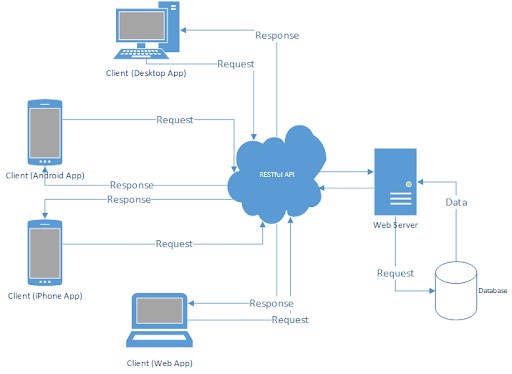
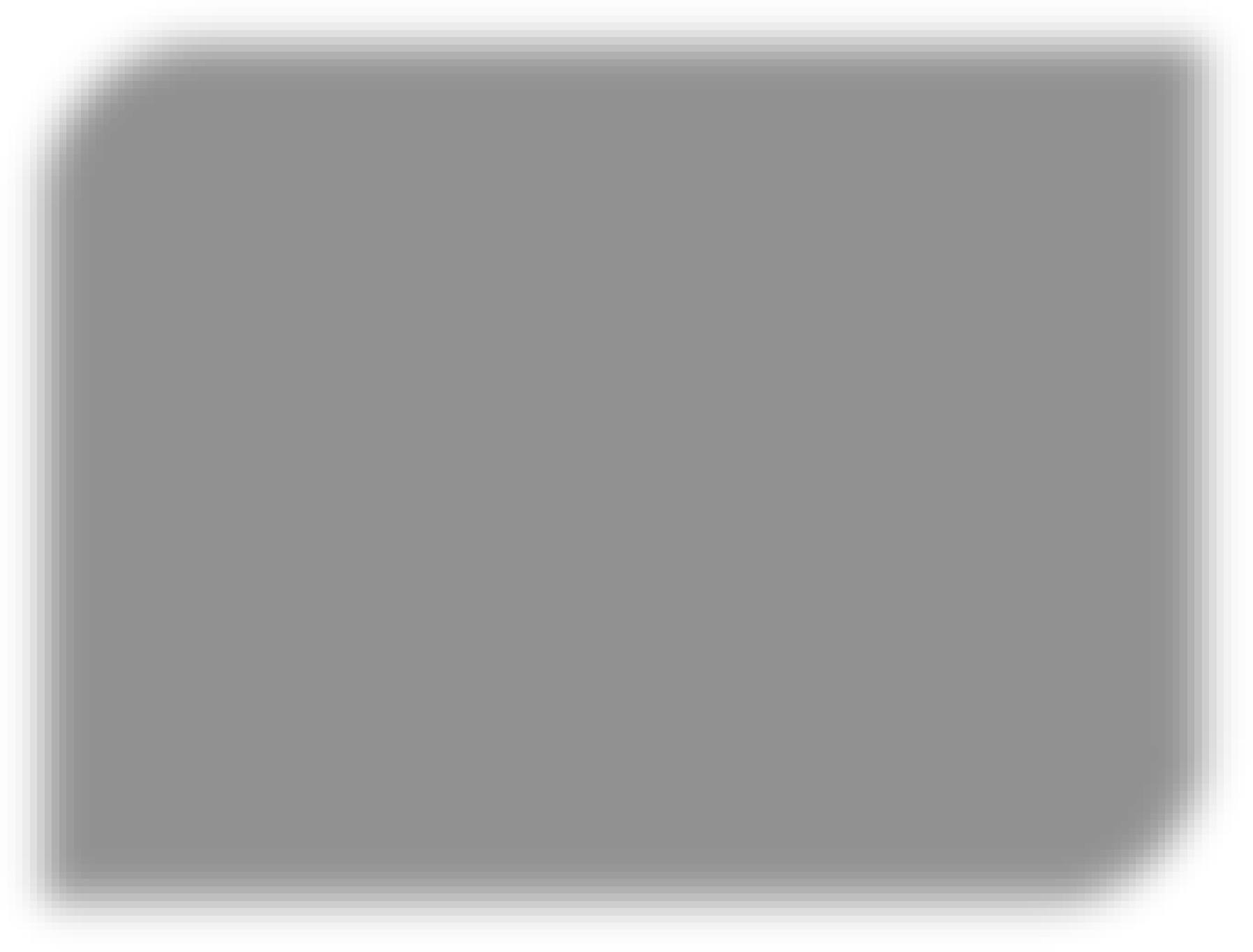
|  |
| --- |
| **­­­­­­­­LAPORAN PRAKTIKUM PENGENALAN APLIKASI BERBASIS INTERNET**  **PENGENALAN RESTFUL API** |
| ­­­­ |
| **Agus Pranata Marpaung**  **13323033**  **DIII TEKNOLOGI KOMPUTER** |
| **INSTITUT TEKNOLOGI DEL**  **FAKULTAS VOKASI** |

**Judul Praktikum**

|  |  |  |
| --- | --- | --- |
| **Minggu/Sesi** | : | VII/2 |
| **Kode Mata Kuliah** | : | 1331204 |
| **Nama Mata Kuliah** | : | PENGENALAN APLIKASI BERBASIS INTERNET |
| **Setoran** | : | *Softcopy* |
| **Batas Waktu Setoran** | : | *10 Maret 2024 jam ??:??* |
| **Tujuan** | : | 1. *Pengenalan Restful API* |

**RESTful Web Services**

**Konsep**



**API**

API (Application Protocol Interface) adalah kumpulan dari subroutine definitions, protocol dan juga tools untuk berkomunikasi data antar aplikasi software.

**Web service**

Web service adalah standar yang digunakan untuk melakukan pertukaran data antar aplikasi atau sistem, karena aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada platform yang berbeda. Contoh implementasi dari web service antara lain adalah SOAP dan REST.

Web service yang berbasis arsitektur REST kemudian dikenal sebagai RESTful web services. Layanan web ini menggunakan metode HTTP untuk menerapkan konsep arsitektur REST.REST adalah salah satu implementasi dari web service sebagai sebuah standar yang digunakan untuk pertukaran data antar aplikasi atau sistem.

Biasanya aplikasi atau sistem ini menggunakan bahasa pemrogramman yang berbeda sehingga untuk bisa berkomunikasi satu sama lainnya bisa menggunakan web service ini.

**REST (REpresentational State Transfer)**

merupakan standar arsitektur berbasis web. Jadi bisa dibilang, REST itu adalah salah satu dari desain arsitektur di dalam API.

Meskipun REST dapat digunakan di hampir semua protokol, tapi biasanya memanfaatkan HTTP ketika digunakan untuk Web API. Hal ini membantu pengembang web tidak perlu menginstal library atau perangkat lunak tambahan untuk memanfaatkan desain REST API.

Design REST API pertama kali diperkenalkan oleh Dr. Roy Fielding dalam disertasi doktor tahun 2000-nya. REST API terkenal karena fleksibilitasnya yang luar biasa.

Data tidak terikat dengan metode dan sumber daya, REST memiliki kemampuan untuk menangani beberapa jenis panggilan, mengembalikan format data yang berbeda dan bahkan mengubah secara struktural tentunya dengan implementasi yang benar.

Rest API secara eksplisit memanfaatkan metodologi HTTP yang ditentukan oleh **protokol RFC 2616.**

REST yang digunakan oleh browser dapat dianggap sebagai bahasa internet. Dengan meningkatnya penggunaan cloud, API muncul untuk mengekspos layanan web.

REST adalah pilihan logis untuk membangun API yang memungkinkan pengguna untuk terhubung dan berinteraksi dengan layanan cloud. API telah banyak digunakan oleh situs-situs seperti Amazon, Google, LinkedIn dan Twitter.

Pada arsitektur REST, REST server menyediakan resources (sumber daya/data) dan REST client mengakses dan menampilkan resource tersebut untuk penggunaan selanjutnya.

Setiap resource diidentifikasi oleh URIs (Universal Resource Identifiers) atau global ID. Resource tersebut direpresentasikan dalam bentuk format teks, JSON atau XML. Pada umumnya formatnya menggunakan JSON dan XML.

**Keuntungan REST**

* bahasa dan platform agnostic
* lebih sederhana/simpel untuk dikembangkan ketimbang SOAP
* mudah dipelajari, tidak bergantung pada tools
* ringkas, tidak membutuhkan layer pertukaran pesan (messaging) tambahan
* secara desain dan filosofi lebih dekat dengan web

**Kelemahan REST**

* Mengasumsi model point-to-point komunikasi - tidak dapat digunakan untuk lingkungan komputasi terdistribusi di mana pesan akan melalui satu atau lebih perantara
* Kurangnya dukungan standar untuk keamanan, kebijakan, keandalan pesan, dll, sehingga layanan yang mempunyai persyaratan lebih canggih lebih sulit untuk dikembangkan ("dipecahkan sendiri")
* Berkaitan dengan model transport HTTP

**Berikut metode HTTP yang umum digunakan dalam arsitektur berbasis REST.**

* GET, menyediakan hanya akses baca pada resource
* PUT, digunakan untuk menciptakan resource baru
* DELETE, digunakan untuk menghapus resource
* POST, digunakan untuk memperbarui resource yang ada atau membuat resource baru
* OPTIONS, digunakan untuk mendapatkan operasi yang disupport pada resource

**Lalu bagaimana cara kerja REST API?**

Pertama harus ada sebuah REST server yang akan menyediakan resource/data. Sebuah REST client akan membuat HTTP request ke server melalui sebuah global ID atau URIs dan server akan merespon dengan mengirimkan balik sebuah HTTP response sesuai yang diminta client.

**Komponen dari HTTP request:**

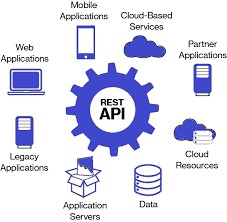
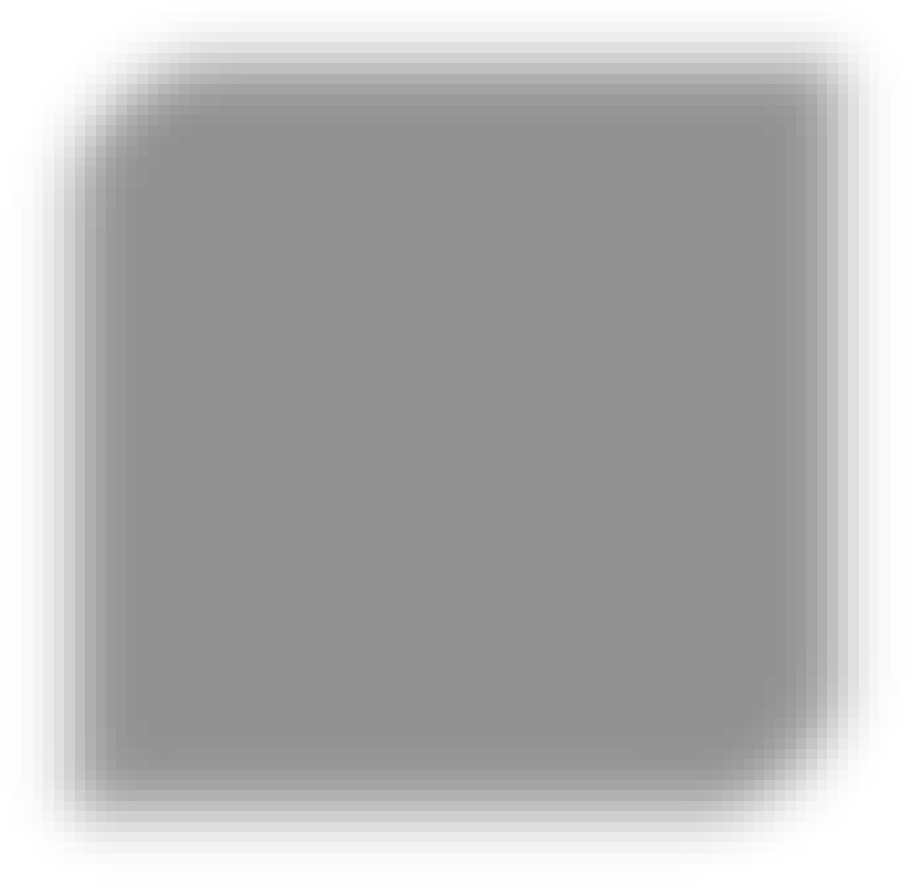
* HTTP method seperti GET, POST, PUT, DELETE dll sesuai dengan tugasnya masing-masing
* URI untuk mengetahui lokasi resource/data pada server
* HTTP Version, seperti HTTP v1.1
* Request Header, berisi metadata seperti Authorization, tipe client/browser, format yang didukung oleh client, format dari body pesan, seting cache dll.
* Request Body, konten dari data yang diberikan client ke server seperi URI params

**RESTfull API**

API dapat dikatakan **“RESTful”** jika memiliki fitur berikut :

* Client – server : client menangani front end dan server menangani back end dan keduanya dapat diganti secara independen satu sama lain.
* Stateless : Tidak ada data klien yang disimpan di server ketika ada permintaan dan status sesi disimpan di klien.
* Cacheable : Klien dapat men-cache respon (seperti browser yang men-cache elemen statis halaman web) untuk meningkatkan kinerja.

Keuntungan terbesar dari Restful API adalah Anda tidak perlu memasang apa pun di sisi klien. SDK atau framework tidak diperlukan. Yang harus Anda lakukan adalah membuat permintaan HTTP sederhana ke layanan end point dari API target, biarkan server melakukan layanannya untuk Anda dan dapatkan hasilnya kembali. Sangat mudah dilakukan.



Contoh paling sederhana adalah Google API untuk login. Anda tidak perlu menghabiskan banyak waktu hanya untuk membuat sistem login untuk member di website. Anda hanya perlu memanggil API dari google, pengguna hanya perlu login ke Google kemudian Anda akan mendapatkan data semisal alamat email / nama dari pengguna tersebut. Tentunya jika pengguna telah memberikan izin mengunakan data mereka

Jika kita jabarkan kembali, maka RESTful API memiliki 4 komponen penting di dalamnya diantaranya adalah:

* **URL Design**
* **HTTP Verbs**
* **HTTP Response Code**
* **Format Response**

**URL Design**

RESTful API diakses menggunakan protokol HTTP. Penamaan dan struktur URL yang konsisten akan menghasilkan API yang baik dan mudah untuk dimengerti developer. URL API biasa disebut endpoint dalam pemanggilannya. Contoh penamaan URL / endpoint yang baik adalah seperti berikut :

*/users*

*/users/1234*

*/users/1234/photos*

*/users/1234/photos/abc*

**HTTP Verbs**

Setiap request yang dilakukan terdapat metode yang dipakai agar server mengerti apa yang sedang di request client, diantaranya yang umum dipakai adalah :

**GET**

GET adalah metode HTTP Request yang paling simpel, metode ini digunakan untuk membaca atau mendapatkan data dari sumber.

**Contoh :**

*GET /users : Mengembalikan daftar user*

*GET /users/1234 : Mengembalikan data user dengan ID 1234*

**POST**

POST adalah metode HTTP Request yang digunakan untuk membuat data baru dengan menyisipkan data dalam body saat request dilakukan.

**Contoh :**

*POST /users : Membuat data user baru*

**PUT**

PUT adalah metode HTTP Request yang biasanya digunakan untuk melakukan update data resource.

**Contoh:**

*PUT /users/1234 : Mengupdate data user dengan ID 1234*

**DELETE**

DELETE adalah metode HTTP Request yang digunakan untuk menghapus suatu data pada resource.

**Contoh:**

*DELETE /users/1234 : Menghapus data user dengan ID 1234*

Selain HTTP Verbs diatas, masih ada metode HEAD dan PATCH dalam HTTP Request, tetapi jarang sekali digunakan.

**HTTP Response Code**

HTTP response code adalah kode standarisasi dalam menginformasikan hasil request kepada client. Secara umum terdapat 3 kelompok yang biasa kita jumpai pada RESTful API yaitu :

* **2XX** : adalah response code yang menampilkan bahwa request berhasil.
* **4XX** : adalah response code yang menampilkan bahwa request mengalami kesalahan pada sisi client.
* **5XX** : adalah response code yang menampilkan bahwa request mengalami kesalahan pada sisi server.

Berikut adalah response code yang biasa digunakan pada REST:

* **200 OK**

Response code ini menandakan bahwa request yang dilakukan berhasil.

* **201 Created**

Response code ini menandakan bahwa request yang dilakukan berhasil dan data telah dibuat. Kode ini digunakan untuk mengkonfirmasi berhasilnya request PUT atau POST.

* **400 Bad Request**

Response code ini menandakan bahwa request yang dibuat salah atau data yang dikirim tidak ada.

* **401 Unauthorized**

Response code ini menandakan bahwa request yang dibuat membutuhkan authentication sebelum mengakses resource.

* **404 Not Found**

Response Code ini menandakan bahwa resource yang di dipanggil tidak ditemukan.

* **405 Method Not Allowed**

Response code ini menandakan bahwa request endpoint ada tetapi metode HTTP yang digunakan tidak diizinkan.

* **409 Conflict**

Response code ini menandakan bahwa request yang dibuat terdapat duplikasi, biasanya informasi yang dikirim sudah ada sebelumnya.

* **500 Internal Server Error**

Response code ini menandakan bahwa request yang dilakukan terdapat kesalahan pada sisi server atau resource.

**Format Response**

* **Status/response code**, mengindikasikan status server terhadap resource yang direquest. misal : 404, artinya resource tidak ditemukan dan 200 response OK.

* **HTTP Version**, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.

* **Response Header,** berisi metadata untuk HTTP Response.

Contoh, type server, panjang content, tipe content, waktu response, dan lainnya.

* **Response Body**, konten dari data yang diberikan

Setiap request yang dilakukan client akan menerima data response dari server, response tersebut biasanya berupa data XML ataupun JSON. Setelah mendapatkan data response tersebut barulah client bisa menggunakannya dengan cara memparsing data tersebut dan diolah sesuai kebutuhan.

**Contoh:**

**XML**

*HTTP/1.1 200 OK*

*Date: Sat, 06 Oct 2001 23:20:04 GMT*

*Server: Apache.1.3.12 (Unix)*

*Connection: close*

*Content-Type: text/xml*

*Content-Length: 124*

*<?xml version=”1.0″?>*

*<methodResponse>*

*<params>*

*<param>*

*<value><double>18.24668429131</double></value>*

*</param>*

*</params>*

*</methodResponse>*

**JSON**

*GET /users/1234*

*HTTP/1.1 200 OK*

*Content-Type: application/vnd.api+json*

*{*

*“id”: “1234”,*

*“first\_name”: “jhon”,*

*“last\_name”: “doe”,*

*“created”: “2015-05-22T14:56:29.000Z”,*

*“updated”: “2015-05-22T14:56:29.000Z”*

*}*

**Postman, Software untuk pengujian Restfull API**

Saya ulas Kembali bahwa REST merupakan singkatan dari Representational State Transfer. Yang merupakan gaya arsitektur untuk merancang aplikasi yang saling terhubung. Dengan menggunakan HTTP sederhana untuk memungkinkan komunikasi antar mesin. Jadi, anda dapat menggunakan URL untuk memanipulasi beberapa informasi pengguna, REST mengirimkan permintaan HTTP seperti GET, POST, DELETE, dll ke URL untuk memanipulasi data.

Sebagai contoh, sebagai ganti membuat sebuah permintaan GET ke URL seperti:

*/deleteuser?id=10*

permintaannya akan menjadi seperti:

*DELETE /user/10*

**Untuk tahu lebih apa itu Rest API, silahkan and baca modul praktikum lain yang telah saya upload pada CIS.**

Postman merupakan salah satu alat paling populer untuk melakukan pengujian API.Postman muncul pada tahun 2012 sebagai proyek sampingan oleh Abhinav Asthana untuk menyederhanakan alur kerja API dalam pengujian dan pengembangan. API memungkinkan aplikasi perangkat lunak untuk berkomunikasi satu sama lain melalui panggilan API.

Dalam praktikum ini, Anda akan belajar:

* Mengapa menggunakan postman?
* [Cara](https://www.guru99.com/postman-tutorial.html#3) menggunakan postman
* [Bekerja](https://www.guru99.com/postman-tutorial.html#4) dengan GET Requests
* Bekerja dengan POST Requests
* Cara Parameterisasi Permintaan
* [Bagaimana](https://www.guru99.com/postman-tutorial.html#7) membuat testing dengan postman
* [Bagaimana](https://www.guru99.com/postman-tutorial.html#8) membuat collections

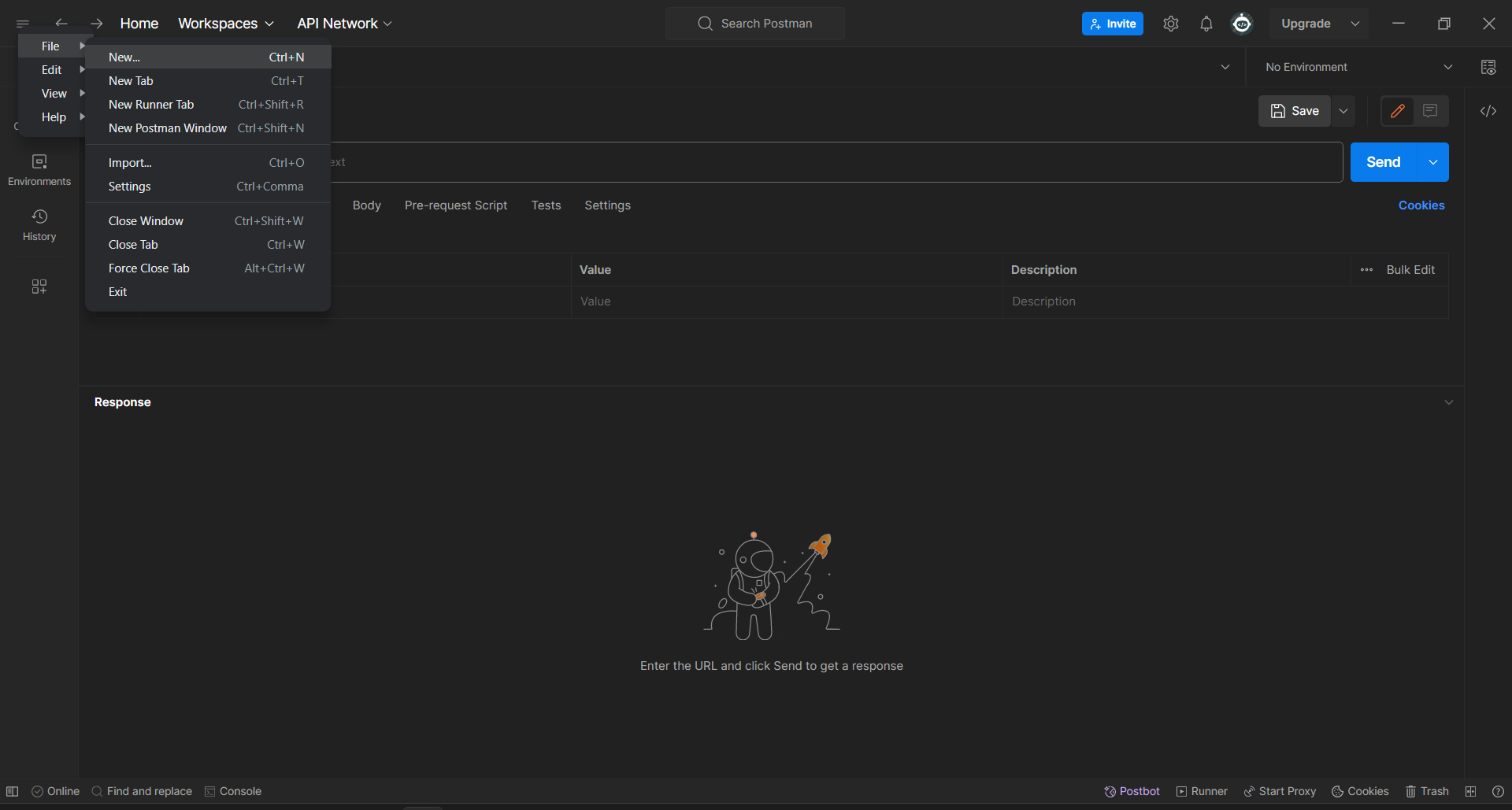
**Mengapa Menggunakan Postman?**

Dengan lebih dari 4 juta pengguna saat ini, Postman telah menjadi alat pilihan karena alasan berikut:

* **Aksesibilitas** - Untuk menggunakan Postman, seseorang hanya perlu masuk ke akun mereka sendiri sehingga memudahkan untuk mengakses file kapan saja, di mana saja selama aplikasi Postman dipasang di komputer.
* **Fitur collection** - Postman memungkinkan pengguna membuat collection untuk panggilan API mereka. Setiap collection dapat membuat subfolder dan beberapa permintaan. Ini membantu dalam mengatur suite tes Anda.
* **Kolaborasi** - Collection dan lingkungan dapat diimpor atau diekspor sehingga mudah untuk berbagi file. Tautan langsung juga dapat digunakan untuk berbagi collection.
* **Menciptakan environments** - Memiliki beberapa lingkungan bantu dalam pengulangan API tes yang lebih sedikit karena seseorang dapat menggunakan collection yang sama tetapi untuk lingkungan yang berbeda. Di sinilah parameterisasi akan terjadi yang akan kita bahas dalam pelajaran selanjutnya.
* **Pembuatan Tes** - Pos pemeriksaan seperti memverifikasi untuk status respons HTTP yang berhasil dapat ditambahkan ke setiap panggilan API yang membantu memastikan cakupan pengujian.
* **Pengujian Otomasi** - Melalui penggunaan Runner Collection atau Newman, pengujian dapat dijalankan dalam beberapa iterasi menghemat waktu untuk pengujian berulang.
* **Debugging** - Konsol Postman membantu untuk memeriksa data apa yang telah diambil sehingga memudahkan untuk men-debug tes.
* **Integrasi Berkelanjutan** - Dengan kemampuannya untuk mendukung integrasi berkelanjutan, praktik pengembangan dipertahankan.

[**Cara**](https://www.guru99.com/postman-tutorial.html#3) **menggunakan postman**

Di bawah ini adalah workspace Postman. Mari kita jelajahi berbagai fitur tools ini:

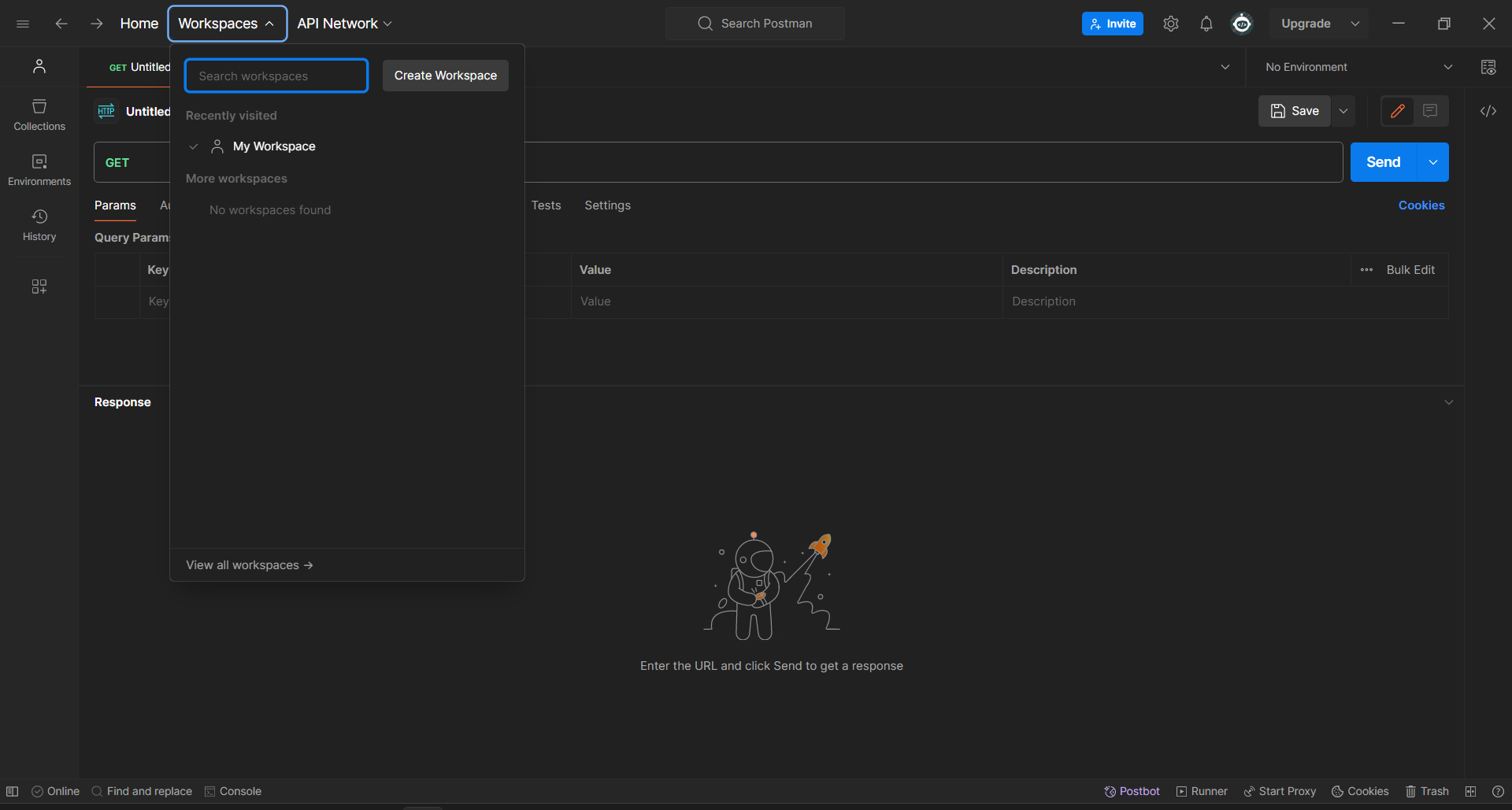


**2**

**3**

**4**

**1**

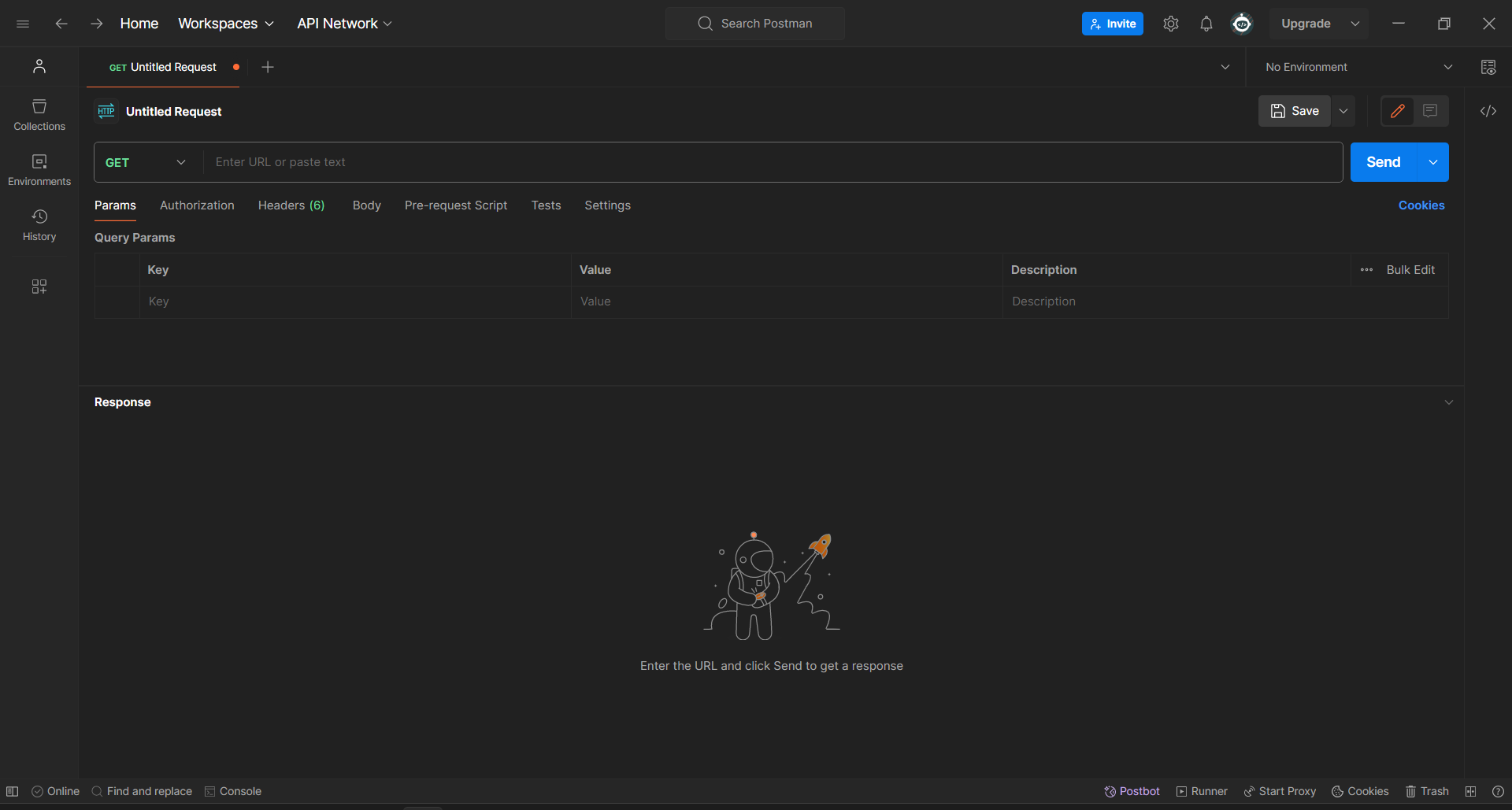


**7**

**8**

**6**

**5**



**18**

**17**

**16**

**15**

**13**

**14**

**12**

**11**

**10**

**9**

1. **New** - This is where you will create a new request, collection or environment.
2. **Import** - This is used to import a collection or environment. There are options such as import from file, folder, link or paste raw text.
3. **Runner** - Automation tests can be executed through the Collection Runner. This will be discussed further in the next lesson.
4. **Open New** - Open a new tab, Postman Window or Runner Window by clicking this button.
5. **My Workspace** - You can create a new workspace individually or as a team.
6. **Invite** - Collaborate on a workspace by inviting team members.
7. **History** - Past requests that you have sent will be displayed in History. This makes it easy to track actions that you have done.
8. **Collections** - Organize your test suite by creating collections. Each collection may have subfolders and multiple requests. A request or folder can also be duplicated as well.
9. **Request tab** - This displays the title of the request you are working on. By default, "Untitled Request" would be displayed for requests without titles.
10. **HTTP Request** - Clicking this would display a dropdown list of different requests such as GET, POST, COPY, DELETE, etc. In testing, the most commonly used requests are GET and POST.
11. **Request URL** - Also known as an endpoint, this is where you will identify the link to where the API will communicate with.
12. **Save** - If there are changes to a request, clicking save is a must so that new changes will not be lost or overwritten.
13. **Params** - This is where you will write parameters needed for a request such as key values.
14. **Authorization** - In order to access APIs, proper authorization is needed. It may be in the form of a username and password, bearer token, etc.
15. **Headers** - You can set headers such as content type JSON depending on the needs of the organization.
16. **Body** - This is where one can customize details in a request commonly used in POST request.
17. **Pre-request Script** - These are scripts that will be executed before the request. Usually, pre-request scripts for the setting environment are used to ensure that tests will be run in the correct environment.
18. **Tests** - These are scripts executed during the request. It is important to have tests as it sets up checkpoints to verify if response status is ok, retrieved data is as expected and other tests.

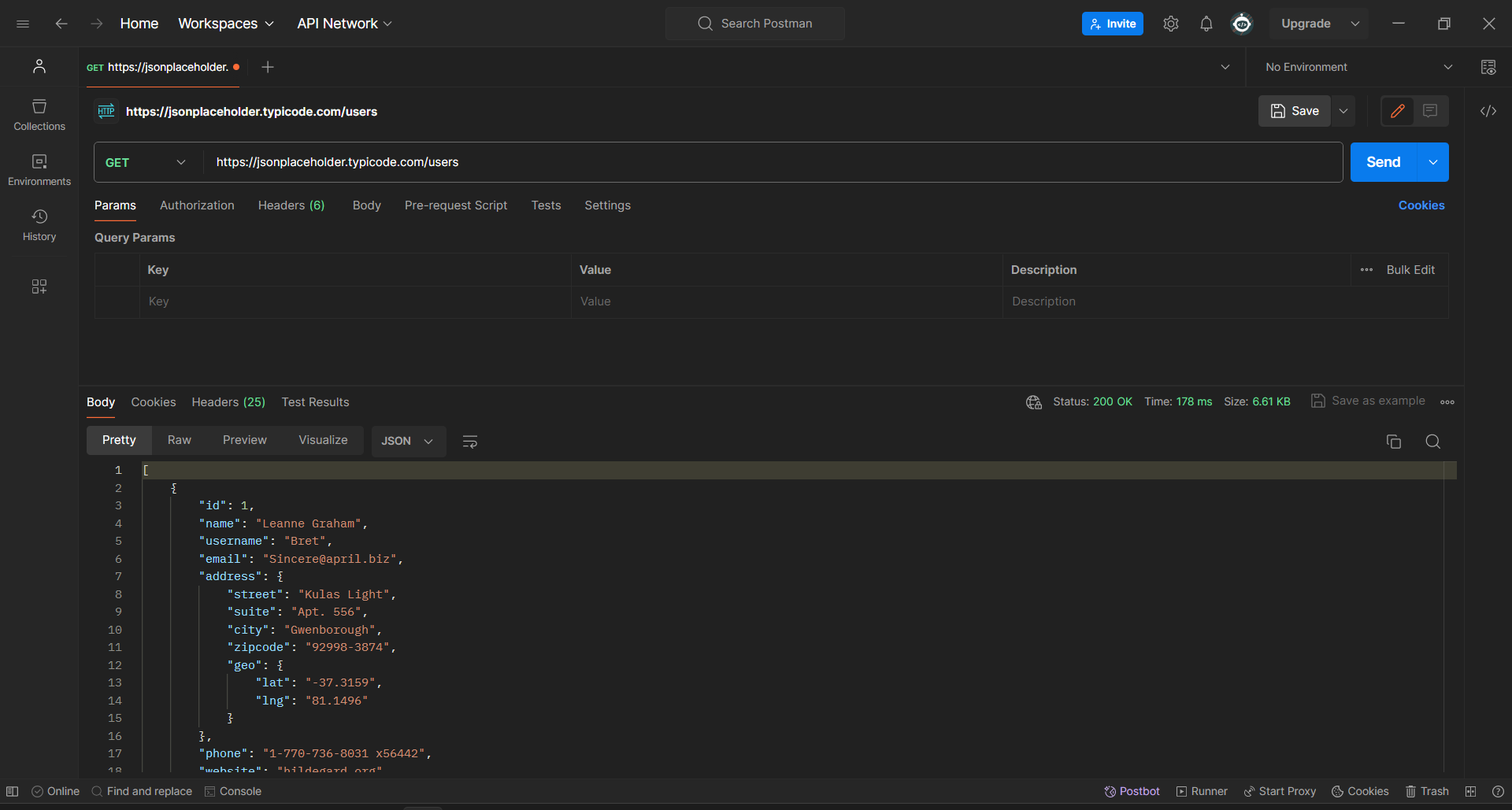
**Bekerja dengan GET Requests**

**GET Requests** digunakan untuk mengambil informasi dari URL yang diberikan. Tidak akan ada perubahan yang dilakukan ke **endpoint**.

Mahasiswa akan menggunakan URL berikut untuk semua contoh dalam praktikum ini:

*https://jsonplaceholder.typicode.com/users*

Jawab:



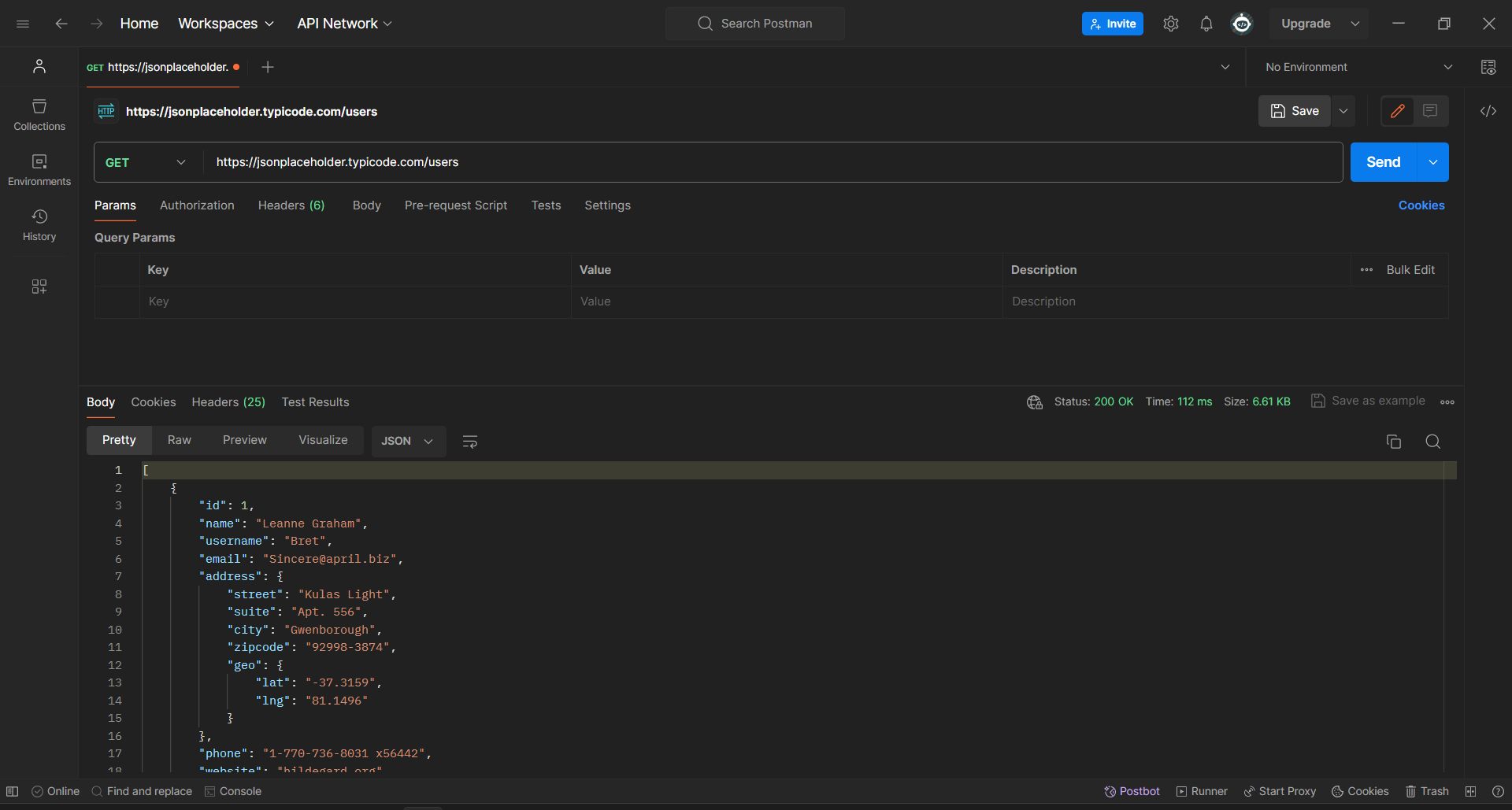
**JSONPlaceholder adalah:**

Fake Online REST API untuk Testing dan Prototyping yang melayani ~350M requests per bulan. Kita menggunakan resource ini karena pada praktikum ini kita fokus bagaimana mengakses/menguji Rest API yang gratis(opensource) dari client menggunakan tools Postman. Anda dapat melihat kumpulan Resource dan route Rest API pada link berikut:

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

Langkah-langkah pada Postman:

* set permintaan HTTP Anda ke **GET**.
* Pada request URL Field, masukkan link berikut
* Klik **Send/Kirim**
* Anda akan melihat **200**/Pesan OK

Seharusnya ada 10 hasil pengguna pada body yang menunjukkan bahwa tes Anda telah berhasil dijalankan. 

**5**

**4**

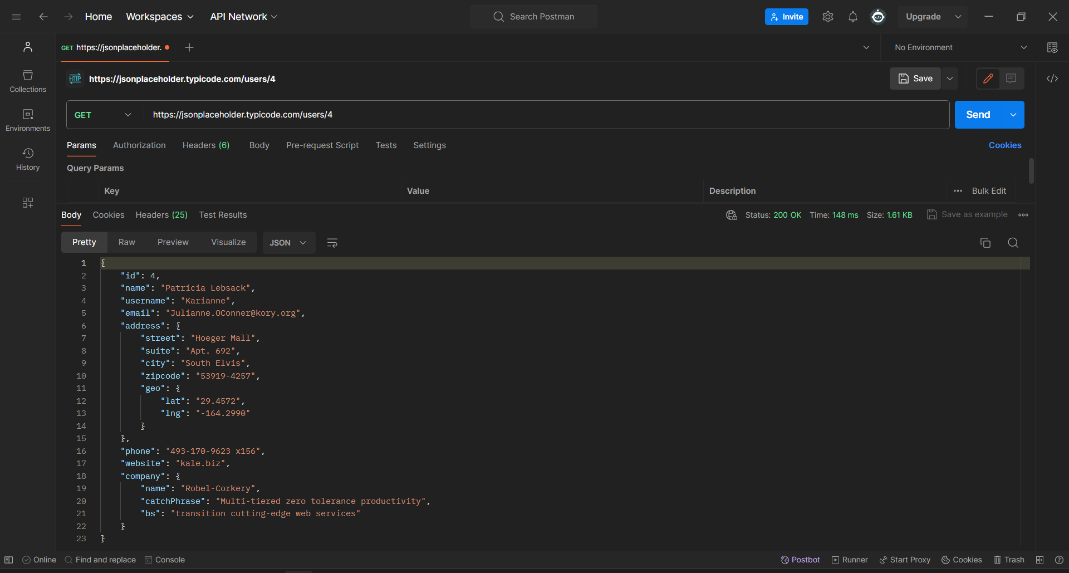
**3**

**2**

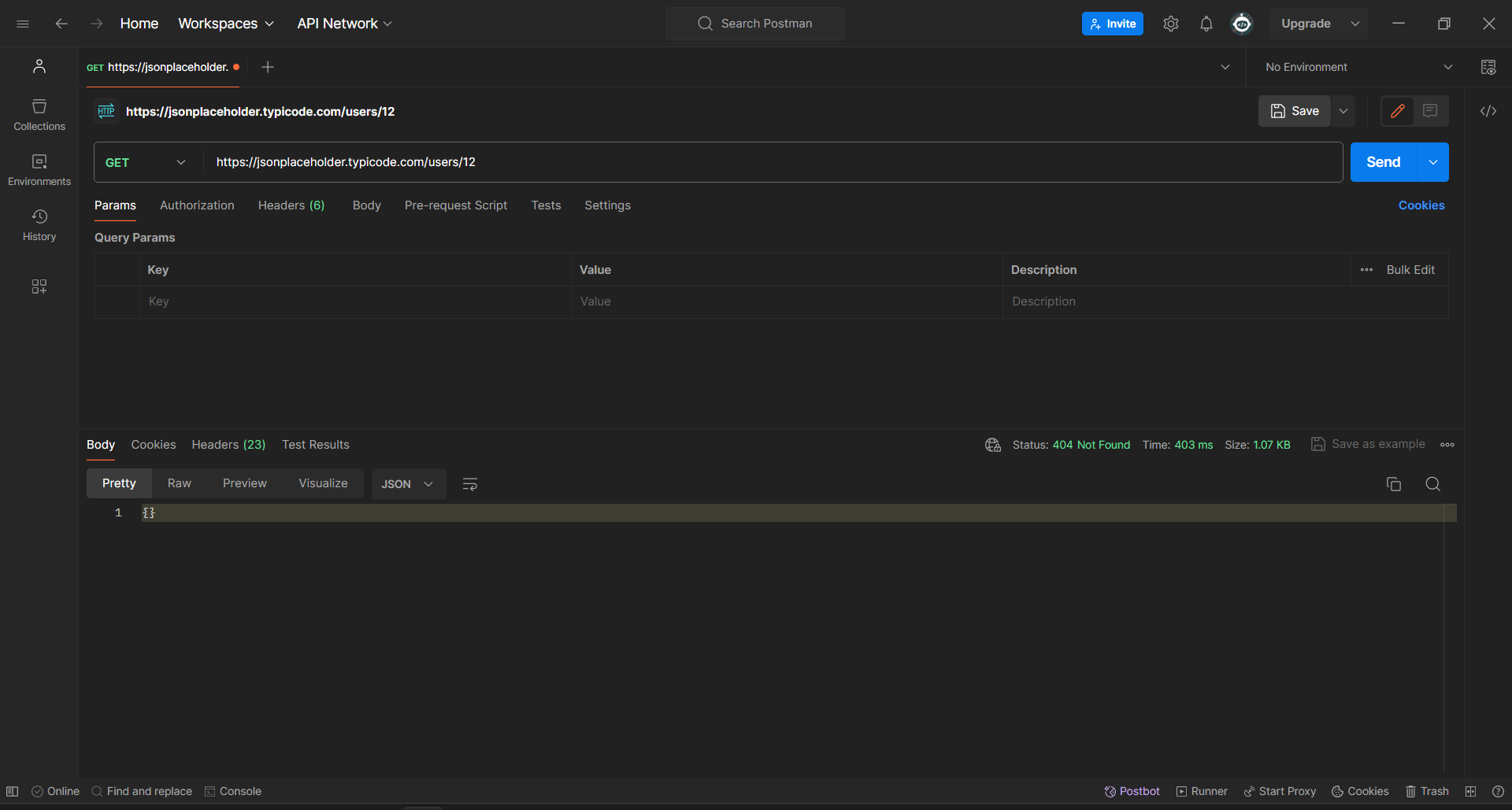
**1**

\***Catatan**: Mungkin ada beberapa kasus yang GET Request nya mungkin tidak berhasil. Itu bisa karena request URL yang tidak valid atau diperlukan otentikasi. Bagaimana jika anda ingin mengambil data **users** dengan ID: 4?

Anda hanya perlu mengganti URL field menjadi seperti berikut:

https://jsonplaceholder.typicode.com/users/4

Bagaimana jika anda ingin mengambil data **users** dengan **ID: 12**? Apa yang akan terjadi?



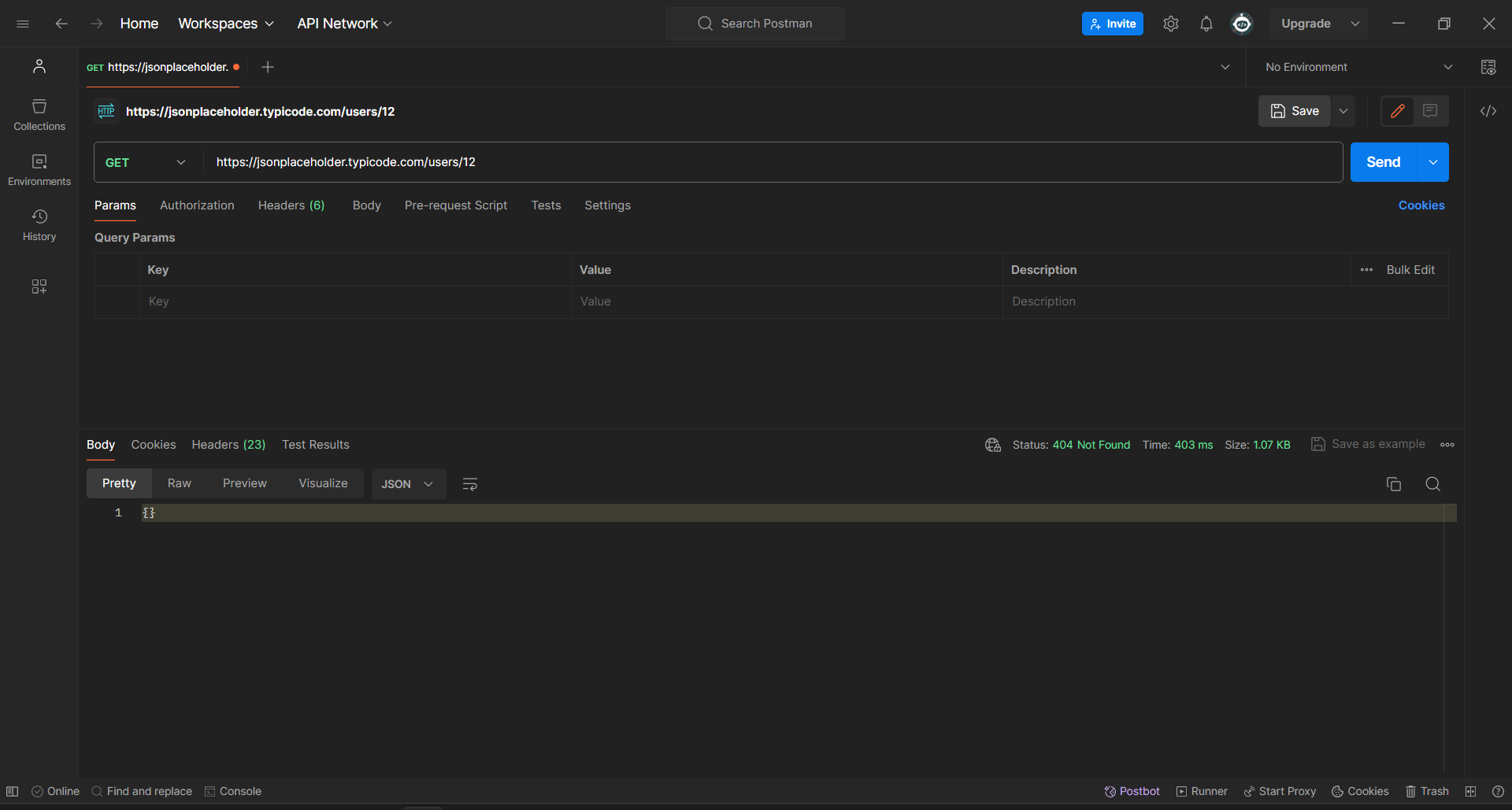
**Result:**

Pada workspace akan tertampil status code **404: Not Found**, artinya data yang kita **GET** tidak ada/tidak ditemukan. Ini dikarenakan **jsonplaceholder** endpoint hanya memiliki data users sejumlah 10, sementara data users yang kita request adalah data users pada **ID ke 12**.

**Bekerja dengan POST Requests**

Post Request berbeda dari GET Request karena pada POST Request ada manipulasi data dengan pengguna/anda menambahkan data ke endpoint. Menggunakan data yang sama dari langkah sebelumnya di GET Request, mari sekarang tambahkan pengguna dengan data anda sendiri.

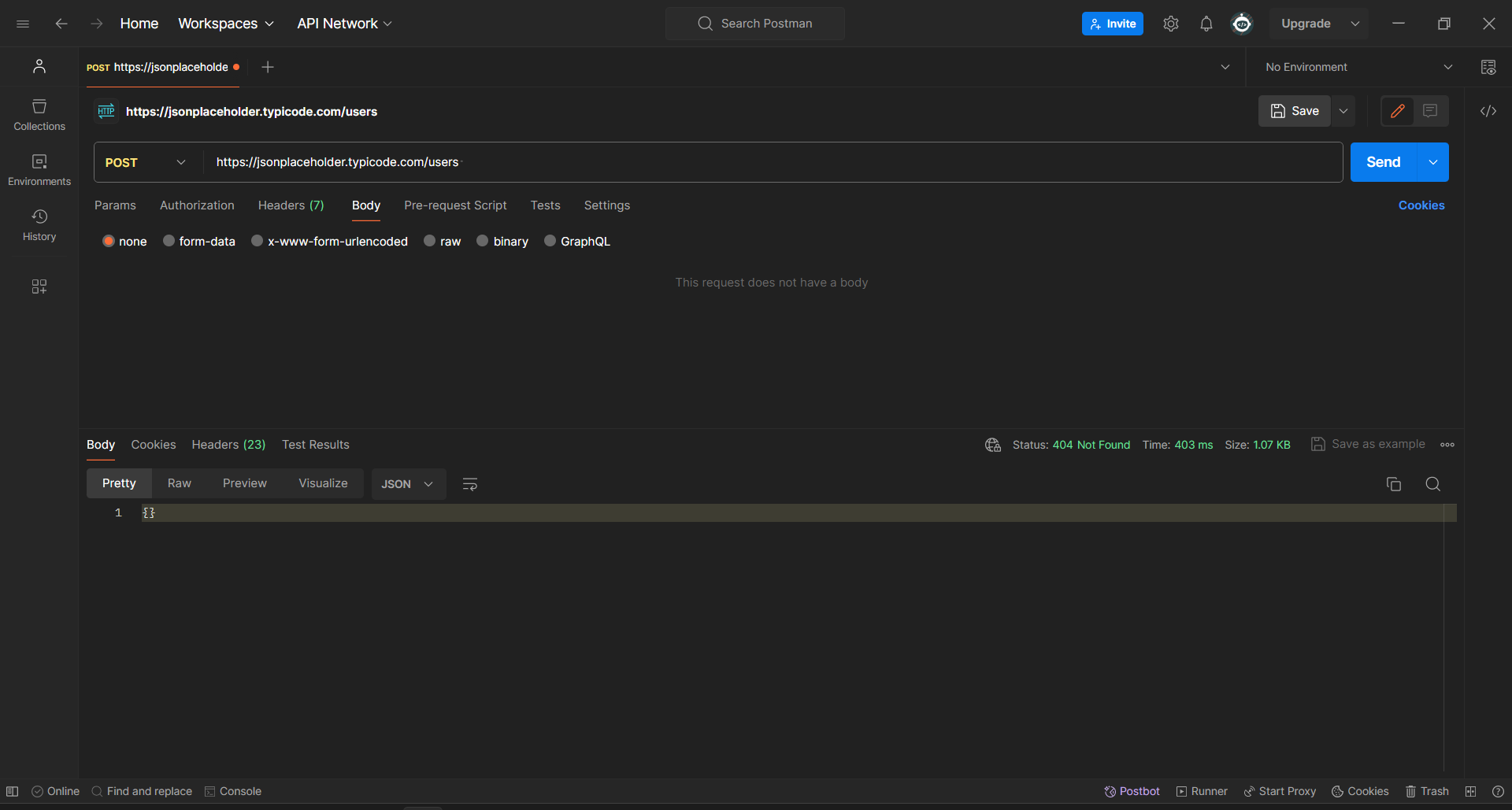
**Langkah 1)** Klik tab baru untuk membuat **new Request**.



**Langkah 2)** Di tab baru

* Setel HTTP Request anda ke POST.
* Masukkan tautan yang sama di url permintaan:

*https://jsonplaceholder.typicode.com/users*

* beralih ke tab body. 

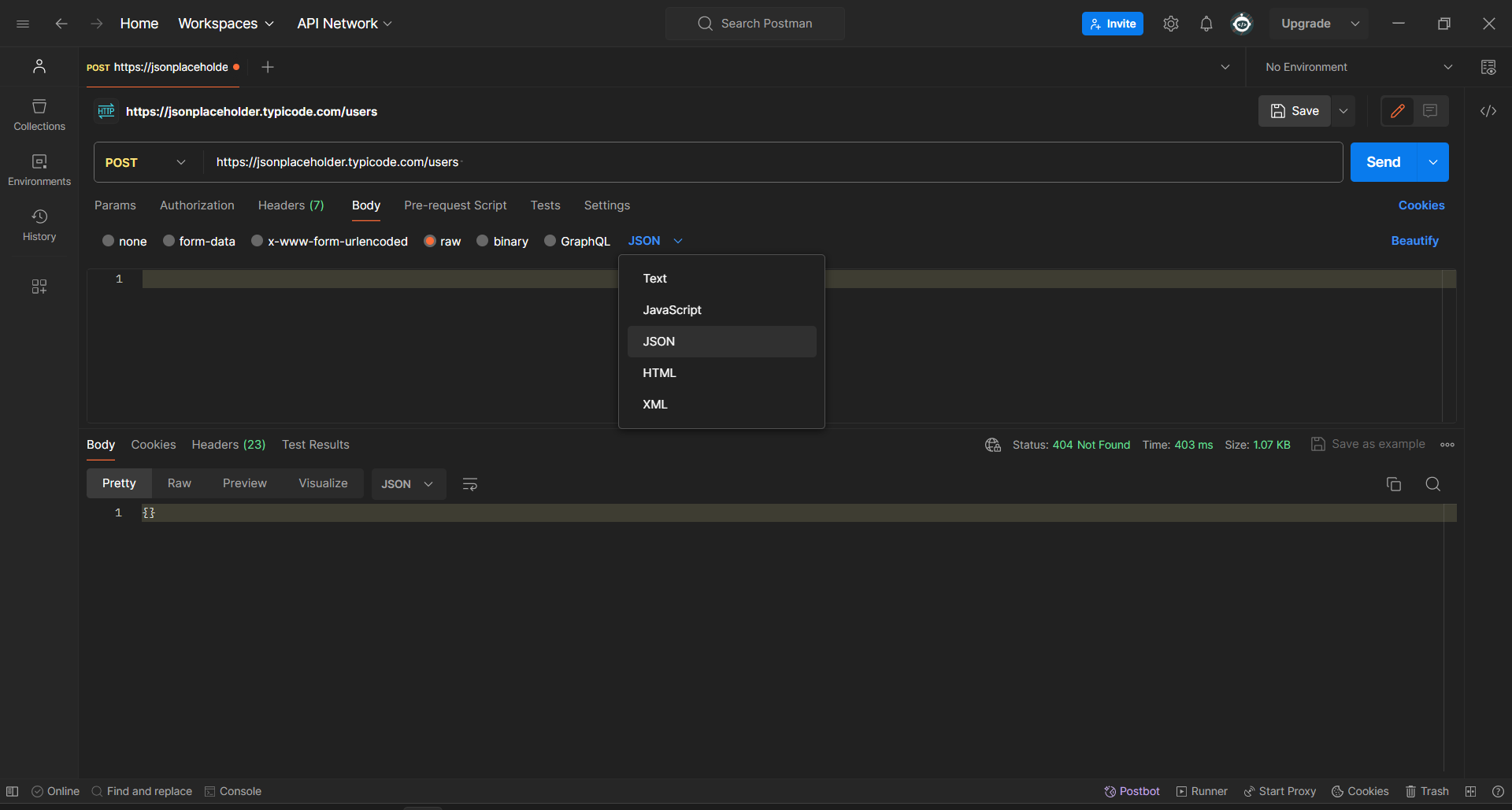
**3**

**2**

**1**

Langkah 3) Di Body,

* Klik Raw
* Pilih JSON



**2**

**1**

**Langkah 4)** Copy dan paste hanya satu hasil pengguna dari permintaan sebelumnya seperti di bawah ini.

* Pastikan kode telah disalin dengan benar dengan kurung kurawal dan kurung.
* Ubah id ke 11 dan beri nama ke nama yang diinginkan. Anda juga dapat mengubah detail lainnya seperti alamat.

{

"id": 11,

"name": "Jansutris Purba si Babang Tamvan",

"username": "jansutris12",

"email": "apritenjansut@gmail.com",

"address": {

"street": "jl.Sisingamangaraja, Laguboti",

"suite": "Rusun 3 IT Del",

"city": "Toba",

"zipcode": "123456",

"geo": {

"lat": "-71.4197",

"lng": "71.7478"

}

},

"phone": "0821-2345-6789",

"website": "jansutrisAja.com",

"company": {

"name": "Pro Evolution Soccer",

"catchPhrase": "Synchronised bottom-line interface",

"bs": "rajin ngoding pangkal kaya"

}

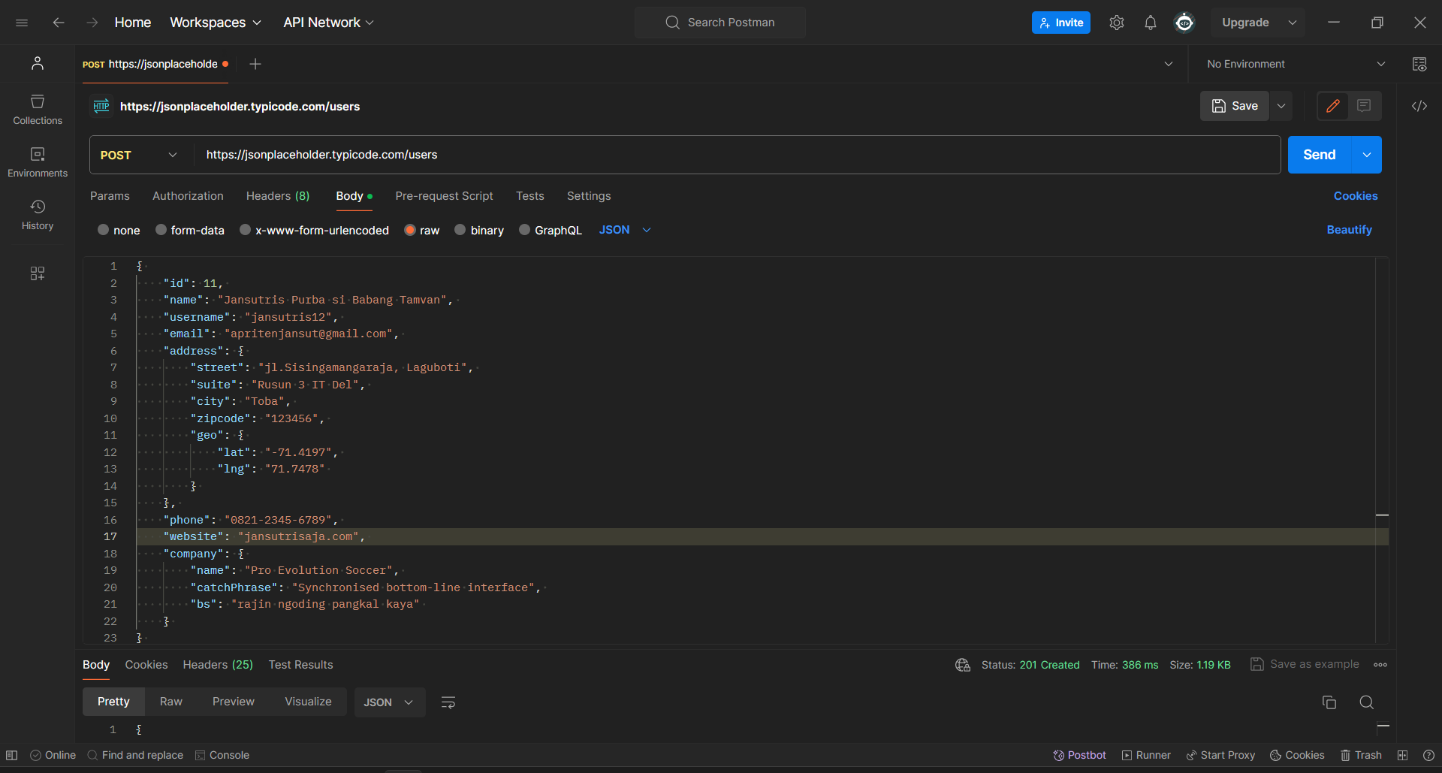
}



**Catatan:**

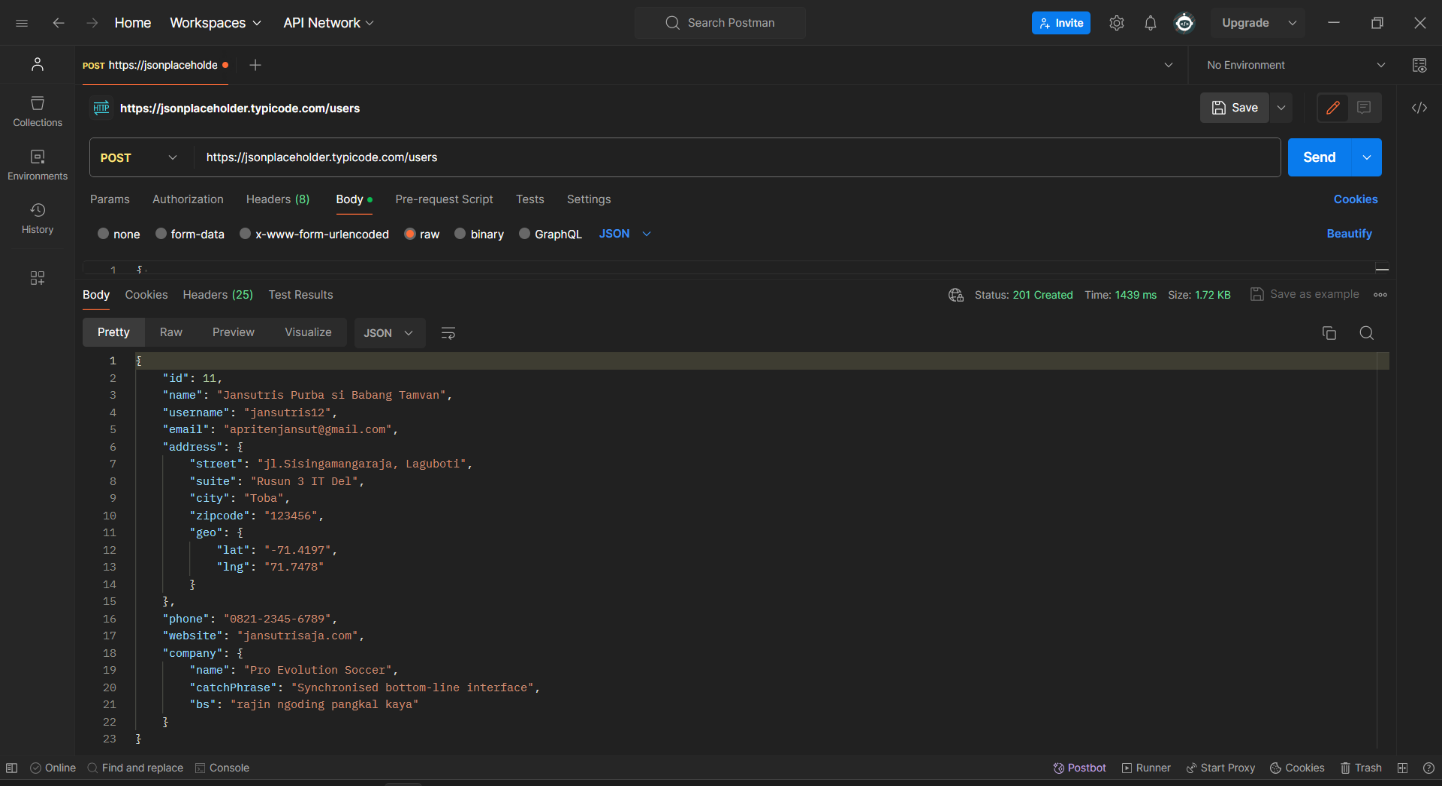
**Request POST** harus memiliki format yang benar untuk memastikan bahwa data yang diminta akan dibuat. Ini adalah praktik yang baik untuk menggunakan Dapatkan dulu untuk memeriksa format permintaan JSON. Anda dapat menggunakan tools online seperti:

<https://jsonformatter.curiousconcept.com/>



Langkah 5) Next,

* Klik Send/Kirim.
* Status: 201 Dibuat harus ditampilkan
* Data yang diposting muncul di Body.



**Cara Parameterisasi Permintaan**

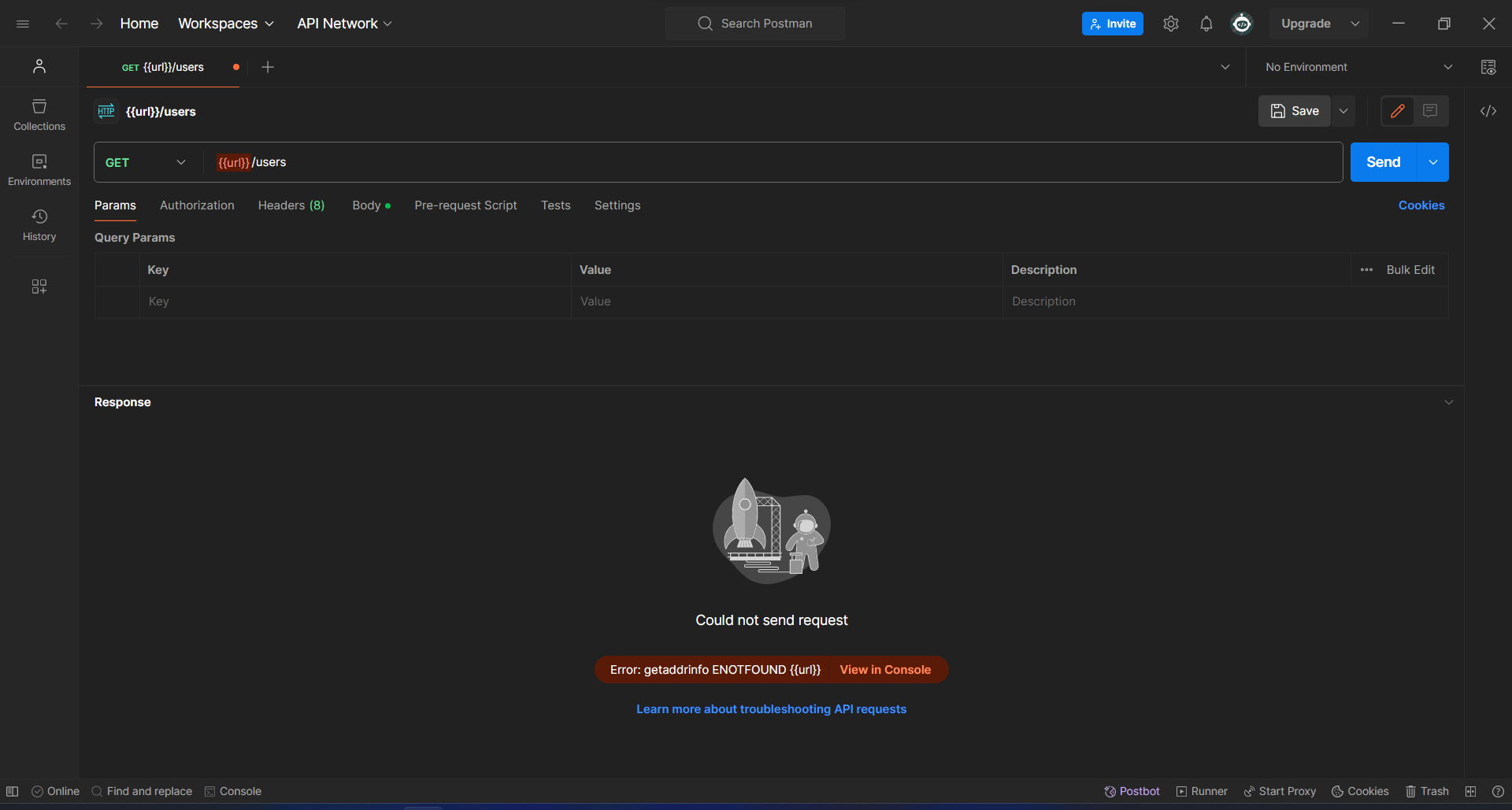
Parameterisasi Data adalah salah satu fitur paling berguna dari Postman . Alih-alih membuat permintaan yang sama dengan data yang berbeda, Anda bisa menggunakan variabel dengan parameter. Data ini dapat dari file data atau variabel lingkungan. Parameterisasi membantu untuk menghindari pengulangan tes yang sama dan iterasi dapat digunakan untuk pengujian otomasi.

Parameter dibuat melalui penggunaan *double curly brackets*/kurung keriting ganda:

**{{sample}}.** Mari kita lihat contoh penggunaan parameter dalam Request sebelumnya:

Langkah 1)

* Setel permintaan HTTP Anda ke GET
* Masukkan tautan ini: [https://jsonplaceholder.typicode.com/users.](https://jsonplaceholder.typicode.com/users) Ganti bagian pertama tautan dengan parameter seperti {{url}}. Url permintaan sekarang harus {{url}} / users.
* Klik Send/kirim.
* Pasti tidak ada respon karena kita belum menetapkan sumber parameter kita.



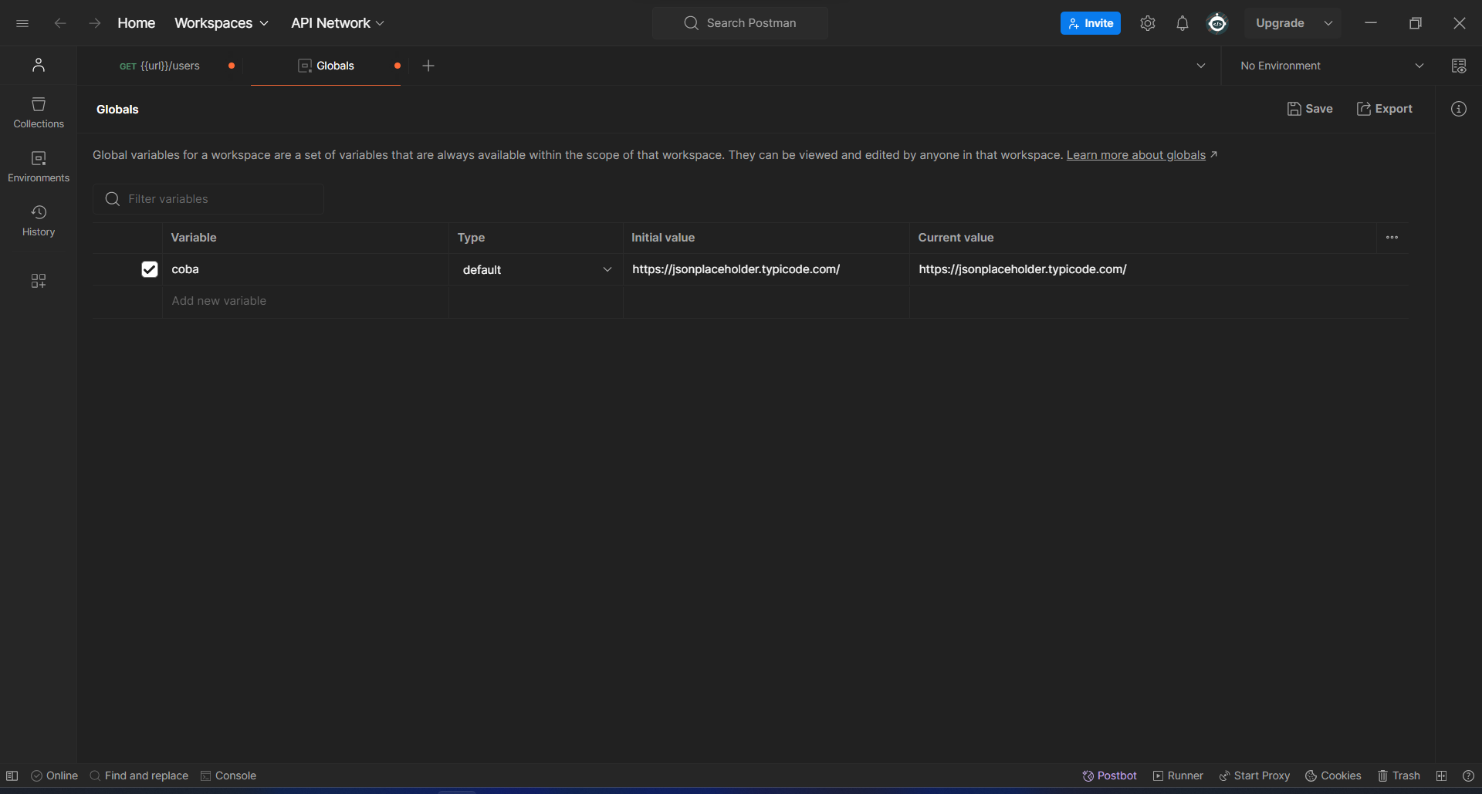
**3**

**1**

**2**

Langkah 2) Untuk menggunakan parameter yang Anda butuhkan untuk mengatur lingkungan

* Klik **ikon mata**
* Klik **edit** untuk mengatur variabel ke lingkungan global yang dapat digunakan di semua collection/collection.
* Isi sesuai dengan gambar pada Langkah 2 berikut
* Klik **Persist All**

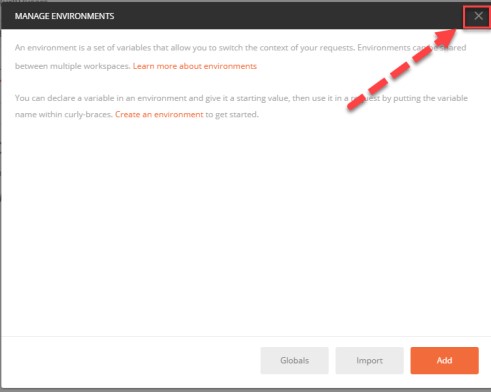


**3**

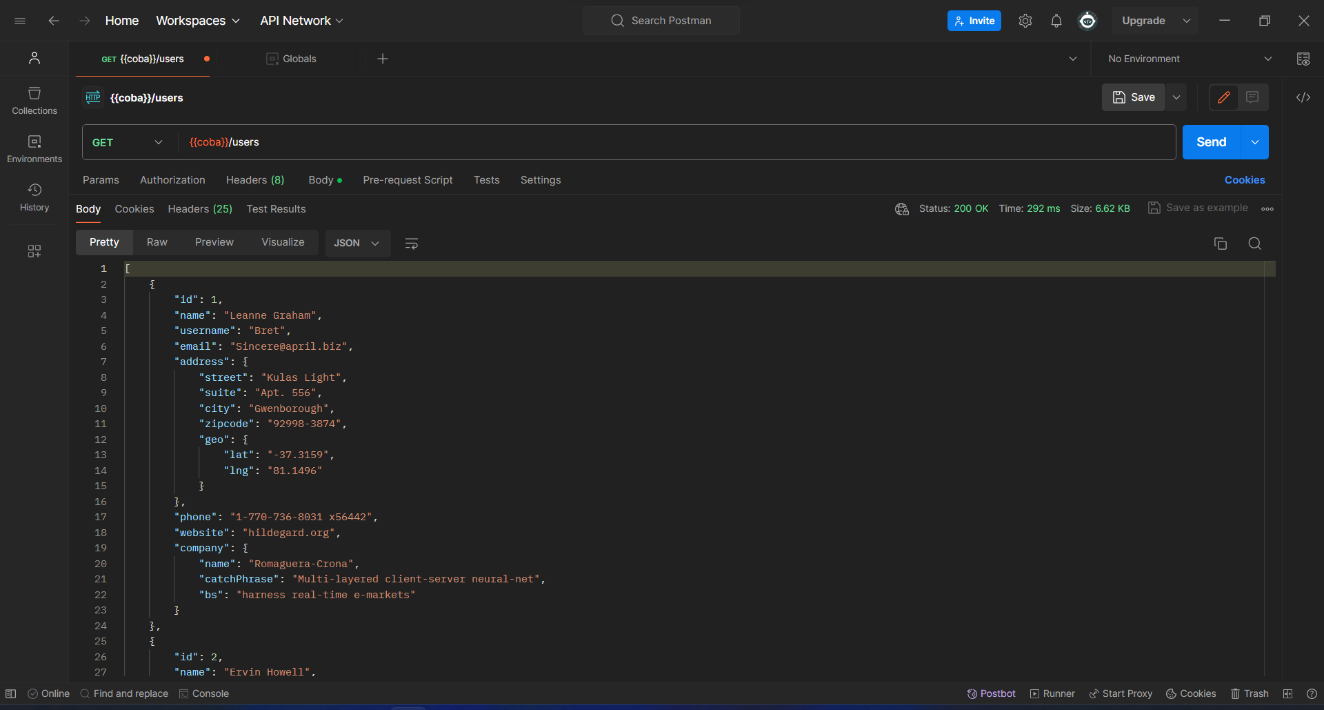
**2**

**1**

Langkah 4) Klik tutup jika Anda melihat layar berikutnya.



Langkah 5) Kembali ke Request GET, lalu klik **Send/kirim.** Seharusnya sekarang ada hasil untuk permintaan Anda.



**Catatan:**

Selalu pastikan bahwa parameter Anda memiliki sumber seperti environment variable atau file data untuk menghindari kesalahan.

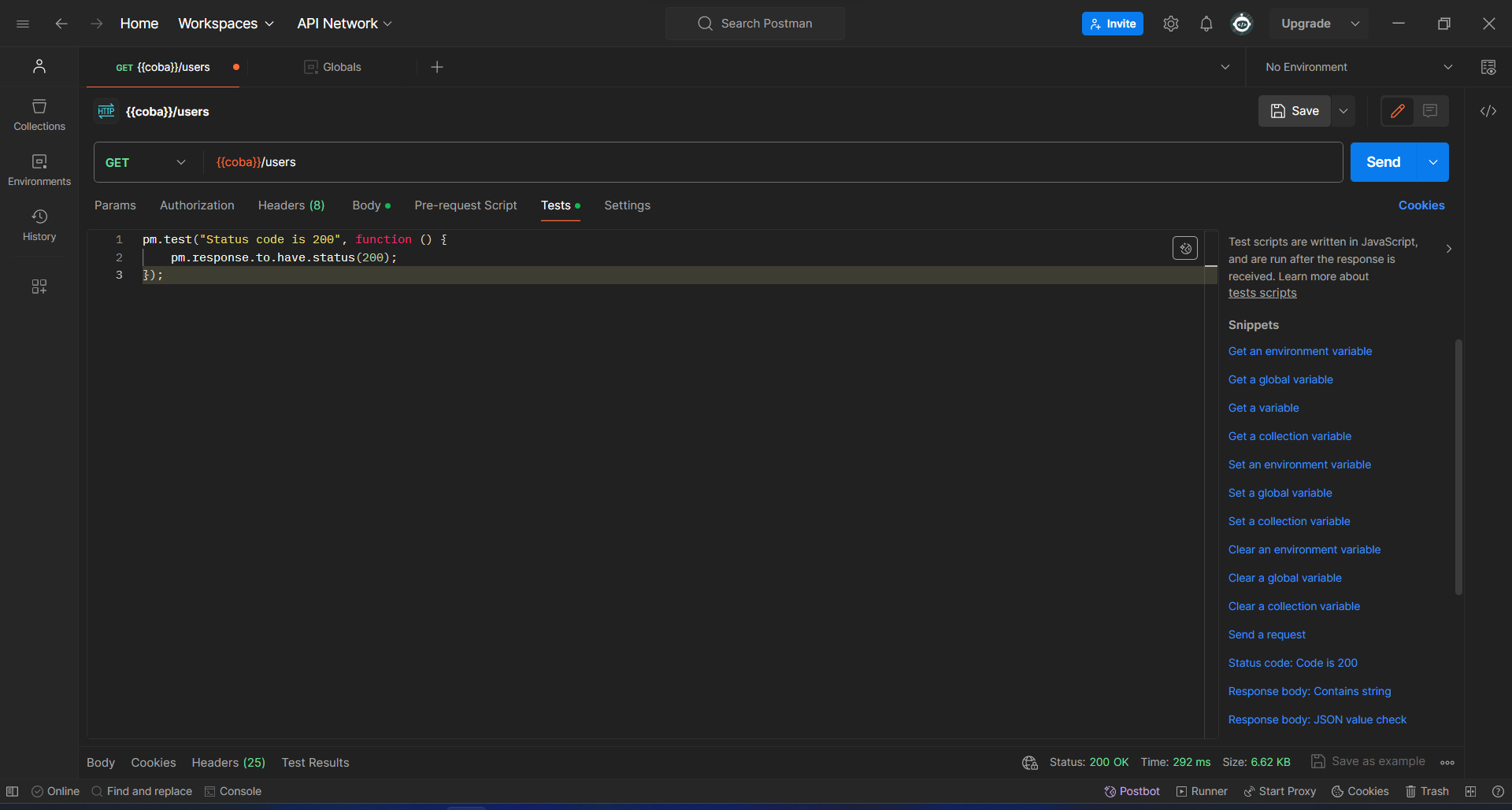
# Cara Membuat Tes Postman

Postman Tests adalah kode JavaScript yang ditambahkan ke permintaan yang membantu Anda memverifikasi hasil seperti status berhasil atau gagal, perbandingan hasil yang diharapkan, dll. Biasanya dimulai dengan **pm.test**. Itu dapat dibandingkan dengan menegaskan, memverifikasi perintah yang tersedia di alat lain.

Mari kita buat beberapa tes dasar untuk permintaan parameterisasi dari pelajaran sebelumnya.

Langkah 1) Buka permintaan pengguna GET Anda dari tutorial sebelumnya.

* Beralih ke tab tes. Di sebelah kanan adalah kode snippet.
* Dari bagian cuplikan, klik "Kode status: Kode adalah 200".
* Panel diisi secara otomatis.

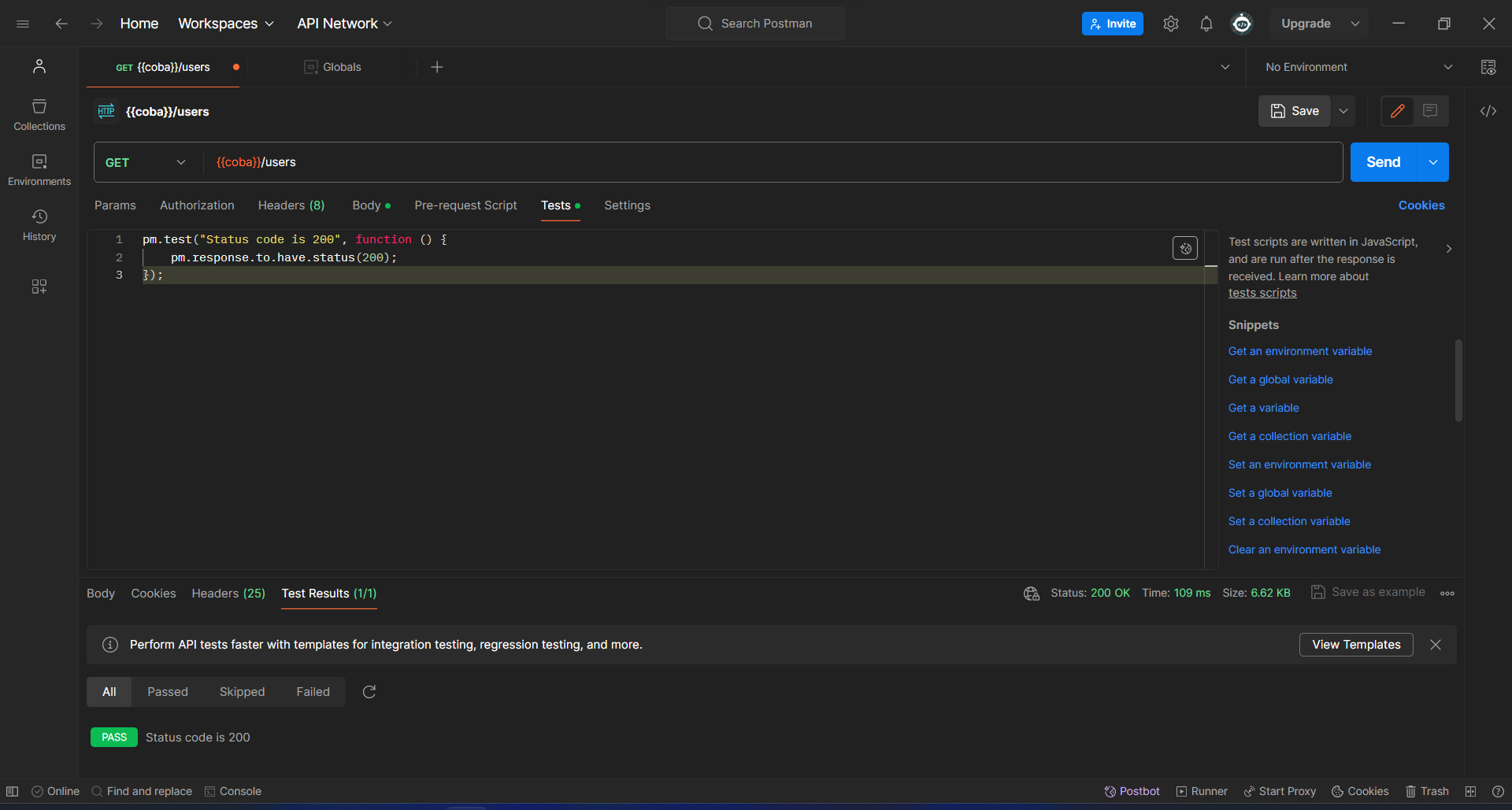


**3**

**2**

**1**

Langkah 2) Sekarang klik **Send/Kirim.** Hasil tes sekarang harus ditampilkan.

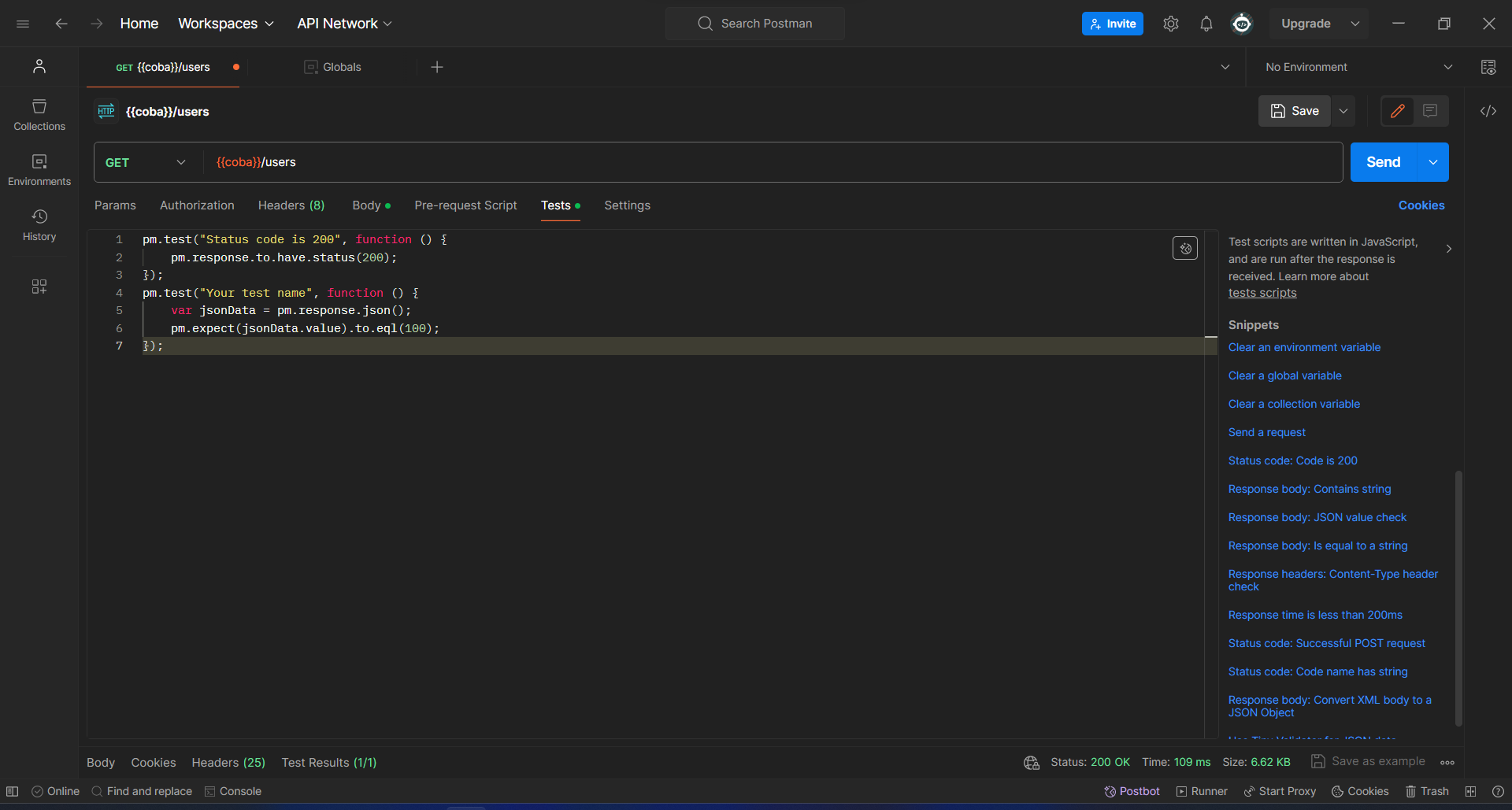


**2**

**1**

**Langkah 3)** Kembali ke tab tes dan mari kita tambahkan tes lain.

* Kali ini kita akan membandingkan hasil yang diharapkan dengan hasil yang sebenarnya.
* Dari bagian cuplikan, klik **"Response body:JSON value check"**. Kami akan memeriksa apakah **Leanne Graham** memiliki **userid 1**.



**2**

**1**

# Langkah 4)

Ganti **"Your Test Name"** dari kode dengan **"Check if user with id1 is Leanne Graham"** sehingga nama tes menentukan dengan tepat apa yang ingin kami uji.

Ganti jsonData.value dengan jsonData [0] .name. Untuk mendapatkan jalur, periksa badan di dapatkan hasil lebih awal. Karena Leanne Graham adalah userid 1, jsonData berada di hasil pertama yang harus dimulai dengan 0. Jika Anda ingin mendapatkan hasil kedua, gunakan jsonData [1] dan seterusnya untuk hasil selanjutnya.

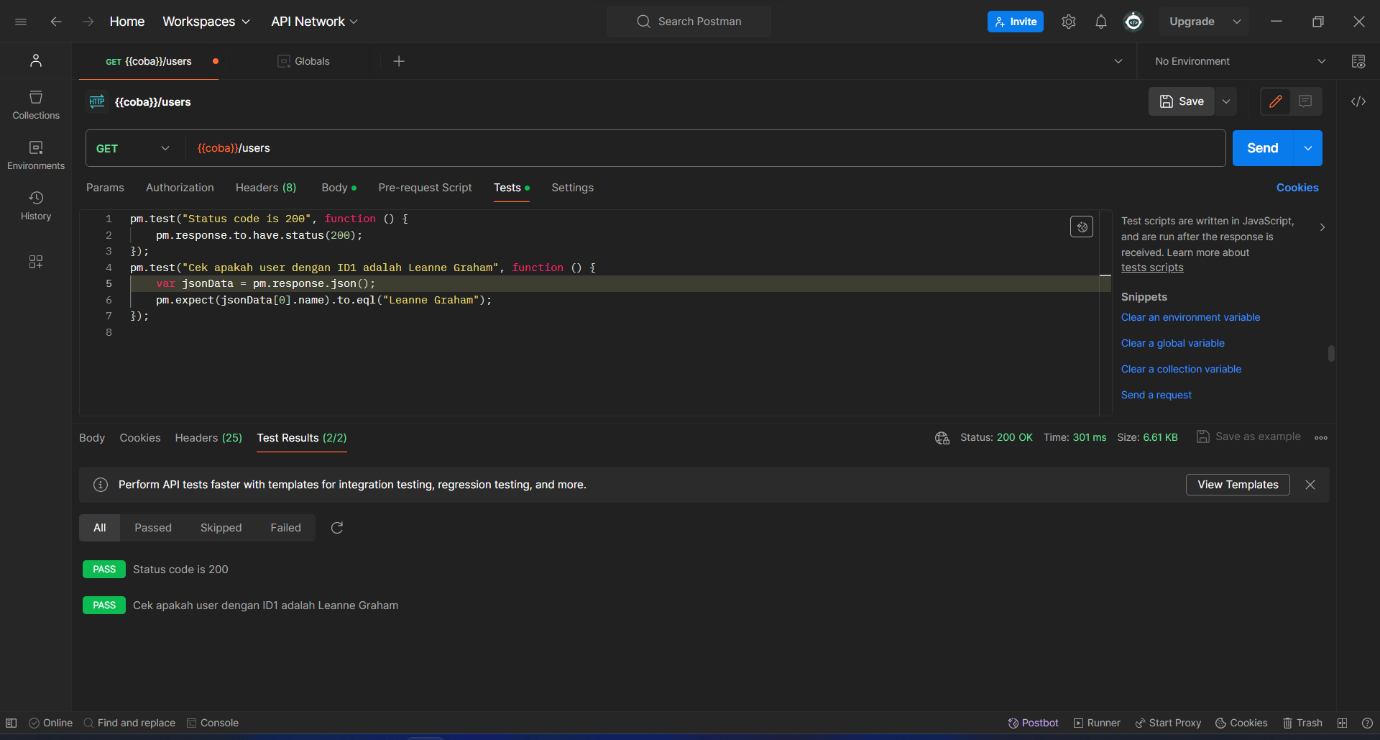
Dalam to eql, masukkan **"Leanne Graham"**

pm.test("Cek apakah user dengan ID1 adalah Leanne Graham", function () { var jsonData = pm.response.json();

pm.expect(jsonData[0].name).to.eql("Leanne Graham");

});

**Langkah 5)** Klik Send/kirim. Sekarang harus ada dua hasil tes yang lulus untuk permintaan Anda.



**Catatan:**

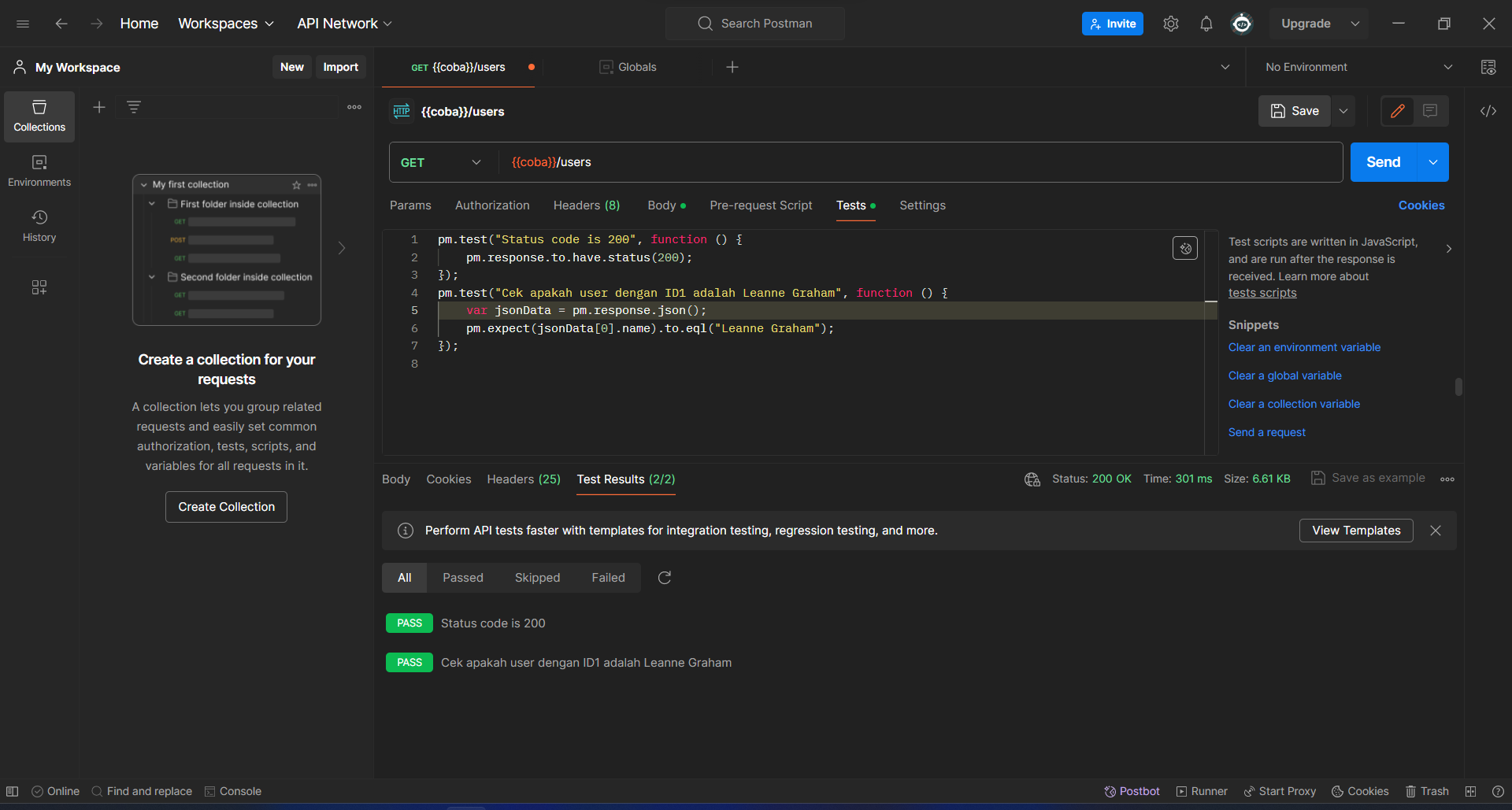
Ada beberapa jenis tes yang dapat dibuat di Postman . Coba jelajahi alat ini dan lihat tes apa yang sesuai dengan kebutuhan Anda.

# Cara Membuat Collection

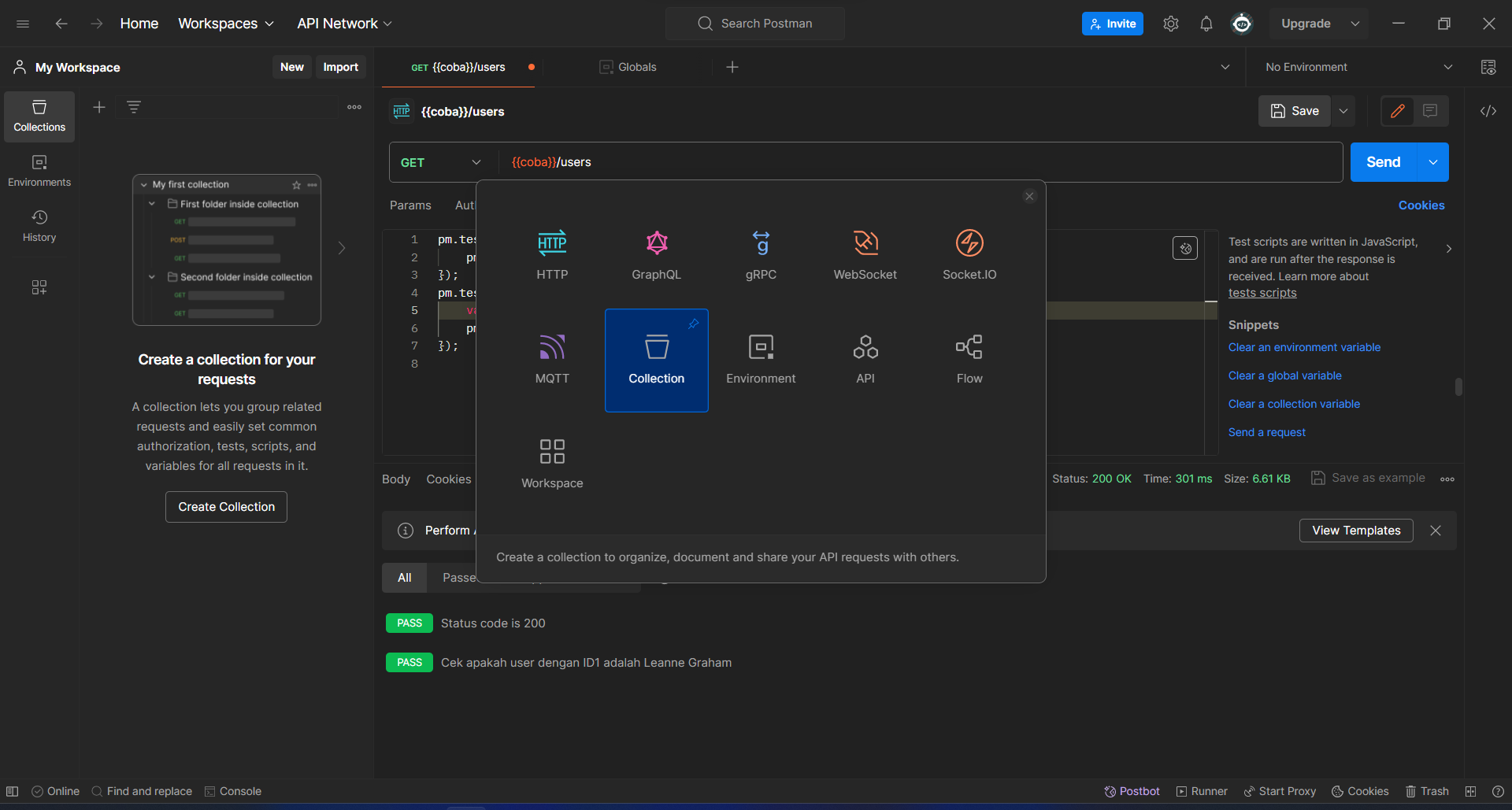
Collection memainkan peran penting dalam mengatur suite tes. Dapat diimpor dan diekspor sehingga mudah untuk berbagi collection di antara tim. Dalam praktikum ini, kita akan belajar cara membuat dan menjalankan collection.

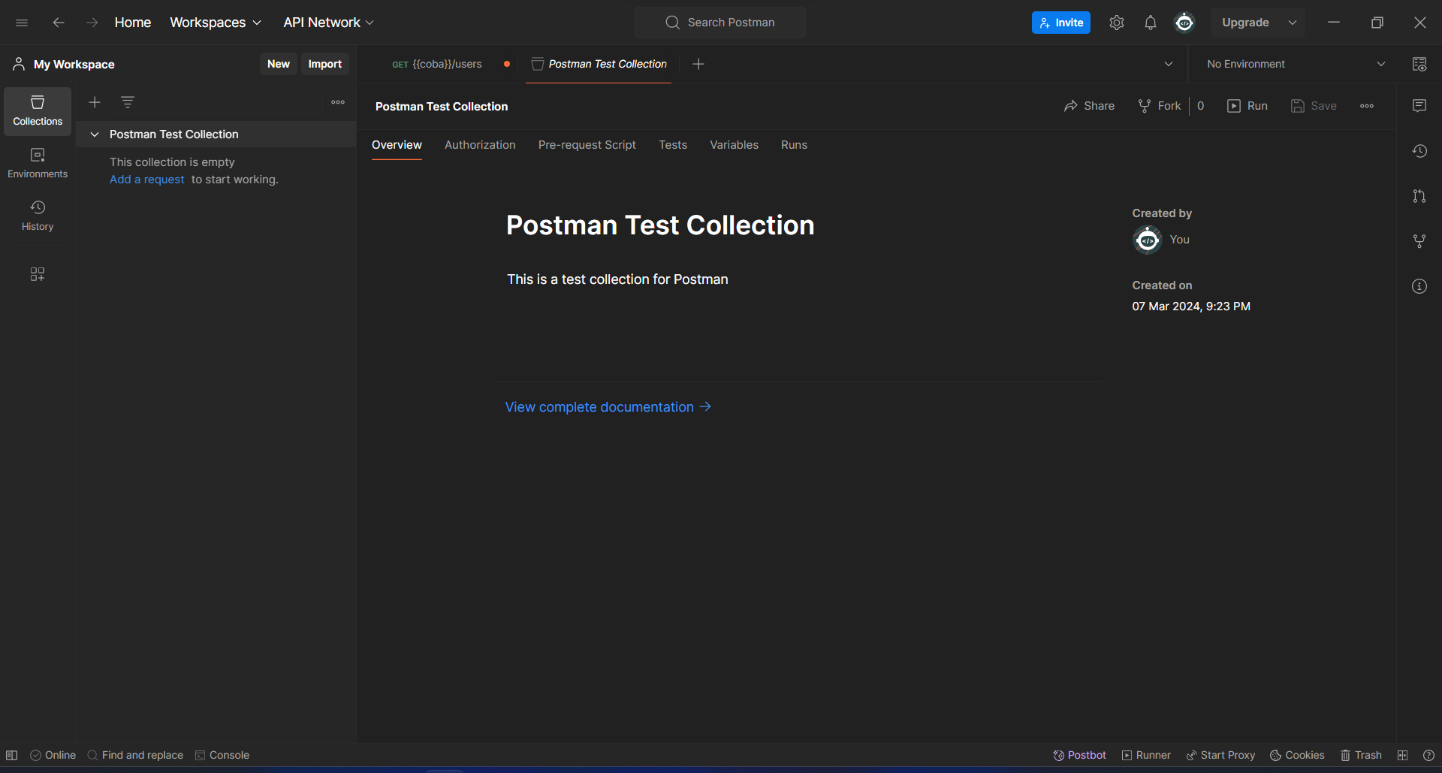
Mari mulai membuat collection:

**Langkah 1)** Klik tombol Baru di sudut kiri atas halaman.

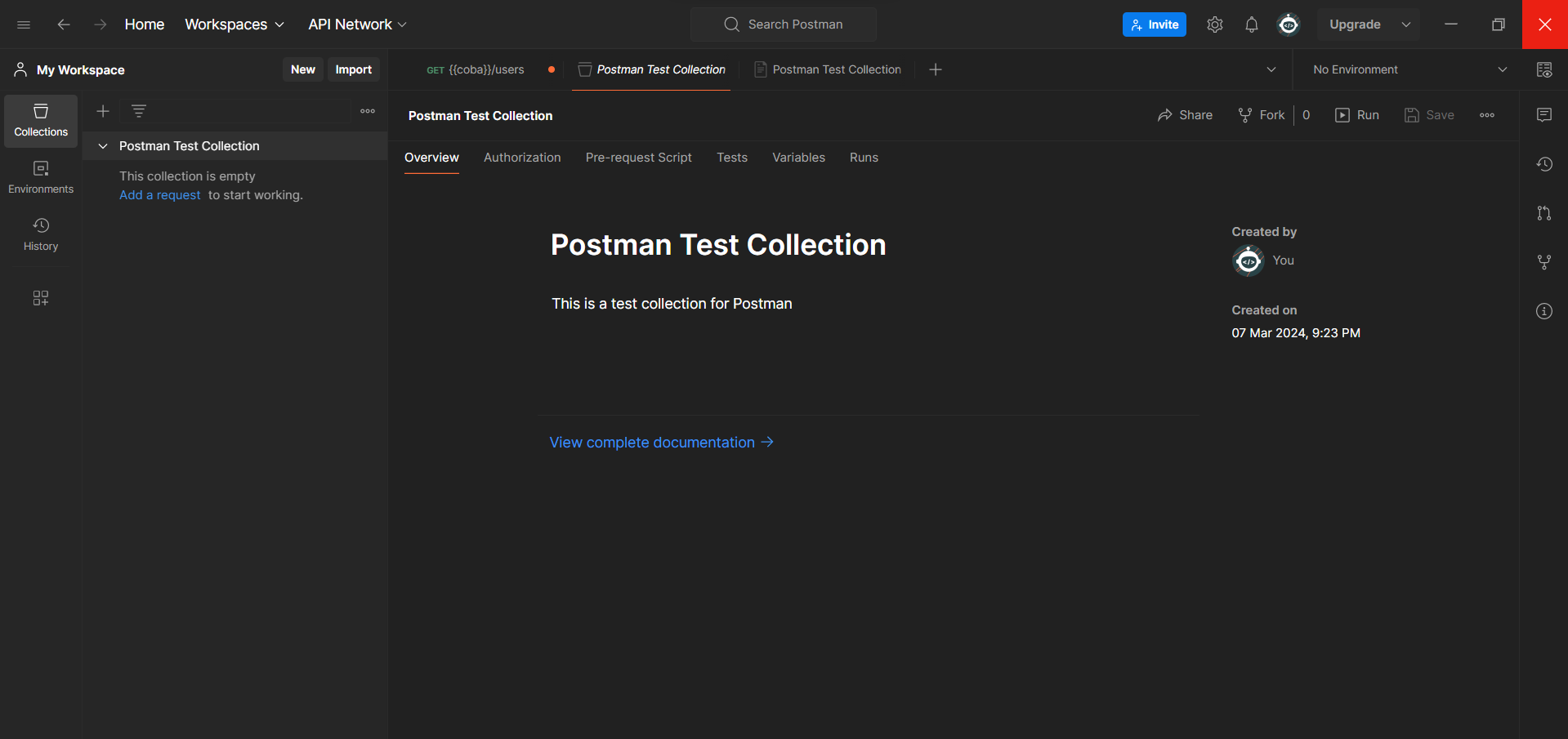


**Langkah 2)** Pilih Collection. Buat jendela collection harus muncul.

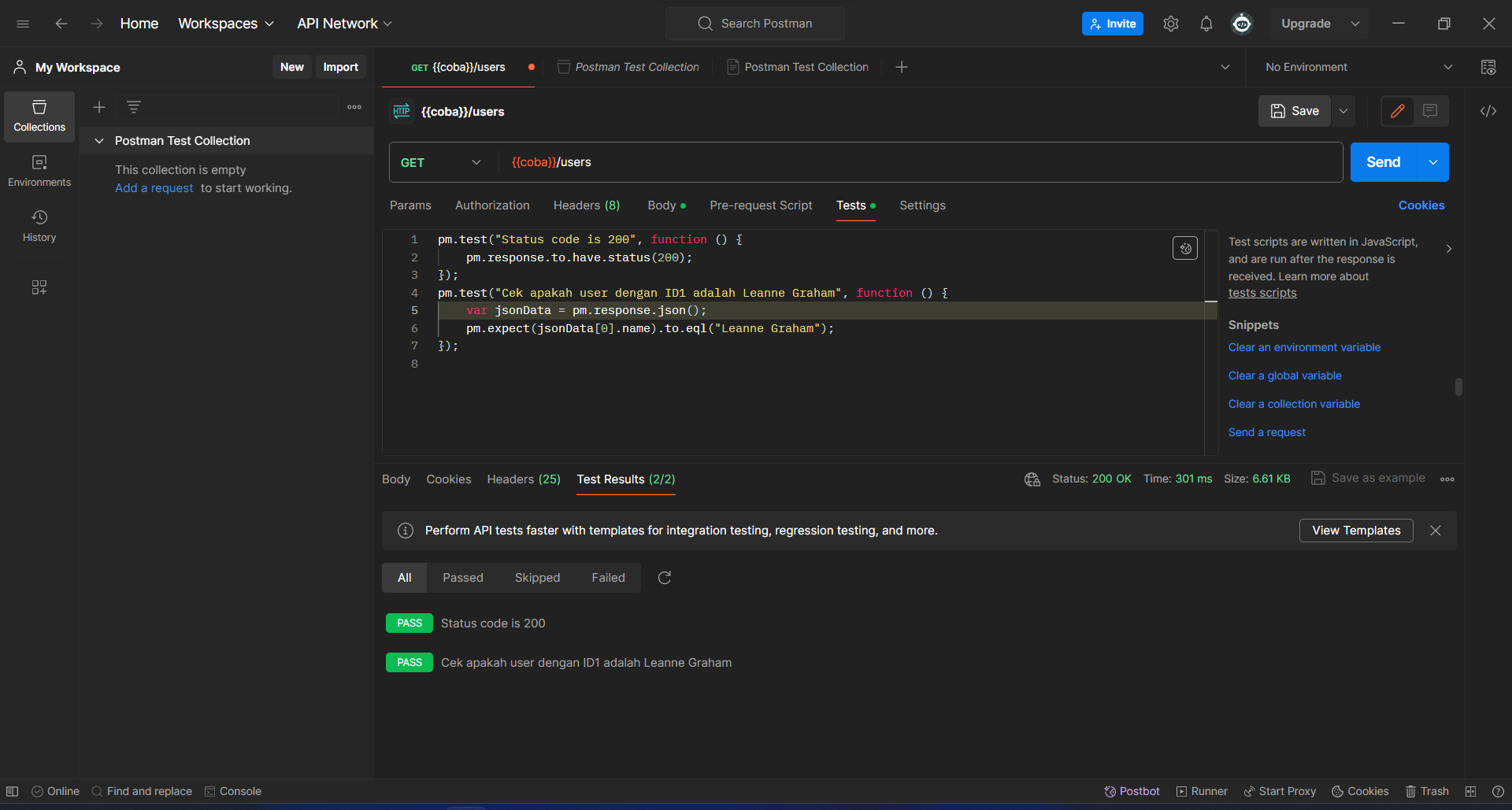




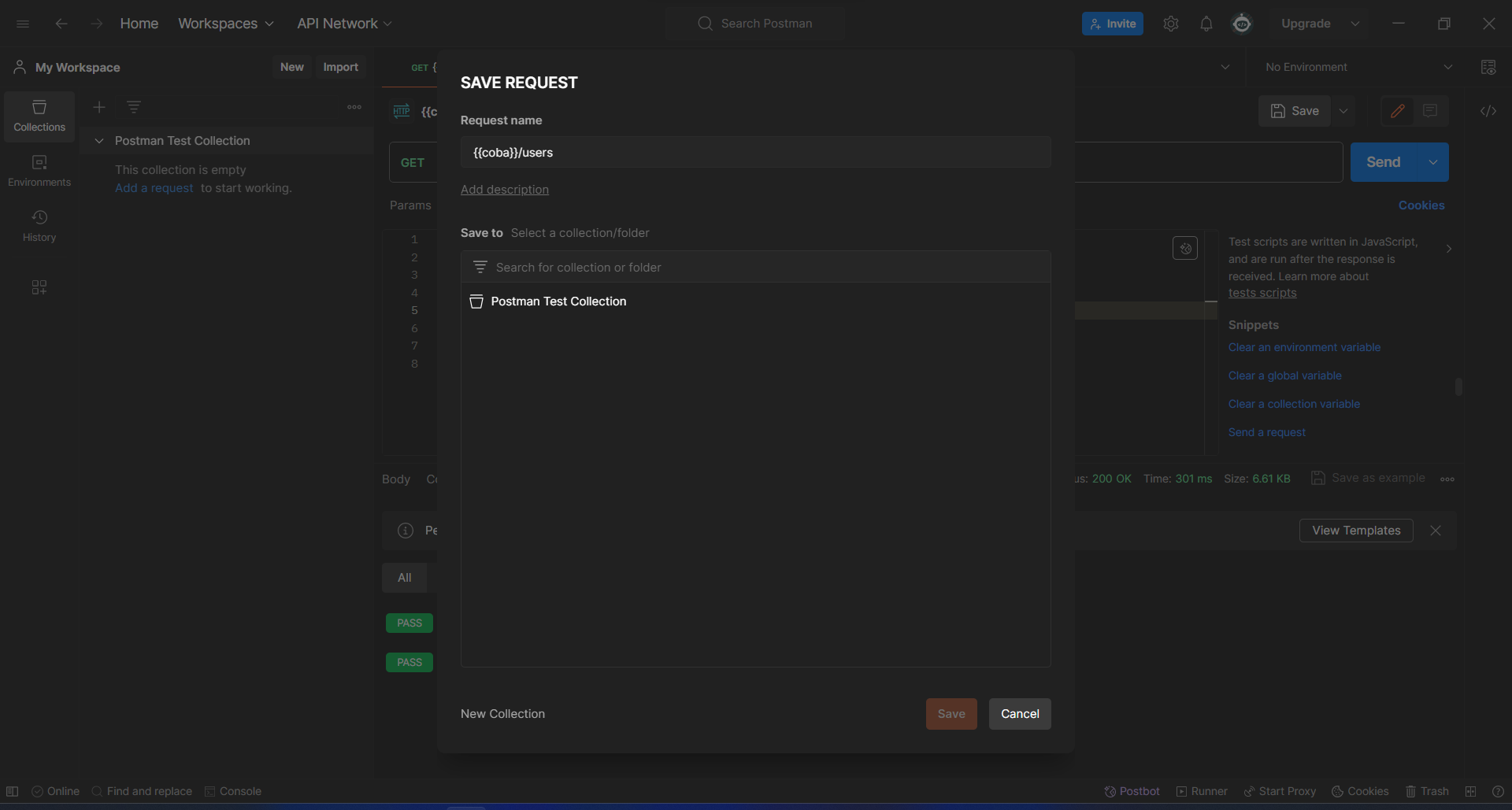
**Langkah 3)** Masukkan nama dan deskripsi collection yang diinginkan kemudian klik buat. Collection sekarang harus dibuat.

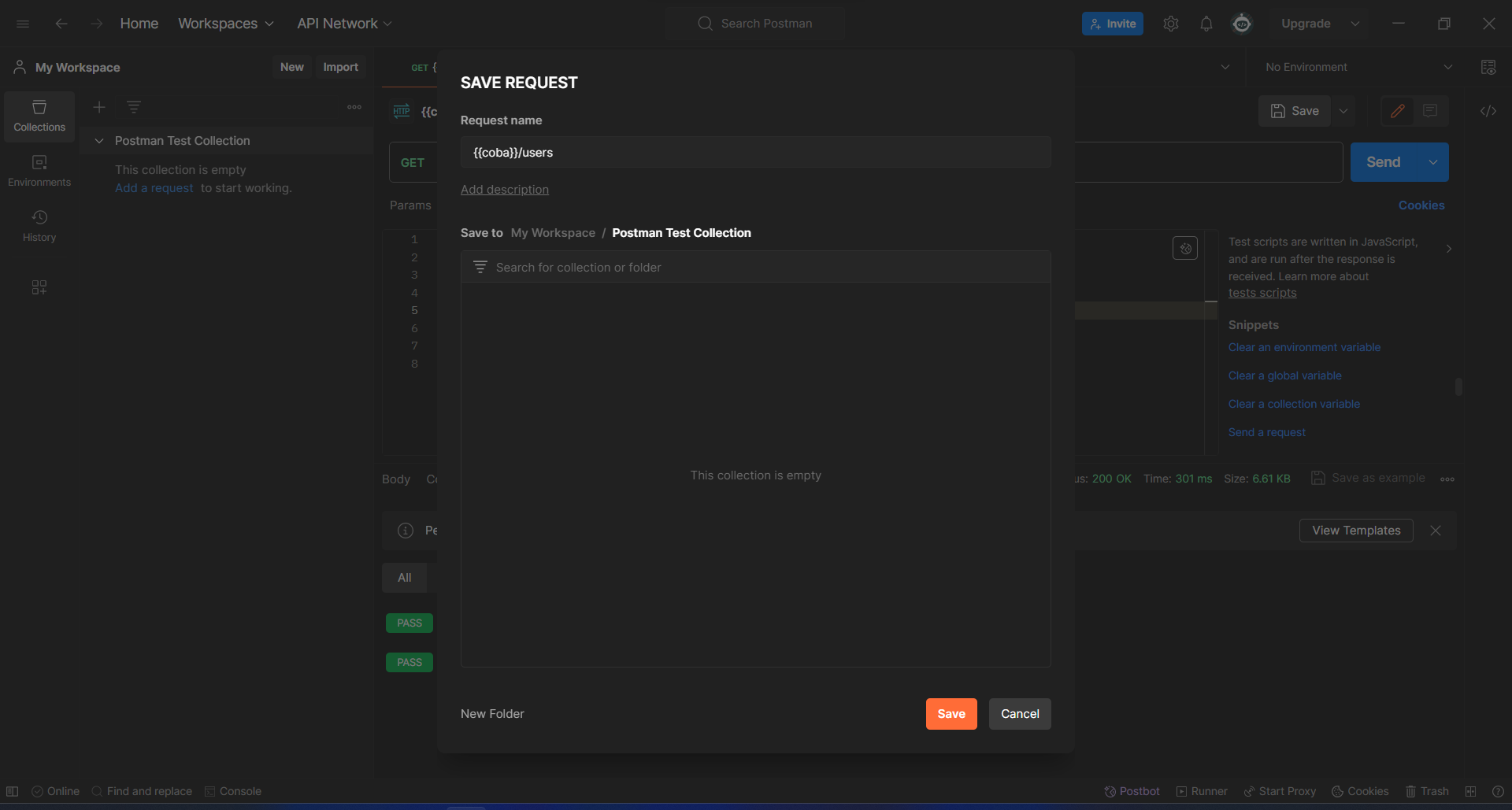


**Langkah 4)** Kembali ke Get request Sebelumnya. Klik **Save/Simpan**

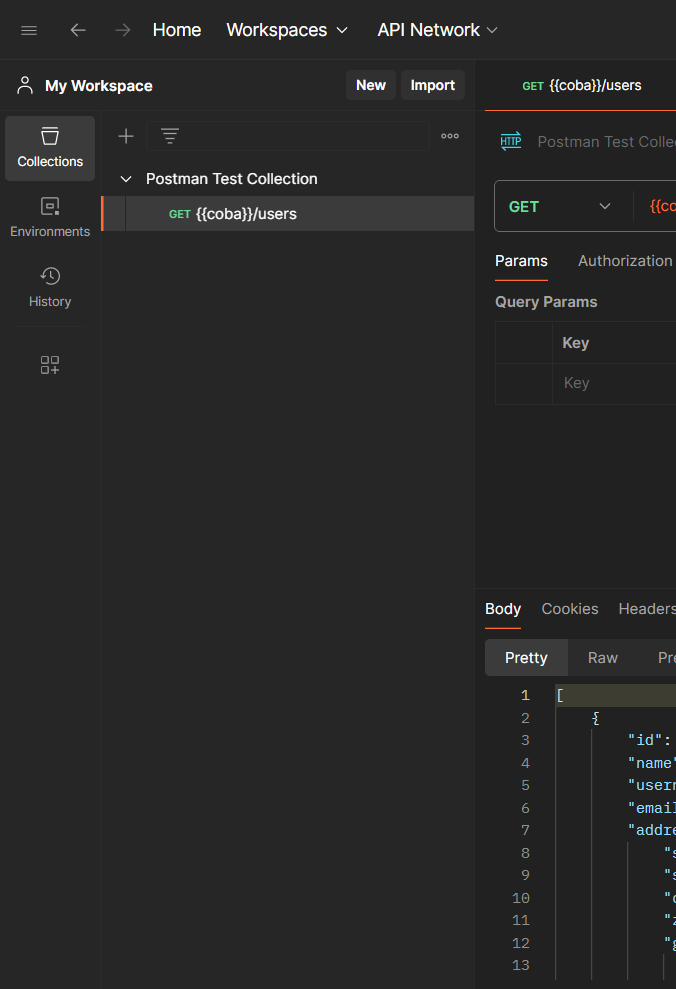


# Langkah 5)

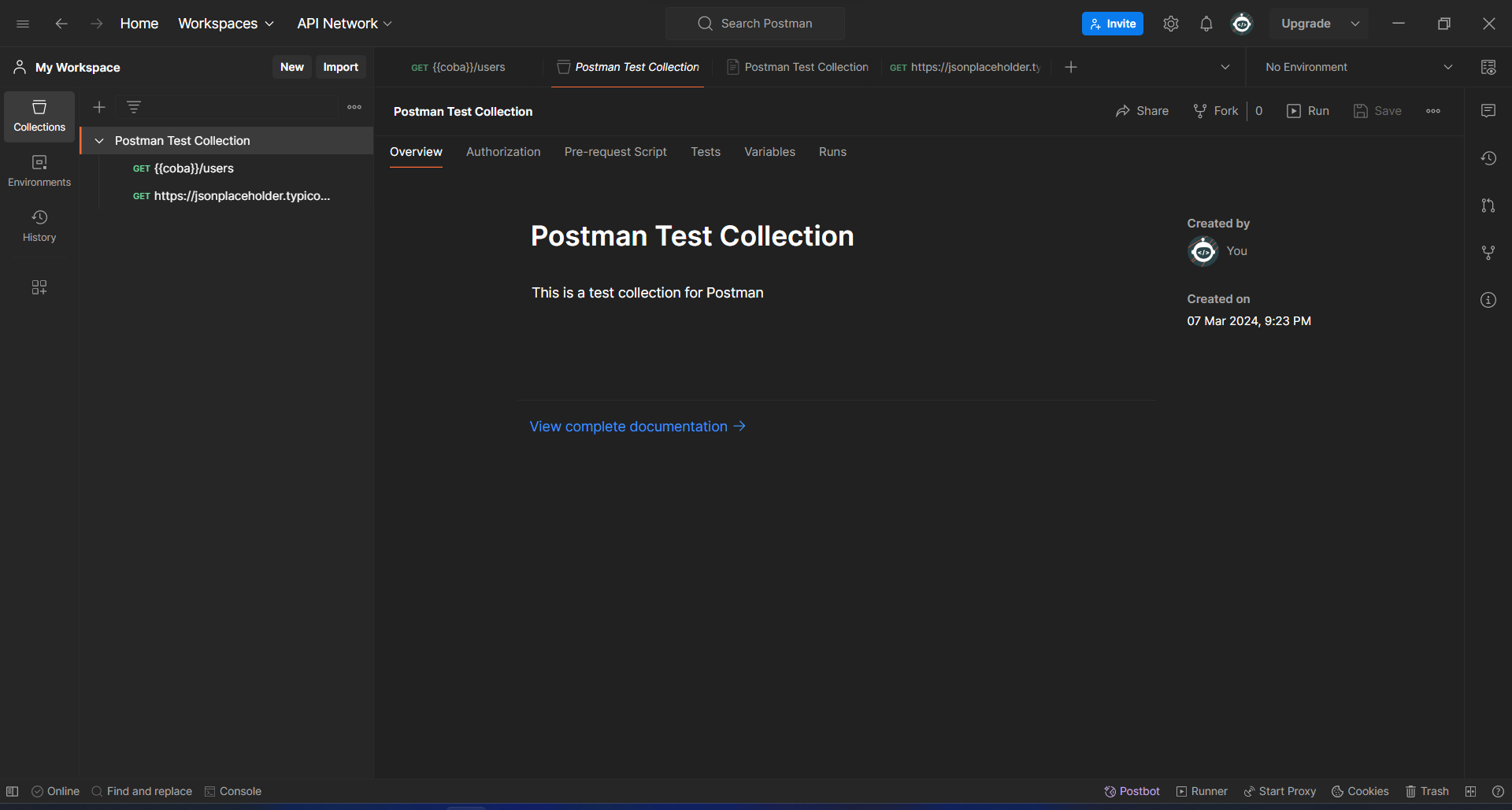
* Pilih Postman test Collection. 
* Klik Save to Postman test Collection.



**Langkah 6)** Postman Test Collection sekarang pasti sudah berisi satu permintaan.



**Langkah 7)** Ulangi langkah 4-5 untuk Request POST sebelumnya sehingga collection sekarang akan memiliki dua permintaan.



# Summary

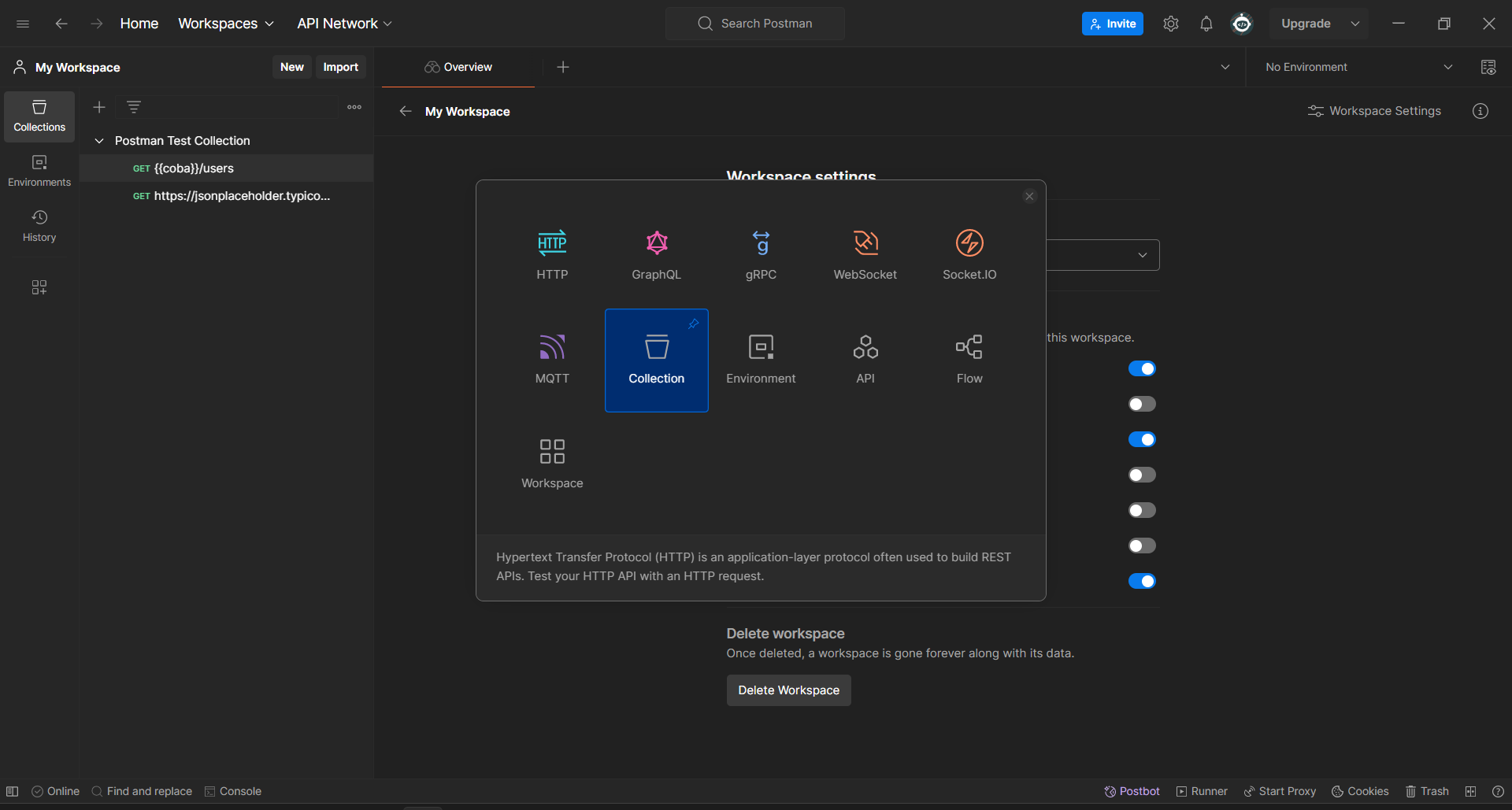
* Postman saat ini adalah salah satu alat paling populer yang digunakan dalam pengujian API
* Aksesibilitas, Penggunaan Collection, Kolaborasi, Integrasi Berkelanjutan, adalah beberapa fitur utama untuk dipelajari di Postman
* Disarankan Anda membuat akun di Postman, sehingga collection Anda tersedia secara online
* Anda dapat membuat parameter permintaan di Postman
* Anda dapat membuat Tes untuk memverifikasi permintaan Postman
* Collection dapat dijalankan menggunakan Newman atau Collection Runner

**Tugas:**

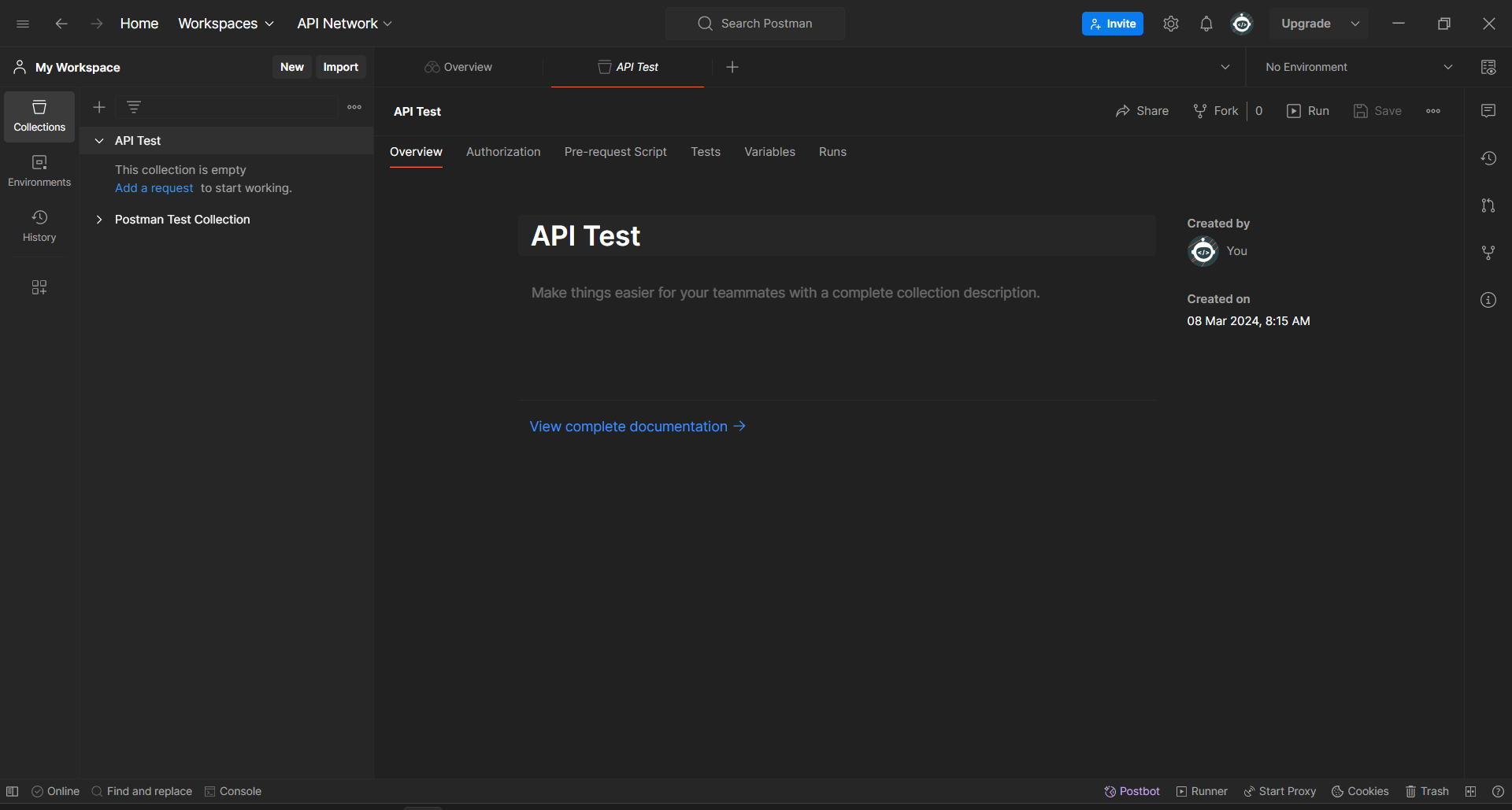
# Bagaimana cara menjalankan Collection menggunakan Collection Runner di Postman? (Praktikkan dan dokumentasikan)

Jawab:

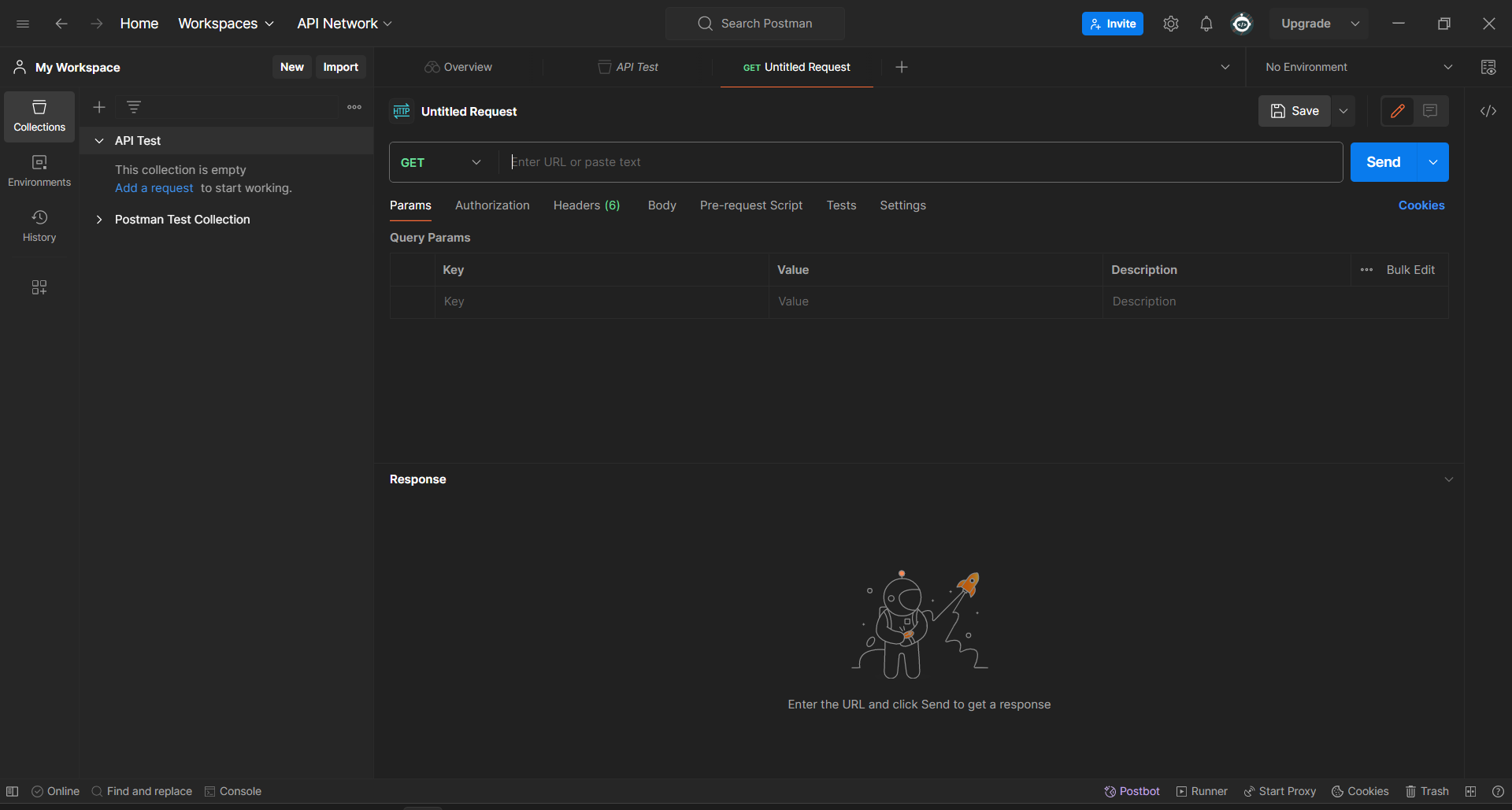
1. Buat Collection Baru



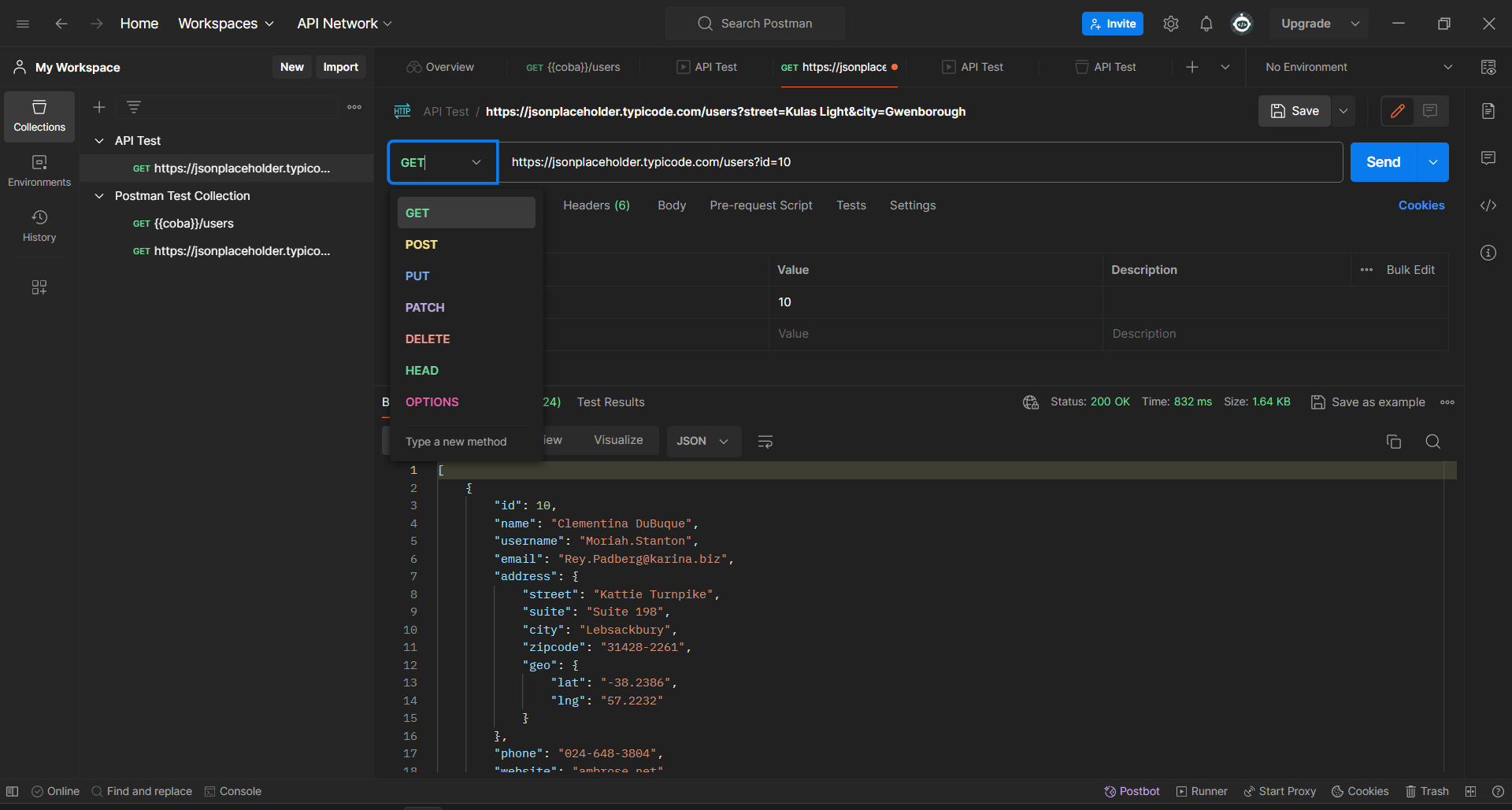
1. Ubah nama judul



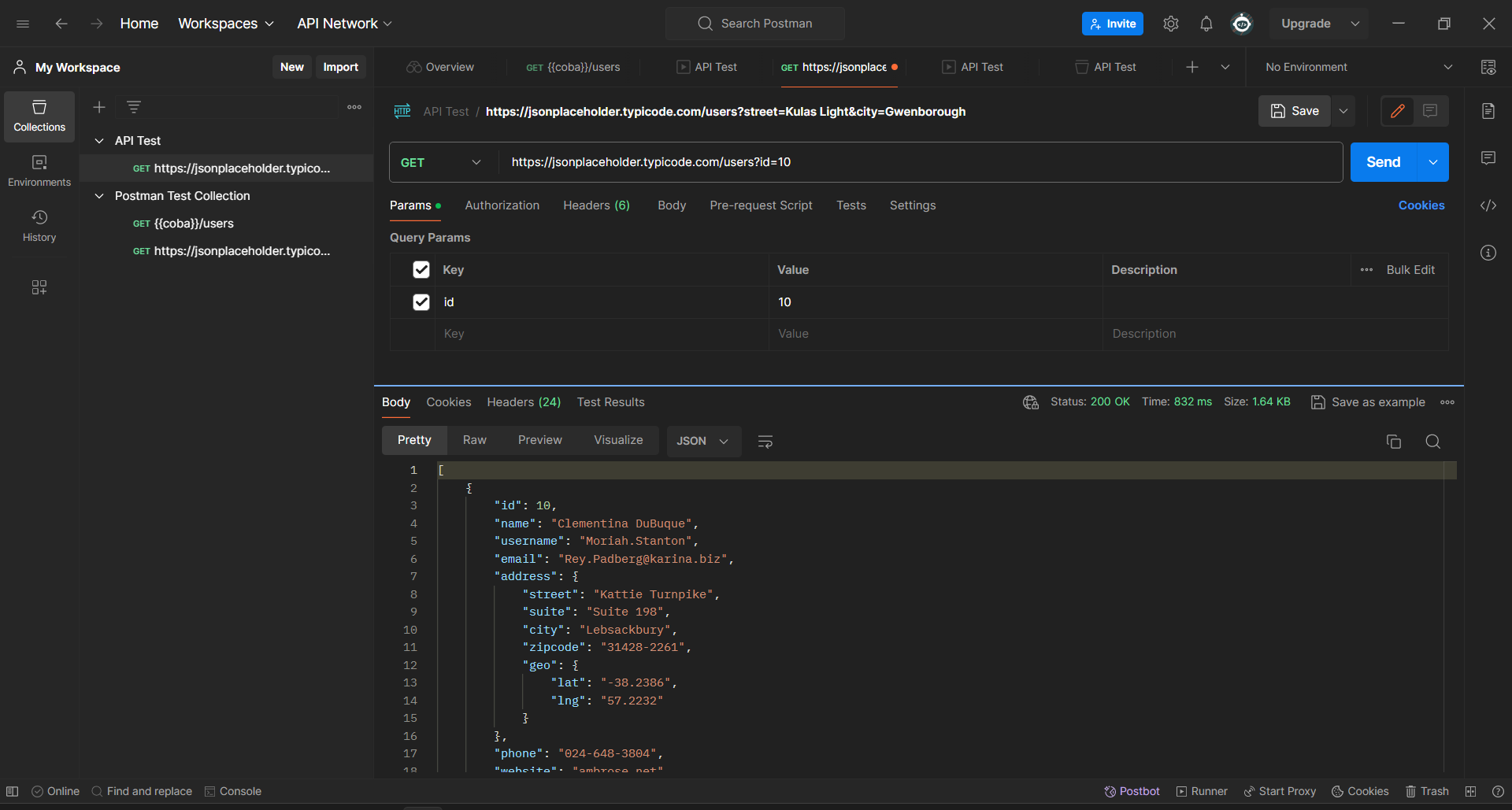
1. Klik ikon (+) dan masukkan URL



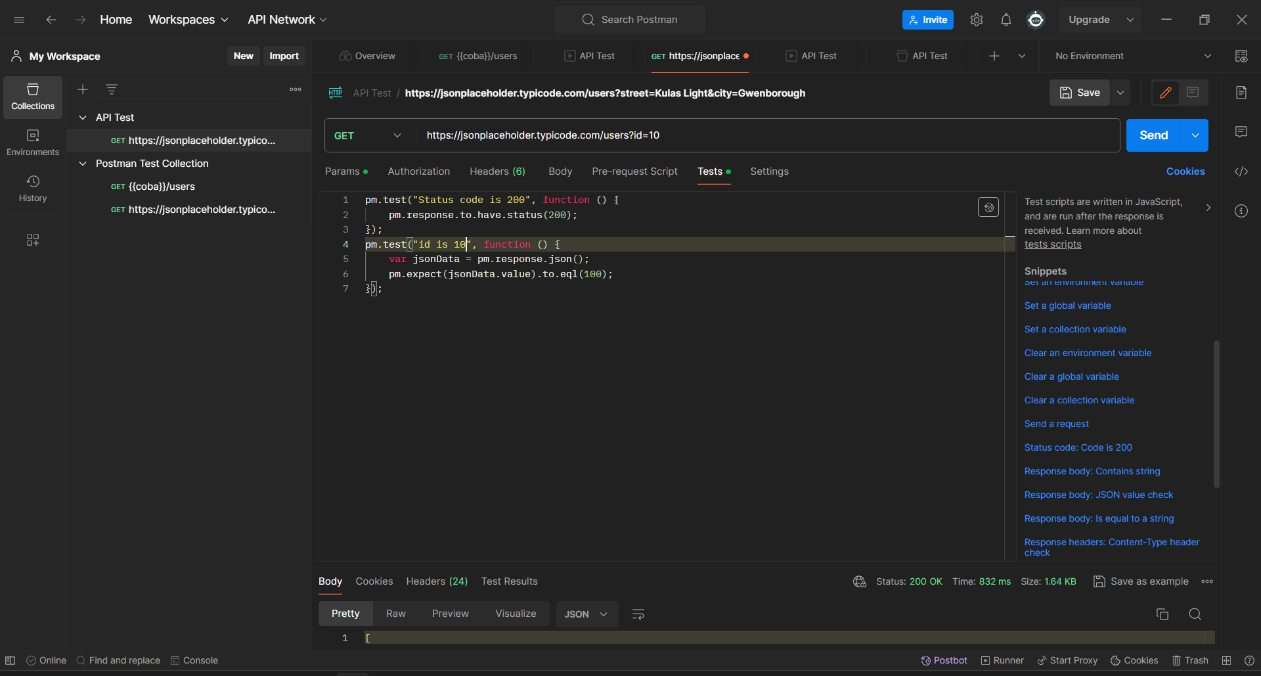
1. Pilih GET



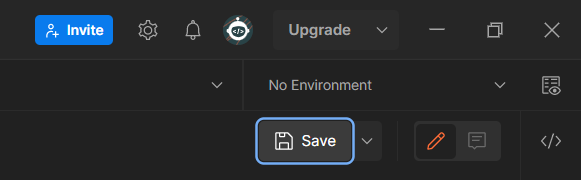
1. Tentukan variabel nya



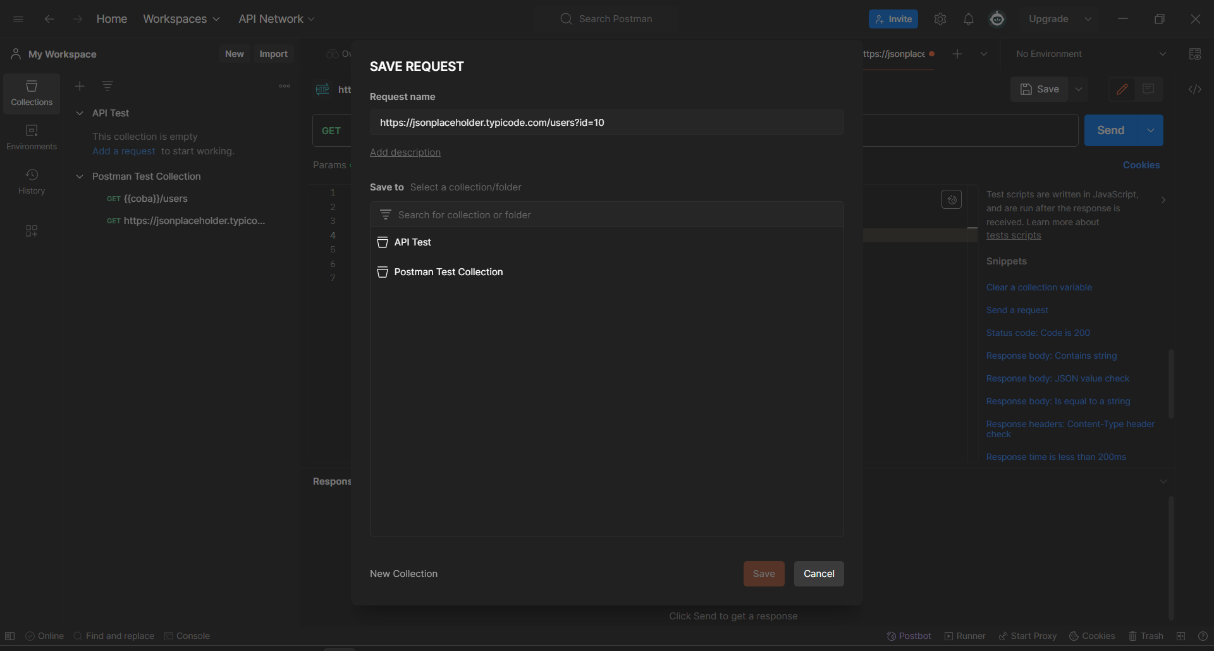
1. Konfigurasikan Tes



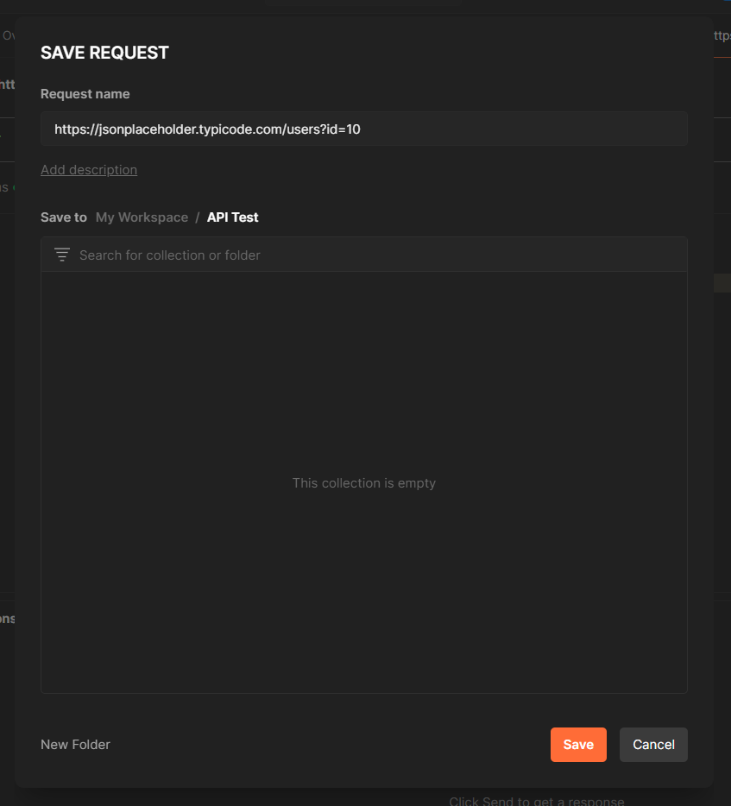
1. Kemudian Klik **Save**



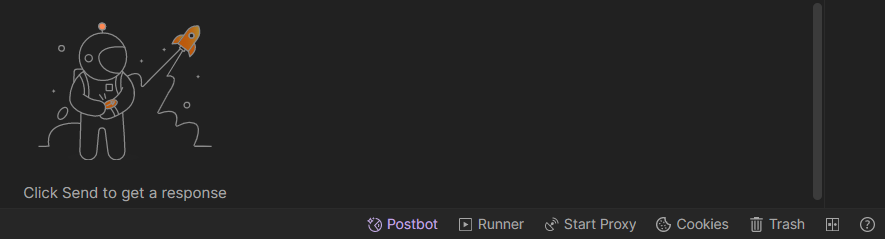
1. Pilih salah satu **Collection/Folder**



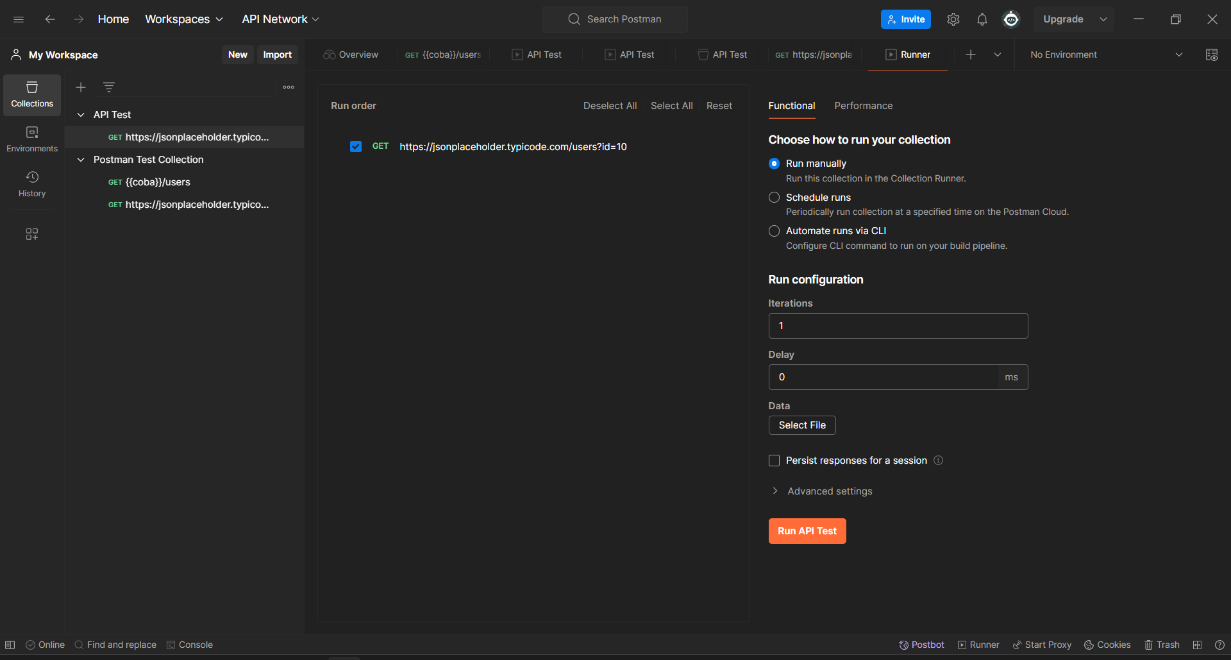
1. Klik **Save**



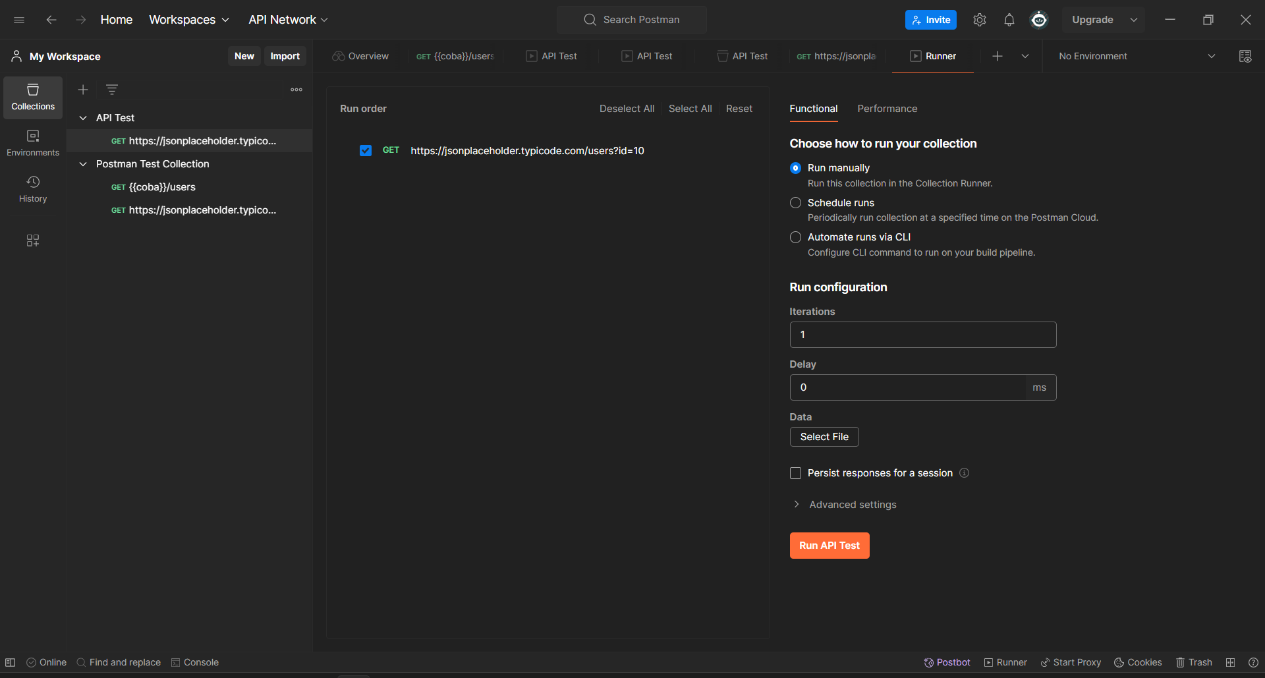
1. Untuk menggunakan **Collection Runner**, klik **Runner** di bagian kanan bawah



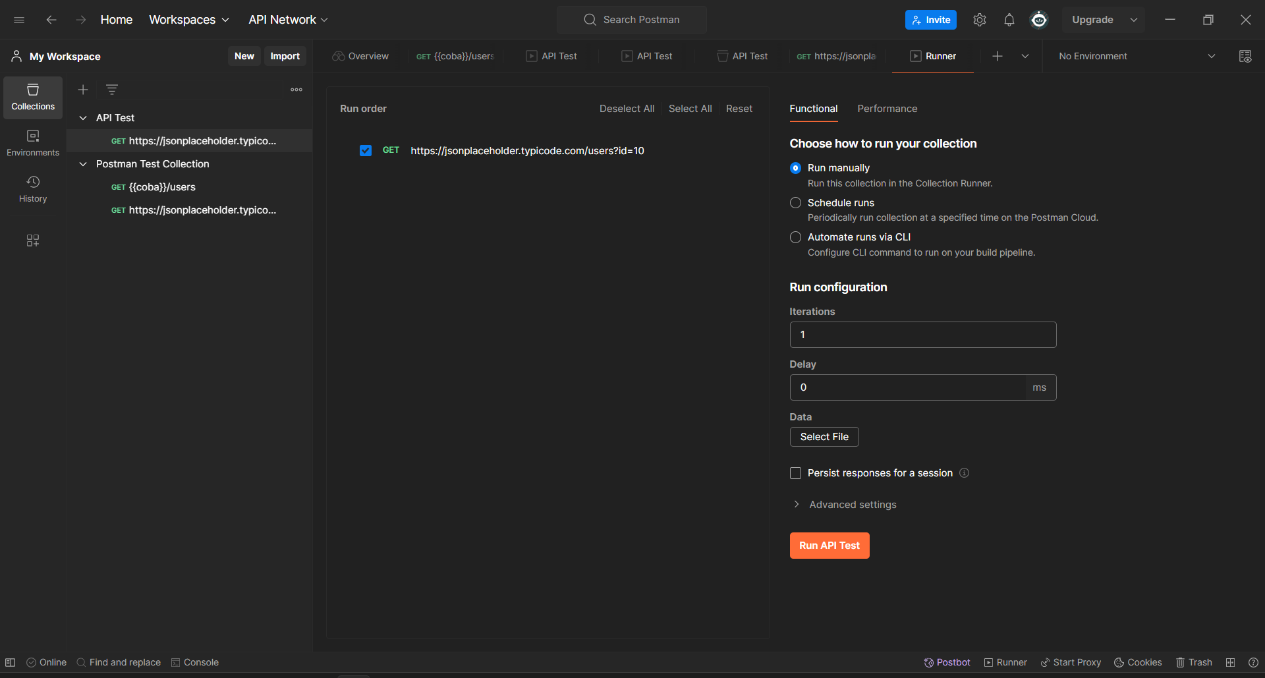
1. Pilih **Collection** kemudian seret dan jatuhkan File Request yang ada di **Collection** ke **Run order**



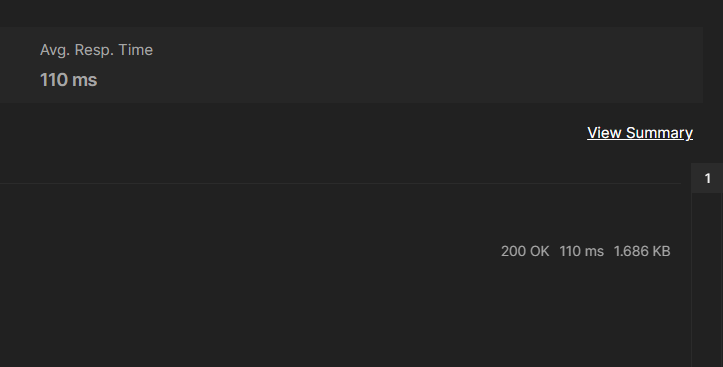
1. Tetapkan opsi seperti jumlah iterasi dan penundaanya atau pilih file data jika ada.



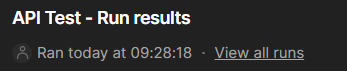
1. Klik **Run API Test**



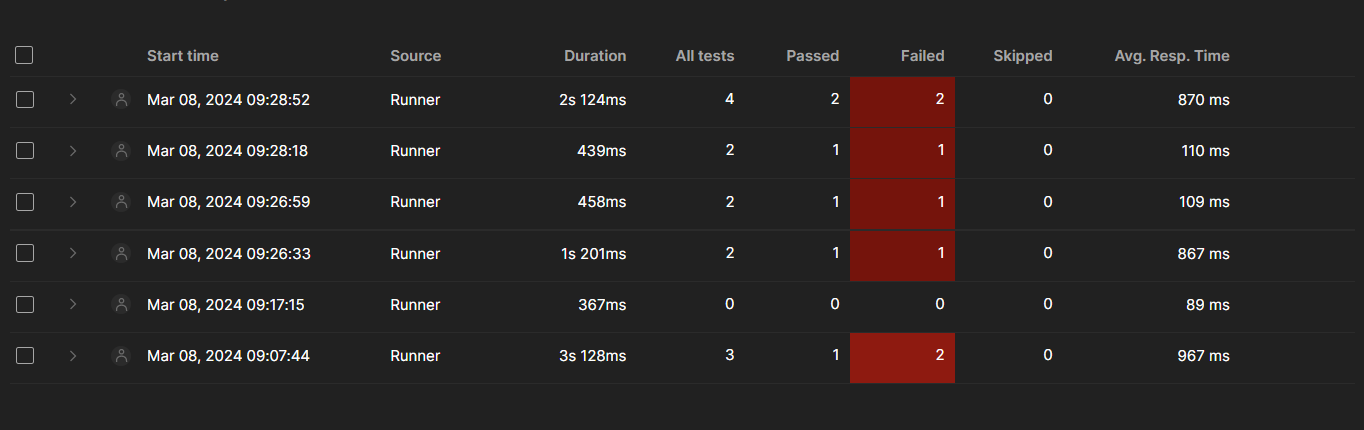
1. Lihat **Detail Respon** nya



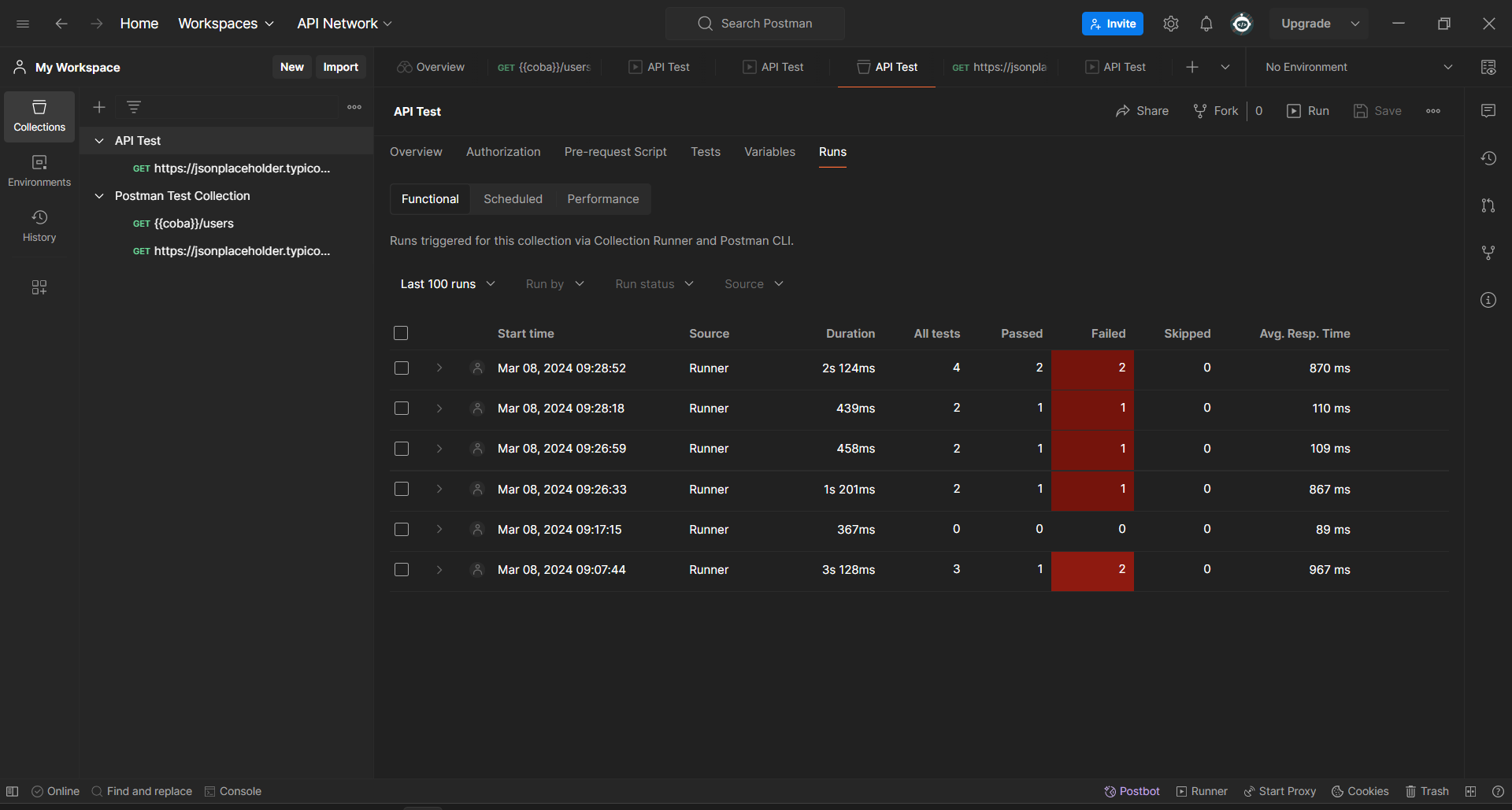
1. Klik **View All runs** untuk melihat tampilan yang pernah dijalankan sebelumnya



1. Berikut hasil yang sudah dijalankan



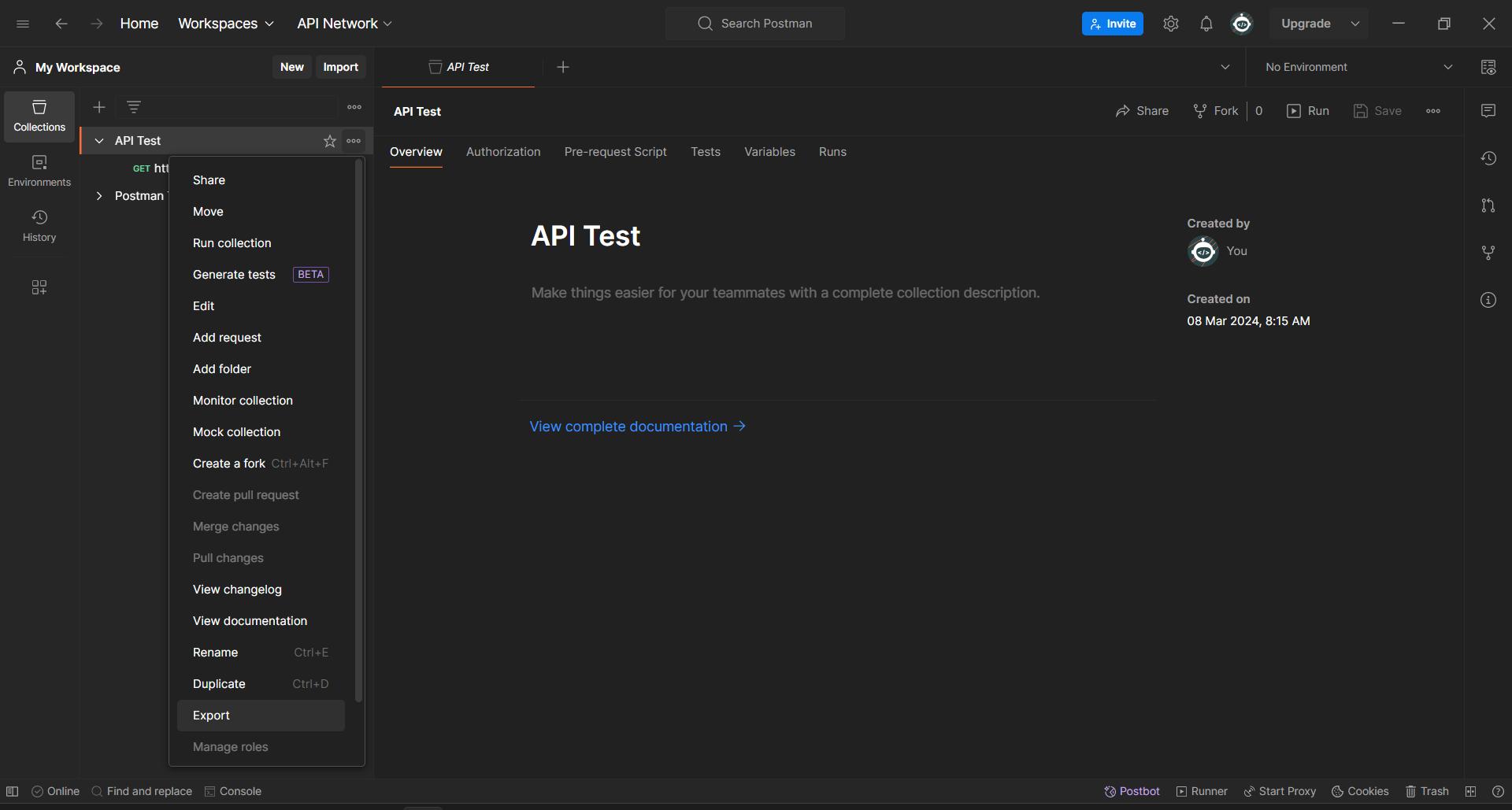
1. Ekspor hasil yang sudah dijalankan



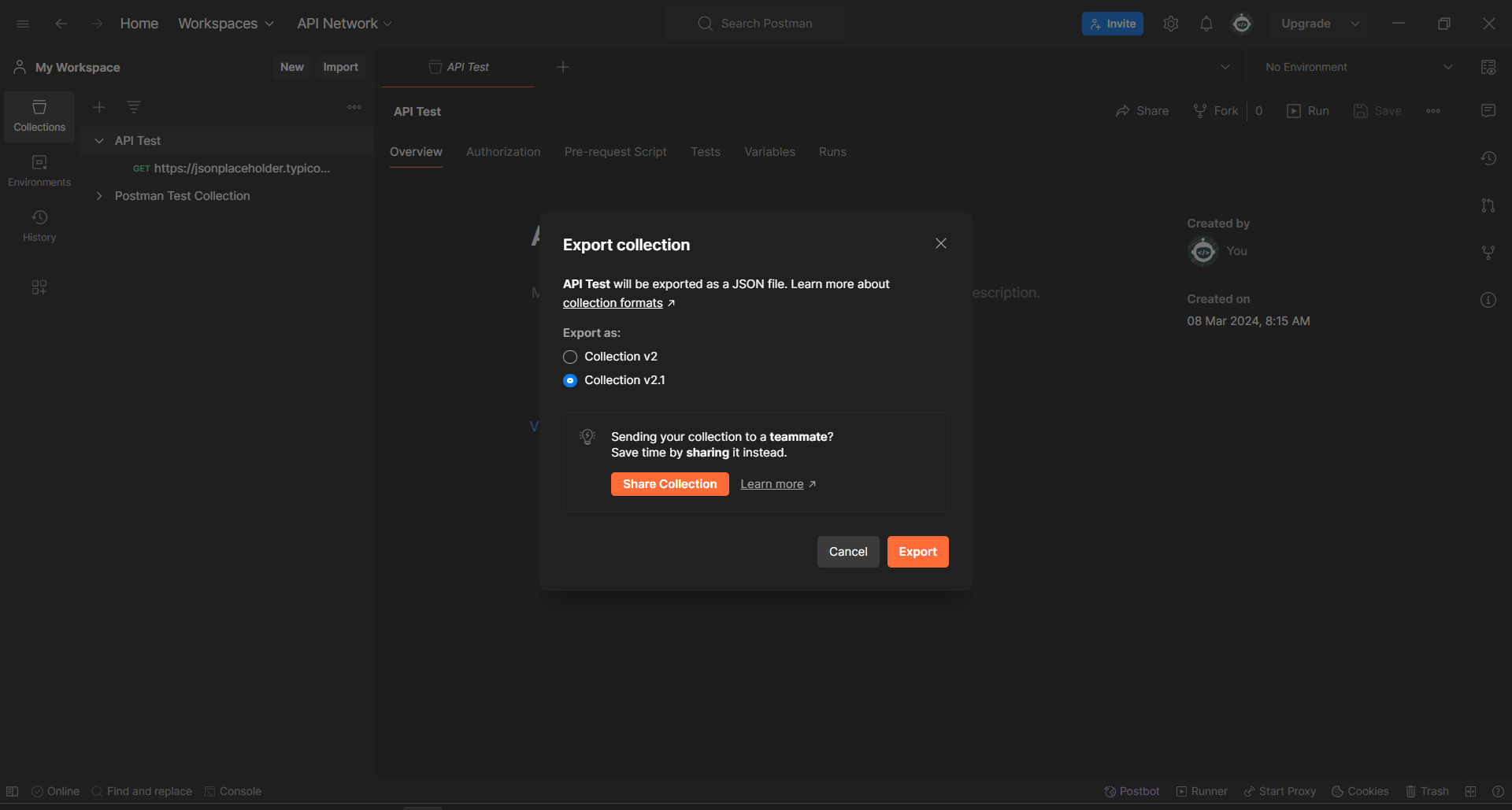
# Bagaimana cara menjalankan Collection menggunakan Newman di Postman? (Praktikkan dan dokumentasikan)

# Jawab:

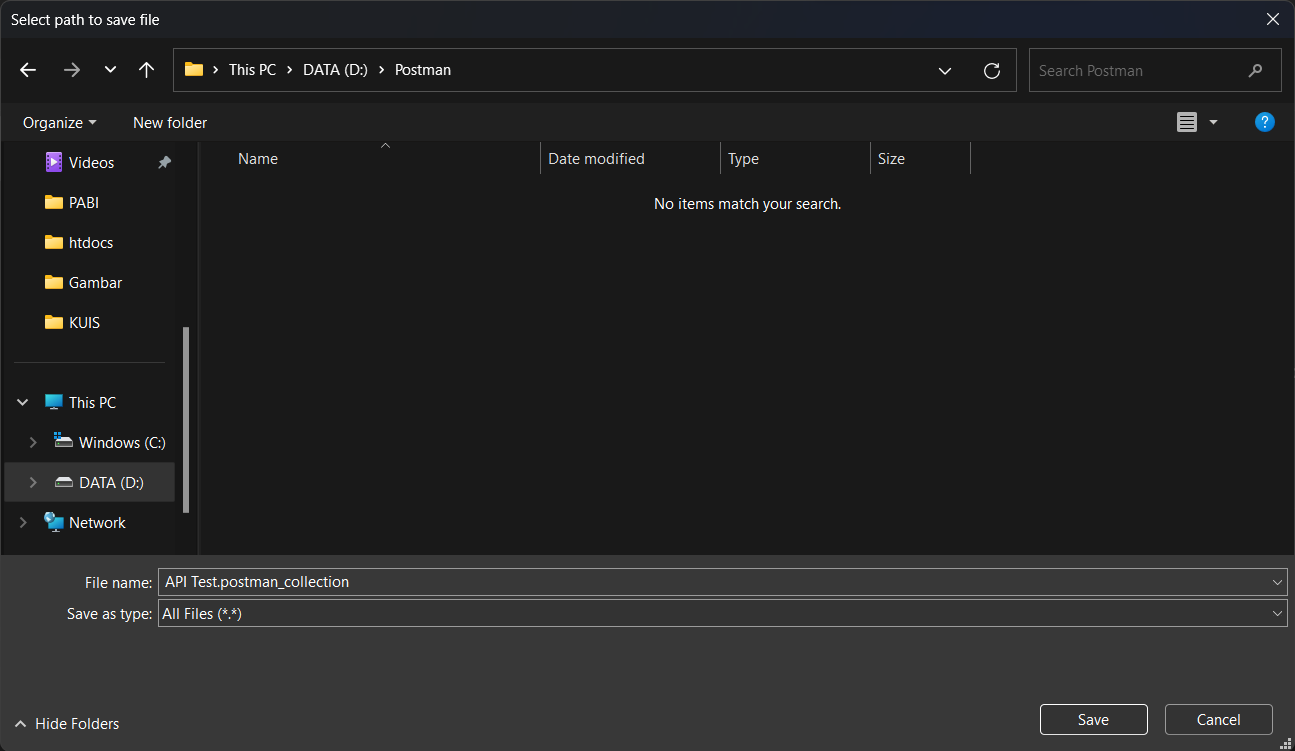
# Klik titik tiga disamping Folder Collection kemudian klik Export



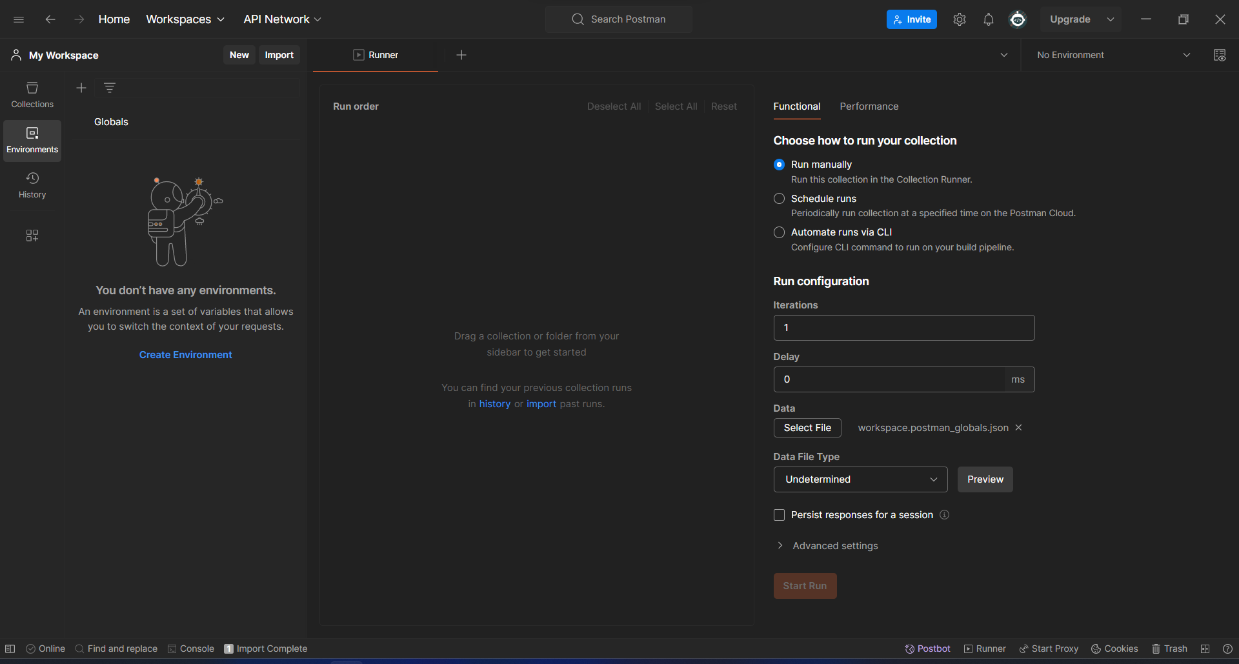
# Pilih opsi Collection v2.1 dan klik Export



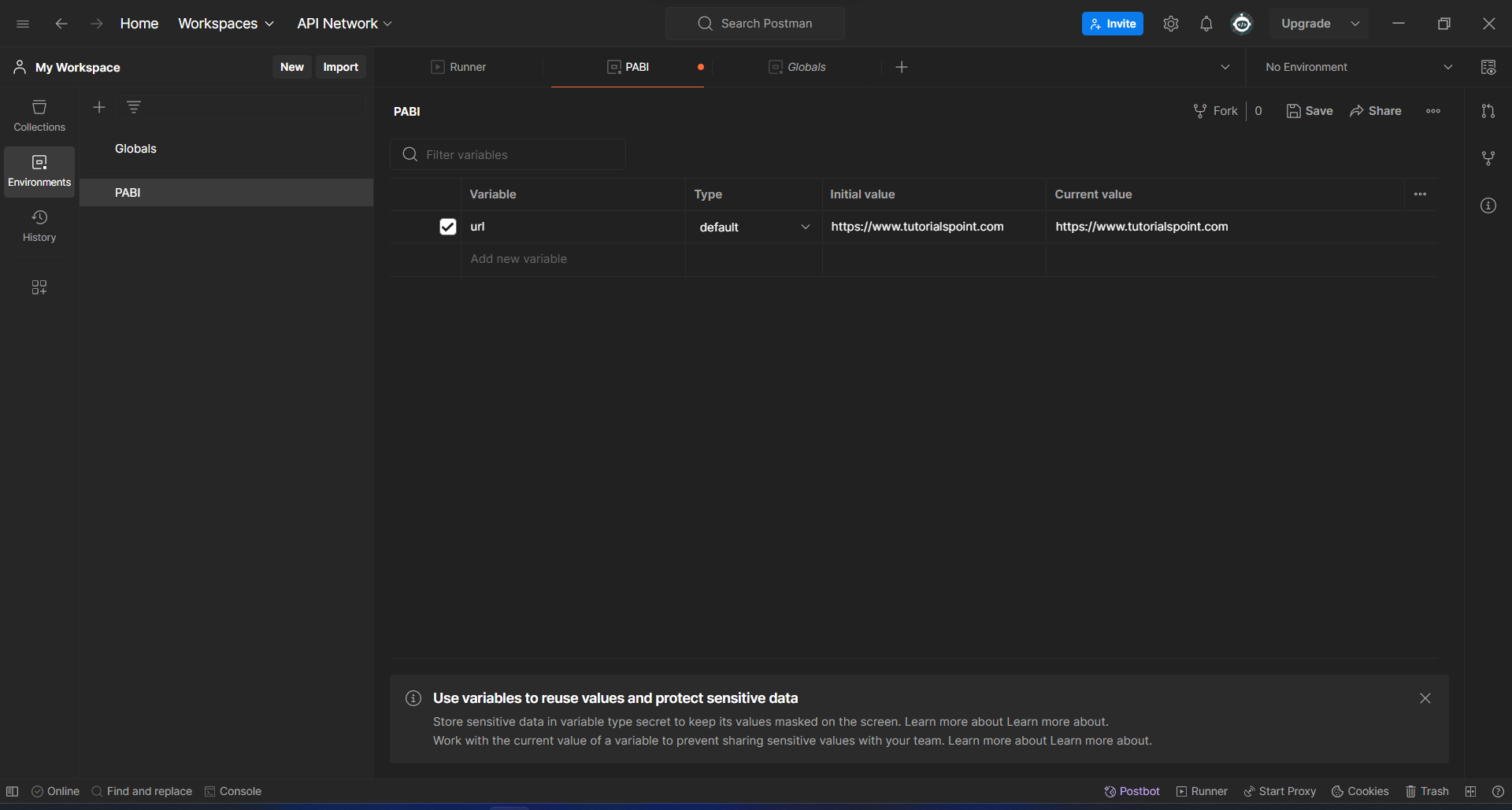
1. Pilih file yang ingin anda simpan dan klik **Save**



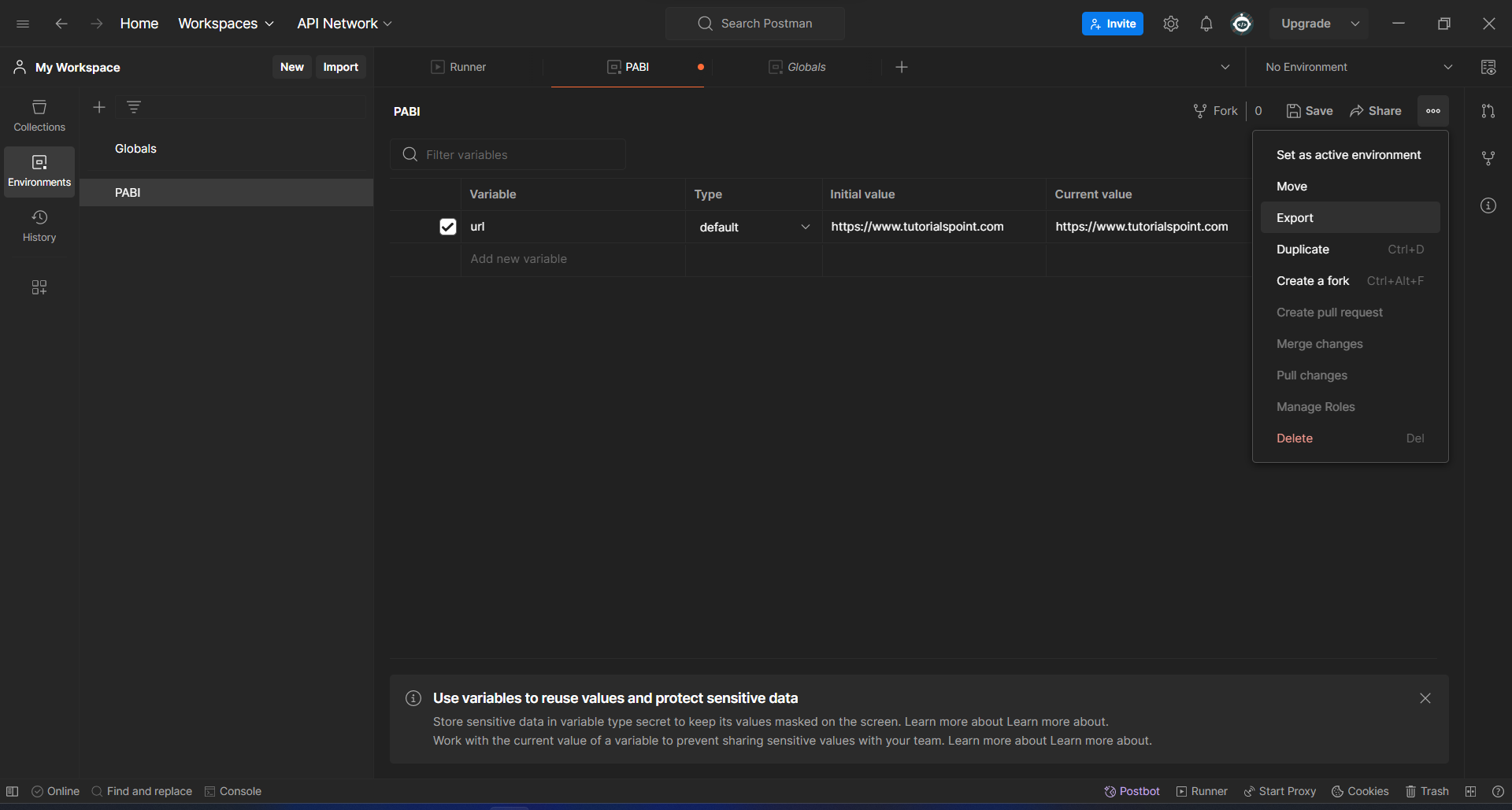
1. Klik tab **Environments** dan Buat Baru



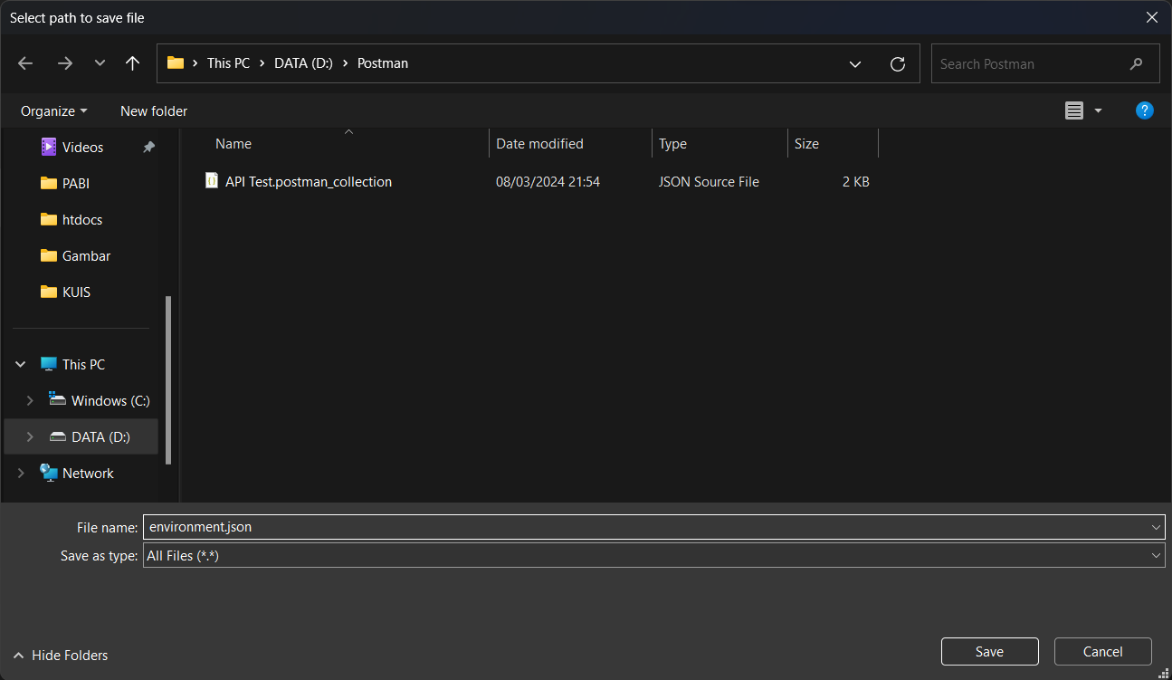
1. Isi Environment nya



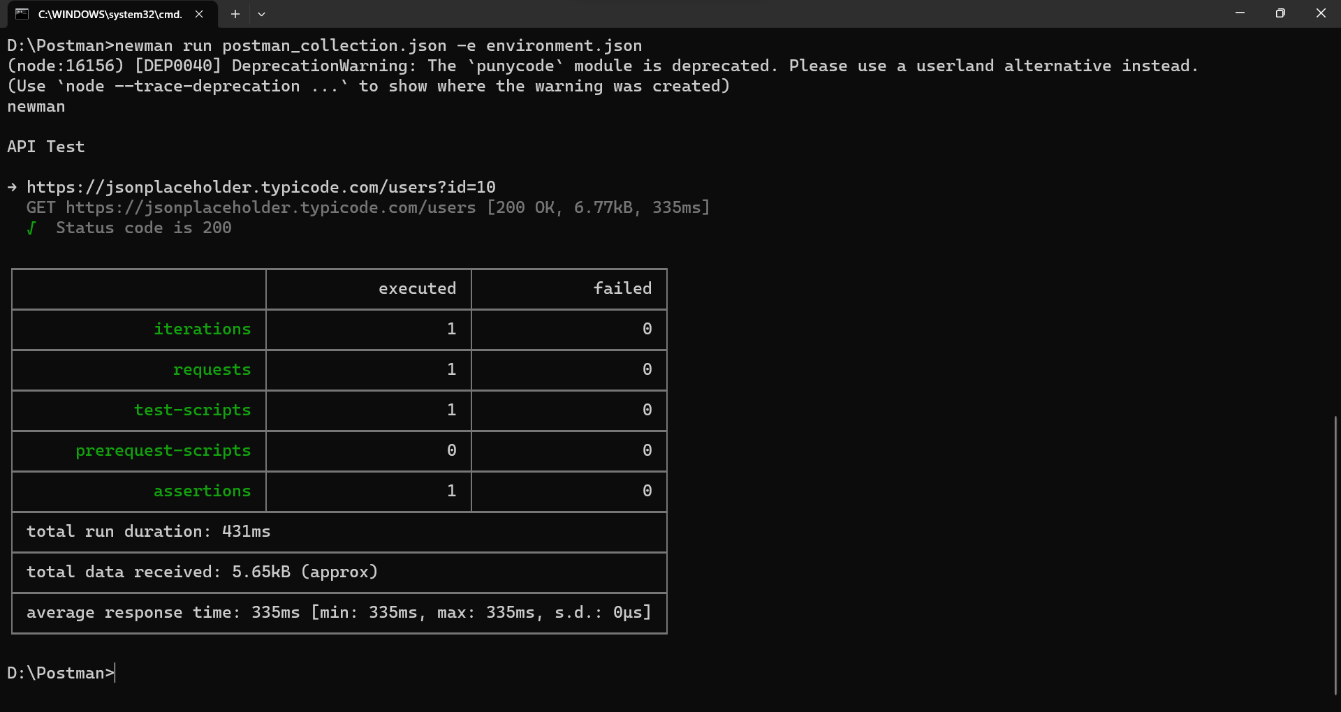
1. Klik ikon titik 3 dan pilih **Export**



1. Pilih lokasi penyimpanan dan klik **Save**



1. Ketik cmd di tempat lokasi menyimpan JSON dan ketik command di kotak ini dan berikut output yang diberikan

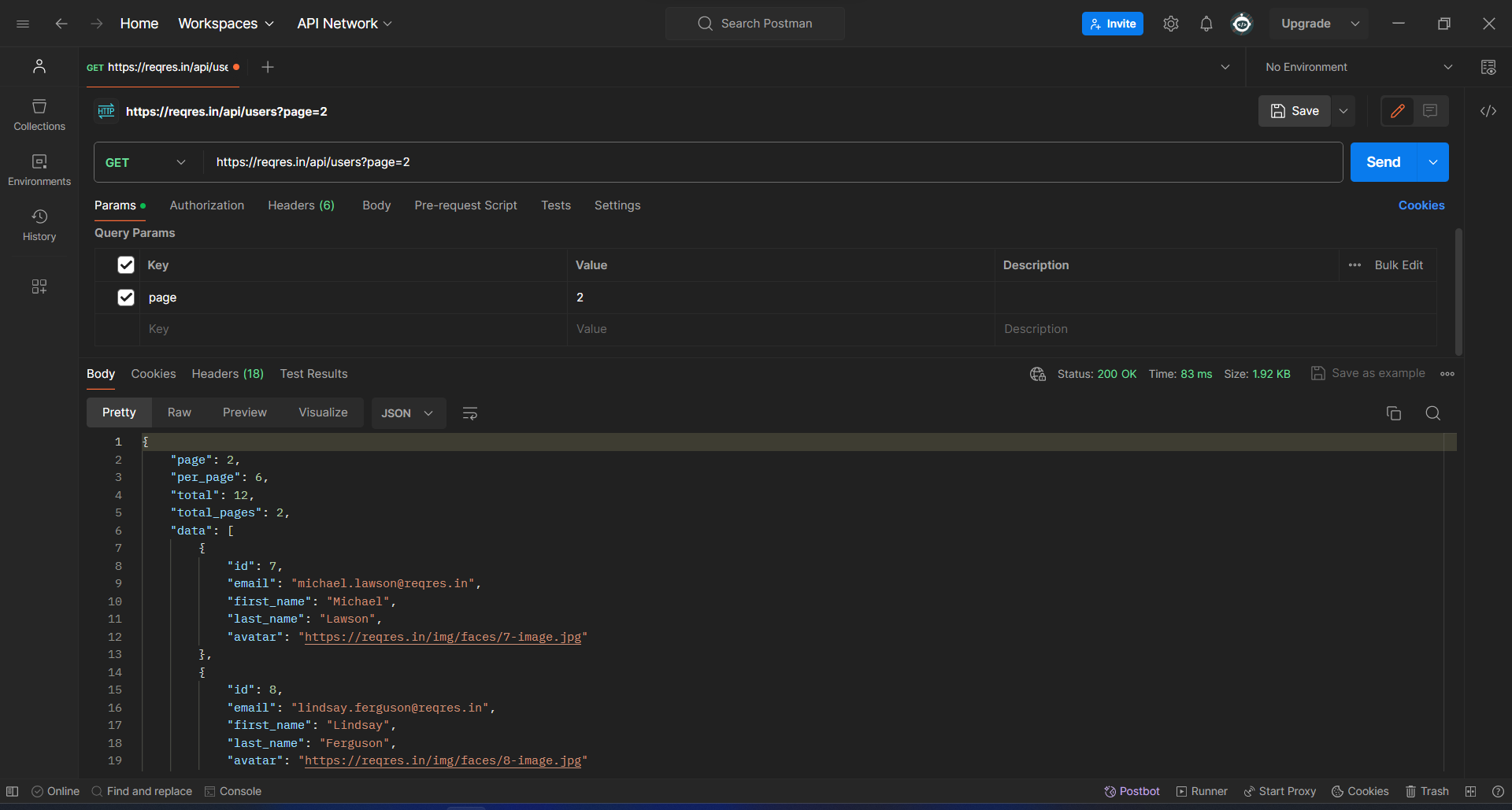


# Akseslah website <https://reqres.in/>untuk mengambil sample REST API. Dalam website ini ada sample-sample untuk melakukan API Test.

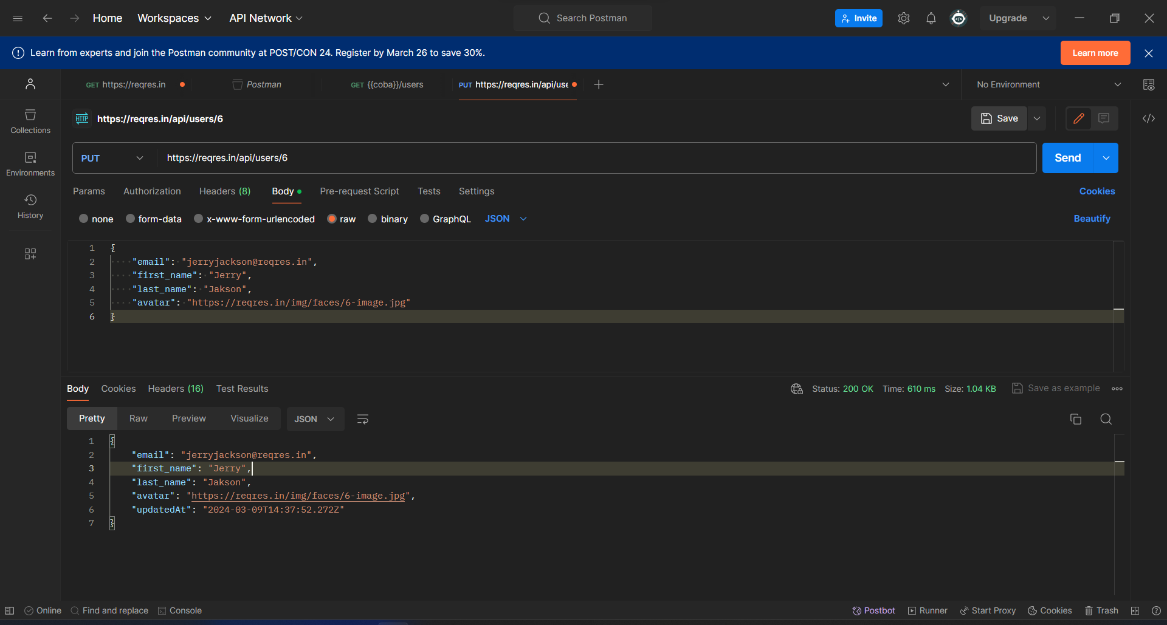
# Lalu coba melakukan perintah-perintah seperti : request get, put, post dan delete dan terapkan pada Postman kesayangan anda dengan memanfaatkan<https://reqres.in/>sebagai Rest API endpoint. (manfaatkanlah internet anda untuk eksplorasi)

**Jawab:**

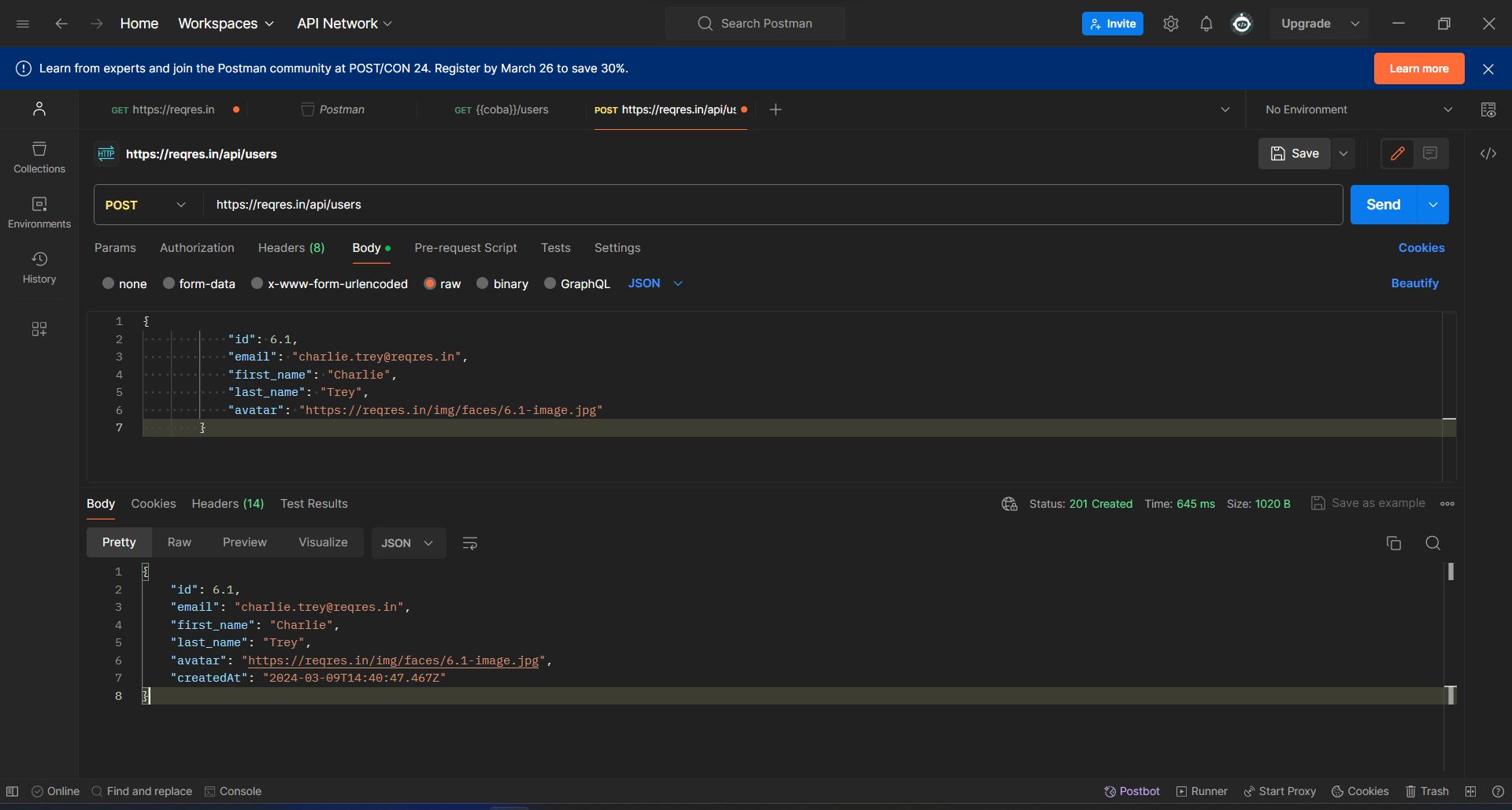
* **GET**

****

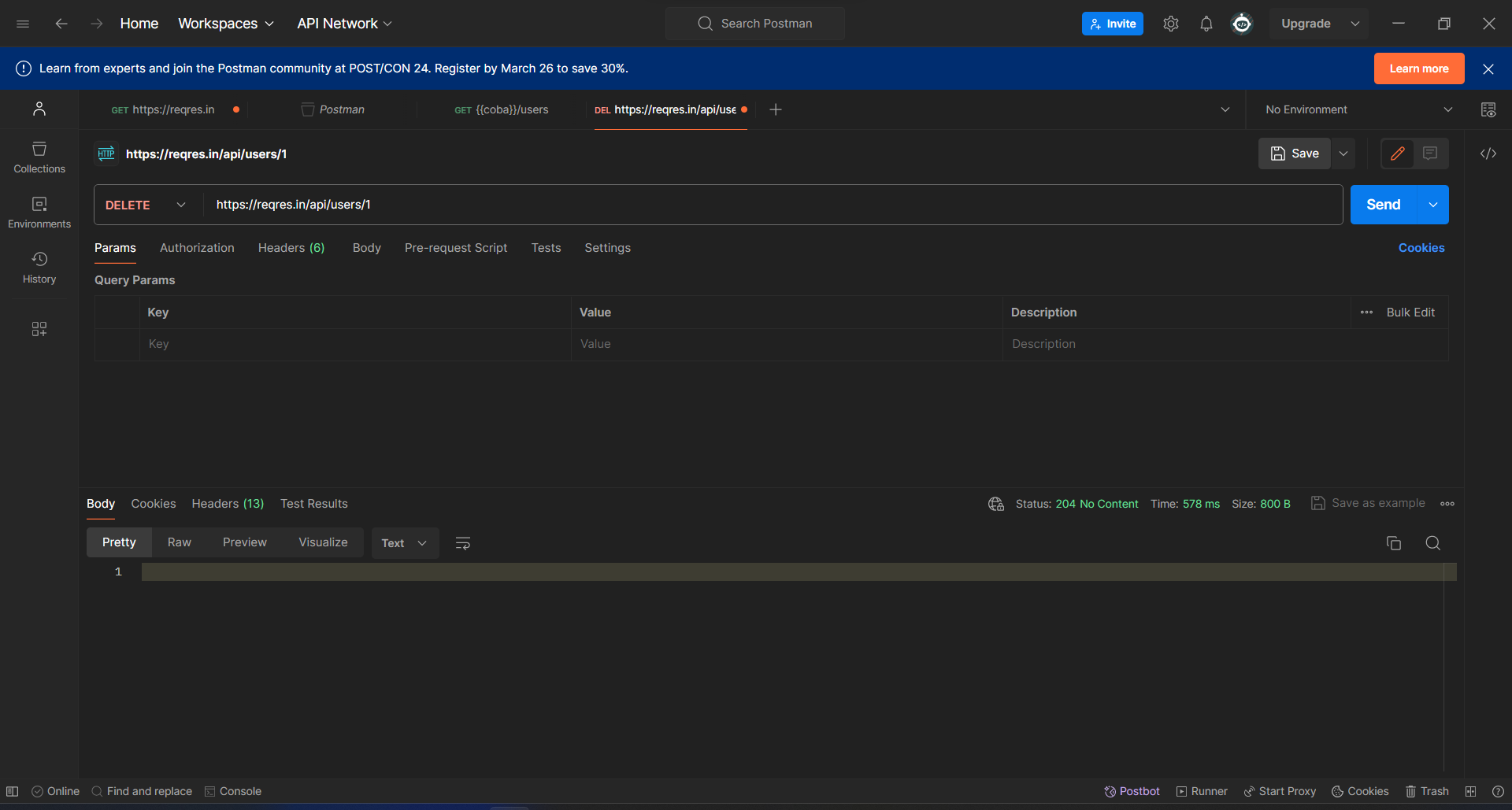
* **PUT**

****

* **POST**

****

* **DELETE**

****

# Dokumentasikanlah Langkah demi Langkah praktikum dan tugas anda dalam laporan praktikum dengan nama file:

# PrakW10S02\_KodeKelas\_NIM Lengkap.pdf