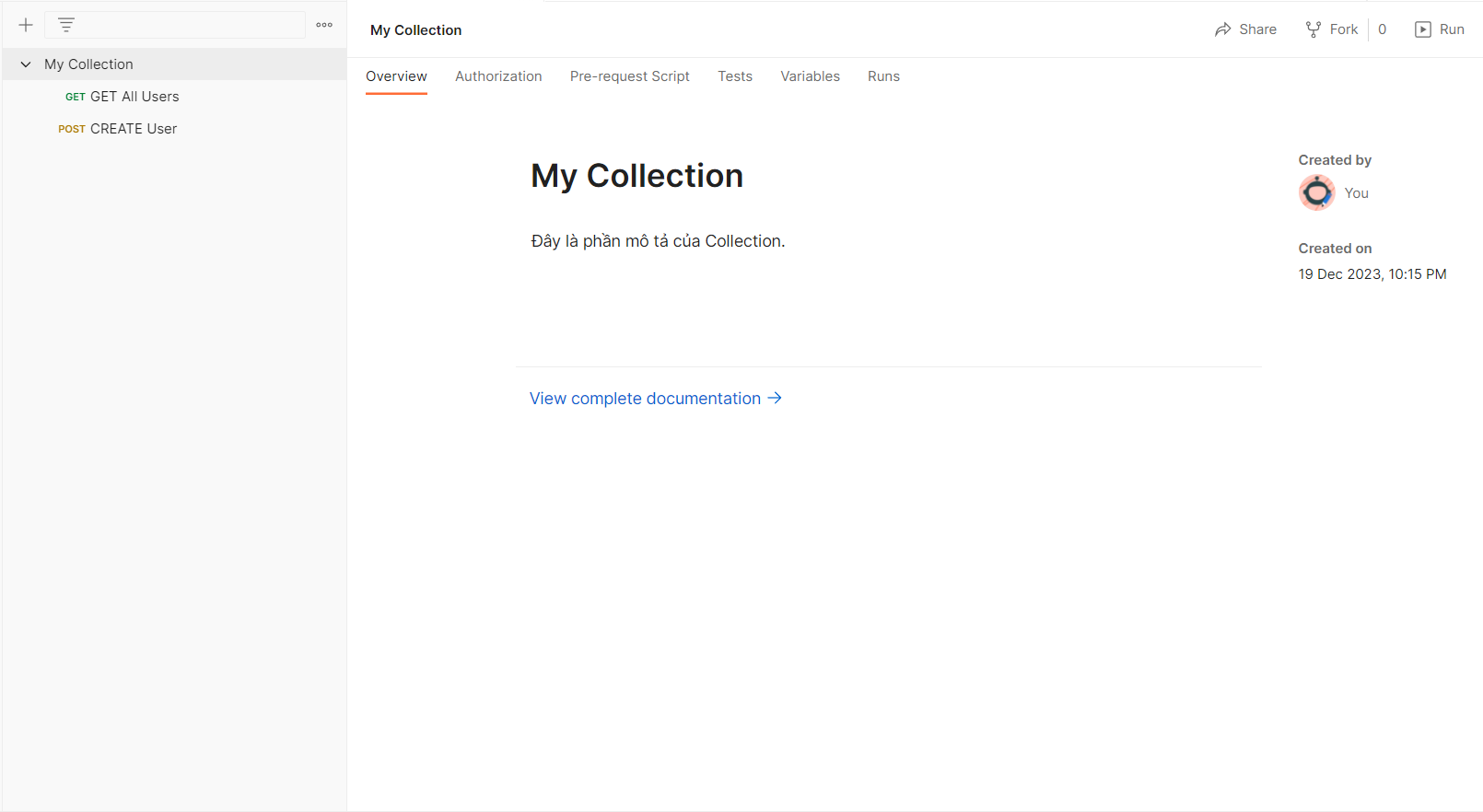
*Hình 21. Thiết lập tên và mô tả cho Collecction*

Bước 3: Trở lại Request GET phần trước, kích nút **Save**. Cửa sổ **SAVE REQUEST** hiện ra, nhấp chọn vào Collection vừa tạo.

Bước 4: Kích nút **Save**.Collection bây giờ sẽ chứa một Request.

Bước 5: Lặp lại Bước 3 - 4 cho request POST phần trước. Collection bây giờ sẽ có hai Request.

Kết quả thực hiện sẽ như hình bên dưới:



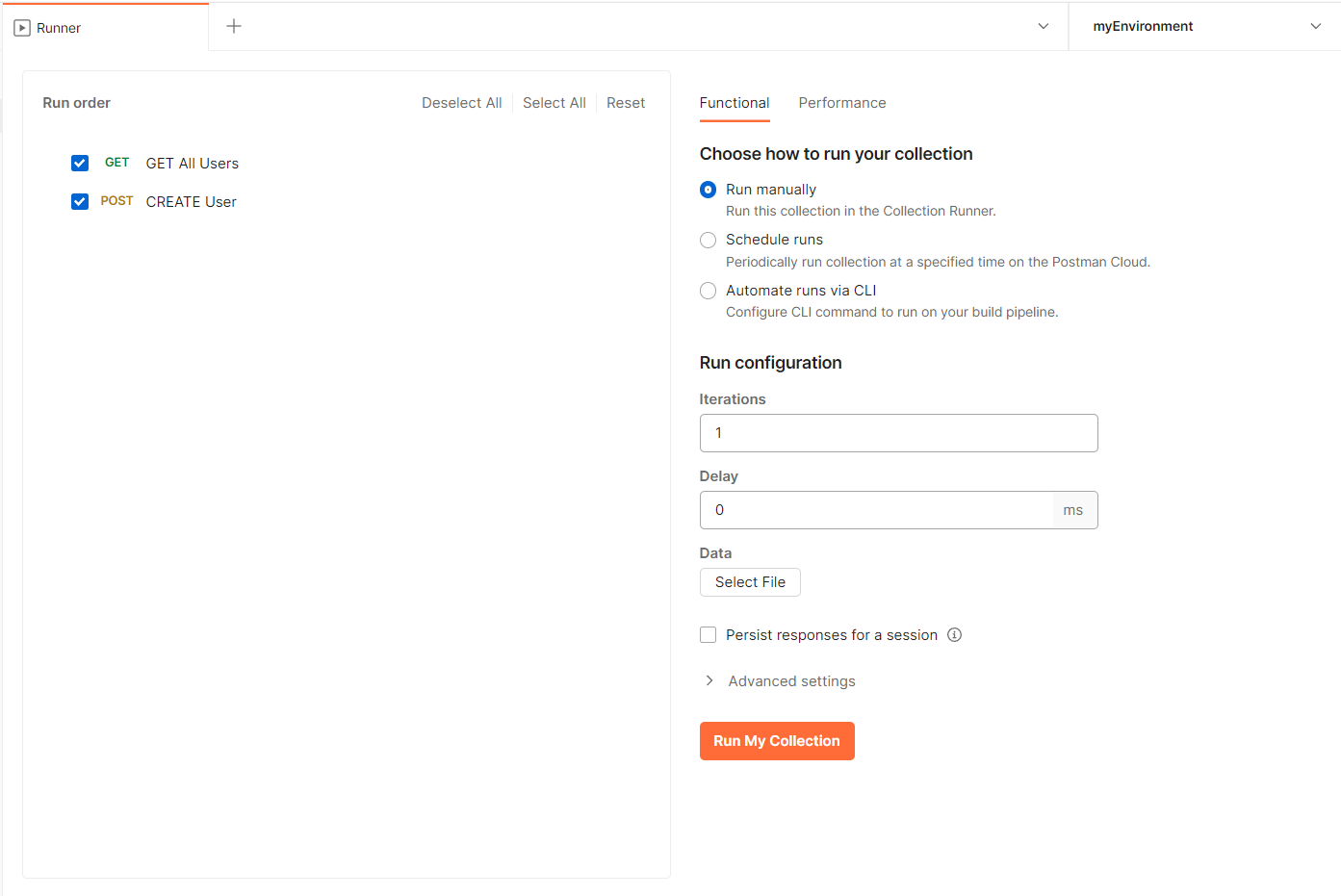
*Hình 22. Thiết lập Collection hoàn tất*

* 1. **Chạy Collection sử dụng Runner**

Runner cho phép tự động hóa các bước kiểm thử bằng cách thực hiện một loạt các request trong một Collection, thực hiện nhiều request cùng một lúc và kiểm soát kết quả của chúng.. Điều này giúp giảm thời gian và công sức so với việc thực hiện thủ công từng bước.

Các bước thực hiện:

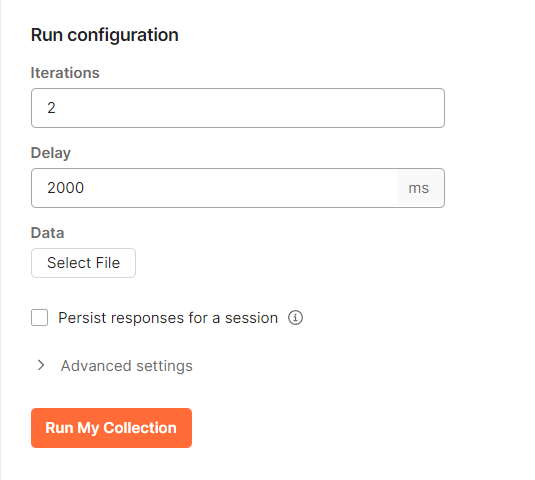
Bước 1: Nhấp phải chuột vào tên Collection, chọn **Run collection** hoặc vào **File** 🡪 **New Runner Tab**, chọn Collection muốn chạy bằng cách kéo thả Collection vào tab **Run order**. Trang  Runner sẽ xuất hiện như ở bên dưới.



*Hình 23. Runner Tab*

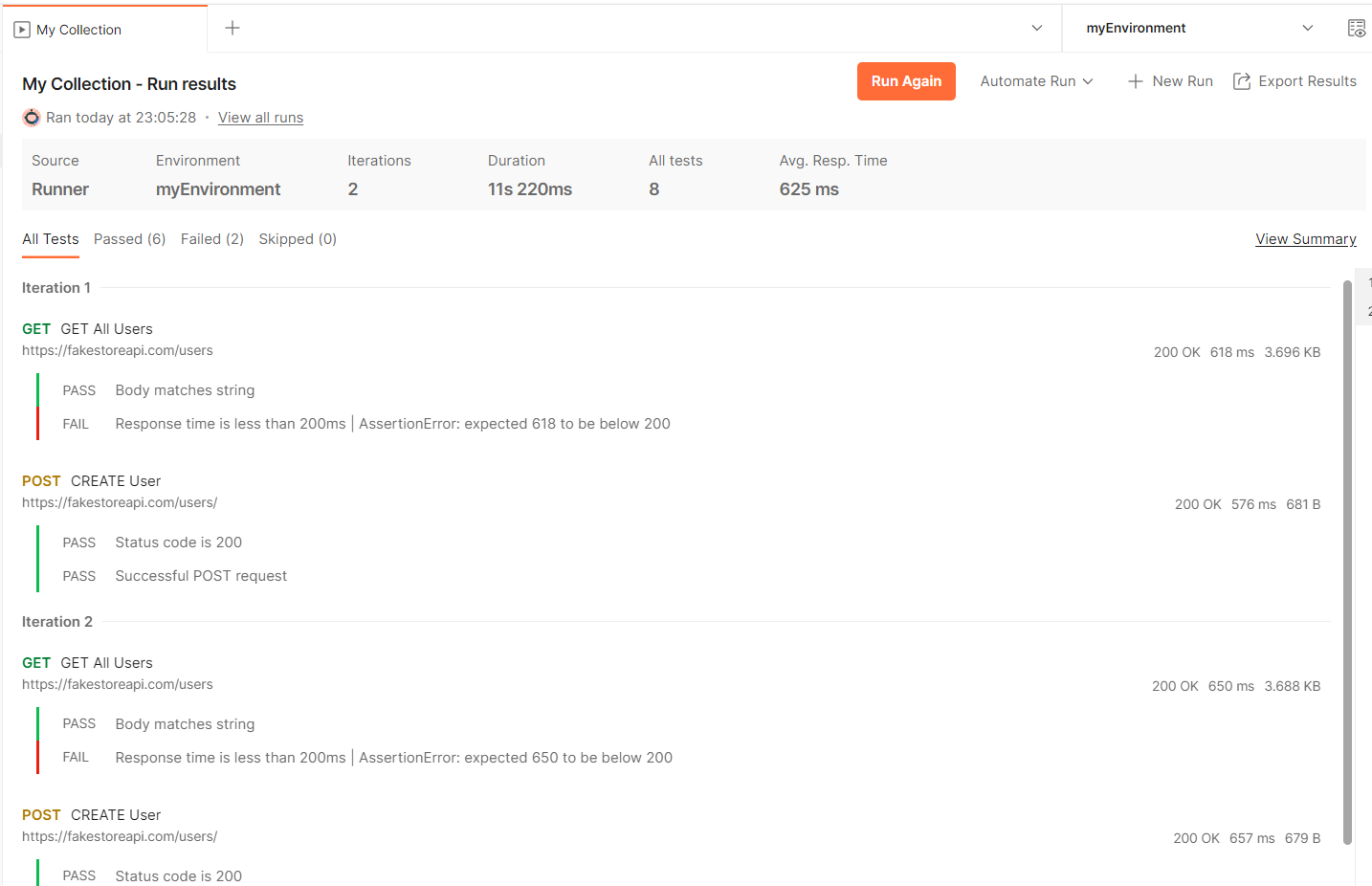
Bước 2: Chọn Environments và Variables trong danh sách (nếu có).

Bước 3: Thiết lập **Iterations** (số lần lặp) và **Delay** (thời gian chờ) trong cửa sổ **Run configuration**.



*Hình 24. Thiết lập Iterations và Delay*

Bước 4: Kích nút **Run** **<tên collection>.** Trang kết quả chạy sẽ được hiển thị sau khi kích nút Run.



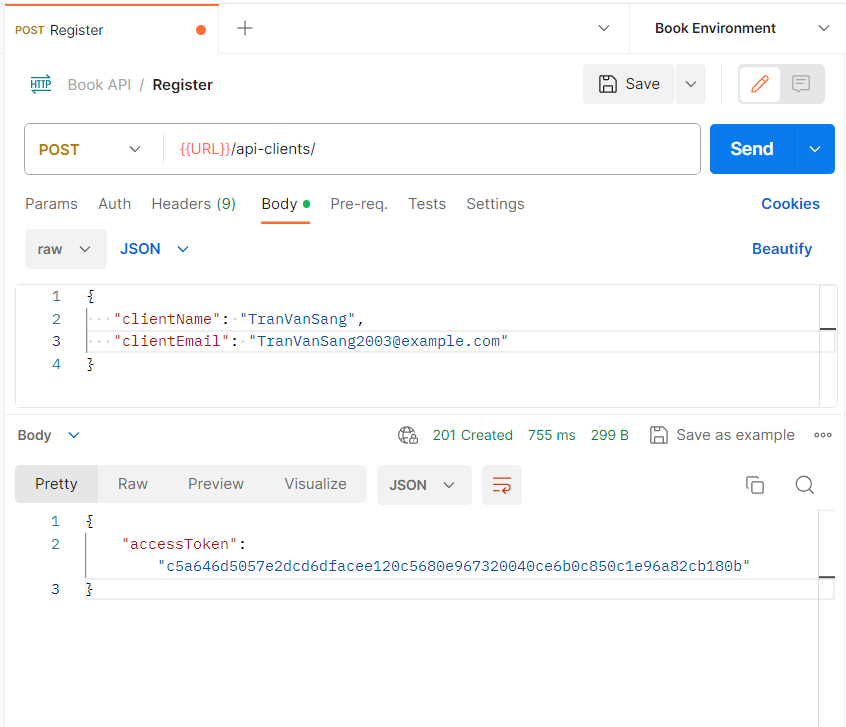
Khi test kết thúc, ta có thể nhìn thấy trạng thái Passed hoặc Failed và kết quả mỗi lần lặp (iteration).

1. KẾT QUẢ NGHIÊN CỨU
   1. Kết quả kiểm thử của các Request với Book API

Book API là một dịch vụ web API đơn giản được thiết kế để quản lý thông tin về sách. API này cung cấp các chức năng cơ bản để thêm, sửa đổi, xóa và truy xuất thông tin đơn hàng và thông tin về sách.

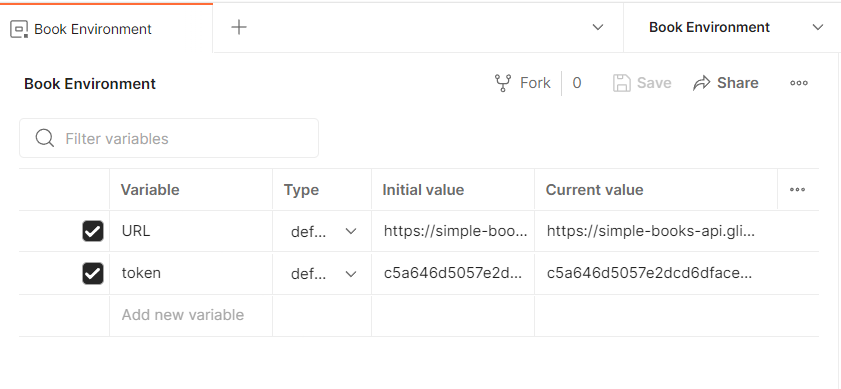
* + 1. Authentication với Book API

Đăng kí một tài khoản với clientName (tên máy khách) và clientEmail (Email máy khách) bằng cách thực hiện Request POST.



*Hình 25. Đăng kí tài khoản*

Sau khi đăng kí thành công, phần Body sẽ trả về một chuỗi Access Token cùng với trạng thái 201 (Created).

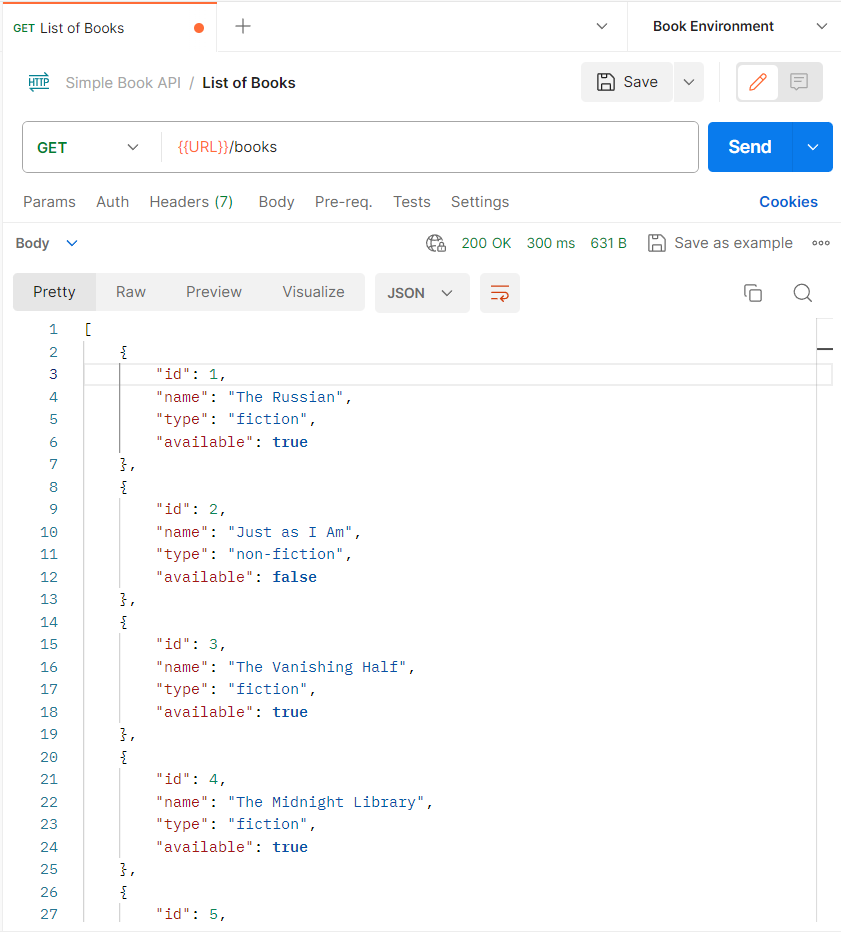


*Hình 26. Lưu Access Token*

Lưu Access Token vừa nhận được vào Biến môi trường để phục vụ các request tiếp theo.

* + 1. Kết quả thực hiện với Request GET
       1. Lấy danh sách các quyển sách

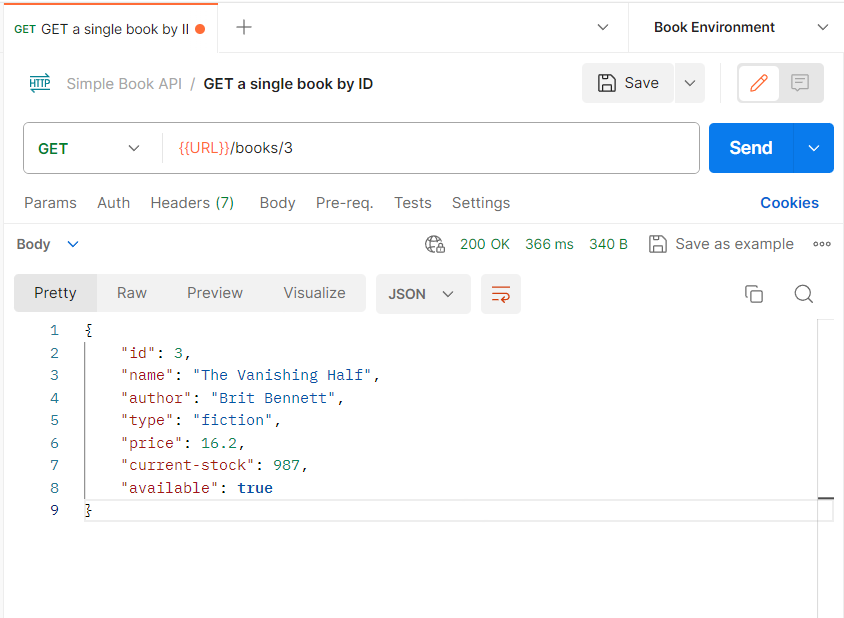
Lấy danh sách tất cả các quyển sách có sẵn trong nguồn dữ liệu bằng cách thực hiện Request GET.



*Hình 27. Lấy danh sách các quyển sách*

* + - 1. Lấy một quyển sách theo ID

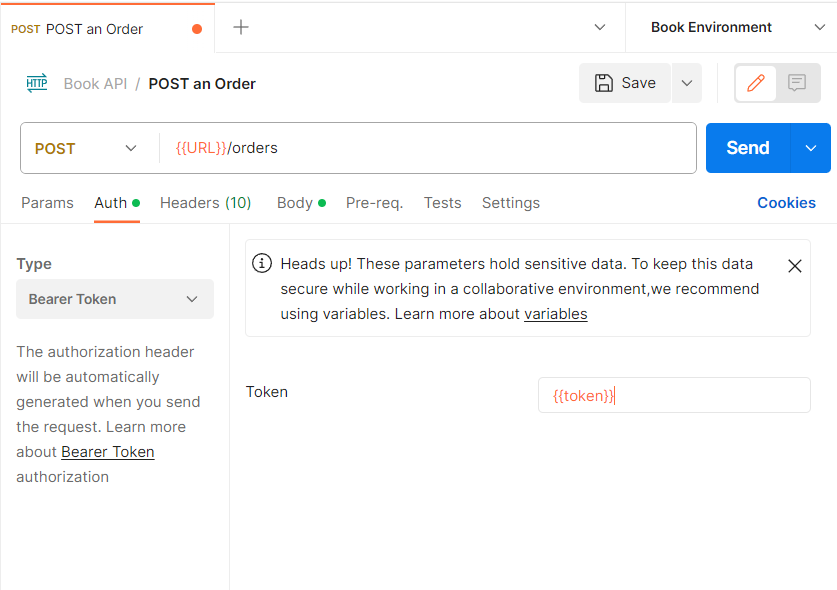
Lấy một quyển sách trong nguồn dữ liệu bằng cách thực hiện Request GET.



*Hình 28. Lấy một quyểm sách theo ID*

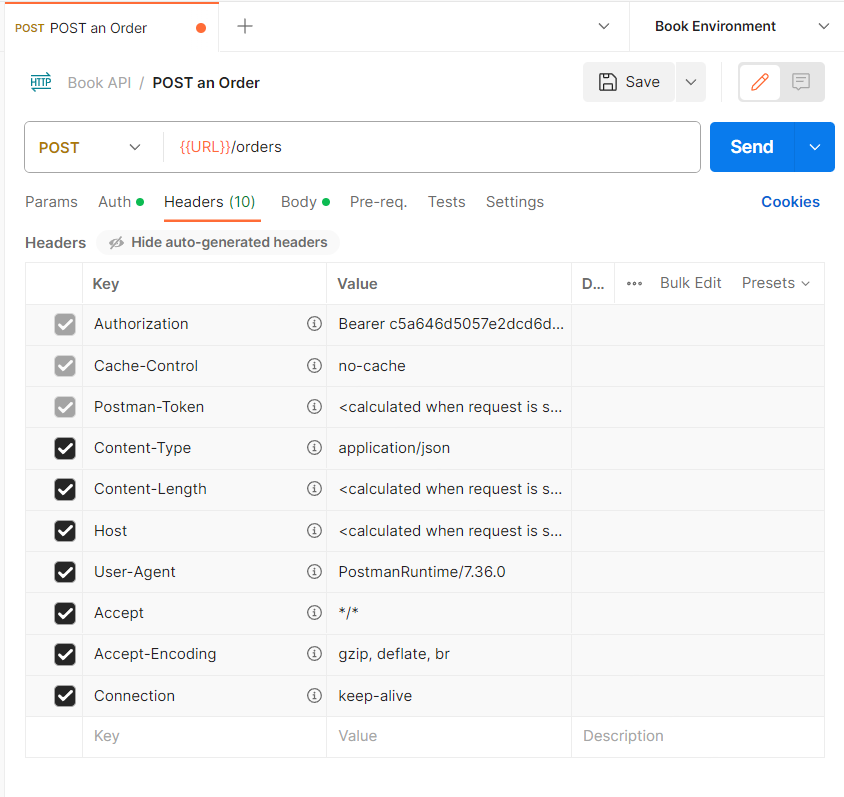
* + 1. Kết quả thực hiện với Request POST

Để thực hiện được Request POST (gửi đơn đặt hàng), cần phải đính kèm Token trong phần Headers để xác minh danh tính và quyền truy cập của người dùng khi đặt hàng.



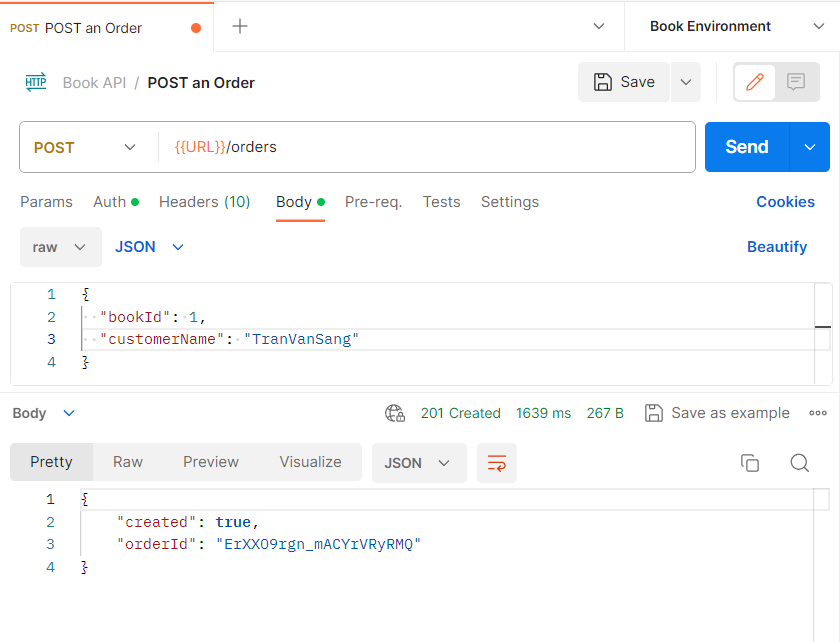
*Hình 29. Cài đặt Token*

Request trên sử dụng loại Authentication là Bearer Token. Trong phần Token, sử dụng biến vừa lưu ở request trên.



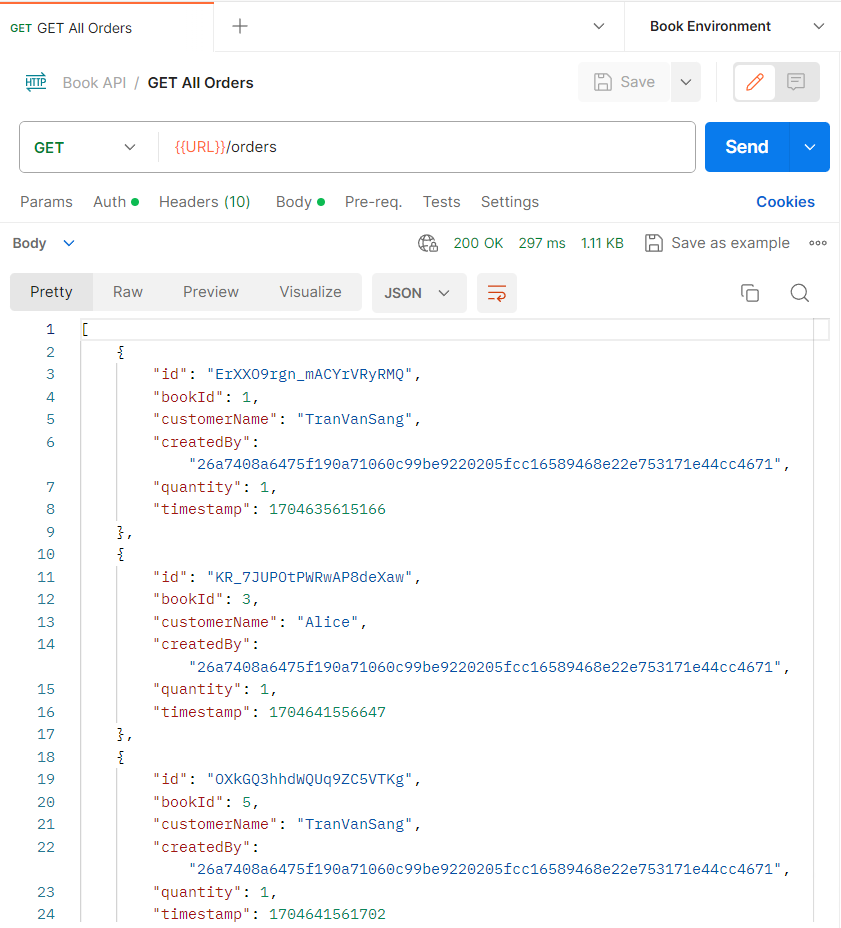
*Hình 30. Request POST sử dụng Bearer Token*

Trong tab Body, thêm thông tin người dùng và ID của quyển sách cần đặt hàng.



*Hình 31. Gửi đơn đặt hàng*

Như vậy thông tin về đơn hàng và người đặt hàng đã được gửi đi. Có thể xem xem danh sách các đơn hàng đã đặt bằng Request GET.

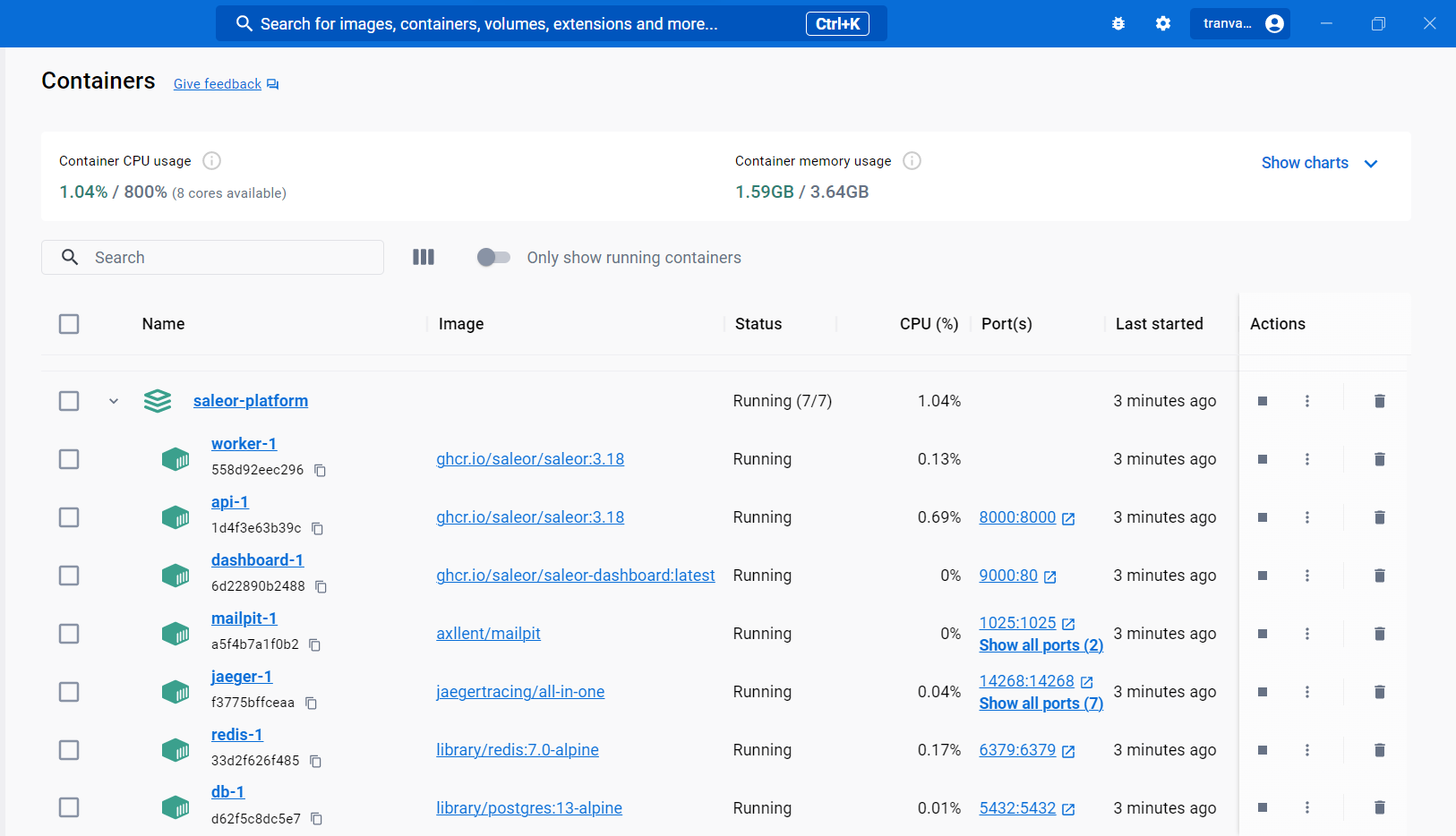


*Hình 32. Danh sách đơn hàng đã đặt*

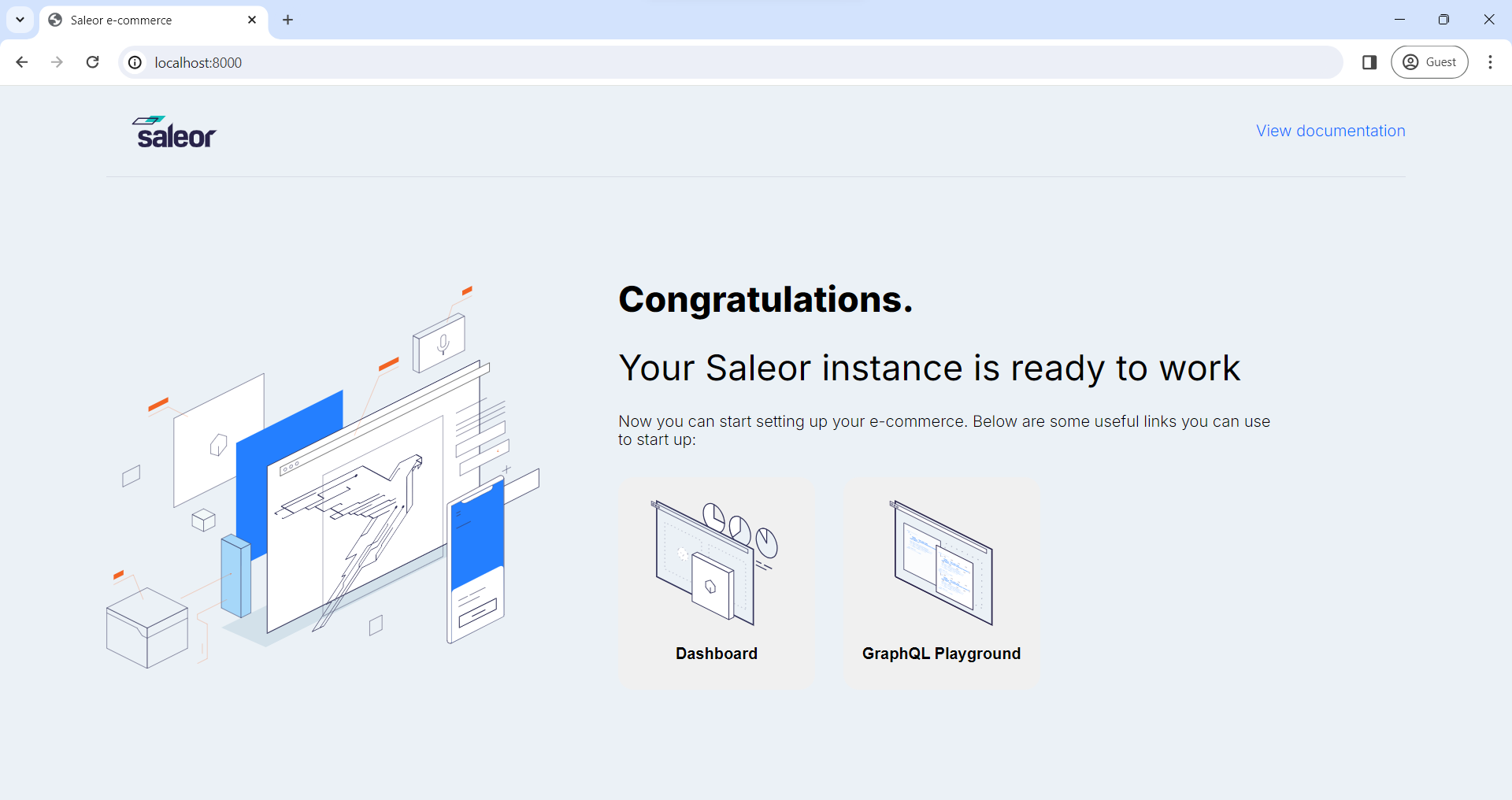
* 1. Kết quả triển khai ứng dụng Backend trên Docker
     1. Triển khai SaleOr trên Docker

Saleor là một hệ thống mã nguồn mở để xây dựng cửa hàng trực tuyến (e-commerce) được xây dựng trên nền tảng Python và Django. Nó cung cấp một cách linh hoạt để tạo ra các trang web bán hàng trực tuyến với khả năng tùy chỉnh cao.

Saleor giúp người phát triển xây dựng các cửa hàng trực tuyến độc đáo với các tính năng như quản lý sản phẩm, quản lý đơn đặt hàng, thanh toán, và nhiều tính năng khác. Do là mã nguồn mở, nên có thể tùy chỉnh và mở rộng Saleor theo nhu cầu cụ thể.



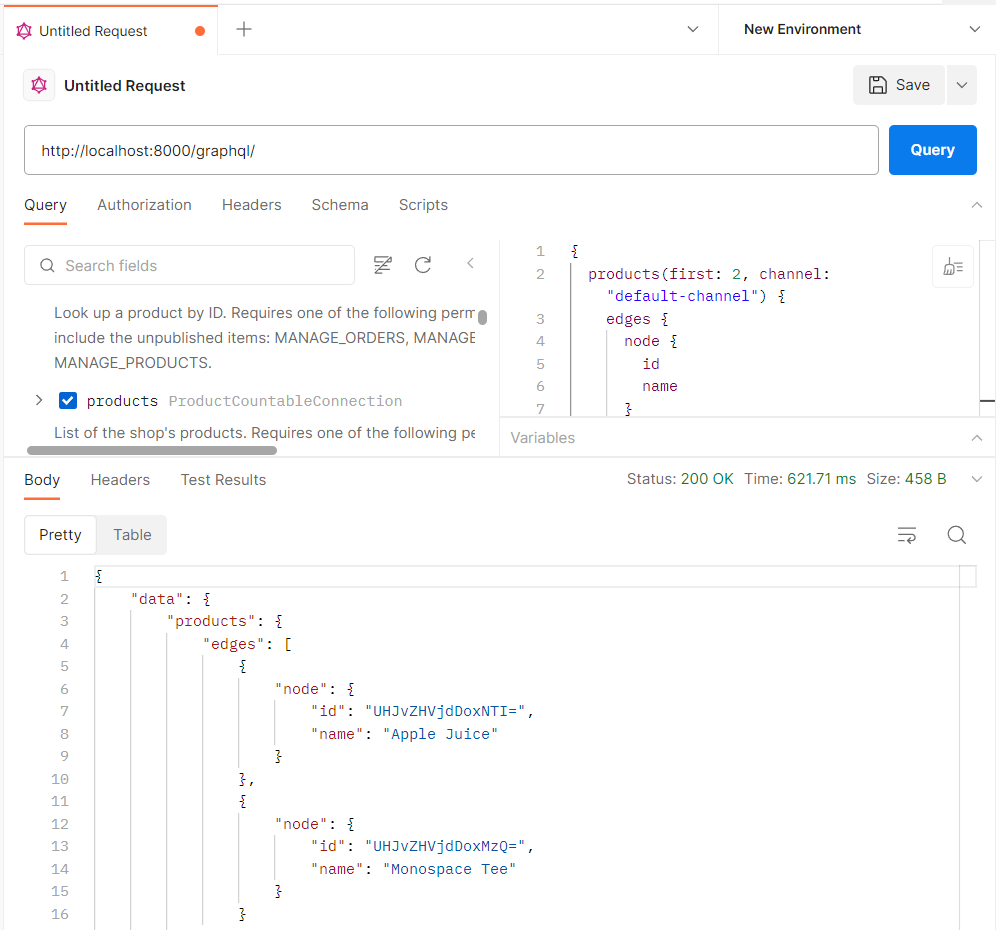
*Hình 33. Triển khai SaleOr trên Docker Deskstop*



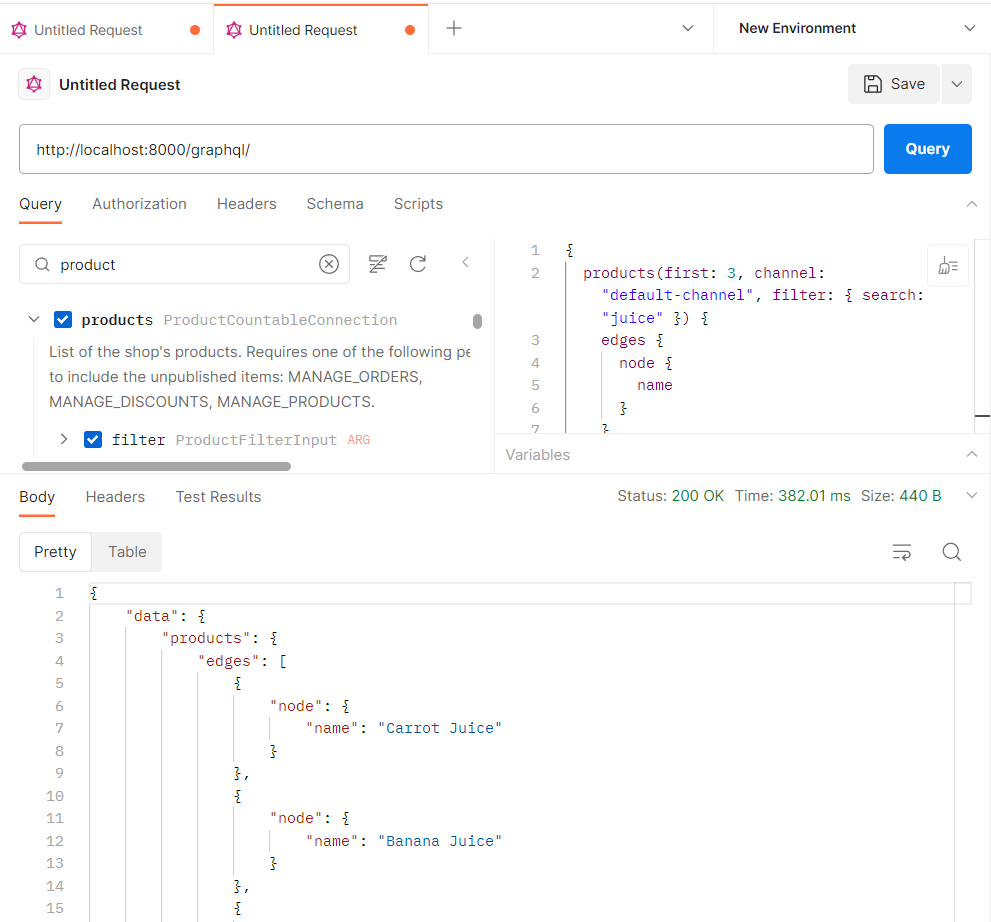
*Hình 34. Kết quả triển khai SaleOr*

* + 1. Kiểm thử với GraphQL

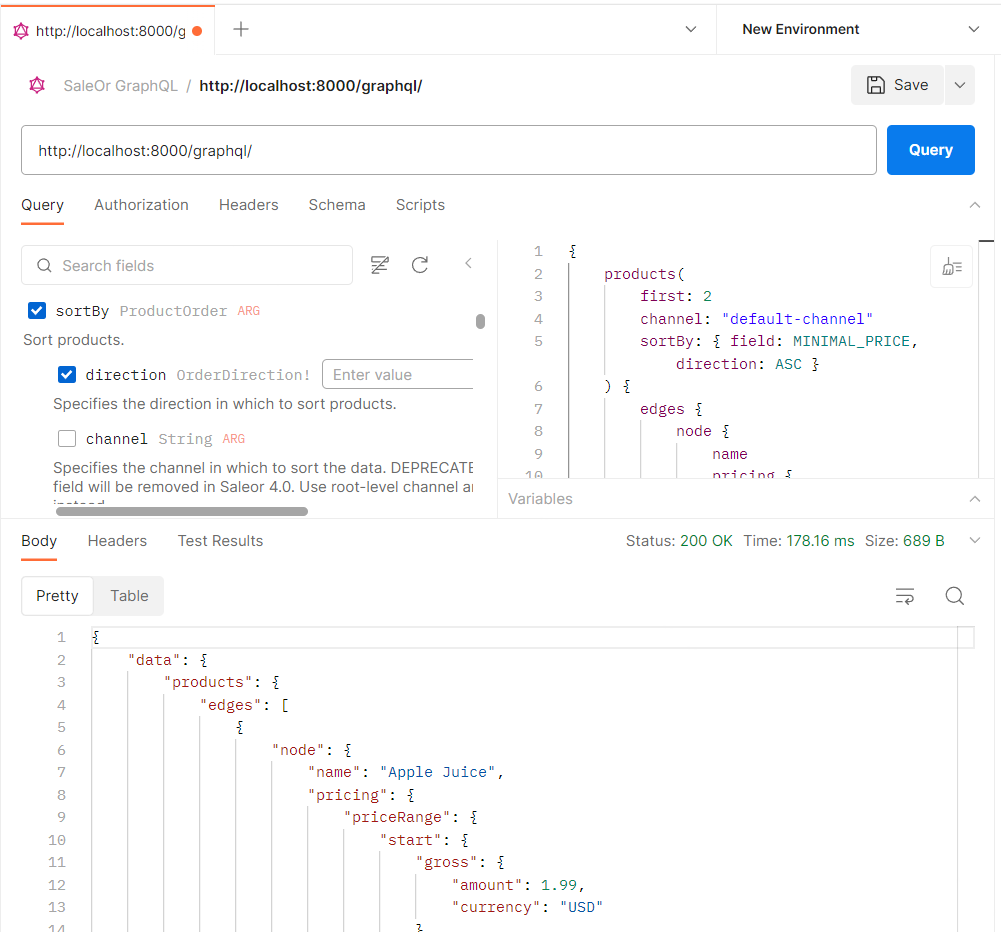
Một số Request đơn giản với SaleOr:



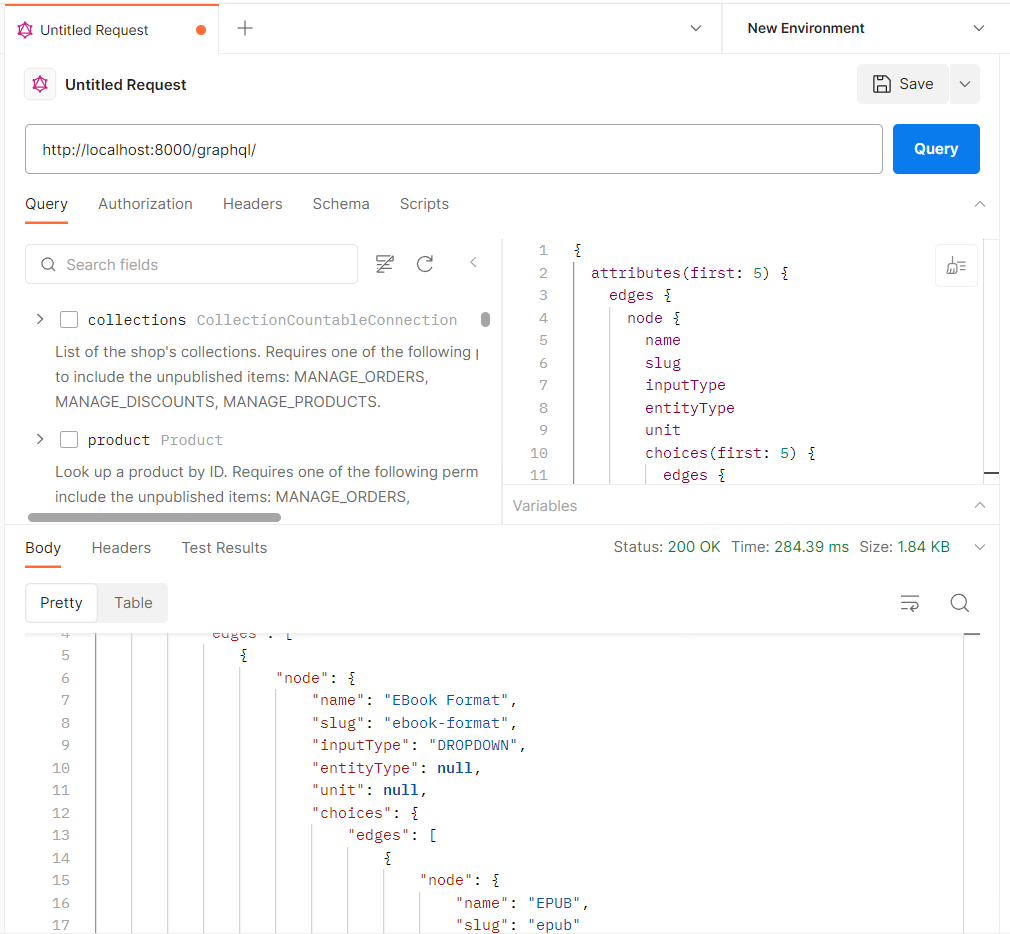
*Hình 35. Lấy danh sách sản phẩm*



*Hình 36. Lọc sản phẩm theo tên*



*Hình 37. Sắp xêp các sản phẩm theo giá*



*Hình 38. Lấy tất cả các thuộc tính*

1. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN
   1. **Kết luận**

Qua quá trình thực hiện đồ án, nghiên cứu và đánh giá các tính năng của Postman trong việc kiểm tra RESTful API, ta có thể nhân thấy Postman là một công cụ mạnh mẽ và linh hoạt, cung cấp nhiều tính năng giúp cải thiện hiệu năng, trải nghiệm người dùng. Postman là một công cụ hữu ích cho việc kiểm tra RESTful API, đặc biệt là đối với các dự án có quy mô lớn và phức tạp.

* 1. **Đề xuất mới**

Bảo mật trong quá trình kiểm thử: Xem xét các biện pháp bảo mật khi sử dụng Postman để kiểm thử API. Điều này bao gồm cách quản lý thông tin đăng nhập, mã hóa dữ liệu, và xác thực để đảm bảo an toàn và bảo mật trong quá trình kiểm thử.

Kiểm thử tích hợp với dịch vụ Cloud: nghiên cứu cách sử dụng Postman để kiểm thử API tích hợp với các dịch vụ đám mây như AWS, Azure hoặc Google Cloud Platform. Điều này sẽ giúp hiểu rõ cách Postman tương tác với môi trường đám mây và giải quyết các thách thức đặc biệt của nền tảng đó.

Xây dựng bộ kiểm thử tự động từ Postman: đề xuất cách sử dụng các kịch bản Postman để tự động hóa việc xây dựng bộ kiểm thử tự động. Điều này có thể tích hợp vào các frameworks tự động hóa kiểm thử như Selenium để đảm bảo kiểm thử toàn diện từ cả API đến giao diện người dùng.

Tóm lại, Postman là một công cụ test API mạnh mẽ và linh hoạt, cung cấp nhiều tính năng giúp cải thiện hiệu năng, trải nghiệm người dùng và giao diện chức năng. Qua quá trình thực hiện đồ án, chúng tôi đã đánh giá cao các tính năng của Postman và nhận thấy Postman là một công cụ hữu ích cho việc kiểm tra RESTful API.

**DANH MỤC TÀI LIỆU THAM KHẢO**

[1] Ankit Sobti, Abhinav Asthana, and Abhijit Kane, "What is an API?", 2023. [Trực tuyến]. URL: https://www.postman.com/what-is-an-api/ Truy cập [10/12/2023].

[2] Arlemi Turpault., "A guide to the different types of APIs", 06/07/2022. [Trực tuyến]. URL: https://blog.postman.com/different-types-of-apis/ Truy cập [10/12/2023].

[3] Adam Selipsky, "What is a RESTful API?", 2023. [Trực tuyến]. URL: https://aws.amazon.com/what-is/restful-api/ Truy cập [20/12/2023].

[4] Ankit Sobti, Abhinav Asthana, and Abhijit Kane, "API authentication", 2023. [Trực tuyến]. URL: https://www.postman.com/api-platform/api-authentication/ Truy cập [24/12/2023].

[5] Ankit Sobti, Abhinav Asthana, and Abhijit Kane, "What is Postman?", 2023. [Trực tuyến]. URL: https://www.postman.com/product/what-is-postman/ Truy cập [18/12/2023].

[6] Wikipedia, "Postman (software)", 2023. [Trực tuyến]. URL: https://en.wikipedia.org/wiki/Postman\_(software) Truy cập [24/12/2023].